

D78767GC10  
Edition 1.0  
March 2013  
D81112



# **Oracle Solaris 11 Performance Management**

## **Activity Guide**

**Copyright © 2013, Oracle and/or its affiliates. All rights reserved.**

#### **Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

#### **Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

##### **U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

#### **Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

#### **Author**

David Giroux

#### **Technical Contributors and Reviewers**

Mike Carew, Benoit Chaffanjon, Glynn Foster, John Hathaway, Dominic Kay, Steve Kirby, Rosemary Martinak, Kristi McNeill, Chad Mynhier, Christian Wolbert

**This book was published using: Oracle Tutor**

## Table of Contents

<b>Practices for Lesson 1: Course Introduction .....</b>	<b>1-1</b>
Practices for Lesson 1.....	1-2
<b>Practices for Lesson 2: Introducing Performance Management .....</b>	<b>2-1</b>
Practices for Lesson 2: Overview.....	2-2
Practice 2-1: Assessing System Configuration .....	2-3
Practice 2-2: Using the kstat Utility.....	2-5
Practice 2-3: Using the truss Utility .....	2-7
Solution 2-1: Assessing System Configuration .....	2-8
Solution 2-2: Using the kstat Utility.....	2-12
Solution 2-3: Using the truss Utility .....	2-15
<b>Practices for Lesson 3: kstat Monitoring Tools .....</b>	<b>3-1</b>
Practices for Lesson 3: Overview.....	3-2
Practice 3-1: Using the vmstat, mpstat, and sar Commands .....	3-3
Practice 3-2: Using the iostat Command .....	3-5
Practice 3-3: Using the kstat Utilities.....	3-6
Solution 3-1: Using the vmstat, mpstat, and sar Commands .....	3-7
Solution 3-2: Using the iostat Command .....	3-11
Solution 3-3: Using the kstat Utilities.....	3-13
<b>Practices for Lesson 4: The procfs Monitoring Tools.....</b>	<b>4-1</b>
Practices for Lesson 4: Overview.....	4-2
Practice 4-1: Using the ps and prstat Commands.....	4-3
Practice 4-2: Using the collect Utility to Gather Information on a Running Process (Optional).....	4-6
Solution 4-1: Using the ps and prstat Commands.....	4-8
Solution 4-2: Using the collect Utility to Gather Information on a Running Process (Optional).....	4-15
<b>Practices for Lesson 5: Introduction to DTrace .....</b>	<b>5-1</b>
Practices for Lesson 5: Overview.....	5-2
Practice 5-1: Using the DTrace One-Liners .....	5-3
Practice 5-2: Using the DTrace Toolkit.....	5-5
Solution 5-1: Using the DTrace One-Liners .....	5-7
Solution 5-2: Using the DTrace Toolkit.....	5-12
<b>Practices for Lesson 6: Other Significant Tools .....</b>	<b>6-1</b>
Practices for Lesson 6: Overview.....	6-2
Practice 6-1: Managing System swap Devices .....	6-3
Practice 6-2: Using cpustat to View the cpc Counters.....	6-4
Practice 6-3: Using mdb to View Kernel Parameters.....	6-5
Practice 6-4: Using Oracle Solaris Studio dbx to Test a Common Application (Optional).....	6-8
Practice 6-5: Using the GUDS Script.....	6-10
Solution 6-1: Managing System swap Devices .....	6-15
Solution 6-2: Using cpustat to View the cpc Counters.....	6-17
Solution 6-3: Using mdb to View Kernel Parameters.....	6-19
Solution 6-4: Using Oracle Solaris Studio dbx to Test a Common Application (Optional).....	6-22
Solution 6-5: Using the GUDS Script.....	6-24
<b>Practices for Lesson 7: Processes and Threads.....</b>	<b>7-1</b>
Practices for Lesson 7: Overview.....	7-2
Practice 7-1: Examining Process Life Cycles by Using <code>truss</code> .....	7-3
Practice 7-2: Using the lockstat Command.....	7-4

Practice 7-3: Using the lockstat Provider in DTrace .....	7-5
Practice 7-4: Using plockstat(1M) and the plockstat Provider in DTrace .....	7-6
Practice 7-5: Checking Tunable Parameters with mdb .....	7-7
Practice 7-6: Creating and Managing Processor Sets .....	7-9
Practice 7-7: Changing the Default Scheduling Class to FSS .....	7-11
Practice 7-8: Determining CPU Information and Activity Using DTrace Toolkit Scripts .....	7-13
Practice 7-9: Locating Hot Routines .....	7-18
Solution 7-1: Examining Process Life Cycles by Using truss .....	7-19
Solution 7-2: Using the lockstat Command .....	7-21
Solution 7-3: Using the lockstat Provider in DTrace .....	7-23
Solution 7-4: Using plockstat(1M) and the plockstat Provider in DTrace .....	7-24
Solution 7-5: Checking Tunable Parameters with mdb .....	7-25
Solution 7-6: Creating and Managing Processor Sets .....	7-27
Solution 7-7: Changing the Default Scheduling Class to FSS .....	7-31
Solution 7-8: Determining CPU Information and Activity Using DTrace Toolkit Scripts .....	7-34
Solution 7-9: Locating Hot Routines .....	7-49
<b>Practices for Lesson 8: System Caches and Buses .....</b>	<b>8-1</b>
Practices for Lesson 8: Overview .....	8-2
Practice 8-1: Calculating the Cache-Miss Rate .....	8-3
Practice 8-2: Observing File System Caching .....	8-4
Practice 8-3: Calculating the CPI .....	8-6
Solution 8-1: Calculating the Cache-Miss Rate .....	8-8
Solution 8-2: Observing File System Caching .....	8-9
Solution 8-3: Calculating the CPI .....	8-11
<b>Practices for Lesson 9: System Memory .....</b>	<b>9-1</b>
Practices for Lesson 9: Overview .....	9-2
Practice 9-1: Monitoring Memory Consumption .....	9-3
Practice 9-2: Monitoring Memory Consumption with the DTrace Toolkit .....	9-8
Solution 9-1: Monitoring Memory Consumption .....	9-11
Solution 9-2: Monitoring Memory Consumption with the DTrace Toolkit .....	9-16
<b>Practices for Lesson 10: Disk I/O and ZFS File System .....</b>	<b>10-1</b>
Practices for Lesson 10: Overview .....	10-2
Practice 10-1: Exploring ORION .....	10-3
Practice 10-2: Monitoring Disk I/O Performance .....	10-4
Practice 10-3: Using the DTrace Toolkit and Other Tools to Find the Nature of a Given Test Load .....	10-5
Practice 10-4: Limiting the Size of the ARC in ZFS .....	10-6
Solution 10-1: Exploring ORION .....	10-8
Solution 10-2: Monitoring Disk I/O Performance .....	10-11
Solution 10-3: Using the DTrace Toolkit and Other Tools to Find the Nature of a Given Test Load .....	10-14
Solution 10-4: Limiting the Size of the ARC in ZFS .....	10-15
<b>Practices for Lesson 11: Solaris 11 Network Tuning .....</b>	<b>11-1</b>
Practices for Lesson 11: Overview .....	11-2
Practice 11-1: Configuring the Maximum Transmission Unit (MTU) .....	11-3
Practice 11-2: Configuring a Link Aggregation .....	11-4
Practice 11-3: Configuring an Integrated Load Balancer .....	11-5
Practice 11-4: Controlling Data Flows .....	11-14
Practice 11-5: Monitoring Network Performance .....	11-15
Solution 11-1: Configuring the Maximum Transmission Unit (MTU) .....	11-18

Solution 11-2: Configuring a Link Aggregation.....	11-19
Solution 11-3: Configuring an Integrated Load Balancer .....	11-22
Solution 11-4: Controlling Data Flows .....	11-31
Solution 11-5: Monitoring Network Performance.....	11-32
<b>Practices for Lesson 12: Resource Management.....</b>	<b>12-1</b>
Practices for Lesson 12: Overview.....	12-2
Practice 12-1: Managing Resources .....	12-3
Solution 12-1: Managing Resources .....	12-8
<b>Practices for Lesson 13: Managing Zone-Wide Resources .....</b>	<b>13-1</b>
Practices for Lesson 13: Overview.....	13-2
Practice 13-1: Managing Zone-Wide Resources and Controls .....	13-3
Solution 13-1: Managing Zone-Wide Resources and Controls .....	13-9
<b>Practices for Lesson 14: Performance Analysis and Testing .....</b>	<b>14-1</b>
Practices for Lesson 14.....	14-2



## Preface

## User, Group, and Role Commands

---

Command	Description
<code>groupadd -g 110 eng1</code>	Creates a group name <code>eng1</code>
<code>useradd -u 100 -g 110 \ -d /export/home/dave \ -s /usr/bin/bash \ -c "Dave" -m \ -k /etc/skel dave</code>	Adds user <code>dave</code> to group <code>110</code>
<code>userdel -r acct_login</code>	Removes a user account, including home directory
<code>roleadd -c comment \ -g group -m homedir -u UID -s shell -P profile rolename</code>	Creates a role
<code>usermod -u UID -R +rolename login-name</code>	Assigns a role to a local user
Roles	Determines which roles you can assume
<code>roledel -r role</code>	Removes a role and its home directory
<code>passwd username</code>	Sets a user password
Logins	Displays user and system login information
<code>id -p</code>	Displays the current project ID in addition to the user and group IDs

## Preface

### Performance Monitoring Commands

Command	Description
iostat	Reports terminal and disk I/O activity and CPU utilization
vmstat	Reports CPU, memory utilization, and disk-I/O statistics
netstat	Reports network statistics
sar	Reports statistics on a wide variety of system activities
mpstat	Reports performance statistics for each processor
nfsstat	Reports Network File System (NFS)–related statistics
kstat	Reports kernel statistics
prstat	Reports various statistics for processes and projects
ps	Prints information about active processes
truss	Traces system calls performed, signals received, and machine faults incurred
apptrace	Traces application function calls to shared libraries
pargs	Displays environment and arguments information for any given process
mdb	Utility for low-level debugging and editing of the OS
zonestat	Reports zone resource consumption
pgstat	Displays utilization statistics about Processor Groups
prtdiag	Displays diagnostic information
prtconf	Prints the system configuration information
cputrack	Monitors the overall behavior of the CPUs in the system
psrinfo	Displays information about system CPUs
fmstat	Reports statistics associated with the Solaris Fault Manager
flowstat	Reports statistics on network flow controls
dladm status	Reports statistics on system data links
lockstat	Displays kernel locking and profiling statistics
plockstat	Displays user-level locking statistics
trapstat	Displays runtime trap statistics on UltraSPARC-based systems
poolstat	Reports active resource pool statistics
rcapstat	Monitors the resource utilization of capped projects
dtrace	Provides a generic interface to all of the essential monitoring services provided by the DTrace facility



## Preface

### Boot Environment (BE) Commands

---

Command	Description
<code>beadm create <i>BENAME</i></code>	Creates a new boot environment
<code>beadm activate <i>BENAME</i></code>	Changes the default boot environment
<code>beadm list</code>	Lists the zone boot environments on a system
<code>beadm list -s</code>	Enables viewing of the zone boot environments and snapshots on a system
<code>beadm create <i>BE@snapshot</i></code>	Creates a snapshot of the existing boot environment
<code>beadm create -e <i>BE@snapshot BENAME</i></code>	Creates a new boot environment for a snapshot
<code>beadm destroy <i>BENAME</i></code>	Deletes a boot environment
<code>beadm rename <i>BENAME newBENAME</i></code>	Changes the name of an existing boot environment to a new name
<code>beadm mount <i>BENAME mountpoint</i></code>	Mounts a BE

## Preface

### Zone Files and Directories

---

File/Directory	Description
/etc/zones	Zone configuration directory
/etc/zones/SYSdefault.xml	Default <code>solaris</code> brand zone template
/etc/zones/SYSblank.xml	Blank zone template
/etc/zones/SYSdefault-shared-ip.xml	Shared-IP zone template
/etc/zones/SYSsolaris10.xml	<code>solaris10</code> brand zone template
/etc/zones/index	Zone index used to start and stop zones
/usr/share/auto_install/manifest/zone_default.xml	Default zone auto-install manifest
/usr/share/auto_install/sc_profiles/enable_sci.xml	Default zone system configuration manifest

## Preface

### zonecfg Commands

---

Command	Description
zonecfg -z <zone>	Creates a zone
zonecfg -z <zone> -f /path/file	Creates a zone from a command file
zonecfg -z <zone> info	Displays zone configuration
zonecfg -z <zone> delete -F	Deletes a zone
zonecfg -z <zone> export -f /path/file	Exports zone configuration to a file in the form of a command file

## Preface

## Zone Resources

Resource	Description
<pre>zonecfg:workzone&gt; add admin zonecfg:workzone:admin&gt; set user=zadmin zonecfg:workzone:admin&gt; set \ auths=login,manage zonecfg:workzone:admin&gt; end</pre>	Delegates an administrator to a zone
<pre>zonecfg:workzone&gt; set ip-type=shared zonecfg:workzone&gt; add net zonecfg:workzone:net&gt; set physical=nge0 zonecfg:workzone:net&gt; set \ address=192.168.0.1 zonecfg:workzone:net&gt; set \ defrouter=10.0.0.1 zonecfg:workzone:net&gt; end</pre>	Adds a network (net) resource to a shared-ip zone
<pre>zonecfg:workzone&gt; set ip-type=exclusive zonecfg:workzone:anet&gt; add anet zonecfg:workzone:anet&gt; set \ linkname=net0 zonecfg:workzone:anet&gt; set \ lower-link=auto zonecfg:workzone:anet&gt; set \ mac-address=random zonecfg:workzone:anet&gt; set \ link-protection=mac-nospoof zonecfg:workzone:anet&gt; end</pre>	Adds an automatic network (anet) resource to an exclusive-ip zone
<pre>zonecfg:workzone&gt; add fs zonecfg:workzone:fs&gt; set \ dir=/shared/fs1 zonecfg:workzone:fs&gt; set \ special=pool1/fs1 zonecfg:workzone:fs&gt; set type=zfs zonecfg:workzone:fs&gt; end</pre>	Adds a file system to a zone
<pre>zonecfg:workzone&gt; add dataset zonecfg:workzone&gt; set name=pool/dataset zonecfg:workzone&gt; set alias=mydata zonecfg:workzone&gt; end</pre>	Adds a dataset to a zone
<pre>zonecfg:workzone&gt; add device zonecfg:workzone:device&gt; set \ match=/dev/*dsk/cXtYdZ* zonecfg:workzone:device&gt; set \ partition=true zonecfg:workzone:device&gt; end</pre>	Adds a device capable of being partitioned to a zone
<pre>zonecfg:workzone&gt; add dedicated-cpu zonecfg:workzone:dedicated-cpu&gt; set \ ncpus=3</pre>	Dedicates CPUs to a zone

## Preface

zonecfg:workzone:dedicated-cpu> set \ importance=2 zonecfg:workzone:dedicated-cpu> end	
zonecfg:workzone> add capped-cpu zonecfg:workzone:capped-cpu> set \ ncpus=2 zonecfg:workzone:capped-cpu> end	Places a CPU cap on a zone
zonecfg:workzone> add capped-memory zonecfg:workzone:capped-memory> set \ physical=100m zonecfg:workzone:capped-memory> set \ swap=100m zonecfg:workzone:capped-memory> set \ locked=40m zonecfg:workzone:capped-memory> end	Places memory caps on a zone
zonecfg:workzone> set pool=pool1	Adds a resource pool to a zone
zonecfg:workzone> set \ scheduling-class=FSS zonecfg:workzone> set cpu-shares=50	Makes FSS the default zone scheduler
zonecfg:workzone> add rctl zonecfg:workzone:rctl> set \ name=zone.cpu-shares zonecfg:workzone:rctl> add value \ (priv=privileged,limit=5,action=deny) zonecfg:workzone:rctl> end	Sets resource controls on a zone
zonecfg:workzone> add attr zonecfg:workzone:attr> set name=comment zonecfg:workzone:attr> set type=string zonecfg:workzone:attr> set \ value="This is my work zone" zonecfg:workzone:attr> end	Creates comments in a zone configuration

## Preface

### Zone States and Associated Commands

---

State	Associated Commands
Configured	<code>zonecfg -z zonename verify</code> <code>zonecfg -z zonename delete</code> <code>zoneadm -z zonename attach</code> <code>zoneadm -z zonename verify</code> <code>zoneadm -z zonename install</code> <code>zoneadm -z zonename clone</code>
Installed	<code>zoneadm -z zonename ready</code> <code>zoneadm -z zonename boot</code> <code>zoneadm -z zonename uninstall</code> <code>zoneadm -z zonename move path</code> <code>zoneadm -z zonename detach</code> <code>zoneadm -z zonename clone newzonename</code> <code>zoneadm -z zonename boot -- boot-options</code>
Incomplete	<code>zoneadm -z zonename uninstall</code>
Ready	<code>zoneadm -z zonename boot</code> <code>zoneadm -z zonename halt</code>
Running	<code>zlogin options zonename</code> <code>zoneadm -z zonename reboot</code> <code>zoneadm -z zonename halt</code> <code>zoneadm -z zonename shutdown</code>
Shutting down/Down	<code>zoneadm -z zonename halt</code>

## Preface

### zoneadm Commands

---

Command	Description
zoneadm -z <zone> install	Installs a zone
zoneadm -z <zone> install \ -m manifest -c profile	Installs a zone by using custom manifest and SC profile
zoneadm -z <zone> uninstall	Uninstalls a zone
zoneadm -z <zone> boot	Boots a zone
zoneadm -z <zone> reboot	Reboots a zone
zoneadm -z <zone> shutdown	Shuts down a zone gracefully
zoneadm -z <zone> halt	Halts a zone immediately
zoneadm -z <zone> verify	Verifies a zone configuration
zoneadm -z <zone> ready	Places a zone in the ready state
zoneadm list -cv	Enables viewing of zone status

## Preface

### System Configuration Commands

Command	Description
svccfg -s svc:/system/identity:node \ setprop config/nodename = "yourhost" svcadm refresh svc:/system/identity:node svcadm restart svc:/system/identity:node	Configures nodename
svccfg -s keymap:default setprop \ keymap/layout = UK-English svcadm refresh keymap svcadm restart keymap	Configures keyboard layout
svccfg -s timezone:default setprop \ timezone/localtime = astring: US/Mountain svcadm refresh timezone:default	Configures locale
sysconfig configure -s	Unconfigures a system and starts the system configuration interactive (SCI) tool on reboot
sysconfig create-profile -o sc-profile.xml	Creates a system configuration profile
sysconfig configure -c sc-profile.xml	Configures a zone from an SC profile
zoneadm -z <zone> install \ -c sc-profile.xml	Installs a zone and configures the zone by using an SC profile



## Preface

### IPS Packaging Commands

---

Command	Description
<code>pkg install &lt;pkg&gt;</code>	Installs a package
<code>pkg install -nv &lt;pkg&gt;</code>	Performs a dry run on a package installation
<code>pkg update</code>	Updates all system packages
<code>pkg update -nv</code>	Performs a dry run on package update
<code>pkg search &lt;pkg&gt;</code>	Searches the repository for a package
<code>pkg list &lt;pkg&gt;</code>	Lists an installed package
<code>pkg contents &lt;pkg&gt;</code>	Displays the contents of a package
<code>pkg publisher</code>	Displays the current publisher
<code>pkg set-publisher \ - G https://oldpublisher/ \ - g https://newpublisher/ solaris</code>	Changes the publisher

## Preface

## Networking Commands

Command	Description
<code>dladm show-phys</code>	Shows physical network devices
<code>ipadm create-ip net0</code> <code>ipadm create-addr -T static \</code> <code>-a local=10.9.8.7/24 net0/addr</code> <code>ipadm show-addr</code>	Creates an IPv4 static IP configuration
<code>ipadm create-ip net0</code> <code>ipadm create-addr -T dhcp net0/addr</code>	Creates an interface with DHCP configuration
<code>route -p add default 192.168.0.1</code>	Creates a default route
<code>dladm create-vnic -l net0 vnic0</code> <code>ipadm create-ip vnic0</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.80 vnic0/v4</code>	Creates a virtual network over an existing physical interface
<code>dladm create-etherstub stub0</code> <code>dladm create-vnic -l stub0 vnic0</code> <code>dladm create-vnic -l stub0 vnic1</code>	Creates a virtual network without physical interface
<code>dladm show-phys</code>	Shows physical network devices
<code>flowadm add-flow -l vnic0 \</code> <code>-a remote_ip=192.168.0.100 flow0</code> <code>flowadm set-flowprop \</code> <code>-p maxbw=20 flow0</code>	Restricts the bandwidth going to IP address 192.168.0.100
<code>dladm create-aggr -l net0 \</code> <code>-l net1 aggr0</code> <code>ipadm create-ip aggr0</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.101/24 aggr0/v4</code>	Configures link aggregation
<code>ipadm create-ip net0</code> <code>ipadm create-ip net1</code> <code>ipadm create-ip net2</code> <code>ipadm create-ipmp ipmp0</code> <code>ipadm add-ipmp -i net0 \</code> <code>-i net1 -i net2 ipmp0</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.102/24 ipmp0/v4</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.110/24 net0/test</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.111/24 net1/test</code> <code>ipadm create-addr -T static \</code> <code>-a 192.168.0.112/24 net2/test</code>	Configures IPMP group

## Preface

### Automatic Installation (AI) Commands and Elements

Command/Element	Description
<code>installadm create-service -n x86_ai \</code> <code>-i &lt;IPAddr&gt; -c 10 -s /path/solaris_ai.iso</code>	Creates a new AI service
<code>installadm create-client \</code> <code>-e 08:00:27:85:C7:D6 -n x86_ai</code>	Adds a client to an AI service
<code>installadm create-manifest -n x86_ai \</code> <code>-f manifest.xml -C criteria.xml</code>	Adds a manifest to an AI service
<code>installadm create-profile -n x86_ai \</code> <code>-f profile.xml -p s11-serv2 -C criteria.xml</code>	Adds a system configuration (SC) profile to an AI service
<code>installadm create-manifest -n x86_ai \</code> <code>-f zone-manifest.xml -c zonename=myzone</code>	Adds a zone manifest to an AI service
<code>installadm create-profile -n x86_ai \</code> <code>-f zone-profile.xml -p profilename \</code> <code>-c zonename=myzone</code>	Adds a zone SC profile to an AI service
<code>&lt;configuration type="zone" name="myzone" \</code> <code>source="http://system/zones/workzone.cfg"/&gt;</code>	Is the zone configuration element in the AI system manifest file
<code>&lt;software_data action="install"&gt;</code> <code>&lt;name&gt;pkg:/group/system/solaris-small-</code> <code>server&lt;/name&gt;</code> <code>&lt;name&gt;pkg:/package_name1&lt;/name&gt;</code> <code>&lt;name&gt;pkg:/package_name2&lt;/name&gt;</code> <code>&lt;name&gt;pkg:/package_name3&lt;/name&gt;</code> <code>&lt;/software_data&gt;</code>	Adds additional package elements to the AI zone manifest file

## Preface

## ZFS Commands

Command	Description
<code>zpool create zfspool1 &lt;disk1&gt;</code>	Creates a ZFS storage pool
<code>zpool attach zfspool1 disk1&gt; &lt;disk2&gt;</code>	Creates a mirror in an existing storage pool
<code>zpool status -v</code>	Checks ZFS pool status
<code>zpool destroy zfspool1</code>	Destroys a ZFS storage pool
<code>zpool create zfsmirror1 mirror &lt;disk1&gt; \ &lt;disk2&gt;</code>	Creates a new ZFS mirror storage pool
<code>zpool create zfsraidz1 raidz &lt;disk1&gt; \ &lt;disk2&gt; &lt;disk3&gt;</code>	Creates a ZFS RAIDZ storage pool
<code>zpool add zfspool1 &lt;new_disk&gt;</code>	Expands a ZFS pool
<code>zpool add zfspool1 spare &lt;sparedisk&gt;</code>	Adds a spare disk to a ZFS pool
<code>zpool remove zfspool1 &lt;sparedisk&gt;</code>	Removes a spare disk from a ZFS pool
<code>zfs create zfspool1/dataset1</code>	Creates a ZFS file system (dataset)
<code>zfs set mountpoint=/new_path \ zfspool1/dataset1</code>	Sets a ZFS pool mountpoint
<code>zfs list</code>	Lists ZFS file systems (datasets)
<code>zfs destroy zfspool1/dataset1</code>	Destroys a ZFS file system
<code>zfs set \ share=name=dataset1,path=/zfspool1/dataset1,prot=nfs zfspool1/dataset1</code> <code>zfs set sharenfs=on zfspool1/dataset1</code>	Shares a ZFS file system by using NFS
<code>zfs snapshot zfspool1/dataset1@friday</code>	Creates a ZFS snapshot
<code>zfs rollback zfspool1/dataset1@friday</code>	Rolls back a ZFS file system to its snapshot
<code>zfs send zfspool1/dataset1@friday &gt; \ /var/backups/dataset1.bkp</code>	Backs up a ZFS file system

## Preface

### DTrace One-Liners

One-Liner	Description
<code>dtrace -n 'proc:::exec-success { trace(curpsinfo-&gt;pr_psargs); }'</code>	Views new processes with arguments
<code>dtrace -n 'syscall::open*:entry { printf("%s %s",execname,copyinstr(arg0)); }'</code>	Views files opened by process
<code>dtrace -n 'syscall:::entry { @num[execname] = count(); }'</code>	Views syscall count by program
<code>dtrace -n 'syscall:::entry { @num[probefunc] = count(); }'</code>	Views syscall count by syscall
<code>dtrace -n 'syscall:::entry { @num[pid,execname] = count(); }'</code>	Displays read bytes by process
<code>dtrace -n 'sysinfo:::writech { @bytes[execname] = sum(arg0); }'</code>	Displays write bytes by process
<code>dtrace -n 'sysinfo:::readch { @dist[execname] = quantize(arg0); }'</code>	Displays read size distribution by process
<code>dtrace -n 'sysinfo:::writech { @dist[execname] = quantize(arg0); }'</code>	Displays write size distribution by process
<code>dtrace -n 'io:::start { printf("%d %s %d",pid,execname,args[0]-&gt;b_bcount); }'</code>	Displays disk size by process
<code>dtrace -n 'vminfo:::pgpgin { @pg[execname] = sum(arg0); }'</code>	Displays pages paged in by process
<code>dtrace -n 'vminfo:::as_fault { @mem[execname] = sum(arg0); }'</code>	Displays minor faults by process
<code>dtrace -n 'sdt:::interrupt-start { @num[cpu] = count(); }'</code>	Displays interrupts by CPU
<code>zfs destroy zfspool1/dataset1</code>	Destroys a ZFS file system
<code>dtrace -n 'proc:::signal-send /pid/ { printf("%s -%d %d",execname,args[2],args[1]-&gt;pr_pid); }'</code>	Trace signal details
<code>dtrace -n 'lockstat:::adaptive-block { @time[execname] = sum(arg1); }'</code>	Displays lock time by process name
<code>dtrace -n 'lockstat:::adaptive-block { @time[execname] = quantize(arg1); }'</code>	Displays lock distribution by process name
<code>dtrace -n 'syscall:::entry { @num[zonename] = count(); }'</code>	Displays syscalls by zonename
<code>dtrace -n 'sched:::off-cpu { @[execname, stack()] = count(); }'</code>	Traces why threads are context switching off the CPU, from the kernel perspective
<code>dtrace -n 'fbt:::entry { @calls[probemod] = count(); }'</code>	Displays kernel function calls by module



# **Practices for Lesson 1: Course Introduction**

## **Chapter 1**

## Practices for Lesson 1

---

### Practices Overview

There is no practice for Lesson 1.



# **Practices for Lesson 2: Introducing Performance Management**

## **Chapter 2**

## Practices for Lesson 2: Overview

---

### Practices Overview

In these practices, you will answer questions and perform steps from the material presented in this lesson.

Solutions for each task in this practice are provided after the practice exercises are introduced.

## Practice 2-1: Assessing System Configuration

### Overview

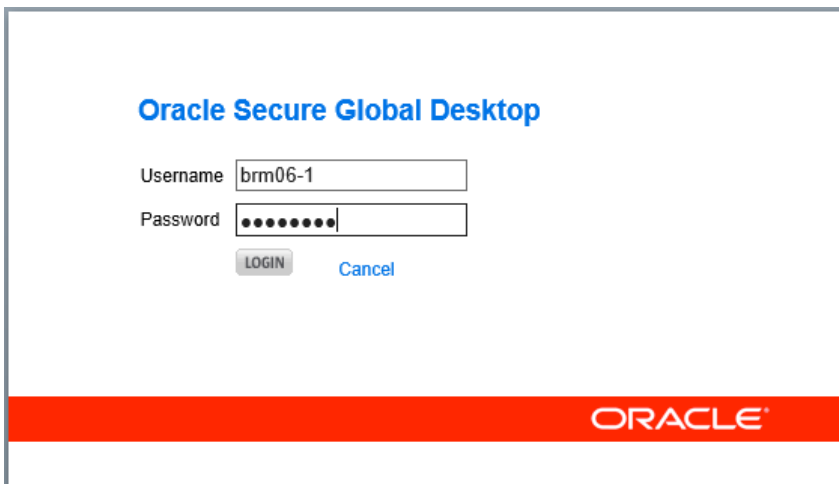
A critical task when you perform system tuning is assessing system configuration. Understanding how the system is configured provides a more complete picture when you analyze a performance problem. And it is not unusual during your analysis to find that the root cause of poor performance is associated with system configuration. In this practice, you use various Oracle Solaris utilities to assess system configuration.

**Note:** The command responses in this guide are examples only. The command responses on your server might be different.

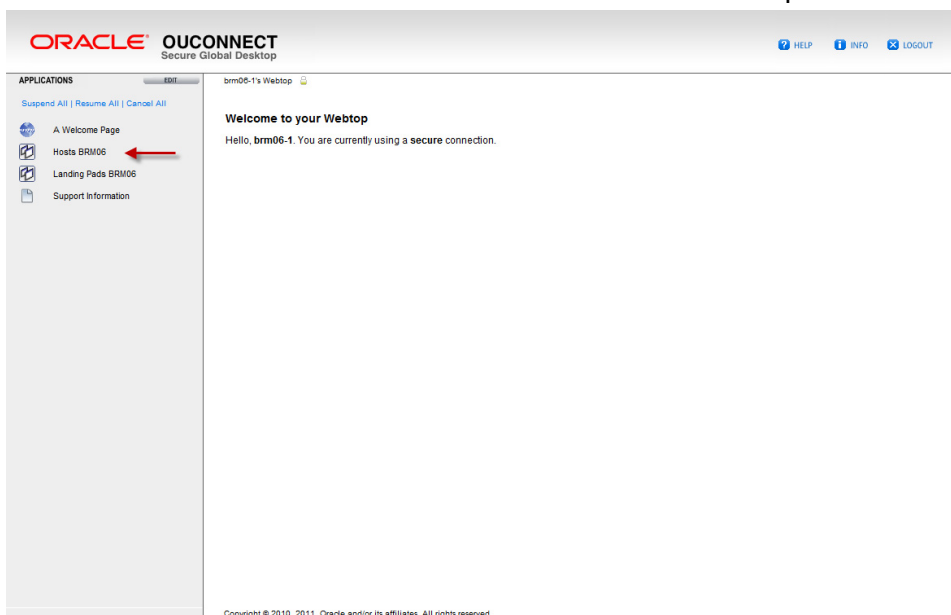
### Task

Perform the following steps to assess system configuration:

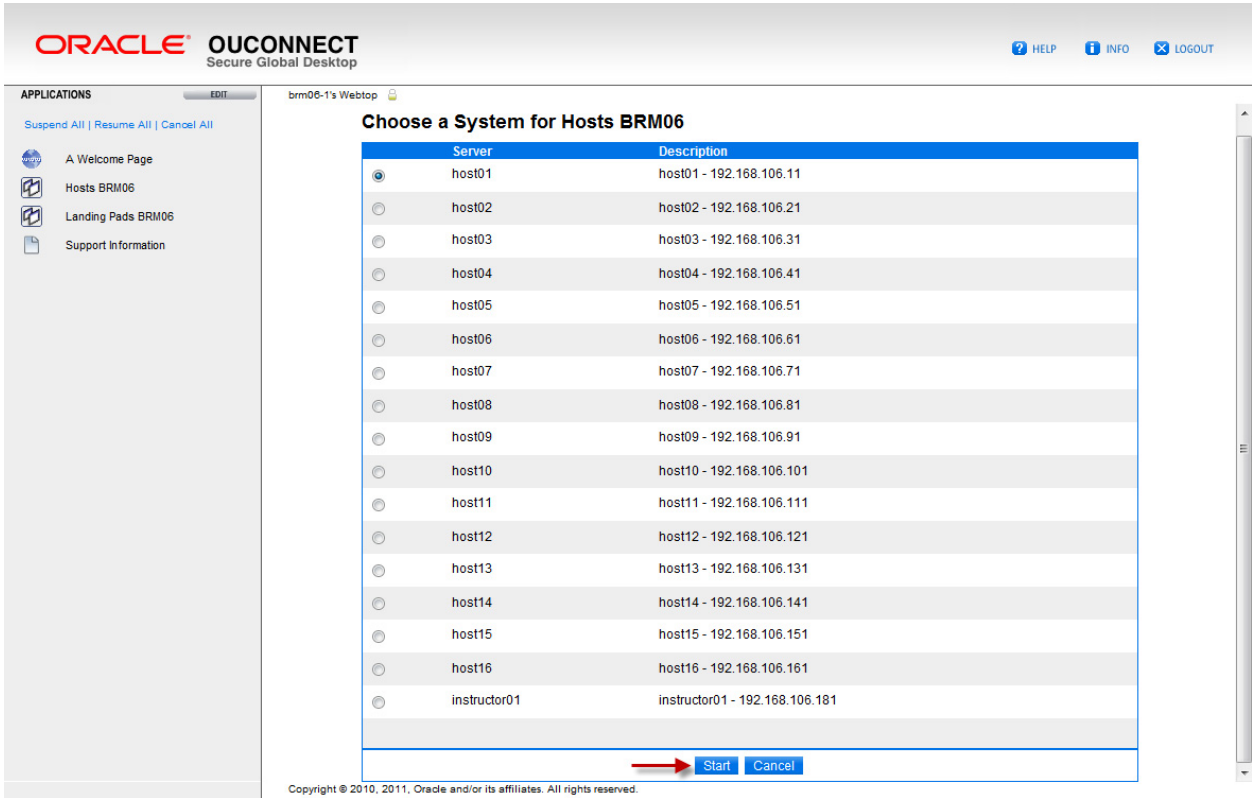
1. Log in to your lab environment. See your instructor for login credentials.

The image shows the Oracle Secure Global Desktop login interface. At the top, the text "Oracle Secure Global Desktop" is displayed in blue. Below this, there are two input fields: "Username" with the value "brm06-1" and "Password" with masked characters. To the right of the password field are two buttons: "LOGIN" and "Cancel". At the bottom of the interface is a red horizontal bar with the "ORACLE" logo in white.

Open a Gnome desktop session on your assigned server by selecting your assigned server from the Oracle OUCONNECT Secure Global Desktop.



Select your assigned host and click Start.



2. If you are not root, run the `su` command to assume primary administrator privileges.

```
root@host01:~$ su -
Password: cangetin
Note: Passwords are site-specific and are subject to change. If the password shown in
the example does not work, check with your instructor for the latest password.
root@host01:~#
```

3. Determine the build number of the installed operating system.
4. Run the `format` command to determine the hard disk configuration.
5. Use `zpool status` to determine the hard disks that are currently being used.
6. Run `zfs list` to determine the file system layout.
7. Determine the size of physical memory.
8. Determine the number of CPUs.
9. Determine the number of data links.
10. Determine the current IP addresses configuration.
11. Determine the current zone configuration.
12. Verify that an IPS service is available.

## Practice 2-2: Using the `kstat` Utility

### Overview

In this practice, you find useful `kstat` statistics.

**Note:** The command responses displayed in your practices are normally different from the examples shown in this guide.

### Task

Practice using the `kstat` utility by performing the following steps:

1. Refer to the man page for the `kstat` command.
2. Complete the following table with the options that you use to display the required information:

Information Required	Option
To list only <code>kstat</code> names	
To display only statistics that match the specified module	
To display the statistics of a specified <code>kstat</code> name	
To display only statistics that match the specified statistic	

3. Run the following command:

```
root@host01:~# kstat -m cpu_stat | grep user
      user                2904
      user               35053
      user                5132
      user                826
      user                 932
      user               23434
      user               31976
      ...
```

Observe the value that is displayed. What action does this command perform?

4. Open a second terminal window and start a series of find operations by using the following command:

```
root@host01:~# while true
> do
> find /
> done > /dev/null 2>&1
```

5. While the `find` commands execute in the other window, issue the following command three times, a few seconds apart from each other. Notice the differences from the value displayed in Step 3.

```
root@host01:~# kstat -m cpu_stat | grep user
...
root@host01:~# kstat -m cpu_stat | grep user
...
root@host01:~# kstat -m cpu_stat | grep user
...
```

6. While the `find` command is still running in the other window, use the following command to identify all `kstats` with the name `heap`:

```
root@host01:~# kstat -n heap
```

7. While the `find` command is still running in the other window, use the following command to identify all `kstat` statistics that count cache hits:

```
root@host01:~# kstat -s hits
```

8. Enter Control -C in the console window that is running the `find` commands to exit the `while` loop.

## Practice 2-3: Using the `truss` Utility

### Overview

In this practice, you use `truss` to find what facility a given tool is using to get its information.

### Task

Practice using the `truss` utility by performing the following steps:

1. Refer to the `man` page for the `truss` command:

```
root@host01:~# man truss
```

- a. What is the `-t` option for? \_\_\_\_\_
- b. What does the `-f` option mean? \_\_\_\_\_

2. Enter the following command:

```
root@host01:~# truss -ft open ps
```

Based on the output, does it look like `ps` is based on `dtrace`, `procfs`, or `kstat`?  
\_\_\_\_\_

3. Enter the following command:

```
root@host01:~# truss -ft ioctl vmstat 1 1
```

Based on the output, does it look like `vmstat` is based on `dtrace`, `procfs`, or `kstat`?  
\_\_\_\_\_

4. Try using `truss -ft open`, `truss -ft ioctl`, or just `truss` to see which facility the following commands are based on: `dtrace`, `procfs`, `kstat`, or none of these?
  - `plockstat date` \_\_\_\_\_
  - `sar 1 1` \_\_\_\_\_
  - `intrstat 2 1` \_\_\_\_\_
  - `ptree $$` \_\_\_\_\_
  - `iostat -xzn 2 1` \_\_\_\_\_
  - `truss -texit date` \_\_\_\_\_
  - `uptime` \_\_\_\_\_

## Solution 2-1: Assessing System Configuration

### Overview

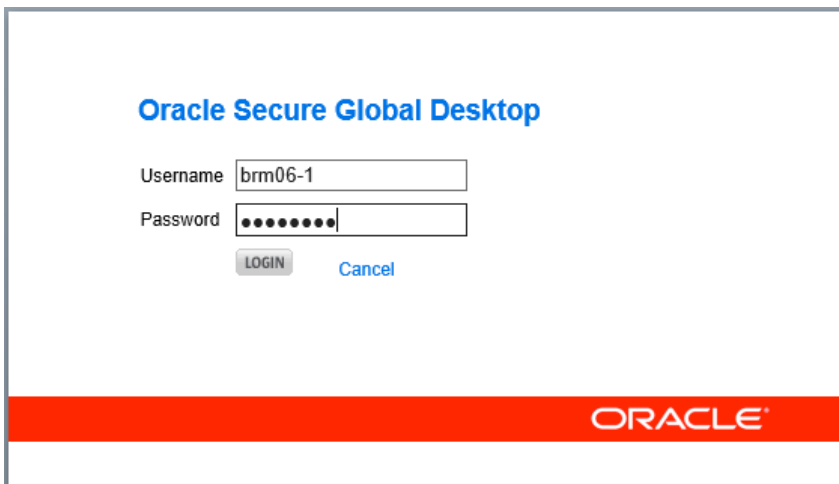
A critical task when you perform system tuning is assessing system configuration. Understanding how the system is configured provides a more complete picture when you analyze a performance problem. It is not unusual during your analysis to find that the root cause of poor performance is associated with system configuration. In this practice, you use various Oracle Solaris utilities to assess system configuration.

**Note:** The command responses in this guide are examples only. The command responses on your server might be different.

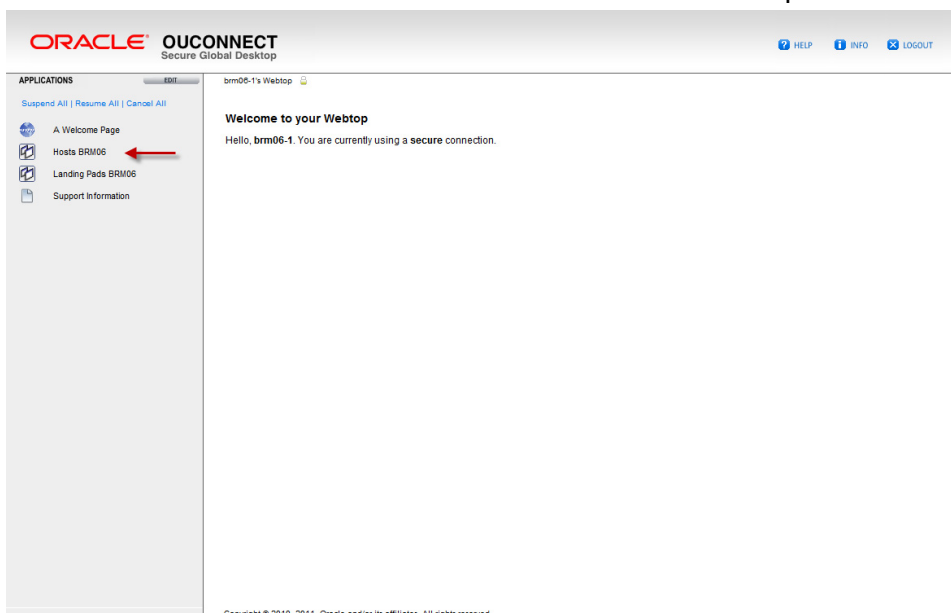
### Task

Perform the following steps to assess system configuration:

1. Log in to your practice environment. See your instructor for login credentials.

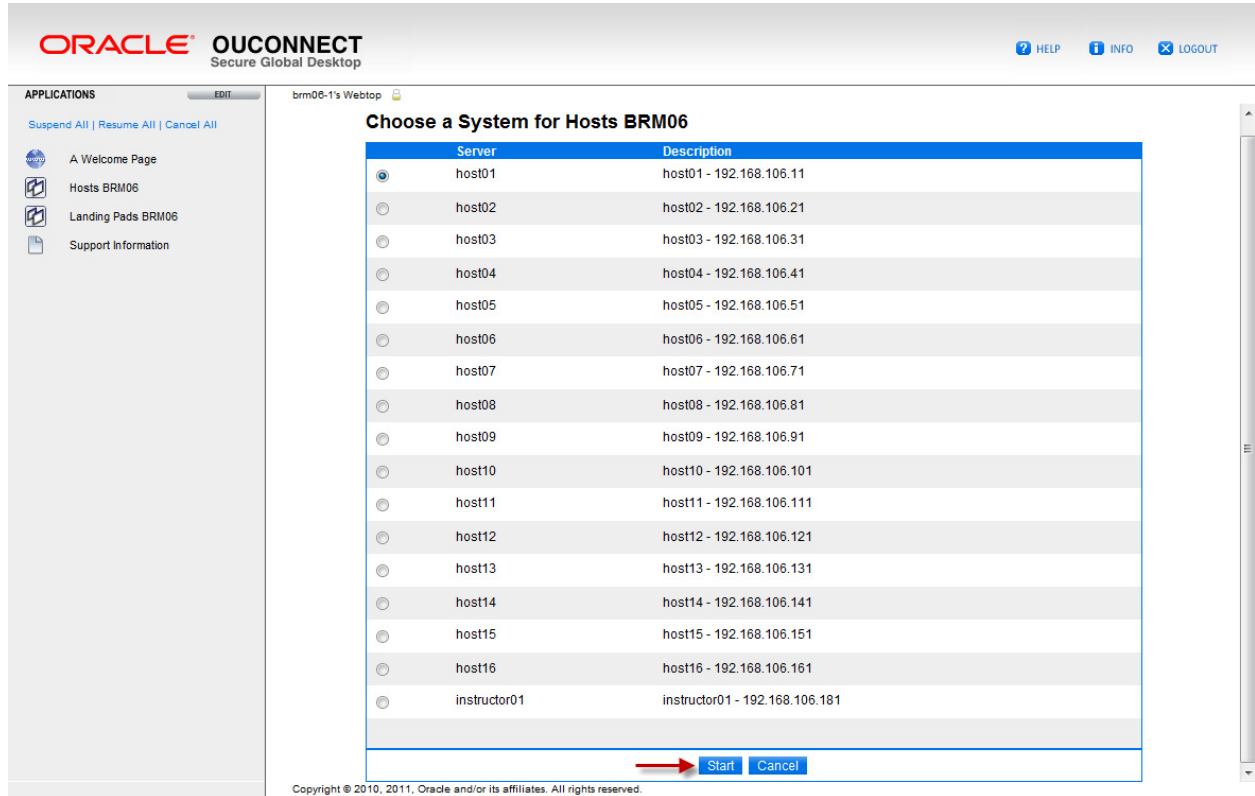


Open a Gnome desktop session on your assigned server by selecting your assigned server from the Oracle OUCONNECT Secure Global Desktop.





Select your assigned host and click Start.



- If you are not root, run the `su` command to assume primary administrator privileges.

```
root@host01:~$ su -
```

```
Password: cangetin
```

**Note:** Passwords are site-specific and are subject to change. If the password shown in the example does not work, check with your instructor for the latest password.

```
root@host01:~#
```

- Determine the build number of the installed operating system.

```
root@host01:~# cat /etc/release
```

```
Oracle Solaris 11 11/11 SPARC
```

```
Copyright(c) 1983, 2012, Oracle and/or its affiliates. All rights reserved.
```

```
Assembled 19 September 2012
```

```
root@host01:~# pkg list entire
```

```
NAME (PUBLISHER)      VERSION                IFO
entire                0.5.11-0.175.1.0.0.24.2  i--
```

- Run the `format` command to determine the hard disk configuration.

```
root@host01:~# format
```

```
Searching for disks...done
```

```
AVAILABLE DISK SELECTIONS:
```

```
0. c3t0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424> Primary
   /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0
1. c3t1d0 <FUJITSU-MAV2073RCSUN72G-0301-68.37GB>
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

```
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@1,0
Specify disk (enter its number): ^D
```

5. Use `zpool status` to determine the hard disks that are currently being used.

```
root@host01:~# zpool status
pool: rpool
state: ONLINE
scan: none requested
config:

    NAME            STATE        READ WRITE CKSUM
    rpool            ONLINE         0     0     0
    c3t0d0s0         ONLINE         0     0     0

errors: No known data errors
```

6. Run `zfs list` to determine the file system layout.

```
root@host01:~# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                               18.7G  48.2G   73.5K  /rpool
rpool/ROOT                          7.13G  48.2G    31K  legacy
rpool/ROOT/solaris                  7.13G  48.2G   6.52G  /
rpool/ROOT/solaris.orig              148K  48.2G   4.36G  /
rpool/ROOT/solaris.orig/var          63K   48.2G   210M  /var
rpool/ROOT/solaris/var               590M  48.2G   534M  /var
rpool/VARSHARE                      46.5K  48.2G   46.5K  /var/share
rpool/dump                          8.19G  48.5G   7.94G  -
...
```

7. Determine the size of physical memory.

```
root@host01:~# prtconf | grep Memory
Memory size: 16256 Megabytes
```

8. Determine the number of CPUs.

```
root@host01:~# psrinfo
0      on-line   since 03/21/2000 19:59:44
1      on-line   since 03/21/2000 19:59:45
2      on-line   since 03/21/2000 19:59:45
3      on-line   since 03/21/2000 19:59:45
4      on-line   since 03/21/2000 19:59:45
...
30     on-line   since 03/21/2000 19:59:45
31     on-line   since 03/21/2000 19:59:45
```

9. Determine the number of data links.

```
root@host01:~# dladm show-link
LINK            CLASS      MTU      STATE      OVER
```

net1	phys	1500	unknown	--
net2	phys	1500	unknown	--
net0	phys	1500	up	--
net3	phys	1500	unknown	--

10. Determine the current IP addresses configuration.

```
root@host01:~# ipadm show-addr
```

ADDROBJ	TYPE	STATE	ADDR
lo0/v4	static	ok	127.0.0.1/8
lo0/zoneadmd.v4	static	ok	127.0.0.1/8
lo0/zoneadmd.v4a	static	ok	127.0.0.1/8
lo0/zoneadmd.v4b	static	ok	127.0.0.1/8
net0/v4	static	ok	192.168.106.151/24
net0/zoneadmd.v4	static	ok	192.168.106.154/24
net0/zoneadmd.v4a	static	ok	192.168.106.155/24
net0/zoneadmd.v4b	static	ok	192.168.106.156/24
lo0/v6	static	ok	::1/128
lo0/zoneadmd.v6	static	ok	::1/128
lo0/zoneadmd.v6a	static	ok	::1/128
lo0/zoneadmd.v6b	static	ok	::1/128
net0/v6	addrconf	ok	fe80::203:baff:fed8:e210/10

11. Determine the current zone configuration.

```
root@host01:~# zoneadm list -cv
```

ID	NAME	STATUS	PATH	BRAND	IP
0	global	running	/	solaris	shared
1	storage	running	/zones/storage	solaris	shared
2	database	running	/zones/database	solaris	shared
3	web	running	/zones/web	solaris	shared

12. Verify that an IPS service is available.

```
root@host01:~# pkg publisher
```

PUBLISHER	TYPE	STATUS	URI
solaris	origin	online	F file:///net/192.168.106.1/seif/os/s11_ul_repo/
ouif	origin	online	F file:///net/192.168.106.1/seif/Solaris/s11_ouif_repo/

```
root@host01:~# pkg search solaris-small-server
```

INDEX	ACTION	VALUE	PACKAGE
pkg.fmri	set	solaris/group/system/solaris-small-server	
pkg:/group/system/solaris-small-server@0.5.11-0.175.1.0.0.24.3			
...			

## Solution 2-2: Using the `kstat` Utility

### Overview

In this practice, you find useful `kstat` statistics.

**Note:** The command responses displayed in your practice are normally different from the examples shown in this guide.

### Task

Practice using the `kstat` utility by performing the following steps:

1. Refer to the man page for the `kstat` command:

```
root@host01:~# man kstat
```

2. Complete the following table with the options that you use to display the required information:

Information Required	Option
To list only <code>kstat</code> names	<code>-l</code>
To display only statistics that match the specified module	<code>-m module</code>
To display the statistics of a specified <code>kstat</code> name	<code>-n name</code>
To display only statistics that match the specified statistic	<code>-s statistic</code>

3. Run the following command:

```
root@host01:~# kstat -m cpu_stat | grep user
      user                2904
      user               35053
      user                5132
      user                 826
      user                 932
      user               23434
      user               31976
      ...
```

Observe the value that is displayed. What action does this command perform?

*This command displays user statistics from the `cpu_stat` module.*

4. Open a second terminal window and start a series of find operations by using the following command as root:

```
root@host01:~$ su -
Password: cangetin
root@host01:~# while true
> do
> find /
> done > /dev/null 2>&1
```

5. While the `find` commands execute in the other window, issue the following command three times, a few seconds apart from each other. Notice the differences from the value displayed in Step 3.

```

root@host01:~# kstat -m cpu_stat | grep user
      user                2905
      user                35138
      user                5132
...
root@host01:~# kstat -m cpu_stat | grep user
...
root@host01:~# kstat -m cpu_stat | grep user
...

```

6. While the `find` command is still running in the other window, use the following command to identify all `kstats` with the name `heap`:

```

root@host01:~# kstat -n heap
module: vmem                instance: 1
name:   heap                class:   vmem
  alloc                4145
  contains              0
  contains_search       0
  crtime                61.617705244
  fail                  0
  free                  238
  lookup                17
  mem_import            0
  mem_inuse             3855335424
  mem_total             17179869184
  populate_fail         0
  populate_wait         0
  search                26
  snaptime              427404.31119684
  vmem_source           0
  wait                  0

```

7. While the `find` command is still running in the other window, use the following command to identify all `kstat` statistics that count cache hits:

```

root@host01:~# kstat -s hits
module: ufs                instance: 0
name:   inode_cache        class:   ufs
  hits                0

module: unix                instance: 0
name:   dnlcstats          class:   misc
  hits                34214237

module: unix                instance: 0

```

```

name:      ncstats                      class:      misc
hits                      34214009

module: unix                      instance: 0
name:      system_taskq            class:      taskq_d
hits                      4924

module: unix                      instance: 10
name:      ds_log_taskq            class:      taskq_d
hits                      0

module: unix                      instance: 11
name:      ds_taskq_1              class:      taskq_d
hits                      84858

module: unix                      instance: 12
name:      ds_taskq_0              class:      taskq_d
hits                      0

...
module: zfs                      instance: 0
name:      zfetchstats            class:      misc
hits                      20978253

```

8. Enter Control -C in the window that is running the find commands to exit the while loop.

## Solution 2-3: Using the `truss` Utility

### Overview

In this practice, you use `truss` to find what facility a given tool is using to get its information.

### Task

Practice using the `truss` utility by performing the following steps:

1. Refer to the man page for the `truss` command:

```
root@host01:~# man truss
...
```

- a. What is the `-t` option for? *The `-t` option allows you to trace or exclude a specific system call.*
- b. What does the `-f` option mean? *The `-f` option follows all children created by the `fork()` or `vfork()` system call, and includes their signals, faults, and system calls in the trace output.*

2. Enter the following command:

```
root@host01:~# truss -ft open ps
16129: open("/var/ld/64/ld.config", O_RDONLY)      Err#2 ENOENT
16129: open("/lib/64/libc.so.1", O_RDONLY)        = 3
16129: open("/proc/self/auxv", O_RDONLY)          = 3
...
```

Based on the output, does it look like `ps` is based on `dtrace`, `procfs`, or `kstat`?

\_\_procfs\_\_

3. Enter the following command:

```
root@host01:~# truss -ft ioctl vmstat 1 1
6078:  ioctl(3, KSTAT_IOC_CHAIN_ID, 0x00000000)    = 2855
6078:  ioctl(3, KSTAT_IOC_READ, "kstat_headers")  Err#12 ENOMEM
6078:  ioctl(3, KSTAT_IOC_READ, "kstat_headers")  = 2855
6078:  ioctl(3, KSTAT_IOC_CHAIN_ID, 0x00000000)    = 2855
6078:  ioctl(3, KSTAT_IOC_READ, "cpu_info0")       = 2855
...
```

Based on the output, does it look like `vmstat` is based on `dtrace`, `procfs`, or `kstat`?

\_\_kstat\_\_

4. Try using `truss -ftopen`, `truss -ftioctl`, or just `truss` to see which facility the following commands are based on: `dtrace`, `procfs`, `kstat`, or none of these?

- `plockstat date` \_\_dtrace\_\_
- `sar 1 1` \_\_kstat\_\_
- `intrstat 2 1` \_\_dtrace\_\_
- `ptree $$` \_\_procfs\_\_
- `iostat -xzn 2 1` \_\_kstat\_\_
- `truss -texit date` \_\_procfs\_\_
- `uptime` \_\_None\_\_





# **Practices for Lesson 3: kstat Monitoring Tools**

## **Chapter 3**

## Practices for Lesson 3: Overview

---

### Practices Overview

In these practices, you will use the following:

- `vmstat`, `sar`, and `mpstat` commands
- `iostat` command
- `kstat` utilities

Solutions for each task in this practice are provided after the practice exercises are introduced, at the end of this guide.

## Practice 3-1: Using the `vmstat`, `mpstat`, and `sar` Commands

### Overview

In this practice, you use the `vmstat`, `sar`, and `mpstat` commands to determine what they report for different types of load.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Practice using the `vmstat`, `mpstat`, and `sar` commands by performing the following steps:

1. Run the script `/opt/ora/course_files/lab3/while1`.

This script creates an infinite loop that emulates a compute-bound program.

```
root@host01:~# /opt/ora/course_files/lab3/while1&
```

2. Run the `vmstat 3` command.

What columns reflect the activity added by the compute-bound process?

\_\_\_\_\_

3. Try the same test with `sar 3 20` instead of `vmstat`.

4. Run the `kstat -ps ncpus` command.

How many instances of your program would it take to saturate the CPUs on your system?

\_\_\_\_\_

Which column in `vmstat` would indicate saturation? \_\_\_\_\_

5. Test the CPU saturation point by running multiple instances of the `while1` script.

6. Try the same test with `sar -q`.

What do the `runq-sz` and `%runqocc` columns represent?

\_\_\_\_\_

7. Kill all the `while1` compute-bound processes.

8. Run the `mpstat 5` command.

9. Watch the `usr`, `sys`, and `idl` columns and note their typical values.

`usr` - \_\_\_\_\_

`sys` - \_\_\_\_\_

`idl` - \_\_\_\_\_

10. Also watch and note the values in the `csw`, `icsw`, and `migr` columns.

`csw` - \_\_\_\_\_

`icsw` - \_\_\_\_\_

`migr` - \_\_\_\_\_

Remember that `csw` is the total number of context switches on a processor and `icsw` is the number of involuntary context switches on a processor.

11. In a separate window, run one instance of your compute-bound program (`while1`).  
Observe the window that is running `mpstat 5`.  
Which columns were affected? \_\_\_\_\_  
Was the process running on only one CPU or did it seem to migrate from CPU to CPU?  
\_\_\_\_\_  
What will happen if you drive the CPUs to saturation?  
Will migrations (`migr`) jump up? \_\_\_\_\_  
What will happen to the number of involuntary context switches (`icsw`) as the number of compute-bound processes exceeds the number of CPUs on the system?  
\_\_\_\_\_  
12. Test your assumptions by running additional compute-bound processes (`while1`).  
Did you notice any other values start to go up? \_\_\_\_\_  
13. Kill all the compute-bound programs.  
14. Try the same test by using `mpstat`. Enter:  

`root@host01:~# dd if=/dev/zero of=/dev/null &`

  
What are the values for `usr`, `sys`, and `idl` with this load?  
\_\_\_\_\_  
Use more instances of the `dd` command to drive the CPUs to saturation.  
What other columns were affected because of this load?  
\_\_\_\_\_  
15. Kill the `dd` processes by typing:

## Practice 3-2: Using the `iostat` Command

---

### Overview

In this practice, you use the `iostat` command and determine what it reports under load.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Practice using the `iostat` command by performing the following steps:

1. In one window, run:

```
root@host01:~# iostat -xcnCXtdz 5
```

2. In another window, check the `zpool` status.

Note the device name of the disk that is currently used for the `rpool` ZFS pool.

3. Use `dd` to copy the contents of the `rpool` disk to `/dev/null`.
4. In `iostat`, what typical values do you observe in the following columns for device `c3t0d0`?

- `r/s` \_\_\_\_\_
- `kr/s` \_\_\_\_\_
- `%w` \_\_\_\_\_
- `%b` \_\_\_\_\_

What will happen if you started a second instance of this `dd` command?

5. Run a second instance of the `dd` command.  
Is it as you predicted? \_\_\_\_\_
6. Terminate the `iostat` command by entering `control-c`.
7. Run the `mpstat 5` command.

Observe the `mpstat` command output.

What columns indicate an increase in load?

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

8. Terminate the `mpstat` command by entering `control-c`.
9. Kill the `dd` processes by typing:

## Practice 3-3: Using the `kstat` Utilities

---

### Overview

In this practice, you learn more about the relationship between `kstat` and other tools based on the `kstat` counters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Practice using the `kstat` command by performing the following steps:

1. Run the `vmstat -p 3 3` command.
2. In a second window, run the `kstat -n vm` command.

What fields in the `kstat` output correspond to the following columns in `vmstat -p` output?

re - \_\_\_\_\_

sr - \_\_\_\_\_

epi - \_\_\_\_\_

epf - \_\_\_\_\_

api - \_\_\_\_\_

apo - \_\_\_\_\_

apf - \_\_\_\_\_

fpi - \_\_\_\_\_

fpo - \_\_\_\_\_

You can do a similar comparison between the fields listed in `kstat -n sys` and commands such as `sar -c`, `sar -u`, and `mpstat`.

3. Run the `kstat -p ::vm:pgpgin 5` command.

**Note:** By using scripting or the `kstat` library (`lkstat`) and programming, you can generate your own custom tools.

## Solution 3-1: Using the `vmstat`, `mpstat`, and `sar` Commands

### Overview

In this practice, you use the `vmstat`, `sar`, and `mpstat` commands to determine what they report for different types of load.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Practice using the `vmstat`, `mpstat`, and `sar` commands by performing the following steps:

1. Run the script `/opt/ora/course_files/lab3/while1`.

This script creates an infinite loop that emulates a compute-bound program.

```
root@host01:~# /opt/ora/course_files/lab3/while1&
```

2. Run the `vmstat 3` command.

```
root@solaris-11-demo:~# vmstat 3
```

kthr			memory		page				disk				faults		cpu						
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	lf	s2	s3	s4	in	sy	cs	us	sy	id
0	0	0	4912832	4017104	3	0	0	0	0	0	0	0	0	0	0	707	113	365	0	0	100
0	0	0	3743064	2839480	3	7	0	0	0	0	0	0	0	0	0	773	138	419	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	752	98	416	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	758	103	430	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	758	90	414	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	765	100	417	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	764	116	422	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	763	90	413	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	776	100	422	4	0	96
0	0	0	3742808	2839224	0	0	0	0	0	0	0	0	0	0	0	766	90	416	4	0	96

What columns reflect the activity added by the compute-bound process?

us

id

3. Try the same test with `sar 3 20` instead of `vmstat`.

```
root@host01:~# sar 3 20
```

```
SunOS solaris-11-demo 5.11 11.0 sun4v      05/11/2000
```

14:06:03	%usr	%sys	%wio	%idle
14:06:06	4	1	0	95
14:06:09	4	0	0	96
14:06:12	4	0	0	95
14:06:15	4	0	0	96
...				

4. Run the `kstat -ps ncpus` command.

```
root@host01:~# kstat -ps ncpus
pg:1:pg:ncpus      4
pg:2:pg:ncpus     32
pg:3:pg:ncpus     32
pg:4:pg:ncpus      4
pg:5:pg:ncpus      4
pg:6:pg:ncpus      4
pg:7:pg:ncpus      4
pg:8:pg:ncpus      4
pg_hw_perf:1:Integer_Pipeline:ncpus      4
pg_hw_perf:2:Floating_Point_Unit:ncpus   32
pg_hw_perf:3:Data_Pipe_to_memory:ncpus   32
pg_hw_perf:4:Integer_Pipeline:ncpus      4
pg_hw_perf:5:Integer_Pipeline:ncpus      4
pg_hw_perf:6:Integer_Pipeline:ncpus      4
pg_hw_perf:7:Integer_Pipeline:ncpus      4
pg_hw_perf:8:Integer_Pipeline:ncpus      4
unix:0:pset:ncpus      32
unix:0:system_misc:ncpus      32
```

How many instances of your program would it take to saturate the CPUs on your system?

\_\_\_\_\_32\_\_\_\_\_

Which column in `vmstat` would indicate saturation? \_\_\_\_\_r\_\_\_\_\_

5. Test the CPU saturation point by running multiple instances of the `while1` script.
6. Try the same test with `sar -q`:

```
root@host01:~# sar -q 5
SunOS host15 5.11 11.1 sun4v      11/20/2012
08:59:34 runq-sz %runocc swpq-sz %swpocc
08:59:39      1.4      100      0.0      0
```

What do the `runq-sz` and `%runqocc` columns represent?

*Average queue length while occupied, and percent of time occupied*

7. Kill all the `while1` compute-bound processes.

```
root@host01:~# pkill while1
[1]    Terminated          ./while1
...
```



8. Run the `mpstat 5` command.

```

root@host01:~# mpstat 5
CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
 0      0    0    1   200    0    0    0    0    0    0    0    0    0    0 100
 1      0    0    0    1    0    0    0    0    0    0    0    0    0    0 100
 2      0    0    3   14    5    7    0    0    0    0    0    0    0    0 100
 3      0    0    9   66   20   56    0    0    0    0   20    0    0    0 100
 4      0    0    4   13    0    8    0    0    0    0    0    0    0    0 100
 5      0    0    3   69   31   67    0    0    0    0   32    0    0    0 100
 6      0    0    3   10    4    3    0    0    1    0    0    0    0    0 100
 7      0    0    6   25    2   19    0    0    0    0    1    0    0    0 100
 8      0    0    9   27    0   18    0    0    1    0    0    0    0    0 100
 9      0    0    3    8    0    8    0    0    0    0    0    0    0    0 100
10      0    0   10   30    0   21    0    0    1    0    8    0    0    0 100
11      0    0    5   17    0   10    0    0    0    0    2    0    0    0 100
12      0    0    4   15    0   10    0    0    0    0    1    0    0    0 100
13      0    0   32   13    0    8    0    0    0    0    0    0    0    0 100
14      0    0    8   23    0   15    0    0    1    0    0    0    0    0 100
15      0    0    4   15    1   10    0    0    0    0    0    0    0    0 100
16      0    0   11   34    1   22    0    1    1    0    0    0    0    0 100
17      0    0   10   29    1   20    0    1    0    0    9    0    0    0 100
18      0    0    4   34   10   29    0    0    0    0   11    0    0    0 100
19      0    0    9   27    1   18    0    0    0    0    1    0    0    0 100
20      0    0   10   30    0   20    0    0    0    0    0    0    0    0 100
21      0    0    6   19    0   14    0    0    0    0    1    0    1    0  99
22      0    0    6   20    0   13    0    0    1    0   15    0    0    0 100
23      0    0    2    8    0    5    0    0    0    0    0    0    0    0 100

```

9. Watch the `usr`, `sys`, and `idl` columns and note their typical values.

`usr` - 0

`sys` - 0

`idl` - 100

10. Also watch and note the values in the `csw`, `icsw`, and `migr` columns.

`csw` - Typically ranges from 0 to 70

`icsw` - 0

`migr` - 0

Remember that `csw` was the number of total context switches on that processor and `icsw` was the number of involuntary context switches on that processor.

11. In a separate window, run one instance of your compute-bound program (`while1`). Observe the window that is running `mpstat 5`.

```
...
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0    0    6   200    0    0    0    0    0    0    0    0    0    0 100
  1    0    0    0    1    0    0    0    0    0    0    0    0    0    0 100
  2    0    0    5   18    3    9    0    0    0    0    0    0    0    0 100
  3    0    0    3   49   20   45    0    0    0    0   20    0    0    0 100
  4    0    0    5   36   10   30    0    0    0    0   10    0    0    0 100
  5    0    0    6   20    0   12    0    0    0    0    0    0    0    0 100
...

```

Which columns were affected? *idl, usr, and csw*

Was the process running on only one CPU or did it seem to migrate from CPU to CPU?

*Process is running on one CPU.*

What will happen if you drive the CPUs to saturation?

*The idl column will go to 0. The usr column will go to 100. Involuntary context switches (icsw) will occur.*

Will migrations (`migr`) jump up? *Yes*

What will happen to the number of involuntary context switches (`icsw`) as the number of compute-bound processes exceeds the number of CPUs on the system? *Involuntary context switches (icsw) will increase.*

12. Test your assumptions by running additional compute-bound processes (`while1`).

Did you notice any other values start to go up?

*The migr and smtx columns increase.*

13. Kill all the compute-bound programs.

```
root@host01:~# pkill while1
[1]    Terminated                  ./while1
...

```

14. Try the same test by using `mpstat`. Enter:

```
root@host01:~# dd if=/dev/zero of=/dev/null &
```

What are the values for `usr`, `sys`, and `idl` with this load?

0 0 100

Use more instances of the `dd` command to drive the CPUs to saturation.

What other columns went up because of this load?

*The sys column increased.*

15. Kill the `dd` processes by typing:

```
root@host01:~# pkill dd
[1]    Terminated                  dd if=/dev/zero of=/dev/null
...

```

## Solution 3-2: Using the `iostat` Command

### Overview

In this practice, you use the `iostat` command and determine what it reports under load.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Practice using the `iostat` command by performing the following steps:

1. In one window, run:

```
root@host01:~# iostat -xcnCXtdz 5
Monday, November 19, 2012 10:36:19 AM MST

      cpu
    us sy wt id
      0  0  0 100

      extended device statistics

r/s    w/s    kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.1   0   0 c2
0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.1   0   0 c2t0d0
0.0    9.4    3.6    20.0  0.0  0.0    0.0    2.5   0   0 c3
0.0    9.4    3.6    20.0  0.0  0.0    0.0    2.5   0   0 c3t0d0
0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c3t1d0
...
```

2. In another window, check the `zpool` status.

```
root@host01:~# zpool status
...
```

Note the device name of the disk that is currently used for the `rpool` ZFS pool.

\_\_\_\_\_ c3t0d0s0 \_\_\_\_\_

3. Use `dd` to copy the contents of the `rpool` disk to `/dev/null`.

```
root@host01:~# dd if=/dev/rdisk/c3t0d0s0 of=/dev/null&
[1] 14065
```

4. In `iostat`, what typical values do you observe in the following columns for device `c3t0d0`?

- `r/s` \_\_\_\_\_ 8076.1 \_\_\_\_\_
- `kr/s` \_\_\_\_\_ 3891.6 \_\_\_\_\_
- `%w` \_\_\_\_\_ 6 \_\_\_\_\_
- `%b` \_\_\_\_\_ 43 \_\_\_\_\_

What will happen if you start a second instance of this `dd` command?

*The `%w` and `%b` will increase.*

5. Run a second instance of the `dd` command.

```
root@host01:~# dd if=/dev/rdisk/c3t0ds0 of=/dev/null&
[2] 14066
```

Is it as you predicted? \_\_\_\_\_ Yes \_\_\_\_\_

6. Terminate the `iostat` command by entering `control-c`.
7. Run the `mpstat 5` command.

```
root@host01:~# mpstat 5
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
...
```

Observe the `mpstat` command output.

What columns were driven up by this additional load?

\_\_\_\_\_ *xcal* \_\_\_\_\_  
 \_\_\_\_\_ *sys* \_\_\_\_\_  
 \_\_\_\_\_ *smtx* \_\_\_\_\_  
 \_\_\_\_\_ *csw* \_\_\_\_\_

8. Terminate the `mpstat` command by entering `control-c`.
9. Kill the `dd` processes by typing:

```
root@host01:~# pkill dd
[1] - Terminated      dd if=/dev/rdisk/c3t0ds0 of=/dev/null
[2]+  Terminated      dd if=/dev/rdisk/c3t0ds0 of=/dev/null
```

## Solution 3-3: Using the kstat Utilities

### Overview

In this practice, you learn more about the relationship between `kstat` and other tools based on the `kstat` counters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Practice using the `kstat` command by performing the following steps:

1. Run the `vmstat -p 3 3` command:

```
root@host01:~# vmstat -p 3 3
```

memory		page				executable				anonymous		filesystem			
swap	free	re	mf	fr	de	sr	epi	epo	epf	api	apo	apf	fpi	fpo	fpf
10553760	10612144	3	0	0	0	0	0	0	0	0	0	0	0	0	0
9156160	9293464	1	6	0	0	0	0	0	0	0	0	0	0	0	0
9156160	9293336	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2. In a second window, run the `kstat -n vm` command:

```
root@host01:~# kstat -n vm|more
module: cpu                      instance: 0
name:   vm                       class:   misc
       anonfree                  0
       anonpgin                  0
       anonpgout                  0
       as_fault                   18530
...
```

What fields in the `kstat` output correspond to the following columns in `vmstat -p` output?

re - pgrec

sr - scan

epi - execpgin

epf - execfree

api - anonpgin

apo - anonpgout

apf - anonfree

fpi - fspgin

fpo - fspgout

You can do a similar comparison between the fields listed in `kstat -n sys` and commands such as `sar -c`, `sar -u`, and `mpstat`.

3. Run the `kstat -p ::vm:pgpgin 5` command.

```
root@host01:~# kstat -p ::vm:pgpgin 5
cpu:0:vm:pgpgin    1382
cpu:1:vm:pgpgin    3493
cpu:2:vm:pgpgin    1468
cpu:3:vm:pgpgin     151
cpu:4:vm:pgpgin     630
cpu:5:vm:pgpgin    1665
cpu:6:vm:pgpgin    3374
cpu:7:vm:pgpgin    1377
cpu:8:vm:pgpgin    1197
cpu:9:vm:pgpgin    1018
...
cpu:30:vm:pgpgin   1386
cpu:31:vm:pgpgin   3786
...
```

**Note:** By using scripting or the `kstat` library (`lkstat`) and programming, you can generate your own custom tools.

# **Practices for Lesson 4: The procfs Monitoring Tools**

## **Chapter 4**

## Practices for Lesson 4: Overview

---

### Practices Overview

In this practice, you use the following:

- `ps` and `prstat` commands
- Oracle Solaris Studio `collect(1)` and `analyzer` utilities

Solutions for each task in this practice are provided after the practices exercises are introduced, at the end of this guide.



## Practice 4-1: Using the `ps` and `prstat` Commands

---

### Overview

In this practice, you use `ps`, `prstat`, the `proc` tools, and the `truss` utilities to observe process state information.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

1. Run the `ps -cle` command.

What scheduling class are `sched`, `pageout`, and `fsflush` in?

\_\_\_\_\_

What are their priorities?

\_\_\_\_\_

Which process, of the ones shown, has accumulated the most execution time?

\_\_\_\_\_

What is `init`'s scheduling class? \_\_\_\_\_

Who is `init`'s parent? \_\_\_\_\_

What does the `s` column show? \_\_\_\_\_

What state is `sched` in? \_\_\_\_\_

What do these states mean? \_\_\_\_\_

What state are most of the other processes in? \_\_\_\_\_

What does that mean? \_\_\_\_\_

Press the spacebar until the `ps` completes.

Which processes were in the `O` state? \_\_\_\_\_

2. Enter the `ps -cfe` command.

What does the `-f` option give you? \_\_\_\_\_

3. Now try entering `/usr/ucb/ps auxw`.

What does this give you? \_\_\_\_\_

On a process with a very long name, note its PID. \_\_\_\_\_

Then enter `ps pid#`.

How does the `ps pid#` output differ from the output of `/usr/ucb/ps auxww`?

\_\_\_\_\_

4. Enter the `ps -cLe` command.

What does the `-L` option give you?

\_\_\_\_\_

What are some of the processes that are apparently multithreaded?

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

5. Try running `ps -oabc`.

From the usage format it gives you, create a customized version of `ps`. Print out the PID, the parent's PID, the scheduling class, the priority, the resident set size, and the name of the command.

**Note:** Remember to enter an `-e` option at the end if you want to see all the processes.

6. Create your ideal `ps` command and test it. Write an alias or a shell script so that you can use it easily. For example, create a script that displays the non-global zone processes.
7. Enter the `prstat` command.

What order are the processes listed in by default? \_\_\_\_\_

8. Try running the compute-bound process (from Practice 3) in another window.

```
root@host01:~# /opt/ora/course_files/lab3/while1&
```

Notice that `prstat` tells you which CPU the process/thread is currently running on.

9. Stop (`ctrl-c`) the `prstat` command and enter the `prstat -mL` command.

What does the `-m` option give you? \_\_\_\_\_

What does the `-L` option give you? \_\_\_\_\_

Try running enough instances of your compute-bound process to make the `LAT` column go up.

What is the `LAT` column? \_\_\_\_\_

As the `LAT` column goes up, what happens to the `ICX` column?

What does `ICX` measure? \_\_\_\_\_

Use `pkill` to terminate all your compute-bound processes.

10. Use `pmap` to look at the address space of your current shell by entering: \_\_\_\_\_

How many segments are there? \_\_\_\_\_

What are the first three? \_\_\_\_\_

What is the last segment? \_\_\_\_\_

What are most of the remaining segments being used by?

Scroll to the bottom of the output. What is the total resident set size for the shell (`RSS`)?

What is the total amount for anonymous memory (`Anon`)?

Now try a larger process. Open the Firefox browser.

Use `pmap -x firefox_pid` to look at the Firefox mappings.

Scroll to the bottom of the output. What is the total resident set size for this process (`RSS`)?

What is the total amount for anonymous memory (`Anon`)?

Total resident set size and anonymous memory are important because they represent the bulk of the private mappings for this process.

11. Close the Firefox browser. In a window, enter the `mdb -k` command.
12. In another window, enter `pmap `pgrep mdb -k``.

What are the smallest and largest addresses in the Address column?

\_\_\_\_\_

13. Quit the `mdb` utility.
14. Run the `while1` process in the `lab4` directory (`/opt/ora/course_files/lab4`) in the background. Note its PID. \_\_\_\_\_
15. Run the `ps -cle|grep while1_PID` command.  
What is the state of the process (S column)? \_\_\_\_\_
16. Place the process in the STOP state. Check its state with `ps -cle`.
17. Use `prun` to continue the process.
18. Use `psig` to check how the process is handling signals.  
Is the process handling signals specially (that is, catching signals)? \_\_\_\_\_
19. Compare this to the signal information of a more complex process such as `mdb`.

Enter:

```
root@host01:~# mdb -k
```

In another window, enter:

```
root@host01:~# psig `pgrep mdb`
```

Is the `mdb` handling signals specially? **Hint:** Look for caught.

\_\_\_\_\_

20. In one window, `truss` the `while1` process.  
In another window, use `kill` to send a STOP signal.  
What appeared in the `truss` window? \_\_\_\_\_  
Use `kill` to send the process a CONT (continue) signal.  
What signal number is the continue signal? \_\_\_\_\_
21. Use `kill` to terminate the compute-bound process.  
What happened in the `truss` window? \_\_\_\_\_

## Practice 4-2: Using the `collect` Utility to Gather Information on a Running Process (Optional)

### Overview

This practice is recommended for those planning to use Oracle Solaris Studio. In this practice, you use the Oracle Solaris Studio `collect(1)` and `analyzer` utilities.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Perform the following steps:

1. Verify that Oracle Solaris Studio is installed on the system. If Oracle Solaris Studio is not installed on your system, install it in the `/opt/ora/software` directory:

```
root@host01:~# cd /opt/ora/software
root@host01:~# ls SolarisStudio12.3-solaris-sparc-bin
...
```

**Note:** If the `SolarisStudio12.3-solaris-sparc-bin` directory exists, Oracle Solaris Studio is installed on your system. If not, perform the following step:

```
root@host01:/opt/ora/software# bzip2 SolarisStudio12.3-solaris-sparc-bin.tar.bz2 \
| tar -xvf -
```

**Note:** This step takes approximately 15 minutes to complete.

```
root@host01:/opt/ora/software# cd ~
```

2. Exit to the `oracle` user shell and add Oracle Solaris Studio to your user profile.

```
root@host01:~# exit
root@host01:~$ vi .profile
PATH=$PATH:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin; export PATH
MANPATH=$MANPATH:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/man; export MANPATH
...
:wq!
root@host01:~$ . .profile
```

3. Initiate collection on a process. For example, perform collection on `zpool status`.

```
root@host01:~$ collect zpool status
```

What directory is created? \_\_\_\_\_

4. Use `er_print` to review the data gathered so far.

```
root@host01:~$ er_print -limit 10 -functions test.1.er
```

What function appears at the top? \_\_\_\_\_

What is the difference between exclusive and inclusive time in the output?

5. Open the `er_print` utility in interactive mode:

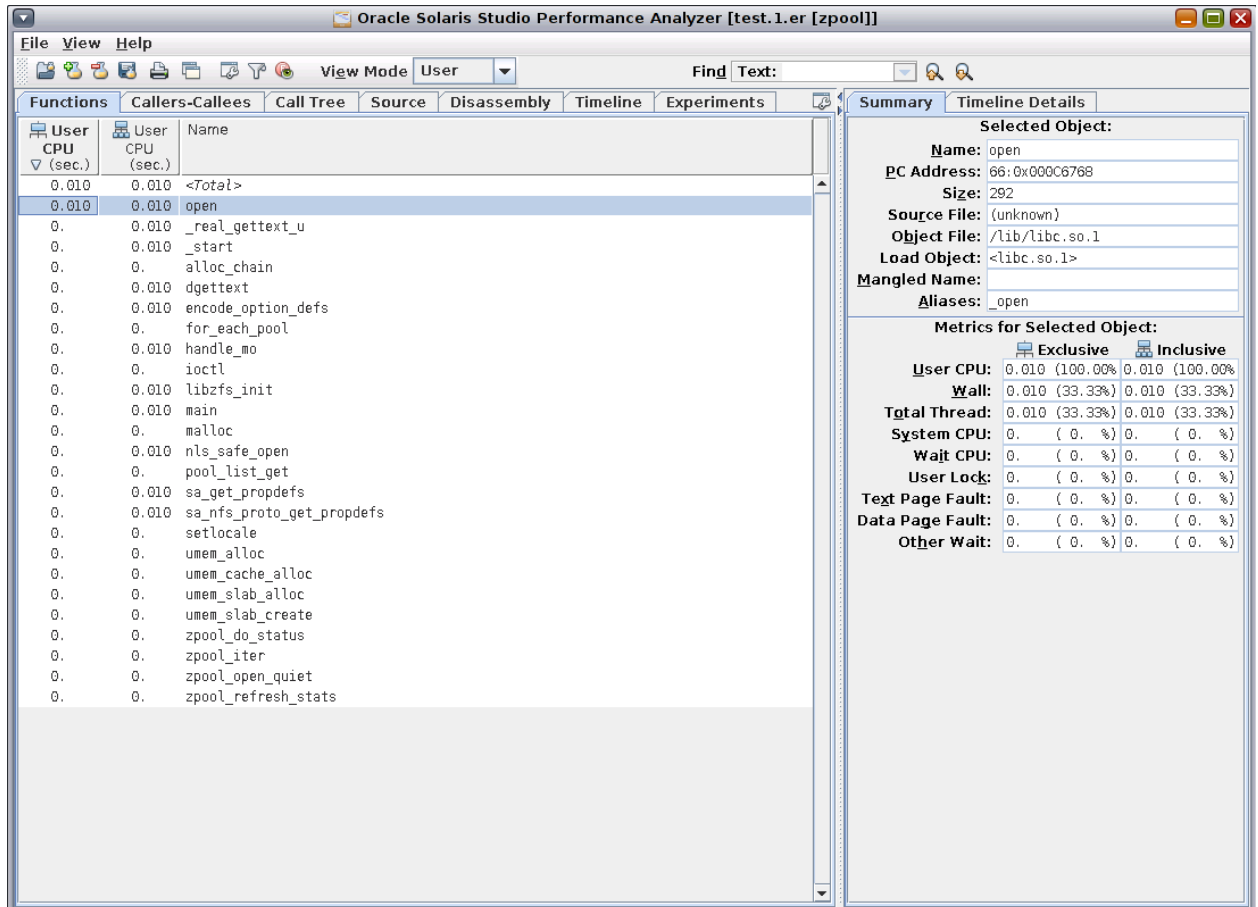
```
root@host01:~$ er_print test.1.er
```

6. Try the following subcommands:

- (er\_print) **functions**
- (er\_print) **version**
- (er\_print) **quit**

7. Run the analyzer `test.1.er&` to display the same performance data, as shown by using the `er_print` command, in a GUI format.

```
root@host01:~$ analyzer test.1.er&
```



Browse the available menus and data panels.

**Note:** To run the analyzer as `root`, you must set the `DISPLAY` variable before you run the analyzer. For example:

a. Get the current `DISPLAY` variable for the `oracle` user.

```
$ echo $DISPLAY
192.168.106.253:39.0
```

b. `su` to `root` using `su -` and enter the `root` password.

c. Set the `DISPLAY` variable for the `root` user (obtained from step a)

```
# export DISPLAY=192.168.106.253:39.0
```

8. Now run the analyzer.

```
root@host01:~$ analyzer test.1.er &
```

## Solution 4-1: Using the `ps` and `prstat` Commands

### Overview

In this practice, you use `ps`, `prstat`, the `proc` tools, and the `truss` utilities to observe process state information.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

1. Run the `ps -cle` command.

```
root@host01:~# ps -cle | more
 F S      UID      PID      PPID     CLS PRI      ADDR      SZ      WCHAN  TTY      TIME CMD
 1 T        0        0        0     SYS  96        ?        0           ?       0:12 sched
 1 S        0        5        0     SDC  99        ?        0           ? ?     9:04 zpool-rp
 1 S        0        6        0     SDC  99        ?        0           ? ?     0:12 kmem_tas
 0 S        0        1        0      TS  59        ?    399           ? ?     0:02 init
 1 S        0        2        0     SYS  98        ?        0           ? ?     0:00 pageout
 1 S        0        3        0     SYS  60        ?        0           ? ?    63:11 fsflush
...
```

What scheduling class are `sched`, `pageout`, and `fsflush` in?

\_\_\_\_\_ *system* \_\_\_\_\_

What are their priorities?

\_\_\_\_\_ *96* \_\_\_\_\_ *98* \_\_\_\_\_ *60* \_\_\_\_\_

Which process, of the ones shown, has accumulated the most execution time?

\_\_\_\_\_ *fsflush* \_\_\_\_\_

What is `init`'s scheduling class? \_\_\_\_\_ *time share* \_\_\_\_\_

Who is `init`'s parent? \_\_\_\_\_ *sched* \_\_\_\_\_

What does the `S` column show? \_\_\_\_\_ *process state* \_\_\_\_\_

What state is `sched` in? \_\_\_\_\_ *T* \_\_\_\_\_

What does that mean? \_\_\_\_\_ *The process is stopped.* \_\_\_\_\_

What state are most of the other processes in? \_\_\_\_\_ *S* \_\_\_\_\_

What does that mean? \_\_\_\_\_ *Sleeping: The process is waiting for an event to complete.* \_\_\_\_\_

Press the spacebar until the `ps` completes.

Which processes were in the `O` state? \_\_\_\_\_ *ps* \_\_\_\_\_

2. Enter the `ps -cfe` command.

```
root@host01:~# ps -cfe | more
 F S      UID      PID      PPID     CLS PRI      ADDR      SZ      WCHAN  STIME TTY      TIME CMD
 1 T    root        0        0     SYS  96        ?        0           Nov 14 ?     0:12 sched
 1 S    root        5        0     SDC  99        ?        0           ?    Nov 14 ?     9:05 zpool-rpool
 1 S    root        6        0     SDC  99        ?        0           ?    Nov 14 ?     0:12 kmem_task
 0 S    root        1        0      TS  59        ?    399           ?    Nov 14 ?     0:02 /usr/sbin/init
```

```

1 S root 2 0 SYS 98 ? 0 ? Nov 14 ? 0:00 pageout
1 S root 3 0 SYS 60 ? 0 ? Nov 14 ? 63:14 fsflush
...

```

What does the -f option give you? \_\_\_\_\_ *Generates a full listing*

3. Now try entering `/usr/ucb/ps auxw`:

```

root@host01:~# /usr/ucb/ps auxw
USER      PID %CPU %MEM    SZ   RSS TT          S      START   TIME COMMAND
...
root       5  0.0  0.0      0      0 ?          S    Nov 14   9:06 zpool-rpool
root    16176  0.0  0.1 9928 7000 pts/2      S 10:41:31   0:01 mpstat 5
root       0  0.0  0.0      0      0 ?          T    Nov 14   0:12 sched
root       1  0.0  0.0 3192 1640 ?          S    Nov 14   0:01 /usr/sbin/init
root       2  0.0  0.0      0      0 ?          S    Nov 14   0:00 pageout
...
root   15914  0.0  0.11477610760 ?          S 09:33:10   0:00 /usr/lib/bonobo-
activation-server --ac-activate --ior-output-fd=21
...

```

What does this give you? *\_ps aux includes the full command line (path and parameters).*\_

On a process with a very long name, note its PID. 15914

Then enter `pargs pid#`.

```

root@host01:~# pargs 15914
15914:  /usr/lib/bonobo-activation-server --ac-activate --ior-output-
fd=21
argv[0]: /usr/lib/bonobo-activation-server
argv[1]: <NULL>
argv[2]: <NULL>

```

How does that differ from the output of `/usr/ucb/ps auxww`?

*\_\_pargs provides the same information in a different format. It presents each argument as a separate line, which could be easier to read for some people.* \_\_

4. Enter the `ps -cLe` command.

```

root@host01:~# ps -cLe | more
  PID   LWP  CLS PRI  TTY          LTIME  CMD
    0     1  SYS  96  ?           0:12  sched
    5     1  SDC  99  ?           0:00  zpool-rp
    5     2  SDC  99  ?           0:01  zpool-rp
    5     3  SDC  99  ?           0:00  zpool-rp
    5     4  SDC  99  ?           0:00  zpool-rp
    5     5  SDC  99  ?           0:00  zpool-rp
    5     6  SDC  99  ?           0:00  zpool-rp
    5     7  SDC  99  ?           0:00  zpool-rp
...

```

What does the -L option give you?

*\_\_Prints information about each lightweight process (lwp) in each selected process* \_\_

What are some of the processes that are apparently multithreaded?

\_\_\_\_\_ *zpool-rp* \_\_\_\_\_  
 \_\_\_\_\_ *vmtasks* \_\_\_\_\_  
 \_\_\_\_\_ *svc.star* \_\_\_\_\_

5. Try entering `ps -oabc`:

```
root@host01:~# ps -oabc
ps -oabc
ps: unknown output format: -o abc
usage: ps [ -aAdefHlhcjLPyZ ] [ -o format ] [ -t termlist ]
      [ -u userlist ] [ -U userlist ] [ -G grouplist ]
      [ -p proclist ] [ -g pgrplist ] [ -s sidlist ] [ -z zonelist ] [-h
lgrplist]
      'format' is one or more of:
      user ruser group rgroup uid ruid gid rgid pid ppid pgid sid taskid ctid
      pri opri pcpu pmem vsz rss osz nice class time etime stime zone zoneid
      f s c lwp nlwp psr tty addr wchan fname comm args projid project pset
lgrp
```

From the usage format it gives you, create a customized version of `ps`. Print out the PID, the parent's PID, the scheduling class, the priority, the resident set size, and the name of the command.

**Note:** Remember to enter an `-e` option at the end if you want to see all the processes.

6. Create your ideal `ps` command and test it. Write an alias or a shell script so that you can use it easily. For example, to create a script that displays the non-global zone processes, run:

```
root@host01:~# vi ps_zones
ps -o pid,zone,zoneid,fname -e|egrep -v global
:wq!
root@host01:~# chmod 755 ps_zones
root@host01:~# ./ps_zones
  PID      ZONE  ZONEID  COMMAND
 5236 database      2  kcfld
 5708      web      3  sendmail
 5509 storage      1  sysconfi
 5297 database      2  net-ipmg
 5610      web      3  nscd
 4586 storage      1  init
...
```

7. Enter the `prstat` command.

```
root@host01:~# prstat
  PID USERNAME  SIZE  RSS STATE  PRI NICE   TIME   CPU PROCESS/NLWP
 15966 root      140M  33M sleep  47   4   0:01:41  0.1% gnome-terminal/2
 15919 root      235M 138M sleep  47   4   0:01:32  0.0% java/43
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



```

  5 root      0K    0K sleep    99  -20   0:09:11  0.0% zpool-rpool/166
16222 root    11M 8328K cpu22    47   4    0:00:00  0.0% prstat/1
16176 root    10M 7256K sleep    47   4    0:00:03  0.0% mpstat/1
1631 root    9152K 6000K sleep    59   0    0:00:00  0.0% cron/1
1571 root     20M   14M sleep    59   0    0:00:43  0.0% nsd/33
 142 root    2000K 1152K sleep    59   0    0:00:01  0.0% utmpd/1
1896 root     12M 7680K sleep    59   0    0:00:00  0.0% gdm-binary/2
1513 root     14M   10M sleep    59   0    0:00:45  0.0% picld/8
 102 root    9952K 1008K sleep    59   0    0:00:31  0.0% in.mpathd/1
  78 netadm   5072K 3352K sleep    59   0    0:00:07  0.0% ipmgmtd/6
 254 root     10M 2272K sleep    59   0    0:00:00  0.0% syseventd/17
  69 root     19M 6008K sleep    59   0    0:00:12  0.0% dlmgmtd/14
1989 root      0K    0K sleep    60  -    0:00:00  0.0% nfs4cbd_kproc/2
 116 root    2336K 1760K sleep    59   0    0:00:00  0.0% pfexecd/3
  44 netcfg   4400K 3040K sleep    59   0    0:00:12  0.0% netcfgd/5
 613 root     13M 9376K sleep    59   0    0:00:14  0.0% devfsadm/7
  13 root     35M  29M sleep    59   0    0:07:52  0.0% svc.configd/32
  11 root     21M  16M sleep    59   0    0:00:30  0.0% svc.startd/12
1783 root     15M 8664K sleep    59   0    0:00:00  0.0% cupsd/1
Total: 172 processes, 1081 lwps, load averages: 0.07, 0.06, 0.08

```

What order are the processes listed in by default? *\_If you do not specify an option, prstat examines all processes and reports statistics sorted by CPU usage.\_*

8. Try running the compute-bound process (from Practice 3) in another window.

```
root@host01:~# /opt/ora/course_files/lab3/while1&
```

Notice that prstat tells you what CPU the process/thread is currently running on.

9. Stop (ctrl-c) the prstat command and enter the prstat -mL command.

```

root@host01:~# prstat -mL

  PID USERNAME  USR  SYS  TRP  TFL  DFL  LCK  SLP  LAT  VCX  ICX  SCL  SIG  PROCESS/LWPID
16573 root        100  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0  25   0   0  while1/1
15966 root         1.1  0.3  0.0  0.0  0.0  0.0  99  0.0  62   1 794   0  0  gnome-termin/1
16223 root         0.3  0.6  0.0  0.0  0.0  0.0  99  0.0  22   0 974   0  0  prstat/1
15919 root         0.2  0.1  0.0  0.0  0.0 100  0.0  0.0 101   0 202   0  0  java/40
15919 root         0.2  0.1  0.0  0.0  0.0 100  0.0  0.0 202   0 303   0  0  java/41
...
Total: 171 processes, 1081 lwps, load averages: 0.05, 0.05, 0.08

```

What does the -m option give you? *\_Reports microstate process accounting information\_*

What does the -L option give you? *\_Reports statistics for each lightweight process (LWP)\_*

Try running enough instances of your compute-bound process to make the LAT column go up.

What is the LAT column? *\_The percentage of time the process has spent waiting for CPUs\_*

As the LAT column goes up, what happens to the ICX column?  
*\_It increases.\_*

What does ICX measure? *\_The number of involuntary context switches\_*

Use pkill to terminate all the compute-bound processes.

10. Use `pmap` to look at the address space of your current shell by entering:

```
root@host01:~# pmap -x $$
15968:  /usr/bin/bash
  Address   Kbytes    RSS    Anon  Locked Mode   Mapped File
00010000      896     896      -      -  r-x--  bash
000F0000       24      24       8      -  rwx--  bash
00100000      256     256     256      -  rw---  [ heap ]
F0CC0000     1464    1208      -      -  r-x--  libc.so.1
F0E3E000       48      48      24      -  rwx--  libc.so.1
FEBE0000       24      24      -      -  r-x--  libgen.so.1
FEBF6000        8       8      -      -  rwx--  libgen.so.1
FEC00000     6752    4232      -      -  r-x--  en_US.UTF-8.so.3
FF2A8000        8       8      -      -  rwx--  en_US.UTF-8.so.3
FF2C0000      192     192      -      -  r-x--  libcurses.so.1
FF2F0000       16      16      -      -  rwx--  libcurses.so.1
FF2F4000       16       8      -      -  rwx--  libcurses.so.1
FF300000       64      64      -      -  rwx--  [ anon ]
FF320000        8       8      -      -  rwx--  [ anon ]
FF330000        8       8       8      -  rw---  [ anon ]
FF340000       64      64      -      -  r-x--  methods_unicode.so.3
FF35E000        8       8      -      -  rwx--  methods_unicode.so.3
FF370000       24      24       8      -  rwx--  [ anon ]
FF380000        8       8       8      -  rw---  [ anon ]
FF390000        8       8      -      -  rw---  [ anon ]
FF3A0000      240     240      -      -  r-x--  ld.so.1
FF3EC000       16      16      16      -  rwx--  ld.so.1
FFBF0000       64      64      64      -  rw---  [ stack ]
-----
total Kb   10216    7432    392      -
```

How many segments are there? Count the number of start addresses (Address column).

What are the first three? 00010000 (bash) 000F0000 (bash) 00100000 (heap)

What is the last segment? FFBF0000 (stack)

What are most of the remaining segments being used by?

Library routines and anonymous memory

Scroll to the bottom of the output. What is the total resident set size for the shell (RSS)?

7432 KB

What is the total amount for anonymous memory (Anon)?

392 KB

Now try a larger process. Open the Firefox browser.

Use `pmap -x firefox_pid` to look at the Firefox mappings.

```
root@host01:~# ps -e|grep firefox
16252 ?          0:17 firefox
root@host01:~# pmap -x 16252
```

```
16252: /usr/bin/firefox
```

Address	Kbytes	RSS	Anon	Locked	Mode	Mapped File
00010000	224	224	-	-	r-x--	firefox
00056000	8	8	-	-	rw-x--	firefox
EA400000	4096	-	-	-	rw---	[ anon ]
EA9FA000	8	8	8	-	rw--R	[ anon ]
EAAFA000	8	8	8	-	rw--R	[ anon ]
EABFA000	8	8	8	-	rw--R	[ stack tid=23 ]
EAC00000	1024	768	768	-	rw---	[ anon ]
EAEFA000	8	8	8	-	rw--R	[ stack tid=22 ]
EAFFA000	8	8	8	-	rw--R	[ stack tid=21 ]
...						

Scroll to the bottom of the output. What is the total resident set size for this process (RSS)?

\_\_140496 KB\_\_

What is the total amount for anonymous memory (Anon)?

\_\_37304 KB\_\_

Total resident set size and anonymous memory are important because they represent the bulk of the private mappings for this process.

11. Close the Firefox browser. In a window, enter the `mdb -k` command.

```
root@host01:~# mdb -k
Loading modules: [ unix genunix specfs dtrace zfs scsi_vhci
sd mpt mac px ldc ip hook neti arp usba kssl fctl sockfs
random mdesc idm cpc crypto fcip fcp ufs logindmux nsmb ptm
sppp nfs lofs ipc ]
>
```

12. In another window, enter:

```
root@host01:~# pmap `pgrep mdb`
```

What are the smallest and largest addresses in the Address column?

\_\_0000000100000000\_\_ \_\_ FFFFE2A15520000\_\_

13. Quit the `mdb` utility.

```
> $q
```

14. Run the `while1` process in the `lab4` directory (`/opt/ora/course_files/lab4`) in the background. Note its PID. \_\_16677\_\_

15. Run the `ps -cle|grep while1_PID` command.

```
root@host01:~# ps -cle|grep 16677
0 0 0 16677 16629 TS 0 ? 236 pts/1 0:32 while1
```

What is the state of the process (s column)? \_\_o (Process is running on a processor.)\_\_

16. Place the process in the STOP state. Check its state with `ps -cle`.

```
root@host01:~# kill -STOP 16677
root@host01:~# ps -cle|grep 16677
0 T 0 16677 16629 TS 0 ? 236 pts/1 0:32 while1
```

17. Use `prun` to continue the process.

```
root@host01:~# prun 16677
```

18. Use `psig` to check how the process is handling signals.

```
root@host01:~# psig 16677
16677: /opt/ora/course_files/lab4/while1
HUP      default
INT      default
QUIT     default
ILL      default
...
```

Is the process handling signals specially (that is, “catching” signals)? \_\_No\_\_

19. Compare this to the signal information of a more complex process such as `mdb`.

Enter:

```
root@host01:~# mdb -k
```

...

In another window, enter:

```
root@host01:~# psig `pgrep mdb`
```

Is the `mdb` handling signals specially? **Hint:** Look for caught. \_\_Yes\_\_

Quit `mdb`.

20. In one window, `truss` the `while1` process.

```
root@host01:~# truss -p 16677
```

In another window, use `kill` to send it a `STOP` signal.

```
root@host01:~# kill -STOP 16677
```

What appeared in the `truss` window? \_\_Stopped by signal #23, SIGSTOP\_\_

Use `kill` to send the process a `CONT` (continue) signal.

```
root@host01:~# kill -CONT 16677
```

What signal number is the continue signal? \_\_Received signal #25, SIGCONT [default]\_\_

*siginfo: SIGCONT pid=16629 uid=0\_\_*

21. Use `kill` to terminate the compute-bound process.

```
root@host01:~# kill 16677
```

What happened in the `truss` window? \_\_Received signal #15, SIGTERM [default]\_\_

*siginfo: SIGTERM pid=16629 uid=0\_\_*

## Solution 4-2: Using the `collect` Utility to Gather Information on a Running Process (Optional)

### Overview

This practice is recommended for those planning to use Oracle Solaris Studio. In this practice, you use the Oracle Solaris Studio `collect(1)` and `analyzer` utilities.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Perform the following steps:

1. Verify that Oracle Solaris Studio is installed on the system. If Oracle Solaris Studio is not installed on your system, install it in the `/opt/ora/software` directory:

```
root@host01:~# cd /opt/ora/software
root@host01:~# ls SolarisStudio12.3-solaris-sparc-bin
...
```

**Note:** If the `SolarisStudio12.3-solaris-sparc-bin` directory exists, Oracle Solaris Studio is installed on your system. If not, perform the following step:

```
root@host01:/opt/ora/software# bzip2 SolarisStudio12.3-solaris-sparc-bin.tar.bz2 \
| tar -xvf -
```

**Note:** This step takes approximately 15 minutes to complete.

```
root@host01:/opt/ora/software# cd ~
```

2. Exit to the `oracle` user shell and add Oracle Solaris Studio to your user profile.

```
root@host01:~# exit
root@host01:~$ vi .profile
PATH=$PATH:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin; export PATH
MANPATH=/usr/share/man:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/man; export MANPATH
...
:wq!
root@host01:~$ . .profile
```

3. Initiate collection on a process. For example, perform collection on `zpool`.

```
root@host01:~$ collect zpool status
```

What directory is created? \_\_\_\_\_

4. Use `er_print` to review the data gathered so far.

```
root@host01:~$ er_print -limit 10 -functions test.1.er
```

What function appears at the top? `_open__`

What is the difference between exclusive and inclusive time in the output?

- *Exclusive time is the amount of execution time that passed while within that function, excluding the time spent in functions called from that function.*
- *Inclusive time is the amount of execution time that passed while within that function, including the time spent in functions called from that function.*

5. Open the `er_print` utility in interactive mode:

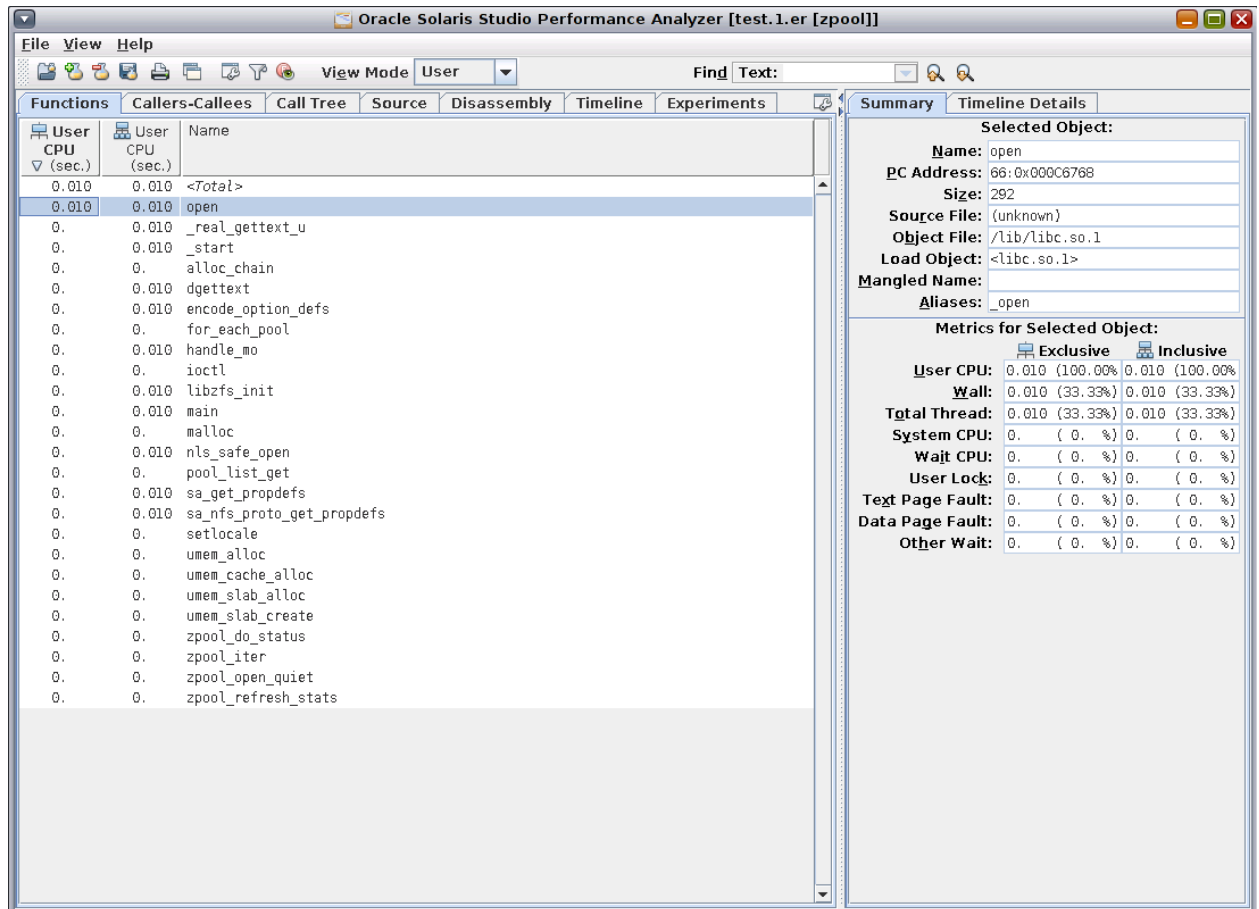
```
root@host01:~$ er_print test.1.er
```

6. Try the following subcommands:

- (er\_print) **functions**
- (er\_print) **version**
- (er\_print) **quit**

7. Use the analyzer utility to view the same data in a GUI.

```
root@host01:~$ analyzer test.1.er&
```



Browse the available menus and data panels.

**Note:** To run the analyzer as `root`, you must set the `DISPLAY` variable before you run the analyzer. For example:

a. Get the current `DISPLAY` variable for the `oracle` user.

```
root@host01:~$ echo $DISPLAY
192.168.106.253:39.0
```

b. `su` to `root` using `su -` and enter the `root` password.

c. Set the `DISPLAY` variable for the `root` user (obtained from step a)

```
root@host01:~# export DISPLAY=192.168.106.253:39.0
```

8. Now run the analyzer.

```
root@host01:~$ analyzer test.1.er &
```

# **Practices for Lesson 5: Introduction to DTrace**

## **Chapter 5**

## Practices for Lesson 5: Overview

---

### Practices Overview

In this practice, you practice working with DTrace one-liners and the DTrace Toolkit.

Solutions for each task in this practice are provided after the practice exercises are introduced, at the end of this guide.



## Practice 5-1: Using the DTrace One-Liners

### Overview

In this practice, you practice working with DTrace one-liners.

**Note:** The output from the DTrace commands shown in this task are examples. Your practice experience should be similar.

**Note:** When executing the DTrace one-liners in this task, let them run for several seconds to allow DTrace to collect a good data sample. Use **CTRL-C** to terminate DTrace.

**Note:** If you are not currently logged in to your server, log in now.

### Task:

Perform the following steps to practice using DTrace one-liners:

1. Create a list of DTrace providers.

```
root@host01:~# dtrace -l | perl -pe \
's/^.*?\S+\s+(\S+?) ([0-9]|\s).*\/\1/' | sort | uniq
```

2. Generate some activity on the system by writing zeros to /dev/null.

3. Determine the number of system calls that are being executed by the CPUs.

```
root@host01:~# dtrace -n 'syscall::entry { @[probefunc] =
count(); }'
```

4. Determine the number of interrupts being handled by each CPU.

```
root@host01:~# dtrace -n 'sdt::interrupt-start { @num[cpu] =
count(); }'
```

5. Terminate the dd processes.

6. Generate some disk I/O activity using the following script:

```
root@host01:~# while [ 1 ]
> do
> tar cf /var/tmp/disk1.tar /opt/ora/course_files/*
> rm /var/tmp/disk1.tar
> done&
[1] 10288
```

7. Determine the write size distribution by process.

```
root@host01:~# dtrace -n 'sysinfo::writech { @dist[execname] =
quantize(arg0); }'
```

8. Determine the read size distribution by process.

```
root@host01:~# dtrace -n 'sysinfo::readch { @dist[execname] =
quantize(arg0); }'
```

9. Determine the physical I/O to ZFS pools, physical disks, disk volumes, and NFS.

```
root@host01:~# dtrace -n 'io:::start { @[pid, execname] =
sum(args[0]->b_bcount); }'
```

10. Determine the file I/O (measured in blocks), summarized per zone.

```
root@host01:~# dtrace -n 'io:::start { @size[zonename] =
quantize(args[0]->b_bcount); }'
```

11. Kill the disk I/O script.

```
root@host01:~# pkill bash
```

12. Generate some network activity by running the following command:

```
root@host01:~# ping -s 192.168.106.255 > /dev/null&
```

13. Determine the number of received packets by host address.

```
root@host01:~# dtrace -n 'ip:::receive { @[args[2]->ip_saddr] =  
count(); }'
```

14. Kill the ping process.

```
root@host01:~# pkill ping
```

## Practice 5-2: Using the DTrace Toolkit

### Overview

In this practice, you use the DTrace Toolkit.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Install the DTrace Toolkit.

```
root@host01:~# pkg search dtrace-toolkit
```

2. Add the DTrace Toolkit to your user profile.

```
root@host01:~# vi .profile
...
PATH=/usr/dtrace/DTT/Bin:$PATH
export PATH
MANPATH=/usr/dtrace/DTT/Man:/usr/share/man
export MANPATH
...
:wq!
root@host01:~# . .profile
```

3. In the `/usr/dtrace/DTT/Docs` directory, look at the file content.

What does the `dtruss` command do? \_\_\_\_\_

Look at the man page for `dtruss`.

Which option prints the elapsed time in microseconds? \_\_\_\_\_

Look at the actual program in `/usr/dtrace/DTT/Bin`. What is the actual DTrace program wrapped in? \_\_\_\_\_

Why? \_\_\_\_\_

4. How many programs are in the `Bin` directory? \_\_\_\_\_

5. Look at the `kill.d` program in the `/usr/dtrace/DTT/Bin` directory.

What does the `kill.d` script do?

\_\_\_\_\_

There is a way to add an `fbt` probe for the kernel routine `sigtoproc()`, which takes three arguments. The first argument, `args[0]`, is a pointer to a `proc_t` structure. The `p_user.u_psargs` member contains the name of the command the signal is being sent to. Using that, you could follow the pointer to the `proc` structure, to the `user` structure, which is in the `proc` structure, and to the field `u_psargs`, which contains the name of the command the signal is being sent to. So as an action statement in a probe for `fbt::sigtoproc:entry`, you could use something like the following to print the command the signal is being sent to:

```
printf(" %s\n", args[0]->p_user.u_psargs);
```

Or you can simply run `dtrace -lv -n 'fbt::sigtoproc: {}'` to obtain the information.  
You can run the following `mdb` command to print out the `proc_t` structure:

```
# echo "::-print -t proc_t" | mdb -k
```

6. Practice running some DTrace Toolkit scripts. View the man pages on the following scripts, and then run each script. Review the output.

- `dtruss`
- `dexplorer`
- `iosnoop`
- `iotop`
- `iopattern`
- `errinfo`
- `fddist`
- `vopstat`
- `threaded.d`
- `sar-c.d`
- `readdist.d`
- `writedist.d`
- `stacksize.d`
- `rwtop`
- `sampleproc`

## Solution 5-1: Using the DTrace One-Liners

---

### Overview

In this practice, you practice working with DTrace one-liners.

#### Note:

1. The output from the DTrace commands shown in this task are examples. Your practice experience should be similar.
2. When executing the DTrace one-liners in this task, let them run for several seconds to allow DTrace to collect a good data sample. Use **CTRL-C** to terminate DTrace.
3. If you are not currently logged in to your server, log in now.

### Task:

Perform the following steps to practice using DTrace one-liners:

1. Create a list of DTrace providers.

```
root@host01:~# dtrace -l | perl -pe \
's/^.*?\s+\s+(\s+?) ([0-9] |\s).*/\1/' | sort | uniq
dtrace
fbt
fpuinfo
fsinfo
io
ip
iscsi
kerberos
lockstat
mib
nfsv
perl
proc
profile
PROVIDER
python
sched
sdt
shadowfs
syscall
sysevent
sysinfo
tcp
udp
vminfo
Xserver
```

2. Generate some activity on the system by writing zeros to /dev/null.

```
root@host01:~# dd if=/dev/zero of=/dev/null&
[1] 1610
root@host01:~# dd if=/dev/zero of=/dev/null&
[2] 1611
root@host01:~# dd if=/dev/zero of=/dev/null&
[3] 1612
root@host01:~# dd if=/dev/zero of=/dev/null&
```

3. Determine the number of system calls that are being executed by the CPUs.

```
root@host01:~# dtrace -n 'syscall:::entry { @[probefunc] =
count(); }'
dtrace: description 'syscall:::entry ' matched 212 probes
^C
```

fstatat	1
lwp_kill	1
mmap	1
schedc	1
sigpending	1
setitimer	2
nanosleep	3
sysconfig	3
clock_gettime	4
setcontext	4
sigaction	4
writev	4
brk	6
pset	21
pollsys	36
gtime	45
lwp_sigmask	54
lwp_park	111
p_online	256
ioctl	8911
write	54720593
read	54720603

4. Determine the number of interrupts being handled by each CPU.

```
root@host01:~# dtrace -n 'sdt:::interrupt-start { @num[cpu] =
count(); }'
dtrace: description 'sdt:::interrupt-start ' matched 3 probes
^C
```

1	271
---	-----

```

2          1011
3          1013
0          3388
...

```

5. Terminate the dd processes.

```

root@host01:~# pkill dd
[1]    Terminated          dd if=/dev/zero of=/dev/null
[2]    Terminated          dd if=/dev/zero of=/dev/null
[3]    Terminated          dd if=/dev/zero of=/dev/null
[4]    Terminated          dd if=/dev/zero of=/dev/null

```

6. Generate some disk I/O activity by using the following script:

```

root@host01:~# while [ 1 ]
> do
> tar cf /var/tmp/disk1.tar /opt/ora/course_files/*
> rm /var/tmp/disk1.tar
> done&
[1] 10288

```

7. Determine the write size distribution by process.

```

root@host01:~# dtrace -n 'sysinfo:::writech { @dist[execname] =
quantize(arg0); }'
dtrace: description 'sysinfo:::writech ' matched 4 probes
^C
dtrace
      value  ----- Distribution ----- count
      0 |                                           0
      1 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 1
      2 |                                           0

...
tar
      value  ----- Distribution ----- count
    1024 |                                           0
    2048 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 10132
    4096 |                                           0

```

8. Determine the read size distribution by process.

```

root@host01:~# dtrace -n 'sysinfo:::readch { @dist[execname] =
quantize(arg0); }'
dtrace: description 'sysinfo:::readch ' matched 4 probes
^C
sshd
      value  ----- Distribution ----- count

```

1		0
2	@@@@@@@@@@@@@@@@@@@@@@	1
4		0
8		0
16		0
32	@@@@@@@@@@@@@@@@@@@@@@	1
64		0
...		
tar		
value	----- Distribution -----	count
-1		0
0	@@@@@@@@@@@@@@@@@@@@@@	27834
1		0
2		0
4		0
8		0
16		0
32		0
64		0
128		0
256	@@@@@@@@@@@@@@	9278
512		0

9. Determine the physical I/O to ZFS pools, physical disks, disk volumes, and NFS.

```
root@host01:~# dtrace -n 'io:::start { @[pid, execname] =
sum(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C
      5  zpool-rpool                1763840
...
```

10. Determine the file I/O (measured in blocks), summarized per zone.

```
root@host01:~# dtrace -n 'io:::start { @size[zonename] =
quantize(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C
global
      value  ----- Distribution -----  count
      256   |
      512   | @@@@@@@@@@@@@@
     1024   | @@@@@@@@@@@@@@
     2048   | @@@@@@@@@@@@@@
     4096   | @@@@@@@@
           64
           70
           69
           34
```



8192		2
16384		0
...		

11. Kill the disk I/O script.

```
root@host01:~# pkill bash
...
```

12. Generate some network activity by running the following command:

```
root@host01:~# ping -s 192.168.106.255 > /dev/null&
```

13. Determine the number of received packets by host address.

```
root@host01:~# dtrace -n 'ip:::receive { @[args[2]->ip_saddr] =
count(); }'
dtrace: description 'ip:::receive ' matched 5 probes
^C
192.168.106.221          2
192.168.106.167        23
192.168.106.154        61
192.168.106.77         122
...
```

14. Kill the ping process.

```
root@host01:~# pkill ping
...
```

## Solution 5-2: Using the DTrace Toolkit

### Overview

In this practice, you use the DTrace Toolkit.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Install the DTrace Toolkit.

```
root@host01:~# pkg search dtrace-toolkit
INDEX          ACTION VALUE                                PACKAGE
pkg.fmri       set      solaris/system/dtrace/dtrace-toolkit
pkg:/system/dtrace/dtrace-toolkit@0.99-0.175.1.0.0.24.2
...
root@host01:~# pkg install dtrace-toolkit
No updates necessary for this image.
Planning linked: 3/3 done
```

2. Add the DTrace Toolkit to your user profile.

```
root@host01:~# vi .profile
...
PATH=$PATH:/usr/dtrace/DTT/Bin
export PATH
MANPATH=/usr/dtrace/DTT/Man:/usr/share/man
export MANPATH
...
:wq!
root@host01:~# . .profile
```

3. In the `/usr/dtrace/DTT/Docs` directory, look at the file content.

What does the `dtruss` command do?

*dtruss prints details on process system calls. It is like a DTrace version of truss, and has been designed to be less intrusive than truss.*

Look at the man page for `dtruss`.

Which option prints the elapsed time in microseconds? `__-e__`

Look at the actual program in `/usr/dtrace/DTT/Bin`. What is the actual DTrace program wrapped in? `__born shell script__`

Why? `__This makes the command flexible (options) and compatible with the legacy version on the Solaris OS.__`

4. How many programs are in the `Bin` directory? `__Run the ls | wc -l command in the /usr/dtrace/DTT/Bin directory. __`

5. Look at the `kill.d` program in the `/usr/dtrace/DTT/Bin` directory.

What does the `kill.d` script do?

*kill.d prints details of process signals as they are sent, such as the PID source and destination and signal number and result.*

There is a way to add an fbt probe for the kernel routine `sigtoproc()`, which takes three arguments. The first argument, `args[0]`, is a pointer to a `proc_t` structure. The `p_user.u_psargs` member contains the name of the command the signal is being sent to. Using that, you could follow the pointer to the `proc` structure, to the `user` structure, which is in the `proc` structure, and to the field `u_psargs`, which contains the name of the command the signal is being sent to. So as an action statement in a probe for `fbt::sigtoproc:entry`, you could use something like the following to print the command the signal is being sent to:

```
printf(" %s\n", args[0]->p_user.u_psargs);
```

Or you can simply run `dtrace -lv -n 'fbt::sigtoproc: {}'` to obtain the information.

You can run the following `mdb` command to print out the `proc_t` structure:

```
# echo ":::print -t proc_t" | mdb -k
```

6. Practice running some DTrace Toolkit scripts. View the man pages on the following scripts, and then run each script. Review the output.

- `dtruss`
- `dexplorer`
- `iosnoop`
- `iotop`
- `iopattern`
- `errinfo`
- `fddist`
- `vopstat`
- `threaded.d`
- `sar-c.d`
- `readdist.d`
- `writedist.d`
- `stacksize.d`
- `rwtop`
- `sampleproc`



# **Practices for Lesson 6: Other Significant Tools**

## **Chapter 6**

## Practices for Lesson 6: Overview

---

### Practices Overview

In these practices, you perform the following:

- Manage system `swap` devices
- Use `cpustat` and `cputrack` to view the `cpc` counters
- Use `mdb` to view kernel parameters
- Use Oracle Solaris Studio `dbx` to test a common application
- Use the GUDS script to collect performance data

Solutions for each task in this practice are provided after the practice exercises are introduced, at the end of this guide.

## Practice 6-1: Managing System swap Devices

---

### Overview

In this practice, you interpret the output of the `swap` command.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Perform the following steps:

1. Execute the `swap -l` command and observe the output.
2. Determine the type of partition-based swap file.
  - a. The first number in the `dev` column is the major number of the device type; the second value is the minor device number. For example, if your output is:

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	4194288	4194288

- b. You can identify the driver as follows:

```
root@host01:~# grep "300$" /etc/name_to_major
zfs 300
```

3. Resize the existing swap device from 2 GB to 4 GB.
4. Add a 1 GB swap volume to the configuration.
5. Observe and interpret the `swap -s` output.
6. In a second terminal window, observe **swap utilization** by running the `vmstat` command.
7. In the first terminal window, remove the 1 GB swap device (`moreswap`).  
Did the `vmstat` swap field change? \_\_\_\_\_
8. Return the original swap device size to 2 GB. Observe the `vmstat` command as you run each of the following commands:

## Practice 6-2: Using `cpustat` to View the `cpc` Counters

---

### Overview

In this practice, you use `cpustat` to view the `cpc` counters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. To see what events can be monitored on your processor, run the `cpustat -h` command.
2. Find the event counter for the Instruction Cache references and the Instruction Cache Hit or Miss. For example, on the UltraSPARC T1, there is an event called `IC_miss`.
3. Using the `IC_miss` event name and the `cpustat` command, measure the instruction cache miss rate on your system.
4. Start the Firefox browser. Use the `cpustrack` command to monitor instruction cache misses for the `firefox` process.



## Practice 6-3: Using `mdb` to View Kernel Parameters

### Overview

In this practice, you use the `mdb` debugger to view the kernel parameters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Start the `mdb` debugger for the kernel.  
Notice the modules that are being loaded as `mdb` starts.
2. Run the `::dcmds` command.
3. A list of `dcmds` commands appears after you run the command. Notice that `mdb` has a built-in `More`, so the output does not scroll off the screen. To see the `dcmds` in a more logical grouping, enter `q` to get out of the previous `More`.  
Enter the `::dmods -l` command, and then quit the `More` query.

4. You can use the escape character “!” to execute a shell command. For example, run:

```
> !pwd
...
```

5. To search for a given command, you could also use the `!` character to pipe the output of an `mdb` command to a shell command such as `grep`. Try entering:

```
> ::dmods -l ! grep ps
...
```

In the output, you should see a reference to a `ps` `dcmd`.

6. To see how to use the `ps` command, enter:

```
> ::help ps
...
```

7. From the synopsis, you can see the basic syntax. Try entering the following command, and then quit:

```
> ::ps
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

This is very similar to the `/usr/bin/ps` command, except that it gives you certain columns by default that are useful while you are in the debugger.

For example, it gives you the hex address of the `proc` structure, under the column `ADDR`.

8. The `-f` option gives you the long version of the command name. Enter:

```
> ::ps -f
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

9. The `-t` gives you the pointer to the kernel thread structure of all the threads within the process, plus the state of each thread. Try entering the following command, and then quit:

```
> ::ps -t
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

10. The `-l` option gives you the pointer to the `klwp` structure of all the threads within the process, plus the `lwp id` (thread ID). Try entering the following command, and then quit:

```
> ::ps -l
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

11. Try using them both together:

```
> ::ps -lt
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

What extra information do you get from the `-z` option?

What extra information do you get from the `-T` and `-P` options?

12. You can capture your `mdb` session in a log file by using the `log dcmd`. Enter:

```
> ::help log
```

The `-e` option enables logging to the specified log file. The `-d` option disables the logging.

13. Enter the following command, and then quit:

```
> ::nm
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

14. This is `mdb`'s version of the `nm` (name list) command. If you know the variable of interest, you can use it to display only that entry. Enter:

```
> maxusers::nm
...
```

15. You can also see the type of data.

```
> maxusers::nm -f ctype
...
```

What is the size in bytes of the variable `maxusers`? \_\_\_\_\_

To display data in `mdb`, you can use the symbol that represents the address of the data followed by a `/` and a format.

16. To see a list of all formats supported by `mdb`, enter the following command, and then quit:

```
> ::formats
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

17. Try entering:

```
> maxusers/D  
...
```

What is the value of `maxusers` on your system? \_\_\_\_\_

Try the same technique to find the values of the following parameters that will be discussed in the class:

`reserved_procs` - \_\_\_\_\_

`lotsfree` - \_\_\_\_\_

`fastscan` - \_\_\_\_\_

`maxpgio` - \_\_\_\_\_

`maxphys` - \_\_\_\_\_

To exit `mdb`, enter:

```
> $q
```

## Practice 6-4: Using Oracle Solaris Studio dbx to Test a Common Application (Optional)

### Overview

This practice is recommended for those planning to use Oracle Solaris Studio. In this practice, you use the Oracle Solaris Studio dbx tool to test a common application.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Review the source code of the `runtimeException.c` file. Take note of the comments inserted in the code and the instructions at the bottom of the file.

```
root@host01:~# cat /opt/ora/course_files/lab6/runtimeException.c
<output omitted>
```

2. Compile the code as directed in the file.

The `-g` option enables debug information to promote mapping. The compilation is stored in a file named `a.out`. Note that a warning similar to the one displayed as follows can be safely ignored after compilation.

```
root@host01:~# cd /opt/ora/software/SolarisStudio12.3-solaris-
sparc-bin/solarisstudio12.3/bin
root@host01:~# ./cc -g \
  /opt/ora/course_files/lab6/runtimeException.c
root@host01:~# "runtimeException.c", line 18: warning: implicit
function declaration: printf
root@host01:~# ls -l a.out
-rwxr-xr-x  1 root  6864 Nov  8 08:35 a.out*
```

3. Run the compiled program and observe the core dump file.

```
root@host01:~# ./a.out
```

The program produces the following lines just before terminating:

```
1/10 = 0
1/9 = 0
1/8 = 0
1/7 = 0
1/6 = 0
1/5 = 0
1/4 = 0
1/3 = 0
1/2 = 0
1/1 = 1
Arithmetic Exception (core dumped)
```

Observe the core file and verify its source.

```
root@host01:~# ls -l core
```

```
-rw----- 1 root  root 2276174 Nov 8 08:37 core
root@host01:~# file core
core: ELF 32-bit MSB core file SPARC Version 1, from 'a.out'
```

4. Use the dbx tool to examine the core file. Print the stack of called functions.

```
root@host01:~# ./dbx a.out core
For information about new features see `help changes'
To remove this message, put `dbxenv suppress_startup_message 7.8'
in your .dbxrc
Reading a.out
core file header read successfully
Reading ld.so.1
Reading libc.so.1
program terminated by signal FPE (integer divide by zero)
Current function is div
    4                return x/y;
(dbx) where
=>[1] div(x = 1, y = 0), line 4 in "runtimeException.c"
[2] iter(c = 10), line 18 in "runtimeException.c"
[3] main(), line 27 in "runtimeException.c"
(dbx) exit
```

The core file points to the last instruction issued when the process faulted. With the debugging information in the executable, dbx can point to the line of code in question.

5. Use dbx to examine a core dump after the executable is stripped (the output of each of the following commands is omitted):

```
root@host01:~# file a.out
root@host01:~# ls -l a.out
root@host01:~# strip a.out
root@host01:~# ls -l a.out
root@host01:~# file a.out
root@host01:~# ./a.out
root@host01:~# ./dbx a.out core
```

Note that after stripping the executable, dbx can examine only the native compiled form of the failed process.

## Practice 6-5: Using the GUDS Script

### Overview

In this practice, you use the GUDS script to collect performance data.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Locate the GUDS zip file in the `/opt/ora/software` directory.

```
root@host01:~# cd /opt/ora/software
root@host01:/opt/ora/software# ls guds*
guds_3_0.zip
```

2. Unzip the GUDS script.

```
root@host01:/opt/ora/software# unzip guds_3_0.zip
Archive:  guds_3_0.zip
inflating: guds_3_0
```

3. Run the GUDS script. Use the values shown in the following example:

```
root@host01:/opt/ora/software# ./guds_3_0

...
Please enter your Service Request number.
The SR number should only consist of digits or of the pattern
[0-9]-[0-9]+.
SR #:1234

Enter the number of hours for the script to run.
If the number of hours given is zero, the script will
run for the specified number of iterations.
For a positive, non zero number of hours, the script
will run using the specified count, interval and wait
parameters until the given number of hours has been
met or exceeded.
Hours[0]: 0

Please enter the count option for commands that require it.
These are commands like vmstat and iostat which use an
interval and a count to repeat the output.
For example, using a count of 10 and interval of 30
will cause iostat, vmstat, mpstat, etc to output
10 sets of data, with 30 seconds between data sets.
If you have not been given a value for this number,
the default value will be 40.
```

Count[40]: 5

Please enter the interval option for commands that require it.  
These are commands like vmstat and iostat which use an interval and count to repeat the output.

If you have not been given a value for this number, the default value will be 5.

Interval[5]: 5

Please enter the number of total data sets for the script to capture. Each data set contains the output of one SET of a commands output.

For example, a single data set would contain an entire "ps -elf" output, and an entire vmstat output, where the vmstat output is made up of a variable number of data lines, as defined by "count" and "interval"

If you have not been given instructions for this number, the default value will be 5.

Iterations[5]: 5

Please enter the number of seconds to wait between collecting data sets.

To ensure correct data collection, any value entered will be added to the minimum wait time.

If you have not been given instructions for this number, the default value will be 0.

Wait[0]: 0

Do you want to run an extended set of commands?

If you have not been given instructions for this number, the default value will be 2

level 0 : nothing

level 1 : lockstat for contention and hold events

level 2 : L1 + trapstat, lockstat profiling, threadlist, TNF tracing (default)

level 3 : L2 + kmastat, kmausers, memstat

Enter the level : '0', '1', '2' or '3'.

Extended[2]: 3

Do you want to collect prstat cpu, rss, size and zones data? [n]:  
**y**

Do you want to collect configuration data? [y]: **y**

Please enter a one line statement describing the current capture environment. Some examples are "Performance issue is present" or "System is operating normally".

This statement will be added to the INFO file so the support engineer knows the state of the system associated with this data.

Description: **Test run.**

Enter the location of the output directory [/var/tmp/guds]: **<CR>**

Starting Data Gathering

Collecting performance data...

Begin Iteration #1

40/40

End Iteration #1

Waiting for processes to terminate, then sleeping 0 seconds before the next iteration.

--- All processes finished. ---

Begin Iteration #2

5/40

...

--- All processes finished. ---

Collecting System Information

22/22

Creating zip file...

Please send the data to your support engineer.

The data are stored in

/var/tmp/guds/1234/1234\_guds.004eebdb.server1-2012.12.19.11.46.zip

Instructions for uploading a file via a web browser.

Go to URL: <https://supportfiles.sun.com>

Under Step 1, select the file to send

Under Step 2, set the destination to: cores

Under Step 3, click on Upload

Instructions for uploading a file via ftp to supportfiles.sun.com.

```
# ftp supportfiles.sun.com
```

```
login: anonymous
```

```
passwd: your email address
```

```
ftp> cd /cores
```

```
ftp> binary
```

```
ftp> put 1234_guds.004eebdb.server1-2012.12.19.11.46.zip
```

```
ftp> bye
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Instructions for uploading a file through My Oracle Support.

NOTE: Only for files less than 2G in size.

Go to URL: <https://support.oracle.com>

If you are logging the Service Request you can upload the guds output in Step 4 - Upload Files.

If you have already logged the Service Request and need to add the guds output to it, from either the Dashboard or Service Request tab find and click on your Service Request.

Click the Upload button or select Upload from the Actions drop down menu. Browse for your guds .zip file and choose it, then click the Upload button.

Your service request will automatically be updated to alert the support engineer once the data is available. You may also update the service request in My Oracle Support yourself.

4. Change to the GUDS data collection directory and list the contents.

```
root@host01:/opt/ora/software# cd /var/tmp/guds/1234
root@host01:/var/tmp/guds/1234# ls -l
total 7452
-rw-rw-rw-  1 root      root      1907938 Dec 19 05:03
1234_guds.004eebdb.server1-2012.12.19.11.46.zip
drwxr-xr-x  3 root      root          113 Dec 19 05:03
guds.004eebdb.server1-2012.12.19.11.46
```

Note that the zip file is used for uploading your performance data to Oracle Support for analysis. The directory contains the performance data for local analysis.

5. List the contents of the directory containing the performance data files.

```
root@host01:/var/tmp/guds/1234# ls guds.004eebdb.server1-
2012.12.19.11.46 | more
adb.out
arcstat.out
df-gl.out
df-kl.out
dladm-show-aggr-L.out
dladm-show-aggr.out
dladm-show-linkprop.out
dladm-show-phys.out
dmesg.out
drv
dutmp.out
errors.log
```

```
ifconfig-a.out  
...
```

6. Review the various performance data files to get an idea of the type of information that is collected by the GUDS script.

## Solution 6-1: Managing System swap Devices

### Overview

In this practice, you interpret the output of the `swap` command.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Perform the following steps:

1. Execute the `swap -l` command and observe the output.

```
root@host01:~# swap -l
```

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	4194288	4194288

2. Determine the type of partition-based swap file.

- a. The first number in the `dev` column is the major number of the device type; the second value is the minor device number. For example, if your output is:

```
swapfile          dev    swaplo  blocks    free
/dev/zvol/dsk/rpool/swap  300,2      16  4194288  4194288
```

- b. You can identify the driver as follows:

```
root@host01:~# grep "300$" /etc/name_to_major
zfs 300
```

3. Resize the existing swap device from 2 GB to 4 GB.

```
root@host01:~# swap -d /dev/zvol/dsk/rpool/swap
root@host01:~# zfs volsize=4G rpool/swap
root@host01:~# swap -a /dev/zvol/dsk/rpool/swap
root@host01:~# swap -l
```

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	8388592	8388592

4. Add a 1 GB swap volume to the configuration.

```
root@host01:~# zfs create -V 1G rpool/moreswap
root@host01:~# swap -a /dev/zvol/dsk/rpool/moreswap
root@host01:~# swap -l
```

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	8388592	8388592
/dev/zvol/dsk/rpool/moreswap	300,3	16	2097136	2097136

5. Observe and interpret the `swap -s` output.

```
root@host01:~# swap -s
total: 598784k bytes allocated + 211976k reserved = 810760k used,
15631448k available
```

6. In a second terminal window, observe **swap utilization** by running the `vmstat` command.

```
root@host01:~# vmstat 5
```

kthr			memory		page								disk				faults		cpu			
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s2	s3	s4	--	in	sy	cs	us	sy	id	
0	0	0	14399864	14398480	0	1	0	0	0	0	0	2	0	0	0	665	164	319	0	0	100	
0	0	0	15631448	12618280	2	4	0	0	0	0	0	13	0	0	0	926	356	595	0	0	100	
0	0	0	15631448	12618072	0	0	0	0	0	0	0	21	0	0	0	949	307	623	0	0	100	
0	0	0	15631448	12618072	0	0	0	0	0	0	0	14	0	0	0	955	351	626	0	0	100	
0	0	0	15631448	12618072	0	0	0	0	0	0	0	14	0	0	0	914	344	612	0	0	100	
0	0	0	15631448	12618072	0	0	0	0	0	0	0	24	0	0	0	983	346	650	0	0	100	
...																						

7. In the first terminal window, remove the 1 GB swap device (`moreswap`).

```
root@host01:~# swap -d /dev/zvol/dsk/rpool/moreswap
```

Did the `vmstat` swap field change? \_\_\_Yes\_\_\_

8. Return the original swap device size to 2 GB. Observe the `vmstat` command as you run each of the following commands:

```
root@host01:~# swap -d /dev/zvol/dsk/rpool/swap
root@host01:~# zfs volsize=2G rpool/swap
root@host01:~# swap -a /dev/zvol/dsk/rpool/swap
```

## Solution 6-2: Using `cpustat` to View the `cpc` Counters

### Overview

In this practice, you use `cpustat` to view the `cpc` counters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. To see the events that can be monitored on your processor, enter:

```
root@host01:~# cpustat -h
Usage:
    cpustat -c eventspec [-c eventspec]... [-p period] [-Dmnst]
    [-A cor|soc|bins] [-k keys] [-o limit] [-I statfile]
    [-O statfile] [-T d | u] [interval [count]]

...
Generic Events:

    event0:  PAPI_l2_icm PAPI_l2_ldm PAPI_fp_ops PAPI_fp_ins
             PAPI_l1_icm PAPI_l1_dcm PAPI_tlb_im PAPI_tlb_dm

    event1:  PAPI_tot_ins

    See generic_events(3CPC) for descriptions of these events

Platform Specific Events:

    event0:  SB_full FP_instr_cnt IC_miss DC_miss ITLB_miss DTLB_miss
             L2_imiss L2_dmiss_ld

    event1:  Instr_cnt

    attributes: nouser sys

...
```

2. Find the event counter for the Instruction Cache references and the Instruction Cache Hit or Miss. For example, on the UltraSPARC T1, there is an event called `IC_miss`.
3. Using the `IC_miss` event name and the `cpustat` command, measure the instruction cache miss rate on your system.

```
root@host01:~# cpustat -c IC_miss
    time  cpu event      pic0
...
40.001    0  tick        488
40.002    1  tick       11522
40.001    2  tick       1463
```

```

40.001    3  tick      268
40.001    4  tick      582
...
40.002   26  tick    6552
40.002   27  tick   27943
40.002   28  tick  118366
40.001   29  tick     505
40.002   30  tick     190
40.001   31  tick     195
...

```

4. Start the Firefox browser. Use the `cputrack` command to monitor instruction cache misses for the `firefox` process.

```

root@host01:~# ps -e|grep firefox
16252 ?          0:41 firefox
root@host01:~# cputrack -c IC_miss -p 16252
   time lwp      event      pic0
 1.067   1      tick        0
 1.067   2      tick        0
 1.067   3      tick        0
 1.067   4      tick        0
 1.067   5      tick        0
 1.067   6      tick        0
 1.067   7      tick        0
 1.067  13      tick        0
 1.067   9      tick        0
 1.067  10      tick        0
 1.067  17      tick        0
 1.067  49      tick        0
...

```

## Solution 6-3: Using `mdb` to View Kernel Parameters

### Overview

In this practice, you use the `mdb` debugger to view the kernel parameters.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Start the `mdb` debugger for the kernel.

```
root@host01:~# mdb -k
Loading modules: [ unix genunix specfs dtrace zfs scsi_vhci sd
mpt mac px ldc ip hook neti arp usba kssl fctl sockfs random
mdesc idm cpc crypto fcip fcp ufs logindmux nsmb ptm spps nfs
lofs ipc ]
>
```

Notice the modules that are being loaded as `mdb` starts.

2. Run the `::dcmds` command.

```
> ::dcmds
...
```

3. This is a list of all the `dcmds` supported within `mdb`. Notice that `mdb` has a built-in `More`, so the output does not scroll off the screen. To see the `dcmds` in a more logical grouping, enter `q` to get out of the previous `More`.

```
>> More [<space>, <cr>, q, n, c, a] ? q
```

Enter the `::dmods -l` command, and then quit the `More` query.

```
> ::dmods -l
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

4. You can use the escape character `!` to execute a shell command. For example, run:

```
> !pwd
...
```

5. To search for a given command, you could also use the `!` character to pipe the output of an `mdb` command to a shell command such as `grep`. Try entering:

```
> ::dmods -l ! grep ps
...
```

In the output, you should see a reference to a `ps` `dcmd`.

6. To see how to use it, enter:

```
> ::help ps
...
```

7. From the synopsis, you can see the basic syntax. Try entering the following command, and then quit:

```
> ::ps
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

This is very similar to the `/usr/bin/ps` command, except that it gives you certain columns by default that are useful while you are in the debugger.

For example, it gives you the hex address of the `proc` structure, under the column `ADDR`.

8. The `-f` option gives you the long version of the command name. Enter:

```
> ::ps -f
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

9. The `-t` gives you the pointer to the kernel thread structure of all the threads within the process, plus the state of each thread. Try entering the following command, and then quit:

```
> ::ps -t
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

10. The `-l` option gives you the pointer to the `klwp` structure of all the threads within the process, plus the `lwp id` (thread ID). Try entering the following command, and then quit:

```
> ::ps -l
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

11. Try using them both together:

```
> ::ps -lt
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

What extra information do you get from the `-z` option?

\_\_\_A list of *processes related to non-global zones*\_\_\_

What extra information do you get from the `-T` and `-P` options?

\_\_\_A list of *task identifiers* and a list of *project identifiers*\_\_\_

12. You can capture your `mdb` session in a log file by using the `log dcmd`. Enter:

```
> ::help log
```

The `-e` option enables logging to the specified log file. The `-d` option disables the logging.

13. Enter the following command, and then quit:

```
> ::nm
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

14. This is `mdb`'s version of the `nm` (name list) command. If you know the variable of interest, you can use it to display only that entry. Enter:

```
> maxusers::nm
...
```



15. You can also see the type of data.

```
> maxusers::nm -f ctype
...
```

What is the size in bytes of the variable `maxusers`? \_\_\_\_\_

To display data in `mdb`, you can use the symbol that represents the address of the data followed by a `/` and a format.

16. To see a list of all the formats supported by `mdb`, enter the following command, and then quit:

```
> ::formats
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

17. Try entering:

```
> maxusers/D
...
```

What is the value of `maxusers` on your system? 2048

Try the same technique to find the values of the following parameters that will be discussed in the class:

`reserved_procs` - 5

`lotsfree` - 0

`fastscan` - 0

`maxpgio` - 0

`maxphys` - 131072

To exit `mdb`, enter:

```
> $q
```

In a future lesson practice, you will use `mdb -k` to look at more of the tunable parameters that are used for process management.

## Solution 6-4: Using Oracle Solaris Studio dbx to Test a Common Application (Optional)

### Overview

This practice is recommended for those planning to use Oracle Solaris Studio. In this practice, you use the Oracle Solaris Studio dbx tool to test a common application.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Review the source code of the `runtimeException.c` file. Take note of the comments inserted in the code and the instructions at the bottom of the file.

```
root@host01:~# cat /opt/ora/course_files/lab6/runtimeException.c
<output omitted>
```

2. Compile the code as directed in the file.

The `-g` option enables debug information to promote mapping. The compilation is stored in a file named `a.out`. Note that a warning similar to the one displayed as follows can be safely ignored after compilation:

```
root@host01:~# cd /opt/ora/software/SolarisStudio12.3-solaris-sparc-bin/solarisstudio12.3/bin
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-bin/solarisstudio12.3/bin# ./cc -g
/opt/ora/course_files/lab6/runtimeException.c
"/opt/ora/course_files/lab6/runtimeException.c", line 18:
warning: implicit function declaration: printf
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-bin/solarisstudio12.3/bin# ls -l a.out
-rwxr-xr-x  1 root  6864 Nov  8 08:35 a.out
```

3. Run the compiled program and observe the core dump file.

```
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-bin/solarisstudio12.3/bin# ./a.out
```

The program produces the following lines just before terminating:

```
1/10 = 0
1/9 = 0
1/8 = 0
1/7 = 0
1/6 = 0
1/5 = 0
1/4 = 0
1/3 = 0
1/2 = 0
1/1 = 1
Arithmetic Exception (core dumped)
```

Observe the core file and verify its source.

```
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ls -l core
-rw----- 1 root  root  2276174 Jan 8 08:37 core
root@host01:~# file core
core: ELF 32-bit MSB core file SPARC Version 1, from 'a.out'
```

4. Use the dbx tool to examine the core file. Print the stack of called functions.

```
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ./dbx a.out core
For information about new features see `help changes'
To remove this message, put `dbxenv suppress_startup_message 7.8'
in your .dbxrc
Reading a.out
core file header read successfully
Reading ld.so.1
Reading libc.so.1
program terminated by signal FPE (integer divide by zero)
Current function is div
    4                return x/y;
(dbx) where
=>[1] div(x = 1, y = 0), line 4 in "runtimeException.c"
[2] iter(c = 10), line 18 in "runtimeException.c"
[3] main(), line 27 in "runtimeException.c"
(dbx) exit
```

The core file points to the last instruction issued when the process faulted. With the debugging information in the executable, dbx can point to the line of code in question.

5. Use dbx to examine a core dump after the executable is stripped (the output of each of the following commands is omitted):

```
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# file a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ls -l a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# strip a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ls -l a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# file a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ./a.out
root@host01:/opt/ora/software/SolarisStudio12.3-solaris-sparc-
bin/solarisstudio12.3/bin# ./dbx a.out core
```

Note that after stripping the executable, dbx can examine only the native compiled form of the failed process.

## Solution 6-5: Using the GUDS Script

### Overview

In this practice, you use the GUDS script to collect performance data.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Locate the GUDS zip file in the `/opt/ora/software` directory.

```
root@host01:~# cd /opt/ora/software
root@host01:/opt/ora/software# ls guds*
guds_3_0.zip
```

2. Unzip the GUDS script.

```
root@host01:/opt/ora/software# unzip guds_3_0.zip
Archive:  guds_3_0.zip
inflating: guds_3_0
```

3. Run the GUDS script. Use the values shown in the following example:

```
root@host01:/opt/ora/software# ./guds_3_0
...
Please enter your Service Request number.
The SR number should only consist of digits or of the pattern
[0-9]-[0-9]+.
SR #:1234

Enter the number of hours for the script to run.
If the number of hours given is zero, the script will
run for the specified number of iterations.
For a positive, non zero number of hours, the script
will run using the specified count, interval and wait
parameters until the given number of hours has been
met or exceeded.
Hours[0]: 0

Please enter the count option for commands that require it.
These are commands like vmstat and iostat which use an
interval and a count to repeat the output.
For example, using a count of 10 and interval of 30
will cause iostat, vmstat, mpstat, etc to output
10 sets of data, with 30 seconds between data sets.
If you have not been given a value for this number,
the default value will be 40.
Count[40]: 5
```

Please enter the interval option for commands that require it.  
These are commands like vmstat and iostat which use an interval and count to repeat the output.

If you have not been given a value for this number,  
the default value will be 5.

Interval[5]: 5

Please enter the number of total data sets for the script  
to capture. Each data set contains the output of one SET  
of a commands output.

For example, a single data set would contain an entire  
"ps -elf" output, and an entire vmstat output, where  
the vmstat output is made up of a variable number of data  
lines, as defined by "count" and "interval"

If you have not been given instructions for this number,  
the default value will be 5.

Iterations[5]: 5

Please enter the number of seconds to wait between  
collecting data sets.

To ensure correct data collection, any value entered  
will be added to the minimum wait time.

If you have not been given instructions for this number,  
the default value will be 0.

Wait[0]: 0

Do you want to run an extended set of commands?

If you have not been given instructions for this number,  
the default value will be 2

level 0 : nothing

level 1 : lockstat for contention and hold events

level 2 : L1 + trapstat, lockstat profiling, threadlist,  
TNF tracing (default)

level 3 : L2 + kmastat, kmausers, memstat

Enter the level : '0', '1', '2' or '3'.

Extended[2]: 3

Do you want to collect prstat cpu, rss, size and zones data? [n]:  
**y**

Do you want to collect configuration data? [y]: **y**

Please enter a one line statement describing the current capture environment. Some examples are "Performance issue is present" or "System is operating normally".

This statement will be added to the INFO file so the support engineer knows the state of the system associated with this data.

Description: **Test run.**

Enter the location of the output directory [/var/tmp/guds]: **<CR>**

Starting Data Gathering

Collecting performance data...

Begin Iteration #1

40/40

End Iteration #1

Waiting for processes to terminate, then sleeping 0 seconds before the next iteration.

--- All processes finished. ---

Begin Iteration #2

5/40

...

--- All processes finished. ---

Collecting System Information

22/22

Creating zip file...

Please send the data to your support engineer.

The data are stored in

/var/tmp/guds/1234/1234\_guds.004eebdb.server1-2012.12.19.11.46.zip

Instructions for uploading a file via a web browser.

Go to URL: <https://supportfiles.sun.com>

Under Step 1, select the file to send

Under Step 2, set the destination to: cores

Under Step 3, click on Upload

Instructions for uploading a file via ftp to supportfiles.sun.com.

# ftp supportfiles.sun.com

login: anonymous

passwd: your email address

ftp> cd /cores

ftp> binary

ftp> put 1234\_guds.004eebdb.server1-2012.12.19.11.46.zip

ftp> bye

Instructions for uploading a file through My Oracle Support.

NOTE: Only for files less than 2G in size.

Go to URL: <https://support.oracle.com>

If you are logging the Service Request you can upload the guds output in Step 4 - Upload Files.

If you have already logged the Service Request and need to add the guds output to it, from either the Dashboard or Service Request tab find and click on your Service Request.

Click the Upload button or select Upload from the Actions drop down menu. Browse for your guds .zip file and choose it, then click the Upload button.

Your service request will automatically be updated to alert the support engineer once the data is available. You may also update the service request in My Oracle Support yourself.

4. Change to the GUDS data collection directory and list the contents.

```
root@host01:/opt/ora/software# cd /var/tmp/guds/1234
root@host01:/var/tmp/guds/1234# ls -l
total 7452
-rw-rw-rw-  1 root    root    1907938 Dec 19 05:03
1234_guds.004eebdb.server1-2012.12.19.11.46.zip
drwxr-xr-x  3 root    root      113 Dec 19 05:03
guds.004eebdb.server1-2012.12.19.11.46
```

Note that the zip file is used for uploading your performance data to Oracle Support for analysis. The directory contains the performance data for local analysis.

5. List the contents of the directory containing the performance data files.

```
root@host01:/var/tmp/guds/1234# ls guds.004eebdb.server1-
2012.12.19.11.46 | more
adb.out
arcstat.out
df-gl.out
df-kl.out
dladm-show-aggr-L.out
dladm-show-aggr.out
dladm-show-linkprop.out
dladm-show-phys.out
dmesg.out
drv
dutmp.out
errors.log
ifconfig-a.out
...
```

6. Review the various performance data files to get an idea of the type of information that is collected by the GUDS script.



# **Practices for Lesson 7: Processes and Threads**

## **Chapter 7**

## Practices for Lesson 7: Overview

---

### Practices Overview

In these practices, you will perform the following:

- Examine process life cycles by using `truss`
- Use the `lockstat` command
- Use the `lockstat` provider in DTrace
- Use `plockstat(1M)` and the `plockstat` provider in DTrace
- Check tunable parameters with `mdb`

Solutions for each task in this practice are provided after the practices exercises are introduced, at the end of this guide.

## Practice 7-1: Examining Process Life Cycles by Using `truss`

---

### Overview

In this practice, you look at the system calls that comprise the stages of a process lifetime.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Perform the following steps:

1. Change to the `/opt/ora/course_files/lab7` directory and display the contents of the `fork_exec_wait.c` program.
2. Describe the purpose of the `fork()` function:

3. Run the `fork_exec_wait` program.
4. Use `truss -f` to trace the `fork_exec_wait` system calls.

Identify the system calls that:

- Create a new process: \_\_\_\_\_
- Have the new process throw away the parent's address space and map in a new address space: \_\_\_\_\_
- Cause the child to give up its resources and enter the zombie state: \_\_\_\_\_
- Reap the exit status of the child program: \_\_\_\_\_

What does `-f` do in the `truss` command? \_\_\_\_\_

Why do you see two calls to `execve()` and `_exit()`?  
\_\_\_\_\_

You should see two lines for `fork1()`. Even though it is called only once by the parent process, it returns twice. To the parent, `fork` returns the PID of the child. To the child process, `fork` returns 0.

## Practice 7-2: Using the `lockstat` Command

### Overview

In this practice, you use `lockstat` to display lock activity in the kernel, to distinguish between hold events and contention events, and to use the `lockstat` provider in DTrace to examine the processes causing lock contention. In addition, you use `plockstat (1M)` and its options.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Run the `lockstat` command.
2. Write a `lockstat` command that lists all the locks used in the kernel, over a 5-second period, in the order of the locks that were held for the greatest time. Send the output to a file called `/tmp/lockstat.out`.

```
root@host01:~# lockstat -A -o /tmp/lockstat.out sleep 5
...
root@host01:~# more /tmp/lockstat.out
Adaptive mutex spin: 545 events in 5.663 seconds (115
events/sec)
...
```

Were any locks taken during the 5 seconds? \_\_\_\_\_

What kind of lock was most commonly taken? \_\_\_\_\_

3. Change the `lockstat` command to limit it to the top 10 events for each type of lock (use `-D`).
4. Write a `lockstat` command that lists all the lock events for the ZFS file system driver over a 10-second period. Send the output to a file called `/tmp/lockstat_ZFS.out`.

**Note:** Run `modinfo|grep zfs` first to gather the ZFS file drive information.

How many adaptive mutex spin events were detected? \_\_\_\_\_

How many adaptive mutex hold events were detected? \_\_\_\_\_

How many R/W writer hold events were detected? \_\_\_\_\_

How many R/W reader hold events were detected? \_\_\_\_\_

Of the R/W reader events, which caller had the highest percentage? \_\_\_\_\_

## Practice 7-3: Using the lockstat Provider in DTrace

### Overview

In this practice, you use the `lockstat` provider in DTrace.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Use the following DTrace one-liner to trace all the locks acquired by a simple command such as `date`. Enter this `dtrace` command in one window:

```
root@host01:~# dtrace -n 'lockstat:::*acquire \
/execname == "date"/ { @[probename] = count(); }'
```

In another window, enter:

```
root@host01:~# date
```

2. Enter `^c` in the DTrace window.

Did `date` use any locks in the kernel? \_\_\_\_\_

Try this one-liner to count the number of lock acquisitions:

```
root@host01:~# dtrace -n 'lockstat:::*acquire /execname ==
"date"/{@[execname]=count()}'
```

Run the `date` command again.

Now enter a `^c` in the DTrace window. How many locks did the `date` command acquire?  
\_\_\_\_\_

3. Try this version to see if all commands are using that many locks:

Enter this in one window:

```
root@host01:~# dtrace -n 'lockstat:::*acquire
{@[execname]=count()}'
```

In another window, enter several simple commands such as:

- `pwd`
- `ps`
- `cd .`
- `ls`
- `sh`
- `bash`

Now enter a `^c` in the DTrace window.

<output omitted>

Remember that all the executables that were active while the DTrace command was running will be listed.

Which of these commands took the most kernel locks during that time?  
\_\_\_\_\_

## Practice 7-4: Using plockstat (1M) and the plockstat Provider in DTrace

---

### Overview

In this practice, you use `plockstat (1M)` and the `plockstat` provider in DTrace.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. In a terminal window, enter:

```
root@host01:~# plockstat
Usage:
    plockstat [-vACHV] [-n count] [-s depth] [-e secs] [-x
opt [=val]]
    command [arg...]
    plockstat [-vACHV] [-n count] [-s depth] [-e secs] [-x
opt [=val]]
    -p pid
```

2. As a simple example, enter:

```
root@host01:~# plockstat -A date
...
```

What type of locking events showed most commonly? \_\_\_\_\_

3. Try it with a few simple commands, such as `ls`, `pwd`, and `cd /var/tmp`.
4. One of the really nice options is `-v` because it gives you the basis for writing your own DTrace script by using the `plockstat` provider. Try it by entering:

```
root@host01:~# plockstat -AV date
...
```

## Practice 7-5: Checking Tunable Parameters with `mdb`

### Overview

In this practice, you look at the values of the kernel tunable parameters for processes to see if they are set at system default values.

Using the same technique as in the previous lesson titled “Other Significant Tools,” you check the tunable parameters associated with process management.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Run the `mdb -k` command.
2. Use the same technique as in the practice for the previous lesson titled “Other Significant Tools” to retrieve the value of the kernel tunable parameters for processes. Run `maxusers::nm` to refresh your memory.

What is the size in bytes of the variable `maxusers`? \_\_\_\_\_

3. You can also see the type of data.

```
> maxusers::nm -f ctype
C Type
-----
int
```

**Note:** To display data in `mdb`, you can use the symbol that represents the address of the data followed by a `/` and a format. To see a list of all the formats supported by `mdb`, run the `::formats` command.

4. Display the `maxusers` property in decimal signed integer format.

```
> maxusers/D
maxusers:      2048
```

5. Using the `D` format, note the value of:

```
maxusers _____
max_nprocs _____
maxpid _____
pidmax _____
maxuprc _____
reserved_procs _____
```

6. While you are in `mdb`, check the value of:

```
nproc _____
nthread _____
```

7. Compare this to what you see by using `prstat`. Are they close?

Does `prstat` count system threads? \_\_\_\_\_

8. Compare this to what `kstat` gives you for `nproc`.  
Does `kstat` match `mdb` or `prstat` more closely? \_\_\_\_\_
9. Use `!kstat -n segkp` to search for information on the kernel tunable parameter `segkp` (the kernel pageable memory segment).  
What is `mem_total` set to on your system? \_\_\_\_\_  
How many kernel threads can you run on your system? \_\_\_\_\_  
Should you adjust the `segkp` parameter if you were going to increase the `pidmax` and `max_nprocs` parameters?  
\_\_\_\_\_
10. Exit from `mdb`.



## Practice 7-6: Creating and Managing Processor Sets

### Overview

In this practice, you create a processor set and assign CPUs and processes to the processor set.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Perform the following steps:

1. Open two terminal windows and record the process ID values of the shells in each window.  
Record the PID of each shell here:  
PID1: \_\_\_\_\_  
PID2: \_\_\_\_\_
2. In the first window, display the processor statistics by running the `mpstat -a 5` command.  
Note the `smtx` field. What does this field indicate?  
\_\_\_\_\_

3. In the other two windows, run the following commands:

```
oracle@host01:~$ exec csh
server1% while 1
? end
```

4. Observe the values in the `smtx` column. Note the increase in the mutex spin locks.
5. Create a processor set and add processor 3 to it by executing the following command and pressing Return:

```
oracle@host01:~$ su -
Password: cangetin
oracle@host01:~# psrset -c 3
created processor set 1
processor 3: was not assigned, now 1
```

What changed in the `mpstat` output window?

6. Run the `prset -i` and `psrinfo` commands to verify the results.
7. Create another set consisting of processor 2. Observe the `mpstat` output.
8. Now add processor 1 to processor set 2. Observe the `mpstat` output.
9. Bind the looping processes in the two windows to the first processor set by typing the following command. Observe the `mpstat` output.

**Hint:** Use `prstat` to obtain the PIDs.

```
oracle@host01:~# psrset -b 1 window1_PID window2_PID
process id XXXX: was not bound, now 1
process id XXXX: was not bound, now 1
```

10. Begin two new processes in the second set by executing the following commands. Observe the `mpstat` output window.

```
root@host01:~# psrset -e 2 dd if=/dev/zero of=/dev/null&  
[2] 18633  
root@host01:~# psrset -e 2 dd if=/dev/zero of=/dev/null&  
[3] 18634
```

11. Delete the second processor set by entering the following command. Observe the `mpstat` output window.

```
root@host01:~# psrset -d 2
```

What happened to the two `dd` processes? What happened to the `mpstat` output?

12. Run the `prset -i` command to verify the results.
13. Delete the remaining processor set (Set 1). Observe the `mpstat` output window.
14. Terminate the other practice windows.

## Practice 7-7: Changing the Default Scheduling Class to FSS

### Overview

In this practice, you perform the following tasks to change the default scheduling class to Fair Share Scheduling (FSS):

- Adding some simple projects to the `/etc/project` file
- Changing the default scheduling class to FSS
- Adding some new tasks in each of the created projects
- Observing the CPU consumption of the compute-bound processes in these projects by using the `prstat -J` command

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task 1: Adding Projects

Perform the following steps:

1. Create some projects by entering:

```
root@host01:~# projadd -U root -K \
"project.cpu-shares=(priv,3,none)" payroll
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,5,none)" eng
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,2,none)" research
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,0,none)" sales
```

2. Validate that the projects were added correctly by entering:

```
root@host01:~# projects -l
...
```

3. Change the default scheduling class to Fair Share Scheduling.
4. Make this configuration take effect immediately, without rebooting.

### Task 2: Adding New Tasks to Each Project

Perform the following steps:

1. Start an infinite loop in the sales project.

In a window, enter:

```
root@host01:~# newtask -p sales dd if=/dev/zero of=/dev/null&
```

2. In a separate window, run `prstat -JR` to observe the distribution of CPU time.

Leave the window that is running `prstat` open for the duration of this exercise.

After a short time, which process is consuming the bulk of the CPU time?

What is its PID? \_\_\_\_\_

In the lower portion of the window, what are the current project IDs on the system?

Which is consuming the bulk of the CPU? \_\_\_\_\_

How many processes are in that project (NPROC)? \_\_\_\_\_

What is the current CPU% value for the sales project? \_\_\_\_\_

What does the CPU% value indicate? \_\_\_\_\_

How many `cpu-shares` did that project get (see `/etc/project`)?  
\_\_\_\_\_

3. Enter:

```
root@host01:~# newtask -p eng dd if=/dev/zero of=/dev/null
```

What new project ID is shown in `prstat`? \_\_\_\_\_

How many processes are in that project (NPROC)? \_\_\_\_\_

Which project is getting most of the CPU? \_\_\_\_\_

4. Start another infinite loop in the payroll and research projects by entering:

```
root@host01:~# newtask -p payroll dd if=/dev/zero of=/dev/null&
root@host01:~# newtask -p research dd if=/dev/zero \
of=/dev/null&
```

After a short time, note that all the projects are getting the same percentage of CPU time even though you assigned different share values to each project.

Why? \_\_\_\_\_

5. Now start an infinite loop in each project by entering:

```
root@host01:~# newtask -p research dd if=/dev/zero of=/dev/null&
[5] 20254
root@host01:~# newtask -p eng dd if=/dev/zero of=/dev/null&
[6] 20260
root@host01:~# newtask -p sales dd if=/dev/zero of=/dev/null&
[7] 20261
root@host01:~# newtask -p research dd if=/dev/zero of=/dev/null&
[8] 20262
root@host01:~# newtask -p payroll dd if=/dev/zero of=/dev/null&
```

How many processes are in that project (NPROC)? \_\_\_\_\_

After a short time, see if the project CPU percentages are still equal? If the answer is Yes, repeat step 5.

**Note:** In the bash shell, you can use the up arrow key to access command history.

At what point did the CPU shares start taking effect?

6. Repeat step 5 a few more times and observe how the assignment of CPU shares affects CPU% for each project.
7. When you have finished experimenting with the Fair Share Scheduler, kill the `dd` processes and revert to the TS scheduler.

## Practice 7-8: Determining CPU Information and Activity Using DTrace Toolkit Scripts

### Overview

In this practice, you use the various CPU-related DTrace scripts that are found in the DTrace Toolkit.

You observe CPU activity by using the following scripts:

- `cputypes.d` to list CPU types
- `cpuwalk.d` to measure the CPUs that a process runs on
- `dispqlen.d` to display dispatcher queue length by CPU
- `runocc.d` to display run queue occupancy by CPU
- `pridist.d` to display process priority distribution
- `cswstat.d` to display context-switch time statistics

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Enter:

```
root@host01:~# cputypes.d
...
```

How many CPUs are there? \_\_\_\_\_

What processor group are they in? \_\_\_\_\_

What are their clock speeds? \_\_\_\_\_

What is the LGRP column for? (See `man cputypes.d`.)

2. Enter:

```
root@host01:~# cpuwalk.d 10
...
```

What do the value and count columns represent? (See `man cputypes.d`.)

\_\_\_\_\_

Which process has been executing the most? \_\_\_\_\_

Which CPUs are being used? \_\_\_\_\_

Are all the CPUs being used fairly equally? \_\_\_\_\_

3. In a separate window, go to the `/opt/ora/course_files/lab7` directory and run the `while1` program in the background by entering:

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

4. Enter:

```

root@host01:~# cpuwalk.d 10
...
PID: 20329      CMD: while1

      value  ----- Distribution ----- count
      10 |
      11 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 9917
      12 |

```

Did the `while1` process run on all CPUs? \_\_\_\_\_

How many times did it approximately run altogether? \_\_\_\_\_

5. Try running a few more `while1` processes in the background, and then run `cpuwalk.d` again.

```

root@host01:~# /opt/ora/course_files/lab7/while1&
...
root@host01:~# /opt/ora/course_files/lab7/while1&
...

```

```

root@host01:~# cpuwalk.d 10
...

```

What happened to the CPU distribution?

\_\_\_\_\_

6. Kill the `while1` processes by entering:

```

root@host01:~# pkill while1
...

```

7. Enter:

```

root@host01:~# man dispqlen.d
...

```

What is its stability? \_\_\_\_\_

How many times a second does it sample the length of the dispatch queue?

\_\_\_\_\_

8. Enter:

```

root@host01:~# dispqlen.d
...

```

9. After the "Sampling... Hit Ctrl-C to end" message is displayed, count to five, and then press `^C`.

What is being displayed in the count column? \_\_\_\_\_

What were the run queue lengths? \_\_\_\_\_

10. Run a `while1` process for each CPU by entering (once for each CPU):

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

11. Enter:

```
root@host01:~# dispqlen.d
...
```

After the "Sampling... Hit Ctrl-C to end" message is displayed, count to five, and then press **^c**.

What were the run queue lengths? \_\_\_\_\_

Why? \_\_\_\_\_

12. In one window, enter:

```
root@host01:~# runocc.d
...
```

What does it show? \_\_\_\_\_

13. After you have finished, kill the `while1` process:

```
root@host01:~# pkill while1
...
```

**Note:** Remember that this would be a good way to check for CPU saturation.

14. Look at the man page for `pridist.d`.

```
root@host01:~# man pridist.d
...
```

Try running it.

```
root@host01:~# pridist.d
...
```

What does this program show?

\_\_\_\_\_

15. Run a `while1` program in the background by entering:

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

Note the PID.

```
root@host01:~# pgrep while1
...
```

16. Enter the following DTrace one-liner, let it run for several seconds, and then press **^c**.

```
root@host01:~# dtrace -s /usr/demo/dtrace/profpri.d
PID_of_while1
...
^C

while1
```

value	----- Distribution -----	count
< 0		0
0	@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@	6116
10	@@@@@@	1600
20	@@@@@	1278
30	@@@	880
40	@@	440
50	@@	429
60		0

Observe the `while1` entry. What priorities did `while1` spend most of its time on? \_\_\_\_\_

17. Run the `pridist.d` DTrace script and observe the `while1` process.

```
root@host01:~# pridist.d
...
CMD: while1          PID: 20380
```

value	----- Distribution -----	count
< 0		0
0	@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@	3706
5		0
10	@@@@@@	1220
15		0
20	@@@@@	782
25		0
30	@@@	560
35		0
40	@@	280
45		0
50	@@	273
55		0

Notice that the `profpri.d` one-liner and the `pridist.d` script have similar outputs.

18. Try changing the `profpri.d` program to a one-liner.
- Use a predicate of `/execname == "while1"/` instead of `/pid == $1/`.
  - Use a step value of 5 instead of 10 in the call to `lquantize()` to allow a finer granularity of distribution.
  - Test your one-liner on the `while1` process. For example:

```
root@host01:~# dtrace -n 'profile-1001 /execname == "while1"/ {
  @["while1 priority dist"] = lquantize(curlwpsinfo->pr_pri, 0,
  100, 5)}' -c /opt/ora/course_files/lab7/while1
...
^C
while1 priority dist
value ----- Distribution ----- count
```



< 0		0
0	@@@@@@@@@@@@@@@@@@@@@@	12896
5		0
10	@@@@@@@	3948
15		0
20	@@@@	3197
25		101
30	@@@	2095
35		79
40	@@	960
45		49
50	@@	911
55		39
60		0

19. Look at the man page for `cswstat.d`.

```
root@host01:~# man cswstat.d
...
```

20. Enter:

```
root@host01:~# cswstat.d
...
```

What is the typical amount of time spent in context switching (`CSWTIME(us)`)?

What is the typical average time per context switch (`AVGTIME(us)`)?

21. Kill the `while1` processes when you have finished by entering:

```
root@host01:~# pkill while1
...
```

## Practice 7-9: Locating Hot Routines

---

### Overview

There is a program in the `/opt/ora/course_files/lab7` directory called `fault1`. In this practice, you identify the hot routines in this application by using `mpstat`, `prstat`, and `DTrace`.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. In one window, run the program `fault1` by entering:

```
root@host01:~# /opt/ora/course_files/lab7/fault1
```

2. In another window, run:

```
root@host01:~# mpstat 5
```

Does it look like this load is CPU intensive? \_\_\_\_\_

How many CPUs is it consuming? \_\_\_\_\_

3. Verify that it is a single-threaded application that is using `prstat` or `ps`.

Is it single threaded? \_\_\_\_\_

Is it spending most of its time in user or kernel mode?

\_\_\_\_\_

## Solution 7-1: Examining Process Life Cycles by Using `truss`

---

### Overview

In this practice, you look at the system calls that comprise the stages of a process lifetime.

**Note:** The output from the commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Task

Perform the following steps:

1. Change to the `/opt/ora/course_files/lab7` directory and display the contents of the `fork_exec_wait.c` program.

```
root@host01:/opt/ora/course_files/lab7# more fork_exec_wait.c
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

main()
{
    pid_t cpid;
    int estat;
    cpid = fork();
    if (cpid == 0) {
        execl("/usr/bin/pwd", "/usr/bin/pwd");
    }
    if (cpid > 0 ) {
        waitpid(cpid, &estat, WNOWAIT);
        printf("parent child's exit status is %d\n", estat);
        exit(0);
    }
    if (cpid < 0) {
        perror("Fork Failed");
        exit(1);
    }
}
```

- Describe the purpose of the `fork()` function:

*The `fork()` function is used to create a new process by duplicating the existing process from which it is called. The existing process from which this function is called becomes the parent process and the newly created process becomes the child process. The child is a duplicate copy of the parent but there are some exceptions:*

- The child has a unique PID like any other process running in the operating system.*
- The child has a parent process ID, which is same as the PID of the process that created it.*
- Resource utilization and CPU time counters are reset to zero in a child process.*
- The set of pending signals in a child is empty.*
- The child does not inherit any timers from its parent.*

- Run the `fork_exec_wait` program.

```
root@host01:/opt/ora/course_files/lab7# ./fork_exec_wait
/opt/ora/course_files/lab7
parent child's exit status is 0
```

- Trace the `fork_exec_wait` system calls.

```
root@host01:/opt/ora/course_files/lab7# truss -f
./fork_exec_wait
18203:   execve("fork_exec_wait", 0xFFBFFC54, 0xFFBFFC5C)   argc
= 1
18203:   sysinfo(SI_MACHINE, "sun4v", 257)                   = 6
18203:   mmap(0x00000000, 32, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF390000
18203:   memcntl(0xFF3A0000, 38284, MC_ADVISE, MADV_WILLNEED,
0, 0) = 0
18203:   mmap(0x00000000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF380000
...
```

Identify the system calls that:

- Create a new process: `__fork()` `__`
- Have the new process throw away the parent's address space and map in a new address space: `__fork()` `__`
- Cause the child to give up its resources and enter the zombie state: `__fork()` `__`
- Reap the exit status of the child program: `__fork()` `__`

What does `-f` do in the `truss` command? `__ -f follows all the children created by fork() or vfork(), and includes their signals, faults, and system calls in the traceout.` `__`

Why do you see two calls to `execve()` and `__exit()`?

One to start the process and one to end the process

You should see two lines for `fork()`. Even though it is called only once by the parent process, it returns twice. To the parent, `fork` returns the PID of the child. To the child process, `fork` returns 0.

## Solution 7-2: Using the lockstat Command

### Overview

In this practice, you use `lockstat` to display lock activity in the kernel, to distinguish between hold events and contention events, and to use the `lockstat` provider in DTrace to examine the processes causing lock contention. In addition, you use `plockstat (1M)` and its options.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Run the `lockstat` command.

```
root@host01:~# lockstat
Usage: lockstat [options] command [args]

Event selection options:

-C          watch contention events [on by default]
-E          watch error events [off by default]
-H          watch hold events [off by default]
-I          watch interrupt events [off by default]
...

```

2. Write a `lockstat` command that lists all the locks used in the kernel, over a 5-second period, in the order of the locks that were held for the greatest time. Send the output to a file called `/tmp/lockstat.out`.

```
root@host01:~# lockstat -A -o /tmp/lockstat.out sleep 5
...
root@host01:~# more /tmp/lockstat.out
Adaptive mutex spin: 545 events in 5.663 seconds (115
events/sec)
...

```

Were any locks taken during the 5 seconds? 545

What kind of lock was most commonly taken? adaptive mutex spin

3. Change the `lockstat` command to limit it to the top 10 events for each type of lock (use `-D`).

```
root@host01:~# lockstat -A -D10 -o /tmp/lockstat.out sleep 5
...
root@host01:~# more /tmp/lockstat.out
Adaptive mutex spin: 649 events in 5.663 seconds (115
events/sec)
Count  indiv  cuml  rcnt          nsec Lock                      Caller
-----
  44    7%    7%  0.00      4956 0x40325a60138      taskq_thread_wait+0x40
  29    4%   11%  0.00      5103 0x40325a611a0      taskq_thread_wait+0x40

```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

27	4%	15%	0.00	251793	0x40325da8f70	taskq_thread+0x3d0
26	4%	19%	0.00	5790	0x40325a60250	taskq_thread_wait+0x40
25	4%	23%	0.00	3653	0x40325da9600	taskq_thread_wait+0x40
21	3%	27%	0.00	6737	0x40325a612b8	taskq_thread_wait+0x40
21	3%	30%	0.00	4966	0x40325a60020	taskq_thread_wait+0x40
19	3%	33%	0.00	5972	0x40325da9c90	taskq_thread_wait+0x40
18	3%	35%	0.00	4228	0x40325da9b78	taskq_thread_wait+0x40
18	3%	38%	0.00	5394	0x40325da9718	taskq_thread_wait+0x40
...						

4. Write a lockstat command that lists all the lock events for the ZFS file system driver over a 10-second period. Send the output to a file called /tmp/lockstat\_ZFS.out.

**Note:** Run `modinfo|grep zfs` first to gather the ZFS file drive information.

```

root@host01:~# modinfo|grep zfs
 14 7be00000 c68d8 8 1 zfs (ZFS filesystem version 34)
 14 7be00000 c68d8 300 1 zfs (ZFS storage pool)
root@host01:~# lockstat -A -f 0x7be00000,0xc68d8 \
-o /tmp/lockstat_ZFS.out sleep 5
root@host01:~# more /tmp/lockstat_ZFS.out

Adaptive mutex spin: 1 events in 5.296 seconds (0 events/sec)

Count indiv cuml rcnt      nsec Lock                      Caller
-----
 1 100% 100% 0.00      3624 0x400016641e8          txg_thread_wait+0x4c
-----

Adaptive mutex hold: 522 events in 5.296 seconds (99 events/sec)

Count indiv cuml rcnt      nsec Lock                      Caller
...
R/W reader hold: 78 events in 5.296 seconds (15 events/sec)
Count indiv cuml rcnt      nsec Lock                      Caller
-----
 8 10% 10% 0.00      9382 0x403262fb780          dmu_object_size_from_db+0x6
...

```

How many adaptive mutex spin events were detected? \_1\_

How many adaptive mutex hold events were detected? \_522\_

How many R/W writer hold events were detected? \_0\_

How many R/W reader hold events were detected? \_78\_

Of the R/W reader events, which caller had the highest percentage? \_\_

dmu\_object\_size\_from\_db+0x6\_\_

## Solution 7-3: Using the lockstat Provider in DTrace

### Overview

In this practice, you use the `lockstat` provider in DTrace.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Use the following DTrace one-liner to trace all the locks acquired by a simple command such as `date`. Enter this DTrace command in one window:

```
root@host01:~# dtrace -n 'lockstat:::*acquire \
/execname == "date"/ { @[probename] = count(); }'
```

In another window, enter:

```
root@host01:~# date
```

2. Enter `^c` in the DTrace window.

Did `date` use any locks in the kernel? Yes

Try this one-liner to count the number of lock acquisitions:

```
root@host01:~# dtrace -n 'lockstat:::*acquire /execname == \
"date"/{@[execname]=count()}'
```

Run the `date` command again.

Now enter a `^c` in the DTrace window. How many locks did the `date` command acquire?  
6423

3. Try this version to see if all commands are using that many locks:

Enter this in one window:

```
root@host01:~# dtrace -n 'lockstat:::*acquire
{@[execname]=count()}'
```

In another window, enter several simple commands such as:

- `pwd`
- `ps`
- `cd`
- `ls`
- `sh`
- `bash`

Now enter a `^c` in the DTrace window.

<output omitted>

Remember that all the executables that were active while the DTrace command was running will be listed.

Which of these commands took the most kernel locks during that time?

ps

## Solution 7-4: Using plockstat (1M) and the plockstat Provider in DTrace

---

### Overview

In this practice, you use `plockstat (1M)` and the `plockstat` provider in DTrace.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. In a terminal window, enter:

```
root@host01:~# plockstat
Usage:
    plockstat [-vACHV] [-n count] [-s depth] [-e secs] [-x
opt [=val]]
        command [arg...]
    plockstat [-vACHV] [-n count] [-s depth] [-e secs] [-x
opt [=val]]
        -p pid
```

2. As a simple example, enter:

```
root@host01:~# plockstat -A date
...
```

What type of locking events showed most commonly? \_\_\_ Mutex hold \_\_\_

3. Try it with a few simple commands, such as `ls`, `pwd`, and `cd /var/tmp`.
4. One of the really nice options is `-v` because it gives you the basis for writing your own DTrace script by using the `plockstat` provider. Try it by entering:

```
root@host01:~# plockstat -AV date
...
```



## Solution 7-5: Checking Tunable Parameters with `mdb`

### Overview

In this practice, you look at the values of the kernel tunable parameters for processes to see if they are set at system default values.

Using the same technique as in the previous lesson titled “Other Significant Tools,” you check the tunable parameters associated with process management.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Run the `mdb -k` command.

```
root@host01:~# mdb -k
Loading modules: [ unix genunix specfs dtrace zfs scsi_vhci sd
mpt mac px ldc ip hook neti arp usba kssl fctl sockfs random idm
cpc crypto fcp fcip mdesc ufs logindmux nsmb ptm spps nfs lofs
ipc ]
>
```

2. Use the same technique as in the practice for the previous lesson titled “Other Significant Tools” to retrieve the value of the kernel tunable parameters for processes. Run `maxusers::nm` to refresh your memory.

```
> maxusers::nm
Value          Size          Type  Bind  Other Shndx  Name
0x00000000100dc998|0x0000000000000004|OBJT|GLOB|0x0|10|maxusers
```

What is the size in bytes of the variable `maxusers`? `_ 0x0000000000000004 _`

3. You can also see the type of data.

```
> maxusers::nm -f ctype
C Type
-----
int
```

**Note:** To display data in `mdb`, you can use the symbol that represents the address of the data followed by a `/` and a format. To see a list of all the formats supported by `mdb`, enter the following command, and then quit:

```
> ::formats
...
>> More [<space>, <cr>, q, n, c, a] ? q
```

4. Display the `maxusers` property in decimal signed integer format.

```
> maxusers/D
maxusers:      2048
```

5. Using the `D` format, note the value of:  
maxusers: \_\_2048\_\_  
max\_nprocs: \_\_30000\_\_  
maxpid: \_\_30000\_\_  
pidmax: \_\_30000\_\_  
maxuprc: \_\_29995\_\_  
reserved\_procs: \_\_5\_\_
6. While you are in `mdb`, check the value of:  
nproc: \_\_152\_\_  
nthread: \_\_1801\_\_
7. Compare this to what you see by using `prstat`. Are they close?  
\_\_Yes\_\_  
Does `prstat` count system threads? \_\_Yes, see the value in the lwps field.\_\_
8. Compare this to what `kstat` gives you for `nproc`.  
Does `kstat` match `mdb` or `prstat` more closely? \_\_prstat\_\_
9. Use `!kstat -n segkp` to search for information about the kernel tunable parameter `segkp` (the kernel pageable memory segment).  
What is `mem_total` set to on your system? \_\_ 2147483648 (2 GB by default) \_\_  
How many kernel threads can you run on your system? \_\_ The number of kernel threads (LWPs) depends on the amount of RAM or to be more precise on the size of segkp. The default size of 2 Gbytes allows creation of 24 Kbyte stacks for more than 87,000 kernel threads.\_\_  
Should you adjust the `segkp` parameter if you were going to increase the `pidmax` and `max_nprocs` parameters?  
\_\_Yes\_\_
10. Exit from `mdb`.  

`> ::quit`

## Solution 7-6: Creating and Managing Processor Sets

### Overview

In this practice, you create a processor set and assign CPUs and processes to the processor set.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Open two terminal windows and record the process ID values of the shells in each window.

Record the PID of each shell here:

PID1: 20153

PID2: 20162

2. In the first window, display the processor statistics by running the `mpstat -a 5` command.

```
root@host01:~# mpstat -a 5
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    0    0  295 1015  134  681    1    9  14    0  486    0    0    0 100  29
...
```

Note the `smtx` field. What does this field indicate? It shows spins on reader/writer locks (lock not acquired on first try).

3. In the other two windows, run the following commands:

```
oracle@host01:~$ exec csh
server1% while 1
? end
```

4. Observe the values in the `smtx` column. Note the increase in mutex spin locks.
5. Create a processor set and add processor 3 to it by executing the following commands:

```
oracle@host01:~$ su -
Password: cangetin
oracle@host01:~# psrset -c 3
created processor set 1
processor 3: was not assigned, now 1
```

What changed in the `mpstat` output window?

*After the processor set was created, the new set (Set 1) appears in the `mpstat` output. The size of Set 0 (`sze` field) decreases from 32 CPUs to 31 CPUs.*

*mpstat output example:*

**Before processor set 1 was created:**

```
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    0    0   31  306  171 8753   28   64 3344 4199 715982  23  27  0  50  32
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    0    0   30  303  173 6990   28   56 3457 3309 728566  24  27  0  49  32
```

**After processor set 1 was created:**

```
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
```

0	153	1	107	25	12	194	1	8	11	6	2391	0	1	0	99	31
1	27	0	37	336	183	60	0	2	3	1	895	0	0	0	99	1
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze																
0	0	0	36	117	36	7045	73	79	3646	3370	725453	31	38	0	31	31
1	0	0	45	241	138	0	0	0	4	0	0	0	0	0	100	1
...																

6. Run the `prset -i` and `psrinfo` commands to verify the results.

```
oracle@host01:~# psrset -i
user processor set 1: processor 3
oracle@host01:~# psrinfo
0 on-line since 11/10/2012 14:05:40
1 on-line since 11/10/2012 14:05:40
2 on-line since 11/10/2012 14:05:40
3 on-line since 11/10/2012 14:05:39
...
```

7. Create another set consisting of processor 2. Observe the `mpstat` output.

```
oracle@host01:~# psrset -c 2
created processor set 2
processor 2: was not assigned, now 2
```

*mpstat output example:*

**Before processor set 2 was created:**

SET	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl	sze
0	0	0	55	148	43	6571	94	113	3727	3107	727195	31	36	0	33	31
1	0	0	48	239	137	0	0	0	2	0	0	0	0	0	100	1

**After processor set 2 was created:**

SET	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl	sze
0	0	0	54	218	4	6039	204	32	3496	2812	722888	47	53	0	0	30
1	0	0	106	239	137	0	0	0	2	0	0	0	0	0	100	1
2	0	0	57	62	60	0	0	0	0	0	0	0	0	0	100	1
...																

8. Now add processor 1 to processor set 2. Observe the `mpstat` output.

```
oracle@host01:~# psrset -a 2 1
processor 1: was not assigned, now 2
```

*mpstat output example:*

**Before processor 1 was moved to processor set 2:**

0	0	0	54	218	4	6039	204	32	3496	2812	722888	47	53	0	0	30
1	0	0	106	239	137	0	0	0	2	0	0	0	0	0	100	1
2	0	0	57	62	60	0	0	0	0	0	0	0	0	0	100	1
...																

**After processor 1 was moved to processor set 2:**

SET	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl	sze
0	0	0	54	218	4	6039	204	32	3496	2812	722888	47	53	0	0	29
1	0	0	106	239	137	0	0	0	2	0	0	0	0	0	100	1

2	0	0	57	62	60	0	0	0	0	0	0	0	0	0	0	100	2
...																	

9. Bind the looping processes (performed in Step 3) in the two windows to the first processor set by typing the following command. Observe the `mpstat` output.

**Hint:** Use `prstat` to obtain the PIDs.

```
root@host01:~# prstat
  PID USERNAME  SIZE   RSS STATE   PRI NICE   TIME   CPU PROCESS/NLWP
  6609 oracle    9056K 6144K cpu22     0    4   0:03:50 3.1% csh/1
  6619 oracle    9024K 6080K cpu13     0    4   0:00:14 1.5% csh/1
  6552 oracle     130M   25M sleep    47    4   0:00:22 0.0% gnome-terminal/2
  6511 oracle     235M  134M sleep    47    4   0:01:28 0.0% java/44
    5 root         0K     0K sleep    99   -20   0:00:36 0.0% zpool-rpool/164
...
root@host01:~# psrset -b 1 6609 6619
process id 6609: was not bound, now 1
process id 6619: was not bound, now 1
```

*mpstat output example:*

```
Loops not yet bound to set 1:
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
0      0  0   1 102   2 238   98   0  14   0 410646  52  48  0   0  29
1      0  0   74 234  132   0   0   0   1   0   0   0   0  0 100   1
2      0  0   29  35   30   8   0   0   0   0   0   0   0  0 100   2
...

Loops now bound to set 1:
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
0      0  0   4   9   2 324   7   0   0   0   572   1   8  0  91  29
1      0  0   2 235  132  16  13   0   3   0 420209  53  47  0   0   1
2      0  0   8  61   57  18   0   0   0   0   0   0   0  0 100   2
...
```

10. Begin two new processes in the second set by executing the following commands. Observe the `mpstat` output window.

```
root@host01:~# psrset -e 2 dd if=/dev/zero of=/dev/null&
[2] 18633
root@host01:~# psrset -e 2 dd if=/dev/zero of=/dev/null&
[3] 18634
```

*mpstat output example:*

```
Both dd commands are now in set 2:
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
0      0  0   1 102   2 238   98   0  14   0 410646  52  48  0 100  29
1      0  0   0  11   0  10  10   0   0   0  98484  40  60  0   0   1
2      0  0  22  38  22   0  14   0   1   0 181503  67  33  0   0   2
...
```

11. Delete the second processor set by entering the following command. Observe the `mpstat` output window.

```
root@host01:~# psrset -d 2
```

What happened to the two `dd` processes? What happened to the `mpstat` output?

*The two `dd` processes were moved to processor set 0. The `mpstat` output is not showing set 2 anymore. The CPUs in processor set 2 were moved to processor set 0.*

*`mpstat` output example:*

**Before processor set 2 was deleted:**

SET	minf	mjf	xcal	intr	ithr	cs	ic	migr	smtx	srw	syscl	usr	sys	wt	idl	sz
0	0	0	1	102	2	238	98	0	14	0	410646	52	48	0	100	29
1	0	0	0	11	0	10	10	0	0	0	98484	40	60	0	0	1
2	0	0	22	38	22	0	14	0	1	0	181503	67	33	0	0	2

**After processor set 2 was deleted:**

SET	minf	mjf	xcal	intr	ithr	cs	ic	migr	smtx	srw	syscl	usr	sys	wt	idl	sz
0	0	0	291	966	114	625	14	7	25	0	260383	5	2	0	93	31
1	0	0	0	12	0	10	11	0	0	0	125203	39	61	0	0	1
...																

12. Run the `psrset -i` command to verify the results.

```
root@host01:~# psrset -i
user processor set 1: processor 3
```

13. Delete the remaining processor set (Set 1). Observe the `mpstat` output window.

```
root@host01:~# psrset -d 1
removed processor set 1
```

*`mpstat` output example:*

**Before processor set 1 was deleted:**

SET	minf	mjf	xcal	intr	ithr	cs	ic	migr	smtx	srw	syscl	usr	sys	wt	idl	sz
0	0	0	1	102	2	238	98	0	14	0	410646	52	48	0	100	30
1	0	0	0	11	0	10	10	0	0	0	98484	40	60	0	0	1
...																

**After processor set 1 was deleted:**

SET	minf	mjf	xcal	intr	ithr	cs	ic	migr	smtx	srw	syscl	usr	sys	wt	idl	sz
0	0	0	291	966	114	625	14	7	25	0	260383	5	2	0	87	32
...																

14. Terminate the other practice windows.

## Solution 7-7: Changing the Default Scheduling Class to FSS

### Overview

In this practice, you perform the following tasks to change the default scheduling class to Fair Share Scheduling (FSS):

- Adding some simple projects to the `/etc/project` file
- Changing the default scheduling class to FSS
- Adding some new tasks in each of the created projects
- Observing the CPU consumption of the compute-bound processes in these projects by using the `prstat -J` command

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task 1: Adding Projects

Perform the following steps:

1. Create some projects by entering:

```
root@host01:~# projadd -U root -K \
"project.cpu-shares=(priv,3,none)" payroll
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,5,none)" eng
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,2,none)" research
root@host01:~## projadd -U root -K \
"project.cpu-shares=(priv,0,none)" sales
```

2. Validate that the projects were added correctly by entering:

```
root@host01:~# projects -l
...
```

3. Change the default scheduling class to Fair Share Scheduling.

```
root@host01:~# dispadmin -d FSS
```

4. Make this configuration take effect immediately, without rebooting:

```
root@host01:~# priocntl -s -c FSS -i all
```

### Task 2: Adding New Tasks to Each Project

Perform the following steps:

1. Start an infinite loop in the sales project.

In a window, enter:

```
root@host01:~# newtask -p sales dd if=/dev/zero of=/dev/null&
```

- In a separate window, observe the distribution of CPU time by entering:

```
root@host01:~# prstat -JR
...
```

Leave the window that is running `prstat` open for the duration of this exercise.

Which process is consuming the bulk of the CPU time?

`_dd_`

What is its PID? `_20240_`

In the lower portion of the window, what are the current project IDs on the system? `__103 (sales) __10 (group.staff) __1 (user.root) __3 (default) __0 (system)_`

Which is consuming the bulk of the CPU? `_103 (sales)_`

How many processes are in that project (NPROC)? `_1_`

What is the current CPU% value for the sales project? `_3.1%_`

What does the CPU% value indicate? `__The sales project is consuming 3.1% of the total system CPU resources ((100/32 CPUs) x (1 NPROC)).__`

How many cpu-shares did that project get (see `/etc/project`)?

`_0_`

- Enter:

```
root@host01:~# newtask -p eng dd if=/dev/zero of=/dev/null
```

What new project ID is shown in `prstat`? `_101 (eng)_`

How many processes are in that project (NPROC)? `_1_`

Which project is getting most of the CPU? `_eng and sales projects are equal._`

- Start another infinite loop in the payroll and research projects by entering:

```
root@host01:~# newtask -p payroll dd if=/dev/zero of=/dev/null&
root@host01:~# newtask -p research dd if=/dev/zero \
of=/dev/null&
```

After a short time, note that all the projects are getting the same percentage of CPU time even though you assigned different share values to each project.

Why? `_There is still no contention between projects for CPU resources._`

- Now start an infinite loop in each project by entering:

```
root@host01:~# newtask -p research dd if=/dev/zero of=/dev/null&
[5] 20254
root@host01:~# newtask -p eng dd if=/dev/zero of=/dev/null&
[6] 20260
root@host01:~# newtask -p sales dd if=/dev/zero of=/dev/null&
[7] 20261
root@host01:~# newtask -p research dd if=/dev/zero of=/dev/null&
[8] 20262
root@host01:~# newtask -p payroll dd if=/dev/zero of=/dev/null&
```

How many processes are in that project (NPROC)? `_2_`

Are the project CPU percentage still equal? If the answer is Yes, repeat step 5.

**Note:** In the bash shell, you can use the up-arrow key to access command history.



At what point did the CPU shares start taking effect?

*You start seeing CPU contention when the sum of all the project `CPU%` values exceeds 100% and the sum of all project `NPROC` values exceeds the number of CPUs in the system.*

6. Repeat step 5 a few more times and observe how the assignment of CPU shares affects CPU% for each project.
7. When you have finished experimenting with the Fair Share Scheduler, kill the `dd` processes and revert to the TS scheduler.

```

root@host01:~# pkill dd
root@host01:~# dispadmin -d TS
root@host01:~# priocntl -s -c TS -i all
root@host01:~# ps -ace

```

PID	CLS	PRI	TTY	TIME	CMD
0	SYS	96	?	0:13	sched
5	SDC	99	?	8:32	zpool-rp
6	SDC	99	?	0:18	kmem_tas
1	TS	59	?	0:11	init
2	SYS	98	?	0:00	pageout
3	SYS	60	?	236:48	fsflush
7	SYS	60	?	0:48	intrd
8	SYS	60	?	0:32	vmtasks
11	TS	59	?	1:12	svc.star
13	TS	59	?	4:14	svc.conf
954	TS	59	?	0:00	sshd
64	TS	59	?	0:50	dlmgmtd
...					

## Solution 7-8: Determining CPU Information and Activity Using DTrace Toolkit Scripts

### Overview

In this practice, you use the various CPU-related DTrace scripts that are found in the DTrace Toolkit.

You observe CPU activity by using the following scripts:

- `cputypes.d` to list CPU types
- `cpuwalk.d` to measure the CPUs that a process runs on
- `dispqlen.d` to display dispatcher queue length by CPU
- `runocc.d` to display run queue occupancy by CPU
- `pridist.d` to display process priority distribution
- `cswstat.d` to display context-switch time statistics

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Task

Perform the following steps:

1. Enter:

```
root@host01:~# cputypes.d
...
```

How many CPUs are there? \_\_32\_\_

What processor group are they in? \_\_0\_\_

What are their clock speeds? \_\_1000\_\_

What is the LGRP column for? (See `man cputypes.d`.) \_\_0\_\_

2. Enter:

```
root@host01:~# cpuwalk.d 10
...
```

What do the value and count columns represent? (See `man cputypes.d`.)

*This program is for multi-CPU servers, and can help identify whether a process is running on multiple CPUs concurrently or not. The Y axis represents the CPUs and the X axis represents the distribution of the process.*

Which process has been executing the most? \_\_zpool-rpool\_\_

Which CPUs are being used? \_\_1, 6, 8, 12, 17, 20, 24\_\_

Are all the CPUs being used fairly equally? \_\_No\_\_

3. In a separate window, go to the `/opt/ora/course_files/lab7` directory and run the `while1` program in the background by entering:

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

4. Enter:

```
root@host01:~# cpuwalk.d 10
...
PID: 20329      CMD: while1

      value  ----- Distribution ----- count
      10 |                                           0
      11 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 9917
      12 |                                           0
```

Did the `while1` process run on all CPUs? \_\_No\_\_

How many times did it approximately run altogether? \_\_Once\_\_

5. Try running a few more `while1` processes in the background, and then run `cpuwalk.d` again.

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

```
root@host01:~# cpuwalk.d 10
...
```

What happened to the CPU distribution? \_\_Each while1 instance is running on a different CPU.\_\_

6. Kill the `while1` processes by entering:

```
root@host01:~# pkill while1
...
```

7. Enter:

```
root@host01:~# man dispqlen.d
...
```

What is its stability? \_\_unstable - walks private kernel structs\_\_

How many times a second does it sample the length of the dispatch queue? \_\_This script measures this activity by sampling at 1000 Hertz (1000 times per second) per CPU.\_\_

8. Enter:

```
root@host01:~# dispqlen.d
...
```

9. After the "Sampling... Hit Ctrl-C to end" message is displayed, count to five, and then press `^c`.

What is being displayed in the count column? \_\_Increasing sampling values\_\_

What are the run queue lengths? \_\_0 for all CPUs\_\_

10. Run a `while1` process for each CPU by entering (once for each CPU):

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

11. Enter:

```
root@host01:~# dispqlen.d

Sampling... Hit Ctrl-C to end.
^C
CPU 15
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0

CPU 16
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0

CPU 18
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0

CPU 19
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0

CPU 23
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0

CPU 30
      value  ----- Distribution ----- count
      < 0 |                                     0
      0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11302
      1 |                                     0
```

CPU 31			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11302
	1		0
CPU 0			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0
CPU 3			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0
CPU 6			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0
CPU 7			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0
CPU 8			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0
CPU 9			
	value	----- Distribution -----	count
	< 0		0
	0	@@	11303
	1		0

CPU 14

value	Distribution	count
< 0		0
0	@@	11303
1		0

CPU 17

value	Distribution	count
< 0		0
0	@@	11297
1		5
2		0

CPU 1

value	Distribution	count
< 0		0
0	@@	11298
1		5
2		0

CPU 22

value	Distribution	count
< 0		0
0	@@	11289
1		13
2		0

CPU 2

value	Distribution	count
< 0		0
0	@@	11275
1		28
2		0

CPU 20

value	Distribution	count
< 0		0
0	@@	11260
1		42
2		0

CPU 4

value	----- Distribution -----	count
< 0		0
0	@@@	11207
1		96
2		0

CPU 29

value	----- Distribution -----	count
< 0		0
0	@@@	11205
1		97
2		0

CPU 12

value	----- Distribution -----	count
< 0		0
0	@@@	11206
1		97
2		0

CPU 13

value	----- Distribution -----	count
< 0		0
0	@@@	11125
1	@	178
2		0

CPU 10

value	----- Distribution -----	count
< 0		0
0	@@@	11118
1	@	185
2		0

CPU 24

value	----- Distribution -----	count
< 0		0
0	@@@	11108
1	@	194
2		0

## CPU 28

```

value      ----- Distribution ----- count
< 0 |                                           0
  0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11108
  1 | @                                           194
  2 |                                           0

```

## CPU 25

```

value      ----- Distribution ----- count
< 0 |                                             0
  0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 11010
  1 | @                                           292
  2 |                                             0

```

## CPU 5

```

value    ----- Distribution ----- count
< 0 |                                           0
  0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 10944
  1 | @                                           359
  2 |                                           0

```

## CPU 11

```

value    ----- Distribution ----- count
< 0 |                                           0
  0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 10938
  1 | @                                           365
  2 |                                           0

```

## CPU 27

```

value      ----- Distribution ----- count
< 0 |                                           0
  0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 10905
  1 | @                                           397
  2 |                                           0

```

## CPU 21

```

value      ----- Distribution ----- count
< 0 |                                           0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 10874
    1 | @@                                          428
    2 |                                           0

```



CPU 26

value	----- Distribution -----	count
< 0		0
0	@@@	10824
1	@@	478
2		0

\*\*\*\*\*After while  
loop\*\*\*\*\*

root@host02:~# dispqlen.d  
Sampling... Hit Ctrl-C to end.  
^C

CPU 0

value	----- Distribution -----	count
< 0		0
0	@@@	7251
1		0

CPU 3

value	----- Distribution -----	count
< 0		0
0	@@@	7251
1		0

CPU 4

value	----- Distribution -----	count
< 0		0
0	@@@	7251
1		0

CPU 7

value	----- Distribution -----	count
< 0		0
0	@@@	7251
1		0

```

CPU 13
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 14
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 15
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 29
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 30
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 31
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    0

CPU 20
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7252
    1 |                                                    0

```

```

CPU 23
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7252
    1 |                                                    0

CPU 25
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7252
    1 |                                                    0

CPU 27
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7252
    1 |                                                    0

CPU 28
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7252
    1 |                                                    0

CPU 16
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7251
    1 |                                                    1
    2 |                                                    0

CPU 18
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7250
    1 |                                                    2
    2 |                                                    0

CPU 22
value  ----- Distribution ----- count
  < 0 |                                                    0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 7250

```

	1		2
	2		0
CPU 6			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7248
	1		3
	2		0
CPU 8			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7239
	1		12
	2		0
CPU 2			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7178
	1		73
	2		0
CPU 1			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7173
	1		78
	2		0
CPU 17			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7168
	1		83
	2		0
CPU 19			
	value	----- Distribution -----	count
	< 0		0
	0	@@	7167

	1		85
	2		0
CPU 24			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7159
	1	@	93
	2		0
CPU 5			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7150
	1	@	101
	2		0
CPU 9			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7150
	1	@	101
	2		0
CPU 21			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7149
	1	@	103
	2		0
CPU 12			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7123
	1	@	128
	2		0
CPU 26			
	value	----- Distribution -----	count
	< 0		0
	0	@@@	7067

1	@	185
2		0
CPU 11		
value	----- Distribution -----	count
< 0		0
0	@@@@	6945
1	@@	306
2		0
CPU 10		
value	----- Distribution -----	count
< 0		0
0	@@@@	6501
1	@@@@	750
2		0

After the “Sampling... Hit Ctrl-C to end” message is displayed, count to five, and then press ^c.

What happened to the run queue lengths? *The run queue length increased on the CPUs where the while1 process is running.* \_\_

12. In one window, enter:

```
root@host01:~# runocc.d
...
```

What does it show? *It shows the dispatcher run queue occupancy by CPU each second.*\_\_

13. After you have finished, kill the while1 process:

```
root@host01:~# pkill while1
...
```

**Note:** Remember that this would be a good way to check for CPU saturation.

14. Look at the man page for prdist.d.

```
root@host01:~# man prdist.d
...
```

Try running it.

```
root@host01:~# prdist.d
...
```

What does this program show?

*It shows the process that is on the CPU and its priority.*\_\_

15. Run a while1 program in the background by entering:

```
root@host01:~# /opt/ora/course_files/lab7/while1&
...
```

Note the PID.

```
root@host01:~# pgrep while1
...
```

16. Enter the following DTrace one-liner, let it run for several seconds, and then press **^C**.

```
root@host01:~# dtrace -s /usr/demo/dtrace/profpri.d
PID_of_while1
...
^C

while1
value  ----- Distribution ----- count
  < 0 |                                     0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 6116
   10 | @@@@@@@@                          1600
   20 | @@@@@@@                          1278
   30 | @@@@                            880
   40 | @@                             440
   50 | @@                             429
   60 |                                     0
```

Observe the `while1` entry. What priorities did `while1` spend most of its time on? `_0_`

17. Run the `pridist.d` DTrace script and observe the `while1` process.

```
root@host01:~# pridist.d
...
CMD: while1          PID: 20380

value  ----- Distribution ----- count
  < 0 |                                     0
    0 | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 3706
    5 |                                     0
   10 | @@@@@@@@@@                          1220
   15 |                                     0
   20 | @@@@@@@                          782
   25 |                                     0
   30 | @@@@                            560
   35 |                                     0
   40 | @@                             280
   45 |                                     0
   50 | @@                             273
   55 |                                     0
```

Notice that the `profpri.d` one-liner and the `pridist.d` script have similar outputs.

18. Try changing the `profpri.d` program to a one-liner.
- Use a predicate of `/execname == "while1"/` instead of `/pid == $1/`.
  - Use a step value of 5 instead of 10 in the call to `lquantize()` to allow a finer granularity of distribution.
  - Test your one-liner on the `while1` process. For example:

```
root@host01:~# dtrace -n 'profile-1001 /execname == "while1"/ {
@["while1 priority dist"] = lquantize(curlwpsinfo->pr_pri, 0,
100, 5)}' -c /opt/ora/course_files/lab7/while1

...
^C
while1 priority dist
```

value	----- Distribution -----	count
< 0		0
0	@@@@	12896
5		0
10	@@@@	3948
15		0
20	@@@@	3197
25		101
30	@@@	2095
35		79
40	@@	960
45		49
50	@@	911
55		39
60		0

19. Look at the man page for `cswstat.d`.

```
root@host01:~# man cswstat.d
...
```

20. Enter:

```
root@host01:~# cswstat.d
...
```

What is the typical amount of time spent in context switching (`CSWTIME(us)`)?

approximately 5000 us

What is the typical average time per context switch (`AVGTIME(us)`)?

9 us

21. Kill the `while1` processes when you have finished by entering:

```
root@host01:~# pkill while1
...
```



## Solution 7-9: Locating Hot Routines

---

### Overview

There is a program in the `/opt/ora/course_files/lab7` directory called `fault1`. In this practice, you identify the hot routines in this application by using `mpstat`, `prstat`, and `DTrace`.

**Note:** The output from the commands shown in this practice are examples. Your practice experience should be similar.

### Tasks

Perform the following steps:

1. In one window, run the program `fault1` by entering:

```
root@host01:~# /opt/ora/course_files/lab7/fault1
```

2. In another window, run:

```
root@host01:~# mpstat 5
```

Does it look like this load is CPU intensive? Yes

How many CPUs is it consuming? 1 CPU

3. Verify that it is a single-threaded application that is using `prstat` or `ps`.

Is it single-threaded? 1

Is it spending most of its time in user or kernel mode?

usr mode



# **Practices for Lesson 8: System Caches and Buses**

## **Chapter 8**

## Practices for Lesson 8: Overview

---

### Practices Overview

In these practices, you will:

- Calculate the cache-hit rate
- Observe file system caching
- Calculate the Cycles Per Instruction (CPI)

Solutions for each task in this practice are provided at the back of this guide.

## Practice 8-1: Calculating the Cache-Miss Rate

---

### Overview

In this practice, you estimate an increase in the overall response time by using cache hit rate and the given time costs of cache hits and misses.

Use the following values to represent the costs of hits and misses in a machine:

- Cache hit-cost = 2 nanoseconds
- Cache miss-cost = 50 nanoseconds

Use the following formula to estimate changes in execution time based on changes in the cache's hit rate:

$$\text{Execution time} = (\text{Hit-cost} * n) + (\text{Miss-cost} * (100 - n))$$

Where  $n$  is the cache-hit rate being achieved.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task

Perform these steps:

1. How much time do 100 data access operations take if the hit rate is 100 percent?  
\_\_\_\_\_
2. What is the cost in time if the hit rate is 99 percent?  
\_\_\_\_\_
3. What is the cost in time if the hit rate is 95 percent?  
\_\_\_\_\_
4. What is the cost in time if the hit rate is 90 percent?  
\_\_\_\_\_

## Practice 8-2: Observing File System Caching

### Overview

In this practice, you observe file system caching.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task

Perform these steps:

1. Change the directory to the `/opt/ora/course_files/lab8` directory.
2. To see the effects of the file system cache when reading, execute the `fsc1` script by using the following command:

```
root@host01:/opt/ora/course_files/lab8# sh ./fsc1.ksh
```

3. This program finds all files under the `/usr/include` directory.  
Press the **Return** key to run the `find` command a second and third time. After each run, timings are displayed. Record the timings below:

a. Find 1:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

b. Find 2:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

c. Find 3:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

Were there any major differences in the timings? Yes or no?

\_\_\_\_\_

If so, what were the major differences?

\_\_\_\_\_

Can you provide an explanation for the differences in timing?

\_\_\_\_\_

4. To see the effects on the file system cache when writing to files, execute the `fsc2.ksh` script by using the following command:

```
root@host01:/opt/ora/course_files/lab8# sh ./fsc2.ksh
```

5. The script is written to create the same file content three times by using the `cat` command. Press **Return** for each `cat` operation. Each time the operation runs, the timings are displayed. Record the timings below:

a. Copy 1:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

b. Copy 2:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

c. Copy 3:

Real time: \_\_\_\_\_

User time: \_\_\_\_\_

System time: \_\_\_\_\_

Were there any major differences in the timings? Yes or no?

\_\_\_\_\_

If so, what were the major differences?

\_\_\_\_\_

Can you provide an explanation for the differences in timing?

\_\_\_\_\_

## Practice 8-3: Calculating the CPI

### Overview

Many times, a memory performance problem is manifested by stalls to a memory bus or a CPU interconnect. Although these are not always directly measurable through low-level tools like `cpustat`, they can sometimes be inferred by measuring the number of cycles per instruction.

In this practice, you calculate the cycles per instruction (CPI).

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Tasks

1. Enter:

```
root@host01:~# cpustat -h
...
Note the events.
```

2. Open a new terminal window and enter:

```
root@host01:~$ man generic_events
...
```

Define the following events:

- `PAPI_br_cn`: Conditional branch instructions
- `PAPI_br_ins`: Branch instructions
- `PAPI_br_msp`: Conditional branch instructions mispredicted
- `PAPI_br_ntk`: Conditional branch instructions not taken
- `PAPI_br_prc`: Conditional branch instructions correctly predicted
- `PAPI_br_tkn`: Conditional branch instructions taken
- `PAPI_br_uct`: Unconditional branch instructions
- `PAPI_bru_idl`: Cycles branch units are idle
- `PAPI_btac_m`: Branch target address cache misses
- `PAPI_ca_cln`: Requests for exclusive access to clean cache line
- `PAPI_ca_inv`: Requests for cache invalidation
- `PAPI_ca_itv`: Requests for cache line intervention

Here are the platform-specific events for the T2000 server:

- `IC_miss sl`: Number of instruction cache (L1) misses. Greater than 7 percent is considered high.
- `DC_miss sl`: Number of data cache (L1) misses for loads (store misses are not included as the cache is write-through, non-allocating). Greater than 11 percent is considered high.
- `ITLB_miss`: Number of instruction TLB (translation buffer) miss trap taken (includes real\_translation misses). Greater than 0.001% is considered high.
- `DTLB_miss`: Number of data TLB miss trap taken (includes real\_translation misses). Greater than 0.005% is considered high.



- **L2\_imiss:** Number of secondary cache (L2) misses due to instruction cache requests. Greater than 2 percent is considered high.
3. Each of the 32 UltraSPARC T1 strands has a set of hardware performance counters that can be monitored by using the `cpustat` command. `cpustat` can collect two performance instrumentation counters (`pic0` and `pic1`) in parallel, the second always being the instruction count. For example, to collect iTLB misses and instruction counts for every strand on the chip, run:

```
root@host01:~# cpustat -c pic0=ITLB_miss,pic1=Instr_cnt,sys 1 10
```

time	cpu	event	pic0	pic1
1.001	0	tick	56	1037327
1.001	1	tick	14	596313
1.001	2	tick	50	645672
1.001	4	tick	540	1692547
1.001	5	tick	14	109569
1.002	6	tick	65	140162
1.001	7	tick	16	37696
...				

The performance counters indicate that each strand is executing 37K to 1M instructions per second. To determine how many instructions are executing per core, aggregate counts from four strands. Strands zero, one, two, and three are in the first core, strands four, five, six, and seven are in the second core, and so on. Performance counter `pic0` shows the instruction translation buffer misses.

## Solution 8-1: Calculating the Cache-Miss Rate

---

### Overview

In this practice, you estimate an increase in the overall response time by using cache-hit rate and the given time costs of cache hits and misses.

Use the following values to represent the costs of hits and misses in a machine:

- Cache hit-cost = 2 nanoseconds
- Cache miss-cost = 50 nanoseconds

Use the following formula to estimate changes in execution time based on changes in the cache's hit rate:

$$\text{Execution time} = (\text{Hit-cost} * n) + (\text{Miss-cost} * (100 - n))$$

Where  $n$  is the cache-hit rate being achieved.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task

Perform these steps:

1. How much time do 100 data access operations take if the hit rate is one hundred percent?  
*200 nanoseconds.*
2. What is the cost in time if the hit-rate is 99 percent?  
*248 nanoseconds, 24% longer.*
3. What is the cost in time if the hit-rate is 95 percent?  
*440 nanoseconds, 120% longer.*
4. What is the cost in time if the hit-rate is 90 percent?  
*680 nanoseconds, 340% longer.*

*If you know the time cost, but do not know the hit rate, subtract the time cost from the cost of 100% misses. Then divide this difference by the difference between one miss and one hit. If your total time cost for 100 accesses was 600 nanoseconds, the hit rate can be calculated as follows:*

$$(100 * 50) - 600 / 50 - 2 = 91.667\% \text{ hit rate.}$$

## Solution 8-2: Observing File System Caching

### Overview

In this practice, you observe file system caching.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task

Perform these steps:

1. Change the directory to the `/opt/ora/course_files/lab8` directory.
2. To see the effects of the file system cache when reading, execute the `fsc1` script by using the following command:

```
root@host01:/opt/ora/course_files/lab8# sh ./fsc1.ksh
```

3. This program finds all files under the `/usr/include` directory.  
Press the **Return** key to run the `find` command a second and third time. After each run, timings are displayed. Record the timings below:

a. Find 1:

Real time: `_0.806315512_`

User time: `_0.113448976_`

System time: `_0.690917144_`

b. Find 2:

Real time: `_0.319212584_`

User time: `_0.092365456_`

System time: `_0.225460428_`

c. Find 3:

Real time: `_0.298047260_`

User time: `_0.091939452_`

System time: `_0.204484156_`

Were there any major differences in the timings? Yes or no?

Yes.

If so, what were the major differences?

*The real and system times for the first run are longer than the time values for the second and third runs.*

Can you provide an explanation for the differences in timing?

*The first time the find ran, the directory and metadata had to be read from disk. On subsequent viewings, these data are from caches in main memory.*

4. To see the effects on the file system cache when writing to files, execute the `fsc2.ksh` script by using the following command:

```
root@host01:/opt/ora/course_files/lab8# sh ./fsc2.ksh
```

5. The script is written to create the same file content three times by using the `cat` command. Press **Return** for each `cat` operation. Each time the operation runs, timings will be displayed. Record the timings below:
- Copy 1:  
Real time: 0.010057344  
User time: 0.002803220  
System time: 0.005675084
  - Copy 2:  
Real time: 0.009754348  
User time: 0.002747368  
System time: 0.005562404
  - Copy 3:  
Real time: 0.010525032  
User time: 0.002761380  
System time: 0.005736856
- Were there any major differences in the timings? Yes or no?
- Yes.
- If so, what were the major differences?
- The major difference should be in the real time and system time for the first viewing compared to the equivalent values for the second and third viewings.*
- Can you provide an explanation for the differences in timing?
- The first time the file was copied, the contents of the file had to be read from disk. The second and third copy operations were reading the file content from the file system cache.*

## Solution 8-3: Calculating the CPI

### Overview

Many times, a memory performance problem is manifested by stalls to a memory bus or a CPU interconnect. Although these are not always directly measurable through low-level tools like `cpustat`, they can sometimes be inferred by measuring the number of cycles per instruction.

In this practice, you calculate the cycles per instruction (CPI).

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Tasks

1. Enter:

```
root@host01:~# cpustat -h
...
Note the events.
```

2. Open a new terminal window and enter:

```
root@host01:~$ man generic_events
...
```

Define the following events:

- `PAPI_br_cn`: Conditional branch instructions
- `PAPI_br_ins`: Branch instructions
- `PAPI_br_msp`: Conditional branch instructions mispredicted
- `PAPI_br_ntk`: Conditional branch instructions not taken
- `PAPI_br_prc`: Conditional branch instructions correctly predicted
- `PAPI_br_tkn`: Conditional branch instructions taken
- `PAPI_br_uct`: Unconditional branch instructions
- `PAPI_bru_idl`: Cycles branch units are idle
- `PAPI_btac_m`: Branch target address cache misses
- `PAPI_ca_cln`: Requests for exclusive access to clean cache line
- `PAPI_ca_inv`: Requests for cache invalidation
- `PAPI_ca_itv`: Requests for cache line intervention

Here are the platform-specific events for the T2000 server:

- `IC_miss sl`: Number of instruction cache (L1) misses. Greater than 7 percent is considered high.
- `DC_miss sl`: Number of data cache (L1) misses for loads (store misses are not included as the cache is write-through, non-allocating). Greater than 11 percent is considered high.
- `ITLB_miss`: Number of instruction TLB (translation buffer) miss trap taken (includes real\_translation misses). Greater than 0.001% is considered high.
- `DTLB_miss`: Number of data TLB miss trap taken (includes real\_translation misses). Greater than 0.005% is considered high.

- **L2\_iss:** Number of secondary cache (L2) misses due to instruction cache requests. Greater than 2 percent is considered high.
  - **L2\_dmiss\_ld:** Number of secondary cache (L2) misses due to data cache load requests. Greater than 2 percent is considered high.
3. Each of the 32 UltraSPARC T1 strands has a set of hardware performance counters that can be monitored by using the `cpustat` command. `cpustat` can collect two performance instrumentation counters (`pic0` and `pic1`) in parallel, the second always being the instruction count. For example, to collect iTLB misses and instruction counts for every strand on the chip, run:

```
root@host01:~# cpustat -c pic0=ITLB_miss,pic1=Instr_cnt,sys 1 10
```

time	cpu	event	pic0	pic1
1.001	0	tick	56	1037327
1.001	1	tick	14	596313
1.001	2	tick	50	645672
1.001	4	tick	540	1692547
1.001	5	tick	14	109569
1.002	6	tick	65	140162
1.001	7	tick	16	37696
...				

The performance counters indicate that each strand is executing 37K to 1M instructions per second. To determine how many instructions are executing per core, aggregate counts from four strands. Strands zero, one, two, and three are in the first core, strands four, five, six, and seven are in the second core, and so on. Performance counter `pic0` shows the instruction translation buffer misses.

# **Practices for Lesson 9: System Memory**

## **Chapter 9**

## Practices for Lesson 9: Overview

---

### Practices Overview

In these practices, you will:

- Monitor memory consumption
- Monitor memory consumption with the DTrace Toolkit

Solutions for each task in this practice are provided at the back of this guide.



## Practice 9-1: Monitoring Memory Consumption

### Overview

This practice simulates depletion of system memory, so that you can observe the resultant scanning and paging behavior. You lock pages in main memory and add files to the `/tmp` directory, which is backed by the swap device. This action simulates the increase in memory usage.

You also modify the values of the `lotsfree` and `slowsan` kernel tuning parameters.

First, you document the current values of several memory tuning parameters.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

Perform the following steps to monitor memory consumption:

1. Run the following script from the `/opt/ora/course_files/lab9` directory:

```
root@host01:/opt/ora/course_files/lab9# ./paging.sh
```

2. Write down the values of the kernel tuning parameters. The values of the kernel tuning parameters vary depending on the system.

lotsfree: \_\_\_\_\_ pages  
 desfree: \_\_\_\_\_ pages  
 minfree: \_\_\_\_\_ pages  
 slowsan: \_\_\_\_\_ pages  
 fastscan: \_\_\_\_\_ pages  
 handspreadpages: \_\_\_\_\_ pages  
 physmem: \_\_\_\_\_ pages  
 pagesize: \_\_\_\_\_ bytes

3. Convert the value of the `lotsfree` kernel tuning parameter from pages to bytes. You will need the byte value (in decimal notation) later in the exercise.

\_\_\_\_\_ Kbytes

4. In a separate terminal window, run the `vmstat 3` command.

```
root@host01:/opt/ora/course_files/lab9# vmstat 3
```

Keep the window in clear view for monitoring.

What information is shown in the following fields:

w: \_\_\_\_\_  
 swap: \_\_\_\_\_  
 free: \_\_\_\_\_  
 de: \_\_\_\_\_  
 sr: \_\_\_\_\_

**Note:** On a system with 8-GB memory installed, you should see approximately 6 GB in the memory `free` field.

5. To put a load on memory, you do not want the `mem_hog` program to run out of swap space before filling up memory. For this practice, the amount of swap space should be set to at least double the amount of memory on your system. To check how much swap space is configured on your system, enter:

```
root@host01:/opt/ora/course_files/lab9# cd ~
root@host01:~# swap -l
```

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	4194288	4194288

The column labeled blocks is the number of 512-byte sectors for each swap area, and should be summed for all the listed swap areas. This can be also seen with the `df -h` command. The amount of main memory on your system can be shown by entering:

```
root@host01:~# prtconf | grep Memory
Memory size: 16256 Megabytes
```

If the amount of swap space is less than two times the amount of main memory, the easiest way to increase swap space is by recreating the swap volume. For example, if your memory size is 16 GB, you want the total amount of swap to be 32 GB.

If the current amount of swap space is not large enough, re-create the swap volume:

```
root@host01:~# swap -d /dev/zvol/dsk/rpool/swap
root@host01:~# zfs volsize=32G rpool/swap
root@host01:~# swap -a /dev/zvol/dsk/rpool/swap
root@host01:~# swap -l
```

swapfile	dev	swaplo	blocks	free
/dev/zvol/dsk/rpool/swap	300,2	16	67108848	67108848

6. To load memory, you will use a program called `mem_hog`, which is in the `/opt/ora/course_files/lab9` directory, to lock down significant chunks of memory. The `mem_hog` program takes the number of megabytes of memory you want to lock down as an argument.

For example, if you have 12 GB of free memory (see the `free` field in the `vmstat` window), you want to lock down all but approximately 1.5 GB:

```
root@host01:~# cd /opt/ora/course_files/lab9
root@host01:/opt/ora/course_files/lab9# ./mem_hog 10500
```

While `mem_hog` is locking memory, watch the swap and free columns in `vmstat`. They should be going down.

You may have to run the `mem_hog` program a few times until the amount of `freemem` is between 1 and 2 GB. If, on the first try, `freemem` is not low enough, enter `q` to quit. On the next try, increment the `mem_hog` program by 500 (0.5 GB). Do this each time you run it until you get the desired results.

**Note:** Do not try to lock down all or more than the amount of memory you have. Doing so could hang the system. If you hang the system, log in to the SC and run the `reset` command. This will reboot the system. After the system reboots (this may take a few minutes) log back and start this task again. Only this time, be less aggressive with the memory lockdown. Contact your instructor for the SC login credentials.

7. Start filling swap space with files. Do this by opening a new window to run the `filltmp.sh` script, which is in the `/opt/ora/course_files/lab9` directory:

```
root@host01:/opt/ora/course_files/lab9# ./filltmp.sh
...
The script will prompt you to starting filling the /tmp directory with files.
Press RETURN to add 7G of space to tmp: <Return>
Press RETURN and observer the free, de, and sr fields in the vmstat window.
After a few minutes, the script will then prompt you to add more files to the /tmp
directory to consume more memory:
Now press RETURN to add 6.5G of space to tmp: <Return>
Run the script for a few more minutes until you see the message:
Press RETURN to clean tmp:
Note: DO NOT press Enter at this time. The /tmp directory needs to stay filled for this
entire exercise.
```

8. Note the reports from the `vmstat` command. Are there any changes to the scan rate?

\_\_\_\_\_

What could cause the scan rate to change?

9. If the output of the `vmstat w` column indicates that processes or LWPs were swapped out, you can get a list of processes that had swapped out LWPs by running the DTrace script `swappedout.d`. Open another window and enter the following command from the `/opt/ora/course_files/lab9` directory:

**Note:** This script takes a minute or two to run.

```
root@host01:/opt/ora/course_files/lab9# ./swappedout.d
```

NAME	PID	SWAPCNT	LWPCNT
svc.startd	11	6	12
svc.configd	13	3	32
netcfgd	44	1	5
dlnmgmt	69	1	14
ipmgmt	78	3	6
pfexecd	116	1	3
syseventd	254	15	17
rad	270	6	8
dbus-daemon	337	1	1
devfsadm	613	3	7
kcfd	1114	1	5
drd	1218	1	2
nwamd	1446	3	9
picld	1513	3	9
nscd	1571	1	36
iscsid	1604	1	2
console-kit-daem	1625	2	2
rmvolmgr	1689	1	1
rpcbind	1702	1	1

automountd	1731	1	2
automountd	1733	1	5
fmd	1771	24	37
cupsd	1783	1	1
in.ndpd	1812	1	1
auditd	1841	2	5
login	1842	1	1
devchassisd	1866	2	4
gdm-binary	1896	2	2
smtp-notify	1923	1	3
nfsmapid	1980	1	20
statd	1981	1	1
sshd	2046	1	1
inetd	2081	1	4
zoneproxyd	3356	2	13
...			

The `swappedout.d` script gives you the name of the process, its PID number, the number of LWPs in that process that were swapped (`SWAPCNT`), and the number of LWPs in that process (`LWPCNT`). If you want, you can swap some of those processes back in by waking them with a signal.

10. To do this in one window, run `vmstat -S 2`. In another window, enter:

```
root@host01:/opt/ora/course_files/lab9# kill -STOP PID
...
```

When you do this, the count in the `w` column of `vmstat` will typically get decremented.

11. So that the process is not left in a stopped state, you should then enter:

```
root@host01:/opt/ora/course_files/lab9# kill -CONT PID
...
```

You do not need to do this to swap in the processes. They will be swapped in by the swapper when they need to run.

12. If you did get your system to swap, you can check to see whether it was soft-swapping or hard-swapping with `mdb`. Enter:

```
root@host01:/opt/ora/course_files/lab9# mdb -k
...
```

13. To check for hard-swapping, enter:

```
> hardswap/D
```

Describe hard-swapping:

---



---



---

Was your system hard-swapping? \_\_\_\_\_

14. To check for soft-swapping, enter:

```
> softswap/D
```

Describe soft-swapping:

---

---

---

Was your system soft-swapping? \_\_\_\_\_

What was the number? \_\_\_\_\_

15. Terminate the `memhog` program by entering `q`.  
16. Finish the `filltmp.sh` script to clean up the `/tmp` directory:

Press RETURN to clean /tmp: <RETURN>

You will be using this technique to load memory to test some of the other tools. Quit out of the loads and tools you have running.

## Practice 9-2: Monitoring Memory Consumption with the DTrace Toolkit

### Overview

In this practice, you use the same programs to load memory as in Practice 9-1, but you use some of the tools from the DTrace Toolkit to gather more information.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Tasks

Perform the following steps:

1. In one window, run:

```
root@host01:~# vmstat 3
...
```

Try to keep this window visible throughout the lab.

2. In another window, enter:

```
root@host01:~# man anonpgpid.d
...
```

What does this script give you?

What is the *STABILITY*? \_\_\_\_\_

In the output fields, what is in the D column?

Which processes would be doing the writes to swap?

3. Try running it by entering:

```
root@host01:~# anonpgpid.d
Tracing... Hit Ctrl-C to end.
^C
  PID CMD                D BYTES
  978 fmd                  R 40960
 5285 gnome-terminal      R 40960
 3959 nautilus            R 65536
 3531 gnome-settings-d    R 163840
```

What kind of activity did you see? \_\_\_\_\_

4. In a separate window, make sure you are in the `/opt/ora/course_files/lab9` directory and use the `mem_hog` process to lock half your memory.
5. Run `anonpgpid.d` again.
6. In a separate window, make sure you are in `/opt/ora/course_files/lab9` and run `filltmp`.

7. Watch the activity in the `vmstat` window. After the system has been paging for a while, enter a `^c` in the window in which you are running `anonpgpid.d`.

What kind of output do you see now? \_\_\_\_\_

Has pageout run? \_\_\_\_\_

What was in the `D` column for pageout? \_\_\_\_\_

8. Quit `filltmp`, and let it clean up the temporary files.

9. Enter:

```
root@host01:~# man minfbypid.d
...
```

What does this script give you?

What is its stability? \_\_\_\_\_

10. Enter:

```
root@host01:~# minfbypid.d
...
```

Wait several seconds and type in a `^c`.

What was causing most of the minor faults? \_\_\_\_\_

11. Configure the `load5.sh` program and run it from the lab directory by entering:

```
root@host01:~# cp /usr/dict/words \
/opt/ora/course_files/test_files
root@host01:~# cd /opt/ora/course_files/lab9
root@host01:/opt/ora/course_files/lab9# vi load5.sh
...
# ./randomwr >>tmp & ### Comment out this line.
...
:wq!
root@host01:/opt/ora/course_files/lab9# ./load5.sh
```

12. Run `minfbypid.d` again.

```
root@host01:~# minfbypid.d
...
```

13. Watch the `vmstat` window. After it has made the system page for a while, enter `^c` in the `minfbypid.d` window.

What processes have accrued the most minor faults now?

\_\_find\_\_

\_\_wc\_\_

\_\_nm\_\_

14. While `load5` is still running, enter:

```
root@host01:~# minfbypid.d
...
```

15. Let it run for several seconds and press ^c.

What is different about this version? \_\_*The version gives you a summary of processes causing minor faults.*\_\_

Note that the last two toolkit scripts were just one-liners, so they can be easily modified if you want to customize them.

16. Using the same procedure, test the `pgpginbypid.d` and the `pgpginbyproc.d` scripts.

What do they give you?

---

---

17. If there are `mem_hog` processes still out there, go ahead and exit out of them.

18. Try using the same procedure for the `vmbypid.d` script.

Use the `load5.sh` script, `mem_hog`, and `filltmp` to load the system.

What does `vmbypid.d` give you?

---

When you are finished, make sure all the memory loads have exited.



## Solution 9-1: Monitoring Memory Consumption

### Overview

This practice simulates depletion of system memory, so that you can observe the resultant scanning and paging behavior. You lock pages in main memory and add files to the `/tmp` directory, which is backed by the swap device. This action simulates the increase in memory usage.

You also modify the values of the `lotsfree` and `slowscan` kernel tuning parameters.

First, you document the current values of several memory tuning parameters.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

Perform the following steps to monitor memory consumption:

1. Run the following script from the `/opt/ora/course_files/lab9` directory:

```
root@host01:/opt/ora/course_files/lab9# ./paging.sh
```

2. Write down the values of the kernel tuning parameters. The values of the kernel tuning parameters vary depending on the system.

`lotsfree`: \_\_31875\_\_ pages

`desfree`: \_\_15937\_\_ pages

`minfree`: \_\_7968\_\_ pages

`slowscan`: \_\_100\_\_ pages

`fastscan`: \_\_106169\_\_ pages

`handspreadpages`: \_\_106169\_\_ pages

`physmem`: \_\_2041178\_\_ pages

`pagesize`: \_\_8192\_\_ bytes

3. Convert the value of the `lotsfree` kernel tuning parameter from pages to bytes. You will need the byte value (in decimal notation) later in the exercise.

*The value for this varies depending on the system. Multiply the value returned from the script by 8192 (pagesize).*

*You can then convert the value to KB by dividing the result by 1024.*

*For example:*

$31875 \text{ (lotsfree)} \times 8192 \text{ (pagesize)} = 261120000$

$261120000 / 1024 = 255,000 \text{ KB}$

4. In a separate terminal window, run the `vmstat 3` command.

```
root@host01:~# vmstat 3
```

Keep the window in clear view for monitoring.

What information is shown in the following fields:

`w`: \_\_ *The number of swapped out lightweight processes (LWPs) that are waiting for processing resources to finish.* \_\_

`swap`: \_\_ *The available swap space (in Kbytes).* \_\_

free: \_\_ *The size of the free list (in Kbytes).*\_\_

de: \_\_ *The anticipated short-term memory shortfall (Kbytes).*\_\_

sr: \_\_ *Pages scanned by clock algorithm.*\_\_

**Note:** On a system with 8-GB memory installed, you should see approximately 6 GB in the memory free field.

- To put a load on memory, you do not want the `mem_hog` program to run out of swap space before filling up memory. For this practice, the amount of swap space should be set to at least double the amount of memory on your system. To check how much swap space is configured on your system, enter:

```
root@host01:~# swap -l
swapfile                                dev      swaplo    blocks      free
/dev/zvol/dsk/rpool/swap 300,2          16  4194288  4194288
```

The column labeled blocks is the number of 512-byte sectors for each swap area, and should be summed for all the listed swap areas. This can be also seen with the `df -h` command. The amount of main memory on your system can be shown by entering:

```
root@host01:/opt/ora/course_files/lab9# prtconf | grep Memory
Memory size: 16256 Megabytes
```

If the amount of swap space is less than two times the amount of main memory, the easiest way to increase swap space is by re-creating the swap volume. For example, if your memory size is 16 GB, you want the total amount of swap to be 32 GB.

If the current amount of swap space is not large enough, re-create the swap volume:

```
root@host15:~# swap -d /dev/zvol/dsk/rpool/swap
root@host15:~# zfs volsize=32G rpool/swap
root@host15:~# swap -a /dev/zvol/dsk/rpool/swap
root@host15:~# swap -l
swapfile                                dev      swaplo    blocks      free
/dev/zvol/dsk/rpool/swap 300,2          16  67108848  67108848
```

- To load memory, you are going to use a program called `mem_hog`, which is in the `/opt/ora/course_files/lab9` directory, to lock down significant chunks of memory. The `mem_hog` program takes the number of megabytes of memory you want to lock down as an argument.

For example, if you have 12 GB of free memory (see the free field in the `vmstat` window), you want to lock down all but approximately 1.5 GB:

```
root@host01:/opt/ora/course_files/lab9# ./mem_hog 10500
```

While `mem_hog` is locking memory, watch the swap and free columns in `vmstat`. They should be going down.

You may have to run the `mem_hog` program a few times until the amount of `freemem` is between 1 and 2 GB. If, on the first try, `freemem` is not low enough, enter `q` to quit. On the next try, increment the `mem_hog` program by 500 (0.5 GB). Do this each time you run it until you get the desired results.

**Note:** Do not try to lock down all or more than the amount of memory you have. Doing so could hang the system. If you hang the system, log in to the SC and run the `reset` command. This will reboot the system. After the system has reboot (this may take a few

minutes) log back and start this task again. Only this time, be less aggressive with the memory lockdown. Contact your instructor for the SC login credentials.

7. Start filling swap space with files. Do this by opening a new window to run the `filltmp.sh` script, which is in the `/opt/ora/course_files/lab9` directory:

```
root@host01:/opt/ora/course_files/lab9# ./filltmp.sh
...
The script will prompt you to starting filling the /tmp directory with files.
Press RETURN to add 7G of space to tmp: <Return>
Press RETURN and observer the free, de, and sr fields in the vmstat window.
After a few minutes, the script will then prompt you to add more files to the /tmp
directory to consume more memory:
Now press RETURN to add 6.5G of space to tmp: <Return>
Run the script for a few more minutes until you see the message:
Press RETURN to clean tmp:
Note: DO NOT press enter at this time. The /tmp directory needs to stay filled for this
entire exercise.
```

8. Note the reports from the `vmstat` command. Are there any changes to the scan rate?  
Yes

What could cause the scan rate to change?

*The scanner starts scanning when free memory is lower than `lotsfree` number of pages free plus a small buffer factor, deficit. The scanner starts scanning at a rate of `slowscan` pages per second at this point and gets faster as the amount of free memory approaches zero. The system parameter `lotsfree` is calculated at startup as 1/64th of memory, and the parameter deficit is either zero or a small number of pages set by the page allocator at times of large memory allocation to let the scanner free a few more pages above `lotsfree` in anticipation of more memory requests.*

9. If the output of the `vmstat w` column indicates that processes or LWPs were swapped out, you can get a list of processes that had swapped out LWPs by running the DTrace script `swappedout.d`. Open another window and enter the following command from the `/opt/ora/course_files/lab9` directory:

**Note:** This script takes a minute or two to run.

```
root@host01:/opt/ora/course_files/lab9# ./swappedout.d
```

NAME	PID	SWAPCNT	LWPCNT
svc.startd	11	6	12
svc.configd	13	3	32
netcfgd	44	1	5
dlnmgmt	69	1	14
ipmgmt	78	3	6
pfexecd	116	1	3
syseventd	254	15	17
rad	270	6	8
dbus-daemon	337	1	1
devfsadm	613	3	7
kcfd	1114	1	5
drd	1218	1	2

nwamd	1446	3	9
picld	1513	3	9
nsd	1571	1	36
iscsid	1604	1	2
console-kit-daem	1625	2	2
rmvolmgr	1689	1	1
rpcbind	1702	1	1
automountd	1731	1	2
automountd	1733	1	5
fmd	1771	24	37
cupsd	1783	1	1
in.ndpd	1812	1	1
auditd	1841	2	5
login	1842	1	1
devchassisd	1866	2	4
gdm-binary	1896	2	2
smtp-notify	1923	1	3
nfsmapid	1980	1	20
statd	1981	1	1
sshd	2046	1	1
inetd	2081	1	4
zoneproxyd	3356	2	13
...			

The `swappedout.d` script gives you the name of the process, its PID number, the number of LWPs in that process that were swapped (`SWAPCNT`), and the number of LWPs in that process (`LWPCNT`). If you want, you can swap some of those processes back in by waking them with a signal.

10. To do this in one window, run `vmstat -S 2`. In another window, enter:

```
root@host01:/opt/ora/course_files/lab9# kill -STOP PID
...
```

When you do this, the count in the `w` column of `vmstat` typically gets decremented.

11. So that the process is not left in a stopped state, you should then enter:

```
root@host01:/opt/ora/course_files/lab9# kill -CONT PID
...
```

You do not need to do this to swap in the processes. They will be swapped in by the swapper when they need to run.

12. If you did get your system to swap, you can check to see whether it was soft-swapping or hard-swapping with `mdb`. Enter:

```
root@host01:/opt/ora/course_files/lab9# mdb -k
...
```

13. To check for hard-swapping, enter:

```
> hardswap/D
```

Describe hard-swapping:

*During hard-swapping, the kernel is requested to unload all modules and cache memory that are not currently active. Then, processes are swapped out until the desired amount of free memory is available.*

*Hard-swapping occurs when all the following three conditions are true:*

- *At least two processes are on the run queue, waiting for CPU.*
- *Free memory is less than the `desfree` parameter on average for more than 30 seconds.*
- *The sum of pageins and pageout operations exceeds `maxpgio`, or `freemem` is less than `minfree`.*

Was your system hard-swapping? \_No\_

14. To check for soft-swapping, enter:

```
> softswap/D
```

Describe soft-swapping:

*Soft swapping occurs when free memory is less than `desfree` for 5 seconds or more. The swapper, a process called the `sched`, swaps out processes that have been sleeping longer than `maxslp` (20 seconds).*

Was your system soft-swapping? \_Yes\_

What was the number? \_\_443\_\_

15. Terminate the `memhog` program by entering `q`.

16. Finish the `filltmp.sh` script to clean up the `/tmp` directory:

```
Press RETURN to clean /tmp: <RETURN>
```

You will be using this technique to load memory to test some of the other tools. Quit out of the loads and tools you have running.

## Solution 9-2: Monitoring Memory Consumption with the DTrace Toolkit

### Overview

In this practice, you use the same programs to load memory as in Practice 9-1, but you use some of the tools from the DTrace Toolkit to gather more information.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Tasks

Perform the following steps:

1. In one window, run:

```
root@host01:~# vmstat 3
...
```

Try to keep this window visible throughout the lab.

2. In another window, enter:

```
root@host01:~# man anonpgpid.d
...
```

What does this script give you?

*\_\_This script helps identify which processes are affected by a system with low memory, which is paging to the physical swap device. A report of the process on the CPU when paging occurred is printed. \_\_*

What is the *STABILITY*? *\_\_Unstable\_\_*

In the output fields, what is in the D column?

*\_\_Direction, Read or Write\_\_*

Which processes would be doing the writes to swap?

3. Try running it by entering:

```
root@host01:~# anonpgpid.d
Tracing... Hit Ctrl-C to end.
^C

  PID CMD                D BYTES
   978 fmd                R 40960
  5285 gnome-terminal    R 40960
  3959 nautilus          R 65536
  3531 gnome-settings-d  R 163840
```

What kind of activity did you see? *\_\_Reads\_\_*

4. In a separate window, make sure you are in the `/opt/ora/course_files/lab9` directory and use the `mem_hog` process to lock half your memory.
5. Run `anonpgpid.d` again.
6. In a separate window, make sure you are in the `/opt/ora/course_files/lab9` directory and run `filltmp`.

7. Watch the activity in the `vmstat` window. After the system has been paging for a few minutes, enter a `^c` in the window you are running `anonpgpid.d`.

What kind of output do you see now? \_\_Reads and Writes\_\_

Has pageout run? \_\_Yes\_\_

What was in the D column for pageout? \_\_1748271104\_\_

8. Quit `filltmp`, and let it clean up the temporary files.

9. Enter:

```
root@host01:~# man minfbypid.d
...
```

What does this script give you?

\_\_minfbypid.d is a DTrace one-liner to a report the number of minor faults by process name. \_\_

What is its stability? \_\_Stable\_\_

10. Enter:

```
root@host01:~# minfbypid.d
Tracing... Hit Ctrl-C to end.
^C
  PID CMD                      MINFAULTS
  628 nwamd                      1
 4236 nwam-manager              1
 5063 svc.configd              1
 5285 gnome-terminal           1
 3793 metacity                  2
 5062 svc.configd              2
 6652 vmstat                    2
 6807 minfbypid.d              10
 3959 nautilus                  26
```

Wait several seconds and type in a `^c`.

Who was causing most of the minor faults? \_\_nautilus\_\_

11. Configure the `load5.sh` program and run it from the lab directory by entering:

```
root@host01:~# cp /usr/dict/words \
/opt/ora/course_files/test_file
root@host01:~# cd /opt/ora/course_files/lab9
root@host01:/opt/ora/course_files/lab9# vi load5.sh
...
# ./randomwr >>tmp & ### Comment out this line.
...
# ./randomwr >>tmp & ### Comment out this line.
...
:wq!
root@host01:/opt/ora/course_files/lab9# ./load5.sh
```

12. While `load5.sh` is running, run `minfbypid.d` again.

```
root@host01:~# minfbypid.d
...
```

13. Watch the `vmstat` window. After it has made the system page for a while, enter `^c` in the `minfbypid.d` window.

What processes have accrued the most minor faults now?

\_\_find\_\_

\_\_wc\_\_

\_\_nm\_\_

14. While `load5` is still running, enter:

```
root@host01:~# minfbypid.d
...
```

15. Let it run for several seconds and press `^c`.

What is different about this version? \_\_The version gives you a summary of processes causing minor faults.\_\_

Note that the last two toolkit scripts were just one-liners, so they can be easily modified if you want to customize them.

16. Using the same procedure, test the `pgpginbypid.d` and `pgpginbyproc.d` scripts.

What do they give you?

\_\_pgpginbypid.d reports the number of pages paged in from the disks by process ID. This is an indicator that processes are reading from the disks.\_\_

\_\_pgpginbyproc.d is a DTrace used one-liner to report the number of pages paged in by process name.\_\_

17. If there are `mem_hog` processes still out there, go ahead and exit out of them.

18. Try using the same procedure for the `vmbypid.d` script.

Use the `load5.sh` script, `mem_hog`, and `filltmp` to load the system.

What does `vmbypid.d` give you?

\_\_vmbypid.d prints Virtual Memory statistics by process.\_\_

When you are finished, make sure all the memory loads have exited.



# **Practices for Lesson 10: Disk I/O and ZFS File System**

## **Chapter 10**

## Practices for Lesson 10: Overview

---

### Practices Overview

In these practices, you will:

- Work with the Oracle I/O Numbers Calibration Tool (ORION)
- Monitor disk I/O activity
- Tune ZFS

Solutions for each task in this practice are provided at the back of this guide.

## Practice 10-1: Exploring ORION

---

### Overview

In this practice, you explore the capabilities of the Oracle I/O Numbers Calibration Tool (ORION).

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

Perform the following steps to explore ORION:

1. Change to the `/opt/ora/software` directory.
2. Unzip the `orion_solaris_sparc64.gz` package.
3. Rename the `orion_solaris_sparc64` file to `orion` and make it executable.
4. Run the `format` command to determine what disks are configured in the system.
5. Run `zpool status` to determine which of the configured disks is currently in use.
6. Using a text editor, create a file named `mytest.lun`. Add one unused raw disk(s) to this file.

```
root@host01:/opt/ora/software# vi mytest.lun
/dev/rdisk/c3t1d0s0
:wq!
```

7. Verify that the raw disk is accessible.
  8. Run an ORION `advanced` test.
- ```
root@host01:/opt/ora/software# ./orion -run advanced -testname \
mytest -num_disks 4 -verbose -duration 10
```
9. List the ORION test reports.
  10. Peruse the ORION test reports to determine your system's disk I/O performance metrics. In the `mbps.csv` report, what was the maximum data throughput (MB/s) achieved in the test?

\_\_\_\_\_

In the `*iops.csv` report, what was the maximum input/output operations per second (IOPS) achieved in the test?

\_\_\_\_\_

In the `*lat.csv` report, what was the latency for the maximum amount of outstanding small reads performed in the test? \_\_\_\_\_

## Practice 10-2: Monitoring Disk I/O Performance

### Overview

In this practice, you create load for a system disk. You monitor system performance as you gradually increase the load to the disk I/O saturation point.

### Tasks

Perform the following steps to monitor disk I/O performance:

**Note:** If you are not currently logged in to your server, log in now.

1. Open a terminal window and run the `format` command to determine what disks are configured in the system.
2. Run `zpool status` to determine which of the configured disks is currently in use.
3. Create a ZFS pool named `test` by using one of the available disks.
4. Create a ZFS file system named `load` in the `test` zpool.
5. Run the `vmstat` command at 5-second intervals 1000 times.
6. Open a second window, run the following `dd` command:

```
root@host01:~# dd if=/dev/random of=/test/load/largefile1 \
bs=1024 count=4000000&
```

7. Monitor the `vmstat` output in the first window.  
Was there any change to the `r` or `b` field to indicate any excessive queuing?  
\_\_\_\_\_  
Does the system response time feel degraded? \_\_\_\_\_
8. In the second window, run the `dd` command. Run additional `dd` commands while incrementing `largefile#` (`largefile2`, `largefile3`, and so on) with each addition `dd` command until system response degradation becomes noticeable.  
Was there any change to the `r` or `b` field to indicate any excessive queuing?  
\_\_\_\_\_
9. Terminate the `vmstat` command by entering `CTL-C`.
10. In the second window, run the `fsstat` command to get a report on kernel I/O activity for the `test/load` file system.
11. In the second window, run the following `dtrace` one-liner to show disk I/O size distribution plots.

```
root@host01:~# dtrace -n 'io:::start { @size[execname] =
quantize(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C
...
```

12. Kill the `dd` processes.

## Practice 10-3: Using the DTrace Toolkit and Other Tools to Find the Nature of a Given Test Load

---

### Overview

In this practice, you use the following tools to try to find the nature of the two programs explored in this practice:

- `iostat`
- `truss`
- `vmstat`
- `mpstat`
- DTrace one-liners
- DTrace using the syscall provider
- DTrace Toolkit:
  - `iosnoop`
  - `iopattern`
  - `seeksize`

### Tasks

1. In the `/opt/ora/software` directory, run the `orion normal` test.

```
root@host01:/opt/ora/software# ./orion -run normal -testname \
mytest -num_disks 4 -verbose
```

2. In a second window, run the following commands to determine the disk loading characteristics:

- `iostat`
- `truss`
- `vmstat`
- `mpstat`
- DTrace one-liners
- DTrace Toolkit:
  - `iosnoop`
  - `iopattern`
  - `seeksize`

What tools did you find helpful in determining the type of load this program was putting on the disk?

3. Kill the `orion` process.

## Practice 10-4: Limiting the Size of the ARC in ZFS

### Overview

In this practice, you limit the size of the ARC in ZFS.

### Tasks

1. Run `mdb -k`.
2. Use `::arc` to see information about arc statistics in `mdb`.  
Note the value of `c_max`.  
\_\_\_\_\_
3. Use `::memstat` to determine the amount of memory that is currently used for ZFS pages.  
Note the amount of memory used by ZFS File Data \_\_\_\_\_.
4. Use `zfs_arc_max/E` to see the tunable parameter that is used to set the maximum target size of the ZFS adaptive replacement cache.  
If the tunable parameter `zfs_arc_max` is set to 0, the system sets the target maximum size of the cache to 3/4 of memory, or memory -1 GB, whichever is larger. To check the size of physical memory.  
Quit `mdb`.
5. View the target maximum size for the arc by using the `kstat` command to display the `c_max` property.  
Note the target maximum size.  
\_\_\_\_\_
6. In a separate window, use `kstat` to monitor the size of the arc every five seconds.
7. To increase the size of the arc, try entering some `cat` commands, such as:

```
root@host01:~# cat /opt/ora/course_files/test_file > t1
root@host01:~# cat t1 > t2
root@host01:~# cat t1 t2 > t3
root@host01:~# cat t1 t2 t3 > t4
...
```

You should see in the `kstat` window the size of the cache grow until it reaches the target maximum and stay near that figure.

8. In `mdb`, use `::memstat` to check the amount of memory used for ZFS File Data.  
ZFS File Data \_\_\_\_\_
9. To see what happens when other requests for memory exist outside of ZFS, you will run the `mem_hog` program in `lab9` to lock down about half of physical memory. To observe the amount of free memory that remains on your system when the ZFS arc has to reclaim some of the pages from the arc, run the following `dtrace` command.

```
root@host01:~# dtrace -n 'fbt:zfs:arc_reclaim_needed:return
/arg1/{ printf("freemem = %d\n", `freemem); exit(0);}'
```

**Note:** The routine `arc_reclaim_needed` returns a nonzero value (in `arg1`) if the size of the arc has to be reduced.

10. In another window, run the `mem_hog` program from `/opt/ora/course_files/lab9` to lock all but 1 GB of free memory. For example, if the system has 6 GB of free memory, this would lock down 5000 MB.

```
root@host01:~# cd /opt/ora/course_files/lab9
root@host01:/opt/ora/course_file/lab9# ./mem_hog 5000
Please wait...

Hit "q" and Enter to exit...
```

**Note:** If this does not trigger the `dtrace` command, quit the `mem_hog` program and increase the memory to be locked down.

What was the value of `freemem` when the reclaim started?

\_\_\_\_\_ (freemem is in units of pages)

11. Set the maximum target size for the `arc` variable (`zfs:zfs_arc_max`) to 2/3 of memory (for example, if your system has 16 GB, add this line to the `/etc/system` file):

```
root@host01:~# vi /etc/system
...
set zfs:zfs_arc_max=10737418240
...
:wq!
```

This would set the `arc` max to 10 GB.

12. Reboot the system to have this take effect.  
**Note:** This logs you off your lab system. After a couple minutes, open another Gnome session on your system and continue the practice.
13. When the system is backed up, check the value of the `zfs_arc_max` property in `mdb`.
14. Check the target maximum size for the `arc` by using the `kstat` command to display the `c_max` property.
15. Verify that the limit has taken effect by repeating steps 6 through 8 with the new limit.

## Solution 10-1: Exploring ORION

### Overview

In this practice, you explore the capabilities of the Oracle I/O Numbers Calibration Tool (ORION).

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

Perform the following steps to explore ORION:

1. Change to the `/opt/ora/software` directory.
2. Unzip the `orion_solaris_sparc64.gz` package.

```
root@host01:/opt/ora/software# gunzip orion_solaris_sparc64.gz
```

3. Rename the `orion_solaris_sparc64` file to `orion` and make it executable.

```
root@host01:/opt/ora/software# mv orion_solaris_sparc64 orion
root@host01:/opt/ora/software# chmod 755 orion
```

4. Run the `format` command to determine what disks are configured in the system.

```
root@host01:~# format
AVAILABLE DISK SELECTIONS:
    0. c3t0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
        /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0
    1. c3t1d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
Specify disk (enter its number): ^D
```

5. Run `zpool status` to determine which of the configured disks is currently in use.

```
root@host01:~# zpool status
pool: rpool
state: ONLINE
scan: none requested
config:

    NAME                STATE        READ WRITE CKSUM
    rpool                ONLINE             0     0     0
        c3t0d0s0        ONLINE             0     0     0

errors: No known data errors
```

6. Using a text editor, create a file named `mytest.lun`. Add one unused raw disk(s) to this file.

```
root@host01:/opt/ora/software# vi mytest.lun
/dev/rdisk/c3t1d0s0
:wq!
```

**Note:** ORION uses this file to determine which physical disks to use during testing.



7. Verify that the raw disk is accessible.

```
root@host01:/opt/ora/software# dd if=/dev/rdisk/c3t1d0s0 \
of=/dev/null bs=32k count=1024
1024+0 records in
1024+0 records out
```

8. Run an ORION advanced test.

```
root@host01:/opt/ora/software# ./orion -run advanced -testname \
mytest -num_disks 4 -verbose -duration 10
```

**Note:** The `-num_disks 4` field is used by ORION to determine I/O loading characteristics. It does not mean the number of physical disks.

**Note:** There are three levels of testing in ORION: simple, normal, and advanced.

The simple test measures the performance of small random reads at different loads and then large random reads at different loads. It takes approximately 30 minutes to run.

The normal test is the same as simple but also generates combinations of small random I/O and large random I/O workloads for a range of loads. This test takes approximately three hours to run.

The advanced test is a user defined test. If no specific test parameters are entered, this test defaults to simple.

```
ORION: ORacle IO Numbers -- Version 11.1.0.7.0
```

```
mytest_20000609_2151
```

```
Test will take approximately 5 minutes
```

```
Larger caches may take longer
```

```
Name: /dev/rdisk/c3t1d0s2
```

```
Size: 73394847744
```

```
1 FILE found.
```

```
TEST START
```

```
-----
```

```
Running point: Small=0, Large=0
```

```
Point 1 out of 29
```

```
Valid
```

```
Running point: Small=1, Large=0
```

```
Point 2 out of 29
```

```
Valid
```

```
...
```

9. List the ORION test reports.

```
root@host01:/opt/ora/software# ls mytest_20000609_2151*
mytest_20000609_2151_iops.csv      mytest_20000609_2151_summary.txt
mytest_20000609_2151_lat.csv      mytest_20000609_2151_trace.txt
mytest_20000609_2151_mbps.csv
```

10. Peruse the ORION test reports to determine your system's disk I/O performance metrics. In the `mbps.csv` report, what was the maximum data throughput (MB/s) achieved in the test?

\_\_41.28 MB/s\_\_

In the `*iops.csv` report, what was the maximum input/output operations per second (IOPS) achieved in the test?

\_\_268 MB/s\_\_

In the `*lat.csv` report, what was the latency for the maximum amount of outstanding small reads performed in the test? \_\_74.48 ms\_\_

**Note:** The ORION reports contain:

- `*summary.txt`: The command input parameters, the maximum I/O throughput, the maximum I/O rate, and minimum latency.
- `*mbps.csv`: The data transfer rates (MB/s) results for the large random/sequential workload. For example, the following data indicates that with 20 outstanding large reads and no small I/Os, you get a data throughput of 41.28 MB/s.

```
...
1, 35.89
2, 37.16
...
20, 41.28
```

- `*iops.csv`: The I/O throughput (in IOPS) results for the small random workload. For example, with 20 outstanding small reads and no large reads, you get a throughput of 268 IOPS.

```
Large/Small, 1, 2, 3, 4, 5, 6, 7, ... 20
0, 130, 197, 235, 273, 295, 311, 322, ... 268
```

- `*lat.csv`: The latency results for small random workload. For example, with 20 outstanding small reads and no large reads, you get a turn-round latency of 74.48 milliseconds.

```
Large/Small, 1, 2, 3, 4, 5, 6, ... 20
0, 7.68, 10.12, 12.74, 14.62, 16.90, 19.28, ... 74.48
```

- `*trace.txt`: This file contains expended test output.

## Solution 10-2: Monitoring Disk I/O Performance

### Overview

In this practice, you create load for a system disk. You monitor system performance as you gradually increase the load to the disk I/O saturation point.

### Tasks

Perform the following steps to monitor disk I/O performance:

**Note:** If you are not currently logged in to your server, log in now.

1. Open a terminal window and run the `format` command to determine what disks are configured in the system.

```
root@host01:~# format
AVAILABLE DISK SELECTIONS:
    0. c3t0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
        /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0
    1. c3t1d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
Specify disk (enter its number): ^D
```

2. Run `zpool status` to determine which of the configured disks is currently in use.

```
root@host01:~# zpool status
pool: rpool
state: ONLINE
scan: none requested
config:

    NAME                STATE                READ WRITE CKSUM
    rpool                ONLINE                0     0     0
        c3t0d0s0        ONLINE                0     0     0

errors: No known data errors
```

3. Create a ZFS pool named `test` by using one of the available disks.

```
root@host01:~# zpool create -f test c3t1d0
root@solaris-11-demo:~# zpool status
pool: rpool
state: ONLINE
scan: none requested
config:

    NAME                STATE                READ WRITE CKSUM
    rpool                ONLINE                0     0     0
        c3t0d0s0        ONLINE                0     0     0

errors: No known data errors
```

```

pool: test
state: ONLINE
scan: none requested
config:

      NAME          STATE          READ WRITE CKSUM
      test           ONLINE           0     0     0
      c3t3d0         ONLINE           0     0     0

errors: No known data errors
pool: rpool
state: ONLINE
scan: none requested
config:

      NAME          STATE          READ WRITE CKSUM
      rpool          ONLINE           0     0     0
      c3t1d0s0       ONLINE           0     0     0

errors: No known data errors

```

4. Create a ZFS file system named `load` in the `test` zpool.

```

root@host01:~# zfs create test/load
root@host01:~# zfs list|grep load
test/load                               31K  66.9G    31K  /test/load

```

5. Run the `vmstat` command at 5-second intervals 1000 times.

```

root@host01:~# vmstat 5 1000
 kthr    memory          page          disk          faults          cpu
 r  b  w  swap    free    re  mf  pi  po  fr  de  sr  s2  s3  s4  s5  in   sy   cs  us  sy  id
0  0  0  6701904  5896272  539 848  9   0   0  0  33  58   0   0   0 1408 42674 25440 4  4  92
0  0  0  6588816  5686616  396 589  0   0   0  0  22   0   0   0 1255 43120 27542 2  3  95
0  0  0  6588520  5684296  538 608  0   0   0  0   3   0   0   0 1425 41315 25490 2  3  95
0  0  0  6588696  5686376  411 573  0   0   0  0   2   0   0   0 1382 41053 25442 2  3  95
...

```

6. Open a second window, run the following `dd` command.

```

root@host01:~# dd if=/dev/random of=/test/load/largefile1 \
bs=1024 count=4000000&

```

7. Monitor the `vmstat` output in the first window.

Was there any change to the `r` or `b` field to indicate any excessive queuing? \_\_No\_\_

Does the system response time feel degraded? \_\_No\_\_

8. In the second window, run the `dd` command. Run additional `dd` commands while incrementing the `largefile#` (`largefile2`, `largefile3`, and so on) with each addition `dd` command until system response degradation becomes noticeable.

**Note:** Use the up-arrow key at access the bash command history.

```

root@host01:~# dd if=/dev/random of=/test/load/largefile2 \
bs=1024 count=4000000&
root@host01:~# dd if=/dev/random of=/test/load/largefile3 \
bs=1024 count=4000000&
root@host01:~# dd if=/dev/random of=/test/load/largefile4 \
bs=1024 count=4000000&
root@host01:~# dd if=/dev/random of=/test/load/largefilea \
bs=1024 count=4000000&
...

```

Was there any change to the *r* or *b* field to indicate any excessive queuing? At approximately 17-20 dd instances you should start seeing queuing in the *r* field.

9. Terminal the `vmstat` command by entering `CTL-C`.
10. In the second window, run the `fsstat` command to get a report on kernel I/O activity for the `test/load` file system.

```

root@host01:~# fsstat -i /test/load
read read  write write rddir rddir rwlock rwulock
ops bytes  ops bytes  ops bytes  ops      ops
990 170K 50.6K 50.3M   12 2.45K 51.6K 51.6K /test/load

```

11. In the second window, run the following `dtrace` one-liner to show disk I/O size distribution plots.

```

root@host01:~# dtrace -n 'io:::start { @size[execname] =
quantize(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C
...
zpool-test

```

| value  | ----- Distribution ----- | count |
|--------|--------------------------|-------|
| 256    |                          | 0     |
| 512    | @@@@@@@@@@@@@@@@@@       | 58    |
| 1024   | @@@@@@                   | 26    |
| 2048   | @@@@@@@@@                | 37    |
| 4096   | @@@@                     | 16    |
| 8192   | @                        | 5     |
| 16384  | @@                       | 9     |
| 32768  | @                        | 3     |
| 65536  |                          | 0     |
| 131072 | @@                       | 7     |
| 262144 |                          | 2     |
| 524288 |                          | 0     |

12. Kill the `dd` processes.

```

root@host01:~# pkill dd
...

```

## Solution 10-3: Using the DTrace Toolkit and Other Tools to Find the Nature of a Given Test Load

### Overview

In this practice, you use the following tools to try to find the nature of the two programs explored in this practice:

- iostat
- truss
- vmstat
- mpstat
- DTrace one-liners
- DTrace using the syscall provider
- DTrace Toolkit:
  - iosnoop
  - iopattern
  - seeksize

### Tasks

1. In the /opt/ora/software directory, run the orion normal test.

```
root@host01:/opt/ora/software# ./orion -run normal -testname \
  mytest -num_disks 4 -verbose
ORION: ORacle IO Numbers -- Version 11.1.0.7.0
mytest_20000609_2211
...
```

2. In a second window, run the following commands to determine the disk-loading characteristics:

- iostat
- truss
- vmstat
- mpstat
- DTrace one-liners
- DTrace Toolkit:
  - iosnoop
  - iopattern
  - seeksize

What tools did you find helpful in determining the type of load this program was putting on the disk?

    All these programs are useful for assessing I/O performance.    

3. Kill the orion process.

```
root@host01:~# pkill orion
...
```

## Solution 10-4: Limiting the Size of the ARC in ZFS

### Overview

In this practice, you limit the size of the ARC in ZFS.

### Tasks

1. Run `mdb -k`.

```
mdb -k
```

```
Loading modules: [ unix genunix specfs dtrace zfs scsi_vhci sd
mpt mac px ldc ip hook neti arp usba kssl fctl sockfs random
mdesc idm cpc crypto fcip fcp ufs logindmux nsmb ptm spps nfs
lofs ipc ]
>
```

2. Use `::arc` to see information about arc statistics in `mdb`.

```
> ::arc
hits = 2639844
misses = 124985
demand_data_hits = 851615
demand_data_misses = 58745
demand_metadata_hits = 1748911
demand_metadata_misses = 40384
prefetch_data_hits = 447
prefetch_data_misses = 4841
prefetch_metadata_hits = 38871
prefetch_metadata_misses = 21015
mru_hits = 302879
mru_ghost_hits = 0
mfu_hits = 2304731
mfu_ghost_hits = 0
deleted = 85799
mutex_miss = 1133
hash_elements = 18446744073709493203
hash_elements_max = 18446744073709551615
hash_collisions = 8325
hash_chains = 18446744073709546824
hash_chain_max = 4
p = 1230 MB
c = 1230 MB
c_min = 64 MB
c_max = 14922 MB
...
```

Note the value of `c_max`.

14922 MB

3. To determine the amount of memory that is currently used for ZFS pages, enter:

```
> ::memstat
Page Summary                Pages                MB    %Tot
-----
Kernel                      182042                1422    9%
ZFS File Data               135087                1055    6%
Anon                       942547                7363   45%
Exec and libs               4363                  34     0%
Page cache                  9921                   77     0%
Free (cachelist)           49713                  388     2%
Free (freelist)            757095                5914   36%
Total                      2080768               16256
>
```

Note the amount of memory used by ZFS File Data. 1422 MB (9%)

4. Use `zfs_arc_max/E` to see the tunable parameter that is used to set the maximum target size of the ZFS adaptive replacement cache, then quit `mdb`.

```
> zfs_arc_max/E
zfs_arc_max:
zfs_arc_max: 0
>
```

If the tunable parameter `zfs_arc_max` is set to 0, the system will set the target maximum size of the cache to 3/4 of memory, or all but 1 GB of the total memory, whichever is larger. To check the size of physical memory, enter:

```
root@host01:~# prtconf | grep Mem
Memory size: 16384 Megabytes
```

Quit `mdb`.

```
> $q
```

5. View the target maximum size for the `arc` by using the `kstat` command to display the `c_max` property.

```
root@host01:~# kstat -p zfs::arcstats:c_max
zfs:0:arcstats:c_max 15647588352
```

Note the target maximum size.

15647588352

6. In a separate window, use `kstat` to monitor the size of the `arc` every five seconds.

```
root@host01:~# kstat -p zfs::arcstats:size 5
zfs:0:arcstats:size 1211232512

zfs:0:arcstats:size 1211230016

zfs:0:arcstats:size 1211233512
...
```



7. To increase the size of the arc, try entering some `cat` commands, such as:

```
root@host01:~# cat /opt/ora/course_files/test_file > t1
root@host01:~# cat t1 > t2
root@host01:~# cat t1 t2 > t3
root@host01:~# cat t1 t2 t3 > t4
...
```

You should see in the `kstat` window the size of the cache grow until it reaches the target maximum and stay near that figure.

8. In `mdb`, use `::memstat` to check the amount of memory used for ZFS File Data.

```
root@host01:~# mdb -k
> ::memstat
```

| Page Summary     | Pages   | MB    | %Tot |
|------------------|---------|-------|------|
| Kernel           | 181950  | 1421  | 9%   |
| ZFS File Data    | 135160  | 1055  | 6%   |
| Anon             | 941568  | 7356  | 45%  |
| Exec and libs    | 2201    | 17    | 0%   |
| Page cache       | 10149   | 79    | 0%   |
| Free (cachelist) | 52011   | 406   | 2%   |
| Free (freelist)  | 757729  | 5919  | 36%  |
| Total            | 2080768 | 16256 |      |

ZFS File Data \_\_\_\_\_

9. To see what happens when other requests for memory exist outside of ZFS, you will run the `mem_hog` program in `lab9` to lock down about half of physical memory. To observe the amount of free memory that remains on your system when the ZFS arc has to reclaim some of the pages from the arc, run the following `dtrace` command.

```
root@host01:~# dtrace -n 'fbt:zfs:arc_reclaim_needed:return
/arg1/{ printf("freemem = %d\n", `freemem); exit(0);}'
dtrace: description 'fbt:zfs:arc_reclaim_needed:return' matched
1 probe
```

**Note:** The routine `arc_reclaim_needed` returns a nonzero value (in `arg1`) if the size of the arc has to be reduced.

10. In another window, run the `mem_hog` program from `/opt/ora/course_files/lab9` to lock all but 1 GB of free memory. For example, if the system has 6 GB of free memory, this would lock down 5000 MB.

```
root@host01:~# vmstat 5
```

| kthr |   | memory |          | page    |    |    |    | disk |    |    |     | faults |    | cpu |    |     |     |     |    |    |     |
|------|---|--------|----------|---------|----|----|----|------|----|----|-----|--------|----|-----|----|-----|-----|-----|----|----|-----|
| r    | b | w      | swap     | free    | re | mf | pi | po   | fr | de | sr  | s2     | s3 | s4  | -- | in  | sy  | cs  | us | sy | id  |
| 0    | 0 | 281    | 30434040 | 6814272 | 9  | 29 | 5  | 0    | 1  | 0  | 241 | 10     | 3  | 0   | 0  | 933 | 330 | 810 | 0  | 2  | 98  |
| 0    | 0 | 252    | 30426336 | 6478728 | 2  | 4  | 0  | 0    | 0  | 0  | 0   | 0      | 0  | 0   | 0  | 926 | 364 | 579 | 0  | 0  | 100 |

```
root@host01:~# cd /opt/ora/course_files/lab9
root@host01:/opt/ora/course_file/lab9# ./mem_hog 5000
```

```
Please wait...
```

```
Hit "q" and Enter to exit...
```

**Note:** If this does not trigger the `dtrace` command, quit the `mem_hog` program and increase the memory to be locked down.

```
...
dtrace: description 'fbt:zfs:arc_reclaim_needed:return ' matched
1 probe
CPU      ID                      FUNCTION:NAME
  30    33681          arc_reclaim_needed:return freemem = 241368
```

What was the value of `freemem` when the reclaim started?

24168 (`freemem` is in units of pages)

11. Set the maximum target size for the `arc` variable (`zfs:zfs_arc_max`) to 2/3 of memory (for example, if your system has 16 GB, add this line to the `/etc/system` file):

```
root@host01:~# vi /etc/system
...
set zfs:zfs_arc_max=10737418240
...
:wq!
```

This would set the `arc` max to 10 GB.

12. Reboot the system to have this take effect.

```
root@host01:~# reboot
```

**Note:** This logs you off your lab system. After a couple minutes, open another Gnome session on your system and continue the practice.

13. When the system is backed up, check the value of the `zfs_arc_max` property in `mdb`.

```
root@host01:~# mdb -k
Loading modules: [ unix genunix specfs dtrace zfs scsi_vhci sd
mpt mac px ldc ip hook neti arp usba kssl fctl sockfs random
mdesc idm cpc crypto fcip fcp ufs logindmux nsmb ptm spps nfs
lofs ipc ]
> zfs_arc_max/E
zfs_arc_max:
zfs_arc_max: 10737418240
> $q
```

14. Check the target maximum size for the `arc` by using the `kstat` command to display the `c_max` property.

```
root@host01:~# kstat -p zfs::arcstats:c_max
zfs:0:arcstats:c_max 10737418240
```

15. Verify that the limit has taken effect by repeating steps 6 through 8 with the new limit.
- a. *In a separate window, use `kstat` to monitor the size of the `arc` every five seconds.*

```
root@host01:~# kstat -p zfs::arcstats:size 5
zfs:0:arcstats:size 896881368

zfs:0:arcstats:size 896881368

zfs:0:arcstats:size 896881368

...
```

- b. *To increase the size of the `arc`, try entering some `cat` commands, such as:*

```
root@host01:~# cat /opt/ora/course_files/test_file > t1
root@host01:~# cat t1 > t2
root@host01:~# cat t1 t2 > t3
root@host01:~# cat t1 t2 t3 > t4

...
```

*You should see in the window in which you are monitoring the size of the cache with the `kstat -p zfs::arcstats:size 5` command the size of the cache grow until it reaches the target maximum and stay near that figure.*

- c. *In `mdb`, use `::memstat` to check the amount of memory used for ZFS File Data.*

```
root@host01:~# mdb -k
> ::memstat
```

| Page Summary     | Pages   | MB    | %Tot |
|------------------|---------|-------|------|
| -----            | -----   | ----- | ---- |
| Kernel           | 181074  | 1414  | 9%   |
| ZFS File Data    | 69954   | 546   | 3%   |
| Anon             | 70252   | 548   | 3%   |
| Exec and libs    | 2813    | 21    | 0%   |
| Page cache       | 15525   | 121   | 1%   |
| Free (cachelist) | 10175   | 79    | 0%   |
| Free (freelist)  | 1730975 | 13523 | 83%  |
| Total            | 2080768 | 16256 |      |



# **Practices for Lesson 11: Solaris 11 Network Tuning**

## **Chapter 11**

## Practices for Lesson 11: Overview

---

### Practices Overview

In this practice, you optimize network performance.

Solutions for each task in this practice are provided at the back of this guide.

## Practice 11-1: Configuring the Maximum Transmission Unit (MTU)

---

### Overview

Jumbo frame support is designed to enhance Ethernet networking throughput and significantly reduce the CPU utilization of large file transfers, like large multimedia files or large data files, by enabling more efficient, larger payloads per packet. By sending larger payloads per packet, fewer packets need to be routed, thereby reducing the overhead on the CPU and potentially improving networking throughput. The maximum data frame size for an Ethernet interface is determined by the MTU property.

In this practice, you adjust the default MTU for a network interface to match the Oracle Database session data unit (SDU) default size.

Perform these steps to reconfigure the MTU for network interface `net3`:

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

1. Halt the web, storage, and database zones.
2. Determine the network physical interfaces by running the `dladm show-phys` command.
3. Determine the MTU of interface `net3` by running the `dladm show-link` command.
4. Edit the `/kernel/drv/e1000g.conf` file to support frame sizes up to 4 KB on the fourth instance of `e1000g` (`net3`).
5. Use `dladm` to set the `mtu` property for interface `net3` to 2048 and display the results.

## Practice 11-2: Configuring a Link Aggregation

---

### Overview

Link aggregation allows you to enhance the network availability and performance by combining multiple network interfaces together to form an aggregation of those interfaces, which acts as a single network interface with greatly enhanced availability and performance. When you aggregate multiple network interfaces, you create a new network interface on top of the aggregated physical interfaces combined in the link layer.

Link aggregation requires at least two network interfaces. The network interfaces must be unplumbed before they can be aggregated. In this practice, you aggregate four network interfaces on your system as the persistent network interface.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Configure a Link Aggregation

Perform these steps to configure a link aggregation:

1. List the network links currently configured in the system.
2. Create a link aggregation named `speedway0` consisting of network interfaces `net1`, `net2`, and `net3`, and show the results.
3. Use the `dladm show-aggr` command to view the link aggregation properties.
4. Use the `ipadm create-ip` command to create an IP interface for data link `speedway0` and show the results.
5. Use the `ipadm create-addr` command to create the static IPv4 address for the interface `speedway0` on the same subnet as interface `net0`. Show the results.
6. Test the new link aggregation interface.

### Task 2: Remove a Link Aggregation

Perform these steps to remove a link aggregation:

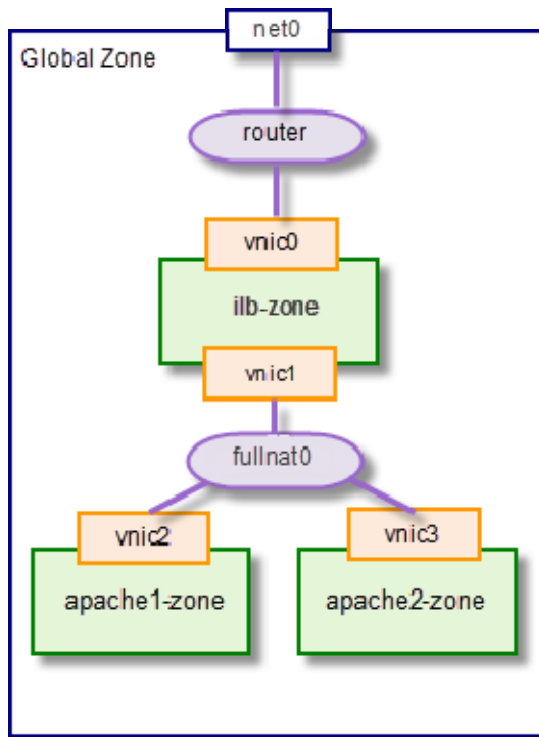
1. Run the `ipadm delete-ip` command to remove the static IPv4 address on interface `speedway0`, and show the results.
2. Run the `ipadm delete-ip` command to remove the IP interface for `speedway0`.
3. Run the `dladm delete-aggr` command to remove the `speedway0` data link.



## Practice 11-3: Configuring an Integrated Load Balancer

### Overview

In this practice, you load balance incoming traffic to the virtual server across two zones, each running the Apache server. The load balancer itself is be configured as a multihomed zone (`ilb-zone`), as shown in the following figure. One interface of `ilb-zone` (`vnic0`) is connected to the outside network. The other interface (`vnic1`) is connected to a full-NAT private virtual network. In the ILB zone (`ilb-zone`), the advertised (data) IP address is referred to as the virtual IP address (VIP) of a load-balancing rule.



**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task: Configure the Integrated Load Balancer (ILB)

Perform these steps to configure the ILB:

1. Create the virtual network interfaces (VNICS) and etherstub and show the results.

```

root@host01:~# dladm create-vnic -l net0 vnic0
root@host01:~# dladm create-etherstub fullnat0
root@host01:~# dladm create-vnic -l fullnat0 vnic1
root@host01:~# dladm create-vnic -l fullnat0 vnic2
root@host01:~# dladm create-vnic -l fullnat0 vnic3
root@host01:~# dladm show-link
LINK                CLASS      MTU      STATE    OVER
net1                 phys       1500     down     --
net2                 phys       1500     down     --

```

|          |           |      |         |          |
|----------|-----------|------|---------|----------|
| net0     | phys      | 1500 | up      | --       |
| net3     | phys      | 1500 | down    | --       |
| vnic0    | vnic      | 1500 | up      | net0     |
| fullnat0 | etherstub | 9000 | unknown | --       |
| vnic1    | vnic      | 9000 | up      | fullnat0 |
| vnic2    | vnic      | 9000 | up      | fullnat0 |
| vnic3    | vnic      | 9000 | up      | fullnat0 |

2. Create the ILB zone (ilb-zone).

```

root@host01:~# zonecfg -z ilb-zone
ilb-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:ilb-zone>create
zonecfg:ilb-zone>set zonepath=/zones/ilb-zone
zonecfg:ilb-zone>add net
zonecfg:ilb-zone:net>set physical=vnic0
zonecfg:ilb-zone:net>end
zonecfg:ilb-zone>add net
zonecfg:ilb-zone:net>set physical=vnic1
zonecfg:ilb-zone:net>end
zonecfg:ilb-zone>remove anet
zonecfg:ilb-zone>verify
zonecfg:ilb-zone>commit
zonecfg:ilb-zone>exit

```

3. Install the ilb-zone zone.

```

root@host01:~# zoneadm -z ilb-zone install
...

```

4. Boot the ilb-zone zone.

```

root@host01:~# zoneadm -z ilb-zone boot
root@host01:~# zoneadm list -v ilb-zone
ID NAME      STATUS  PATH      BRAND    IP
0  global     running /          solaris  shared
4  ilb-zone   running /zones/ilb-zone solaris  excl

```

5. Use `zlogin -C` to log in to the ilb-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks you through various system configuration tasks.

**Note:** You might have to wait a couple of minutes before the SCI tool is available.

Use the following configuration for the ilb-zone zone:

- Computer name: ilb-zone
- vnic0 IP Address: *<unused IP address>*

**Note:** To find an available (unused) IP address, run the `ping` command from the global zone. Test various addresses until you find one that does not return an answer. For example:

```
root@host04:~# ping 192.168.106.233
no answer from 192.168.106.233
```

- DNS: Do not configure DNS
- Alternate Name Service: None
- Time zone: Your local time zone
- Root password: oracle1
- Your real name: oracle
- User login: oracle
- User password: oracle1

6. After the ilb-zone is configured, log in as oracle and su to root.
7. Configure the vnic1 network interface.

```
root@ilb-zone:~# ipadm create-ip vnic1
root@ilb-zone:~# ipadm create-addr -T static -a \
local=192.168.1.100/24 vnic1/v4
```

8. Install the Apache 2.2 service.

```
root@ilb-zone:~# pkg install apache-22
      Packages to install:  7
      Create boot environment: No
      Create backup boot environment: No
      Services to change:  1

DOWNLOAD                                PKGS          FILES
XFER (MB)   SPEED
Completed                                7/7           665/665
8.7/8.7   1.6M/s

PHASE                                ITEMS
Installing new actions                916/916
Updating package state database         Done
Updating image state                   Done
Creating fast lookup database          Done
root@ilb-zone:~#
```

9. Return (~.) to the global zone.
10. Configure the apache1-zone zone.

```
root@host01:~# zonecfg -z apache1-zone
apache1-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:apache1-zone>create
zonecfg:apache1-zone>set zonepath=/zones/apache1-zone
zonecfg:apache1-zone>add net
zonecfg:apache1-zone:net>set physical=vnic2
zonecfg:apache1-zone:net>end
```

```

zonecfg:apache1-zone>remove anet
zonecfg:apache1-zone>verify
zonecfg:apache1-zone>commit
zonecfg:apache1-zone>exit

```

#### 11. Configure the apache2-zone zone.

```

root@host01:~# zonecfg -z apache2-zone
apache2-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:apache2-zone>create
zonecfg:apache2-zone>set zonepath=/zones/apache2-zone
zonecfg:apache2-zone>add net
zonecfg:apache2-zone:net>set physical=vnic3
zonecfg:apache2-zone:net>end
zonecfg:apache2-zone>remove anet
zonecfg:apache2-zone>verify
zonecfg:apache2-zone>commit
zonecfg:apache2-zone>exit

```

#### 12. Clone the apache1-zone zone from the ilb-zone zone.

```

root@host01:~# zoneadm -z ilb-zone shutdown
root@host01:~# zoneadm -z apache1-zone clone ilb-zone
The following ZFS file system(s) have been created:
    rpool/zones/apache1-zone
Progress being logged to
/var/log/zones/zoneadm.20000304T184726Z.apache1-zone.clone
Log saved in non-global zone as /zones/apache1-
zone/root/var/log/zones/zoneadm.20000304T184726Z.apache1-
zone.clone

```

#### 13. Boot the apache1-zone zone.

```

root@host01:~# zoneadm -z apache1-zone boot

```

#### 14. Use `zlogin -C` to log in to the apache1-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks through various system configuration tasks.

**Note:** You might have to wait a couple minutes before the SCI tool is available.

Use the following configuration for the apache1-zone zone:

- Computer name: apache1-zone
- vnic2 IP Address: 192.168.1.101
- DNS: Do not configure DNS
- Alternate Name Service: None Time zone: your local time zone
- Root password: oracle1
- Your real name: oracle
- User login: oracle
- User password: oracle1

15. After the apache1-zone is configured, log in as oracle and su to root.  
 16. Enable the Apache 2.2 service.

```
root@apache1-zone:~# svcadm enable svc:/network/http:apache22
root@apache1-zone:~# svcs svc:/network/http:apache22
STATE STIME FMRI
online 14:17:20 svc:/network/http:apache22
```

17. Move back (~.) to the global zone.  
 18. Boot the ilb-zone zone.

```
root@host01:~# zoneadm -z ilb-zone boot
```

19. Log in to the ilb-zone zone and enable IPv4 forwarding.

```
root@host01:~# zlogin ilb-zone
Oracle Corporation      SunOS 5.11 11.1 September 2012
root@ilb-zone:~# routeadm -u -e ipv4-forwarding
root@ilb-zone:~# routeadm
```

|                  | Configuration<br>Option | Current<br>Configuration           | Current<br>System State |
|------------------|-------------------------|------------------------------------|-------------------------|
| -----            |                         |                                    |                         |
|                  | IPv4 routing            | enabled                            | enabled                 |
|                  | IPv6 routing            | disabled                           | disabled                |
|                  | IPv4 forwarding         | enabled                            | enabled                 |
|                  | IPv6 forwarding         | disabled                           | disabled                |
|                  | Routing services        | "route:default ripng:default"      |                         |
| Routing daemons: |                         |                                    |                         |
|                  | STATE                   | FMRI                               |                         |
|                  | disabled                | svc:/network/routing/ripng:default |                         |
|                  | disabled                | svc:/network/routing/legacy-       |                         |
| routing:ipv4     |                         |                                    |                         |
|                  | disabled                | svc:/network/routing/legacy-       |                         |
| routing:ipv6     |                         |                                    |                         |
|                  | disabled                | svc:/network/routing/rdisc:default |                         |
|                  | online                  | svc:/network/routing/route:default |                         |
|                  | online                  | svc:/network/routing/ndp:default   |                         |

**Note:** You can test communication to the outside world from within the ILB zone by pinging an external IP address.

20. Return (~.) to the global zone.  
 21. Add a route to reach apache1-zone.

```
root@host01:~# route -p add 192.168.1.0 192.168.106.233
add net 192.168.1.0: gateway 192.168.106.233
add persistent net 192.168.1.0: gateway 192.168.106.233
```

22. Clone the apache2-zone zone from the apache1-zone zone.

```
root@host01:~# zoneadm -z apache1-zone shutdown
root@host01:~# zoneadm -z apache2-zone clone apache1-zone
The following ZFS file system(s) have been created:
    rpool/zones/apache2-zone
Progress being logged to
/var/log/zones/zoneadm.20000304T213745Z.apache2-zone.clone
Log saved in non-global zone as /zones/apache2-
zone/root/var/log/zones/zoneadm.20000304T213745Z.apache2-
zone.clone
```

23. Boot the apache1-zone and apache2-zone zones.

```
root@host01:~# zoneadm -z apache1-zone boot
root@host01:~# zoneadm -z apache2-zone boot
```

24. Use `zlogin -C` to log in to the apache2-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks through various system configuration tasks.

**Note:** You might have to wait a couple minutes before the SCI tool is available.

Use the following configuration for the apache2-zone zone:

- **Computer name:** apache2-zone
- **Vnic3 IP Address:** 192.168.1.102
- **DNS:** None
- **Time zone:** your local time zone
- **Root password:** oracle1
- **Your real name:** oracle
- **User login:** oracle
- **User password:** oracle1

25. Log in to the apache2-zone zone as `oracle` and `su` to `root`. Verify that the Apache server is enabled.

```
root@apache2-zone:~# svcs svc:/network/http:apache22
STATE STIME FMRI
online 14:17:20 svc:/network/http:apache22
```

26. Return (`~.`) to the global zone.

27. Use `zlogin` to log in to the ilb-zone zone.

28. Install the `ilb` package.

```
root@ilb-zone:~# pkg install ilb

Packages to install:  1
Create boot environment: No
Create backup boot environment: No
Services to change:  1

DOWNLOAD                                PKGS      FILES
XFER (MB)    SPEED
```

|                                 |       |       |
|---------------------------------|-------|-------|
| Completed                       | 1/1   | 23/23 |
| 0.2/0.2 966k/s                  |       |       |
| PHASE                           | ITEMS |       |
| Installing new actions          | 56/56 |       |
| Updating package state database | Done  |       |
| Updating image state            | Done  |       |
| Creating fast lookup database   | Done  |       |

29. Enable the ilb service.

```
root@ilb-zone:~# svcadm enable ilb
```

30. Create an ILB server group consisting of apache1-zone and apache2-zone. Zones.

```
root@ilb-zone:~# ilbadm create-servergroup -s \
servers=192.168.1.101:8080,192.168.1.102:8080 apachegroup
root@ilb-zone:~# ilbadm show-servergroup
```

| SGNAME      | SERVERID      | MINPORT | MAXPORT | IP_ADDRESS    |
|-------------|---------------|---------|---------|---------------|
| apachegroup | apachegroup.0 | 8080    | 8080    | 192.168.1.101 |
| apachegroup | apachegroup.1 | 8080    | 8080    | 192.168.1.102 |

31. Create a load-balancing rule named apacherule\_rr based on the following criteria:

- The rule must be persistent.
- The VIP for this rule: *<unused IP address>*

**Note:** To find an available (unused) IP address, run the ping command from the global zone. Test various addresses until you find one that does not return an answer. For example:

```
root@host01:~# ping 192.168.106.234
no answer from 192.168.106.234
```

- The load-balancing algorithm: round-robin
- Protocol: TCP
- Network topology (type): NAT
- Proxy source address: 192.168.1.100
- Port: 80
- Server group: apachegroup

```
root@ilb-zone:~# ilbadm create-rule -e -p -i
vip=192.168.106.234,port=80 -m lbalg=rr,type=NAT,proxy-
src=192.168.1.100 -o servergroup=apachegroup apacherule_rr
root@ilb-zone:~# ilbadm show-rule
```

| RULENAME      | STATUS | LBALG      | TYPE | PROTOCOL | VIP             | PORT |
|---------------|--------|------------|------|----------|-----------------|------|
| apacherule_rr | E      | roundrobin | NAT  | TCP      | 192.168.106.234 | 80   |

```
root@ilb-zone:~# ilbadm show-rule -f
RULENAME: apacherule_rr
STATUS: E
PORT: 80
PROTOCOL: TCP
LBALG: roundrobin
```

```

        TYPE: NAT
    PROXY-SRC: 192.168.1.100-192.168.1.100
        PMASK: /32
        HC-NAME: --
        HC-PORT: --
    CONN-DRAIN: 0
    NAT-TIMEOUT: 120
    PERSIST-TIMEOUT: 60
    SERVERGROUP: apachegroup
        VIP: 192.168.106.234
    SERVERS: _apachegroup.0, _apachegroup.1

```

### 32. Advertise the VIP to the outside world.

```

root@ilb-zone:~# dladm show-vnic vnic0
LINK      OVER    SPEED  MACADDRESS      MACADDRTYPE  VID
vnic0     ?        1000   2:8:20:fd:4a:7d  random       0
root@ilb-zone:~# arp -s 192.168.106.234 2:8:20:fd:4a:7d pub \
permanent
root@ilb-zone:~# arp -a|grep 192.168.106.234
vnic0 192.168.106.234 255.255.255.255 SPA 02:08:20:fd:4a:7d
root@ilb-zone:~#

```

## Task 2: Remove the Integrated Load Balancer

In this task, you remove the ILB components and delete the nonglobal zones.

Perform these steps to remove the ILB:

1. Log in to the ilb-zone zone.

```
root@host01:~# zlogin ilb-zone
```

2. Delete the load-balancing rule named apacherule\_rr.

```
root@ilb-zone:~# ilbadm delete-rule apachedrule_rr
```

3. Remove the load-balancing rule named apachegroup.

```
root@ilb-zone:~# ilbadm delete-servergroupapachegroup
```

4. Return (~.) to the global zone.

5. Halt the ilb-zone, apache1.zone, and apache2.zone zones.

```

root@host01:~# zoneadm -z ilb-zone halt
root@host01:~# zoneadm -z apache1-zone halt
root@host01:~# zoneadm -z apache2-zone halt

```

6. Uninstall the ilb-zone, apache1.zone, and apache2.zone zones.

```

root@host01:~# zoneadm -z ilb-zone uninstall
...
root@host01:~# zoneadm -z apache1-zone uninstall
...
root@host01:~# zoneadm -z apache2-zone uninstall
...

```



7. Remove the VIP from the arp table.

```
root@host01:~# arp -d 192.168.106.115  
192.168.106.115 (192.168.106.115) deleted
```

8. If the web, storage, and database zones are currently halted, boot them now.

```
root@host01:~# zoneadm -z web boot  
root@host01:~# zoneadm -z storage boot  
root@host01:~# zoneadm -z database boot
```

## Practice 11-4: Controlling Data Flows

---

### Overview

In this task, you use data flow controls to create a policy for inbound HTTP traffic. You do this by restricting HTTP data flow on one of the virtual NICs. For example: `vnic4`.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Control Data Flow on a Network Interface

In this task, you add flow control to a virtual network interface.

Perform these steps to control virtual network data flow:

1. Use `dladm show-vnic` to verify that `vnic4` does not currently exist.
2. Create interface `vnic4` and use the `flowadm` command to control HTTP data on `vnic4`.
3. Use the `flowadm show-flow` command to display the flow controls currently configured in the system.
4. Throttle HTTP traffic across the `vnic4` VNIC to 100 Mb/s.
5. Set the priority on `vnic4` to low.
6. Display the flow controls properties.

Now, network interface `vnic4` can be used to enforce the HTTP policy.

## Practice 11-5: Monitoring Network Performance

### Overview

In this practice, you monitor the performance of the network by completing the following tasks:

- Determining the status of a client system
- Sending a data packet to the client system
- Tracing the route of the data packet to the destination system
- Testing the reliability of data packets
- Capturing data packets and saving the information in a file

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Using `spray` to Determine Network Performance

1. Use the `ping` command to determine the status of the client computer. Ask your instructor or a classmate for their server IP address. What is the status of the client?
2. Use the `ping -s` command to send a data packet to the client computer. How much time did the round trip from the client computer take?
3. Trace the route of the data packet to the destination computer.  
Which command did you use?  
How many times did the data hop to reach the destination computer?
4. Use the `spray` command to test the reliability of data packets. To do this, the machine that will be sprayed must enable the service. On the *target* machine, enter:

```
root@host01:~# spray -c 20000 -d 10 -l 8845 <target>
```

For example:

```
# spray -c 20000 -d 10 -l 8845 192.168.106.77
sending 20000 packets of length 8845 to 192.168.106.77 ...
      3 packets (0.015%) dropped by 192.168.106.77
    1364 packets/sec, 12068237 bytes/sec
```

How many data packets were dropped during data transmission?

### Task 2: Monitor the Network by Using `dlstat`

`dlstat` lets you to generate reports containing runtime statistics about data links.

Perform these steps to monitor the network by using the `dlstat` command:

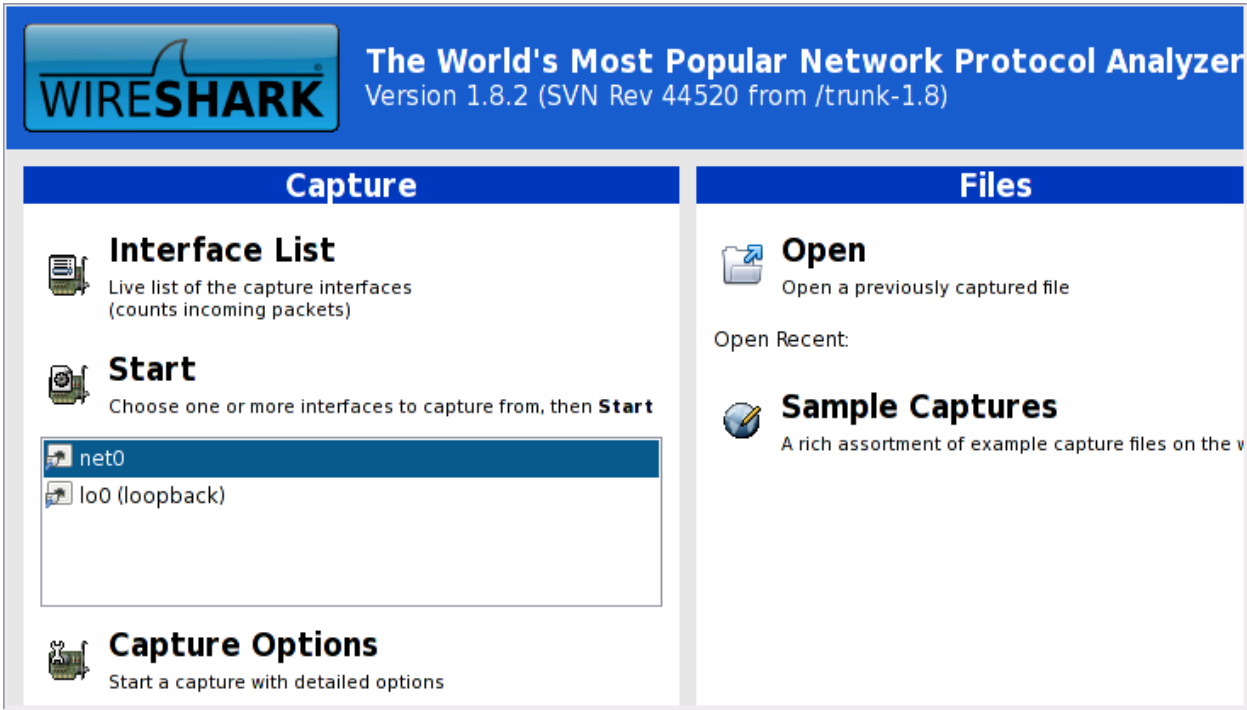
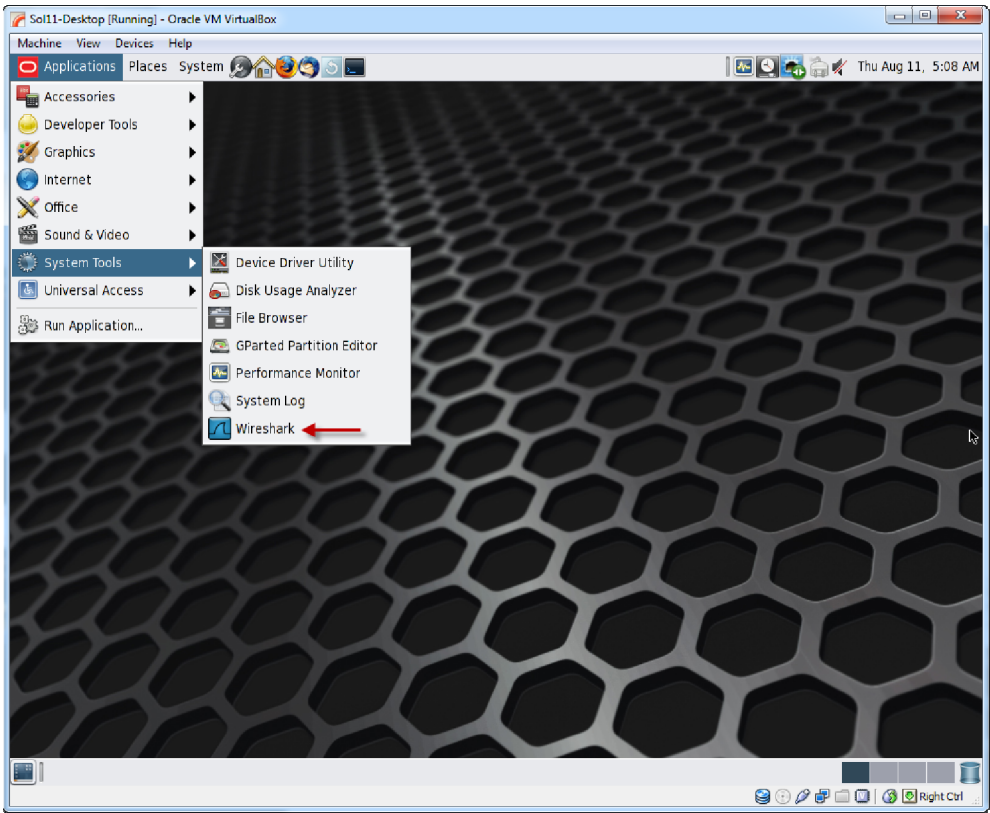
1. Run the `dladm` command to display statistics for all the network links.
2. Run the `dladm show-phys` command to display statistics for all physical network devices.
3. Run the `dladm show-link` command to display statistics for all network links.

### Task 3: Monitor the Network by Using Wireshark

Oracle Solaris 11 adds a variety of robust network utilities. For network observability, the new Wireshark utility has been added. Wireshark is a powerful network protocol analyzer that lets you to capture and interactively browse the traffic running on a computer network.

Perform these steps to monitor the network by using Wireshark:

1. Install the Wireshark package.
2. To start Wireshark, open the Applications menu and select System Tools. Click the Wireshark icon.



3. Click the List Available Capture Interfaces icon to begin your capture:



4. Click the Options button for interface `net0` and set the Capture Filter value to `host Your_IP_Address` and the Capture File to `/var/tmp/Your_IP_Address.cap`. Then click the Start button.
5. To generate network traffic between this system and the Image Package Server, run the `pkg search entire` command.
6. After the package search has completed, click the Stop The Running Live Capture button to stop your capture.



7. Click the Close This Capture File button to close and save your capture.



8. From the Files menu in the Wireshark main screen, select Open and browse to the `/var/tmp` directory. Select the `192.168.106.112.cap` file and click Open.
9. Take a few minutes and read through the packet trace.
10. Click the Statistics tab and select Summary.
11. Click the Statistics tab and select Protocol Hierarchy.
12. Click the Statistics tab and select Endpoints.
13. Click the Statistics tab and select IO Graphs.
14. Click the Close This Capture File button to close and save your capture.
15. In the Wireshark main screen, click File and then click Quit to close Wireshark.

## Solution 11-1: Configuring the Maximum Transmission Unit (MTU)

### Overview

Jumbo frame support is designed to enhance Ethernet networking throughput and significantly reduce the CPU utilization of large file transfers, like large multimedia files or large data files, by enabling more efficient, larger payloads per packet. By sending larger payloads per packet, fewer packets need to be routed, thereby reducing the overhead on the CPU and potentially improving networking throughput. The maximum data frame size for an Ethernet interface is determined by the MTU (maximum transmission unit) property.

In this practice, you adjust the default MTU for a network interface to match the Oracle Database session data unit (SDU) default size.

Perform these steps to reconfigure the MTU for network interface `net3`:

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

**Note:** If you are not currently logged in to your server, log in now.

### Tasks

1. Halt the web, storage, and database zones.

```
root@host01:~# zoneadm -z web halt
root@host01:~# zoneadm -z storage halt
root@host01:~# zoneadm -z database halt
```

2. Determine the network physical interfaces by running the `dladm show-phys` command:

```
root@host01:~# dladm show-phys
```

| LINK | MEDIA    | STATE   | SPEED | DUPLEX  | DEVICE  |
|------|----------|---------|-------|---------|---------|
| net1 | Ethernet | unknown | 0     | unknown | e1000g1 |
| net2 | Ethernet | unknown | 0     | unknown | e1000g2 |
| net0 | Ethernet | up      | 1000  | full    | e1000g0 |
| net3 | Ethernet | unknown | 0     | unknown | e1000g3 |

3. Determine the MTU of interface `net3` by running the `dladm show-link` command:

```
root@host01:~# dladm show-link net3
```

| LINK | CLASS | MTU  | STATE   | OVER |
|------|-------|------|---------|------|
| net3 | phys  | 1500 | unknown | --   |

4. Edit the `/kernel/drv/e1000g.conf` file to support frame sizes up to 4 KB on the fourth instance of `e1000g` (`net3`).

```
root@host01:~# vi /kernel/drv/e1000g.conf
...
MaxFrameSize=0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0;
...
```

5. Use `dladm` to set the `mtu` property for interface `net3` to 2048 and display the results.

```
root@host01:~# dladm set-linkprop -p mtu=2048 net3
root@host01:~# dladm show-link net3
```

| LINK | CLASS | MTU  | STATE   | OVER |
|------|-------|------|---------|------|
| net3 | phys  | 2048 | unknown | --   |

## Solution 11-2: Configuring a Link Aggregation

### Overview

Link aggregation allows you to enhance the network availability and performance by combining multiple network interfaces together to form an aggregation of those interfaces, which acts as a single network interface with greatly enhanced availability and performance. When you aggregate multiple network interfaces, you create a new network interface on top of the aggregated physical interfaces combined in the link layer.

Link aggregation requires at least two network interfaces. The network interfaces must be unplumbed before they can be aggregated. In this practice, you aggregate four network interfaces on your system as the persistent network interface.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task: Configure a Link Aggregation

Perform these steps to configure a link aggregation:

1. List the network links currently configured in the system.

```
root@host01:~# dladm show-link
```

| LINK | CLASS | MTU  | STATE   | OVER |
|------|-------|------|---------|------|
| net1 | phys  | 1500 | unknown | --   |
| net2 | phys  | 1500 | unknown | --   |
| net0 | phys  | 1500 | up      | --   |
| net3 | phys  | 2048 | unknown | --   |

2. Create a link aggregation named `speedway0` consisting of network interfaces `net1`, `net2`, and `net3`, and show the results.

```
root@host01:~# dladm create-aggr -l net1 -l net2 -l net3 \
speedway0
root@host01:~# dladm show-link
```

| LINK      | CLASS | MTU  | STATE   | OVER           |
|-----------|-------|------|---------|----------------|
| net1      | phys  | 1500 | down    | --             |
| net2      | phys  | 1500 | down    | --             |
| net0      | phys  | 1500 | up      | --             |
| net3      | phys  | 2048 | down    | --             |
| speedway0 | aggr  | 1500 | unknown | net1 net2 net3 |

```
root@host01:~# dladm show-aggr
```

| LINK      | POLICY | ADDRPOLICY | LACPACTIVITY | LACPTIMER | FLAGS |
|-----------|--------|------------|--------------|-----------|-------|
| speedway0 | L4     | auto       | off          | short     | ----- |

3. Use the `dladm show-aggr` command to view the link aggregation properties.

```
root@host01:~# dladm show-aggr
```

| LINK      | MODE  | POLICY | ADDRPOLICY | LACPACTIVITY | LACPTIMER |
|-----------|-------|--------|------------|--------------|-----------|
| speedway0 | trunk | L4     | auto       | off          | short     |

- Use the `ipadm create-ip` command to create an IP interface for data link `speedway0` and show the results.

```
root@host01:~# ipadm create-ip speedway0
root@host01:~# ipadm show-if speedway0
IFNAME      CLASS      STATE      ACTIVE OVER
speedway0   ip         down       no      --
```

- Use the `ipadm create-addr` command to create the static IPv4 address for the interface `speedway0` on the same subnet as interface `net0`. Show the results.

```
root@host01:~# ipadm show-addr net0
ADDROBJ      TYPE      STATE      ADDR
net0/v4      static    ok         192.168.106.112/24
net0/v6      addrconf ok         fe80::214:4fff:fe82:8caa/10
root@host01:~# ipadm create-addr -T static \
-a 192.168.106.113/24 speedway0/v4
root@host01:~# ipadm show-addr
ADDROBJ      TYPE      STATE      ADDR
lo0/v4       static    ok         127.0.0.1/8
net0/v4      static    ok         192.168.106.112/24
speedway0/v4 static    ok         192.168.106.113/24
lo0/v6       static    ok         ::1/128
net0/v6      addrconf ok         fe80::214:4fff:fe82:8caa/10
```

- Test the new link aggregation interface.

```
root@host01:~# ping 192.168.106.113
192.168.106.113 is alive
```

## Task 2: Remove a Link Aggregation

Perform these steps to remove a link aggregation:

- Run the `ipadm delete-ip` command to remove the static IPv4 address on interface `speedway0`, and show the results.

```
root@host01:~# ipadm delete-addr speedway0/v4
root@host01:~# ipadm show-addr
ADDROBJ      TYPE      STATE      ADDR
lo0/v4       static    ok         127.0.0.1/8
net0/v4      static    ok         192.168.106.112/24
lo0/v6       static    ok         ::1/128
net0/v6      addrconf ok         fe80::214:4fff:fe82:8caa/10
```

- Run the `ipadm delete-ip` command to remove the IP interface for `speedway0`.

```
root@host01:~# ipadm delete-ip speedway0
root@host01:~# ipadm show-if
IFNAME      CLASS      STATE      ACTIVE OVER
lo0         loopback  ok         yes    --
net0        ip        ok         yes    --
```



3. Run the `dladm delete-aggr` command to remove the `speedway0` data link.

```

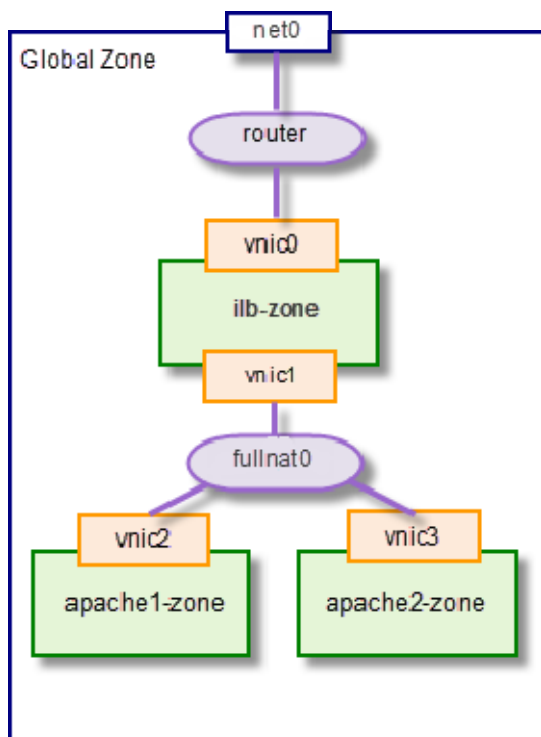
root@host01:~# dladm show-link
LINK          CLASS      MTU      STATE    OVER
net1          phys       1500    down     --
net2          phys       1500    down     --
net0          phys       1500    up       --
net3          phys       2048    down     --
speedway0     aggr       1500    unknown  net1 net2 net3
root@host01:~# dladm delete-aggr speedway0
root@host01:~# dladm show-link
LINK          CLASS      MTU      STATE    OVER
net1          phys       1500    unknown  --
net2          phys       1500    unknown  --
net0          phys       1500    up       --
net3          phys       2048    unknown  --

```

## Solution 11-3: Configuring an Integrated Load Balancer

### Overview

In this practice, you load balance incoming traffic to the virtual server across two zones, each running the Apache server. The load balancer itself is configured as a multihomed zone (`ilb-zone`), as shown in the following figure. One interface of `ilb-zone` (`vnic0`) is connected to the outside network. The other interface (`vnic1`) is connected to a full-NAT private virtual network. In the ILB zone (`ilb-zone`), the advertised (data) IP address is referred to as the virtual IP address (VIP) of a load-balancing rule.



**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Configure the Integrated Load Balancer

Perform these steps to configure the Integrated Load Balancer (ILB):

1. Create the virtual network interfaces (VNICS) and etherstub and show the results.

```
root@host01:~# dladm create-vnic -l net0 vnic0
root@host01:~# dladm create-etherstub fullnat0
root@host01:~# dladm create-vnic -l fullnat0 vnic1
root@host01:~# dladm create-vnic -l fullnat0 vnic2
root@host01:~# dladm create-vnic -l fullnat0 vnic3
root@host01:~# dladm show-link
```

| LINK | CLASS | MTU  | STATE | OVER |
|------|-------|------|-------|------|
| net1 | phys  | 1500 | down  | --   |
| net2 | phys  | 1500 | down  | --   |

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

|          |           |      |         |          |
|----------|-----------|------|---------|----------|
| net0     | phys      | 1500 | up      | --       |
| net3     | phys      | 1500 | down    | --       |
| vnic0    | vnic      | 1500 | up      | net0     |
| fullnat0 | etherstub | 9000 | unknown | --       |
| vnic1    | vnic      | 9000 | up      | fullnat0 |
| vnic2    | vnic      | 9000 | up      | fullnat0 |
| vnic3    | vnic      | 9000 | up      | fullnat0 |

2. Create the ILB zone (ilb-zone).

```

root@host01:~# zonecfg -z ilb-zone
ilb-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:ilb-zone>create
zonecfg:ilb-zone>set zonepath=/zones/ilb-zone
zonecfg:ilb-zone>add net
zonecfg:ilb-zone:net>set physical=vnic0
zonecfg:ilb-zone:net>end
zonecfg:ilb-zone>add net
zonecfg:ilb-zone:net>set physical=vnic1
zonecfg:ilb-zone:net>end
zonecfg:ilb-zone>remove anet
zonecfg:ilb-zone>verify
zonecfg:ilb-zone>commit
zonecfg:ilb-zone>exit

```

3. Install the ilb-zone zone.

```

root@host01:~# zoneadm -z ilb-zone install
...

```

4. Boot the ilb-zone zone.

```

root@host01:~# zoneadm -z ilb-zone boot
root@host01:~# zoneadm list -v ilb-zone
ID NAME      STATUS  PATH      BRAND    IP
0  global     running /          solaris  shared
4  ilb-zone   running /zones/ilb-zone solaris  excl

```

5. Use `zlogin -C` to log in to the ilb-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks through various system configuration tasks.

**Note:** You might have to wait a couple of minutes before the SCI tool is available.

Use the following configuration for the ilb-zone zone:

- Computer name: ilb-zone
- vnic0 IP Address: *<unused IP address>*

**Note:** To find an available (unused) IP address, run the `ping` command from the global zone. Test various addresses until you find one that does not return an answer. For example:

```
root@host04:~# ping 192.168.106.233
no answer from 192.168.106.233
```

- DNS: Do not configure DNS
- Alternate Name Service: None
- Time zone: Your local time zone
- Root password: oracle1
- Your real name: oracle
- User login: oracle
- User password: oracle1

6. After the ilb-zone is configured, log in as oracle and su to root.
7. Configure the vnic1 network interface.

```
root@ilb-zone:~# ipadm create-ip vnic1
root@ilb-zone:~# ipadm create-addr -T static -a \
local=192.168.1.100/24 vnic1/v4
```

8. Install the Apache 2.2 service.

```
root@ilb-zone:~#pkg install apache-22
      Packages to install:  7
      Create boot environment: No
      Create backup boot environment: No
      Services to change:  1

DOWNLOAD                                PKGS          FILES
XFER (MB)   SPEED
Completed                                7/7           665/665
8.7/8.7   1.6M/s

PHASE                                ITEMS
Installing new actions                916/916
Updating package state database         Done
Updating image state                   Done
Creating fast lookup database           Done
root@ilb-zone:~#
```

9. Return (~.) to the global zone.
10. Configure the apache1-zone zone.

```
root@host01:~# zonecfg -z apache1-zone
apache1-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:apache1-zone>create
zonecfg:apache1-zone>set zonepath=/zones/apache1-zone
zonecfg:apache1-zone>add net
zonecfg:apache1-zone:net>set physical=vnic2
zonecfg:apache1-zone:net>end
```

```
zonecfg:apache1-zone>remove anet
zonecfg:apache1-zone>verify
zonecfg:apache1-zone>commit
zonecfg:apache1-zone>exit
```

#### 11. Configure the apache2-zone zone.

```
root@host01:~# zonecfg -z apache2-zone
apache2-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:apache2-zone>create
zonecfg:apache2-zone>set zonepath=/zones/apache2-zone
zonecfg:apache2-zone>add net
zonecfg:apache2-zone:net>set physical=vnic3
zonecfg:apache2-zone:net>end
zonecfg:apache2-zone>remove anet
zonecfg:apache2-zone>verify
zonecfg:apache2-zone>commit
zonecfg:apache2-zone>exit
```

#### 12. Clone the apache1-zone zone from the ilb-zone zone.

```
root@host01:~# zoneadm -z ilb-zone shutdown
root@host01:~# zoneadm -z apache1-zone clone ilb-zone
The following ZFS file system(s) have been created:
  rpool/zones/apache1-zone
Progress being logged to
/var/log/zones/zoneadm.20000304T184726Z.apache1-zone.clone
Log saved in non-global zone as /zones/apache1-
zone/root/var/log/zones/zoneadm.20000304T184726Z.apache1-
zone.clone
```

#### 13. Boot the apache1-zone zone.

```
root@host01:~# zoneadm -z apache1-zone boot
```

#### 14. Use `zlogin -C` to log in to the apache1-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks you through various system configuration tasks.

**Note:** You might have to wait a couple minutes before the SCI tool is available.

Use the following configuration for the apache1-zone zone:

- Computer name: apache1-zone
- vnic2 IP Address: 192.168.1.101
- DNS: Do not configure DNS
- Alternate Name Service: None Time zone: your local time zone
- Root password: oracle1
- Your real name: oracle
- User login: oracle
- User password: oracle1

15. After the apache1-zone is configured, log in as oracle and su to root.  
 16. Enable the Apache 2.2 service.

```
root@apache1-zone:~# svcadm enable svc:/network/http:apache22
root@apache1-zone:~# svcs svc:/network/http:apache22
STATE STIME FMRI
online 14:17:20 svc:/network/http:apache22
```

17. Move back (~.) to the global zone.  
 18. Boot the ilb-zone zone.

```
root@host01:~# zoneadm -z ilb-zone boot
```

19. Log in to the ilb-zone zone and enable IPv4 forwarding.

```
root@host01:~# zlogin ilb-zone
Oracle Corporation      SunOS 5.11 11.1 September 2012
root@ilb-zone:~# routeadm -u -e ipv4-forwarding
root@ilb-zone:~# routeadm
```

|                  | Configuration<br>Option | Current<br>Configuration           | Current<br>System State |
|------------------|-------------------------|------------------------------------|-------------------------|
| -----            |                         |                                    |                         |
|                  | IPv4 routing            | enabled                            | enabled                 |
|                  | IPv6 routing            | disabled                           | disabled                |
|                  | IPv4 forwarding         | enabled                            | enabled                 |
|                  | IPv6 forwarding         | disabled                           | disabled                |
|                  | Routing services        | "route:default ripng:default"      |                         |
| Routing daemons: |                         |                                    |                         |
|                  | STATE                   | FMRI                               |                         |
|                  | disabled                | svc:/network/routing/ripng:default |                         |
|                  | disabled                | svc:/network/routing/legacy-       |                         |
| routing:ipv4     |                         |                                    |                         |
|                  | disabled                | svc:/network/routing/legacy-       |                         |
| routing:ipv6     |                         |                                    |                         |
|                  | disabled                | svc:/network/routing/rdisc:default |                         |
|                  | online                  | svc:/network/routing/route:default |                         |
|                  | online                  | svc:/network/routing/ndp:default   |                         |

**Note:** You can test communication to the outside world from within the ILB zone by pinging an external IP address.

20. Return (~.) to the global zone.  
 21. Add a route to reach apache1-zone.

```
root@host01:~# route -p add 192.168.1.0 192.168.106.233
add net 192.168.1.0: gateway 192.168.106.233
add persistent net 192.168.1.0: gateway 192.168.106.233
```

22. Clone the apache2-zone zone from the apache1-zone zone.

```
root@host01:~# zoneadm -z apache1-zone shutdown
root@host01:~# zoneadm -z apache2-zone clone apache1-zone
The following ZFS file system(s) have been created:
    rpool/zones/apache2-zone
Progress being logged to
/var/log/zones/zoneadm.20000304T213745Z.apache2-zone.clone
Log saved in non-global zone as /zones/apache2-
zone/root/var/log/zones/zoneadm.20000304T213745Z.apache2-
zone.clone
```

23. Boot the apache1-zone and apache2-zone zones.

```
root@host01:~# zoneadm -z apache1-zone boot
root@host01:~# zoneadm -z apache2-zone boot
```

24. Use `zlogin -C` to log in to the apache2-zone zone. You will be prompted by the system configuration tool (SCI). This interactive tool walks through various system configuration tasks.

**Note:** You might have to wait a couple minutes before the SCI tool is available.

Use the following configuration for the apache2-zone zone:

- Computer name: apache2-zone
- Vnic3 IP Address: 192.168.1.102
- DNS: None
- Time zone: your local time zone
- Root password: oracle1
- Your real name: oracle
- User login: oracle
- User password: oracle1

25. Log in to the apache2-zone zone as `oracle` and `su` to `root`. Verify that the Apache server is enabled.

```
root@apache2-zone:~# svcs svc:/network/http:apache22
STATE STIME FMRI
online 14:17:20 svc:/network/http:apache22
```

26. Return (`~.`) to the global zone.

27. Use `zlogin` to log in to the ilb-zone zone.

28. Install the `ilb` package.

```
root@ilb-zone:~# pkg install ilb

Packages to install:  1
Create boot environment: No
Create backup boot environment: No
Services to change:  1

DOWNLOAD                                PKGS          FILES
XFER (MB)    SPEED
```

|                                 |       |       |
|---------------------------------|-------|-------|
| Completed                       | 1/1   | 23/23 |
| 0.2/0.2 966k/s                  |       |       |
| PHASE                           | ITEMS |       |
| Installing new actions          | 56/56 |       |
| Updating package state database | Done  |       |
| Updating image state            | Done  |       |
| Creating fast lookup database   | Done  |       |

29. Enable the ilb service.

```
root@ilb-zone:~# svcadm enable ilb
```

30. Create an ILB server group consisting of apache1-zone and apache2-zone zones.

```
root@ilb-zone:~# ilbadm create-servergroup -s \
servers=192.168.1.101:8080,192.168.1.102:8080 apachegroup
root@ilb-zone:~# ilbadm show-servergroup
```

| SGNAME      | SERVERID      | MINPORT | MAXPORT | IP_ADDRESS    |
|-------------|---------------|---------|---------|---------------|
| apachegroup | apachegroup.0 | 8080    | 8080    | 192.168.1.101 |
| apachegroup | apachegroup.1 | 8080    | 8080    | 192.168.1.102 |

31. Create a load-balancing rule named apacherule\_rr based on the following criteria:

- The rule must be persistent.
- The VIP for this rule: *<unused IP address>*

**Note:** To find an available (unused) IP address, run the ping command from the global zone. Test various addresses until you find one that does not return an answer. For example:

```
root@host04:~# ping 192.168.106.234
no answer from 192.168.106.234
```

- The load-balancing algorithm: round-robin
- Protocol: TCP
- Network topology (type): NAT
- Proxy source address: 192.168.1.100
- Port: 80
- Server group: apachegroup

```
root@ilb-zone:~# ilbadm create-rule -e -p -i
vip=192.168.106.234,port=80 -m lbalg=rr,type=NAT,proxy-
src=192.168.1.100 -o servergroup=apachegroup apacherule_rr
root@ilb-zone:~# ilbadm show-rule
```

| RULENAME      | STATUS | LBALG      | TYPE    | PROTOCOL | VIP             | PORT |
|---------------|--------|------------|---------|----------|-----------------|------|
| apacherule_rr | E      | roundrobin | NAT TCP |          | 192.168.106.234 | 80   |

```
root@ilb-zone:~# ilbadm show-rule -f
```

```

RULENAME: apacherule_rr
STATUS: E
PORT: 80
PROTOCOL: TCP
LBALG: roundrobin
```



```

        TYPE: NAT
    PROXY-SRC: 192.168.1.100-192.168.1.100
        PMASK: /32
        HC-NAME: --
        HC-PORT: --
    CONN-DRAIN: 0
    NAT-TIMEOUT: 120
    PERSIST-TIMEOUT: 60
    SERVERGROUP: apachegroup
        VIP: 192.168.106.234
    SERVERS: _apachegroup.0, _apachegroup.1

```

32. Advertise the VIP to the outside world.

```

root@ilb-zone:~# dladm show-vnic vnic0
LINK      OVER    SPEED  MACADDRESS      MACADDRTYPE  VID
vnic0     ?        1000   2:8:20:fd:4a:7d  random       0
root@ilb-zone:~# arp -s 192.168.106.234 2:8:20:fd:4a:7d pub \
permanent
root@ilb-zone:~# arp -a|grep 192.168.106.234
vnic0 192.168.106.234      255.255.255.255 SPA    02:08:20:fd:4a:7d
root@ilb-zone:~#

```

*Now you have a high-performance Apache server ideal for Cloud environments.*

## Task 2: Remove the Integrated Load Balancer

In this task, you remove the ILB components and delete the nonglobal zones.

Perform these steps to remove the ILB:

1. Log in to the `ilb-zone` zone.

```
root@host01:~# zlogin ilb-zone
```

2. Delete the load-balancing rule named `apachedrule_rr`.

```
root@ilb-zone:~# ilbadm delete-rule apacherule_rr
```

3. Remove the load-balancing rule named `apachegroup`.

```
root@ilb-zone:~# ilbadm delete-servergroup apachegroup
```

4. Return (`~.`) to the global zone.

5. Halt the `ilb-zone`, `apache1.zone`, and `apache2.zone` zones.

```

root@host01:~# zoneadm -z ilb-zone halt
root@host01:~# zoneadm -z apache1-zone halt
root@host01:~# zoneadm -z apache2-zone halt

```

6. Uninstall the `ilb-zone`, `apache1.zone`, and `apache2.zone` zones.

```

root@host01:~# zoneadm -z ilb-zone uninstall
...
root@host01:~# zoneadm -z apache1-zone uninstall
...
root@host01:~# zoneadm -z apache2-zone uninstall
...

```

7. Remove the VIP from the arp table.

```
root@host01:~# arp -d 192.168.106.234  
192.168.106.115 (192.168.106.115) deleted
```

8. If the web, storage, and database zones are currently halted, boot them now.

```
root@host01:~# zoneadm -z web boot  
root@host01:~# zoneadm -z storage boot  
root@host01:~# zoneadm -z database boot
```

## Solution 11-4: Controlling Data Flows

### Overview

In this task, you use data flow controls to create a policy for inbound HTTP traffic. You do this by restricting HTTP data flow on one of the virtual NICs. For example: `vnic4`.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Control Data Flow on a Network Interface

In this task, you add flow control to a virtual network interface.

Perform these steps to control virtual network data flow:

1. Use `dladm show-vnic` to verify that `vnic4` does not currently exist.

```
root@host01:~# dladm show-vnic vnic4
```

2. Create interface `vnic4` and use the `flowadm` command to control HTTP data on `vnic4`.

```
root@host01:~# dladm create-vnic -l net0 vnic4
root@host01:~# dladm show-vnic vnic4
```

| LINK  | OVER | SPEED | MACADDRESS     | MACADDRTYPE | VID |
|-------|------|-------|----------------|-------------|-----|
| vnic4 | net0 | 1000  | 2:8:20:a1:1d:d | random      | 0   |

```
root@host01:~# flowadm add-flow -l vnic4 \
-a transport=tcp,local_port=80 http1
```

3. Use the `flowadm show-flow` command to display the flow controls currently configured in the system.

```
root@host01:~# flowadm show-flow
```

| FLOW  | LINK  | IPADDR | PROTO | LPORT | RPORT | DSFLD |
|-------|-------|--------|-------|-------|-------|-------|
| http1 | vnic4 | --     | tcp   | 80    | --    | --    |

4. Throttle HTTP traffic across the `vnic4` VNIC to 100 Mb/s.

```
root@host01:~# flowadm set-flowprop -p maxbw=100M http1
```

5. Set the priority on `vnic4` to low.

```
root@host01:~# dladm set-linkprop -p priority=low vnic4
```

6. Display the flow controls properties.

```
root@host01:~# flowadm show-flowprop http1
```

| FLOW  | PROPERTY | VALUE | DEFAULT | POSSIBLE |
|-------|----------|-------|---------|----------|
| http1 | maxbw    | 100   | --      | --       |

```
root@host01:~# dladm show-linkprop -p priority vnic4
```

| LINK  | PROPERTY | PERM | VALUE | DEFAULT | POSSIBLE          |
|-------|----------|------|-------|---------|-------------------|
| vnic4 | priority | rw   | low   | high    | low, medium, high |

Now, network interface `vnic4` can be used to enforce the HTTP policy.

## Solution 11-5: Monitoring Network Performance

### Overview

In this practice, you monitor the performance of the network by completing the following tasks:

- Determining the status of a client system
- Sending a data packet to the client system
- Tracing the route of the data packet to the destination system
- Testing the reliability of data packets
- Capturing data packets and saving the information in a file

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Using `spray` to Determine Network Performance

1. Use the `ping` command to determine the status of the client computer. Ask your instructor or a classmate for their server IP address. What is the status of the client?

*If the client responds, the answer will be <client> is alive. For example:*

```
root@host01:~# ping 192.168.106.77
192.168.106.77 is alive
```

2. Use the `ping -s` command to send a data packet to the client computer. How much time did the round trip from the client computer take?

*This value is reported in the time field at the end of the output string. For example:*

```
root@host01:~# ping -s 192.168.106.77
PING 192.168.106.77: 56 data bytes
64 bytes from 192.168.106.77 (192.168.1.10): icmp_seq=0.
time=0.417 ms
64 bytes from 192.168.106.77 (192.168.1.10): icmp_seq=1.
time=0.293 ms
(...)
```

3. Trace the route of the data packet to the destination computer.  
Which command did you use?

*The `traceroute <client>` command. For example:*

```
root@host01:~# traceroute 192.168.106.77
traceroute to 192.168.106.77 (192.168.1.10), 30 hops max, 40 byte
packets
1  192.168.106.77 (192.168.1.10)  0.364 ms  0.204 ms  0.155 ms
```

How many times did the data hop to reach the destination computer?

*If the target system is on the same network, one hop will be reported.*

4. Use the `spray` command to test the reliability of data packets. To do this, the machine that will be sprayed must enable the service. On the *target* machine, enter:

```
root@host01:~# spray -c 20000 -d 10 -l 8845 <target>
```

For example:

```
# spray -c 20000 -d 10 -l 8845 192.168.106.77
sending 20000 packets of length 8845 to 192.168.106.77 ...
```

3 packets (0.015%) dropped by 192.168.106.77  
 1364 packets/sec, 12068237 bytes/sec

How many data packets were dropped during data transmission?

*Some number of drops should occur.*

## Task 2: Monitor the Network by Using `dlstat`

`dlstat` lets you to generate reports containing runtime statistics about data links.

Perform these steps to monitor the network by using the `dlstat` command:

1. Run the `dladm` command to display statistics for all the network links.

```
root@host01:~# dlstat
```

| LINK | IPKTS | RBYTES | OPKTS | OBYTES |
|------|-------|--------|-------|--------|
| net1 | 0     | 0      | 0     | 0      |
| net2 | 0     | 0      | 0     | 0      |
| net3 | 0     | 0      | 0     | 0      |
| ...  |       |        |       |        |

2. Run the `dladm show-phys` command to display statistics for all physical network devices.

```
root@host01:~# dlstat show-phys
```

| LINK | IPKTS | RBYTES | OPKTS | OBYTES  |
|------|-------|--------|-------|---------|
| net1 | 0     | 0      | 0     | 0       |
| net2 | 0     | 0      | 0     | 0       |
| net0 | 4.99M | 2.30G  | 2.45M | 666.46M |
| net3 | 0     | 0      | 0     | 0       |

3. Run the `dladm show-link` command to display statistics for all network links.

```
root@host01:~# dlstat show-link
```

| LINK | IPKTS | RBYTES | OPKTS | OBYTES |
|------|-------|--------|-------|--------|
| net1 | 0     | 0      | 0     | 0      |
| net2 | 0     | 0      | 0     | 0      |
| net3 | 0     | 0      | 0     | 0      |
| ...  |       |        |       |        |

## Task 3: Monitor the Network by Using Wireshark

Oracle Solaris 11 adds a variety of robust network utilities. For network observability, the new Wireshark utility have been added. Wireshark is a powerful network protocol analyzer that lets you to capture and interactively browse the traffic running on a computer network.

Perform these steps to monitor the network by using Wireshark:

1. Install the Wireshark package.

```
root@host01:~# pkg install wireshark
```

```

Packages to install:  3
Create boot environment: No
Create backup boot environment: No
Services to change:  2

Planning linked: 0/6 done; 1 working: zone:database
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

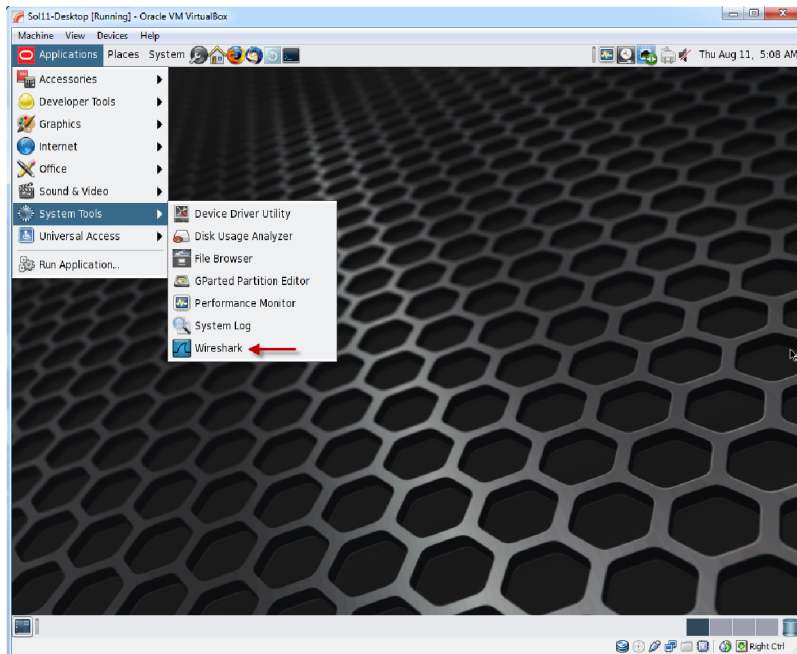
```

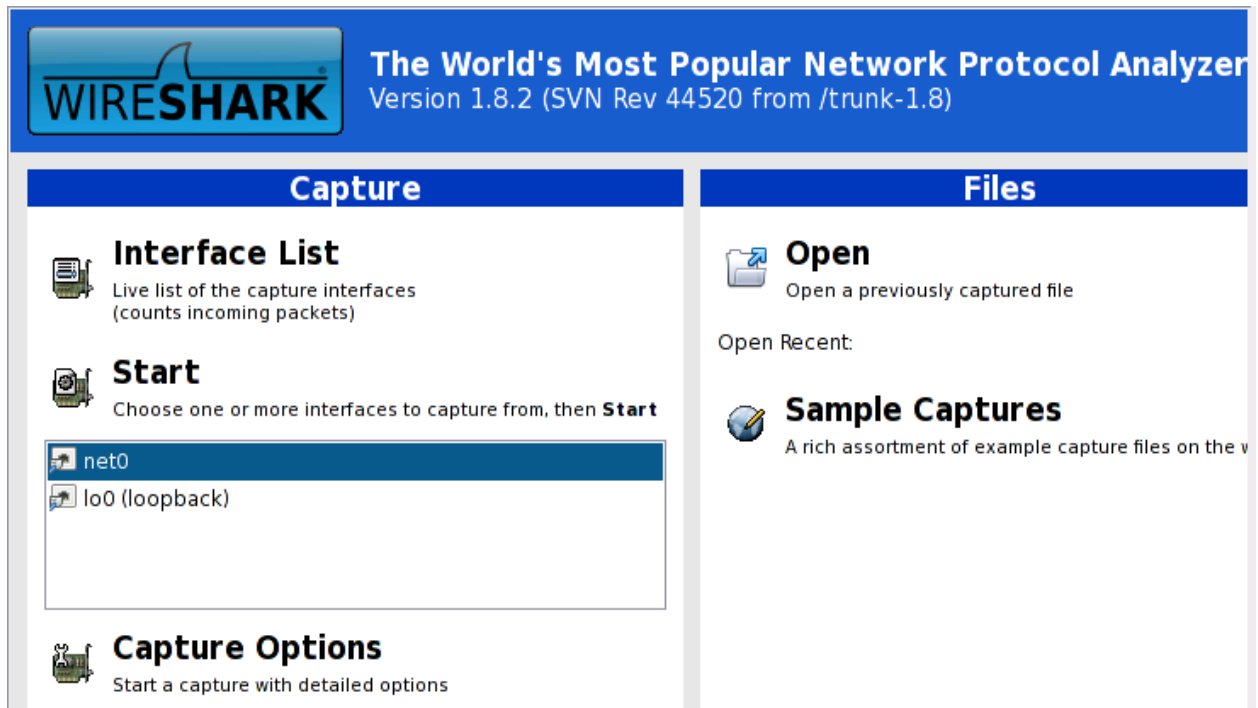
Planning linked: 1/6 done; 1 working: zone:storage
Planning linked: 2/6 done; 1 working: zone:web
Planning linked: 3/6 done
DOWNLOAD                                PKGS          FILES
XFER (MB)    SPEED
Completed                                3/3          294/294
20.0/20.0    0B/s

Downloading linked: 0/6 done; 1 working: zone:database
Downloading linked: 1/6 done; 1 working: zone:storage
Downloading linked: 2/6 done; 1 working: zone:web
Downloading linked: 3/6 done
PHASE                                ITEMS
Installing new actions                389/389
Updating package state database        Done
Updating image state                   Done
Creating fast lookup database          Done
Reading search index                   Done
Updating search index                  3/3
Executing linked: 0/6 done; 1 working: zone:database
Executing linked: 1/6 done; 1 working: zone:storage
Executing linked: 2/6 done; 1 working: zone:web
Executing linked: 3/6 done

```

- To start Wireshark, open the Applications menu and select System Tools. Click the Wireshark icon.





3. Click the List Available Capture Interfaces icon to begin your capture:



4. Click the Options button for interface `net0` and set the Capture Filter value to `hostYour_IP_Address` and the Capture File to `/var/tmp/Your_IP_Address.cap`. Then, click the Start button.
5. To generate network traffic between this system and the Image Package Server, run the `pkg search entire` command.
6. After the package search has completed, click the Stop The Running Live Capture button to stop your capture.



7. Click the Close This Capture File button to close and save your capture.



8. From the Files menu in the Wireshark main screen, select Open and browse to the `/var/tmp` directory. Select the `192.168.106.112.cap` file and click Open.
9. Take a few minutes and read through the packet trace.
10. Click the Statistics tab and select Summary.
11. Click the Statistics tab and select Protocol Hierarchy.
12. Click the Statistics tab and select Endpoints.
13. Click the Statistics tab and select IO Graphs.
14. Click the Close This Capture File button to close and save your capture.
15. In the Wireshark main screen, click File and then click Quit to close Wireshark.





# **Practices for Lesson 12: Resource Management**

## **Chapter 12**

## Practices for Lesson 12: Overview

---

### Practices Overview

Resource management facilities permit you to modify the default behavior of the operating system with respect to different workloads. Behavior primarily refers to the set of decisions that are made by operating system algorithms when an application presents one or more resource requests to the system. You can use resource management facilities to do the following:

- Deny resources or prefer one application over another for a larger set of allocations than otherwise permitted.
- Treat certain allocations collectively instead of through isolated mechanisms.

The key area of resource management explored in this practice is:

- Managing resources

Solutions for each task in this practice are provided at the back of this guide.

## Practice 12-1: Managing Resources

---

### Overview

The Oracle Solaris 11 environment provides an effective solution for adjusting to varying workloads that are generated by applications on running on a system. A workload is the amount of processing that the computer has been given to do at a given time. The workload consists of a number of processes running in the computer and often involves a number of users connected to and interacting with the computer's applications.

If resource management features are not used, the Oracle Solaris operating system responds to workload demands by adapting to new application requests dynamically. The default system is to give requesting processes equal access to system resources. This might not be desirable in cases where you want to give one application priority over another. Resource management features enable you to prioritize workloads based on their relative importance.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Manage Processor Pool Resources

In this task, you create a processor pool and assign it to four projects (workloads). You assign a priority to each workload. You then observe how the system handles each workload when using FSS scheduling.

Perform the following steps to manage workload resources:

1. Log in to your server as user `oracle` and `su` to `root`.
2. Run `dispadm -l` to list the scheduler classes currently configured in the system.
3. List the CPUs currently installed in the system.
4. Determine the status of the processor pools service. If it is currently disabled, enable it now.
5. The pool configuration file is `/etc/pooladm.conf`. By default, this file does not exist. Create the pool configuration file.
6. Create a processor set (`pset`) named `eng-pset1` that contains a minimum of one CPU and a maximum of two CPUs.
7. Create a resource pool named `eng-pool`.
8. Associate the `eng-pset1` processor set with the `eng-pool` pool.
9. Run `pooladm` to list the current processor pools.
10. Use `pooladm -c` to configure the `eng-pset1` processor set and the `eng-pool` processor pools in the system.
11. Use the `poolstat` command to check the `eng-pool` pool status.
12. List the projects currently configured in the system.
13. Create four projects by typing:

```
root@host01:~# projadd -U root eng_team1
root@host01:~# projadd -U root eng_team2
root@host01:~# projadd -U root eng_team3
root@host01:~# projadd -U root eng_team4
```

14. Assign the `eng-pool` processor pool to the new projects.
15. List the projects currently configured in the project database.
16. Make FSS the default class at boot time.

17. **Make this configuration take effect immediately, without rebooting.**
18. Run the `dispadmin -l` and `ps -ec` commands to determine the default scheduler class being used.
19. You prioritize the workloads by setting the number of CPU shares for each project. Assigning a larger share of CPU resources to a project increases its priority over other projects. Set the number of CPU shares for each project you created in the previous task.

```

root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,3,none)" eng_team1
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,5,none)" eng_team2
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,2,none)" eng_team3
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,0,none)" eng_team4
root@host01:~# projects -l

...
eng_team1
    projid : 100
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,3,none)
           project.pool=eng-pool
eng_team2
    projid : 101
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,5,none)
           project.pool=eng-pool
eng_team3
    projid : 102
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,2,none)
           project.pool=eng-pool
eng_team4
    projid : 103
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,0,none)
           project.pool=eng-pool

```

20. Create a workload for the `eng_team1` project by starting a copy routine that runs in an infinite loop.

```
root@host01:~# newtask -p eng_team1 \
dd if=/dev/zero of=/dev/null&
```

21. Use the `prstat -JR` command to monitor the `eng_team1` project (workload) resource consumption.

Let the workload run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

\_\_\_\_\_

On which CPU is this process running?

\_\_\_\_\_

In the lower portion of the `prstat` output, what is the `eng_team1` project's process identifier (PID)?

\_\_\_\_\_

What are the current project IDs on the system?

\_\_\_\_\_

How many cpu-shares did the `eng_team1` project get ( `projects -l`)?

22. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.
23. Create a workload for the `eng_team2` project by starting a copy routine that runs in an infinite loop as in step 20.
24. Use the `prstat -JR` command to monitor the resource consumption of `eng_team1` and `eng_team2` projects (workloads).
- Let the workloads run for a minute or so.
- In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

\_\_\_\_\_

On which CPU is this process running?

\_\_\_\_\_

In the lower portion of the `prstat` output, what is the `eng_team2` project's process identifier (PID)?

\_\_\_\_\_

How many cpu-shares did the `eng_team2` project get ( `projects -l`)?

\_\_\_\_\_

Based on the `prstat` output, which project has the highest priority (more important)?

25. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.
26. Create a workload for the `eng_team3` project by starting a copy routine that runs in an infinite loop as in steps 20 and 23.

27. Use the `prstat -JR` command to monitor the resource consumption of `eng_team1`, `eng_team2`, and `eng_team3` projects (workloads).

Let the workloads run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

\_\_\_\_\_

On which CPU is this process running?

\_\_\_\_\_

In the lower portion of the `prstat` output, what is the `eng_team3` project's process identifier (PID)?

\_\_\_\_\_

How many `cpu-shares` did the `eng_team2` project get ( `project -l`)?

\_\_\_\_\_

Based on the `prstat` output, which "eng\_team" project has the lowest priority (least important)?

\_\_\_\_\_

28. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.
29. Create a workload for the `eng_team4` project by starting a copy routine that runs in an infinite loop as in steps 20, 23, and 26.
30. Use the `prstat -JR` command to monitor the resource consumption of `eng_team4` projects (workloads).
- Let the workloads run for a minute or so.
- In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

\_\_\_\_\_

On which CPU is this process running?

\_\_\_\_\_

In the lower portion of the `prstat` output, what is the `eng_team4` project's process identifier (PID)?

\_\_\_\_\_

How many `cpu-shares` did the `eng_team4` project get ( `project -l`)?

\_\_\_\_\_

Based on the `prstat` output, which projects have the highest priority (most important) and lowest priority (least important)?

\_\_\_\_\_

Note that after a couple minutes, the `eng_team4` project is no longer shown in the `prstat` output. Why?

31. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.
- Note that the `size` field has a value of 1. Why?
32. Use `pkill -J` to kill the `eng_team1`, `eng_team2`, `eng_team3`, and `eng_team4` projects.
33. Make TS the default class at boot time.
34. **Make this configuration take effect immediately, without rebooting.**

## Task 2: Manage Physical Memory Resources

You can control the amount of physical memory that workloads consume by setting memory resource caps. A resource cap is an upper bound placed on the consumption of a resource, such as physical memory. The resource-capping daemon and its associated utilities provide mechanisms for physical memory resource cap enforcement and administration. Like the processor pool resource control, the physical memory resource cap can be defined by using attributes of project entries in the project database. The total amount of physical memory that is available to processes in the project is determined by the `rcap.max-rss` attribute.

In this task, you set physical memory caps for projects you created in Task 1.

Perform the following steps to manage physical memory resources:

1. Determine the amount of memory currently installed in the system.
2. Determine the status of the `rcap` service. If it is currently disabled, enable it now.
3. Set physical memory caps on the "eng\_team" project by typing:

```
root@host01:~# projmod -a -K rcap.max-rss=50KB eng_team1
root@host01:~# projmod -a -K rcap.max-rss=5MB eng_team2
root@host01:~# projmod -a -K rcap.max-rss=5MB eng_team3
root@host01:~# projmod -a -K rcap.max-rss=50KB eng_team4
```

4. List the projects currently configured in the project database.
5. Create a workload for each "eng\_team" project by starting a copy routine that runs in an infinite loop.

```
root@host01:~# newtask -p eng_team1 \
dd if=/dev/zero of=/dev/null&
[1] 5522
root@host01:~# newtask -p eng_team2 \
dd if=/dev/zero of=/dev/null&
[2] 5523
root@host01:~# newtask -p eng_team3 \
dd if=/dev/zero of=/dev/null&
[3] 5524
root@host01:~# newtask -p eng_team4 \
dd if=/dev/zero of=/dev/null&
[4] 5525
```

6. Use the `rcapstat` utility to access the effectiveness of the physical memory cap set for each project.  
Do you see any problems with the current memory caps set for each project? If so, what is the problem and how does this adversely impact the projects?
7. Use the `vmstat` utility to access system memory.
8. Kill the `eng_team1`, `eng_team2`, `eng_team3`, and `eng_team4` projects.

## Solution 12-1: Managing Resources

### Overview

The Oracle Solaris 11 environment provides an effective solution for adjusting to varying workloads that are generated by applications on running on a system. A workload is the amount of processing that the computer has been given to do at a given time. The workload consists of a number of processes running in the computer and often involves a number of users connected to and interacting with the computer's applications.

If resource management features are not used, the Oracle Solaris operating system responds to workload demands by adapting to new application requests dynamically. The default system is to give requesting processes equal access to system resources. This might not be desirable in cases where you want to give one application priority over another. Resource management features enable you to prioritize workloads based on their relative importance.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Manage Processor Pool Resources

In this task, you create a processor pool and assign it to four projects (workloads). You assign a priority to each workload. You then observe how the system handles each workload when using FSS scheduling.

Perform the following steps to manage workload resources:

1. Log in to your server as user `oracle` and `su` to `root`.
2. Run `dispadmin -l` to list the scheduler classes currently configured in the system.

```
root@host01:~# dispadmin -l
CONFIGURED CLASSES
=====

SYS    (System Class)
TS      (Time Sharing)
SDC     (System Duty-Cycle Class)
FX      (Fixed Priority)
```

3. List the CPUs currently installed in the system.

```
root@host01:~# psrinfo
0          on-line   since 05/15/2000 17:23:44
1          on-line   since 05/15/2000 17:23:45
2          on-line   since 05/15/2000 17:23:45
3          on-line   since 05/15/2000 17:23:45
4          on-line   since 05/15/2000 17:23:45
...
```

4. Determine the status of the processor pools service. If it is currently disabled, enable it now.

```
root@host01:~# svcs system/pools
STATE          STIME          FMRI
disabled        8:39:58   svc:/system/pools:default
root@host01:~# svcadm enable system/pools:default
```



```

root@host01:~# svcs system/pools
STATE          STIME      FMRI
online         17:30:43  svc:/system/pools:default

```

5. The pool configuration file is `/etc/pooladm.conf`. By default, this file does not exist. Create the pool configuration file.

```

root@host01:~# ls /etc/pooladm.conf
/etc/pooladm.conf: No such file or directory
root@host01:~# poolcfg -c discover
root@host01:~# ls /etc/pooladm.conf
/etc/pooladm.conf
root@host01:~# more /etc/pooladm.conf
...

```

6. Create a processor set (pset) named `eng-pset1` that contains a minimum of one CPU and a maximum of two CPUs.

```

root@host01:~# poolcfg -c 'create pset eng-pset1 \
(uint pset.min = 1; uint pset.max = 2)'
root@host01:~# poolcfg -c 'info' /etc/pooladm.conf
...
      pset eng-pset1
            int      pset.sys_id -2
            boolean  pset.default false
            uint     pset.min 1
            uint     pset.max 2
            string   pset.units population
            uint     pset.load 0
            uint     pset.size 0
            string   pset.comment

```

7. Create a resource pool named `eng-pool`.

```

root@host01:~# poolcfg -c 'create pool eng-pool'

```

8. Associate the `eng-pset1` processor set with the `eng-pool` pool.

```

root@host01:~# poolcfg -c 'associate pool eng-pool \
(pset eng-pset1)'

```

9. Run `pooladm` to list the current processor pools.

```

root@host01:~# pooladm
system default
      string  system.comment
      int     system.version 1
      boolean system.bind-default true
      string  system.poold.objectives wt-load

      pool pool_default
            int     pool.sys_id 0

```

```

        boolean pool.active true
        boolean pool.default true
        int     pool.importance 1
        string  pool.comment
        pset    pset_default

pset pset_default
    int     pset.sys_id -1
    boolean pset.default true
    uint    pset.min 1
    uint    pset.max 65536
    string  pset.units population
    uint    pset.load 1036
    uint    pset.size 24
    string  pset.comment

    cpu

        int     cpu.sys_id 21
        string  cpu.comment
        string  cpu.status on-line

    cpu

        int     cpu.sys_id 20
        string  cpu.comment
        string  cpu.status on-line

...

```

Note that the `eng-pset1` processor set and the `eng-pool` processor pools are not listed. To persistently configure them in the system, you must run the `pooladm -c` command.

10. Use `pooladm -c` to configure the `eng-pset1` processor set and the `eng-pool` processor pools in the system.

```

root@host01:~# pooladm -c
root@host01:~# pooladm
system default
    string  system.comment
    int     system.version 1
    boolean system.bind-default true
    string  system.poold.objectives wt-load

pool eng-pool
    int     pool.sys_id 5
    boolean pool.active true

```

```

        boolean    pool.default false
        int    pool.importance 1
        string    pool.comment
        pset eng-pset1

pool pool_default
    int    pool.sys_id 0
    boolean    pool.active true
    boolean    pool.default true
    int    pool.importance 1
    string    pool.comment
    pset pset_default

pset eng-pset1
    int    pset.sys_id 1
    boolean    pset.default false
    uint    pset.min 1
    uint    pset.max 2
    string    pset.units population
    uint    pset.load 0
    uint    pset.size 2
    string    pset.comment

    cpu
        int    cpu.sys_id 1
        string    cpu.comment
        string    cpu.status on-line

    cpu
        int    cpu.sys_id 0
        string    cpu.comment
        string    cpu.status on-line

pset pset_default
    int    pset.sys_id -1
    boolean    pset.default true
    uint    pset.min 1
    uint    pset.max 65536
    string    pset.units population
    uint    pset.load 46
    uint    pset.size 22
    string    pset.comment

```

```
cpu
    int    cpu.sys_id 21
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 20
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 23
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 22
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 17
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 16
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 19
    string  cpu.comment
    string  cpu.status on-line

cpu
    int    cpu.sys_id 18
    string  cpu.comment
    string  cpu.status on-line

cpu
```

```
        int    cpu.sys_id 5
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 4
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 7
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 6
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 3
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 2
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 13
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 12
        string  cpu.comment
        string  cpu.status on-line

cpu
        int    cpu.sys_id 15
        string  cpu.comment
```

```

        string      cpu.status on-line

    cpu
        int    cpu.sys_id 14
        string  cpu.comment
        string  cpu.status on-line

    cpu
        int    cpu.sys_id 9
        string  cpu.comment
        string  cpu.status on-line

    cpu
        int    cpu.sys_id 8
        string  cpu.comment
        string  cpu.status on-line

    cpu
        int    cpu.sys_id 11
        string  cpu.comment
        string  cpu.status on-line

    cpu
        int    cpu.sys_id 10
        string  cpu.comment
        string  cpu.status on-line

```

11. Use the `poolstat` command to check the `eng-pool` pool status.

```

root@host01:~# poolstat -p eng-pool

                                pset
id pool                        size used load
 5 eng-pool                      2 0.00 0.00

```

12. List the projects currently configured in the system.

```

root@host01:~# projects -l
system
    projid : 0
    comment: ""
    users  : (none)
    groups : (none)
    attribs:
...

```

13. Create four projects by typing:

```
root@host01:~# projadd -U root eng_team1
root@host01:~# projadd -U root eng_team2
root@host01:~# projadd -U root eng_team3
root@host01:~# projadd -U root eng_team4
```

14. Assign the eng-pool processor pool to the new projects.

```
root@host01:~# projmod -s -K project.pool=eng-pool eng_team1
root@host01:~# projmod -s -K project.pool=eng-pool eng_team2
root@host01:~# projmod -s -K project.pool=eng-pool eng_team3
root@host01:~# projmod -s -K project.pool=eng-pool eng_team4
```

15. List the projects currently configured in the project database.

```
root@host01:~# projects -l
...
eng_team1
    projid : 100
    comment: ""
    users  : root
    groups : (none)
    attribs: project.pool=eng-pool
eng_team2
    projid : 101
    comment: ""
    users  : root
    groups : (none)
    attribs: project.pool=eng-pool
eng_team3
    projid : 102
    comment: ""
    users  : root
    groups : (none)
    attribs: project.pool=eng-pool
eng_team4
    projid : 103
    comment: ""
    users  : root
    groups : (none)
    attribs: project.pool=eng-pool
```

16. Make FSS the default class at boot time.

```
root@host01:~# dispadmin -d FSS
```

17. Make this configuration take effect immediately, without rebooting:

```
root@host01:~# priocntl -s -c FSS -i all
```

18. Run the `dispadmin -l` and `ps -ec` commands to determine the default scheduler class being used.

```
root@host01:~# dispadmin -l
CONFIGURED CLASSES
=====

SYS    (System Class)
TS      (Time Sharing)
SDC     (System Duty-Cycle Class)
FX      (Fixed Priority)
FSS     (Fair Share)
root@host01:~# ps -ec | grep FSS
  1  FSS  58 ?           0:00 init
475  FSS  29 ?           0:00 cron
 11  FSS  29 ?           0:38 svc.star
 13  FSS  29 ?           0:07 svc.conf
5462 FSS  55 pts/1       0:44 bash
 112 FSS  59 ?          73:22 in.mpath
477  FSS  29 ?           0:00 inetd
 123 FSS  29 ?           0:00 pfexecd
  43 FSS  29 ?           0:00 dlmgmt
...
```

19. You prioritize the workloads by setting the number of CPU shares for each project. Assigning a larger share of CPU resources to a project increases its priority over other projects. Set the number of CPU shares for each project you created in the previous task.

```
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,3,none)" eng_team1
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,5,none)" eng_team2
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,2,none)" eng_team3
root@host01:~# projmod -s -K \
"project.cpu-shares=(privileged,0,none)" eng_team4
root@host01:~# projects -l
...
eng_team1
    projid : 100
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,3,none)
            project.pool=eng-pool
eng_team2
```



```

projid : 101
comment: ""
users  : root
groups : (none)
attrs: project.cpu-shares=(privileged,5,none)
       project.pool=eng-pool

eng_team3
projid : 102
comment: ""
users  : root
groups : (none)
attrs: project.cpu-shares=(privileged,2,none)
       project.pool=eng-pool

eng_team4
projid : 103
comment: ""
users  : root
groups : (none)
attrs: project.cpu-shares=(privileged,0,none)
       project.pool=eng-pool

```

20. Create a workload for the `eng_team1` project by starting a copy routine that runs in an infinite loop.

```

root@host01:~# newtask -p eng_team1 \
dd if=/dev/zero of=/dev/null&

```

21. Use the `prstat -JR` command to monitor the `eng_team1` project (workload) resource consumption.

```

root@host01:~# prstat -JR
PID USERNAME  SIZE   RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
5474 root      1720K  956K cpu0    1    0   0:04:41  33% dd/1
3340 pkg5srv   5112K  3760K sleep   1    0   0:00:07  0.1% htcacheclean/1
5475 root      4656K  3444K cpu1   100   -   0:00:00  0.0% prstat/1
   5 root         0K    0K sleep   99  -20   0:00:05  0.0% zpool-
rpool/138
  523 root      2828K  1732K sleep   59    0   0:00:00  0.0% in.routed/1
1454 noaccess  3332K  1940K sleep   59    0   0:00:00  0.0% asr-notify/2
   43 root      4036K  2528K sleep   29    0   0:00:00  0.0% dlmgmt/13
  136 root      6944K  2824K sleep   29    0   0:00:00  0.0% syseventd/18
  248 daemon    6644K  2540K sleep   29    0   0:00:00  0.0% rcapd/1
   99 netadm    4104K  2708K sleep   29    0   0:00:00  0.0% ipmgmt/5
  110 root      2852K  1032K sleep   29    0   0:00:00  0.0% in.mpathd/1
   13 root         17M   16M sleep   29    0   0:00:08  0.0% svc.configd/18
   11 root         15M   12M sleep   29    0   0:00:02  0.0% svc.startd/12
  313 root      3564K  2380K sleep   29    0   0:00:00  0.0% picld/4
   8 root         0K    0K sleep   99  -20   0:00:00  0.0% vmtasks/2

```

| PROJID                                                          | NPROC | SWAP  | RSS   | MEMORY | TIME    | CPU  | PROJECT     |
|-----------------------------------------------------------------|-------|-------|-------|--------|---------|------|-------------|
| 100                                                             | 1     | 3440K | 848K  | 0.0%   | 1:29:33 | 33%  | eng_team1   |
| 0                                                               | 144   | 681M  | 336M  | 8.0%   | 6:32:07 | 32%  | system      |
| 1                                                               | 3     | 13M   | 7472K | 0.2%   | 0:00:44 | 0.0% | user.root   |
| 10                                                              | 2     | 15M   | 6796K | 0.2%   | 0:10:21 | 0.0% | group.staff |
| 3                                                               | 5     | 20M   | 14M   | 0.3%   | 0:00:15 | 0.0% | default     |
| Total: 155 processes, 870 lwps, load averages: 1.01, 0.91, 0.54 |       |       |       |        |         |      |             |
| ^C                                                              |       |       |       |        |         |      |             |

Let the workload run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

\_\_\_\_\_

On which CPU is this process running?

\_\_\_\_\_

In the lower portion of the `prstat` output, what is the `eng_team1` project's process identifier (PID)?

\_\_\_\_\_

What are the current project IDs on the system?

\_\_\_\_\_

How many cpu-shares did the `eng_team1` project get (`projects -l`)?

22. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.

```
root@host01:~# poolstat -r pset -p eng-pool 5 5
id pool          type rid rset          min  max size used load
  1 eng-pool      pset  1 eng-pset1          1   2   1 0.00 0.99
...
```

23. Create a workload for the `eng_team2` project by starting a copy routine that runs in an infinite loop as in step 20.

```
root@host01:~# newtask -p eng_team2 \
dd if=/dev/zero of=/dev/null&
```

24. Use the `prstat -JR` command to monitor the resource consumption of `eng_team1` and `eng_team2` projects (workloads).

```
root@host01:~# prstat -JR
PID USERNAME  SIZE  RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
5512 root      1720K 956K cpu0   1    0 0:01:04  21% dd/1
5474 root      1720K 956K run    1    0 0:26:54  13% dd/1
5513 root      4656K 3336K cpu1  100   - 0:00:00  0.0% prstat/1
 523 root      2828K 1732K sleep  59    0 0:00:00  0.0% in.routed/1
1454 noaccess  3332K 1940K sleep  59    0 0:00:00  0.0% asr-notify/2
  43 root      4036K 2528K sleep  29    0 0:00:00  0.0% dlmgmt/13
 136 root      6944K 2824K sleep  29    0 0:00:00  0.0% syseventd/18
 248 daemon    6644K 2540K sleep  29    0 0:00:00  0.0% rcapd/1
  99 netadm    4104K 2708K sleep  29    0 0:00:00  0.0% ipmgmt/5
 110 root      2852K 1032K sleep  29    0 0:00:00  0.0% in.mpathd/1
```

|                                                                 |       |       |       |        |         |      |             |      |                |
|-----------------------------------------------------------------|-------|-------|-------|--------|---------|------|-------------|------|----------------|
| 13                                                              | root  | 17M   | 16M   | sleep  | 29      | 0    | 0:00:08     | 0.0% | svc.configd/18 |
| 11                                                              | root  | 15M   | 12M   | sleep  | 29      | 0    | 0:00:02     | 0.0% | svc.startd/12  |
| 313                                                             | root  | 3564K | 2380K | sleep  | 29      | 0    | 0:00:00     | 0.0% | picld/4        |
| 8                                                               | root  | 0K    | 0K    | sleep  | 99      | -20  | 0:00:00     | 0.0% | vmtasks/2      |
| 7                                                               | root  | 0K    | 0K    | sleep  | 60      | -    | 0:00:00     | 0.0% | intrd/1        |
| PROJID                                                          | NPROC | SWAP  | RSS   | MEMORY | TIME    | CPU  | PROJECT     |      |                |
| 101                                                             | 1     | 1720K | 956K  | 0.0%   | 0:01:04 | 21%  | eng_team2   |      |                |
| 100                                                             | 1     | 1720K | 956K  | 0.0%   | 0:26:54 | 13%  | eng_team1   |      |                |
| 1                                                               | 3     | 13M   | 8480K | 0.2%   | 0:00:00 | 0.0% | user.root   |      |                |
| 0                                                               | 144   | 675M  | 354M  | 8.5%   | 0:00:33 | 0.0% | system      |      |                |
| 10                                                              | 2     | 15M   | 8132K | 0.2%   | 0:00:00 | 0.0% | group.staff |      |                |
| Total: 156 processes, 871 lwps, load averages: 1.80, 1.28, 0.99 |       |       |       |        |         |      |             |      |                |

Let the workloads run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

On which CPU is this process running?

In the lower portion of the `prstat` output, what is the `eng_team2` project's process identifier (PID)?

How many cpu-shares did the `eng_team2` project get (`projects -l`)?

Based on the `prstat` output, which project has the highest priority (more important)?

25. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.

```
root@host01:~# poolstat -r pset -p eng-pool 5 5
id pool          type rid rset          min  max size used load
  1 eng-pool      pset  1 eng-pset1      1    2    1 1.00 1.99
...
```

26. Create a workload for the `eng_team3` project by starting a copy routine that runs in an infinite loop as in steps 20 and 23.

```
root@host01:~# newtask -p eng_team3 \
dd if=/dev/zero of=/dev/null&
```

27. Use the `prstat -JR` command to monitor the resource consumption of `eng_team1`, `eng_team2`, and `eng_team3` projects (workloads).

```
root@host01:~# prstat -JR
PID USERNAME  SIZE   RSS STATE PRI NICE   TIME   CPU PROCESS/NLWP
5512 root      1720K  956K run   1    0   0:08:55  15% dd/1
5474 root      1720K  956K cpu0  2    0   0:31:37  9.0% dd/1
5532 root      1720K  956K run   1    0   0:00:16  6.1% dd/1
3342 root      3352K 2584K sleep  1    0   0:00:00  1.1% bash/1
5533 root      4656K 3340K cpu1 100   -   0:00:00  0.0% prstat/1
523 root      2828K 1732K sleep  59   0   0:00:00  0.0% in.routed/1
```

```

1454 noaccess 3332K 1940K sleep 59 0 0:00:00 0.0% asr-notify/2
 43 root 4036K 2528K sleep 29 0 0:00:00 0.0% dlmgmt/13
136 root 6944K 2824K sleep 29 0 0:00:00 0.0% syseventd/18
248 daemon 6644K 2540K sleep 29 0 0:00:00 0.0% rcapd/1
 99 netadm 4104K 2708K sleep 29 0 0:00:00 0.0% ipmgmt/5
110 root 2852K 1032K sleep 29 0 0:00:00 0.0% in.mpathd/1
 13 root 17M 16M sleep 29 0 0:00:08 0.0% svc.configd/18
 11 root 15M 12M sleep 29 0 0:00:02 0.0% svc.startd/12
313 root 3564K 2380K sleep 29 0 0:00:00 0.0% picld/4
PROJID NPROC SWAP RSS MEMORY TIME CPU PROJECT
 101 1 1720K 956K 0.0% 0:08:55 15% eng_team2
 100 1 1720K 956K 0.0% 0:31:37 9.0% eng_team1
 102 1 1720K 956K 0.0% 0:00:16 6.1% eng_team3
   1 3 13M 8488K 0.2% 0:00:00 1.1% user.root
   3 5 20M 14M 0.3% 0:00:07 0.0% default
Total: 157 processes, 873 lwps, load averages: 2.71, 2.16, 1.65

```

Let the workloads run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

On which CPU is this process running?

In the lower portion of the `prstat` output, what is the `eng_team3` project's process identifier (PID)?

How many `cpu-shares` did the `eng_team2` project get (`project -l`)?

Based on the `prstat` output, which "eng\_team" project has the lowest priority (least important)?

28. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.

```

root@host01:~# poolstat -r pset -p eng-pool 5 5
id pool      type rid rset      min  max size used load
  1 eng-pool  pset  1 eng-pset1  1    2    1 1.00 2.99
...

```

29. Create a workload for the `eng_team4` project by starting a copy routine that runs in an infinite loop as in steps 20, 23, and 26.

```

root@host01:~# newtask -p eng_team4 \
dd if=/dev/zero of=/dev/null&

```

30. Use the `prstat -JR` command to monitor the resource consumption of `eng_team4` projects (workloads).

```

root@host01:~# prstat -JR
PID USERNAME  SIZE  RSS STATE PRI NICE      TIME  CPU PROCESS/NLWP
5512 root      1720K 956K cpu0  1    0   0:01:04  27% dd/1

```

```

5474 root      1720K  956K run      1    0   0:26:54  24% dd/1
5532 root      1720K  956K run      1    0   0:05:18  9.1% dd/1
5541 root      1720K  956K run      1    0   0:00:44  3.7% dd/1
5513 root      4656K 3336K cpu1    100   -   0:00:00  0.0% prstat/1
 523 root      2828K 1732K sleep    59    0   0:00:00  0.0% in.routed/1
1454 noaccess  3332K 1940K sleep    59    0   0:00:00  0.0% asr-notify/2
  43 root      4036K 2528K sleep    29    0   0:00:00  0.0% dlmgmt/13
 136 root      6944K 2824K sleep    29    0   0:00:00  0.0% syseventd/18
 248 daemon    6644K 2540K sleep    29    0   0:00:00  0.0% rcapd/1
  99 netadm    4104K 2708K sleep    29    0   0:00:00  0.0% ipmgmt/5
 110 root      2852K 1032K sleep    29    0   0:00:00  0.0% in.mpathd/1
  13 root        17M   16M sleep    29    0   0:00:08  0.0% svc.configd/18
  11 root        15M   12M sleep    29    0   0:00:02  0.0% svc.startd/12
 313 root      3564K 2380K sleep    29    0   0:00:00  0.0% picld/4
   8 root         0K    0K sleep    99  -20   0:00:00  0.0% vmtasks/2
   7 root         0K    0K sleep    60   -   0:00:00  0.0% intrd/1
PROJID   NPROC   SWAP   RSS MEMORY   TIME   CPU PROJECT
  101         2 3440K 1912K   0.0%   0:35:23  27% eng_team2
  100         2 3440K 1912K   0.0%   0:49:31  24% eng_team1
  102         2 3440K 1912K   0.0%   0:10:16  6.7% eng_team3
    1         3  13M 8500K   0.2%   0:00:00  0.0% user.root
  103         1 1720K  956K   0.0%   0:00:00  0.0% eng_team4
Total: 161 processes, 877 lwps, load averages: 7.55, 5.13, 3.78

```

Let the workloads run for a minute or so.

In the upper portion of the `prstat` output, which process is consuming the bulk of the CPU time?

On which CPU is this process running?

In the lower portion of the `prstat` output, what is the `eng_team4` project's process identifier (PID)?

How many `cpu-shares` did the `eng_team4` project get (`project -l`)?

Based on the `prstat` output, which projects have the highest priority (most important) and lowest priority (least important)?

Note that after a couple minutes, the `eng_team4` project is no longer shown in the `prstat` output. Why?

31. Use the `poolstat` command to examine the `eng-pool` pool resource statistics.

```

root@host01:~# poolstat -r pset -p eng-pool 5 5
id pool          type rid rset          min  max size used load
  1 eng-pool      pset  1 eng-pset1          1    2    1 1.00 2.99
...
```

Note that the `size` field has a value of 1. Why?

32. Use `pkill -J` to kill the `eng_team1`, `eng_team2`, `eng_team3`, and `eng_team4` projects.

```
root@host01:~# pkill -J 100
root@host01:~# pkill -J 101
root@host01:~# pkill -J 102
root@host01:~# pkill -J 103
```

33. Make TS the default class at boot time.

```
root@host01:~# dispadmin -d TS
```

34. Make this configuration take effect immediately, without rebooting.

```
root@host01:~# priocntl -s -c TS -i all
```

## Task 2: Manage Physical Memory Resources

You can control the amount of physical memory that workloads consume by setting memory resource caps. A resource cap is an upper bound placed on the consumption of a resource, such as physical memory. The resource-capping daemon and its associated utilities provide mechanisms for physical memory resource cap enforcement and administration. Like the processor pool resource control, the physical memory resource cap can be defined by using attributes of project entries in the project database. The total amount of physical memory that is available to processes in the project is determined by the `rcap.max-rss` attribute.

In this task, you set physical memory caps for projects you created in Task 1.

Perform the following steps to manage physical memory resources:

1. Determine the amount of memory currently installed in the system.

```
root@host01:~# prtconf | grep Memory
Memory size: 8064 Megabytes
```

2. Determine the status of the `rcap` service. If it is currently disabled, enable it now.

```
root@host01:~# svcs rcap
STATE          STIME      FMRI
disabled       9:31:30    svc:/system/rcap:default
root@host01:~# svcadm enable rcap
root@host01:~# svcs rcap
STATE          STIME      FMRI
online         10:15:24    svc:/system/rcap:default
```

3. Set physical memory caps on the "eng\_team" project by typing:

```
root@host01:~# projmod -a -K rcap.max-rss=50KB eng_team1
root@host01:~# projmod -a -K rcap.max-rss=5MB eng_team2
root@host01:~# projmod -a -K rcap.max-rss=5MB eng_team3
root@host01:~# projmod -a -K rcap.max-rss=50KB eng_team4
```

4. List the projects currently configured in the project database.

```
root@host01:~# projects -l
...
eng_team1
```

```

    projid : 100
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,3,none)
            project.pool=eng-pool
            rcap.max-rss=51200

eng_team2
    projid : 101
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,5,none)
            project.pool=eng-pool
            rcap.max-rss=5242880

eng_team3
    projid : 102
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,2,none)
            project.pool=eng-pool
            rcap.max-rss=5242880

eng_team4
    projid : 103
    comment: ""
    users  : root
    groups : (none)
    attribs: project.cpu-shares=(privileged,0,none)
            project.pool=eng-pool
            rcap.max-rss=51200

```

5. Create a workload for each "eng\_team" project by starting a copy routine that runs in an infinite loop.

```

root@host01:~# newtask -p eng_team1 \
dd if=/dev/zero of=/dev/null&
[1] 5522
root@host01:~# newtask -p eng_team2 \
dd if=/dev/zero of=/dev/null&
[2] 5523
root@host01:~# newtask -p eng_team3 \
dd if=/dev/zero of=/dev/null&
[3] 5524

```

```
root@host01:~# newtask -p eng_team4 \
dd if=/dev/zero of=/dev/null&
[4] 5525
```

6. Use the `rcapstat` utility to access the effectiveness of the physical memory cap set for each project.

```
root@host01:~# rcapstat
```

| id  | project   | nproc | vm   | rss  | cap   | at  | avgat | pg    | avgpg |
|-----|-----------|-------|------|------|-------|-----|-------|-------|-------|
| 100 | eng_team1 | 1     | 148K | 280K | 50K   | 62M | 0K    | 3940K | 0K    |
| 101 | eng_team2 | -     | 148K | 372K | 5120K | 0K  | 0K    | 0K    | 0K    |
| 102 | eng_team3 | -     | 148K | 372K | 5120K | 0K  | 0K    | 0K    | 0K    |
| 103 | eng_team4 | 1     | 148K | 280K | 50K   | 54M | 0K    | 2980K | 0K    |

```

id project      nproc    vm    rss    cap    at avgat    pg avgpg
100 eng_team1    1    148K  280K   50K  1412K 1412K   68K   68K
101 eng_team2    -    148K  372K 5120K    0K    0K    0K    0K
102 eng_team3    -    148K  372K 5120K    0K    0K    0K    0K
103 eng_team4    1    148K  280K   50K  1412K 1412K   68K   68K
...
^c
```

Do you see any problems with the current memory caps set for each project? If so, what is the problem and how does this adversely impact the projects?

7. Use the `vmstat` utility to access system memory.

```
root@host01:~# vmstat 5 5
```

| kthr |   |   | memory  |         | page |     |    |    |    |    |     |    |    |    | disk |     |         |      | faults |    | cpu |  |  |  |
|------|---|---|---------|---------|------|-----|----|----|----|----|-----|----|----|----|------|-----|---------|------|--------|----|-----|--|--|--|
| r    | b | w | swap    | free    | re   | mf  | pi | po | fr | de | sr  | s0 | s1 | s2 | s3   | in  | sy      | cs   | us     | sy | id  |  |  |  |
| 2    | 0 | 0 | 3525924 | 3044060 | 167  | 672 | 4  | 0  | 0  | 0  | 121 | 29 | 0  | 0  | 0    | 761 | 784703  | 1628 | 19     | 10 | 71  |  |  |  |
| 3    | 0 | 0 | 3406572 | 2884760 | 8    | 46  | 10 | 0  | 0  | 0  | 0   | 4  | 0  | 0  | 0    | 662 | 1038494 | 299  | 24     | 10 | 65  |  |  |  |
| 3    | 0 | 0 | 3406564 | 2885052 | 0    | 40  | 7  | 0  | 0  | 0  | 0   | 4  | 0  | 0  | 0    | 662 | 1034846 | 305  | 25     | 10 | 65  |  |  |  |
| 3    | 0 | 0 | 3406564 | 2885060 | 0    | 37  | 8  | 0  | 0  | 0  | 0   | 3  | 0  | 0  | 0    | 641 | 1035780 | 295  | 24     | 10 | 66  |  |  |  |
| 3    | 0 | 0 | 3406564 | 2885076 | 0    | 38  | 0  | 0  | 0  | 0  | 0   | 1  | 0  | 0  | 0    | 645 | 1035474 | 332  | 25     | 10 | 65  |  |  |  |

Note that the physical memory caps you set for the `eng_team1` and `eng_team4` projects is forcing the system to swap unnecessarily.

8. Kill the `eng_team1`, `eng_team2`, `eng_team3`, and `eng_team4` projects.

```
root@host01:~# pkill -J 100
root@host01:~# pkill -J 101
root@host01:~# pkill -J 102
root@host01:~# pkill -J 103
```



## **Practices for Lesson 13: Managing Zone-Wide Resources**

### **Chapter 13**

## Practices for Lesson 13: Overview

---

### Practices Overview

Zone-wide resource controls limit the total resource usage of all process entities within a zone. Any of the resource controls used in the global zone can be used to control resources in the non-global zones. The resource controls and attributes used in a zone to control projects, tasks, and processes within that zone are subject to the additional requirements regarding pools and the zone-wide resource controls.

The key area of resource management explored in this practice is:

- Managing zone-wide resources and controls

Solutions for each task in this practice are provided at the back of this guide.

## Practice 13-1: Managing Zone-Wide Resources and Controls

---

A nonglobal zone can be associated with one resource pool, although the pool need not be exclusively assigned to a particular zone. Multiple nonglobal zones can share the resources of one pool. Processes in the global zone, however, can be bound by a sufficiently privileged process to any pool. The resource controller `poolcd` daemon runs only in the global zone, where there is more than one pool for it to operate on. The `poolstat` utility run in a nonglobal zone displays only information about the pool associated with the zone. The `pooladm` command run without arguments in a nonglobal zone displays only information about the pool associated with the zone.

Zone-wide resource controls do not take effect when they are set in the `project` file. A zone-wide resource control is set through the `zonecfg` utility.

CPU caps provide absolute fine-grained limits on the amount of CPU resources that can be consumed by a project or a zone. CPU caps allow workloads to run on any CPUs while limiting their CPU usage and provide fine-grained (specified in fractions of a CPU) limit. In contrast to FSS, CPU caps provide absolute usage limit that does not depend on other workloads running in the system. CPU caps can be used with CPU binding, processor sets, and FSS. When used in conjunction with processor sets, CPU caps limit CPU usage within a set. When used with FSS, CPU resources defined by caps are further subdivided by using FSS shares.

In this practice, you create a processor pool and assign it to the `web`, `storage`, and `database` zones. You limit (cap) each zone CPU and memory resources as follows:

The `web` zone:

- CPU consumption is limited to 15% of one CPU.
- Cap physical memory to 50 MB.

The `storage` zone:

- CPU consumption is limited to 100% of one CPU.
- Cap physical memory to 500 MB.

The `database` zone:

- CPU consumption is limited to 100% of three CPUs.
- Cap physical memory to 2 GB.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Create a Zone Resource Pool

In this task, you create a processor pool to be used by the `web`, `storage`, and `database` zones.

Perform the following steps to create the zone resource pool:

1. Log in to server as user `oracle` and `su` to `root`.
2. Determine the number of CPUs currently installed in the system.
3. Determine the amount of memory currently installed in the system.
4. Determine the status of the processor pools service. If it is currently disabled, enable it now.
5. Determine the status of the `rcap` service. If it is currently disabled, enable it now.

6. Create a processor set (`pset`) named `zone-pset1` that contains a minimum of one CPU and a maximum of four CPUs.
7. Create a resource pool named `zone-pool`.
8. Associate the `zone-pset1` processor set with the `zone-pool` pool.
9. Display the dynamic pool configuration.
10. Commit the dynamic pool configuration.

## Task 2: Manage the web Zone Resources

In this task, you assign the resource pool to the `web` zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 15% of one CPU.
- Cap physical memory to 50 MB.

Perform the following steps to configure the `web` zone CPU and memory resources:

1. Edit the `web` zone configuration.

```
root@host01:~# zonecfg -z web
zonecfg:web>
```

2. Associate the `zone-pool` pool with the `web` zone.

```
zonecfg:web> set pool=zone-pool
zonecfg:web> verify
```

3. Cap the number of CPUs to 15% of one CPU.

```
zonecfg:web> add capped-cpu
zonecfg:web:capped-cpu> set ncpus=0.15
zonecfg:web:capped-cpu> end
zonecfg:web> verify
```

4. Specify the memory limits for the `web` zone.

```
zonecfg:web> add capped-memory
zonecfg:web:capped-memory> set physical=50m
zonecfg:web:capped-memory> set swap=100m
zonecfg:web:capped-memory> set locked=25m
zonecfg:web:capped-memory> end
zonecfg:web> verify
```

5. Commit the `web` configuration and exit the edit session.

```
zonecfg:web> commit
zonecfg:web> exit
```

6. Display the `web` zone configuration.

7. Reboot the `web` zone.

## Task 3: Manage the storage Zone Resources

In this task, you assign the resource pool to the `storage` zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 100% of one CPU.
- Cap physical memory to 500 MB.

Perform the following steps to configure the storage zone CPU and memory resources:

1. Edit the storage zone configuration.

```
root@host01:~# zonecfg -z storage
zonecfg:storage>
```

2. Associate the zone-pool pool with the storage zone.

```
zonecfg:storage> set pool=zone-pool
zonecfg:storage> verify
```

3. Cap the number of CPUs to 100% of one CPU.

```
zonecfg:storage> add capped-cpu
zonecfg:storage:capped-cpu> set ncpus=1
zonecfg:storage:capped-cpu> end
zonecfg:storage> verify
```

4. Specify the memory limits for the storage zone.

```
zonecfg:storage> add capped-memory
zonecfg:storage:capped-memory> set physical=500m
zonecfg:storage:capped-memory> set swap=1000m
zonecfg:storage:capped-memory> set locked=100m
zonecfg:storage:capped-memory> end
zonecfg:storage> verify
```

5. Commit the storage configuration and exit the edit session.

```
zonecfg:storage> commit
zonecfg:storage> exit
```

6. Display the storage zone configuration.

7. Reboot the storage zone.

#### Task 4: Manage the database Zone Resources

In this task, you assign the resource pool to the database zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 100% of three CPUs.
- Cap physical memory to 2 GB.

Perform the following steps to configure the database zone CPU and memory resources:

1. Edit the database zone configuration.

```
root@host01:~# zonecfg -z database
zonecfg:database>
```

2. Associate the zone-pool pool with the database zone.

```
zonecfg:database> set pool=zone-pool
zonecfg:database> verify
```

3. Cap the number of CPUs to 100% of three CPUs.

```
zonecfg:database> add capped-cpu
zonecfg:database:capped-cpu> set ncpus=3
zonecfg:database:capped-cpu> end
```

```
zonecfg:database> verify
```

- Specify the memory limits for the database zone.

```
zonecfg:database> add capped-memory
zonecfg:database:capped-memory> set physical=2000m
zonecfg:database:capped-memory> set swap=2000m
zonecfg:database:capped-memory> set locked=500m
zonecfg:database:capped-memory> end
zonecfg:database> verify
```

- Commit the database configuration and exit the edit session.

```
zonecfg:database> commit
zonecfg:database> exit
```

- Display the database zone configuration.
- Reboot the database zone.

### Task 5: Create Zone Workloads

In this task, you create workloads on the web, storage, and database zones.

**Note:** Assume that the dd commands used in this task are representative of your production workloads.

Perform the following steps to create the zone workloads:

- Log in to the web zone.
- Create a workload for the web zone by starting copy routines that runs in an infinite loop.

```
root@web:~# dd if=/dev/zero of=/dev/null&
[1] 20242
root@web:~# dd if=/dev/zero of=/dev/null&
[2] 20209
root@web:~# dd if=/dev/zero of=/dev/null&
[3] 20221
root@web:~# dd if=/dev/zero of=/dev/null&
[4] 20234
```

- Return to the global zone.
- Log in to the storage zone.
- Create a workload for the storage zone by starting copy routines that runs in an infinite loop.

```
root@storage:~# dd if=/dev/zero of=/dev/null&
[1] 14232
root@storage:~# dd if=/dev/zero of=/dev/null&
[2] 14241
root@storage:~# dd if=/dev/zero of=/dev/null&
[3] 14251
root@storage:~# dd if=/dev/zero of=/dev/null&
[4] 14258
```

- Return to the global zone.

7. Log in to the database zone.
8. Create a workload for the database zone by starting copy routines that runs in an infinite loop.

```
root@database:~# dd if=/dev/zero of=/dev/null&
[1] 15194
root@database:~# dd if=/dev/zero of=/dev/null&
[2] 15207
root@database:~# dd if=/dev/zero of=/dev/null&
[3] 15220
root@database:~# dd if=/dev/zero of=/dev/null&
[4] 15232
```

9. Return to the global zone.

### Task 6: Assess Zone Resource Utilization and Performance

In this task, you monitor workloads running in the zone to assess their resource utilization and performance.

Perform the following steps to monitor zone resource utilization:

1. Verify that your memory cap for each zone is working effectively.
2. Use the `rcapstat -z` utility from the global zone.
3. Use the `poolstat` utility to determine whether `zone-pool` is keeping up with workload demands.
4. Use the `prstat -z` utility to assess the workload performance in each zone.  
Based on the information from the commands run in Step 2, do you see a problem with workload performance in any of the zones?
5. Halt the `web` zone and change its CPU cap to 100% of two CPUs.
6. Boot the `web` zone and log in to it.
7. Start the workloads and exit back to the global zone.

```
root@web:~# dd if=/dev/zero of=/dev/null&
[1] 12743
root@web:~# dd if=/dev/zero of=/dev/null&
[2] 12761
root@web:~# dd if=/dev/zero of=/dev/null&
[3] 12773
root@web:~# dd if=/dev/zero of=/dev/null&
[4] 12790
root@web:~# exit
```

8. Use tools like `prstat` to verify that the `web` zone workload performance has improved.
9. Try increasing the workload (running more `dd` commands) in each zone and monitor the resulting performance. Run commands like `vmstat` to gather additional information.
10. When you are done exploring zone performance, kill the workloads (`pkill dd`).



## Solution 13-1: Managing Zone-Wide Resources and Controls

A nonglobal zone can be associated with one resource pool, although the pool need not be exclusively assigned to a particular zone. Multiple nonglobal zones can share the resources of one pool. Processes in the global zone, however, can be bound by a sufficiently privileged process to any pool. The resource controller `poolcd` daemon runs only in the global zone, where there is more than one pool for it to operate on. The `poolstat` utility run in a nonglobal zone displays only information about the pool associated with the zone. The `pooladm` command run without arguments in a nonglobal zone displays only information about the pool associated with the zone.

Zone-wide resource controls do not take effect when they are set in the `project` file. A zone-wide resource control is set through the `zonecfg` utility.

CPU caps provide absolute fine-grained limits on the amount of CPU resources that can be consumed by a project or a zone. CPU caps allow workloads to run on any CPUs while limiting their CPU usage and provide fine-grained (specified in fractions of a CPU) limit. In contrast to FSS, CPU caps provide absolute usage limit that does not depend on other workloads running in the system. CPU caps can be used with CPU binding, processor sets, and FSS. When used in conjunction with processor sets, CPU caps limit CPU usage within a set. When used with FSS, CPU resources defined by caps are further subdivided by using FSS shares.

In this practice, you create a processor pool and assign it to the `web`, `storage`, and `database` zones. You limit (cap) each zone CPU and memory resources as follows:

The `web` zone:

- CPU consumption is limited to 15% of one CPU.
- Cap physical memory to 50 MB.

The `storage` zone:

- CPU consumption is limited to 100% of one CPU.
- Cap physical memory to 500 MB.

The `database` zone:

- CPU consumption is limited to 100% of three CPUs.
- Cap physical memory to 2 GB.

**Note:** The output from the lab commands shown in this practice are examples. Your lab experience should be similar.

### Task 1: Create a Zone Resource Pool

In this task, you create a processor pool to be used by the `web`, `storage`, and `database` zones.

Perform the following steps to create the zone resource pool:

1. Log in to server as user `oracle` and `su` to `root`.
2. Determine the number of CPUs currently installed in the system.

```
root@host01:~# psrinfo
0          on-line   since 04/12/2012 07:29:40
1          on-line   since 04/12/2012 07:29:41
...
```

3. Determine the amount of memory currently installed in the system.

```
root@host01:~# prtconf | grep Memory
Memory size: 16256 Megabytes
```

4. Determine the status of the processor pools service. If it is currently disabled, enable it now.

```
root@host01:~# svcs system/pools
STATE          STIME      FMRI
online         7:30:43  svc:/system/pools:default
Note: If the service is disabled, use svcadm to enable it. For example:
# svcadm enable pool
```

5. Determine the status of the rcap service. If it is currently disabled, enable it now.

```
root@host01:~# svcs rcap
STATE          STIME      FMRI
online         7:32:24  svc:/system/rcap:default
Note: If the service is disabled, use svcadm to enable it. For example:
# svcadm enable rcap
```

6. Create a processor set (pset) named zone-pset1 that contains a minimum of one CPU and a maximum of four CPUs.

```
root@host01:~# poolcfg -c 'create pset zone-pset1 \
(uint pset.min=1; uint pset.max=4) '
Note: If you see the error message "poolcfg: cannot load configuration from /etc/pooladm.conf: No such file or directory", run the command pooladm -s. Then try creating the processor set.
```

7. Create a resource pool named zone-pool.

```
root@host01:~# poolcfg -c 'create pool zone-pool'
```

8. Associate the zone-pset1 processor set with the zone-pool pool.

```
root@host01:~# poolcfg -c 'associate pool zone-pool \
(pset zone-pset1) '
```

9. Display the dynamic pool configuration.

```
root@host01:~# poolcfg -c info
...

pool eng-pool
    boolean pool.active true
    boolean pool.default false
    int      pool.importance 1
    string   pool.comment
    pset     eng-pset1

pool zone-pool
    boolean pool.active true
    boolean pool.default false
    int      pool.importance 1
    string   pool.comment
```

```

        pset      zone-pset1
...
    pset eng-pset1
        int      pset.sys_id -2
        boolean  pset.default false
        uint     pset.min 1
        uint     pset.max 2
        string   pset.units population
        uint     pset.load 0
        uint     pset.size 0
        string   pset.comment

    pset zone-pset1
        int      pset.sys_id -2
        boolean  pset.default false
        uint     pset.min 1
        uint     pset.max 4
        string   pset.units population
        uint     pset.load 0
        uint     pset.size 0
        string   pset.comment

```

10. Commit the dynamic pool configuration.

```

root@host01:~# pooladm -c
root@host01:~# pooladm
system default
    string  system.comment
    int     system.version 1
    boolean system.bind-default true
    string  system.poold.objectives wt-load

pool pool_default
    int      pool.sys_id 0
    boolean  pool.active true
    boolean  pool.default true
    int      pool.importance 1
    string   pool.comment
    pset     pset_default

pool eng-pool
    boolean  pool.active true
    boolean  pool.default false
    int      pool.importance 1

```

```

        string pool.comment
        pset    eng-pset1

    pool zone-pool
        boolean pool.active true
        boolean pool.default false
        int      pool.importance 1
        string   pool.comment
        pset     zone-pset1
...

```

## Task 2: Manage the web Zone Resources

In this task, you assign the resource pool to the web zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 15% of one CPU.
- Cap physical memory to 50 MB.

Perform the following steps to configure the web zone CPU and memory resources:

1. Edit the web zone configuration.

```

root@host01:~# zonecfg -z web
zonecfg:web>

```

2. Associate the zone-pool pool with the web zone.

```

zonecfg:web> set pool=zone-pool
zonecfg:web> verify

```

3. Cap the number of CPUs to 15% of one CPU.

```

zonecfg:web> add capped-cpu
zonecfg:web:capped-cpu> set ncpus=0.15
zonecfg:web:capped-cpu> end
zonecfg:web> verify

```

4. Specify the memory limits for the web zone.

```

zonecfg:web> add capped-memory
zonecfg:web:capped-memory> set physical=50m
zonecfg:web:capped-memory> set swap=100m
zonecfg:web:capped-memory> set locked=25m
zonecfg:web:capped-memory> end
zonecfg:web> verify

```

5. Commit the web configuration and exit the edit session.

```

zonecfg:web> commit
zonecfg:web> exit

```

## 6. Display the web zone configuration.

```
root@host01:~# zonecfg -z web info
zonename: web
zonepath: /zones/web
brand: solaris
autoboot: true
bootargs:
file-mac-profile:
pool: zone-pool
limitpriv:
scheduling-class:
ip-type: shared
hostid:
fs-allowed:
fs:
    dir: /opt/ora
    special: /opt/ora
    raw not specified
    type: lofs
    options: [ro]
net:
    address: 192.168.106.24/24
    allowed-address not specified
    configure-allowed-address: true
    physical: net0
    defrouter not specified
capped-cpu:
    [ncpus: 0.15]
capped-memory:
    physical: 50M
    [swap: 100M]
    [locked: 25M]
rctl:
    name: zone.cpu-cap
    value: (priv=privileged,limit=15,action=deny)
rctl:
    name: zone.max-swap
    value: (priv=privileged,limit=104857600,action=deny)
rctl:
    name: zone.max-locked-memory
    value: (priv=privileged,limit=26214400,action=deny)
```

7. Reboot the web zone.

```
root@host01:~# zoneadm -z web reboot
root@host01:~# zoneadm list -cv | grep web
    1 web      running      /zones/web                solaris  shared
```

### Task 3: Manage the storage Zone Resources

In this task, you assign the resource pool to the `storage` zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 100% of one CPU.
- Cap physical memory to 500 MB.

Perform the following steps to configure the `storage` zone CPU and memory resources:

1. Edit the `storage` zone configuration.

```
root@host01:~# zonecfg -z storage
zonecfg:storage>
```

2. Associate the `zone-pool` pool with the `storage` zone.

```
zonecfg:storage> set pool=zone-pool
zonecfg:storage> verify
```

3. Cap the number of CPUs to 100% of one CPU.

```
zonecfg:storage> add capped-cpu
zonecfg:storage:capped-cpu> set ncpus=1
zonecfg:storage:capped-cpu> end
zonecfg:storage> verify
```

4. Specify the memory limits for the `storage` zone.

```
zonecfg:storage> add capped-memory
zonecfg:storage:capped-memory> set physical=500m
zonecfg:storage:capped-memory> set swap=1000m
zonecfg:storage:capped-memory> set locked=100m
zonecfg:storage:capped-memory> end
zonecfg:storage> verify
```

5. Commit the `storage` configuration and exit the edit session.

```
zonecfg:storage> commit
zonecfg:storage> exit
```

6. Display the `storage` zone configuration.

```
root@host01:~# zonecfg -z storage info
zonename: storage
zonepath: /zones/storage
brand: solaris
autoboot: true
bootargs:
file-mac-profile:
pool: zone-pool
```

```

limitpriv:
scheduling-class:
ip-type: shared
hostid:
fs-allowed:
fs:
    dir: /opt/ora
    special: /opt/ora
    raw not specified
    type: lofs
    options: [ro]
net:
    address: 192.168.106.25/24
    allowed-address not specified
    configure-allowed-address: true
    physical: net0
    defrouter not specified
capped-cpu:
    [ncpus: 1.00]
capped-memory:
    physical: 500M
    [swap: 1000M]
    [locked: 100M]
rctl:
    name: zone.cpu-cap
    value: (priv=privileged,limit=100,action=deny)
rctl:
    name: zone.max-swap
    value: (priv=privileged,limit=1048576000,action=deny)
rctl:
    name: zone.max-locked-memory
    value: (priv=privileged,limit=104857600,action=deny)

```

#### 7. Reboot the storage zone.

```

root@host01:~# zoneadm -z storage reboot
root@host01:~# zoneadm list -cv | grep storage
    2 storage      running      /zones/storage  solaris  shared

```

### Task 4: Manage the database Zone Resources

In this task, you assign the resource pool to the database zone. You limit (cap) zone CPU and memory resources as follows:

- CPU consumption is limited to 100% of three CPUs.
- Cap physical memory to 2 GB.

Perform the following steps to configure the database zone CPU and memory resources:

1. Edit the database zone configuration.

```
root@host01:~# zonecfg -z database
zonecfg:database>
```

2. Associate the zone-pool pool with the database zone.

```
zonecfg:database> set pool=zone-pool
zonecfg:database> verify
```

3. Cap the number of CPUs to 100% of three CPUs.

```
zonecfg:database> add capped-cpu
zonecfg:database:capped-cpu> set ncpus=3
zonecfg:database:capped-cpu> end
zonecfg:database> verify
```

4. Specify the memory limits for the database zone.

```
zonecfg:database> add capped-memory
zonecfg:database:capped-memory> set physical=2000m
zonecfg:database:capped-memory> set swap=2000m
zonecfg:database:capped-memory> set locked=500m
zonecfg:database:capped-memory> end
zonecfg:database> verify
```

5. Commit the database configuration and exit the edit session.

```
zonecfg:database> commit
zonecfg:database> exit
```

6. Display the database zone configuration.

```
root@host01:~# zonecfg -z database info
zonename: database
zonepath: /zones/database
brand: solaris
autoboot: true
bootargs:
file-mac-profile:
pool: zone-pool
limitpriv:
scheduling-class:
ip-type: shared
hostid:
fs-allowed:
fs:
    dir: /opt/ora
    special: /opt/ora
    raw not specified
    type: lofs
```



```

options: [ro]
net:
  address: 192.168.106.26/24
  allowed-address not specified
  configure-allowed-address: true
  physical: net0
  defrouter not specified
capped-cpu:
  [ncpus: 3.00]
capped-memory:
  physical: 2.0G
  [swap: 2.0G]
  [locked: 500M]
rctl:
  name: zone.cpu-cap
  value: (priv=privileged,limit=300,action=deny)
rctl:
  name: zone.max-swap
  value: (priv=privileged,limit=2097152000,action=deny)
rctl:
  name: zone.max-locked-memory
  value: (priv=privileged,limit=524288000,action=deny)

```

#### 7. Reboot the database zone.

```

root@host01:~# zoneadm -z database reboot
root@host01:~# zoneadm list -cv | grep database
3 database      running      /zones/database solaris shared

```

### Task 5: Create Zone Workloads

In this task, you create workloads on the web, storage, and database zones.

**Note:** Assume that the `dd` commands used in this task are representative of your production workloads.

Perform the following steps to create the zone workloads:

#### 1. Log in to the web zone.

```

root@host01:~# zlogin web
...
root@web:~#

```

#### 2. Create a workload for the web zone by starting copy routines that runs in an infinite loop.

```

root@web:~# dd if=/dev/zero of=/dev/null&
[1] 20242
root@web:~# dd if=/dev/zero of=/dev/null&
[2] 20209
root@web:~# dd if=/dev/zero of=/dev/null&

```

```
[3] 20221
root@web:~# dd if=/dev/zero of=/dev/null&
[4] 20234
```

3. Return to the global zone.

```
root@web:~# exit
...
root@host01:~#
```

4. Log in to the storage zone.

```
root@host01:~# zlogin storage
...
root@storage:~#
```

5. Create a workload for the storage zone by starting copy routines that runs in an infinite loop.

```
root@storage:~# dd if=/dev/zero of=/dev/null&
[1] 14232
root@storage:~# dd if=/dev/zero of=/dev/null&
[2] 14241
root@storage:~# dd if=/dev/zero of=/dev/null&
[3] 14251
root@storage:~# dd if=/dev/zero of=/dev/null&
[4] 14258
```

6. Return to the global zone.

```
root@storage:~# exit
...
root@host01:~#
```

7. Log in to the database zone.

```
root@host01:~# zlogin database
...
root@database:~#
```

8. Create a workload for the database zone by starting copy routines that runs in an infinite loop.

```
root@database:~# dd if=/dev/zero of=/dev/null&
[1] 15194
root@database:~# dd if=/dev/zero of=/dev/null&
[2] 15207
root@database:~# dd if=/dev/zero of=/dev/null&
[3] 15220
root@database:~# dd if=/dev/zero of=/dev/null&
[4] 15232
```

9. Return to the global zone.

```
root@database:~# exit
...
root@host01:~#
```

### Task 6: Assess Zone Resource Utilization and Performance

In this task, you monitor workloads running in the zone to assess their resource utilization and performance.

Perform the following steps to monitor zone resource utilization:

1. Verify that your memory cap for each zone is working effectively.
2. Use the `rcapstat -z` utility from the global zone.

```
root@host01:~# rcapstat -z
```

| id | zone     | nproc | vm  | rss | cap   | at | avgat | pg | avgpg |
|----|----------|-------|-----|-----|-------|----|-------|----|-------|
| 4  | web      | -     | 35M | 43M | 50M   | 0K | 0K    | 0K | 0K    |
| 5  | storage  | -     | 36M | 51M | 500M  | 0K | 0K    | 0K | 0K    |
| 6  | database | -     | 35M | 49M | 2000M | 0K | 0K    | 0K | 0K    |

3. Use the `poolstat` utility to determine whether zone-pool is keeping up with workload demands.

```
root@host01:~# poolstat -r pset -p zone-pool 5 5
```

| id  | pool      | type | rid | rset       | min | max | size | used | load |
|-----|-----------|------|-----|------------|-----|-----|------|------|------|
| 3   | zone-pool | pset | 2   | zone-pset1 | 1   | 4   | 4    | 0.00 | 14.1 |
| ... |           |      |     |            |     |     |      |      |      |

4. Use the `prstat -z` utility to assess the workload performance in each zone.

```
root@host01:~# zoneadm list -cv
```

| ID | NAME     | STATUS  | PATH            | BRAND   | IP     |
|----|----------|---------|-----------------|---------|--------|
| 0  | global   | running |                 | solaris | shared |
| 4  | web      | running | /zones/web      | solaris | shared |
| 5  | storage  | running | /zones/storag   | solaris | shared |
| 6  | database | running | /zones/database | solaris | shared |

```
root@host01:~# prstat -z 4
```

| PID   | USERNAME | SIZE  | RSS   | STATE | PRI | NICE | TIME    | CPU  | PROCESS/NLWP |
|-------|----------|-------|-------|-------|-----|------|---------|------|--------------|
| 20242 | root     | 1968K | 1560K | wait  | 59  | 0    | 0:00:53 | 0.1% | dd/1         |
| 20209 | root     | 1968K | 1560K | wait  | 59  | 0    | 0:00:53 | 0.1% | dd/1         |
| 20221 | root     | 1968K | 1560K | wait  | 59  | 0    | 0:00:52 | 0.1% | dd/1         |
| 20234 | root     | 1968K | 1560K | wait  | 59  | 0    | 0:00:53 | 0.1% | dd/1         |
| ...   |          |       |       |       |     |      |         |      |              |

```
root@host01:~# prstat -vz 4
```

| PID   | USERNAME | USR | SYS | TRP | TFL | DFL | LCK | SLP | LAT | VCX | ICX | SCL | SIG | PROCESS/NLWP |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------------|
| 20242 | root     | 1.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 96  | 0   | 12  | 15K | 0   | dd/1         |
| 20209 | root     | 1.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 96  | 0   | 10  | 14K | 0   | dd/1         |
| 20221 | root     | 1.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 96  | 0   | 10  | 13K | 0   | dd/1         |
| 20234 | root     | 1.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 96  | 0   | 9   | 12K | 0   | dd/1         |
| ...   |          |     |     |     |     |     |     |     |     |     |     |     |     |              |

```

root@host01:~# prstat -z 5
PID USERNAME  SIZE    RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
14232 root      1968K 1544K cpu10  52    0 0:00:53  0.7% dd/1
14241 root      1968K 1544K wait   52    0 0:00:53  0.6% dd/1
14251 root      1968K 1544K cpu11  52    0 0:00:52  0.6% dd/1
14258 root      1968K 1544K wait   52    0 0:00:53  0.6% dd/1
...
root@host01:~# prstat -vz 5
  PID USERNAME  USR SYS TRP TFL DFL LCK SLP  LAT  VCX  ICX  SCL  SIG PROCESS/NLWP
14232   root    19 6.6 0.0 0.0 0.0 0.0 0.0  75   0  52 91K   0 dd/1
14241   root    19 6.6 0.0 0.0 0.0 0.0 0.0  75   0  52 91K   0 dd/1
14251   root    19 6.6 0.0 0.0 0.0 0.0 0.0  75   0  52 91K   0 dd/1
14258   root    19 6.6 0.0 0.0 0.0 0.0 0.0  75   0  52 91K   0 dd/1
...
root@host01:~# prstat -z 6
PID USERNAME  SIZE    RSS STATE PRI NICE   TIME    CPU PROCESS/NLWP
15194 root      1968K 1544K cpu8    1    0 12:47:58  3.0% dd/1
15232 root      1968K 1544K cpu9    1    0 12:47:45  2.9% dd/1
15207 root      1968K 1544K run    1    0 12:48:03  2.9% dd/1
15220 root      1968K 1544K cpu10   1    0 12:47:51  2.9% dd/1
root@host01:~# prstat -vz 6
  PID USERNAME  USR SYS TRP TFL DFL LCK SLP  LAT  VCX  ICX  SCL  SIG PROCESS/NLWP
15207   root     52 19 0.0 0.0 0.0 0.0 0.0  29   0  90 .2M   0 dd/1
15194   root     52 19 0.0 0.0 0.0 0.0 0.0  29   0  90 .2M   0 dd/1
15220   root     52 19 0.0 0.0 0.0 0.0 0.0  29   0  90 .2M   0 dd/1
15232   root     52 19 0.0 0.0 0.0 0.0 0.0  29   0  90 .2M   0 dd/1
...

```

Based on the information from the commands run in Step 2, do you see a problem with workload performance in any of the zones?

*The `prstat` `STATE` field for the `web` zone shows that the `dd` processes are waiting for CPU resources from the `zone-pool` resource pool. Also, the `USR` and `CPU` fields show that CPU utilization is very low for each workload. This indicates that the CPU resource capping for the `web` zone was too aggressive, causing a CPU “starvation” condition for the zone workloads.*

5. Halt the web zone and change its CPU cap to 100% of two CPUs.

```

root@host01:~# zoneadm -z web halt
root@host01:~# zonecfg -z web
zonecfg:web> select capped-cpu ncpus=0.15
zonecfg:web:capped-cpu> set ncpus=2
zonecfg:web:capped-cpu> end
zonecfg:web> info
zonename: web
...
capped-cpu:
      [ncpus: 2.00]

```

```

capped-memory:
    physical: 50M
    [swap: 100M]
    [locked: 25M]

rctl:
    name: zone.max-swap
    value: (priv=privileged,limit=104857600,action=deny)

rctl:
    name: zone.max-locked-memory
    value: (priv=privileged,limit=26214400,action=deny)

rctl:
    name: zone.cpu-cap
    value: (priv=privileged,limit=200,action=deny)

zonecfg:web> verify
zonecfg:web> commit
zonecfg:web> exit

```

6. Boot the web zone and log in to it.

```

root@host01:~# zoneadm -z web boot
root@solaris-11-demo:~# zlogin web
[Connected to zone 'web' pts/1]
Oracle Corporation      SunOS 5.11      11.0      November 2011
root@web:~#

```

7. Start the workloads and exit back to the global zone.

```

root@web:~# dd if=/dev/zero of=/dev/null&
[1] 12743
root@web:~# dd if=/dev/zero of=/dev/null&
[2] 12761
root@web:~# dd if=/dev/zero of=/dev/null&
[3] 12773
root@web:~# dd if=/dev/zero of=/dev/null&
[4] 12790
root@web:~# exit

```

8. Use tools like `prstat` to verify that the web zone workload performance has improved.

```

root@host01:~# zoneadm list -cv

```

| ID | NAME     | STATUS  | PATH            | BRAND   | IP     |
|----|----------|---------|-----------------|---------|--------|
| 0  | global   | running |                 | solaris | shared |
| 5  | storage  | running | /zones/storage  | solaris | shared |
| 6  | database | running | /zones/database | solaris | shared |
| 7  | web      | running | /zones/web      | solaris | shared |

```

root@host01:~# prstat -z 7

```

| PID   | USERNAME | SIZE  | RSS   | STATE | PRI | NICE | TIME    | CPU  | PROCESS/NLWP |
|-------|----------|-------|-------|-------|-----|------|---------|------|--------------|
| 12743 | root     | 1968K | 1544K | run   | 1   | 0    | 0:28:17 | 1.6% | dd/1         |

```

12773 root 1968K 1544K run 1 0 0:28:18 1.6% dd/1
12790 root 1968K 1544K cpu11 1 0 0:28:16 1.6% dd/1
12761 root 1968K 1544K cpu10 12 0 0:28:20 1.5% dd/1
...
root@host01:~# prstat -vz 7
  PID USERNAME  USR  SYS TRP  TFL  DFL  LCK  SLP  LAT  VCX  ICX  SCL  SIG  PROCESS/NLWP
12743 root      29   10 0.0 0.0 0.0 0.0 0.0 61   0  49 .1M   0 dd/1
12773 root      28   10 0.0 0.0 0.0 0.0 0.0 62   0  53 .1M   0 dd/1
12761 root      28  9.8 0.0 0.0 0.0 0.0 0.0 63   0  49 .1M   0 dd/1
12790 root      28  9.8 0.0 0.0 0.0 0.0 0.0 63   0  24 .1M   0 dd/1
...

```

9. Try increasing the workload (running more `dd` commands) in each zone and monitor the resulting performance. Run commands like `vmstat` to gather additional information.
10. When you are done exploring zone performance, kill the workloads (`pkill dd`).

```

root@host01:~# zlogin web
[Connected to zone 'web' pts/2]
Oracle Corporation      SunOS 5.11      11.0      November 2011
root@web:~# pkill dd
root@web:~# exit
logout
[Connection to zone 'web' pts/2 closed]
root@host01:~# zlogin storage
[Connected to zone 'storage' pts/1]
Oracle Corporation      SunOS 5.11      11.0      November 2011
root@storage:~# pkill dd
root@storage:~# exit
logout
[Connection to zone 'storage' pts/1 closed]
root@host01:~# zlogin database
[Connected to zone 'database' pts/1]
Oracle Corporation      SunOS 5.11      11.0      November 2011
root@database:~# pkill dd
root@database:~# exit
root@host01:~#

```

# **Practices for Lesson 14: Performance Analysis and Testing**

## **Chapter 14**

## Practices for Lesson 14

---

### Practices Overview

There is no practice for Lesson 14.