

D78767GC10

Edition 1.0

March 2013

D81108

ORACLE®

Oracle Solaris 11 Performance Management

Student Guide - Volume I

Author

David Giroux

Technical Contributors & Reviewers

Mike Carew

Benoit Chaffanjon

Glynn Foster

John Hathaway

Dominic Kay

Steve Kirby

Rosemary Martinak

Kristi McNeill

Chad Mynhier

Christian Wolbert

Editors

Malavika Jinka

Vijayalakshmi Narasimhan

Arijit Ghosh

Graphic Designer

Seema Bopaiah

Publishers

Jayanthi Keshavamurthy

Jobi Varghese

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.

Contents

Preface

1 Introduction

- Overview 1-2
- Goals 1-3
- Course Agenda: Day 1 1-4
- Course Agenda: Day 2 1-6
- Course Agenda: Day 3 1-7
- Course Agenda: Day 4 1-8
- Course Agenda: Day 5 1-9
- Introductions 1-10
- Your Learning Center 1-11

2 Introducing Performance Management

- Objectives 2-2
- Agenda 2-3
- Relevance 2-4
- Performance Management 2-5
- Terminology 2-6
- Performance Graphs 2-9
- Trade-Offs of Performance Tuning 2-10
- Trade-Offs: Example 2-11
- Conceptual Model of Performance 2-12
- Block Functional Diagrams 2-13
- Knowing the Speeds and Feeds 2-14
- Hardware Functional Block Diagram 2-15
- Software Functional Block Diagram 2-16
- Agenda 2-17
- Knowns and Unknowns 2-18
- Drill-Down Analysis Strategy 2-19
- Performance Analysis Approach 2-20
- Agenda 2-22
- Monitoring Tools in Oracle Solaris 11 2-23
- kstat Utility 2-24
- procfs File System 2-25

DTrace	2-26
System Configuration Assessment	2-27
Quiz	2-28
Summary	2-31
Practice 2 Overview: Introducing Performance Management	2-32

3 kstat Monitoring Tools

Objectives	3-2
Relevance	3-3
Identifying Performance Problems Using the kstat Tools	3-4
Agenda	3-5
sar(1)	3-6
Agenda	3-9
vmstat(1)	3-10
vmstat -p	3-11
Agenda	3-12
iostat(M)	3-13
iostat(1M)	3-14
Agenda	3-15
mpstat(1M)	3-16
Agenda	3-17
netstat(1M)	3-18
netstat -i	3-19
Agenda	3-20
nfsstat(1M)	3-21
Agenda	3-23
kstat(1M) Utility	3-24
Quiz	3-29
Summary	3-32
Practice 3 Overview: kstat Monitoring Tools	3-33

4 procfs Monitoring Tools

Objectives	4-2
Relevance	4-3
Agenda	4-4
procfs-Based Tools	4-5
Agenda	4-6
ps Command	4-7
/usr/ucb/ps Command	4-11
Agenda	4-13
prstat(1M) Utility	4-14

- prstat(1M) Utility Options 4-15
- prstat Output: Example 4-17
- Examining Processes 4-19
- Agenda 4-21
- truss(1) Utility 4-22
- Agenda 4-28
- Solaris Studio Performance Analyzer 4-29
- Quiz 4-30
- Summary 4-33
- Practice 4 Overview: Using ps, prstat, the proc Tools, and truss 4-34

5 Introduction to DTrace

- Objectives 5-2
- Relevance 5-3
- Agenda 5-4
- DTrace 5-5
- DTrace Architecture 5-6
- Agenda 5-7
- DTrace Command Syntax 5-8
- DTrace Providers 5-9
- DTrace One-Liners 5-11
- Running a DTrace One-Liner 5-12
- Agenda 5-13
- DTrace Toolkit 5-14
- Agenda 5-16
- DTrace Visualization Tool (DLight) 5-17
- DLight GUI 5-18
- Quiz 5-19
- Summary 5-21
- Practice 5 Overview: Using DTrace 5-22

6 Other Significant Tools

- Objectives 6-2
- Relevance 6-3
- Agenda 6-4
- swap(1M) 6-5
- Agenda 6-7
- CPU Performance Counters 6-8
- cpustat(1M) Utility 6-9
- trapstat(1M) Utility 6-14
- Agenda 6-15

- mdb(1) Utility 6-16
- Agenda 6-28
- Solaris Studio dbx(1) Utility 6-29
- Agenda 6-30
- zonestat Utility 6-31
- zonestat Utility: Examples 6-32
- Agenda 6-35
- GUDDS Script 6-36
- Quiz 6-37
- Summary 6-40
- Practice 6 Overview: Using swap, bus counters, and mdb Tools 6-41

7 Processes and Threads

- Objectives 7-2
- Relevance 7-3
- Agenda 7-4
- Layers of the Operating System 7-5
- Process 7-6
- Life Cycle of a Process 7-7
- Typical Life Cycle of a Process 7-8
- System Processes 7-10
- Process and Address Space 7-11
- pmap(1) 7-12
- Interrupts 7-14
- Interrupt Assignments 7-16
- Interrupt Controls 7-17
- Interprocess Communication (IPC) 7-18
- Process Kernel Structures 7-19
- Process-Related Performance Issues 7-20
- Agenda 7-21
- Threads 7-22
- pmap(1): MT Process 7-23
- Threads 7-24
- Locks 7-25
- Locking Problems 7-26
- lockstat Utility 7-27
- DTrace lockstat Provider 7-33
- adaptive_mutex.d 7-34
- Performance Threads 7-35
- Agenda 7-36
- Process-Related Tunable Parameters 7-37

Showing the Maximum Number of Processes and Limitations	7-39
Agenda	7-44
Process Scheduling	7-45
Scheduling State Diagram	7-47
Thread States	7-48
Scheduling Classes	7-49
Priorities	7-50
Timeshare	7-51
Interactive (IA) Scheduling Class	7-52
Fixed Priority (FX)	7-54
System (SYS)	7-55
Real-Time (RT)	7-56
Fair Share (FSS)	7-57
Quiz	7-59
Summary	7-62
Practice 7 Overview: Processes and Threads	7-63

8 System Caches and Buses

Objectives	8-2
Relevance	8-3
Agenda	8-4
Introducing Caches	8-5
Caches and Buses	8-6
Memory Hierarchy	8-7
Relative Access Times	8-8
System Caches	8-9
Cache Operation	8-11
Replacing Cache Data	8-12
Requesting Data from a Cache	8-13
Factors Affecting the Cache-Hit Rate	8-14
Effects of CPU Cache Misses	8-15
Oracle CPU Caches	8-16
Agenda	8-17
UltraSPARC T-Series Processor Family	8-18
CMT Architecture	8-19
CMT Thread Model	8-20
CMT Pipeline	8-21
Performance Issues and CMT	8-23
pgstat Command	8-26
pginfo Command	8-27
Agenda	8-28

System Buses	8-29
Peripheral Buses	8-30
busstat(1M) Command	8-31
prtdiag(1M) Command	8-32
Diagnosing Bus Problems	8-35
prtconf(1M) Command	8-36
cputrack(1M) Command	8-39
Quiz	8-41
Summary	8-44
Practice 8 Overview: System Caches and Buses	8-45

9 Memory

Objectives	9-2
Agenda	9-3
Main Memory	9-4
Non-Uniform Memory Access (NUMA)	9-5
lgrpinfo Command	9-6
Agenda	9-7
Virtual Memory 2 (VM2)	9-8
Agenda	9-10
MMU	9-11
Translation Lookaside Buffer (TLB)	9-12
Multiple Page Size Support (MPSS)	9-13
Implementing MPSS for a Running Process	9-14
trapstat Utility (SPARC Only)	9-15
Paging and Swapping	9-16
Clock Algorithm	9-17
fastscan and handspread_pages	9-18
Page Scanner Processing	9-19
General Memory-Related Tuning Parameters	9-20
Paging-Related Tuning Parameters	9-22
Additional Paging-Related Tuning Parameters	9-24
Swapping	9-26
Swapping Priorities	9-27
Swap Space	9-28
Swap-Related Tuning Parameters	9-29
Agenda	9-30
Intimate Shared Memory (ISM)	9-31
Dynamic Intimate Shared Memory (DISM)	9-32
Optimized Shared Memory (OSM)	9-34
Agenda	9-35

Memory Monitoring	9-36
Memory Summary	9-37
Memory Consumption	9-38
vmstat(1m) Utility	9-39
Identifying Paging Statistics	9-40
Using sar	9-41
Using the vmstat Command	9-42
Using kstat	9-43
Per-Process Paging Activity	9-44
DTrace Toolkit Scripts	9-45
Swapping Statistics	9-46
Memory Requirements for Applications	9-47
I/Os Queued to a Swap Device	9-48
Measuring the Distribution of Memory	9-49
Solaris Studio discover Utility	9-50
Quiz	9-51
Summary	9-54
Practice 9 Overview: Memory	9-55

10 Disk I/O and the ZFS File System

Objectives	10-2
Agenda	10-3
Disk Performance Considerations	10-4
Storage Data Characteristics	10-5
Disk Bottlenecks	10-6
Disk Utilization	10-7
Disk Saturation	10-8
Disk Configuration	10-9
Disk File Systems	10-10
HDDs Versus SSDs	10-11
Agenda	10-13
iostat Command	10-14
vmstat Command	10-15
fsstat Command	10-17
sar -d Command	10-18
zpool iostat Command	10-19
DTrace Disk I/O One-Liners	10-20
Oracle I/O Numbers Calibration Tool (ORION)	10-21
Agenda	10-22
Oracle Solaris ZFS	10-23
ZFS Design	10-24

ZFS Architecture: Overview	10-25
Interface Layer	10-26
Transactional Object Layer	10-27
Data Management Unit	10-28
ZFS Intent Log (ZIL)	10-29
ZAP Layer	10-30
Dataset and Snapshot Layer (DSL)	10-31
Pooled Storage Layer	10-32
ZFS ARC Cache Introduction	10-33
ZFS Data Structures	10-34
vdev Tree	10-35
vdev Label	10-36
Uberblock	10-38
General ZFS Administration	10-39
ZFS Limitations	10-40
Agenda	10-41
General Storage Tuning	10-42
ZFS Tuning Guidelines	10-43
General Storage Pools Considerations	10-45
Root Pool Considerations	10-46
Non-Root Pool Considerations	10-47
ZFS RAID-Z: Examples	10-48
Network-Attached Storage Considerations	10-49
ZFS Storage Pool: Maintenance and Monitoring	10-50
ZFS File System Considerations	10-52
Agenda	10-53
ZFS ARC Cache	10-54
Viewing ZFS ARC Statistics	10-56
ZFS ARC Cache Tuning Parameters	10-58
Additional ZFS Tuning Parameters	10-59
Viewing ZFS Kernel Parameters	10-61
Quiz	10-62
Summary	10-64
Practice 10 Overview: Disk I/O and the ZFS File System	10-65

11 Solaris 11 Network Tuning

Objectives	11-2
Agenda	11-3
Relevance	11-4
Introducing Network Performance	11-5
Terms Used for Network Analysis	11-6

Packets	11-7
Network Utilization	11-8
Network Errors	11-9
Effects of Misconfigured Components	11-10
Agenda	11-11
Oracle Solaris 11 Networking	11-12
Oracle Solaris 10 Network Model	11-13
Oracle Solaris 11 Network Model	11-14
Agenda	11-15
Data Packet Encapsulation	11-16
Reconfiguring the MTU	11-18
IP Multipathing (IPMP)	11-19
IPMP Configurations	11-20
Configuring IPMP: Active-Active	11-21
Configuring IPMP: Active-Standby	11-22
Link Aggregation	11-23
Configuring Link Aggregation	11-24
Network Virtualization	11-25
Consolidating to Virtual Networking	11-26
Bandwidth Management	11-27
Managing Bandwidth	11-28
Integrated Load Balancer (ILB)	11-29
ILB Example: DSR	11-30
ILB Example: NAT	11-31
ILB Example Using Zones	11-32
ILB Components and Terms	11-33
Load-Balancing Rule	11-35
ILB Algorithms	11-36
ilbadm Utility	11-37
Agenda	11-38
Monitoring Network Performance	11-39
Testing Response Time	11-40
Network Status	11-41
kstat Command (Network)	11-42
ipadm Utility	11-43
dladm Utility	11-44
Introducing the snoop Command	11-45
snoop Command	11-46
wireshark Utility	11-47
flowstat Utility	11-48
flowstat: Examples	11-49

traceroute Utility	11-50
Testing the Reliability of Packet Sizes	11-51
dlstat Utility	11-52
dlstat: Examples	11-53
TCP Statistics from DTrace	11-55
IP Statistics from DTrace	11-56
ICMP Statistics from DTrace	11-57
Agenda	11-58
Displaying and Setting Network Tunable Parameters	11-59
IP Tuning Parameters	11-61
IP Tuning Parameters: ipadm	11-63
TCP Connections	11-64
TCP Tuning Parameters	11-65
TCP Tuning Parameters: ipadm	11-67
NFS Tuning Parameters	11-68
Additional Network Tuning Parameters	11-74
Per-Route Tuning	11-76
Isolating Problems	11-77
Quiz	11-78
Summary	11-81
Practice 11 Overview: Monitoring Network Performance	11-82

12 Resource Management

Objectives	12-2
Agenda	12-3
Resource Management	12-4
When to Use Resource Management	12-5
Agenda	12-7
Projects and Tasks	12-8
Project Database	12-10
Project and Task Commands	12-11
Displaying the Project Database	12-12
Adding a Project to the Project Database	12-13
Agenda	12-14
Resource Management Control Mechanisms	12-15
Resource Control Enforcement	12-17
Resource Control Values, Privileges, and Actions	12-18
Configuring Resource Controls in a Project	12-20
Resource Control Commands	12-22
Agenda	12-23
Fair-Share Scheduler (FSS)	12-24

- Agenda 12-26
- Resource Pools 12-27
- Dynamic Resource Pools 12-28
- Resource Pool Properties 12-29
- Configuration Pool Objectives 12-31
- Resource Pool Commands 12-32
- Configuring Static Resource Pools 12-33
- Configuring Objectives 12-36
- Adding FSS to a Pool 12-37
- poolstat Command 12-38
- Agenda 12-39
- Resource Capping 12-40
- Resource-Capping Commands 12-41
- rcapstat Command 12-42
- Quiz 12-43
- Summary 12-47
- Practice 12: Overview 12-48

13 Oracle Solaris Virtualization Performance Management

- Objectives 13-2
- Agenda 13-3
- How Zones Work 13-4
- Global Zone Characteristics 13-6
- Nonglobal Zone Characteristics 13-7
- Agenda 13-8
- zonestat Utility 13-9
- zonestat Utility: Examples 13-10
- Agenda 13-13
- Zone-Wide Resource Controls 13-14
- rctl Resource Control 13-16
- rctl Resource Properties 13-17
- Zone-Wide Resource Control: Examples 13-18
- Agenda 13-21
- Oracle VM for SPARC 13-22
- CPU Whole Cores and CPU Cap 13-23
- Viewing CPU Whole Cores Configurations 13-24
- CPU Threading Modes and Workloads 13-25
- Viewing CPU Threading Modes 13-26
- Agenda 13-27
- Quiz 13-28
- Summary 13-33
- Practice 13: Overview 13-34

14 Performance Analysis and Testing

Objectives	14-2
Relevance	14-3
Agenda	14-4
Introduction	14-5
Maintaining System Performance	14-6
Performance Analysis Approach	14-7
Errors	14-8
Misconfigurations	14-9
Eliminate Bottlenecks	14-10
Fine-Tuning	14-11
Viewing the Values of Tuning Parameter	14-12
Setting Tuning Parameters	14-13
Recovering /etc/system File Settings	14-14
Unknown Bugs	14-15
Agenda	14-16
Types of Performance Testing	14-17
Industry Benchmarks	14-19
Agenda	14-20
Performance-Testing Tools	14-21
Workload Assessment	14-22
Performance-Testing Tools: File Systems	14-23
Performance-Testing Tools: Network	14-24
Performance-Testing Tools: CPU	14-25
Performance-Testing Tools: Memory	14-26
Functional Diagram	14-27
Understand Benchmark Software	14-29
Sanity Test	14-30
Double-Check	14-31
Drive Resources to Saturation	14-32
Document Your Test System	14-33
Testing File Servers	14-34
Ways to Avoid Client Caching	14-35
Distribute Client Load	14-36
Disk Matter	14-37
Check Your Storage Profile	14-38
Summary	14-39

Preface

Profile

Before You Begin This Course

Before you begin this course, you should be able to:

- Employ advanced systems administration skills in a networked Oracle Solaris 11 OS server environment
- Manage pseudo and distributed file systems
- Create and manage logical volumes
- Configure a network adapter and use `ndd` to modify its settings

How This Course Is Organized

Oracle Solaris 11 Performance Management is an instructor-led course featuring lectures and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills that are introduced.

Related Publications

Oracle Publications

Title	Part Number
<i>System Administration Guide: Basic Administration</i>	821-1451-xx
<i>System Administration Guide: Advanced Administration</i>	821-1452-xx
<i>System Administration Guide: Solaris Zones, Oracle Solaris 10 Containers, Resource Management</i>	821-1460-xx
<i>Oracle Solaris ZFS Administration Guide</i>	821-1448-xx
<i>DTrace User Guide</i>	819-5488-xx
<i>Solaris Tunable Parameters Reference Manual</i>	821-1450-xx

Related Publications

Additional Publications

- System release bulletins
- Installation and user's guides
- *read.me* files
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

The following two lists explain Oracle University typographical conventions for words that appear within regular text or within code samples.

1. Typographic Conventions for Words Within Regular Text

Convention	Object or Term	Example
Courier New	User input; commands; column, table, and schema names; functions; PL/SQL objects; paths	Use the <code>SELECT</code> command to view information stored in the <code>LAST_NAME</code> column of the <code>EMPLOYEES</code> table. Enter <code>300</code> . Log in as <code>scott</code>
Initial cap	Triggers; user interface object names, such as button names	Assign a When-Validate-Item trigger to the ORD block. Click the Cancel button.
Italic	Titles of courses and manuals; emphasized words or phrases; placeholders or variables	For more information on the subject see <i>Oracle SQL Reference</i> <i>Manual</i> Do <i>not</i> save changes to the database. Enter <i>hostname</i> , where <i>hostname</i> is the host on which the password is to be changed.
Quotation marks	Lesson or module titles referenced within a course	This subject is covered in Lesson 3, “Working with Objects.”

2. Typographic Conventions for Words Within Code Samples

Convention	Object or Term	Example
Uppercase	Commands, functions	SELECT employee_id FROM employees;
Lowercase, italic	Syntax variables	CREATE ROLE <i>role</i> ;
Initial cap	Forms triggers	Form module: ORD Trigger level: S_ITEM.QUANTITY item Trigger name: When-Validate-Item . . .
Lowercase	Column names, table names, filenames, PL/SQL objects	. . . OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer')) . . . SELECT last_name FROM employees;
Bold	Text that must be entered by a user	CREATE USER scott IDENTIFIED BY tiger ;

1

Introduction

ORACLE

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

Overview

- Course goals
- Course agenda
- Introductions
- Your learning center

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

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

Welcome to the *Oracle Solaris 11 Performance Management* course. This course introduces you to the performance tuning principles, monitoring utilities, and tuning tools for the Oracle Solaris 11 Operating System (Oracle Solaris 11 OS). The presentation includes a review of the Solaris subsystems and utilities provided to monitor system efficiency, including `kstat`, `sar`, `vmstat`, `iostat`, `netstat`, `mpstat`, `nfsstat`, `ps`, `prstat`, `pmap`, the `proc` tools, `truss`, `dtrace`, the DTrace Toolkit, `cpustat`, `cputrack`, `swap`, `lockstat`, and `mdb`.

This revision also includes a section on resource management using zones and pools, as well as coverage of some of the Oracle Solaris Studio utilities and capabilities. The course format is divided into three major sections: tools, OS theory, and strategy and actions. The *Oracle Solaris 11 Performance Management* course includes practices to reinforce skills development.

To begin, we would like to take about 45 minutes to give you an introduction to the course. We'll start with the course goals, followed by the agenda, and introductions. We'll conclude with a few notes that are specific to your learning center.

Goals

The goals of this course are to enable you to:

- Use the Oracle Solaris OS and third-party tools to analyze performance
- View and set tunable parameters
- Monitor and report on process and thread activity
- Describe system caches and system buses
- Modify CPU scheduling and virtual memory operations
- Describe disk input/output (I/O) and ZFS file systems and network subsystems tuning
- Describe and perform resource management
- Describe and perform virtualization tuning

ORACLE

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

Course Agenda: Day 1

- Lesson 1: Introduction
- Lesson 2: Introducing Performance Management
 - Introduction to Performance Management
 - Performance Analysis Concepts
 - Monitoring Tools in Oracle Solaris 11
- Lesson 3: The kstat Monitoring Tools
 - The sar tool
 - The vmstat tool
 - The iostat tool
 - The mpstat tool
 - The netstat tool
 - The nfsstat tool
 - The kstat tool

ORACLE

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

Course Agenda: Day 1

- Lesson 4: The procfs Monitoring Tools
 - Introduction to procfs-based tools
 - The ps command
 - The prstat(1) utility
 - The truss(1) utility
 - Other significant tools

ORACLE

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

Course Agenda: Day 2

- Lesson 5: DTrace
 - DTrace providers
 - DTrace toolkit
 - DTrace visualization tools
- Lesson 6: Other Significant Tools
 - The swap utility
 - CPU performance counters
 - The mdb(1) utility
 - The Solaris Studio dbx(1) utility
- Lesson 7: Processes and Threads
 - Introduction to operating system theory
 - Process concepts
 - Threads and locking
 - Process-related tunable parameters and process limits
 - Process scheduling

ORACLE

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

Course Agenda: Day 3

- Lesson 8: System Caches and Buses
 - Cache concepts
 - The `prtdiag(1M)` command
 - The `prtconf(1M)` command
 - The `cputrack(1M)` command
- Lesson 9: Memory
 - Memory concepts
 - Swapping, paging, and caching

ORACLE

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

Course Agenda: Day 4

- Lesson 10: Disk I/O and File Systems
 - Disk I/O–related concepts
 - ZFS and related concepts
- Lesson 11: Solaris 11 Network Tuning
 - Network concepts
 - Monitoring network performance
 - TCP/IP

ORACLE

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

Course Agenda: Day 5

- Lesson 12: Resource Management
 - Projects
 - Resource pools
 - Resource capping
- Lesson 13: Oracle Solaris Virtualization Performance Management
 - Controlling zone resource
 - Monitoring zone resource consumption
 - Controlling VM for SPARC resource
- Lesson 14: Performance Analysis and Testing
 - Performance analysis
 - Performance testing

ORACLE

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

Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to topics presented in this course
- Reasons for enrolling in this course
- Expectations for this course

ORACLE

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

Your Learning Center

- Logistics
 - Restrooms
 - Break rooms and designated smoking areas
 - Cafeteria and restaurants in the area
- Emergency evacuation procedures
- Instructor contact information
- Cell phone usage
- Online course attendance confirmation form

ORACLE

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

2

Introducing Performance Management

ORACLE

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

Objectives

After completing this lesson, you will be able to:

- Describe the principles of performance tuning
- Describe the performance tuning process
- Define the terms used to describe performance aspects
- Describe the drill-down analysis strategy
- Describe the performance analysis approach
- Identify the Oracle Solaris observability tools
- List the `kstat`-based and `procfs`-based utilities
- List the DTrace-based utilities
- List the configuration assessment utilities

ORACLE

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

Agenda

- Introduction to Performance Management
- Performance Analysis Concepts
- Monitoring Tools in Oracle Solaris 11

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- How can you tell if a system is performing well or poorly?
- Name the performance objectives that you consider most important for the systems you support.
- What tools or benchmarks does your company now use to assess system performance?

ORACLE

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

Performance Management

- Perceptions and expectations
- Proactive tuning
 - Knowing your application requirements
 - Knowing your system hardware and OS
 - Tuning
 - Resource management
 - Benchmarking
- Reactive tuning
 - Observation
 - Knowing the available tools
 - Interpreting them
 - Tuning
 - Configuration files and tools
 - Tunable parameters
 - Experimentation: Devising load test

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

Defining acceptable system performance depends on the interests and role of the observer. End users, for example, often focus on expedience (response time). By contrast, an IT manager who plans resource usage against a budget needs to know how much work a system can process over time (throughput). Company executives who need to forecast costs may want to know how well-utilized a system is.

Terminology

- **Bandwidth:** Refers to the maximum theoretical speed of a bus
- **Throughput:** Refers to the achieved data flow rate, such as on a bus; usually implies over a period of time, or explicitly stated as sustained throughput
- **Utilization:** Shows which capacity of a resource was consumed over a time interval. 100% utilized means that a resource is always processing requests, and cannot handle any more.
- **Busy:** Shows the percentage of time for which a resource was active. 100% busy, unlike utilization, does not mean that the resource cannot handle more.

ORACLE

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

Bandwidth is often used to describe the maximum signal (“data”) that can be processed or passed between two points in a fixed period of time. The product of a component’s clock rate and the number of data lines it has is a common formula for calculating the component’s bandwidth, or raw physical capability.

Some manufacturers specify burst and sustained transfer rates for their components. These terms are often used to define the capability of a component using a shared bus, such as PCI or SCSI.

Utilization describes the percentage of elapsed time that a component or service spends processing requests. Utilization by itself can be a tricky number to interpret, for several reasons:

- Low utilization may be a symptom of several causes: lack of available work, a bottleneck in a component that sends data, or underestimation of the production capacity of the system.
- High utilization rates can indicate either very efficient use of a resource or a bottleneck. A 100% utilized system might in fact be saturated, and operating with very poor response time. But high utilization in conjunction with few pending requests may show a system that is running at top efficiency.

Terminology

- Saturation: A measure to which a resource is over committed, after it is already 100% utilized
- Latency: Time for an event (for example, I/O) to be fully processed. This includes service times and wait times.
- Wait Time: Time spent waiting or in a queue for an event to be serviced
- Service Time: Time for a single component to complete an event

ORACLE

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

A queue of waiting requests can be used as a measure of saturation.

Response time, at any level of analysis, is the sum total of:

- Wait time, or the time spent in a queue or lock pool until a needed resource is accessible
- Service time, or the time it takes a given resource to complete every required operation

Whenever you measure the time it takes for control to return after a request, you are considering the response time of the request itself.

Wait time values can express how much of the overall response time can be attributed to contending for required resources. In other cases, low wait time may express how efficient a system is at providing services, such as allocating memory or creating a new process.

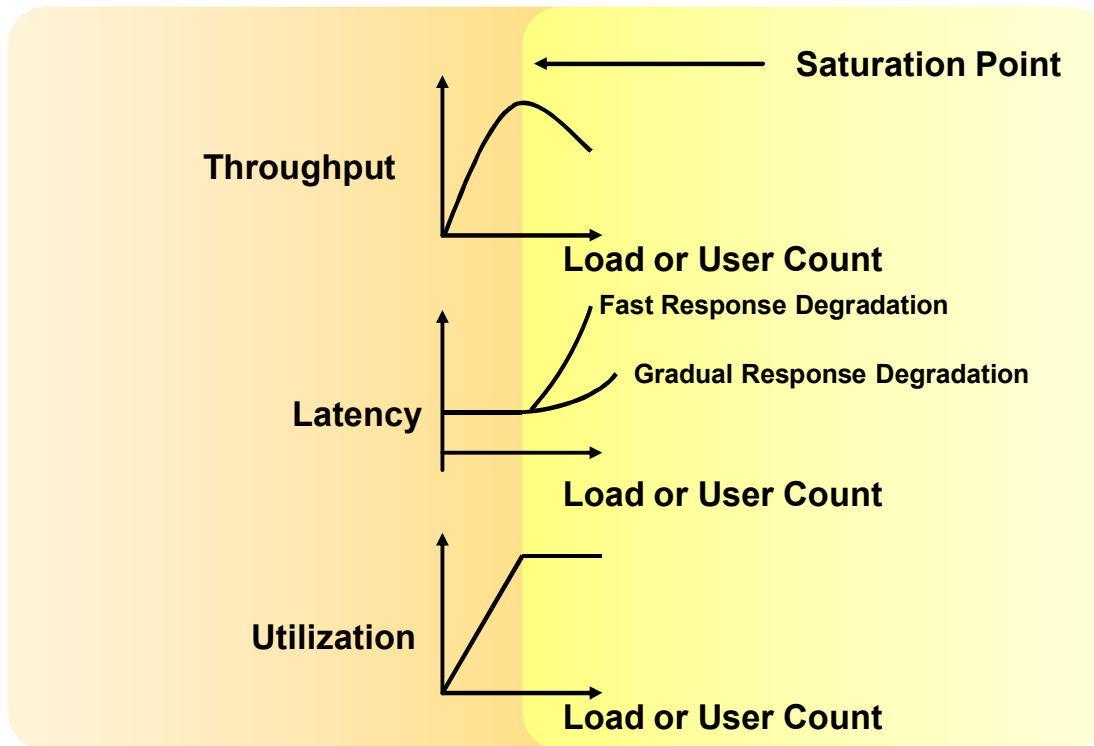
Terminology

- Working Set: For instance, the amount of “hot” data that is in frequent use. The operating system (OS) endeavors to cache the working set in dynamic RAM (DRAM) if possible.
- Errors: Software- or hardware-induced failures, which often can cause performance degradation

ORACLE

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

Performance Graphs



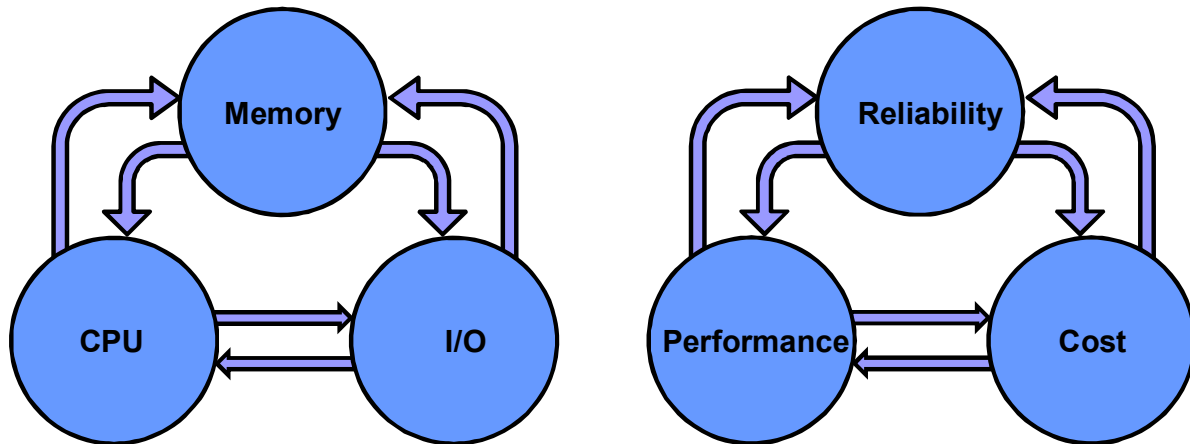
ORACLE

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

The graph in the slide shows how system throughput and response time degrade when a system loses its capacity to do more work. Performance is acceptable up to a point, and then it begins to drop. Without the utilization graph, it would be difficult to see saturation as the problem. Using only the utilization graph, however, it would be difficult to see that “total” utilization might degrade overall performance.

The vertical line drawn across all the graphs shows total utilization of system capacity. At this point, the system is saturated. All subsequent requests are then queued while waiting for resources, eventually increasing response times. Furthermore, throughput is degraded because more time must be spent to manage queued requests.

Trade-Offs of Performance Tuning



ORACLE

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

Assuming that system resources are constant, it is generally true that achieving one of these goals comes with the sacrifice of the other two. To make a highly available system perform at its peak, for example, usually involves greater cost. A low-cost system that performs well, by contrast, will not include the redundancy features of a highly available system.

Trade-Offs: Example

- Understand that trade-offs are often at play. The reason behind a performance issue may be a choice the customer has made.
 - Memory versus CPU: Lookup tables
 - Memory versus I/O: Paging and swapping
 - CPU versus Memory: DTrace buffer policies
 - Capacity versus Performance: ZFS Raid 5 versus Mirroring
 - Cost versus Performance: Trunked 1-Gb Ethernet versus 10-Gb Ethernet
 - Reliability versus Performance: `autoup`, logging
- Consider trade-offs for different types of RAID.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

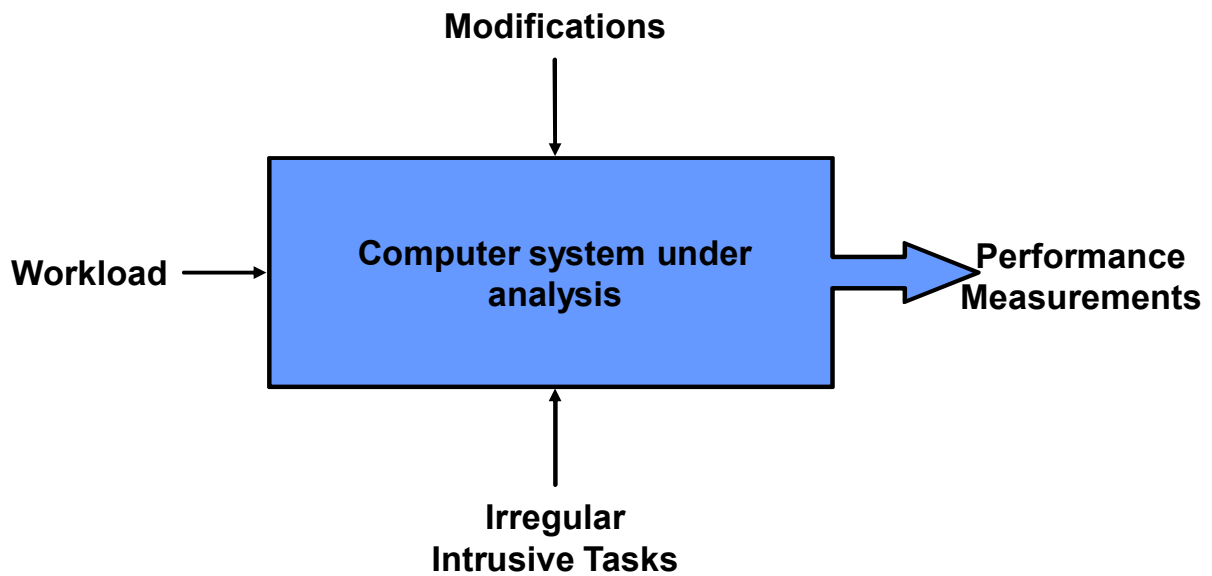
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Consider a higher-level case in which an administrator considers the mount options available for a UFS instance. One way to increase the availability of a file system is to enable journaling. The system administrator can:

- Enable journaling for faster file system recovery after a crash, although journaling adds computational overhead
- Disable journaling to minimize run-time overhead, although the slow recovery times of an `fsck` integrity check means less overall availability in the event of system crashes

Journaling in UFS organizes reads and writes in a more efficient manner, thereby improving input/output (I/O) performance.

Conceptual Model of Performance



ORACLE

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

- **Workload:** This refers to the processes that are normally scheduled to run on the system.
- **Irregular or intrusive tasks:** Some processes may be poorly suited to the target system, such as network-based backup jobs on a busy server or batch job loads running on a multiuser system. In some cases, it may not actually be an irregular task, but the result of regular business cycles that happen to coincide—for example, daily cycles, weekly cycles, monthly cycles, and so on.
- **Modifications:** Changes in system resources, including device driver, firmware kernel updates, or changing kernel tunable parameters, are a potential source of performance changes.

Block Functional Diagrams

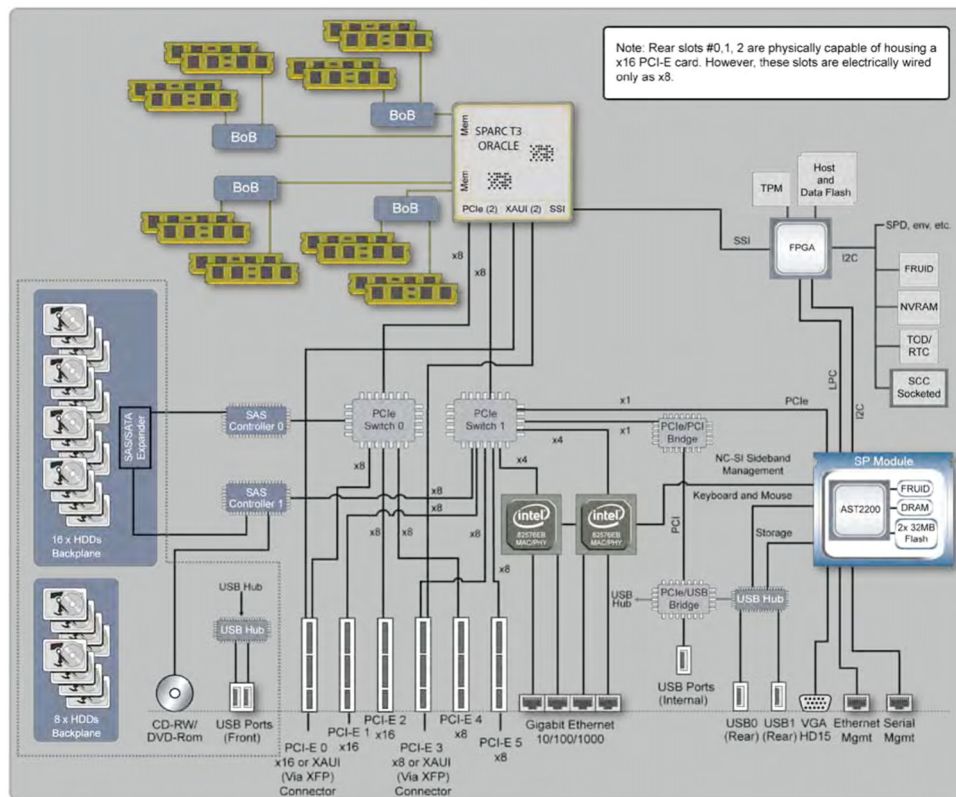
Much of performance tuning is related to locating bottlenecks.

- Consider the entire path that the data (or code) must traverse.
- Block functional diagrams can assist in keeping track of all the components, both hardware and software, in the data path.
- The next few slides show two examples of functional block diagrams that can aid in diagnosis.
- DTrace can be of great assistance for detailing the software path.

ORACLE

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

Knowing the Speeds and Feeds

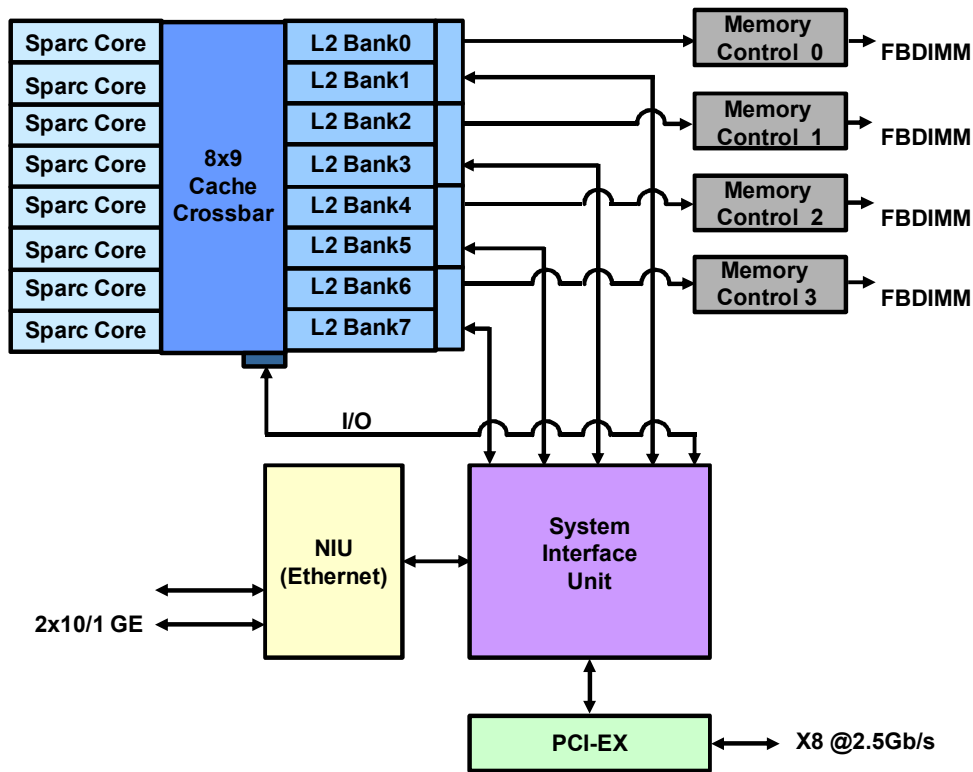


ORACLE

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

An essential step when you tune a server is to identify the major components that make up the system. A system diagram is an ideal tool for putting the server “speeds and feeds” into perspective. This diagram in the slide shows the layout of a typical T-series server.

Hardware Functional Block Diagram

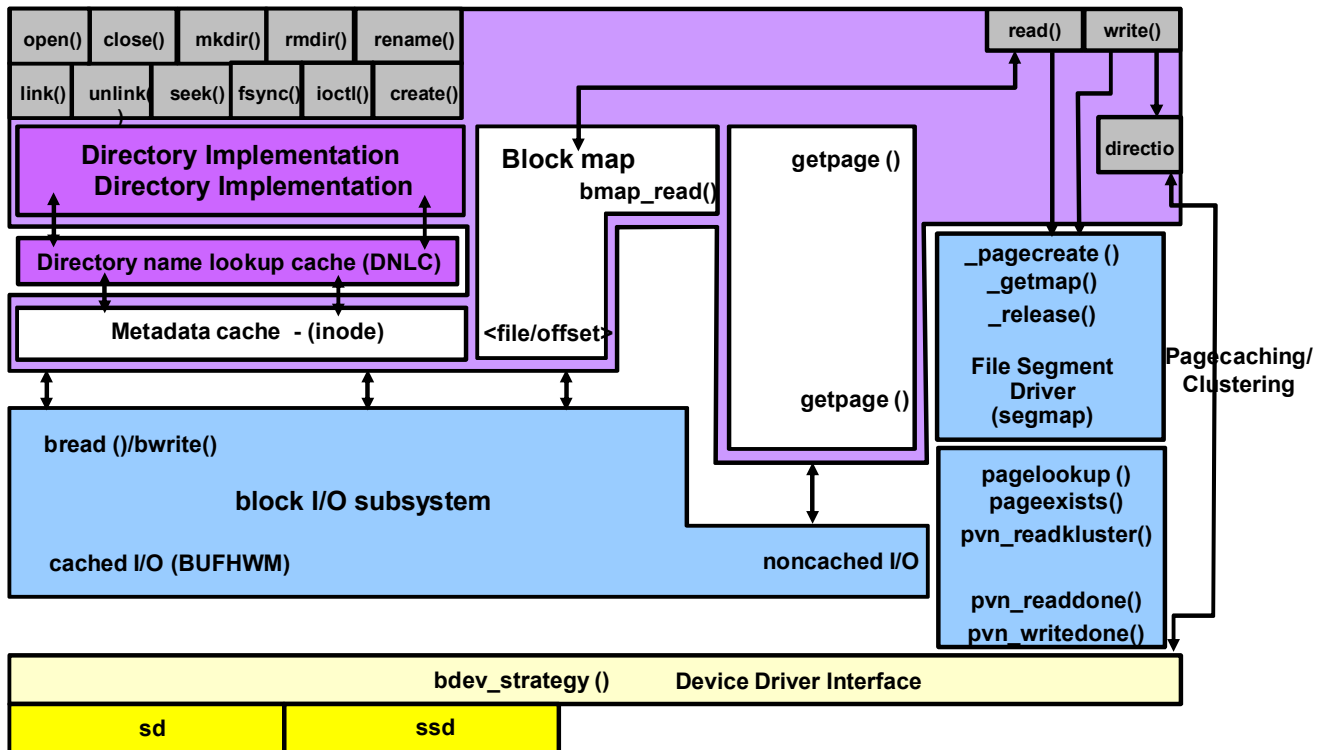


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide is meant to serve only as an example of a good functional block diagram for hardware diagnosis.

It comes from the paper titled "UltraSPARC T2: A Highly-Threaded, Power-Efficient, SPARC SOC" by M. Shah and others.

Software Functional Block Diagram



ORACLE

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

The diagram in the slide is meant to serve only as an example of a good functional block diagram for software diagnosis.

It is derived from *Solaris Internals Core Kernel Architecture* (page 578).

Agenda

- Introduction to Performance Management
- **Performance Analysis Concepts**
- Monitoring Tools in Oracle Solaris 11

ORACLE

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

Knowns and Unknowns

- When tracing data flow, for every component, identify:
 - Utilization
 - Saturation
 - Errors
- Even if you cannot, knowing that you cannot has value.
- It is turning unknown “unknowns” into known “unknowns.”

ORACLE

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

Refer to the article titled “What Data Is Needed In Order to Troubleshoot My Solaris Performance Issue?” on <https://support.oracle.com>. This article is designed to help you identify and detail the performance problem.

Drill-Down Analysis Strategy

1. Monitoring
 - Central monitoring: `orca`, `nagios`
2. Identification
 - System-wide analysis: `vmstat`, `mpstat`, `iostat`, `kstat`
 - GUDS
3. Analysis
 - Process and component analysis: `truss`, `dtrace`

ORACLE

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

- For more information about Orca, see the website at <http://www.sun.com/bigadmin/features/articles/orca.html>.
- You can download the GUDS script from. <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=SCRIPT&id=1285485.1>

Performance Analysis Approach

Response to a Problem

1. Errors
2. Misconfigurations
 - Check software and firmware versions so that you do not experience known bugs, for instance, by using Oracle Explorer.
3. Load
 - Understand workload requirements.
 - Eliminate unnecessary work—for example, 10 windows open at the same time.
4. Bottlenecks
 - Upgrade a saturated resource, for instance.

ORACLE

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

Oracle Explorer can be obtained from the Oracle Services Tools Bundle website:
<http://www.oracle.com/us/support/systems/premier/services-tools-bundle-sun-systems-163717.html>

Performance Analysis Approach

5. Fine-tuning
 - Solaris tunable parameters
 - Reference manual
 - Apache httpd.conf
 - MySQL my.cnf etc.
 - Tuning one thing at a time
6. Known and unknown bugs

ORACLE

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

This approach is explained in detail in the lesson titled “Applying the Performance Analysis Approach.”

Agenda

- Introduction to Performance Management
- Performance Analysis Concepts
- **Monitoring Tools in Oracle Solaris 11**

ORACLE

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

Monitoring Tools in Oracle Solaris 11

kstat	procfs	DTrace
vmstat (1M)	ps (1)	lockstat (1M)
iostat (1M)	prstat (1M)	plockstat (1M)
mpstat (1M)	truss (1)	intrstat (1M)
sar (1)	pmap (1)	dtrace (1M)

ORACLE

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

Most of the observability tools are based on `kstat`, `procfs`, and `DTrace`, which will be covered more thoroughly in the next three lessons. Tools such as `cpustat`, `swap`, and `mdb` that do not fit into one of these categories will be covered in the lesson titled “Other Significant Tools.”

If you want to determine the interface that a given command falls into, use `truss`—for example:

- `# truss -ftioctl vmstat:` To see `kstat` reads
- `# truss -ftopen ps -ef:` To see opens in `/proc`
- `# truss command:` To see system calls

kstat **Utility**

- Used to name and format the key performance statistics gathered by the OS. To list them all, use `kstat -l`.
- Format of the `kstat` description:
 - `module:instance:name:statistic`
 - `module`: Kernel driver or subsystem
 - `instance`: Instance of the driver
 - `name`: Named group of statistics within a module
 - `statistic`: Name of the kernel statistic

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

See `usr/include/sys/kstat.h` and `man -s 1M kstat` for more details.

procfs **File System**

- `procfs` provides a file system interface for accessing information about processes and LWPs.
- It is used by debuggers and process tools such as `ps(1)`, `truss(1)`, `pmap(1)`, `ptree(1)`, and `prstat(1M)`.
- `/proc` contains an entry for each PID.
- Subdirectories and files provide uniform, well-defined access to process and LWP information—for example:
 - `psinfo`: Information about the process
 - `lpsinfo`: Information about LWPs
 - `sigact`: Information about signal disposition

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

See `man -s 4 proc`.

DTrace

- DTrace is the Solaris Dynamic Tracing Framework.
- It allows users to dynamically trace events of interest in both user and kernel modes.
- DTrace provides:
 - D – The DTrace programming language
 - `dtrace` – The dynamic tracing compiler and tracing utility
- DTrace one-liners
- DTrace ToolKit

ORACLE

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

For more information about DTrace, see the lesson titled “DTrace.”

System Configuration Assessment

Subcommand	Description
<code>prtconf</code>	Prints the system configuration information, including the total amount of memory and system peripherals formatted as a device tree
<code>psrinfo</code>	Displays information about processors
<code>format</code>	Displays information about the disks in your system
<code>zpool status</code>	Displays information about ZFS pools
<code>zfs list</code>	Displays information about ZFS file systems
<code>iscsiadm list</code>	Displays iSCSI information for the initiator node on the host
<code>dladm show*</code>	Displays information about network links, virtual devices, bridges, aggregations, and Wi-Fi
<code>ipadm show*</code>	Displays information about IP addresses, network interfaces, and IP properties

ORACLE

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

A critical task when you perform system tuning is assessing system configuration. Understanding how your system is configured provides a more complete picture when you analyze performance problems. It is not unusual during your analysis to find that the root cause of poor performance is associated with system configuration.

Quiz

Which Oracle Solaris 11 utility enables users to dynamically trace events of interest in both user and kernel modes?

- a. procfs
- b. DTrace
- c. kstat
- d. None of the above

ORACLE

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

Answer: b

Quiz

The format of the `kstat` description is
`module:instance:name:statistic`.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

Identify the stages involved in the drill-down performance analysis strategy.

- a. Monitoring
- b. Identification
- c. Analysis
- d. All of the above

ORACLE

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

Answer: d

Summary

In this lesson, you should have learned how to:

- Describe the principles of performance tuning
- Describe the performance tuning process
- Define the terms used to describe performance aspects
- Describe the drill-down analysis strategy
- Describe the performance analysis approach
- Identify the Oracle Solaris observability tools
- List the `kstat`-based and `procfs`-based utilities
- List the DTrace-based utilities
- List the configuration assessment utilities

ORACLE

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

Practice 2 Overview: Introducing Performance Management

This practice covers the following topics:

- Assessing system configuration
- Using the `kstat` Utility
- Using the `truss` Utility

ORACLE

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

kstat Monitoring Tools



ORACLE

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

Objectives

After completing this lesson, you will be able to describe and use the following:

- sar
- vmstat
- iostat
- mpstat
- netstat
- nfsstat
- kstat

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- What symptoms of system behavior might suggest that it is not running as effectively as possible?
- How do performance tools help you to monitor a system?

ORACLE

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

Identifying Performance Problems Using the `kstat` Tools

Tool	Reports
<code>sar</code>	Overall performance of the system
<code>vmstat</code>	Virtual memory statistics
<code>iostat</code>	Input/output (I/O)–related statistics
<code>mpstat</code>	Statistics for each processor
<code>netstat</code>	Network–related statistics
<code>nfsstat</code>	Network File System (NFS)–related statistics
<code>kstat</code>	Direct access to kernel statistics

ORACLE

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

The `kstat` data structures maintained in the kernel store the data that the performance monitoring tools collect. These tools extract raw data from these structures and report it in a tabular format.

You can specify the time interval for which the performance monitoring tools should report data. Short intervals offer the most granular view of performance. Long-interval data collection, using time intervals of 20 minutes or more, should be used for more general trend analysis.

Agenda

- `sar`
- `vmstat`
- `iostat`
- `mpstat`
- `netstat`
- `nfsstat`
- `kstat`

ORACLE

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

sar (1)

The `sar (1)`, system activity reporter, command collects and reports information about overall system performance. The `sar` utility provides several reports, including:

- File system and system calls
- Physical and kernel memory usage
- Paging and swap activities
- Block device and TTY activities
- Status of processes
- Interprocess communication (IPC) activity and CPU usage

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

sar (1)

- Is useful for collecting a baseline via `crontab`
- Makes it easy to collect information in a binary file

```
# sar -Ao sarfile 5 999
# sar -pgwf sarfile
# sar -quf sarfile
```
- Is very old and is not being updated
- Has some misleading fields
- Does not have micro-accounting

ORACLE

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

sar (1)

Example:

```
# sar -c 5 12
SunOS unknown 5.10 Generic_120012-14 i86pc 04/16/2009
14:59:24  scall/s   sread/s   swrit/s   fork/s   exec/s   rchar/s   wchar/s
14:59:29      1059        47        29      0.00     0.00     4460     4035
14:59:34      1625       121       72      0.00     0.00     8584     7878
14:59:39      3783       500      226     4.40     3.40    119911    22739
14:59:44      2544       521       37      0.00     0.00   2368956    13679
14:59:49      5351      1465      698     0.80     0.60  1000982   759660
```

To get more detail, use `dtrace`. For example:

```
# dtrace -n 'syscall:::entry{@[execname,pid] = count();}'
```

ORACLE

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

In the output in the slide, note the following fields:

- `scall/s` – The number of system calls per second
- `sread/s` – The number of read system calls per second
- `swrit/s` – The number of write system calls per second
- `fork/s` – The number of fork system calls per second
- `exec/s` – The number of exec system calls per second
- `rchar/s` – The number of characters read per second
- `wchar/s` – The number of characters written per second

Agenda

- sar
- vmstat
- iostat
- mpstat
- netstat
- nfsstat
- kstat

ORACLE

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

vmstat (1)

The `vmstat` utility reports the performance data related to the virtual memory of a system.

# vmstat 3 4																					
kthr		memory				page				disk				faults				cpu			
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	dd	s0	--	--	in	sy	cs	us	sy	id
29	0	0	444512	8080	18	79	32	11	11	0	0	2	0	25	0	565	2283	1180	5	94	1
33	0	0	446080	8376	19	88	24	27	27	0	0	7	2	35	0	588	1900	946	6	91	3
35	0	0	447876	9980	3	122	51	0	0	0	0	5	0	21	0	550	1943	1142	6	94	1
30	0	0	449176	11032	0	111	0	0	0	0	0	1	0	10	0	423	1496	901	6	94	0

ORACLE

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

The most useful fields in the preceding output are:

- `kthr` – The number of kernel threads in dispatch queues (`r`), the number of blocked kernel threads (`b`), and the number of swapped-out LWPs (`w`)
- `memory` – The amount of free memory in KB (`free`) and the amount of unreserved swap space in KB (`swap`)
- `page` – Pages reclaimed (`re`), KB paged in (`pi`), KB paged out (`po`), KB freed (`fr`), and the number of pages scanned (`sr`)
- `faults` – The number of interrupts (`in`), system calls (`sy`), and context switches (`cs`)
- `cpu` – The percentage usage of CPU time spent in user (`us`), system (`sy`), and idle (`id`) time
- The `sr` (scan rate) field, which represents the number of pages that the page daemon looks at per second. A non-zero value for this field indicates that the system is short of memory. If this happens, one way to get a more detailed breakout of the paging activity is to use `vmstat -p`.

Refer to the article titled “How to use DTrace and mdb to Interpret vmstat Statistics” on <https://support.oracle.com> for more information about `vmstat` command output.

`vmstat -p`

The `vmstat -p` command breaks up paging into executable, anonymous, and file system pages.

- Executable pages – Contain instructions or code
- File system pages – Are backed up by regular files
- Anonymous pages – Do not have a name in the file system, including:
 - Stack
 - Heap
 - Copy On Write (modified data)

ORACLE

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

A non-zero value for `apo` indicates memory shortage because only the page daemon or the swapper writes pages to swap, that is, anonymous pages.

- `epi` – Executable pages paged in per second
- `epo` – Executable pages paged out per second
- `epf` – Executable pages freed per second
- `api` – Anonymous pages paged in per second
- `apo` – Anonymous pages paged out per second
- `apf` – Anonymous pages freed per second
- `fpi` – File system pages paged in per second
- `fpo` – File system pages paged out per second
- `fpf` – File system pages freed per second

Agenda

- sar
- vmstat
- iostat
- mpstat
- netstat
- nfsstat
- kstat

ORACLE

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

iostat (M)

The `iostat` utility reports data about the terminal, disk, and tape I/O activities—for example:

```
# iostat -xn 5
extended device statistics
r/s      w/s      kr/s      kw/s wait  actv wsvc_t  asvc_t  %w  %b device
268.0    195.0    747.6     855.9  0.0  13.0   0.0     28.1   1  98 c1t0d0
289.6      0.0   2442.5       0.0  0.0   7.2   0.0     24.8   0  97 c1t1d0
0.0     328.3      0.0  33796.1  0.0  26.9   0.0     82.0   1  84 c1t2d0
0.0      0.0      0.0      0.0  0.0   0.0   0.0      0.0   0   0 c0t0d0
0.0     42.7      0.0   4744.7  0.0   2.3   0.0     55.1   0   9 c1t3d0
```

- Other useful options include `-p`, `-z`, `-c`, `-C`, and `-e`.
- For storage arrays, throughput (`kr/s` and `kw/s`) numbers may be the only useful statistics.

ORACLE

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

In the output in the slide, note the following fields:

- `r/s` – The number of reads per second
- `w/s` – The number of writes per second
- `kr/s` – The number of KB read per second
- `kw/s` – The number of KB written per second
- `wait` – The average number of transactions waiting for service
- `actv` – The average number of transactions that are being serviced
- `svc_t` – The average service time in milliseconds
- `%w` – The percentage of time there are transactions waiting for service
- `%b` – The percentage of time for which the disk is busy

Note: If you are using a volume manager, the `iostat` utility might not display enough information about the volumes in a disk. You can use the `vxstat` utility provided with VERITAS Volume Manager (VxVM) to report I/O data on VxVM-created logical volumes.

iostat(1M)

The output of `iostat -En` is useful to view all the devices, as well as the error information.

```
# iostat -En
c1t0d0      Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: FUJITSU Product: MAP3367N SUN36G Revision: 0401 Serial No:
0403N07JCM
Size: 36.42GB <36420075008 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0
c1t1d0      Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: FUJITSU Product: MAP3367N SUN36G Revision: 0401 Serial No:
0403N07JCF
Size: 36.42GB <36420075008 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0
c1t2d0      Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
...(output omitted)
```

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

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

Agenda

- sar
- vmstat
- iostat
- **mpstat**
- netstat
- nfsstat
- kstat

ORACLE

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

mpstat(1M)

The `mpstat` utility reports CPU activity on a per-processor basis.

- The following output of the `mpstat` utility is taken from a system with four CPUs:

# mpstat 2 30																
CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl	
0	0	2	69	399	79	731	99	81	10	0	4039	44	7	0	49	
1	0	3	234	624	502	809	90	67	28	0	7131	43	8	0	49	
2	0	2	32	122	21	962	60	36	28	0	3076	74	4	0	22	
3	4	0	28	254	48	1290	120	50	29	0	3544	65	5	0	30	

- This output can be shown per processor set with the `-a`, `-p`, and `-P` options.

ORACLE

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

In the output in the slide, the following fields are included:

- `CPU` – The CPU ID on the system
- `minf` – The number of minor faults
- `mjf` – The number of major faults
- `xcal` – The number of interprocessor cross-calls
- `intr` – The number of interrupts
- `ithr` – The number of interrupts handled as threads
- `csw` – The number of context switches
- `icsw` – The number of involuntary context switches
- `migr` – The number of thread migrations to other processors
- `smtx` – The number of times mutex locks are not acquired at the first attempt
- `srw` – The number of times rw locks are not acquired at the first attempt
- `syscl` – The number of system calls
- `usr sys wt idl` – The percentage of user, system, wait (always 0), and idle time on the CPU

Refer to the article titled “Using DTrace to understand `mpstat` output” on <https://support.oracle.com> for more information about `mpstat` command output.

Agenda

- sar
- vmstat
- iostat
- mpstat
- netstat
- nfsstat
- kstat

ORACLE

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

netstat(1M)

- The `netstat` utility reports on a variety of network data and run-time states.
- The default output lists active sockets per protocol. Other forms of the `netstat` command include reports on the following:
 - Interface state and inbound and outbound statistics (`netstat -i`)
 - Socket state
 - Dynamic Host Configuration Protocol (DHCP) state (`netstat -D`)
 - Routing tables (`netstat -r`)
 - STREAMS statistics (`netstat -m`)
 - Multicast routing table (`netstat -M`)

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, is positioned on the right side of a red horizontal bar.

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

netstat -i

The following is a sample output of the `netstat -i` utility:

```
# netstat -i
Name Mtu  Net/Dest      Address  Ipkts  Ierrs  Opkts  Oerrs  Collis  Queue
lo0   8232  loopback      localhost 17367   0    17367   0      0       0
hme0 1500    SUN           SUN      150518  84    161769  42     1532    0
qfe0 1500    router-qfe0   router-qfe0 6821    0     4658    0      115     0
qfe1 1500    router-qfe1   router-qfe1 4241    0     1156    0       0     0
qfe2 1500    router-qfe2   router-qfe2 0        0      952     0       0     0
qfe3 1500    router-qfe3   router-qfe3 902      0     1989    0       0     0
...<output omitted>...
```

ORACLE

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

In the output in the slide, note the following fields:

- **Name** – The name and instance of the Ethernet interface
- **Mtu** – The maximum size of the packets that are transmitted on the network
- **Net/Dest** – The network to which the interface is attached
- **Address** – The address of the interface on the attached network
- **Ipkts** – The number of packets received
- **Ierrs** – The number of received packets that have errors
- **Opkts** – The number of packets sent
- **Oerrs** – The number of sent packets that incurred errors
- **Collis** – The number of output packets that result in collision
- **Queue** – The number of packets that are queued

Agenda

- sar
- vmstat
- iostat
- mpstat
- netstat
- **nfsstat**
- kstat

ORACLE

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

`nfsstat(1M)`

The `nfsstat` utility reports on the NFS activity and the remote procedure call (RPC) interfaces. The `nfsstat` utility provides options that report on the following:

- Client-side information
- Server-side information
- NFS-mounted file system information
- NFS and RPC information

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

The `nfsstat` utility provides information about both host and client systems. You can use this utility to locate a problem in a network. You can also use the output of the utility to analyze the RPC activity on a system. This information relates to network activity for many client/server activities, such as NFS or any of the several DBMS packages.

nfsstat(1M)

To show only the main sections of the report:

```
# nfsstat | grep ':'  
Server rpc:  
Connection oriented:  
Connectionless:  
Server nfs:  
Version 2: (0 calls)  
Version 3: (0 calls)  
Version 4: (0 calls)  
Version 4: (0 operations)  
Server nfs_acl:  
Version 2: (0 calls)  
Version 3: (0 calls)  
Client rpc:  
... <output omitted>
```

ORACLE

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

The output for one `nfsstat` interval exceeds 100 lines. In the output, there is a breakdown of server and client statistics for `nfs`, `rpc`, and `nfs_acl` activities. The subheadings under each “server” or “client” heading indicate a breakout among session types (connection-oriented and connectionless), NFS protocol versions (2 through 4), and `nfs_acl` protocol versions.

Note: Refer to the man pages for full details on the fields reported and the options available with the `nfsstat` utility.

Agenda

- sar
- vmstat
- iostat
- mpstat
- netstat
- nfsstat
- kstat

ORACLE

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

kstat(1M) Utility

This utility examines the kernel statistics published by a kernel subsection, such as a driver or module.

```
kstat [-lpq] [-T u | d ] [-c class] [-m module]
[-i instance] [-n name] [-s statistic][interval [count]]
```

- The `-m`, `-i`, `-n`, `-s` options enable you to list the statistics with the matching module, instance, name, and statistic. This allows useful grouping of related statistics, such as:

```
# kstat -n vm
# kstat -m zfs
```

- The `-l` option allows you to list matching `kstats` without values.

ORACLE

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

The Oracle Solaris OS counts and stores performance-relevant data as part of its normal operations. These data values are referred to as *statistics*, or *kstats*. These *kstats* name and formalize key performance-relevant events in the kernel. Each *kstat* has a name. Library calls such as `kstat_lookup(3KSTAT)` provide the means to report *kstat* content through utilities such as `sar`.

```
$ kstat -m cpu_info
module: cpu_info           instance: 0
name:  cpu_info0           class: misc
      clock_MHz             502
      cpu_type               sparcv9
      crtime                 51.243930052
      fpu_type               sparcv9
      snaptime               1640.715883529
      state                  on-line
      state_begin            1037429029
```

Note that the `cpu_info` *kstat* reports on component properties (`clock_MHz`, `cpu_type`, `fpu_type`, and `state`) instead of activity.

kstat(1M) Utility

```
# kstat -n vm
module: cpu                      instance: 0
name:   vm                      class: misc
      anonfree                  1936908
      anonpgin                  56932
      anonpgout                 1923861
      as_fault                  5092571
      cow_fault                 1174058
      crtime                    184.8701526
      dfree                     3674361
      execfree                  9616
      execpgin                  29926
... <output omitted>
      zfod                      9535668
module: cpu                      instance: 1
name:   vm                      class: misc
      anonfree                  24928
...<output omitted>
```



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

The information provided by `kstat -n vm` is also available as a provider in DTrace. You will notice that these statistics provide information for `vmstat` and `vmstat -p`. The advantage of accessing information from DTrace is that it allows you to see who is causing the activity in the virtual memory subsystem, as well as how much activity is being generated.

Another provider offered in DTrace is the `sysinfo` provider, which enables DTrace to observe the activity that can be seen through `kstat -n sys`.

kstat(1M) Utility

```
kstat [-lpq] [-T u | d ] [-c class]
[module:instance:name:statistic]... [interval [count]]
```

- This format allows a `kstat` to be specified by its `module:instance:name:statistic`.

```
# kstat caps:4:swapresv_zone_4:usage
module: caps                instance: 4
name:   swapresv_zone_4     class:
zone_caps
      usage                150052864
```

- `-p` displays the output in parsable format:

```
# kstat -p caps:4:swapresv_zone_4:usage
caps:4:swapresv_zone_4:usage 150052864
```

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

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

kstat(1M) Utility

- Statistics are collected per CPU.
- Values represent an accumulated count since boot.
- The number of statistics can grow as needed.
- `kstat` can specify an interval and count.
- `-T` prints a time stamp.

```
# kstat -T d -p caps:4:swapresv_zone_4:usage
Thu Dec 17 14:53:59 2009
caps:4:swapresv_zone_4:usage 150052864
```

- `kstat` enables grouping the output by class, such as `bus`, `device_error`, `disk`, `hat`, `kmem_cache`, `net`, or `nfs`.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

By collecting information in per CPU structure, this utility avoids lock contention when updating statistics. Because data is not shared between CPUs, it does not have to be locked each time it is updated.

kstat(1M) Utility

- To list the modules:

```
# kstat -l | awk -F: '{print $1}' | sort -u
```

- To list the names or statistics:

```
# kstat -l | awk -F: '{print $3}' | sort -u  
# kstat -l | awk -F: '{print $4}' | sort -u
```

- To find a stat, pipe to `grep`.
- To find the stat for the number of processes or the number of threads on the system

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered on a red rectangular background.

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

Note

Refer to `man -s 3KSTAT kstat` for more details.

Quiz

The `mpstat` utility reports CPU activity on a per-processor basis.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

The `nfsstat` utility provides options that report on:

- a. Client-side and server-side information
- b. NFS-mounted file system information
- c. NFS and RPC information
- d. All of the above

ORACLE

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

Answer: d

Quiz

The advantage of accessing information from DTrace is that it enables you to see who is causing the activity in the virtual memory subsystem, as well as how much activity is being generated.

- a. True
- b. False

ORACLE

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

Answer: a

Summary

In this lesson, you should have learned how to describe and use the following:

- `sar`
- `vmstat`
- `iostat`
- `mpstat`
- `netstat`
- `nfsstat`
- `kstat`

ORACLE

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

Practice 3 Overview: `kstat` Monitoring Tools

This practice covers the following topics:

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

ORACLE

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

4

procs Monitoring Tools

ORACLE

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

Objectives

After completing this lesson, you will be able to describe and use:

- `ps`
- `prstat`
- `truss`
- `apptrace`
- `pmap`
- `pargs`
- `pwait` and `preap`
- `ptree`, `psig`, `pstop`, and `prun`

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- What tools are available to discover aberrant process behavior?
- What tools will help isolate the cause of process performance problems?
- What tools can be used to see the processes that are consuming memory or CPU resources?
- What tools allow you to examine individual threads or examine thread stacks?
- How do you see the processes that are created by other processes on the system?
- How do you find out what signals are being handled by a given process?
- How do you get rid of a zombie?

ORACLE

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

Agenda

- **Introduction to `procfs`-Based Tools**
- The `ps` Command
- The `prstat(1)` Utility
- The `truss(1)` Utility
- Other Significant Tools

ORACLE

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

procfs-Based Tools

- `ps(1)`
- `prstat(1M)`
- `truss(1)` and `apptrace(1)`
- The `proc` tools, including:
 - `pmap(1)`
 - `ptree(1)`
 - `psig(1)`
 - `pstack(1)` and `pargs(1)`
 - `pfiles(1)`
 - `pstop(1)` and `prun(1)`
 - `pwait(1)` and `preap(1)`

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

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

The `procfs` utilities report the state of processes and associated lightweight processes (LWPs) running on a system.

Agenda

- Introduction to `procfs`-Based Tools
- `ps` **Command**
- `prstat(1)` Utility
- `truss(1)` Utility
- Other Significant Tools

ORACLE

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

ps Command

- Lists a snapshot of all processes and optionally `lwp`s
- Enables customization of output with the `-o` option
- Examples:

```
# ps -le
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
1	T	0	0	0	0	0	SY	?	0		?	0:31	sched
0	S	0	1	0	0	40	20	?	623		? ?	0:00	init
1	S	0	2	0	0	0	SY	?	0		? ?	0:00	pageout
1	S	0	3	0	0	0	SY	?	0		? ?	0:24	fsflush
0	S	0	151	1	0	40	20	?	1270		? ?	0:01	nscd
0	S	0	7	1	0	40	20	?	3185		? ?	0:02	svc.star
...													

ORACLE

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

The `ps` command with the `-le` option shows a calculated priority under the `PRI` column; for `SYS` and `RT` processes, it shows a priority of 0. For all other scheduling classes, the priority is calculated by subtracting the actual priority from 99. This is to make it similar to the old priorities that were used in SunOS 4.X, when lower priorities were better (that is, executed first). To see the correct priority, include the `-c` option or customize the `ps` output with `-o`.

See `man ps` for more details.

ps Command

The `-f` option shows the complete command name:

```
# ps -clef
```

F	S	UID	PID	PPID	CLS	PRI	ADDR	SZ	WCHAN	STIME	TTY	TIME	CMD
1	T	root	0	0	SYS	96	?	0		08:23:33	?	0:31	sched
0	S	root	1	0	TS	59	?	623	?	08:23:39	?	0:00	/sbin/init
1	S	root	2	0	SYS	98	?	0	?	08:23:39	?	0:00	pageout
1	S	root	3	0	SYS	60	?	0	?	08:23:39	?	0:24	fsflush
0	S	root	151	1	TS	59	?	1270	?	08:23:58	?	0:01	/usr/sbin/
...													

ORACLE

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

ps Command

The `-L` option shows a line per LWP:

# ps -cLe						
PID	LWP	CLS	PRI	TTY	LTIME	CMD
0	1	SYS	96	?	0:31	sched
1	1	TS	59	?	0:00	init
2	1	SYS	98	?	0:00	pageout
3	1	SYS	60	?	0:25	fsflush
151	1	TS	59	?	0:00	nscd
151	2	TS	59	?	0:00	nscd
151	3	TS	59	?	0:00	nscd
151	4	TS	59	?	0:00	nscd
151	5	TS	59	?	0:00	nscd
151	6	TS	59	?	0:00	nscd
151	7	TS	59	?	0:00	nscd

ORACLE

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

ps Command

The `-o` option enables you to customize the output format:

```
usage: ps [ -aAdeflcljLPyZ ] [ -o format ] [ -t termlist ]
        [ -u userlist ] [ -U userlist ] [ -G grouplist ]
        [ -p proclist ] [ -g pgrplist ] [ -s sidlist ] [ -z zonelist ]
'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
```

```
# ps -o s,uid,pid,ppid,class,pri,vsz,rss,time,fname -e
S UID PID PPID CLS PRI  VSZ  RSS      TIME  COMMAND
T  0   0   0  SYS  96    0    0      00:43  sched
S  0   1   0  FSS  59 2904 1008      05:43  init
S  0   2   0  SYS  98    0    0      00:14  pageout
S  0   3   0  SYS  60    0    0      01:41:52  fsflush
S  0 394 393  FSS  59 2376  576      00:00  smcboot
S  0   7   1  FSS  29 13840 1592      00:14  svc.star
...
```

ORACLE

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

/usr/ucb/ps Command

```
# /usr/ucb/ps aux
```

USER	PID	%CPU	%MEM	SZ	RSS	TT	S	START	TIME	COMMAND
daemon	252	0.6	0.0	7880	1272	?	S	Dec 05	128:28	/usr/lib/rcap/rcapd
root	3	0.2	0.0	0	0	?	S	Dec 05	101:58	fsflush
root	29528	0.1	0.37028843136			?	S	10:56:04	3:12	/usr/lib/mixer_applet2
... <output omitted>										



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

/usr/ucb/ps Command

To see the complete command name, include the `-ww` option:

```
# /usr/ucb/ps auxww
USER  PID  %CPU %MEM    SZ   RSS TT S      START  TIME    COMMAND
daemon 252    0.6  0.0  7880 1272  ? S      Dec 05 128:40
/usr/lib/rcap/rcapd
root   631    0.4  0.1  4000 3264 pts/2    O 22:05:03 0:00 /usr/ucb/ps -
auxww
root    3    0.2  0.0    0    0      ?    S Dec 05 102:02 fsflush
root 29528    0.1  0.37028843136 ? S      10:56:04 3:14
/usr/lib/mixer_applet2 --oaf-activate-
iid=OAFIID:GNOME_MixerApplet_Factory --
oaf-ior-fd=43
<output omitted>
```

ORACLE

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

To see the complete command name, use `pargs (1)`. This command will break up the command into its separate arguments and it is much easier to read.

See `man pargs`.

Agenda

- Introduction to `procfs`-Based Tools
- `ps` Command
- `prstat(1)` **Utility**
- `truss(1)` Utility
- Other Significant Tools

ORACLE

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

prstat(1M) Utility

- Is a text-based process monitor
- Reports data similar to `Top`
- Runs continually, refreshing the screen on a regular basis
- Shows only one window of processes
- Can be sorted by different columns with the `-s` option
- Is useful for seeing load summations by:
 - Project
 - Zone
 - User, group, or session
 - pset (processor set)

ORACLE

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

prstat(1M) Utility Options

prstat Option	Description
-a	Prints separate concurrent usage reports by process and by user
-c	Appends new reports to existing output
-C <list>	Prints usage report for the listed processor sets
-L	Prints usage reports per LWP
-m	Prints microstate process accounting information; extends the verbose output provided by -v
-p <list>	Prints a report only on listed processes
-s/S <key>	Prints a report sorted in descending or ascending order by CPU usage (<code>cpu</code>), priority (<code>pri</code>), Resident Set Size (<code>rss</code>), process image size (<code>size</code>), or execution time (<code>time</code>)
-v	Prints verbose information

ORACLE

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

prstat(1M) Utility Options

prstat Option	Description
-Z	Prints a summary of zone usage information
-J	Prints process and project information
-R	Puts <code>prstat</code> in real time; can be used only by <code>superuser</code>

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

prstat Output: Example

```
root@server1:~# prstat
```

PID	USERNAME	SIZE	RSS	STATE	PRI	NICE	TIME	CPU	PROCESS/NLWP
252	daemon	7880K	1280K	sleep	1	0	2:09:45	0.1%	rcapd/1
29617	root	97M	55M	sleep	1	4	0:00:54	0.1%	gnome-terminal/2
29528	root	69M	42M	sleep	1	4	0:03:51	0.1%	mixer_applet2/1
834	root	4048K	3376K	cpu3	1	4	0:00:00	0.0%	prstat/1
29526	root	70M	43M	sleep	1	4	0:01:23	0.0%	gnome-netstatus/1
29512	root	77M	50M	sleep	1	4	0:00:53	0.0%	gnome-panel/1
29471	root	15M	12M	sleep	41	4	0:00:55	0.0%	gconfd-2/1
29508	root	65M	41M	sleep	1	4	0:00:07	0.0%	metacity/1
15599	noaccess	219M	120M	sleep	1	0	0:07:53	0.0%	java/18
22610	noaccess	223M	112M	sleep	1	0	0:05:46	0.0%	java/18
23396	noaccess	219M	124M	sleep	1	0	0:09:22	0.0%	java/18
3308	noaccess	187M	88M	sleep	1	0	0:10:18	0.0%	java/18
2182	noaccess	215M	116M	sleep	55	0	0:35:51	0.0%	java/18
1008	noaccess	211M	95M	sleep	56	0	0:09:43	0.0%	java/18
18701	root	94M	22M	sleep	59	0	0:04:04	0.0%	poold/8
Total: 253 processes, 956 lwps, load averages: 0.04, 0.04, 0.03									



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

prstat Output: Example

```
root@server1:~# 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@server1:~# 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
...
```



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

Examining Processes

Utility	Description
pflags	Prints the <code>/proc</code> tracing flags, pending and held signals, and other <code>/proc</code> status information for each LWP of a process
pcred	Prints the credentials of a process
pmap	Prints the address space mapping of a process
pldd	Prints the shared-object libraries (<code>.so</code> files) that each process and process number uses
psig	Prints the signal actions of a process
pstack	Prints a stack trace for each LWP in a process
pfiles	Prints information from the <code>fstat(2)</code> and <code>fcntl(2)</code> methods for all the open files in a process
pwdx	Prints the current working directory of a process

ORACLE

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

The `proc` tools take the `pid` of a process as an argument. Some of the `proc` tools have extra arguments that may display more information—for example:

- `pflags` – Has an `-r` option to display machine registers
- `pcred` – Allows `root` to modify a given `pid`'s real or effective uid or gid with the `-l`, `-u`, `-G`, and `-g` options
- `pmap` – Has an `-x` option that displays additional information per mapping, such as:
 - Size of each mapping
 - Amount of resident physical memory (RSS)
 - Amount of anonymous memory
 - Amount of memory locked

`pflags`, `pcred`, `pldd`, `pstack`, and `pmap` can be used to examine information in a core file, as well as a live process.

Examining Processes

Utility	Description
pstop	Stops the specified process
prun	Sets the specified process to running
pwait	Waits for the specified process to terminate
ptree	Prints an indented list showing the relationship of a process with respect to its child and parent processes
ptime	Prints the real, user, and system time taken to execute a command
preap	Forces a defunct process to be reaped
pargs	Prints command-line arguments and argument vectors of a process

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

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

The `preap(1)` command is how you can make a zombie process go away.

Agenda

- Introduction to `procfs`-Based Tools
- `ps` Command
- `prstat(1)` Utility
- `truss(1)` **Utility**
- Other Significant Tools

ORACLE

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

truss(1) Utility

- The `truss` command enables you to trace:
 - System calls
 - Signals
 - Faults
- You trace user code with the `-u` option
 - `a.out` file
 - Libraries
- Run `truss -p PID#` to truss an existing process.
- Run `truss -c` to display a count of system calls, signals, and faults.
- The `apptrace(1)` command is used to trace calls to Oracle Solaris shared libraries.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

truss(1) Utility

```
# truss test_gm
execve("test_gm", 0xFFBFFACC, 0xFFBFFAD4) argc = 1
resolvepath("/usr/lib/ld.so.1", "/lib/ld.so.1", 1023) = 12
...<output omitted>
setustack(0xFF392A88)
mmap(0x00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1,
0) = 0xFF370000
mmap(0x00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1,
0) = 0xFF350000
sigaction(SIGCANCEL, 0xFFBFF6E0, 0x00000000) = 0
sysconfig(_CONFIG_STACK_PROT) = 7
sysconfig(_CONFIG_PAGESIZE) = 8192
mmap(0x00000000, 1032192, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_NORESERVE|MAP_ANON, -1, 0) = 0xFF100000
mmap(0x00010000, 65536, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_ANON|MAP_ALIGN, -1, 0) = 0xFF0E0000
mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC,
...<output omitted>
```

ORACLE

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

truss(1) Utility

```
# truss -u a.out test_gm
execve("test_gm", 0xFFBFFACC, 0xFFBFFAD4) argc = 1
resolvepath("/usr/lib/ld.so.1", "/lib/ld.so.1", 1023) = 12
...<output omitted>
/1@1: -> main(0x1, 0xffbfffacc, 0xffbfffad4, 0x2169c)
/1: mmap(0x00000000, 65536, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANON, -1, 0) = 0xFF370000
...<output omitted>

/1:   lwp_create(0xFFBFF7C8, LWP_SUSPENDED, 0xFFBFF7C4) = 2
/2:   lwp_create() (returning as new lwp ...)           = 0
/1:   schedctl()                                       = 0xFF33E000
/1:   lwp_continue(2)                                 = 0
/2:   setustack(0xFF0E0288)
/2:   schedctl()                                       = 0xFF33E010
/2@2: -> thread_function(0x112e8, 0xff1fc000, 0x0, 0x0) ...
```



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

The `-u` argument to `truss` is used to trace user code with the keyword `a.out`. It traces the main program rather than a library. Notice that this is a multithreaded program. `/1:` indicates calls made by the first thread and `/2:` indicates calls made by the second thread.

truss(1) Utility

```
# truss -p 1
pollsys(0x0002B9DC, 1, 0xFFBFF588, 0x00000000) (sleeping...)
Received signal #22, SIGPOLL, in pollsys() [caught]
siginfo: SIGPOLL POLL_IN fd=1536 band=0
pollsys(0x0002B9DC, 1, 0xFFBFF588, 0x00000000)      Err#4 EINTR
lwp_sigmask(SIG_SETMASK, 0x00220000, 0x00000000)    = 0xFFBFFFEFF
[0x0000FFFF]
read(0, "\0\0\001\0\0 !CB", 8)                      = 8
read(0, 0xFFBFF048, 8)                              = 0
setcontext(0xFFBFEEF0)
stat("/etc/inittab", 0xFFBFF298)                     = 0
open("/etc/inittab", O_RDONLY)                       = 1
ioctl(1, TCGETA, 0xFFBFF054)                        Err#25 ENOTTY
fstat64(1, 0xFFBFF0C8)                              = 0
fstat64(1, 0xFFBF70)                                = 0
...<output omitted>
```

ORACLE

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

The `-p` argument to `truss` allows it to trace a process already in execution.

truss(1) Utility

```
# truss -c more /usr/dict/words
```

```
10th
```

```
1st
```

```
...<output omitted>
```

syscall	seconds	calls	errors
_exit	.000	1	
read .	.000	3	
write	.000	40	
open	.000	6	1
close	.000	5	
brk	.000	4	
getpid	.000	1	
ioctl	.000	13	1
execve	.000	1	
sigaction	.000	5	
getcontext	.000	1	

ORACLE

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

With the `-c` option, `truss` can only count the number of system calls rather than display information about each system call.

truss(1) Utility

setustack	.000	1	
mmap	.000	17	
munmap	.000	5	
getrlimit	.000	1	
memcntl	.000	3	
schedctl	.000	1	
resolvepath	.000	6	
stat64	.000	6	
fstat64	.000	3	
open64	.000	1	

sys totals:	.002	124	2
usr time:	.002		
elapsed:	2.000		

ORACLE

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

The `truss -c` command also points out the number of times a process must enter the kernel and return to user mode (due to system calls), for even a simple command such as `more` on a file.

Agenda

- Introduction to `procfs`-Based Tools
- `ps` Command
- `prstat(1)` Utility
- `truss(1)` Utility
- **Other Significant Tools**

ORACLE

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

Solaris Studio Performance Analyzer

- Full-featured toolkit for instrumenting and sampling process activity
- Built in two parts:
 - Performance statistics collection
 - `collect(1)` utility: Gathers statistics
 - Performance statistics presentation
 - `er_print(1)` utility (terminal-based)
 - `analyzer(1)` utility (GUI-based)
- Statistics that may be gathered depend on:
 - The `collect(1)` options invoked
 - The compiler options to build an application

ORACLE

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

The Solaris Studio software packages are installed in `/opt` by default. In the course lab environment, these binaries are included in `PATH`.

The `collect(1)` utility can be used to start an application or to attach an application to a running process. To see some sample data, perform the following:

```
# collect -P `pgrep nsd`
<output of segments being read>
```

In another terminal window, list the contents of `/var`:

```
# ls -R /var
<output of directories and file content>
```

When the output completes, interrupt the collect process in the first window. Look for a directory named `test.1.er`. This directory contains the data output from the run. Use `er_print` to read this data:

```
# er_print test.1.er
```

Alternatively, you can invoke the `analyzer(1)` utility and open the directory after the GUI appears.

Quiz

The `procfs` utilities report the state of processes and associated lightweight processes (LWPs) running on a system.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

Which option of the `ps` command shows the longer command names in the output?

- a. `c`
- b. `l`
- c. `e`
- d. `f`

ORACLE

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

Answer: d

Quiz

The `collect(1)` utility from Solaris Studio:

- a. Prints process activity–related statistics
- b. Gathers process activity–related statistics
- c. Analyses process-related statistics
- d. None of the above

ORACLE

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

Answer: b

Summary

In this lesson, you should have learned how to describe and use the following:

- `ps`
- `prstat`
- `truss`
- `apptrace`
- `pmap`
- `pargs`
- `pwait` and `preap`
- `ptree`, `psig`, `pstop`, and `prun`

ORACLE

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

Practice 4 Overview: **Using `ps`, `prstat`, the `proc` Tools, and `truss`**

This practice covers using the `ps` and `prstat` commands.

ORACLE

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

5

Introduction to DTrace

ORACLE

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

Objectives

After completing this lesson, you will be able to:

- List the advantages of DTrace
- Describe the DTrace architecture
- List the DTrace providers
- Describe DTrace one-liners
- Download and use scripts from the DTrace toolkit

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- How were probes put in the kernel before SunOS 5.11?
- What were the disadvantages of using a debugger on kernel or user code to observe run-time events?

ORACLE

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

Agenda

- Introduction to DTrace
- DTrace Providers and One-Liners
- DTrace Toolkit
- DTrace Visualization Tool

ORACLE

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

DTrace

- DTrace (Dynamic Tracing Utility) traces execution in user and kernel modes.
- Probes cause no overhead until they are enabled, unlike tools such as `truss`.
- DTrace is safe to use and will not induce system failure.
- It includes the `dtrace(1M)` command and the “D” language interface.
- Documentation is available on the Oracle Technology Network documentation website.

The Oracle logo, consisting of the word "ORACLE" in white, uppercase letters on a red rectangular background.

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

DTrace is the Solaris Dynamic Tracing Framework. It allows users to trace execution in user and kernel modes by dynamically modifying the system to record arbitrary data at specified points of execution called probes. DTrace allows probes to be dynamically added to kernel or user code without rewriting or restarting applications.

Properly enabled probes represent much lower overhead than traditional tracing tools such as `truss`. DTrace is safe to use in production environments.

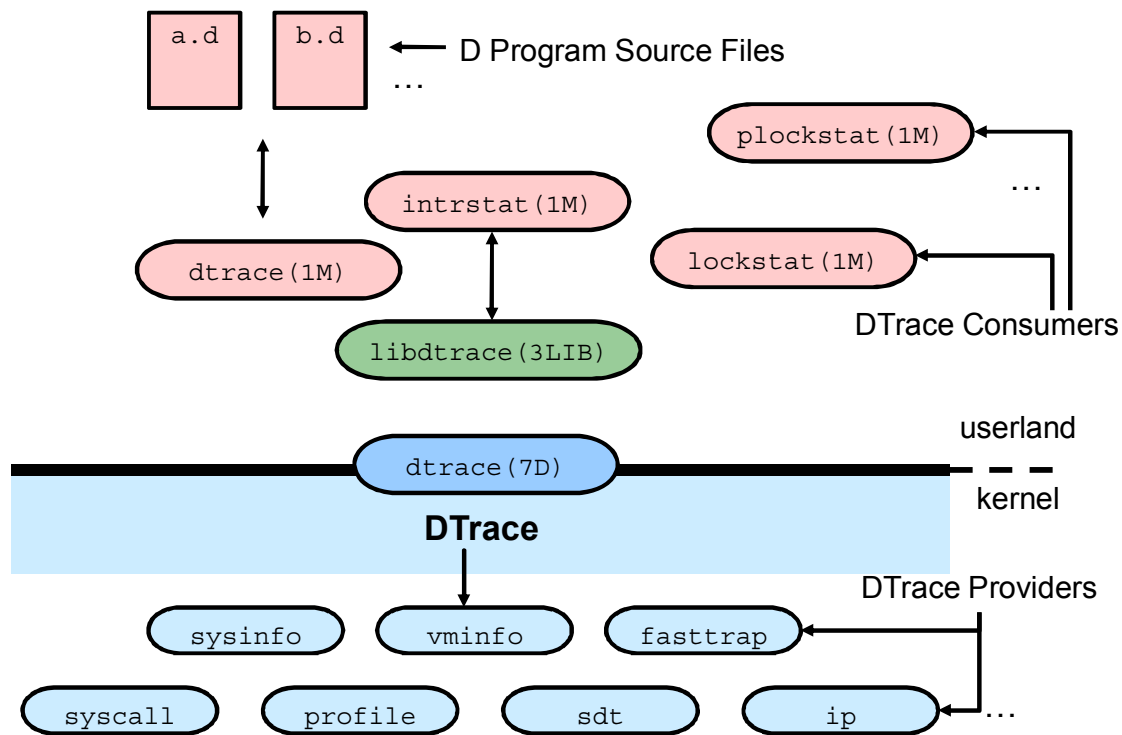
By default, DTrace can be used only by `root`. However, DTrace privileges can be given to users through the “Least Privilege facility” in Oracle Solaris 11.

There is a `dtrace(1M)` command for listing and setting simple probes from the command line. There is also a programming interface called the “D” programming language that provides an interpreted C-like interface for writing D scripts or programs.

For more information about the DTrace framework and programming interface, visit the following websites:

- <https://wikis.oracle.com/display/DTrace/Documentation>
- http://docs.oracle.com/cd/E26502_01/html/E28556/index.html

DTrace Architecture



ORACLE

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

Consumers such as the `dtrace(1M)` or `lockstat(1M)` command interface with the kernel DTrace framework through a private library interface. Consumers can also be programs that are written in a new programming language referred to as “D.” They define instrumentation points called probes. A set of providers within the kernel implement and enable user-defined probes.

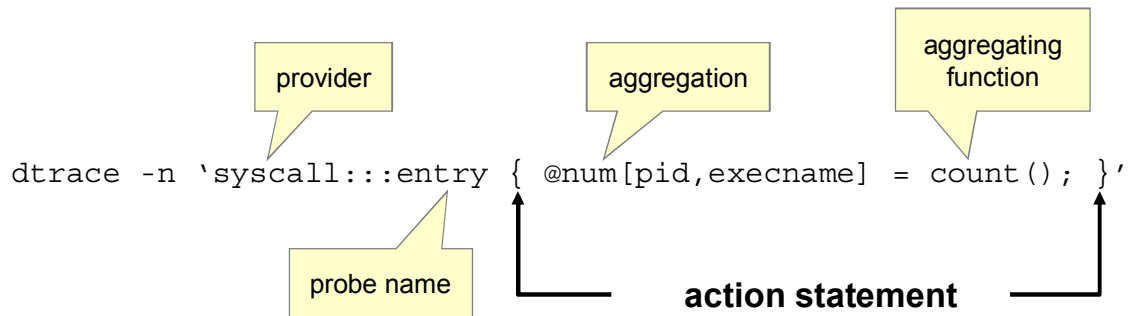
Agenda

- Introduction to DTrace
- DTrace Providers and One-Liners
- DTrace Toolkit
- DTrace Visualization Tool

ORACLE

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

DTrace Command Syntax



ORACLE

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

DTrace Providers

A number of providers are built into the kernel that supply information for DTrace, including:

- `syscall`
- `fbt`
- `profile`
- `sched`
- `lockstat`
- `pid`
- `proc`
- `vminfo`
- `sysinfo`
- `IO`

ORACLE

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

- `syscall`: Provides probes for entry and return from system calls
- `fbt`: Is the function boundary trace for tracing entry and return for kernel functions
- `profile`: Is used for time-based interrupt firing at fixed intervals
- `sched`: Is used for tracing CPU scheduling events, such as context switching
- `lockstat`: Is used for tracing access to synchronization objects
- `pid`: Allows tracing of entry and return of any function in a user process, and any instruction specified by an absolute address or function offset
- `proc`: Probes for process and LWP creation, termination, execution, and signal handling and delivery
- `vminfo`: Provides probes for the vm-named kernel statistics corresponding to `kstat -n vm`
- `sysinfo`: Provides probes for the sys-named kernel statistics corresponding to `kstat -n sys`
- `IO`: Provides probes related to disk I/O and NFS

DTrace Providers

- `ip`
- `tcp`
- `dtrace`
- `sdt`
- `mib`
- `plockstat`
- `fsinfo`
- `fpuinfo`
- Language providers for Java, JavaScript, Perl, Ruby, Python, and PHP

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

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

- `ip`: Is used for tracing IP packets
- `tcp`: Is used for analyzing TCP connections
- `dtrace`: Is used for DTrace itself, including `BEGIN`, `END`, and `ERROR`
- `sdt` (Static Defined Tracing): Allows programmers to choose locations of interest to DTrace users
- `mib`: Provides probes corresponding to counters in the Solaris management information bases (MIBs) used by the Simple Network Management Protocol (SNMP)
- `plockstat`: Is used for tracing access to user-level synchronization objects
- `fsinfo`: Is used for tracing generic file system operations (`arg0` points to a `fileinfo_t` struct)
- `fpuinfo`: Is used for tracing kernel simulation of hardware floating-point instructions on SPARC architectures
- Language providers for Java, JavaScript, Perl, Ruby, Python, and PHP. See the DTrace toolkit for sample scripts that use language-specific providers.

DTrace One-Liners

- Top user functions running on CPU (% usr time):

```
dtrace -n 'profile-997hz /arg1/ { @[execname, ufunc(arg1)] = count(); }'
```

- Processes executing the most system calls:

```
dtrace -n 'syscall:::entry { @[pid, execname] = count(); }'
```

- Tracking memory page faults by process name:

```
dtrace -n 'vminfo:::as_fault { @mem[execname] = sum(arg0); }'
```

- Showing disk I/O size as distribution plots by process name:

```
dtrace -n 'io:::start { @size[execname] = quantize(args[0]->b_bcount); }'
```

- Received IP packets by host address:

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

- Network connections:

```
dtrace -n 'tcp:::accept-established { @[args[3]->tcps_raddr, args[3]->tcps_lport] = count(); }'
```

ORACLE

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

DTrace scripts can be invoked directly from the command line, providing one or more probes and actions as arguments. These are typically referred to as “one-liners.” Some helpful one-liners for viewing system information are as follows. Note that the syntax for a one-liner is `ptrace -n probe action`.

The `-n` option allows you to specify the probe name to trace in the form of:

provider:module:function:name, module:function:name, function:name, or name.

Note: For additional information about DTrace one-liners, see

DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD by Brendan Gregg and Jim Mauro at www.dtracebook.com.

Running a DTrace One-Liner

```
# dtrace -n 'syscall:::entry { @[pid, execname] = count(); }'
dtrace: description 'syscall:::entry ' matched 214 probes
^C
11  svc.startd                1
42  netcfgd                   1
50  dlmgmt                    1
796 automountd               1
887 in.routed                 1
13  svc.configd               2
192 utmpd                     2
674 nscd                      2
840 fmd                       5
5312 sshd                     8
713 hald                      10
719 hald-addon-acpi           13
244 devfsadm                  14
1209 dhcpagent                15
104  in.mpathd                 21
1294 sendmail                  30
5324 dtrace                    4229
```

ORACLE

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

The slide shows an example of running a DTrace one-liner that displays the processes executing the most system calls.

Agenda

- Introduction to DTrace
- DTrace Providers and One-Liners
- **DTrace Toolkit**
- DTrace Visualization Tool

ORACLE

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

DTrace Toolkit

DTrace Toolkit:

- Is well-organized
- Is documented and has a set of `man` pages
- Checks for stability
- Requires root to use it
- Starts from `/opt/DTT/README`
- Contains good examples with comments

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a solid red rectangular background.

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

The DTrace toolkit can be downloaded from:

<http://opensolaris.org/os/community/dtrace/dtracetoolkit/>

The DTrace toolkit contains some useful scripts that are well-organized and documented. No installation is required, but it is a good idea to run the installation script, which puts the files in known directories.

You may then want to add `/opt/DTT/Man` to your `$MANPATH` variable. You may also want to add `/opt/DTT/Bin` to your `$PATH`. A good starting point is to follow the instructions in the `/opt/DTT/README` file.

DTrace Toolkit

Some example scripts include:

- `dtruss`: Prints the `syscall` information for a given process
- `dexplorer`: Runs a set of scripts and archives the result
- `iosnoop`: Is a comprehensive utility to snoop disk I/O
- `iotop`: Lists the top processes consuming disk I/O
- `errinfo`: Reports system call failures with details (such as `errno`)
- `dvmstat`: Uses the `vminfo` provider to display `vmstats` per process
- `fddist`: Prints distributions for read and write events by process

ORACLE

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

Because these commands use DTrace, you must be `root` or have DTrace privileges to use them.

Note: One thing to be wary of is scripts that are using the function boundary trace (`fbt`) provider, because they refer to kernel routines that may change name or functionality at any update. On the man page for a given script, look in the section labeled *STABILITY*. Those scripts that use the `fbt` provider or that refer to kernel variables tend to be unstable. Any field in a kernel data structure can change name, meaning, or even cease to exist at any update. Any kernel data that is referenced through a translated data structure, such as the `psinfo` or `lwpsinfo` structure, is considered to be stable. Also, any data that is referenced through a DTrace built-in variable, such as `curthread` or `pid`, is also stable. For more information, see the chapters titled “Stability” and “Translators” in the *Solaris Dynamic Tracing Guide*. You can check the stability of a script or command by including the `-v` option to the `dtrace` command.

Agenda

- Introduction to Dtrace
- DTrace Providers and One-Liners
- DTrace Toolkit
- **DTrace Visualization Tool**

ORACLE

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

DTrace Visualization Tool (DLight)

- Is a graphical tool for visualizing DTrace aggregations
- Is part of Oracle Solaris Studio
- Makes graphical visualization of data easier
- Enables you to explore:
 - Thread microstates
 - CPU usage
 - Memory usage
 - Thread usage
 - I/O usage

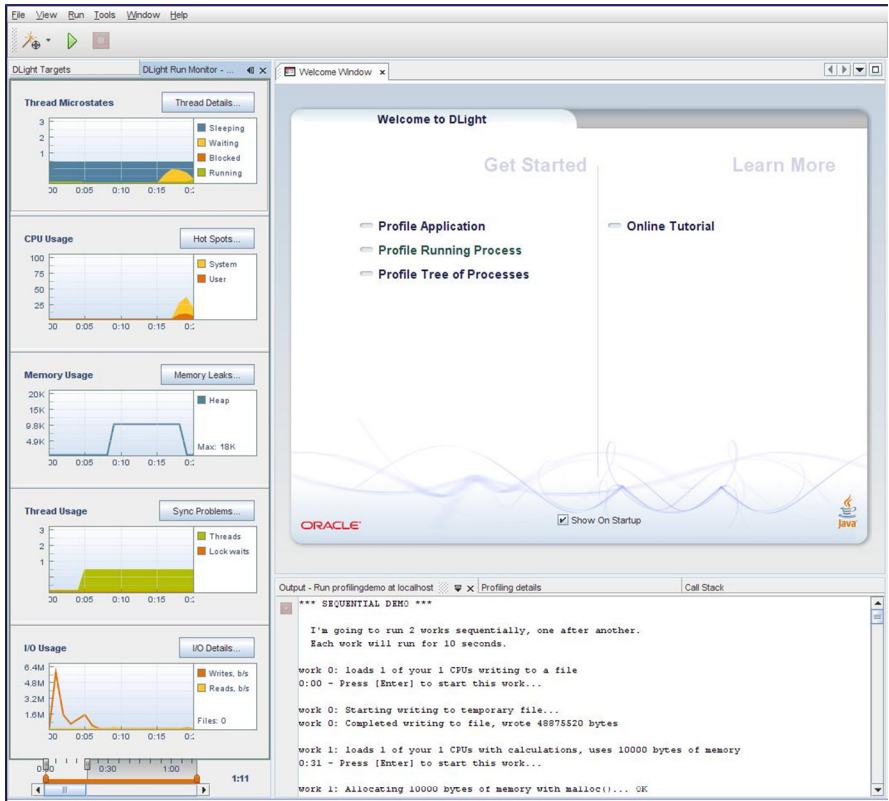
The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

Oracle Solaris Studio DLight is a visual profiling tool that unifies application and system profiling, using DTrace technology on Oracle Solaris platforms.

Refer to the *DLight Tutorial* on the Oracle Technology Network documentation website.

DLight GUI



ORACLE

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

Quiz

DTrace allows users to trace execution in user and kernel modes by dynamically modifying the system to record arbitrary data at specified points of execution called probes.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

The DTrace toolkit contains some useful scripts that are well-organized and documented. You must run the installation script, which puts the files in known directories.

- a. True
- b. False

ORACLE

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

Answer: b

Summary

In this lesson, you should have learned how to:

- List the advantages of DTrace
- Describe the DTrace architecture
- List DTrace providers
- Describe DTrace one-liners
- Download and use scripts from the DTrace toolkit

ORACLE

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

Practice 5 Overview: Using DTrace

This practice covers the following topics:

- Practicing with DTrace one-liners
- Using the DTrace toolkit

ORACLE

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

Other Significant Tools



ORACLE

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

Objectives

After completing this lesson, you will be able to:

- Describe and interpret the `swap` command
- Describe and use `cpustat`
- Describe and use `mdb -k` to examine kernel structures
- Describe the `zonestat` utility
- Describe the GUDS script

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- How do you measure the cache hit rate?
- Which caches can you measure?
- How can you tell if memory accesses are stalling the CPU?
- How do you look at the current value of tunable parameters such as `max_nprocs` and `lotsfree`?
- How do you examine the information that the kernel keeps track of for a given process such as the contents of the `proc` and `user` structure?

ORACLE

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

Agenda

- swap
- CPU Performance Counters
- mdb (1) Utility
- Solaris Studio dbx (1) Utility
- zonestat Utility
- GUDS Script

ORACLE

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

swap(1M)

- The `swap -s` command reports a summary of the swap information taken from the `k_anoninfo` structure.

```
# swap -s
total: 1115600k bytes allocated + 189224k reserved =
1304824k used, 25195584k available
```

- The 1115600 KB allocated is for anonymous pages that have been accessed in memory.
- The 189224 KB reserved is for anonymous pages that have been reserved, but not yet accessed in memory.
- In this example, the 1304824 KB used refers to the amount of virtual swap space that has been reserved.
- The 25195584 KB available is the amount of unreserved swap space. This is what `vmstat` reports in the `swap` column.

ORACLE

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

The `swap` command has four options: two for adding and deleting swap areas and two for reporting swap information. The hardest to interpret is `swap -s`. It is useful to know that swap space is reserved when a segment, which is backed by `swap`, is created. The segments that are backed up by `swap` include stack, heap, `/dev/zero` mappings, `tmpfs` files, shared memory segments, and data, because it may be modified.

Actual disk space is not assigned to a page of swap until it is paged out, by either the page daemon or the swapper.

swap(1M)

- `swap -l` lists information about all the swap areas.

```
# swap -l
swapfile                dev      swaplo  blocks      free
/dev/dsk/c1t0d0s1        32,1      16  1048688  1048688
/swapfile                -          16  25165808  25165808
/dev/zvol/dsk/conpool/swapvol 256,1     16 16777200 16777200
```

- `swap -a` adds a specified swap area:
 - UFS disk slice—for example, `/dev/dsk/c1t0d0s1`
 - UFS regular file—for example, `/swapfile`
 - ZFS volume—for example, `/dev/zvol/dsk/conpool/swapvol`
- `swap -d` deletes a specified swap area.

ORACLE

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

Note

See `man -s 1M swap`.

Agenda

- swap
- **CPU Performance Counters**
- mdb (1) Utility
- Solaris Studio dbx (1) Utility
- zonestat Utility
- GUDS Script

ORACLE

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

CPU Performance Counters

- Three utilities are available to track CPU-based performance counters:
 - `cpustat`: Tracks counters on a per-CPU basis
 - `cpustrack`: Reports CPU counters on a per-process basis
 - `trapstat`: Displays run-time trap statistics on SPARC-based systems
- The counters are referred to as Performance Instrumentation Counters (PICs).
- The default sampling interval is five seconds.
- The number of PICs and the events that are available are CPU dependent.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

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

Three utilities are available to track CPU-based performance counters. The `cpustat` utility tracks counters on a per-CPU basis. The `cpustrack` utility reports CPU counters on a per-process basis. `cpustrack` is discussed in more detail in the lesson titled “System Caches and Buses.”

The `cpustat` utility reports event counters for each CPU in the system.

The PIC instances `pic0` and `pic1` serve as column headers. They can be omitted if desired—for example:

```
# cpustat -c IC_ref,IC_miss 1 5
```

The events counted in each `pic` field are totaled on the last output line.

cpustat(1M) Utility

Use `-h` to see the events, for example:

```
# cpustat -h
```

Usage:

```
cpustat [-c events] [-p period] [-nstD] [interval [count]]
```

```
-c events specify processor events to be monitored
```

```
-n suppress titles
```

```
-p period cycle through event list periodically
```

```
-s run user soaker thread for system-only events
```

```
-t include %tick register
```

```
-D enable debug mode
```

```
-h print extended usage information
```

```
Use cputrack(1) to monitor per-process statistics.
```

```
CPU performance counter interface: UltraSPARC IIIi & IIIi+
```

ORACLE

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

Note

See `man -s 1M cpustat`.

cpustat(1M) Utility

```
event specification syntax:
[picn=]<eventn>[,attr[n] [=<val>]] [, [picn=]<eventn>[,attr[n] [=<val>]]
,...]
event0: Cycle_cnt Instr_cnt Dispatch0_IC_miss IC_ref DC_rd DC_wr
        EC_ref EC_snoop_inv Dispatch0_br_target Dispatch0_2nd_br
        Rstall_storeQ Rstall_IU_use EC_write_hit_RTO EC_rd_miss
        ...<output omitted>
event1: Cycle_cnt Instr_cnt Dispatch0_mispred EC_wb EC_snoop_cb
        IC_miss_cancelled Re_FPU_bypass Re_DC_miss Re_EC_miss
        IC_miss DC_rd_miss DC_wr_miss Rstall_FP_use EC_misses
        EC_ic_miss Re_PC_miss ITLB_miss DTLB_miss WC_miss
        ...<output omitted>
```

- See the *UltraSPARC III User's Manual* for descriptions of these events.
- Documentation for Sun processors can be found on the Oracle Technology Network.

ORACLE

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

The `cpustat` utility must be run by the `superuser`, because there is an intrinsic conflict between the use of the CPU performance counters system-wide by `cpustat` and the use of the CPU performance counters to monitor an individual process (for example, by `cputrack`). After any instance of this utility has started, no further per-process or per-LWP use of the counters is allowed until the last instance of the utility terminates.

cpustat(1M) Utility

```
# cpustat -c pic0=IC_ref,pic1=IC_miss
```

time	cpu	event	pic0	pic1
5.001	0	tick	1985778	153565
5.002	1	tick	65312	9993
5.002	2	tick	1813445	92483
5.002	3	tick	8741192	767272
10.001	0	tick	1481775	113706
10.002	1	tick	56911	10026
10.002	2	tick	19734	3643
10.002	3	tick	11500168	948813



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

The program instruction counter instances `pic0` and `pic1` serve as column headers. They can be omitted if desired.

cpustat(1M) Utility

```
# cpustat -nc IC_ref,IC_miss|awk \
'{printf("%3.0f\t%d\t%3.2f%\n", $1, $2, ($5/$4)*100)}'
```

5	0	9.63%
5	1	3.20%
5	2	15.11%
5	3	8.93%
10	0	0.04%
10	1	0.30%
10	2	0.02%
10	3	0.05%
15	0	0.02%
15	1	0.03%
15	2	0.03%
15	3	0.05%



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

The `-n` option omits all header output, which is useful if `cpustat` is the beginning of a pipeline, as in this example. The output is piped into `awk` to print out the cache miss rate as a percentage.

cpustat(1M) Utility

```
# cpustat -c Cycle_cnt,Instr_cnt -c IC_ref,IC_miss 3
time  cpu event      pic0      pic1
3.001  0  tick    4383686  1379499 # Cycle_cnt,Instr_cnt
3.002  1  tick    558537   87713  # Cycle_cnt,Instr_cnt
3.003  2  tick    675672   165064 # Cycle_cnt,Instr_cnt
3.003  3  tick   44223344  16587087 # Cycle_cnt,Instr_cnt
6.001  0  tick    907829   72652  # IC_ref,IC_miss
6.002  1  tick    110963   14163  # IC_ref,IC_miss
6.003  2  tick     17415    3037  # IC_ref,IC_miss
6.003  3  tick   6277770   446720 # IC_ref,IC_miss
9.001  0  tick   4728626  1370233 # Cycle_cnt,Instr_cnt
9.002  1  tick    915147   268238 # Cycle_cnt,Instr_cnt
9.003  2  tick     71262    17700 # Cycle_cnt,Instr_cnt
9.003  3  tick  33177115  11750487 # Cycle_cnt,Instr_cnt
12.001 0  tick   1520402   142411 # IC_ref,IC_miss
12.002 1  tick    681937    74934 # IC_ref,IC_miss
12.003 2  tick    13382     3186 # IC_ref,IC_miss
12.003 3  tick   8768686   676440 # IC_ref,IC_miss
```

ORACLE

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

Note in the preceding examples that the two performance instruction counters, `pic0` and `pic1`, can be used only one at a time. Each slot can report only one statistic that is listed as available for that slot.

However, multiple `-c` options can be specified, in which case the command cycles between the different event settings on each sample, as shown in this example.

trapstat(1M) Utility

```
# trapstat
vct name          |  cpu0  cpu1  cpu2  cpu3  cpu4  cpu5
-----+-----
   9 immu-miss    |    0    0    0    0    1    0
  24 cleanwin     |    0    0    0    0   17    0
  31 dmmu-miss    |    0    0    0    0    4    0
  41 level-1      |    0    0    0    0    1    0
  46 level-6      |    0    0    2    0    0    0
  4a level-10     |   100    0    3    2    4   16
  4d level-13     |    1    1    2    3    4   15
  4e level-14     |   100    1    4    3    6   17
  6c dtlb-prot    |    0    0    0    0    1    0
  7c cpu_mondo    |    1    1    2    3    4   15
  7d dev_mondo    |    0    0    2    0    0    0
 84 spill-user-32 |    0    0    0    0    4    0
 8c spill-user-32-cln |    0    0    0    0    2    0
 98 spill-kern-64 |   291    2   31   25   73  192
 a4 spill-asuser-32 |    0    0    0    0    4    0
 ac spill-asuser-32-cln |    0    0    0    0    1    0
 c4 fill-user-32  |    0    0    0    0    4    0
 cc fill-user-32-cln |    0    0    0    0    1    0
 d8 fill-kern-64  |    90    1   26   22   67  175
108 syscall-32   |    0    0    0    0    4    0
124 getts        |    0    0    0    0    0    0
127 gethrtime    |    0    0    0    0    0    0
```



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

Agenda

- swap
- CPU Performance Counters
- **mdb (1) Utility**
- Solaris Studio dbx (1) Utility
- zonestat Utility
- GUDS Script

ORACLE

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

mdb(1) Utility

- Replacement for the adb debugger utility in SunOS 5.8
- To start the mdb debug on the kernel, run:
 - `mdb -k [w]`
- Useful `mdb` commands:
 - `::dcmds:` Lists all debugger commands
 - `::walkers:` List all walkers (used for walking link lists or AVL trees)
 - `::dmods -l:` Lists all debugger commands and walkers grouped by the debugger module
 - `::help dcmd:` Displays help for a debugger command
 - `::quit` or `$q:` Exits the debugger

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a solid red rectangular background.

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

Only `root` can run `mdb` on the kernel, that is, with the `-k` option.

When `mdb` initially starts, it shows the debugger modules that are loaded.

mdb(1) Utility

- The format for a typical debugger command is:

```
[address]::dcmd [arg(s)]
```

- For example:

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

- To display a kernel structure, use the `print dcmd`:

```
[address of struct]::print typedef  
> ffffffff80ebfce8::print proc_t
```

- If the kernel structure has no typedef, use `struct`:

```
> 0xfffffffff811d9140::print struct as
```

ORACLE

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

mdb(1) Utility

To print out a data structure with the `c` data types defined for members, use `::print -t typedef:`

```
> 0000060029a63908::print -t proc_t
{
    struct vnode *p_exec = 0x6002ac1a300
    struct as *p_as = 0x6003dc0fd10
    struct plock *p_lockp = 0x6002027c3c0
    kmutex_t p_crlock = {
        void *[1] _opaque = [ 0 ]
    }
    struct cred *p_cred = 0x6004ed89cf0
    int p_swapcnt = 0
    char p_stat = '\002'
    char p_wcode = '\0'
    ushort_t p_pidflag = 0
    int p_wdata = 0
    pid_t p_ppid = 0xec8
    struct proc *p_link = 0 ...
```

ORACLE

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

mdb(1) Utility

- To get one field from the `proc` structure, such as `p_as`, enter:

```
> 0000060029a63908::print proc_t p_as
```

- To pipe the output from one `mdb` command into another, use:

```
> 0000060029a63908::print proc_t p_as | ::print struct as
```

- Use `!` to execute a shell command or to pipe the output of an `mdb` command to a shell command:

```
> ::ps !grep syslogd  
> ::dcmds !grep quit
```

ORACLE

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

mdb(1) Utility

Provides Emacs-style command-line editing such as:

- `^P` or up-arrow: Fetch the previous command line.
- `^N` or down-arrow: Fetch the next command line.
- `^R[string]`: Search backward in history for a command that contains the string.
- `^B` or left-arrow: Move the cursor to the left.
- `^F` or right-arrow: Move the cursor to the right.
- `^K`: Delete from the cursor to the end of the line.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered within a solid red rectangular bar.

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

mdb(1) Utility

The `nm` dcmd is used to print symbols. The usage is:

```
[address]::nm [-DPdghnopuvx] [-f format] [-t types] [object]
```

For example, to list all kernel symbols:

```
> ::nm
Value                Size                Type  Bind  Other Shndx Name
0x0000000000000000 | 0x0000000000000000 | NOTY  | LOCL | 0x0   | UNDEF |
0x0000000001839050 | 0x0000000000000008 | OBJT  | LOCL | 0x2   | 13    | ops
0x00000000018394e0 | 0x0000000000000008 | OBJT  | LOCL | 0x2   | 13    | romp
...
```

ORACLE

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

Value represents the symbol's virtual address; Size is its size in bytes; Name is the symbol name. See `man -s 1 nm` for the other columns.

mdb(1) Utility

To list a given symbol:

```
> lotsfree::nm
Value                Size                Type  Bind  Other Shndx Name
0x00000000001874128|0x00000000000000008|OBJT  |GLOB  |0x0   |4   |lotsfree
```

To see its type in C:

```
> lotsfree::nm -f ctype
C Type
pgcnt_t
```

To look for a symbol:

```
> ::nm ! grep nswap
0x00000000001846528|0x00000000000000004|OBJT  |GLOB  |0x0   |13  |nswapped
0x000000000018ab720|0x00000000000000004|OBJT  |LOCL  |0x0   |6   |nswapfiles
```

ORACLE

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

mdb(1) Utility

To display data in mdb, use:

```
(address expression), (repeat count)/format
```

- Commonly used formats are:

```
x, X, J, d, D, E, w, W, Z, s, S, i, a
```

- Use the `formats dcmd` to get the complete list:

```
> ::formats
+ - increment dot by the count (variable size)
- - decrement dot by the count (variable size)
B - hexadecimal int (1 byte)
C - character using C character notation (1 byte)
D - decimal signed int (4 bytes)
E - decimal unsigned long long (8 bytes)
F - double (8 bytes)
G - octal unsigned long long (8 bytes)
...
```

ORACLE

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

mdb(1) Utility

From the output of the `nm` command, `lotsfree` was 8 bytes. So to display its value, enter:

```
> lotsfree/E
lotsfree:
lotsfree:      30100
> lotsfree::nm -f ctype
C Type
pgcnt_t
> !pagesize
8192
```

To multiply the page count by the `pagesize` in `mdb`, enter:

```
> 0t8192*0t30100=E
246579200
```

ORACLE

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

Note

`0t` tells `mdb` the number is decimal and `=E` tells `mdb` to print the result as an 8-byte decimal number.

mdb(1) Utility

mdb can also be used to examine:

- User processes, that is, ELF executables

```
# mdb dhrystone
> ::objects
    BASE  LIMIT SIZE NAME
    10000 11882 1882
/opt/ses/SA400_G/labs/lab/lab6/dhrystone
> _start:b
> :r
mdb: stop at _start
mdb: target stopped at:
_start: clr %fp
> :c
Dhrystone(1.1) time for 500000 passes = 0
This machine benchmarks at 1250000 dhrystones/second
mdb: target has terminated
> $q
```

ORACLE

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

mdb(1) Utility

User process core files

```
# file /progs/cprogs/core
/progs/cprogs/core: ELF 32-bit LSB core file 80386
Version 1, from 'grow_mmap'
# cd /progs/cprogs
# mdb grow_mmap core
Loading modules: [ libc.so.1 ld.so.1 ]
> $?
no process
SIGBUS: Bus Error
%cs = 0x003b      %eax = 0xfed48000
%ds = 0x0043      %ebx = 0xfef9d000
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered on a red rectangular background.

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

When a user program exits and leaves a `core` file, the signal that caused the process to exit with a core file can be displayed with `$? dcmd` in `mdb`.

mdb(1) Utility

System crash dumps

```
# mdb /var/crash/host01/unix.0 /var/crash/host01/vmcore.0
Loading modules: [ unix krtld genunix specfs dtrace
cpu.generic cpu_ms.AuthenticAMD.15 uppc pcplusmp zfs ip
hook neti sctp arp usba s1394 fctl nca lofs audiosup
cpc random crypto fcip ptm ufs sPPP ipc ]
> ::panicinfo
      cpu 0
      thread ffffffff831aa780
      message
BAD TRAP: type=e (#pf Page fault) rp=fffffe8000e9db30
addr=0 occurred in module "unix" due to a NULL pointer
dereference
...<output omitted>
```



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

Note

mdb has an API for creating your own debugger commands and modules.
See *Solaris Modular Debugger Guide* (816-5041.pdf) and `man -s 1 mdb`.

Agenda

- swap
- CPU Performance Counters
- mdb (1) Utility
- **Solaris Studio dbx (1) Utility**
- zonestat Utility
- GUDS Script

ORACLE

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

Solaris Studio dbx(1) Utility

- The dbx(1) tool operates on the core files generated by C, C++, FORTRAN, and Java applications.
- The symbol table is mapped to the core file memory image:
 - It is assumed that the executable is not stripped.
 - It is better if code is compiled with full debug data.
- Users can:
 - Determine the point of failure; work backward to cause
 - Explore image by using symbol names
 - Read register values
 - Run applications with dbx(1)
 - Set breakpoints; observe key functions

ORACLE

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

The dbx(1) utility not only helps the user to analyze a failed process, but also enables the user to instrument a program and control its execution from the start. Users can set breakpoints in the code to halt a process's execution at any chosen instruction point. A likely first choice is the function entry point that is closest to the location of a repeatable failure.

To use dbx(1) on a core file, determine the application that created it with the `file(1)` command. The executable file maps the core file, which is just the process's memory image at the point of failure:

```
ksh:host28# 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)
```

Agenda

- swap
- CPU Performance Counters
- mdb (1) Utility
- Solaris Studio dbx (1) Utility
- zonestat Utility
- GUDS Script

ORACLE

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

zonestat Utility

- The `zonestat` utility reports utilization statistics on currently running zones.
 - CPUs
 - Memory
 - Networking
 - Resource controls
- When run from within a non-global zone:
 - Only processor sets visible to that zone are reported
 - All of the memory resources are reported
- The `zonestatd` daemon:
 - Starts when the zone boots up
 - Has no configurable components

The Oracle logo, consisting of the word "ORACLE" in white, uppercase, sans-serif font, centered on a red rectangular background.

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

The `zonestat` utility reports the CPU, memory, networking, and resource control utilization of the currently running zones. Each zone's utilization is reported both as a percentage of system resources and the zone's configured limits. The `zonestat` utility prints a series of interval reports at the specified interval. It optionally also prints one or more summary reports at a specified interval.

When run from within a non-global zone, only processor sets visible to that zone are reported.

The non-global zone output includes all of the memory resources and the limits resource.

The `zonestatd` system daemon is started during system boot. The daemon monitors the utilization of system resources by zones, as well as zone and system configuration information such as `psrset` processor sets, pool processor sets, and resource control settings. There are no configurable components.

zonestat Utility: Examples

Display a summary of web zone memory utilization every five seconds.

```
root@s11-serv1:~# zonestat -z web -r physical-memory 5
...
Interval: 2, Duration: 0:00:10
PHYSICAL-MEMORY      SYSTEM MEMORY
mem_default          3499M
                        ZONE  USED  %USED   CAP   %CAP
                        [total] 982M 28.0%   -     -
                        [system] 362M 10.3%   -     -
                        web 46.8M 1.33% 50.0M 93.7%
```

ORACLE

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

This slide shows memory utilization by a web zone at five-second intervals.

The fields displayed are:

- **SYSTEM-MEMORY:** The total amount of memory available on the physical host
- **ZONE:** The zone using the resource
- **[total]:** The total quantity of resource used system wide
- **[system]:** The quantity of resource used by the kernel or in a manner not associated with any particular zone
- **USED:** The amount of resource used
- **%USED:** The amount of resource used as a percent of the total resource
- **CAP:** The cap on a given resource if a zone is configured to have one
- **%CAP:** The amount of resource used as a percent of a zone's configured cap

zonestat Utility: Examples

Report on the processor sets (psets) once a second for one minute.

```
root@s11-serv1:~# zonestat -r psets 1 1m
...
PROCESSOR_SET      TYPE  ONLINE/CPUS  MIN/MAX
cloud_pset1    pool-pset      1/1      1/1
                ZONE    USED %USED      CAP  %CAP    SHRS  %SHR  %SHRU
                [total] 0.00 0.84%      -    -      -    -    -
                [system] 0.00 0.20%      -    -      -    -    -
                storage 0.00 0.32%    0.30 1.07%      -    -    -
                web    0.00 0.31%    0.25 1.27%      -    -    -
...
```

ORACLE

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

In this example, `zonestat` reports on the processor sets (psets) once a second for one minute.

The fields displayed are:

- **ZONE:** The zone using the resource
- **[total]:** The total quantity of resource used system wide
- **[system]:** The quantity of resource used by the kernel or in a manner not associated with any particular zone
- **USED:** The amount of resource used
- **%USED:** The amount of resource used as a percent of the total resource
- **CAP:** If a zone is configured to have a cap on a given resource, the cap is displayed in this column.
- **%CAP:** The amount of resource used as a percent of a zone's configured cap
- **SHRS:** The number of shares allocated to a zone
- **%SHRS:** The fraction of the total shares allocated to a zone
- **%SHRU:** Of the share allocated to a zone, the percentage of CPUs used

zonestat Utility: Examples

Report on high utilizations.

```

root@s11-serv1:~# zonestat -q -R high 5s 20s
Report: High Usage
  Start: Wed May  2 16:55:54 MDT 2012
    End: Wed May  2 16:56:14 MDT 2012
  Intervals: 4, Duration: 0:00:20
SUMMARY          Cpus/Online: 2/2   PhysMem: 3499M  VirtMem: 4523M
          ---CPU---  --PhysMem--  --VirtMem--  --PhysNet--
          ZONE  USED %PART  USED %USED  USED %USED  PBYTE %PUSE
    [total]  0.18 9.20%  965M 27.5% 1432M 31.6%   538 0.00%
    [system] 0.04 2.48%  344M 9.84%  937M 20.7%    -  -
      global 0.13 13.1%  476M 13.6%  361M 7.98%   538 0.00%
    database 0.00 0.05%  24.1M 0.69%  21.4M 0.47%    0 0.00%
      storage 0.00 0.06%  39.6M 1.13%  32.3M 0.71%    0 0.00%
        users 0.00 0.10%  34.0M 0.97%  46.4M 1.02%    0 0.00%
          web 0.00 0.07%  45.5M 1.30%  33.5M 0.74%    0 0.00%

```

ORACLE

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

In this example, `zonestat` monitors silently at five-second intervals for 20 seconds, and then produces a report on high utilizations.

The fields displayed are:

- **ZONE:** The zone using the resource
- **[total]:** The total quantity of resource used system wide
- **[system]:** The quantity of resource used by the kernel or in a manner not associated with any particular zone
- **USED:** The amount of resource used
- **%PART:** The amount of CPU used as a percentage of the total CPU in a processor set to which the zone is bound
- **%USED:** The amount of resource used as a percent of the total resource
- **PBYTES:** The number of transmitted bytes that consume physical bandwidth
- **%PUSE:** The sum of **PRBYTES** and **POBYTES** as a percent of the total available physical bandwidth

Agenda

- swap
- CPU Performance Counters
- mdb (1) Utility
- Solaris Studio dbx (1) Utility
- zonestat Utility
- **GUDS Script**

ORACLE

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

GUDS Script

- The GUDS script is a `ksh` script that is designed to collect the data needed to analyze performance issues.
 - Collects a specific set of files
 - Runs a series of operating system utilities
- The collected performance data is:
 - Stored by default in `/var/tmp/guds/SR#`
 - Packaged and compressed in `zip` format for delivery to Oracle support
- For the best analysis, you should run this script twice.
 - Run the script when your system is experiencing a performance problem.
 - Run it again when your system is doing well.

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, is positioned on the right side of a red horizontal bar.

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

The GUDS script is a `ksh` script that is designed to collect the data needed to analyze performance issues. The script collects a specific set of files and runs a series of operating system utilities, the selection of which is based on the specified level.

The duration, number of iterations, and other variables are provided to the script either as a set of command-line arguments or as prompts for input in interactive mode. The collected data is then packaged and compressed in a standard format for delivery to Oracle support.

For the best analysis, you should run this script twice.

1. Run the script when the system is experiencing a performance problem.
2. Run it again when the machine is doing well.

This will allow you to compare bad data to baseline data. Some numbers are meaningless without baseline data.

Quiz

To report event counters for each CPU in the system, use:

- a. The `swap` utility
- b. The `cputrack` utility
- c. The `cpustat` utility
- d. The `mdb` utility

ORACLE

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

Answer: c

Quiz

mdb has an API for creating your own debugger commands and modules.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

When a user program exits and leaves a core file, which dcmd can be used in mdb to display the signal that caused the process to exit with a core file?

- a. nm
- b. formats
- c. print
- d. \$?

ORACLE

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

Answer: d

Summary

In this lesson, you should have learned how to:

- Describe and interpret the `swap` command
- Describe and use `cpustat`
- Describe and use `mdb -k` to examine kernel structures
- Describe the `zonestat` utility
- Describe the GUDS script

ORACLE

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

Practice 6 Overview: Using `swap`, bus counters, and `mdb` Tools

This practice covers the following topics:

- Interpreting the output of the `swap` command
- Using `cpustat` to view the CPC counters
- Using `mdb` to view kernel parameters
- Using Solaris Studio `dbx` to test a common application
- Using the GUDS script to gather performance data

ORACLE

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

Processes and Threads

ORACLE

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

Objectives

After completing this lesson, you will be able to:

- Define a process and describe the process address space
- Describe the lifetime of a process
- Describe threads
- Describe and use `lockstat(1M)`
- Describe and display parameters for process management
- Compare the performance of single-threaded and multithreaded processes
- Describe the scheduling model and mechanism

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding the content of this lesson:

- What is a thread? Why is it used?
- What is tunable for processes and threads?

ORACLE

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

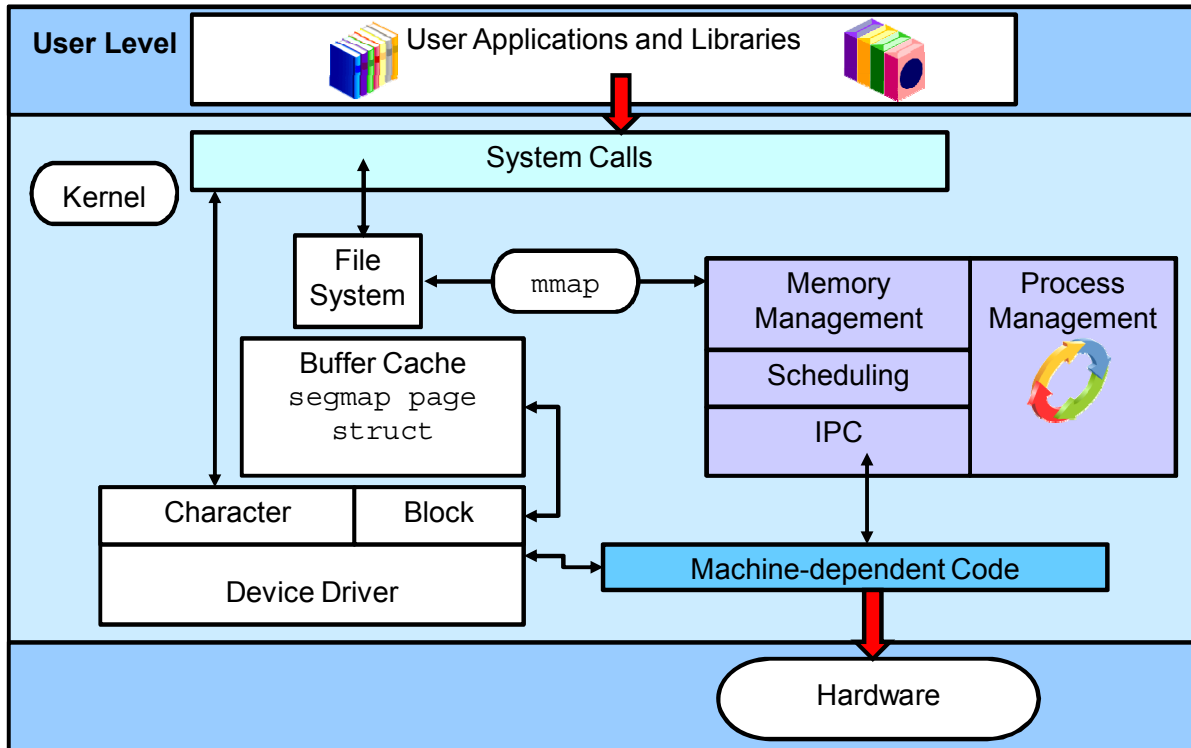
Agenda

- Introduction to the OS process subsystem
- Threads and Locking
- Process-Related Tunable Parameters and Process Limits
- Process Scheduling

ORACLE

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

Layers of the Operating System



ORACLE

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

The following lessons will cover operating system theory and how it relates to system performance. This section of the course will cover:

- Process and threads
- Memory
- Caches and buses
- CPU scheduling
- File systems and disk I/O

The operating system can be thought of as a resource manager. The resources include the hardware—that is, CPUs, memory, disk, and other devices. It also includes process management and file system management.

A process or thread makes an explicit request for the OS to perform a service, which is called a system call. After the system call layer, most of the diagram on the left side of the slide is related to file system management, whereas the right side is related to process management.

Process

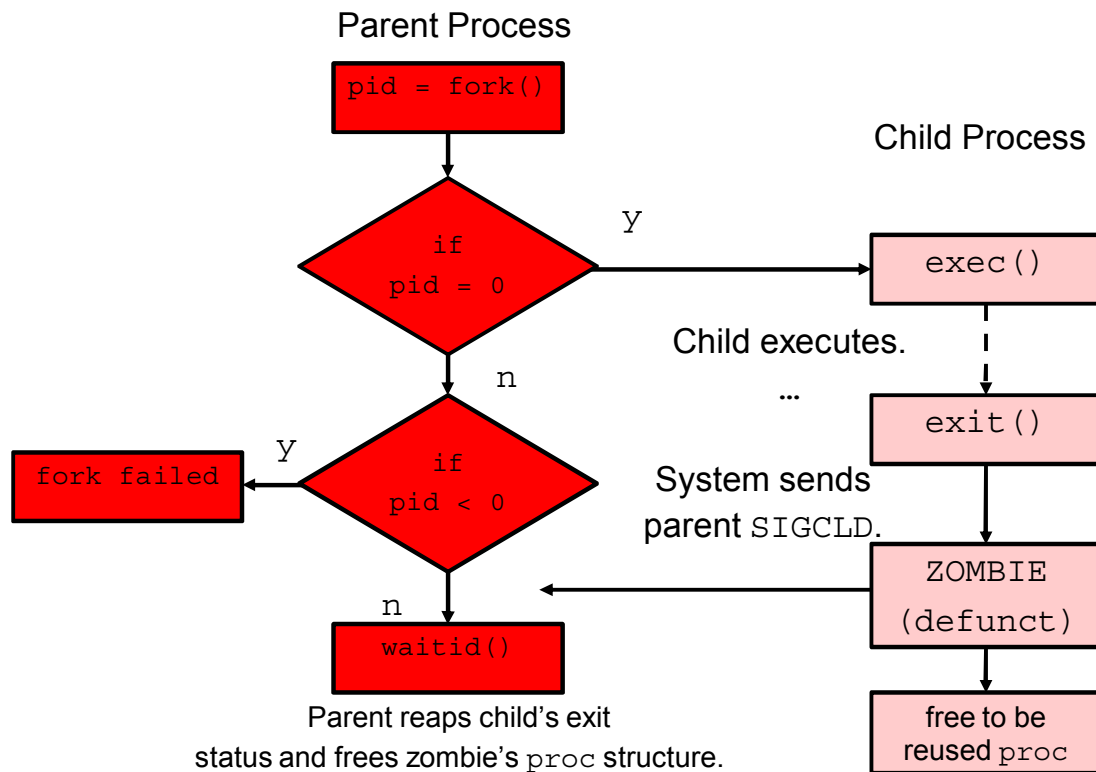
- A process is an instance of a program in execution.
- A process consists of one or more threads of execution in a shared virtual address space.
- The address space is composed of segments such as:
 - Text, data, heap, and stack
 - Library text and data
 - Regular files and temporary work space (anon)
 - Mapped devices, such as frame buffers and raw disk
- A process has stages that are together referred to as a process lifetime.
- A process can communicate with other processes via IPC.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

A process defines the run-time state of a program file. Along with the code instructions and data supplied by the original file, a process includes information about the hardware and operating system resources kept in kernel structures, and current state information in CPU registers.

Life Cycle of a Process



ORACLE

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

A process can issue `fork` requests up to the limit of the user processes that may reside on a system at a time. The limit on the number of nonroot processes that can be created is `max_nprocs - reserved_procs`. A limit may also be set in `/etc/system` on the number of processes a `UID` can create, which is `maxuprc`.

Note: In Oracle Solaris 11, a process may have its rights to create new processes that are restricted through the `ppriv` command. See the man page on the `ppriv` command for more information.

Typical Life Cycle of a Process

The life cycle of a process typically includes the following:

1. The parent calls `fork(2)` to create a child process:
 - Copies kernel structures from parent
 - Duplicates the parent's address space
2. After the child process is created, it may issue an `exec(2)`:
 - `exec(2)` releases the child's old address space.
 - It maps a new address space from an executable file and the child process is now eligible to run.
 - The parent continues or waits for the child to exit.

ORACLE

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

While the child is executing, it can be in one of the following states:

- **On Processor:** The process/thread is executing on a CPU. This state shows as "O" in the `ps` command output, in the `stat (S)` column.
- **Ready:** The process/thread is eligible for CPU time. This state shows as "R" in the `ps` command output.
- **Sleep:** The process/thread is waiting. Reasons include waiting for a lock or waiting for a condition to be true, such as I/O completion or a timer changing to expired. The Sleep state shows as "S" in the `ps` command output.
- **Stopped:** The thread is stopped by a signal or debugger. This state shows as "T" in the `ps` command output.

Typical Life Cycle of a Process

3. When the child process completes, it calls `exit(2)`, which:
 - Sets the child's state (`p_stat`) to `SZOMB`
 - Sets the child's exit status in the `proc` structure
 - Frees the child's other resourcesThe parent can then be notified via a `SIGCHLD` signal.
4. The parent process can then reap the exit status of the child with the `waitid(2)` system call.
5. The `proc` structure is cached so that it may be reused by another process.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered within a solid red rectangular bar.

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

When the child has exited, it can be in the following state:

- **Zombie:** The process has terminated but still has a `proc` structure. This state persists until the parent process reaps the child's exit status.

System Processes

- PID 0 – `sched`: The Swapper (memory scheduler)
- PID 1 – `init`
 - Spawns logins
 - Adopts orphans
 - Manages run levels
- PID 2 – `pageout`: Part of the page daemon that writes out dirty pages
- PID 3 – `fsflush`
 - File system pages synched periodically
 - `autoup` - 30 (seconds)
 - `tune_t_fsflushr` - 1 (once a second)
 - Local disk file systems
 - Not ZFS pages (handled by ARC)



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

The slide lists the four processes created by the kernel at system startup. The swapper (`sched`) is actually the initial process: it forks once to create `init`, a second time to create `pageout`, and a third time to create `fsflush`.

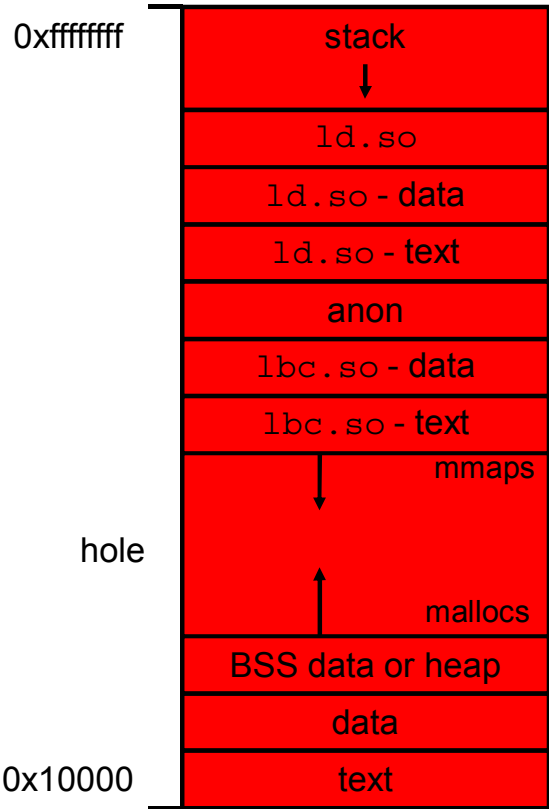
`pageout` and other parts of the page daemon are discussed in the lesson titled “Memory.”

The main reason `fsflush` is syncing pages back to the local file systems is that processes do not have to write all their pages to disk synchronously. To make sure that pages are written to disk in a relatively short time, in case of power failure or panic, `fsflush` cycles through memory every 30 seconds, writing out modified pages to local disks. To avoid causing huge disk I/O spikes every 30 seconds, it runs once a second. The two tunable parameters that govern this are `autoup` and `tune_t_fsflushr`. The `autoup` parameter indicates how long it will take `fsflush` to cycle through memory. The default for `autoup` is 30 seconds. The `tune_t_fsflushr` parameter is the rate to run `fsflush`. This defaults to 1, which means that it is run once per second.

Another reason for running `fsflush` is if the system gets low on memory and the page daemon has to free up pages, most of them should already be written to disk by `fsflush`, so `pageout` will have fewer pages to write out.

Process and Address Space

32-bit SPARC Architecture



ORACLE

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

The area in the address space that holds executable code from the program file is called the *text segment*. The process is given read and execute permissions to the text segment, but not write access. This way the pages in the text segment can be shared between processes running the same program. Data that is paged from the program file is mapped to the *data segment*. Memory that is allocated by the process at run time is mapped to an area called the *heap* or *BSS*. The unused area of the map is referred to as the *hole*.

The *stack segment* is composed of stack frames, one for each function call. The current function's dynamic data, arguments to the function, return values from the function, the return program counter, and the return stack pointer are kept on the *stack*. The stack (in a SPARC architecture) is at the top of the address space and grows down to a limit. Below the stack limit is where mmaped (memory mapped) files are placed. The mapped files include the shared libraries text and data segments, and other user-mapped files.

pmap(1)

```
# pmap `pgrep while1`
8922:   while1
00010000 8K    r-x--  /opt/ses/lab/lab6/while1
00020000 8K    rwx--  /opt/ses/lab/lab6/while1
FF200000 1216K r-x--  /lib/libc.so.1
FF330000 40K    rwx--  /lib/libc.so.1
FF33A000 8K     rwx--  /lib/libc.so.1
FF390000 24K    rwx--  [ anon ]
FF3B0000 208K  r-x--  /lib/ld.so.1
FF3F4000 8K     rwx--  /lib/ld.so.1
FF3F6000 8K     rwx--  /lib/ld.so.1
FFBFE000 8K     rwx--  [ stack ]
total      1536K
```

ORACLE

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

The `pmap` command is an easy way to look at what is in a process address space. This is the address space of a very simple process. Notice that the text is shown at the top and the stack appears at the bottom.

pmap(1)

```
# pmap -x `pgrep while1`
8922:      while1
Address  Kbytes  RSS  Anon  Locked  Mode   Mapped File
00010000 8         8    -    -    r-x--   while1
00020000 8         8    8    -    rwx--   while1
FF200000 1216      848    -    -    r-x--   libc.so.1
FF330000 40        40   40    -    rwx--   libc.so.1
FF33A000 8         8    8    -    rwx--   libc.so.1
FF390000 24        16   16    -    rwx--   [ anon ]
FF3B0000 208       208    -    -    r-x--   ld.so.1
FF3F4000 8         8    8    -    rwx--   ld.so.1
FF3F6000 8         8    8    -    rwx--   ld.so.1
FFBFE000 8         8    8    -    rwx--   [ stack ]
-----
total Kb      1536      1160      96    -
```

ORACLE

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

The `-x` option to `pmap` includes other columns that tell you how many KB of each segment is currently resident in memory (RSS) and how much is backed up by swap (Anon).

Interrupts

- Interrupts are generated mostly by hardware.
- Software interrupts are used for:
 - Real-time timers
 - Cross-calls
- There are 16 interrupt levels defined in the kernel.
- Levels 1–10 can be assigned a thread and block.
- Levels 11–15 cannot block.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

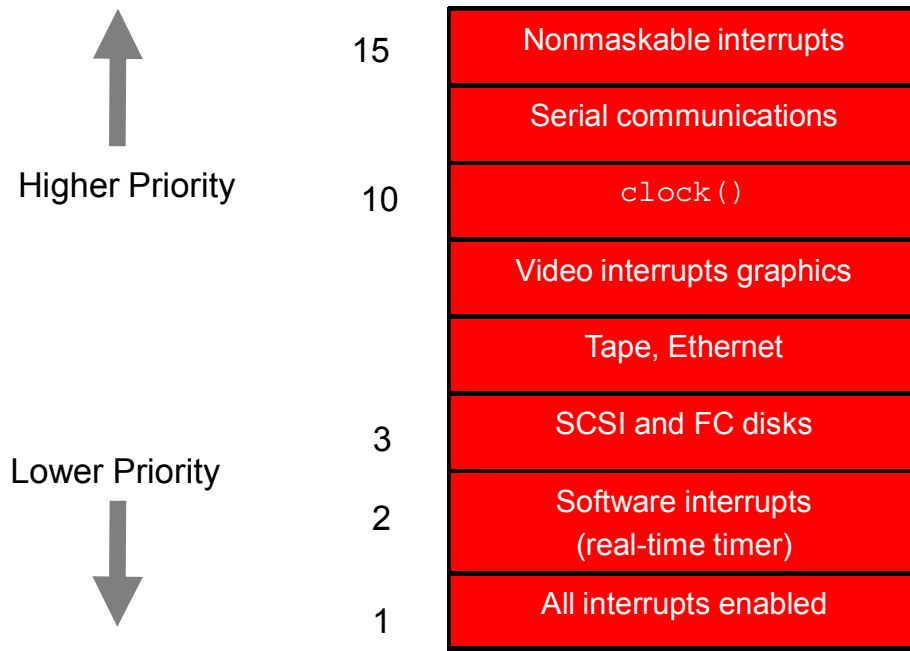
Software interrupts can run at an interrupt level of 14 for cross-calls. Most cross-calls are caused by the memory management unit (MMU).

`clock()` runs at interrupt level 10 a hundred times a second to handle most timers on the system, including time slices on the CPU. High-resolution timers are handled by the cyclic subsystem.

Note: There is a project to make Solaris a tickless kernel:

- <http://hub.opensolaris.org/bin/view/Project+tickless/>
- <http://arc.opensolaris.org/caselog/PSARC/2009/396/mail>

Interrupts



ORACLE

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

Interrupts will not appear in any of the process tools discussed in this lesson because they are not processes. Interrupts at level 10 and below run as system threads.

A *system thread* consists of a kernel thread and a kernel stack and it runs in the kernel's address space (`kas`). System threads are typically visible only in tools such as `mdb` and `dtrace`. They appear as the `sched` process because most of them are created at boot time by `sched`. Some examples of what the system threads are used for include:

- Interrupt threads
- Idle threads
- The callout thread
- The `pageout_scanner`
- `taskq` threads

They can be seen in `mdb -k` by examining the thread list with the `::threadlist` dcmd or `$<threadlist`. The kernel stack is the best way to identify their use.

Interrupt Assignments

```

root@server1:~# echo ":::interrupts" | mdb -k
CPU/Vect  IRQ  IPL  Bus    Trg Type  Share APIC/INT#  ISR
0/0x23    16   9    PCI    Lvl Fixed 1    0x0/0x10  ehci_intr
1/0x20     9   9    PCI    Lvl Fixed 1    0x0/0x9   acpi_wrapper_isr
1/0x21    23   9    PCI    Lvl Fixed 1    0x0/0x17  ehci_intr
2/0x20     -   5    PCI    Edg MSI   1    -         ahci_intr
2/0x21     -   6    PCI    Edg MSI   1    -         e1000g_intr_pciexpress
2/0x22     4  12    ISA    Edg Fixed 1    0x0/0x4   asyintr
3/0x20     -   7    PCI    Edg MSI   1    -         pcieb_intr_handler
all/0xf0   -  15    -      Edg IPI   1    -         xc_serv
all/0xf1   -  11    -      Edg IPI   0    -         poke_cpu
all/0xf2   -  14    -      Edg IPI   1    -         kcpc_hw_overflow_intr
all/0xf3   -  15    -      Edg IPI   1    -         apic_error_intr
all/0xf4   -  14    -      Edg IPI   1    -         cbe_fire
all/0xf5   -  14    -      Edg IPI   1    -         cbe_fire

```

ORACLE

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

This slide shows the current hardware interrupts assignments on a four-core x86 server.

Interrupt Controls

- The `intrd` daemon
 - Is available in Oracle Solaris 11 and later
 - Monitors the assignments between interrupts and CPUs for interrupt imbalance
 - Implements new assignments when needed

```
root@server1:~# svcs intrd STATE STIME FMRI online Jan_13  
svc:/system/intrd:default
```

- The `psradm` utility
 - Is used to change the operational status of processors
 - Includes the `-i` option that sets the specified processors to the `no-intr` state



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

The `intrd` daemon is started at boot time to monitor the assignments between interrupts and CPUs. If `intrd` decides that the current assignments are imbalanced and harmful to system performance, it will generate and implement new assignments.

The `psradm` utility changes the operational status of processors. The legal states for the processor are on-line, off-line, spare, faulted, and no-intr. A no-intr processor processes LWPs but is not interruptible by I/O devices

Interprocess Communication (IPC)

Processes communicate with each other by using the following IPC mechanisms:

- Sockets: BSD-based IPC over IP or local system
- TLI and XTI: Transport Layer Interface from SystemV and Xopen Transport Interface
- Pipes: Uses output from one process as input to another
- System V Shared Memory: Includes Shared Memory, Message Queues, and Semaphores
- POSIX Shared Memory: Is similar to System V Shared Memory; uses memory-mapped files
- Solaris Doors: Enables a process to run a thread in the address space of another process

ORACLE

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

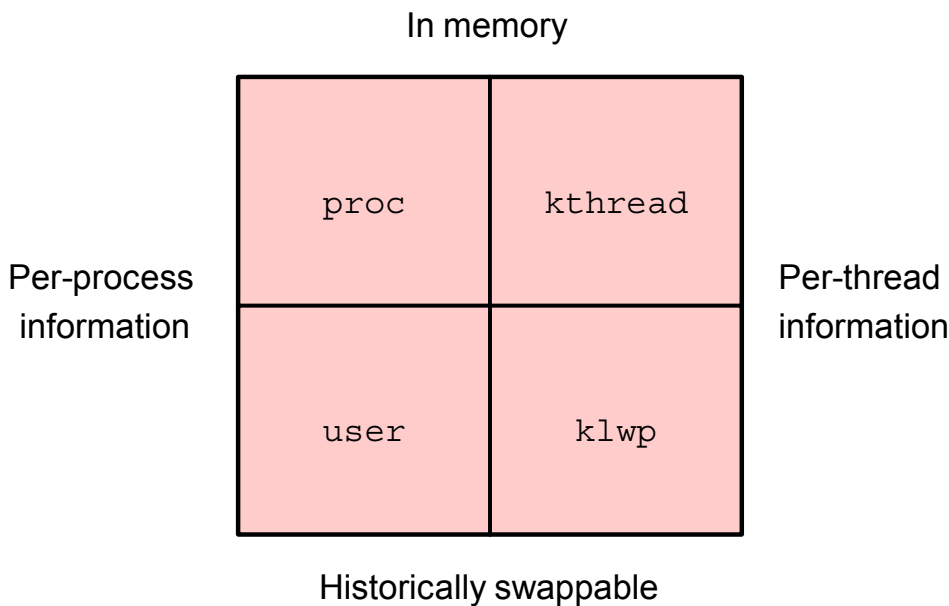
For System V Shared Memory, the IPC resources in use can be reported by using the `ipcs(1)` command.

Processes can also communicate through the use of shared files and locking, or through the use of signals.

The Solaris Doors mechanism allows a process to effectively run a thread in the address space of another process, avoiding having to duplicate mappings to the file system and to physical memory. It uses a fast, low-overhead, context-switching mechanism between the door client and door server, and passes information between them by using door (file) descriptors.

Note: See *UNIX Network Programming, Volume 2: Interprocess Communications* (2nd Edition) by W. Richard Stevens for programming with doors and other forms of IPC. See `man libdoor` or `man -s 3DOOR door_create`. See *Solaris Internals, solaris 11 and OpenSolaris Kernel Architecture* (2nd Edition).

Process Kernel Structures

**ORACLE**

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

`proc`, `kthread`, `user`, and `klwp` are the four main data structures used by the kernel to maintain the state of a process. In older versions of SunOS, there was only a `proc` structure and a `user` structure. The `proc` structure contained all the information that had to be kept in memory all the time even if the process was swapped out. The `user` structure contained the information that had to be in memory while the process was executing, but could be swapped out if the system was low on memory.

When the kernel was multithreaded in SunOS 5.0, structures were needed to keep track of each lightweight process (LWP) or thread. The kernel thread (`kthread`) contained the information that always had to be kept in memory and the `klwp` contained the information that could be swapped out. Now that the kernel has evolved and memory has become much cheaper, the only kernel information that is currently swapped out are the kernel stacks for swapped-out LWPs.

Process-Related Performance Issues

- A rising or high number of context switches caused by:
 - Excessive process creation
 - Preemption by higher-priority threads
 - High volume of short-lived processes
- A large number of zombie processes could keep the system from creating new processes.

ORACLE

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

Agenda

- Introduction to the OS process subsystem
- **Threads and Locking**
- Process-Related Tunable Parameters and Process Limits
- Process Scheduling

ORACLE

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

Threads

A thread is a sequence of instructions that reside in a process.

- Independent tasks can execute simultaneously.
- Create threads with `thr_create(3C)` or the POSIX version `pthread_create(3C)`.
- Threads share the `PID` and `user` structure.
- Each thread is represented by a `kthread` and `klwp` structure in the kernel.
- Each thread has a unique thread ID (`TID`) within the process.
- Each thread has its own stack but shares the rest of the address space.
- Locks are used to protect shared and writable data.

ORACLE

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

Applications can use threads to execute independent tasks simultaneously while sharing the `process` structures and address space. Because each thread can access the address space of a process, protecting data that could be modified by multiple threads is essential. A programmer must ensure that all shared and writable data is protected with locks.

pmap(1): MT Process

```
# pmap -x 9037
```

Address	Kbytes	RSS	Anon	Locked	Mode	Mapped File
00010000	8	8	-	-	r-x--	test_gm
00020000	8	8	8	-	rwX--	test_gm
FF0E0000	64	64	64	-	rwX--	[anon]
FF1FA000	8	8	8	-	rwX-R	[stack tid=2]
FF200000	1216	848	-	-	r-x--	libc.so.1
FF330000	40	40	40	-	rwX--	libc.so.1
FF33A000	8	8	8	-	rwX--	libc.so.1
FF33E000	8	8	-	-	rwxs-	[anon]
FF350000	64	64	64	-	rw---	[anon]
FF370000	64	64	64	-	rw---	[anon]
FF390000	24	16	16	-	rwX--	[anon]
FF3A0000	8	8	-	-	r-x--	libc_psr.so.1
FF3B0000	208	208	-	-	r-x--	ld.so.1
FF3F0000	8	8	8	-	rwX--	[anon]
FF3F4000	8	8	8	-	rwX--	ld.so.1
FF3F6000	8	8	8	-	rwX--	ld.so.1
FFBFE000	8	8	8	-	rwX--	[stack]

ORACLE

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

This is the address space of a multithreaded process. Each thread gets its own stack but shares the remaining segments of the address space with the other threads.

Threads

To list all the threads or LWPs, use the `-L` option in the `ps` command—for example:

```
# ps -cLe
  PID LWP CLS PRI TTY LTIME      CMD
    0   1 SYS  96  ?  0:43      sched
    1   1 FSS  59  ?  5:44      init
    2   1 SYS  98  ?  0:14    pageout
    3   1 SYS  60  ? 173:18  fsflush
  394   1 FSS  59  ?  0:00    smcboot
    7   1 FSS  29  ?  0:00    svc.star
    7   2 FSS  59  ?  0:00    svc.star
    7   3 FSS  59  ?  0:14    svc.star
    7   4 FSS  59  ?  0:00    svc.star
    7   5 FSS  29  ?  0:00    svc.star
    7   6 FSS  29  ?  0:00    svc.star
    7   7 FSS  59  ?  0:00    svc.star
... <output omitted>
```

ORACLE

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

Locks

Synchronization Primitives in Oracle Solaris

Lock	Description
Mutex (Mutual exclusion lock)	Provides exclusive access to data. The current thread must surrender the lock before another thread can access it. The state of the lock is associated with the critical data, not the thread, so it is visible to all processors at once.
Conditional variable	Is used to wait for a particular condition to become true. Access to the condition variable is protected with a mutex lock.
Counting semaphore	Controls thread access to a number of resources. As each thread takes a resource, the count of the number of resources is decremented. When the count goes to 0, subsequent threads requesting a resource will sleep. When a thread gives up a resource, it will wake up any threads that are waiting.
Multiple-reader, single-writer	Several threads have read access to the data. At any time, only one thread can acquire the lock in order to write data.



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

In a multithreaded process, two threads may want to use the same data structure simultaneously. Without some protection scheme around the data, the changes made by each thread could occur in such a way that the resulting data becomes corrupted.

Programmers use locks to ensure orderly access and safe data updates. Locks can protect one datum value or a group of them. These lock types are generally referred to as *synchronization primitives*. A thread sets a lock when accessing shared writable data. The thread will release the lock when the data access is complete. Any thread requesting the lock while it is held will have to block or spin until the lock is released. Spin means to stay on a CPU and keep checking the lock until it is available.

Locking Problems

Errors in locking design cause problems that are difficult to detect:

- Thread A acquires Lock 1 and blocks waiting for Lock 2. Thread B has already acquired Lock 2 and blocks waiting for Lock 1. This condition is called `deadlock`.
- A thread dies without releasing a lock.
- Threads assume that they will always access data in the same order. This causes a race condition, because the outcome depends on the order of execution.
- Multiple locks are acquired in one order, but not released in the reverse order, resulting in a deadlock or race condition.

ORACLE

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

A programmer must decide on how *granular* the locking strategy should be. A *coarse-grained* strategy implies that the locks used protect large areas of data.

Such schemes are typically simple, but at the expense of making a lot of data inaccessible to all but one thread at a time. A *fine-grained* strategy implies more locks to protect smaller areas of data. Fine-grained schemes may provide faster and more scalable concurrent processing, but at the expense of more complex code and a greater likelihood of deadlock, race conditions, and so on. There is also the additional overhead of memory consumption and the processor cycles taken to check and release the locks. These are all the same trade-offs that were considered for the operating system itself.

lockstat Utility

The `lockstat` utility reports statistics on kernel-level locking.

- It uses DTrace.
- Only `root` and users with `dtrace_kernel` privileges can use `lockstat` due to the intrusion effect.
- Locking events are divided into two categories:
 - Hold Events: Acquire and release locks.
 - Contention Events: When the lock is already owned, a thread can:
 - Spin: Repeatedly request a lock, if the owner is on a CPU
 - Block: Until lock is released, if the owner is not on a CPU

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a solid red rectangular background.

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

lockstat Utility

To see usage and options, enter:

```
# 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]
  -A      watch all lock events [equivalent to -CH]
...<output omitted>
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

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

lockstat Utility

Some useful options include:

```
-C      watch contention events [on by default]
-E      watch error events [off by default]
-h      histograms for event times
-d      duration only watch events longer than <duration>
-c      coalesce lock data for arrays like pse_mutex[]
-p      parseable output format (awk(1)-friendly)
-P      sort lock data by (count * avg_time) product
-D      n only display top <n> events of each type
-x      opt[=val] enable or modify DTrace options
```

ORACLE

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

lockstat Utility

This `lockstat` command reports hold events for the top three callers on each lock type:

```
# lockstat -H -D 3 sleep 5
Adaptive mutex hold: 823828 events in 5.342 seconds (154206 events/sec)
Count  indv  cuml  rcnt  nsec  Lock              Caller
-----
118034  14%   14%  0.00  2417  0x60083ce3b80  fsflush_do_pages+0x33c
118033  14%   29%  0.00  2403  0x60083ce3b80  tmp_inactive+0x200
118033  14%   43%  0.00  2408  0x60083ce3b80  vn_rele+0x30
-----
Spin lock hold: 17255 events in 5.342 seconds (3230 events/sec)
Count  indv  cuml  rcnt  nsec  Lock              Caller
-----
1068    6%    6%  0.00  3126  hw_copy_limit_8+0x57  dtrace_hres_tick+0x5c
1017    6%   12%  0.00  6816  xc_sys_mutex+0x6      xc_all+0x438
943     5%   18%  0.00  3171  0x60022b07288         fss_sleep+0x28
-----
... <output omitted>
```

ORACLE

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

lockstat Utility

Lock types break down into adaptive mutexes, spin locks, and Reader/Writer (R/W) writer locks and reader locks.

The fields of the output from the command in the previous slide are as follows:

- **Count:** The total count of this event type
- **indv:** Percentage this event comprises of all events
- **cuml:** Cumulative percentage this event comprises of all events
- **rcnt:** Average reference count
- **nsec:** Average duration of the event type, in nanoseconds
- **Lock:** Address or symbolic name of the called lock
- **Caller:** Address or symbolic name of the calling function

ORACLE

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

Look for high `indv` values coupled with a slow or fast locking time in the output of the `lockstat` command to help identify a problem. You can identify the module that is causing a problem by referring to the lock name, which is usually prefixed with the module name.

The next example uses the `-P` option to see the contention events that are consuming the most time.

lockstat Utility

```
# lockstat -P -D 3 sleep 5
```

Adaptive mutex spin: 33 events in 5.044 seconds (7 events/sec)

Count	indv	cuml	rcnt	nsec	Lock	Caller
-------	------	------	------	------	------	--------

7	23%	23%	0.00	4485	0x300003bb540	timeout_generic+0x4c
2	11%	34%	0.00	7650	0x6002200f8a0	taskq_thread+0x1cc
2	11%	45%	0.00	7650	0x300003bb540	callout_list_expire+0x6c

Spin lock spin: 440 events in 5.044 seconds (87 events/sec)

Count	indv	cuml	rcnt	nsec	Lock	Caller
-------	------	------	------	------	------	--------

213	44%	44%	0.00	825	0x600222282b8	disp+0x7c
149	38%	83%	0.00	1016	cpu0_disp	disp+0x7c
55	11%	94%	0.00	785	0x60022228258	disp+0x7c

ORACLE

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

The `-P` option sorts the output by (count * time) product, where count is the number of events and time is the number of nanoseconds spent on each event.

DTrace lockstat Provider

```
# dtrace -ln lockstat:::
# cat adaptive_mutex.d
#! /usr/sbin/dtrace -qs

lockstat:::adaptive-block
{
    printf("%4s%4s%9s%9s%20s\t%s\n",
        "CPU", "TID", "PID", "UID", "Wait time", "Command" );

    printf("%4d%4d%9d%9d%20d\t%s\n",
        curcpu->cpu_id,
        curlwpsinfo->pr_lwpid,
        curpsinfo->pr_pid,
        curpsinfo->pr_uid,
        arg1,
        curpsinfo->pr_psargs );
    stack();
}
```



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

The DTrace script in the slide looks for one type of contention (blocking) on one type of lock (adaptive mutex), and prints the kernel stack of the process to see what part of the kernel it was in when it had to block. This script could easily be adapted to look for all types of contention on all types of locks.

A similar script could be written for user lock contention by using the `plockstat` provider in DTrace.

adaptive_mutex.d

```
# ./adaptive_mutex.d
CPU TID PID UID Wait time Command
  2   1   3   0      11619800 fsflush
    ufs`logmap_add_buf+0x2c
    ufs`top_log+0xb0
    ufs`ufs_iupdat+0x3f0
    ufs`ufs_trans_iupdat+0xb8
    ufs`ufs_sync_inode+0x130
    ufs`ufs_scan_inodes+0x14c
    ufs`ufs_update+0x238
    ufs`ufs_sync+0x2c
    genunix`fsflush+0x4e0
    unix`thread_start+0x4
CPU TID PID UID Wait time Command
  0   1 13980   0      38800 find /
    ufs`ufs_rmiddle+0x24
... <output omitted>
```

ORACLE

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

The slide shows the output of the `adaptive_mutex.d` script with a `find` running on a system when `fsflush` was trying to sync out dirty file system pages at the same time.

Performance Threads

A multithreaded application:

- Breaks separate tasks into routines that run concurrently
- Usually scales better on MP and CMT systems than a multiprocess approach
- Shares address space and MMU address translation resources
- Avoids the overhead and complexity of IPC
- Reduces the demand on system resources for scheduling and process management

A `fork(2)` can take up to five to six times longer than `thr_create(3C)` to execute.

ORACLE

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

Even on a single processor system, multiple threads have some advantages over multiple processes.

A multithreaded application does not automatically outperform a multiprocess application. The existing application must be analyzed to ensure that a concurrent execution of threads will be more effective. Some applications may have to lock data so frequently or for such long periods of time that the cost of protecting data may outweigh the benefit of parallel thread execution.

Agenda

- Introduction to the OS process subsystem
- Threads and Locking
- **Process-Related Tunable Parameters and Process Limits**
- Process Scheduling

ORACLE

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

Process-Related Tunable Parameters

Parameter	Default Value	Minimum Value	Maximum Value
maxusers	Lesser of the amount of memory in MB or 2048	1	Defaults up to 2048 tunable up to 4096
max_nprocs	(16 x maxusers) + 10	266	maxpid
maxuprc	(max_nprocs - reserved_procs)	1	(maxpid - reserved_procs)
pidmax	30,000	266	999,999
reserved_procs	5	5	MAXINT
ngroups_max	16	0	1024

- Refer to the *Process-Sizing Parameters* section in the *Oracle Solaris 11 Tunable Parameters Reference Manual* at http://docs.oracle.com/cd/E26502_01/html/E29022/index.html.

ORACLE

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

The table in the slide shows the process-related tunable parameters.

- maxusers: A number of subsystems that are still derived from maxusers:
 - The maximum number of processes on the system
 - The number of quota structures held in the system
 - The size of the directory name look-up cache (DNLC)

When the default number of user processes is too low, the following message displays on the system console:

out of processes

You might also change this parameter when the default number of processes is too high, as in the following situations:

- Database servers that have a lot of memory and relatively few running processes can save system memory when the default value of maxusers is reduced.
- If file servers have a lot of memory and a few running processes, you might reduce this value.
- If compute servers have a lot of memory and a few running processes, you might reduce this value.

- `max_nprocs`: Specifies the maximum number of processes that can be created on a system. This includes system processes and user processes. This value is also used in determining the size of several other system data structures:
 - Determining the size of the directory name lookup cache (if `ncsize` is not specified)
 - Verifying that the amount of memory used by configured system V semaphores does not exceed system limits
 - Configuring Hardware Address Translation resources for x86 platforms

Changing this parameter is one of the steps necessary to enable support for more than 30,000 processes on a system.

- `maxuprc`: Specifies the maximum number of processes that can be created on a system by any one user. You use this parameter to specify a hard limit for the number of processes a user can create that is less than the default value of however many processes the system can create.
- `pidmax`: Specifies the value of the largest possible process ID. You must change this parameter to enable support for more than 30,000 processes on a system.
- `reserved_procs`: Specifies the number of system process slots to be reserved in the process table for processes with a UID of `root` (0). Consider increasing to 10 + the normal number of UID0 (`root`) processes on system. This setting provides some cushion if it becomes necessary to obtain a `root` shell when the system is otherwise unable to create user-level processes.
- `ngroups_max`: Specifies the maximum number of supplemental groups per process. Change this parameter when you want to increase the maximum number of groups.

Showing the Maximum Number of Processes and Limitations

There are a number of ways to display the current and the maximum number of processes and limitations.

1. `sar -v`

```
# sar -v 1 1
SunOS host11 5.10 Generic_141444-09 sun4u 12/19/2009

23:11:41 proc-sz    ov inod-sz          ov file-sz    ov lock-sz
23:11:42 265/30000 0 152343/152343 0 2234/2234 0 0/0
```

The `proc-sz` field indicates the current number of processes.

ORACLE

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

Showing the Maximum Number of Processes and Limitations

2. `kstat -p`

```
# kstat -p :::nproc
unix:0:system_misc:nproc      264
# kstat -p :::v_proc
unix:0:var:v_proc      30000
```

Where `v_proc` is set equal to `max_nprocs` at system startup

ORACLE

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

Showing the Maximum Number of Processes and Limitations

3. `mdb -k`

```
# mdb -k
Loading modules: [ unix genunix specfs dtrace ufs sd
mpt pcisch ip hook neti sctp arp usba fcp fctl nca lofs
zfs md cpc random crypto wrsmd fcip logindmux ptm sPPP
nfs ipc ]
> nproc/D
nproc:
nproc:          264
> max_nprocs/D
max_nprocs:
max_nprocs: 30000
> v::print v_proc
v_proc = 0x7530
> 7530=D
          30000
> nthread/D
nthread:
nthread:      1288
```

Where `nthread` is the current number of kernel threads

ORACLE

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

Showing the Maximum Number of Processes and Limitations

4. dtrace -n

```
# dtrace -qn 'tick-5sec{printf("nproc = %d nthread =  
%d\n", `nproc, `nthread);}'  
nproc = 267 nthread = 1294  
nproc = 267 nthread = 1296  
nproc = 267 nthread = 1297  
^C
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

Showing the Maximum Number of Processes and Limitations

5. `ulimit -a`

```
root@server1:~# ulimit -a
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
open files              (-n) 256
pipe size               (512 bytes, -p) 10
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 29995
virtual memory          (kbytes, -v) unlimited
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

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

The `ulimit` command sets or gets limitations on the system resources available to the current shell and its descendents. The `-a` option lists all of the current resource limits.

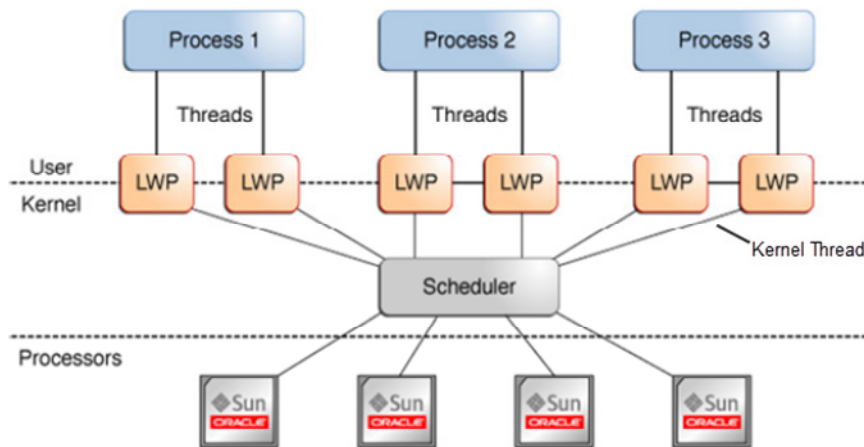
Agenda

- Introduction to the OS process subsystem
- Threads and Locking
- Process-Related Tunable Parameters and Process Limits
- Process Scheduling

ORACLE

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

Process Scheduling



ORACLE

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

A fundamental job of the operating system is to arbitrate which processes get access to the system's resources. The process scheduler, which is also called the dispatcher, is the portion of the kernel that controls the allocation of CPU to processes. The scheduler supports the concept of scheduling classes. Each class defines a scheduling policy that is used to schedule processes within the class.

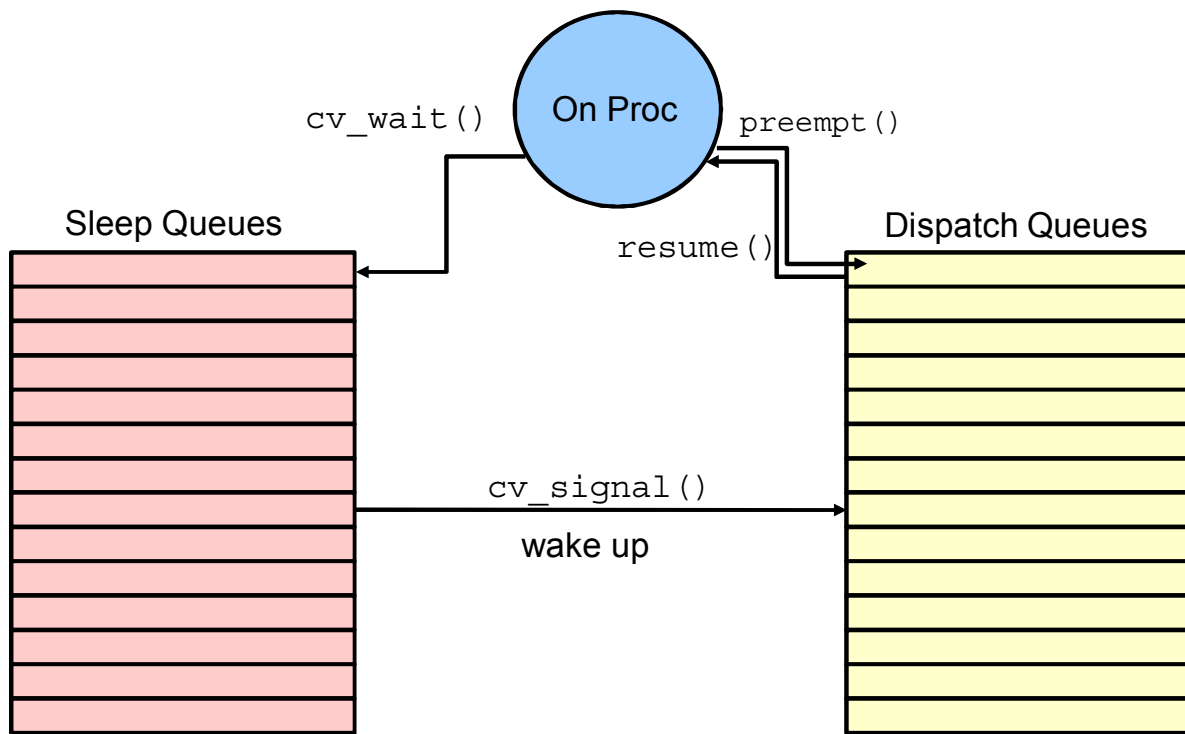
The default scheduler in the Solaris operating system, the TS scheduler, tries to give every process relatively equal access to the available CPUs. However, you might want to specify that certain processes be given more resources than others.

The illustration in the slide shows the relationship between user process threads and process scheduling. Threads are implemented by a library that utilizes the underlying kernel-supported threads of control, called lightweight processes (LWPs).

An LWP is a means of achieving multitasking. An LWP runs in user-space on top of a single kernel thread, and shares its address space and system resources with other LWPs within the same process. All the LWPs in the system are scheduled by the kernel onto the available CPUs according to their scheduling class and priority.

Each process contains one or more LWPs, each of which runs one or more user threads. There is no one-to-one mapping between user threads and LWPs. User-level unbound threads can freely migrate from one LWP to another. If there are available processors, the LWPs run in parallel. The OS has no knowledge about what user threads are or how many are active in each process.

Scheduling State Diagram



ORACLE

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

Other than to exit, the reason a kernel thread would give up a CPU is either of the following motives:

- Voluntary: to go to sleep waiting for:
 - I/O completion
 - A lock
 - Other condition, for example, a signal
- Involuntary
 - Used up time slice
 - Preempted

Wakeup are usually driven by interrupts or signals, or the lock that the thread was waiting for is released.

Thread States

A thread is either executing or waiting for something:

- When executing, the thread is marked as `TS_ONPROC`.
 - This means that it is on a CPU.
 - In `ps` output, its state is set to “O.”
- When waiting for a CPU, the thread state is `TS_RUN`.
 - This means that it is on a Dispatch Queue.
 - In `ps`, its state is set to `R` for runnable.
- When it is waiting for I/O, a lock, a timer, user input, and so on, its state is set to `TS_SLEEP`.
 - It is on a Sleep Queue.
 - In `ps`, its state is set to `S` for sleeping.

ORACLE

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

The other states that a thread or process could be in is `TS_STOPPED`, zombie state (`SZOMB`), or idle state (`SIDL`), which means that it is undergoing creation in fork.

There are three types of sleep:

- `cv_wait()`: Called a good sleep because the thread cannot be woken up by a signal. It is used for threads waiting for events that should complete soon, such as disk I/O.
- `cv_wait_sig()`: Called bad sleep because it can be woken up by a signal
- `cv_wait_sig_swap()`: Called terrible sleep because not only can the thread be woken up by a signal, but also if the system is low on memory, it can be swapped out. This is used for threads waiting for user input or a timer.

Processes stuck in good sleep cannot be killed, even with a `kill -9`.

Scheduling Classes

Solaris supports six default scheduling classes:

- SYS: System Class
- TS: Timeshare
- IA: Interactive
- FSS: Fair Share Scheduling
- FX: Fixed Priority
- RT: Real-Time

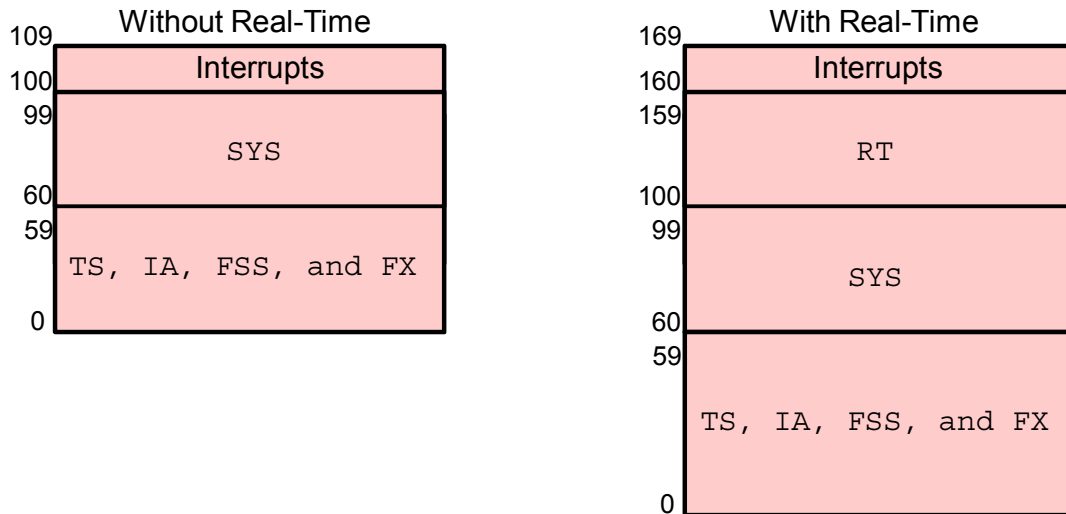
ORACLE

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

Note

In any SVR4 version of UNIX, it is possible to have other non-default scheduling classes. A few have been created by customers and some by Sun Professional Services.

Priorities



ORACLE

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

The slide lists the default priority ranges with and without real-time. Interrupts are always put at the top of all the scheduling classes.

You can display the priorities for each scheduling class by running the `priocntrl -l` command.

Timeshare

Timeshare (TS): The default class for user process threads. The TS class adjusts thread priority over time based on several factors.

- Each thread is allowed to execute for a fixed amount of time, called a time slice or time quantum.
- If a thread uses up its time slice, it is taken off the CPU and its priority is lowered (unless it is already at 0).
- When a thread wakes up, its priority goes up.
- If a thread is waiting on a dispatch queue for too long, its priority is bumped up to prevent CPU starvation.
- TS uses priorities 0 – 59. The priority is assigned from the timeshare dispatch parameter table.

ORACLE

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

Interactive (IA) Scheduling Class

- The IA is the default class for a process/thread that runs in a window environment such as CDE or JDS.
- The Timeshare and Interactive class share:
 - Most of the same routines in the kernel
 - The same dispatch parameter table
 - The same range of priorities: 0–59
- The interactive scheduling class is different because it factors in a boost factor, called `ia_boost`, to a thread when it recalculates its priority.

ORACLE

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

Interactive (IA) Scheduling Class

- It adds `ia_boost` to the priority of the foreground threads in the active window.
- It subtracts `ia_boost` from the priority of threads that are not in the active window or are in the background.
- The boost factor, `ia_boost`, is equal to 10.
- The `nice` command has manual override, so that the `ia_boost` factor is neither added nor subtracted from a `niced` process.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a solid red rectangular background.

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

This was introduced in SunOS 5.4 to give the low-end workstations (SPARC Classics with 16 GB of memory) a more responsive feel. It is implemented with a routine, `ia_set_process_group()`, which is called in response to an `ioctl()` routine called by the X-server. This is done when focus has changed to another window.

Fixed Priority (FX)

- Threads in FX run in the global priority range of 0–60.
- FX threads do not change priority with CPU usage or when they return from sleep.
- This can lead to starvation because:
 - The priority is never lowered even if they exhaust their time slice
 - Their priority is never raised if they are starved on the dispatch queue
- FX has been used for NFS daemons since Solaris 9.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

The fixed priority scheduling class was introduced to fulfill a contractual agreement with an Original Equipment Manufacturer (OEM).

NFS daemons caused a problem for the Solaris Resource Manager (SRM) before Solaris 9, because they were system threads. SRM did not limit or account for the amount CPU used by system threads. The NFS threads that were working on behalf of other processes in a project did not count as processes in that project.

One remedy would be to run NFS daemons as regular LWPs but if they ran in timeshare or the interactive class, they could end up with low priorities, which could induce long network latencies and system overhead.

The solution was to make them regular LWPs, and to put them in the FX scheduling class with a priority of 60. This is the same priority that they had in the `SYS` scheduling class, but their time on the CPU was accountable.

System (sys)

- It is the scheduling class for `pageout`, `fsflush`, and `sched`.
- It is used for system threads, for example:
 - `callout_thread`
 - `pageout_scanner`
 - `taskq` threads
 - Interrupt threads
 - Idle threads
- Most run from 60–99, except:
 - Idle runs at -1
 - Interrupts run at the highest priority
- It is the simplest class: no time slice and fixed priorities.

ORACLE

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

Real-Time (RT)

- RT processes/threads have the highest priority in the system, except for threads that handle interrupts.
- The RT class uses fixed priorities, but are time sliced, so they will round-robin with other RT threads at the same priority.
- The most important thing for real-time is guaranteed dispatch latency.
 - May have to use `psradm -i` to turn off interrupts on a CPU
 - Then bind the RT thread to that CPU with:
 - `pbind()`
 - `psrset()`

ORACLE

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

Bounded response time or dispatch latency means that there is a maximum time limit to get the highest-priority process into execution after it has requested service.

Real-time was not used in older versions of UNIX for two reasons:

- Page faults
- The kernel not being preemptible

The solution for page faults is to allow processes to lock pages in main memory by using the `mlock(3C)` interfaces.

There are two solutions for kernel preemption:

- Preemption points
- A preemptible kernel

Solaris uses a preemptible kernel that is made possible by its multithreaded architecture that is implemented with locks.

Fair Share (FSS)

- This scheduler controls the allocation of available CPU resources among workloads, based on their importance.
- It must not be combined with the TS, FX, and IA classes.
- When using FSS, Oracle Solaris Zones compete with each other, receiving allocation in a `pset` (processor set) proportional to the number of shares assigned to the zone through `zonecfg(1M)`.

```
zonecfg:database> add rctl
zonecfg:database:rctl> set name=zone.cpu-shares
zonecfg:database:rctl> add
    value(priv=privileged,limit=5,action=none)
zonecfg:database:rctl> end
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

Note

See `/usr/include/sys/fss.h` and comments in `fss.c()`.

Fair Share (FSS)

- FSS is integrated with the *project* framework.
- Projects compete with each other for CPU time, receiving CPU allocation that is proportional to the number of shares assigned to the project in `/etc/project`.
 - Each project is allocated a certain number of CPU shares.
 - A project is allocated CPU time based on `cpu-sharevalue`.
- FSS configuration examples:
 - `# projmod -sK "project.cpu-shares=(privileged,3,none)" oracle`
 - `# prctl -r -n project.cpu-shares -v 5 -i project oracle`
 - `# dispadmin -d FSS`
 - `# priocntl -s -c FSS -i all`

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

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

Quiz

A process that has terminated but still has a `proc` structure is in the _____ state, which persists until the parent process reaps the child's exit status.

- a. Sleep
- b. Zombie
- c. Ready
- d. Stopped

ORACLE

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

Answer: b

Quiz

Applications can use threads to execute independent tasks simultaneously while sharing the process structures and address space.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

A multithreaded application does not automatically outperform a multiprocess application. The existing application must be analyzed to ensure that a concurrent execution of threads will be more effective. For instance, some applications may have to lock data so frequently, or for such long periods of time, that the cost of protecting data may outweigh the benefit of parallel thread execution.

- a. True
- b. False

ORACLE

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

Answer: a

Summary

In this lesson, you should have learned how to:

- Define a process and describe the process address space
- Describe the lifetime of a process
- Describe threads
- Describe and use `lockstat(1M)`
- Describe and display parameters for process management
- Compare the performance of single-threaded and multithreaded processes
- Describe the scheduling model and mechanism

ORACLE

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

Practice 7 Overview: Processes and Threads

This practice covers the following topics:

- Examining process life cycles by using `truss`
- Using the `lockstat` command
- Using the `lockstat` provider in DTrace
- Using `plockstat(1M)` and the `plockstat` provider in DTrace
- Checking tunable parameters with `mdb`

ORACLE

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

8

System Caches and Buses

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Define a cache
- Describe the overall memory hierarchy
- Describe the characteristics of a cache
- Describe the concept of caching and the cost of a cache miss
- Identify the cache problems associated with multiple CPUs
- Identify the cache problems associated with cache design
- Describe a bus
- Describe the `prtdiag` utility
- Diagnose the problems associated with buses

ORACLE

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

Relevance

Discussion: The following questions are relevant to understanding how to monitor system caches:

- What is a cache?
- Why do you use caches?
- Which components of a system benefit from the use of a cache?
- What is the purpose of using a bus?
- What are some examples of system buses?
- What are some examples of peripheral buses?
- Which function of a system is affected if a bus does not operate properly?

ORACLE

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

Agenda

- Cache concepts
- CMT
- Buses

ORACLE

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

Introducing Caches

- Cache is the storage where recently used data is held to save processing time.
- Hardware cache size is controlled by physical factors, such as space available on the host component, and the limits of transistor density.
- Software caches may be sized as a percentage of available main memory, or as a function of system activity.

ORACLE

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

Caches and Buses

- The system can be viewed as a set of caches and buses.
- The cache is used to store data.
- The buses move the data between the levels of cache.
- Ideally, it is advisable to keep the data you are about to use as close to the CPU as possible, and the buses should restrict the flow of data as little as possible.
- Think of the system as a Memory Hierarchy.
- This is not a recent concept—“...constructing a hierarchy of memories, each of which has greater capacity ...but which is less quickly accessible.” —von Neumann and others, 1946.

The Oracle logo, consisting of the word "ORACLE" in white, uppercase, sans-serif font, centered on a red rectangular background.

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

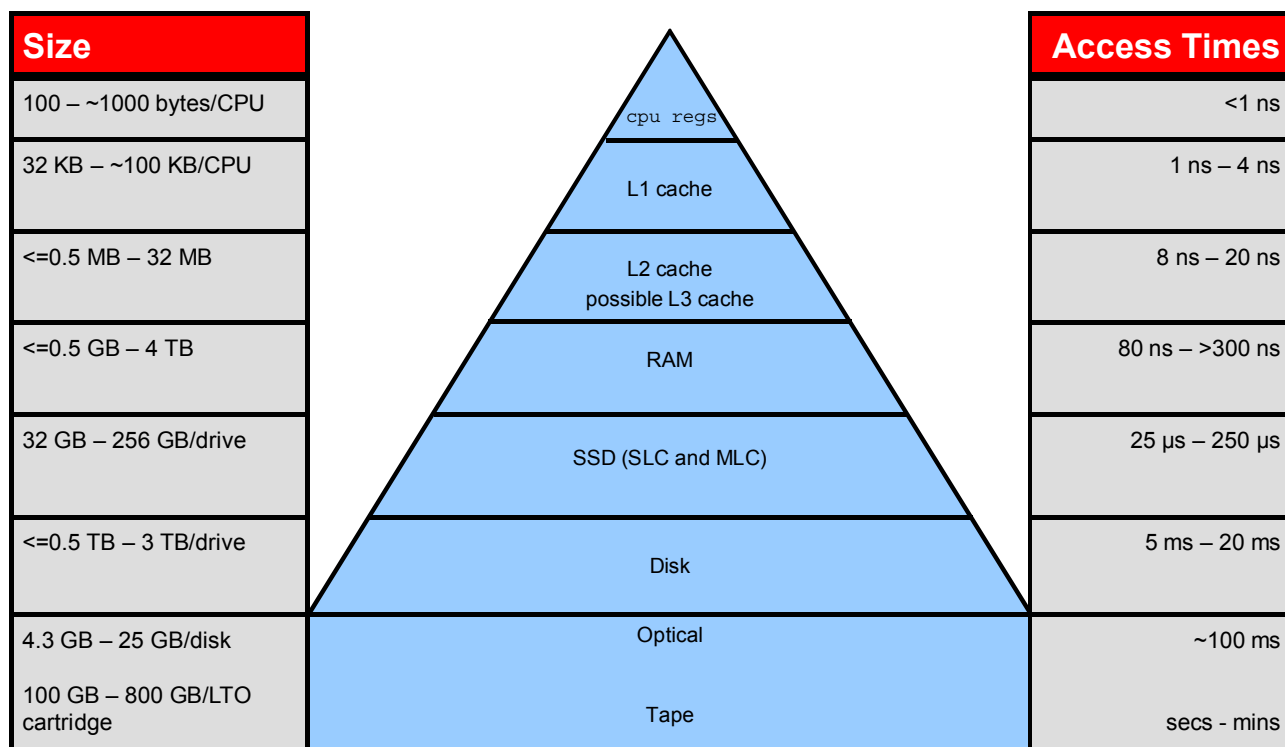
A cache is any hardware or software structure that holds data from a slow or remote storage location. It provides faster access to data. The primary intent of any cache is to keep frequently used data close to its target, thereby reducing the time needed to access it.

Hardware caches on and near the CPU reduce the cost of main memory accesses. As CPU clock speeds have advanced at a much faster rate than memory clock speeds over time, effective caching strategy is an increasingly important aspect of system design.

The I/O subsystem relies on caching strategies to reduce the costs of disk access, including keeping active file system resources in main memory.

The primary measure of a cache's effectiveness is its hit rate, or the percentage of total accesses that are fulfilled by a cache. High hit rates are essential in maximizing application performance, as the cost of a cache miss is typically much higher than the cost of a hit.

Memory Hierarchy



ORACLE

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

The diagram in the slide shows the current memory hierarchy. Although it is constantly evolving, the concepts remain the same. Keep the data you are about to access as close to the CPU as possible. Because you are not sure what will be requested in the future, you must rely on algorithms that approximate that as closely as possible, such as least recently used (LRU), and Adaptive Replacement Cache (ARC) mechanisms.

The figures here are approximate and are just meant to give a reasonable scale.

The other factor that is not shown here is cost. It is hard to put a cost on CPU registers and cache but relatively easy to put a price figure on commodity items such as HDD and SSD and main memory. It is interesting to look at the price per GB of disk versus memory versus optical versus tape, and how quickly that has changed over recent years.

Relative Access Times

Cache Size	Size	Access Time	Human Time Perspective	Managed By
Registers	1 KB	1 cycle	1 second	Compiler
L1 CPU	100 KB	2 cycles	2 seconds	Hardware
L2 CPU	8-16 MB	19 cycles	19 seconds	Hardware
Main memory	128 MB - 512 GB	50 - 300 cycles	50 seconds - 5 minutes	Software
Disk	40 GB to many TB	11 M cycles	4.24 months	Software
Network	No practical limit	80 M cycles	2.57 years	Software

ORACLE

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

System Caches

- Caching takes place at many levels including:
 - The Hardware L1 and L2 cache
 - The translation lookaside buffer (TLB) for caching address translations
 - The translation storage buffer (TSB) for caching TLB entries
 - The metadata buffer cache for UFS
 - The directory name lookup cache (DNLC) for caching directory lookups
 - The `segmap` page cache for file system pages
 - The Adaptive Replacement Cache (ARC), for ZFS
 - The Slab Allocator for caching kernel structures
 - `segkp_cache` for caching kernel stacks
 - Disk controller caches and so on

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered on a red rectangular background.

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

These represent only a fraction of the buffering or caching mechanism in the kernel, and there are other layers used by application code such as databases, the Java Virtual Machine (JVM), web-based applications, graphics, and more.

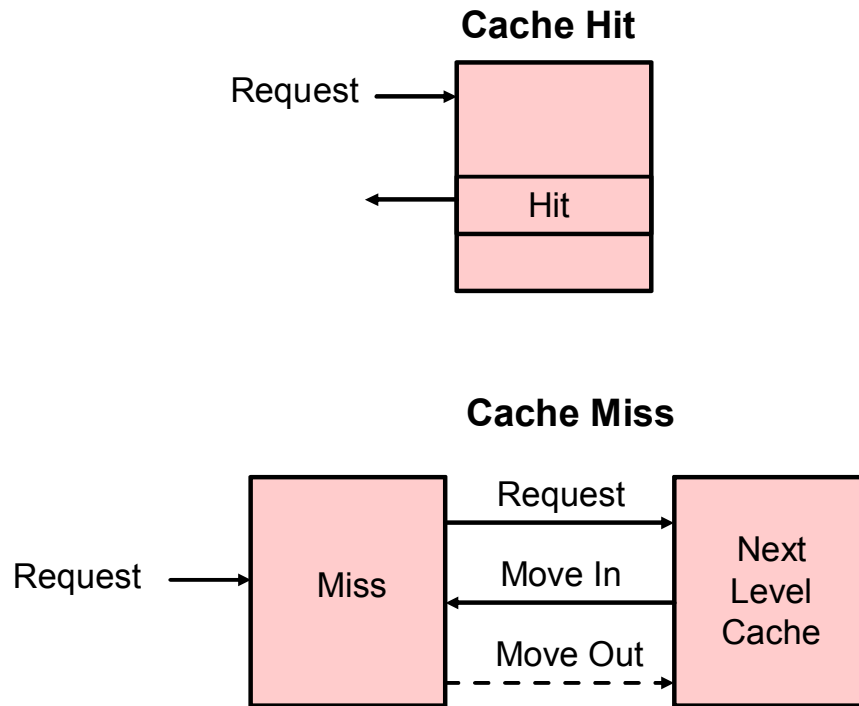
System Caches

- “Computer Science has only three ideas: cache, hash, trash.” – Greg Ganger, CMU
 - This section covers the hardware-managed caches.
 - Cache management techniques are used throughout the system.
 - The basic concepts are similar.
 - The cost of a cache miss is high.

ORACLE

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

Cache Operation



ORACLE

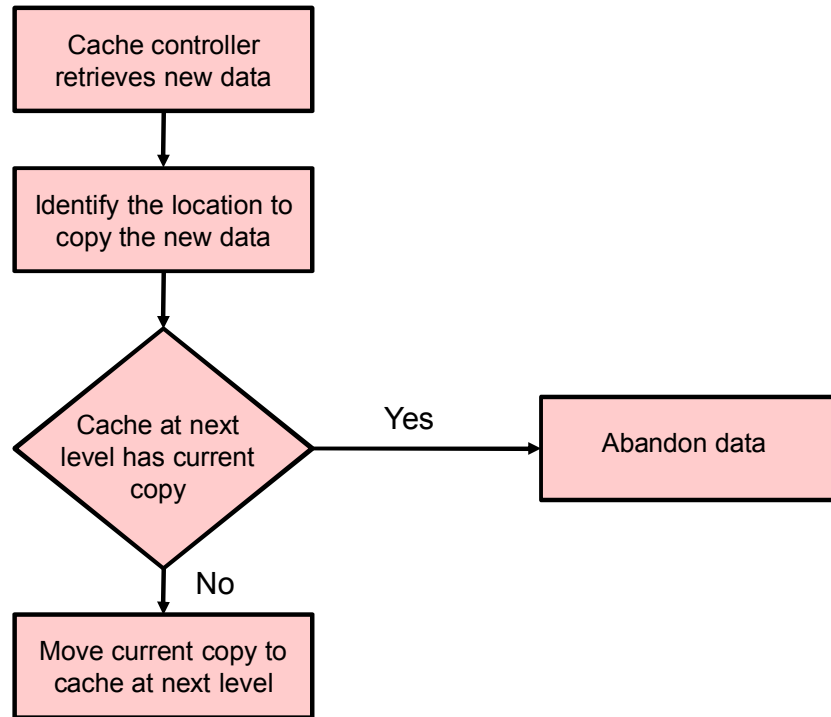
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The figure in the slide shows a logical diagram of hit and miss cache operations.

The value of a cache depends on knowing whether the benefits of using one outweigh the costs. In starting up a cache, for example, there is always additional overhead associated with populating the space with entries.

If the cache controller locates the requested datum in cache, the controller returns it to the requester. This event is called a *cache hit*. Note that the figure in the slide does not reflect the time disparity of trips between a request and the closest cache, and the closest cache to the next level cache.

Replacing Cache Data

**ORACLE**

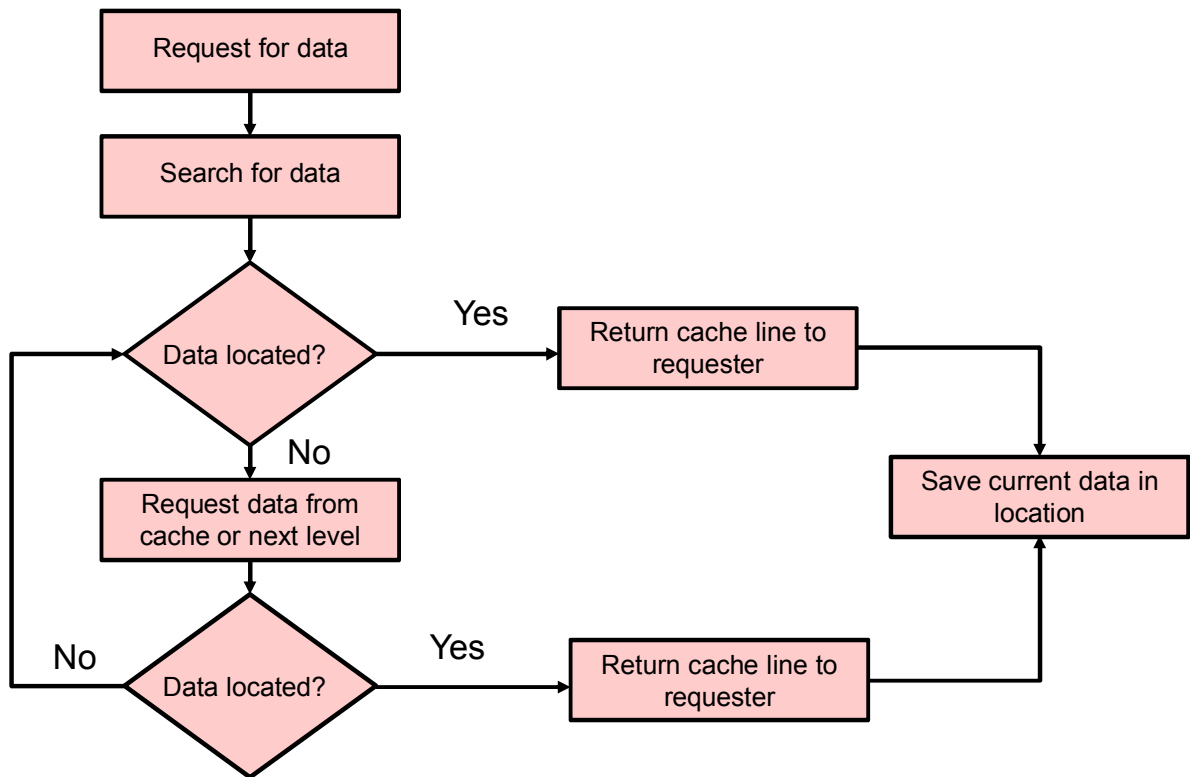
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A *cache miss* occurs if the cache controller cannot supply the datum from cache. In this case, the datum must first be retrieved from a higher (that is, slower) memory level, stored in cache, and then supplied to the requester. Because the wait for a response is relatively long, the CPU may execute other commands in the meantime. This wait time depends on the difference in time cost between a cache access and main memory access. For example, on a 1-GHz CPU, with a cache access time of 1 nanosecond and memory that supports a 50-nanosecond access time, the cost of a cache miss is approximately 50 times more expensive than a cache hit.

When a cache miss occurs, however, the new datum that is retrieved from the memory must be *moved* in to the cache. If the cache is full, an existing entry must first be chosen for *eviction*. This prerequisite step is determined by a heuristic decision called a *replacement policy*. Most common replacement policies are some form of LRU rule.

One of two removal actions is possible in this event. If the cached data's content matches the original content in main memory, it can simply be *abandoned*, adding no significant overhead to the move-in operation. However, if the cache copy has been modified, it must first be *moved out*, or *flushed*, adding to the overall cost of the cache replacement.

Requesting Data from a Cache



ORACLE

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

When an instruction executed on the CPU requests a datum, the *cache controller* generates a *tag* that specifies a location in cache, known as a *cache line* or *block*. A datum's memory address, in turn, is used to calculate an *index* value. If the datum requested currently resides in cache, the generated tag matches it. If the tag matches the index value, the cache controller determines that the entry is present in cache, and returns it to the requester.

If the requested datum is not in the cache, the cache must locate the datum in main memory. After it is located, the existing datum at the given tag location is either abandoned or flushed, and the new data is moved in.

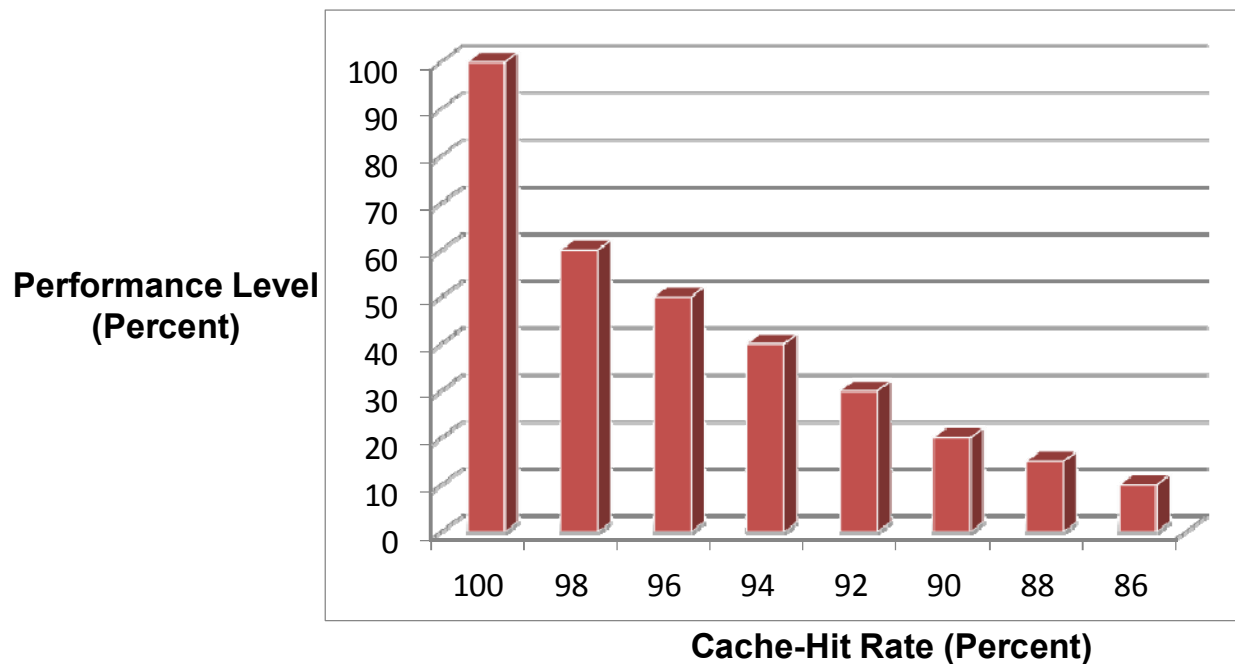
Factors Affecting the Cache-Hit Rate

Factors Affecting Cache-Hit Rate	Description
Cache size	The capacity for more entries in cache can promote cache-hit rate if the most commonly needed entries exceed the current cache size.
Cache line or block size	The data space allotted per entry. A cache line should exceed the size of most requests.
Locality of data	“How close” the next datum reference is to the current one. Locality can be defined as <i>spatial</i> (“next in location”) or <i>temporal</i> (“next in time”).
Cache architecture	How efficiently the structure promotes searching, locating, and moving cache entries.

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

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

Effects of CPU Cache Misses

**ORACLE**

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

CPU cache misses increase process wait time by some multiple of cache access time. For example, if cache access time takes two compute cycles, and a miss (slower memory access) costs 50 cycles, one miss per 100 accesses adds 48 cycles to the cost. The difference in costs between a cache hit and cache miss determines the “degradation” of performance if the hit rate declines.

In the graph in the slide, where a 98-percent cache hit rate results in a 60-percent performance level, the difference in time cost between a cache hit and a cache miss is approximately 33.

CPU cache miss costs propagate through the system as:

- Increased time needed to retrieve data
- Increased process response time
- Decreased system throughput

Oracle CPU Caches

For larger systems such as the Sun SPARC Enterprise M9000 Server:

- SPARC64 VI:
 - 6 MB Shared Level 2 on Chip Cache per Processor
 - 128 KB I-Cache Level 1 per Core
 - 128 KB D-Cache Level 1 per Core
- SPARC64 VII:
 - 6 MB Shared Level 2 on Chip Cache per Processor
 - 64 KB I-Cache Level 1 per Core
 - 64 KB D-Cache Level 1 per Core
- Cache sizes vary with system. However, certain utilities can report them. For instance, `prtdiag` reports cache sizes on SPARC processors and certain Intel/AMD ones.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is positioned on the right side of a solid red horizontal bar.

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

Agenda

- Cache concepts
- CMT
- Buses

ORACLE

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

UltraSPARC T-Series Processor Family

Model	Speed (MHz)	Total Threads*	L1 Dcache (KB)	L1 Icache (KB)	L2 Cache (KB)	L3 Cache (KB)	Year
T1	1000-1400	4×8=32	8	16	3072	N/A	2005
T2	1000-1600	8×8=64	8	16	4096	N/A	2007
T3	1650	8×16=128	8	16	6144	N/A	2010
T4	2850-3000	8×8=64	16x8	16x8	128x8	4096	2011

* Threads per core × number of cores

ORACLE

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

CMT Architecture

In the T-series processors, each hardware thread is treated as a logical CPU.

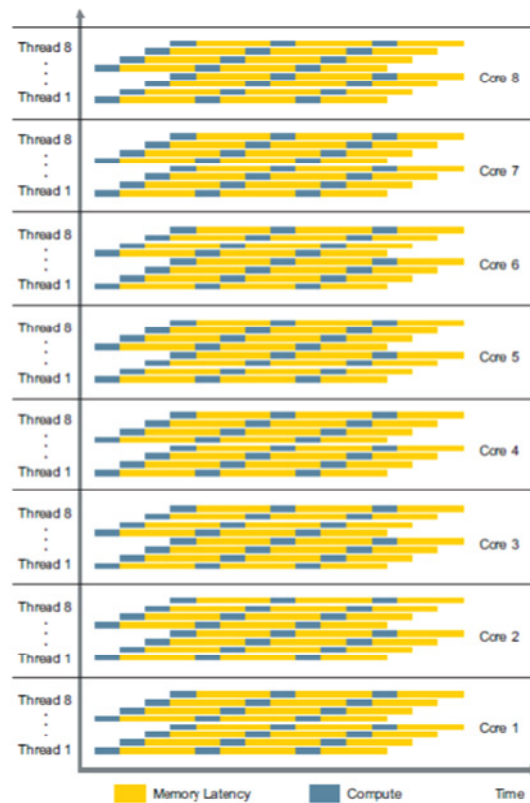
- A hardware thread or strand is essentially a set of CPU registers that define the current state of an executing kernel thread.
- The OS will schedule on each of the hardware threads and let the chip handle the low-level thread switching in the hardware.
- The hardware or firmware will round-robin the hardware strands over a shared pipeline and skip any strands if they are executing a long latency operation such as a load.
- This is be done at each CPU clock cycle.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

Processors 0 through 7 correspond to the hardware threads on the first core; (logical) processors 7 through 15 correspond to the hardware threads on the second core; and so on.

CMT Thread Model

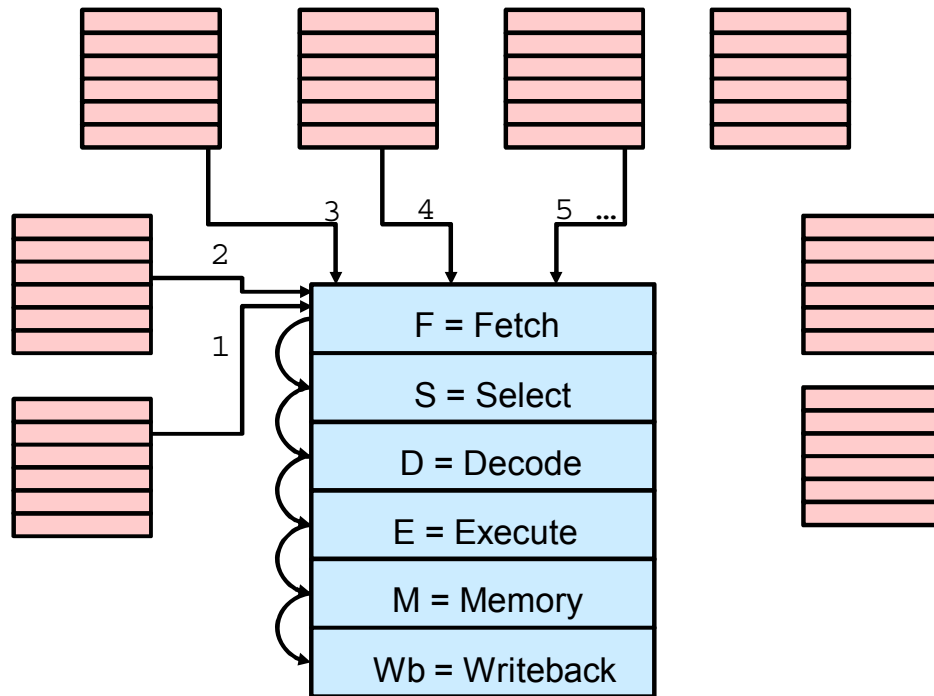


ORACLE

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

Each core in a CMT processor is designed to switch between multiple threads on each clock cycle. As a result, the processor's execution pipeline remains active doing real useful work, even as memory operations for stalled threads continue in parallel. This slide shows an eight core T-series processor supporting up to 64 threads, with up to two threads running in each core simultaneously.

CMT Pipeline



ORACLE

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

Keeping it simple, that is, just one pipeline and all strands are active on every cycle, you can try to execute a different hardware strand.

The shared pipeline consists of F/S/D/E//M/Wb stages:

- F=fetch
- S=select (switch)
- D=decode
- E=execute
- M=memory
- Wb= writeback

CMT Pipeline

1. Select a hardware strand and fetch an instruction and store it in the F stage of the pipeline.
2. Then select another hardware strand (round-robin) and fetch an instruction and store it in the F stage of the same pipeline as you migrate what was in the F stage prior into the D stage.
3. Then select another hardware strand and fetch an instruction and store it in the F stage of the same pipeline and migrate what was in the F stage before into the D stage, and what was in the D stage goes into the E stage, and so on.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a solid red rectangular background.

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

Actually, fetch two instructions and store them in the f Buffer for the hardware strand id. This happens in one cycle. Select and Fetch happen together effectively.

However, certain operations are treated as long latency operations (for example, loads and branches). Therefore, if a hardware strand is stalled, for example, on a load from memory (which has a wait of three cycles), it is assumed that the data is in the L2, so it will take three cycles to fetch the data from the L2. Then, for the next three cycles, this hardware strand will be skipped. Other strands would be selected if they are active and not stalled on other long latency instructions.

For the hardware strand performing the load, if the data is not in the L2, the pipeline is stalled until you get the data from memory. This is a cache miss.

In addition, there are other things that can trigger pipeline stalls; the most notable ones are traps, ASI/BIST accesses, and potentially resource conflicts like crossbar saturation as well as the cache misses mentioned previously.

- **ASI:** Address Space Identifier
- **BIST:** Abbreviation for Built-In Self-Test

Performance Issues and CMT

- The Oracle Solaris 11 OS has been optimized, through the fine granularity of locking, to scale to a large number of processors.
- The OS is aware of the relationship between logical CPUs and physical resources. It spreads the load across physical cores evenly.
- Multiple, single-threaded processes can also take advantage of CMT.
- Idle hardware threads will park rather than run the traditional idle (kernel) thread.
- Core is not really idle until all of its hardware threads are idle.

ORACLE

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

Performance Issues and CMT

Good parallel programming becomes even more relevant:

- Avoid coarse-grained locks whenever possible and keep critical sections short.
- Allow ways to adjust the number of threads or processes spawned.
- When deploying applications on a CMT processor, it is a best practice to configure them to use a large enough number of active LWPs.

ORACLE

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

Performance Issues and CMT

- Critical threads optimization feature:
 - Allows performance critical threads to execute with more exclusive access to hardware resources
 - A user or programmer can adjust the Oracle Solaris scheduler to recognize a *critical thread*.
 - By raising its priority to 60 or above
 - By using `priocntl` from the command-line or `priocntl` and `priocntlset` system calls to a function.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

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

With the critical threads optimization, Solaris 11 extends the capabilities of scheduling priorities (which thread should run first) to include priority over shared resources (which thread should have more "space"). Now, the scheduler not only runs higher priority threads first but also provides them with more exclusive access to hardware resources if they are available.

Using the critical threads optimization feature, a user or programmer can adjust the Oracle Solaris scheduler to recognize a *critical thread* by means of raising its priority to 60 or above by using either the command-line interface or system calls to a function. If this is done, that thread will run by itself on a single core, garnering all resources of that core for itself. The one condition that would prevent this single thread from executing on a single core is when there are more runnable threads than available CPUs. This limit was put into place to prevent resource starvation to other threads.

pgstat Command

The `pgstat` command displays utilization statistics about Processor Groups (PGs).

```
root@server1:~# pgstat 1 1
```

PG	RELATIONSHIP	HW	SW	CPUS
0	System	-	4.8%	0-23
3	Data_Pipe_to_memory	-	4.8%	0-23
2	Floating_Point_Unit	0.0%	4.8%	0-23
1	Integer_Pipeline	0.1%	0.2%	0-3
4	Integer_Pipeline	0.0%	0.0%	4-7
5	Integer_Pipeline	0.0%	0.0%	8-11
6	Integer_Pipeline	4.1%	3.4%	12-15
7	Integer_Pipeline	4.1%	5.4%	16-19
8	Integer_Pipeline	16.3%	19.7%	20-23

ORACLE

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

The `pgstat` command displays utilization statistics about PGs. A PG is a set of CPUs that are grouped together by a common characteristic.

PGs are used by the operating system to represent CPUs that share performance-relevant hardware, such as execution pipelines, caches, and so forth. These PGs are organized into a hierarchy that models the processor topology of the machine. In this hierarchy, each CPU (strand) has a leaf PG that represents the CPUs that share the most hardware with it. Each successive ancestor of the leaf PG shares progressively less hardware with the CPU until the root PG is reached. The root PG contains all of the CPUs in the system and represents the group of CPUs sharing the least hardware with each other.

pginfo Command

The `pginfo` command displays information about the PG hierarchy, its contents, and its characteristics.

```
root@server1:~ # pginfo -pv
0 (System [system,chip]) CPUs: 0-23
  |-- 3 (Data_Pipe_to_memory [system,chip]) CPUs: 0-23
    |-- 2 (Floating_Point_Unit [system,chip]) CPUs: 0-23
      |-- 1 (Integer_Pipeline [core]) CPUs: 0-3
      |-- 4 (Integer_Pipeline [core]) CPUs: 4-7
      |-- 5 (Integer_Pipeline [core]) CPUs: 8-11
      |-- 6 (Integer_Pipeline [core]) CPUs: 12-15
      |-- 7 (Integer_Pipeline [core]) CPUs: 16-19
      |-- 8 (Integer_Pipeline [core]) CPUs: 20-23
```

ORACLE

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

The `pginfo` command displays information about the PG hierarchy, its contents, and its characteristics. A PG is a set of CPUs that are grouped together by a common characteristic. PGs are used by the operating system to represent the CPUs that share performance-relevant hardware, such as the execution pipelines, caches, and so forth. These PGs are organized into a hierarchy that models the processor topology of the machine. In this hierarchy, each CPU (strand) has a leaf PG that represents the CPUs that share the most hardware with it. Each successive ancestor of the leaf PG shares progressively less hardware with the CPU until the root PG is reached. The root PG contains all of the CPUs in the system and represents the group of CPUs sharing the least hardware with each other.

Agenda

- Cache concepts
- CMT
- Buses

ORACLE

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

System Buses

- Ultra Port Architecture (UPA)
- Gigaplane: The Sun Enterprise server (Ex00) product line
- Gigaplane-XB interconnect: Sun Enterprise 10000 servers
- Sun Fireplane interconnect: UltraSPARC III and IV processors
- Jupiter interconnect: Sun SPARC Enterprise M4000, M5000, M8000, and M9000 servers

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

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

The UPA bus is the first Sun interconnect to separate the data bus (implemented as a crossbar in the E10000) from the memory address bus (one-bus or four-bus schemes). It provides support for up to 64 UltraSPARC I or UltraSPARC II CPUs. Peak aggregate bandwidth can be as high as 12.8 GB/sec.

The Gigaplane-XB bus is an extension of the Gigaplane bus and designed specifically for Sun Enterprise 10000 servers. This bus uses four address buses and scaling up to 64 processors. The peak data bandwidth is rated as 12.8 GB/sec.

A Sun Fireplane bus supports up to 106 UltraSPARC III and IV processors in a system. In addition, it can support up to 18 address buses, allowing a peak aggregate bandwidth of up to 172 GB/sec.

The Jupiter interconnect is implemented as point-to-point connections that use packet-switched technology. All routes are unidirectional, noncontentious paths with multiplexed address, data, and control, plus ECC in each direction. System controllers within the interconnect architecture direct the traffic between local CPUs, memory, I/O subsystems, and interconnect paths. It supports a theoretical Peak System Bandwidth of up to 737 GB/sec on the M9000-64 server.

Peripheral Buses

- Peripheral Component Interconnect (PCI) bus
- Peripheral Component Interconnect eXtended (PCI-X) bus
- Peripheral Component Interconnect Express (PCIe) bus
- Compact PCI (cPCI) bus
- Advanced Technology Attachment (ATA) bus
- Serial Advanced Technology Attachment (SATA) bus
- Small Computer System Interface (SCSI) bus
- Serial Attached SCSI (SAS) bus
- Internet Small Computer System Interface (iSCSI) bus
- InfiniBand

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

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

The PCI bus is a synchronous bus architecture that you use to interconnect components, such as chips and expansion boards. A 33-MHz bus with a 32-bit slot offers a maximum data-transfer rate of 132 MB/sec. Similarly, a 66-MHz bus with a 64-bit slot offers a maximum data-transfer rate of 528 MB/sec. PCIe replaced the older PCI. PCI Express 3.0 is capable of bit rates up to of 8 gigatransfers per second (GT/s). The CompactPCI specification is a superset of PCI. It is intended for systems that require a mechanical form factor and features that PCI does not offer. Hot-swapping bus adapters on a CompactPCI system is a key feature. The ATA bus has several market names, including Integrated Drive Electronics (IDE). In 2003, the ATA standard specified a Serial ATA (SATA) interface and renamed the original interface as Parallel ATA (PATA). The latest SATA-released specification is SATA 3.2. It specifies a maximum transfer rate of 6 Gbits/sec. The latest PATA-released specification is ATA-7, known by the market name Ultra DMA 133. It specifies a maximum transfer rate of 133 MB/sec. SCSI is one of the most common peripheral buses found in enterprise environments. SCSI provides specifications for parallel (up to 640 MB/sec), serial (up to 600 MB/sec), and internet (network-dependent speeds) implementations. InfiniBand is a switched fabric communications link used in high-performance computing and enterprise data centers. The effective unidirectional theoretical throughput for InfiniBand is up to 300 Gbits/sec.

busstat(1M) Command

The `busstat` command provides access to the bus-related performance counters in SPARC-based systems.

```
root@server1:~# busstat -l
Busstat Device(s):
dram0 dram1 dram2 dram3 jbus0 jbc0 imu0 imu1 mmu0 mmu1 tlu0 tlu1 lpu0
lpu1
root@server1:~# busstat -e dram0
pic0
mem_reads
mem_writes
...
pic1
mem_reads
mem_writes
root@server1:~# busstat -a -w dram0,pic0=mem_reads,pic1=mem_writes 10
time dev      event0          pic0          event1          pic1
10   dram0    mem_reads      967840        mem_writes      530040
20   dram0    mem_reads      1849260       mem_writes      1059102
...
```

ORACLE

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

The `busstat` command provides command-line access to the bus-related hardware performance counters in the system. It enables the gathering of systemwide bus performance statistics directly from the system hardware.

prtdiag(1M) Command

The `prtdiag` command displays diagnostic information on `sun4u` and `sun4v` systems. It gives a summary of CPU, I/O, and memory configuration information.

```
# prtdiag
System Configuration: Sun Microsystems   sun4u Sun Fire V440
System clock frequency: 183 MHz
Memory size: 15GB
===== CPUs =====
E$      CPU      CPU
CPU Freq  Size      Implementation      Mask      Status
Location
---  -
```

0	1281 MHz	1MB	SUNW,UltraSPARC-IIIi	2.4	on-line	-
1	1281 MHz	1MB	SUNW,UltraSPARC-IIIi	2.4	on-line	-
2	1281 MHz	1MB	SUNW,UltraSPARC-IIIi	2.4	on-line	-
3	1281 MHz	1MB	SUNW,UltraSPARC-IIIi	2.4	on-line	-

ORACLE

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

The `CPUs` section is the first section in the `prtdiag` utility report. This section describes the CPUs located on the system. Note the following information in the `CPUs` section:

- The number of attached system boards
- The number of CPUs
- The speed of each CPU
- The size of external cache (E\$) on each CPU

Overall system performance cannot be determined solely by CPU clock rate. The system bus speed, CPU clock rate, and CPU placement across the system boards all factor into determining maximum throughput. The `prtdiag` command's output helps set expectations on the raw performance limits of the system.

prtdiag(1M) Command

===== IO Devices =====				
Bus Type	Freq MHz	Slot + Status	Name + Path	Model
pci okay	66	MB	pci108e,abba (network) /pci@1c,600000/network@2	SUNW,pci-ce
pci okay	33	MB	isa/su (serial) /pci@1e,600000/isa@7/serial@0,3f8	
pci okay	33	MB	isa/su (serial) /pci@1e,600000/isa@7/serial@0,2e8	
pci okay	33	MB	isa/rmc-comm-rmc_comm (serial+ /pci@1e,600000/isa@7/rmc-comm@0,3e8	
pci okay	33	MB	pci10b9,5229 (ide) /pci@1e,600000/ide@d	
...<output omitted>				

ORACLE

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

The `IO Devices` section displays the name and model number of the interface cards installed on the system.

Information about the interface cards helps you to calculate the *raw bandwidth* for a peripheral bus. The optimum bandwidth is also known as the *requested bandwidth*. A peripheral bus provides optimum throughput when it receives the requested bandwidth. Therefore, you determine the requested bandwidth for a bus to further determine whether any I/O bottlenecks are present on the system.

prtdiag(1M) Command

```

===== Memory Configuration =====
Segment Table:
-----
Base Address          Size          Interleave Factor  Contains
-----
0x0                   4GB          16                BankIDs
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
0x1000000000         4GB          16                BankIDs
16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
0x2000000000         4GB          16                BankIDs
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
0x3000000000         2GB          2                 BankIDs 48,49
0x3200000000         1GB          1                 BankIDs 50
Bank Table:
-----
Physical Location
ID      ControllerID  GroupID  Size      Interleave Way
-----
0        0              0        256MB
...

```

ORACLE

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

This is a sample output of the Memory Configuration section in the `prtdiag` utility report. It shows extensive information about memory size, layout, and addressing. This data gives the system administrator a logical means for confirming the hardware complement, as well as a means for tying the configuration of the system back to the physical devices that support it.

Memory interleaving is the process of laying data across a group of memory banks to increase the data-transfer rate. This technique is similar to striping data across disks.

For example, assume a system has a memory bank that can transfer 8 bytes of data at a time. To write 64 bytes of data to memory, the system requires eight cycles to transfer the data.

If the data transfer can be interleaved between two memory banks, each with a capacity of 8 bytes each, 64 bytes can be transferred in four memory cycles. Similarly, if you use eight memory banks, the same data could be transferred in one cycle.

Diagnosing Bus Problems

- To diagnose problems in a bus, you perform the following:
 - Check the configuration of the bus.
 - Correlate the failures of the bus with its usage.
- The following guidelines help you to avoid problems in buses:
 - Limit devices that are connected to each peripheral bus.
 - Leave empty slots on peripheral buses to improve processing throughput.
 - Monitor system problems and identify faulty buses.
 - Configure a bus or size a cache, based on the changing system configuration.

The Oracle logo, consisting of the word "ORACLE" in white, uppercase, sans-serif font, centered on a red rectangular background.

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

The output from the `prtdiag` utility offers different ways to use the physical limits of the system as a performance ceiling. Because no useful application can take perfect advantage of the system's physical limits, it remains to determine what percentage of the system can be used.

There are several strategies for maximizing performance, such as determining the best adapter-to-device ratio for load balancing. However, if the projected business needs for data transfer capacity reach or exceed the system's physical limits, it is safe to assume that the system itself is ill-suited for the purpose. Refer to the hardware specifications to determine the maximum bandwidth for the hardware and peripheral cards on your system.

You can correlate system failures with system usage to assess the possible cause of a system failure. You can use the diagnostic information provided by the `prtdiag` command to correlate system failure with hardware failure.

For example, the `prtdiag -v` command displays the time of the most recent power failure and other fatal error information about system hardware. You can use this information to correlate a system failure with the usage of hardware at that point in time.

prtconf(1M) Command

The `prtconf` command prints the system configuration information including system peripherals formatted as a device tree.

```
# prtconf
System Configuration:  Sun Microsystems  sun4u
Memory size: 15360 Megabytes
System Peripherals (Software Nodes):
  SUNW,Sun-Fire-V440
    scsi_vhci, instance #0
    packages (driver not attached)
    SUNW,builtin-drivers (driver not attached)
    deblocker (driver not attached)
    disk-label (driver not attached)
    terminal-emulator (driver not attached)
    dropins (driver not attached)
    kbd-translator (driver not attached)
    obp-tftp (driver not attached)
... <output omitted>
```

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

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

The `prtconf` command displays system configuration details that include the memory complement and some device property detail. The `-v` and `-p` options display information related to system hardware.

prtconf(1M) Command

```
# prtconf -v
System Configuration: Sun Microsystems sun4u
Memory size: 15360 Megabytes
System Peripherals (Software Nodes):
  SUNW,Sun-Fire-V440
    System properties:
      name='fm-capable' type=int items=1
      value=00000009
      name='relative-addressing' type=int items=1
      value=00000001
      name='MMU_PAGEOFFSET' type=int items=1
      value=00001fff
      name='MMU_PAGESIZE' type=int items=1
      value=00002000
      name='PAGESIZE' type=int items=1
      value=00002000
    Driver properties:
      name='pm-hardware-state' type=string items=1 dev=none
      value='no-suspend-resume'
... <output omitted>
```

ORACLE

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

The `-v` option shows properties that are applied to the behavior of the listed drivers.

prtconf(1M) Command

```
# prtconf -p
System Configuration:  Sun Microsystems  sun4u
Memory size: 15360 Megabytes
System Peripherals (PROM Nodes):
  Node 'SUNW,Sun-Fire-V440'
    Node 'packages'
    Node 'SUNW,builtin-drivers'
    Node 'deblocker'
    Node 'disk-label'
    Node 'terminal-emulator'
    Node 'dropins'
    Node 'kbd-translator'
    Node 'obp-tftp'
    Node 'SUNW,i2c-ram-device'
    Node 'SUNW,fru-device'
    Node 'ufs-file-system'
    Node 'ufs-file-system'
  Node 'chosen'
  Node 'openprom'
...
```



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

The `-p` option shows properties read from the system PROM.

cpustrack(1M) Command

The `cpustrack` utility monitors one process.

```
# cpustrack -c IC_ref,IC_miss sleep 5
time  lwp  event  pic0  pic1
1.005  1    tick 159622 11995
2.005  1    tick      0      0
3.005  1    tick      0      0
4.005  1    tick      0      0
5.005  1    tick      0      0
5.016  1   exit 165707 12913
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, is centered on a solid red rectangular background.

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

cputrack(1M) Command

```
# cputrack -c Instr_cnt,Cycle_cnt ./dhrystone 9999999&
[4] 4821
loops count set to 9999999
# time lwp event          pic0          pic1
1.089   1   tick 1263180161 1384126014
2.099   1   tick 1148945708 1286445261
3.064   1   tick 1088101936 1232676406
4.069   1   tick 1133399446 1286175419
5.059   1   tick 1113815717 1266130178
6.049   1   tick 1101396136 1265379703
7.039   1   tick 1114694818 1265657166
8.039   1   tick 1127750580 1278226365
Dhrystone(1.1) time for 9999999 passes = 8
This machine benchmarks at 1249999 dhrystones/second
8.108   1   exit 9166392765 10350486228
```



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

Note

- If the `-f` option was specified, `cputrack` would follow any children this process creates through the fork system calls.
- Refer to `man -s 1M cputrack`.

Quiz

The primary measure of a cache's effectiveness is its hit rate or the percentage of total accesses that are fulfilled by a cache. High hit rates are essential to maximizing application performance because the cost of a cache miss is typically much higher than the cost of a hit.

- a. True
- b. False

ORACLE

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

Answer: a

Quiz

The `prtconf` command prints:

- a. Diagnostic information on `sun4u` and `sun4v` systems
- b. A summary of CPU, I/O, and memory configuration information
- c. The system configuration information including system peripherals formatted as a device tree
- d. All of the above

ORACLE

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

Answer: c

Quiz

The `cputrack` utility displays CPU performance information:

- a. On a per-process basis
- b. As a summary of CPU, I/O, and memory configuration information
- c. Because it relates to system configuration information including system peripherals formatted as a device tree
- d. All of the above

ORACLE

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

Answer: a

Summary

In this lesson, you should have learned how to:

- Define a cache
- Describe the overall memory hierarchy
- Describe the characteristics of a cache
- Describe the concept of caching and the cost of a cache miss
- Identify the cache problems associated with multiple CPUs
- Identify the cache problems associated with cache design
- Describe a bus
- Describe the `prtdiag` utility
- Diagnose the problems associated with buses

ORACLE

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

Practice 8 Overview: System Caches and Buses

This practice covers the following topics:

- Calculating the Cache-Hit Rate
- Observing File System Caching
- Calculating the Cycles Per Instruction (CPI)

ORACLE

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

