Архитектура ZFS

Sergey Klyaus
Sergey.Klyaus@Tune-IT.Ru

Версия 2.1

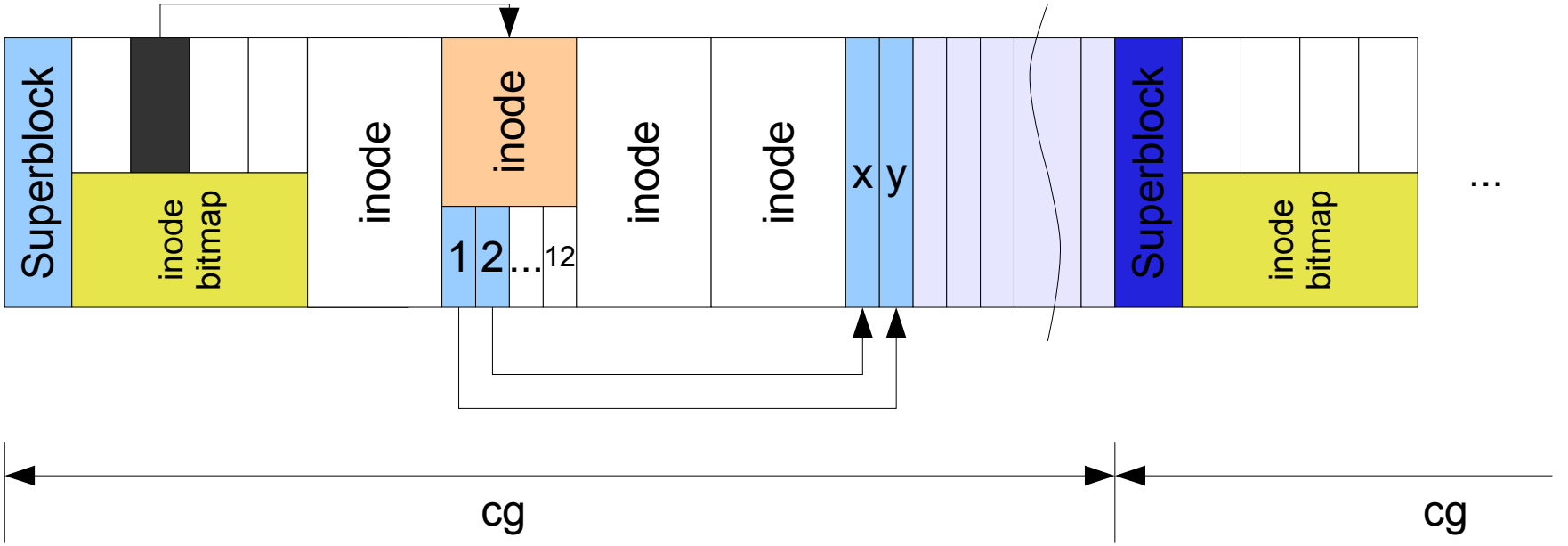
Традиционные ФС (UFS, ext3/4)

- Что такое RAID?
- Что такое том (volume)?
- Что такое раздел (partition)?
- Что такое файловая система (filesystem)?
- Что такое снимок (snapshot) и репликация (replication)?
- Что такое журналирование файловой системы?
- Что такое LBA?

Традиционные ФС (UFS, ext3/4)

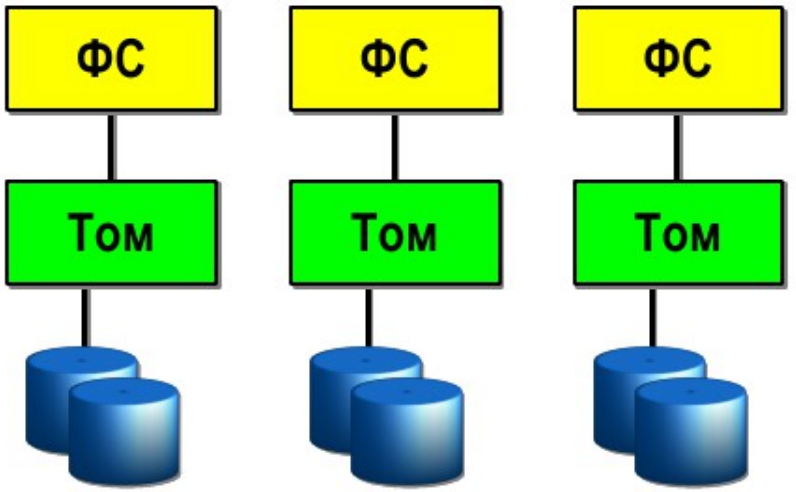
- Диск делится на несколько разделов, каждый имеет собственную файловую систему.
- Аппаратный RAID или менеджер томов создают логический диск из нескольких физических
- Файлы работают с блоками фиксированной длины на диске
  - Ведет к правилу равенства блоков в приложении/файловой системе/менеджеру томов/СХД
- Информация о файле задана внутри inode
- Директория — список пар имя файла-inode

Традиционные ФС (UFS, ext3/4)



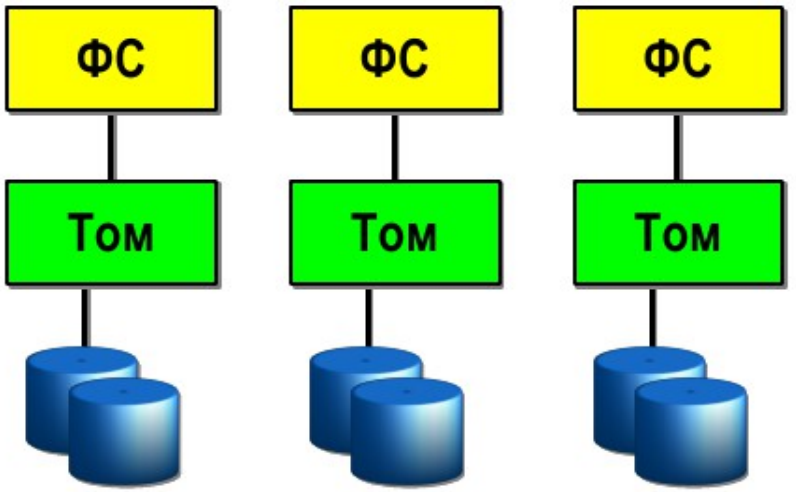
Традиционные ФС (UFS, ext3/4)

- Жесткая иерархия:
  - Том
  - Раздел
  - Группы цилиндров
  - Битовые карты inode, фиксированный блок, списки указателей на блоки данных.
- Нет явной возможности балансировать данные и I/O
- Сложно реализовать снапшоты и репликацию



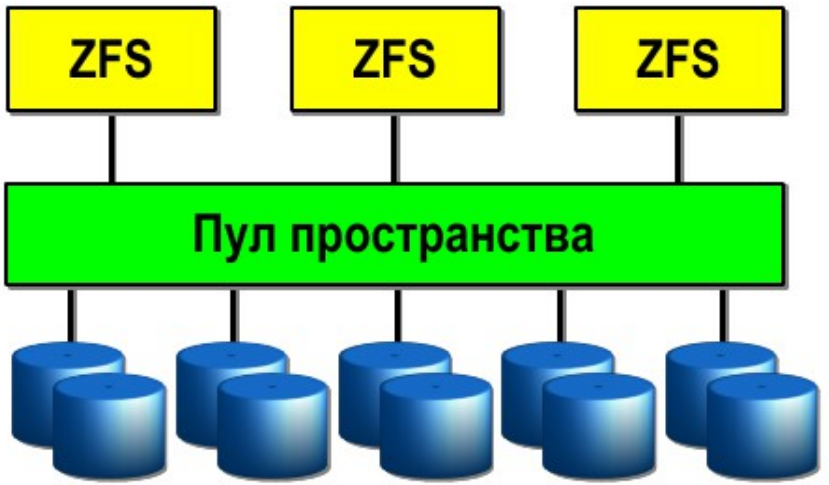
Традиционные ФС (UFS, ext3/4)

- Сложное администрирование
  - Множество инструментов (на уровне дисков, томов, NFS/CIFS)
  - Не переносимы между платформами
- Невысокая надежность
  - Теряют целостность при внезапном отключении устройства без остановки I/O
  - Не отслеживают повреждения данных
  - fsck



ZFS

- Пул пространства
  - Инкапсулирует устройства хранения
  - Организует данные и метаданные в древовидную структуру
- Снапшоты на уровне ФС
- Отстутствие разумных ограничений
- Доказуемая надежность
- Легки в управлении



Делает с дисками тоже самое, что делает виртуальная память с физической

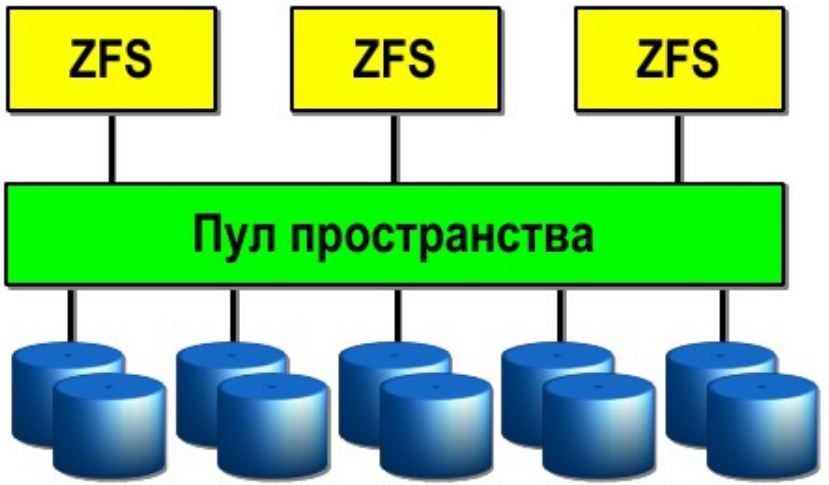
ZFS

Мифы и страхи:

- ARC-кеш — значительные требования к памяти
- Сложный и запутанный формат затрудняет восстановление данных

Недостатки:

- Ограниченная поддержка в операционных системах
- Низкая скорость синхронной записи
- Фрагментация

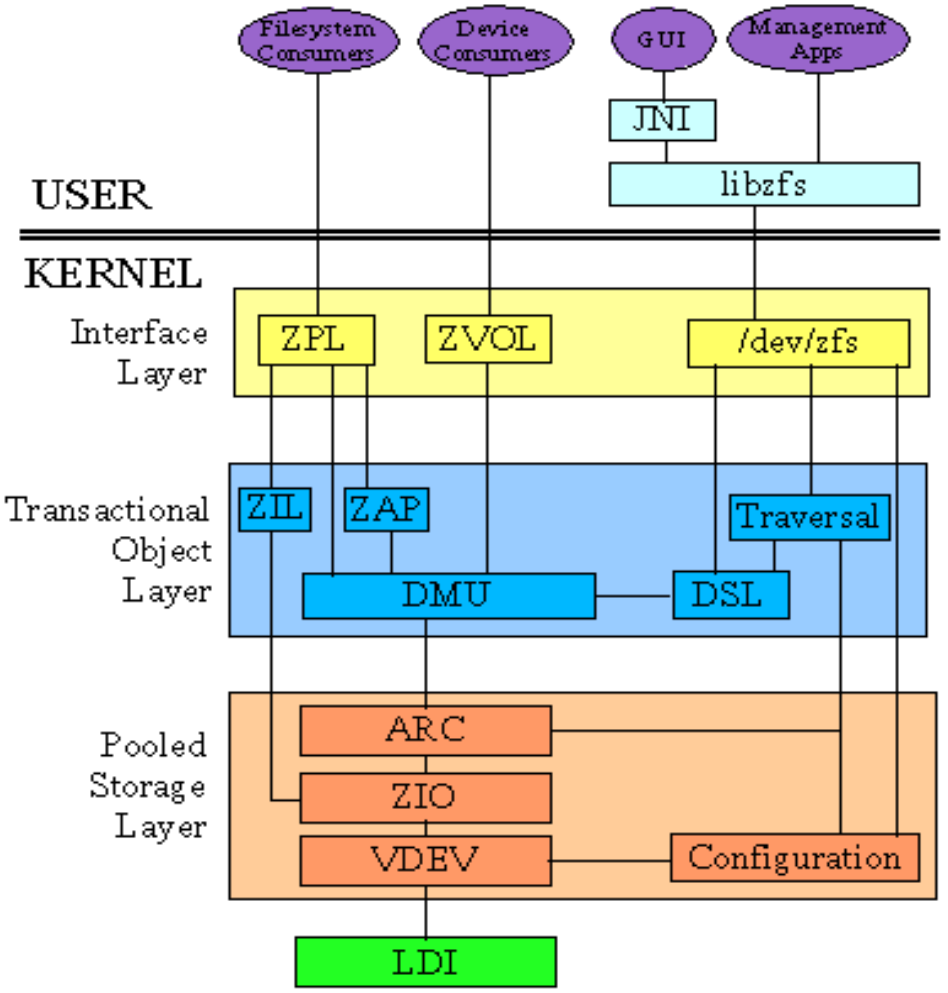


Три кита ZFS

- SPA (Storage Pool Allocator) инкапсулирует блоки в виртуальное устройство (пул)

  По DVA выделяет реальный блок на устройстве

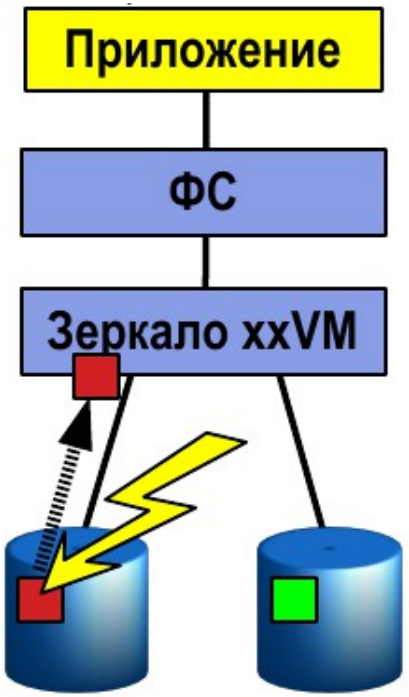
- DMU (Data Management Unit) организует данные и метаданные в виде наборов объектов
- ZPL (ZFS POSIX Level) презентует объекты DMU в виде файлов и директорий VFS
- LDI (Logical Driver Interface) — интерфейс-прослойка между внутренними интерфейсами ядра и ZFS



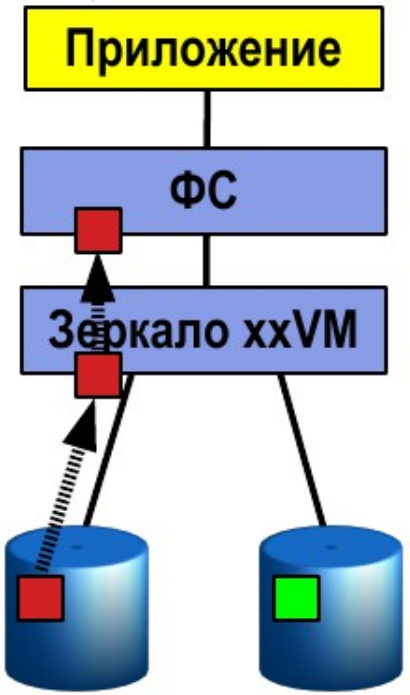
Транзакции в традиционных VM (SVM, VxVM)

- Требуется немедленная и синхронная запись
- Нет возможность выявить сбойный блок в страйпе

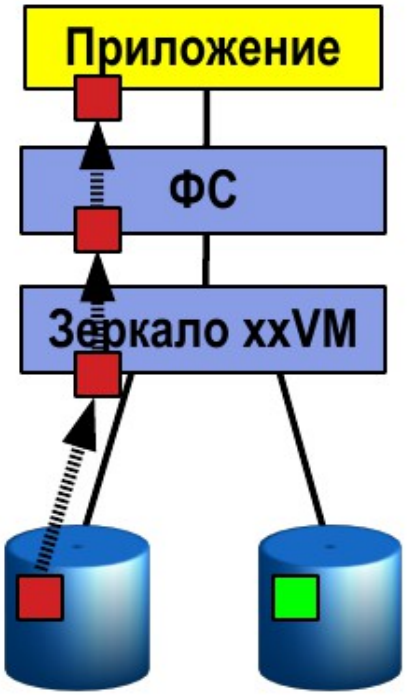
1.



2.



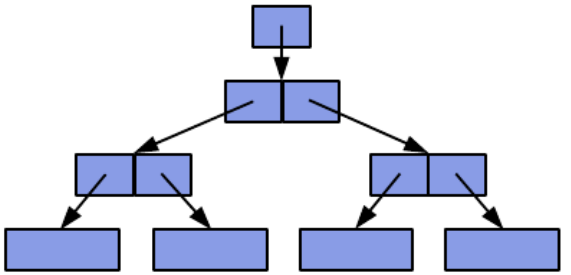
3.



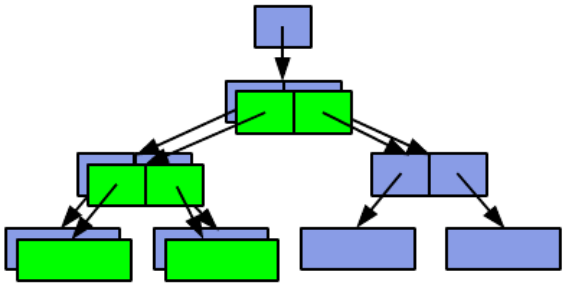
Защита данных в ZFS

- Блоки никогда не стираются, а перезаписываются (CoW)
- После обновления всех блоков требуется только сменить актуальный номер транзакции (практически атомарная операция)
- Все блоки защищены контрольными суммами
- Важные блоки дублируются
- RAID-Z = RAID-5 + Copy on Write

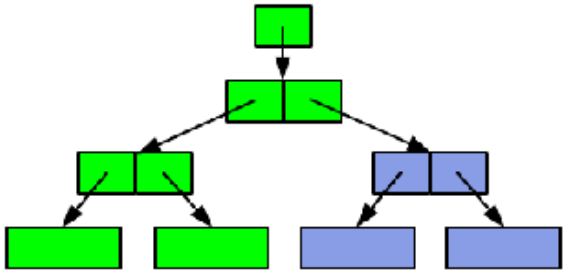
1.



2.

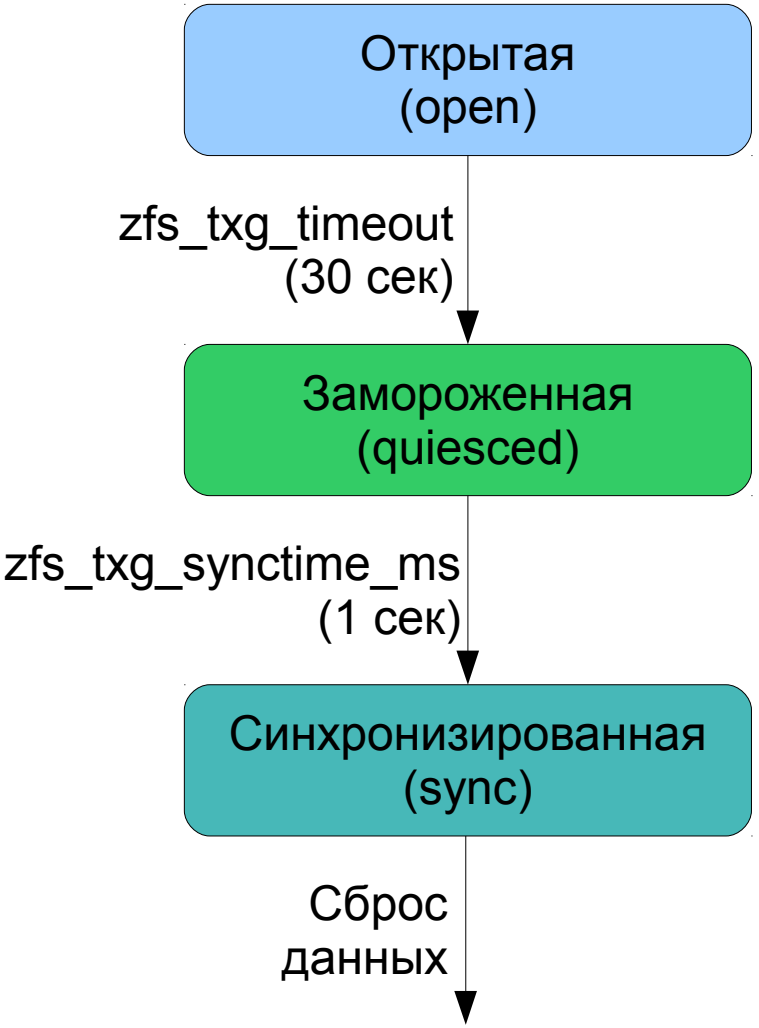


3.



Транзакционная модель ZFS

- Все действия по обновлению данных являются транзакциями (TX)
- Транзакции группируются (TXG)
- TXG записывается целиком или не записывается вовсе
- Нужно, чтобы не так часто обновлять метаданные
- Интенсивность синхронизации определяется переменной zfs_txg_timeout



Полезные ссылки

- ZFS source tour

  http://hub.opensolaris.org/bin/view/Community+Group+zfs/source

- ZFS — The last word in File Systems

  http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/zfslast.pdf

- ZFS under the hood

  http://www.filibeto.org/~aduritz/truetrue/solaris10/zfs-uth_3_v1.1_losug.pdf

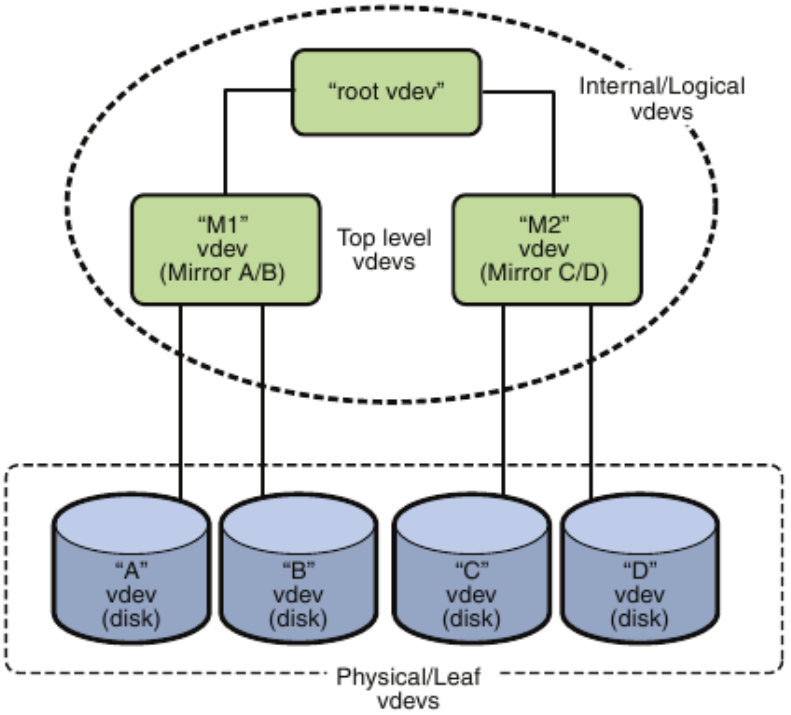
- ZFS on-disk format

  http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/ondiskformat0822.pdf

Storage Pool Allocator

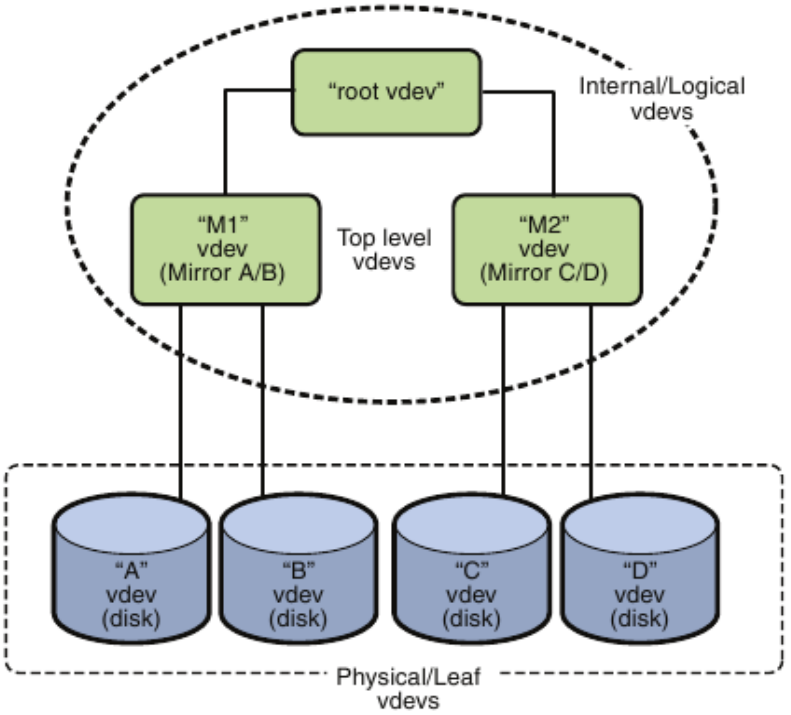
VDEV — Виртуальные устройства

- Реализуют работу с физическими устройствами: физические или листовые vdev
  - Управляют кешем на чтение самих устройств: предвыборка последующих блоков (ZFS prefetch)
  - Планировщик I/O устройства
  - Организация меток на VDEV, содержащих конфигурацию и свойства пула (типа SVM replicas)
  - Включают аллокатор уровня устройства на основе SLAB



VDEV — Виртуальные устройства

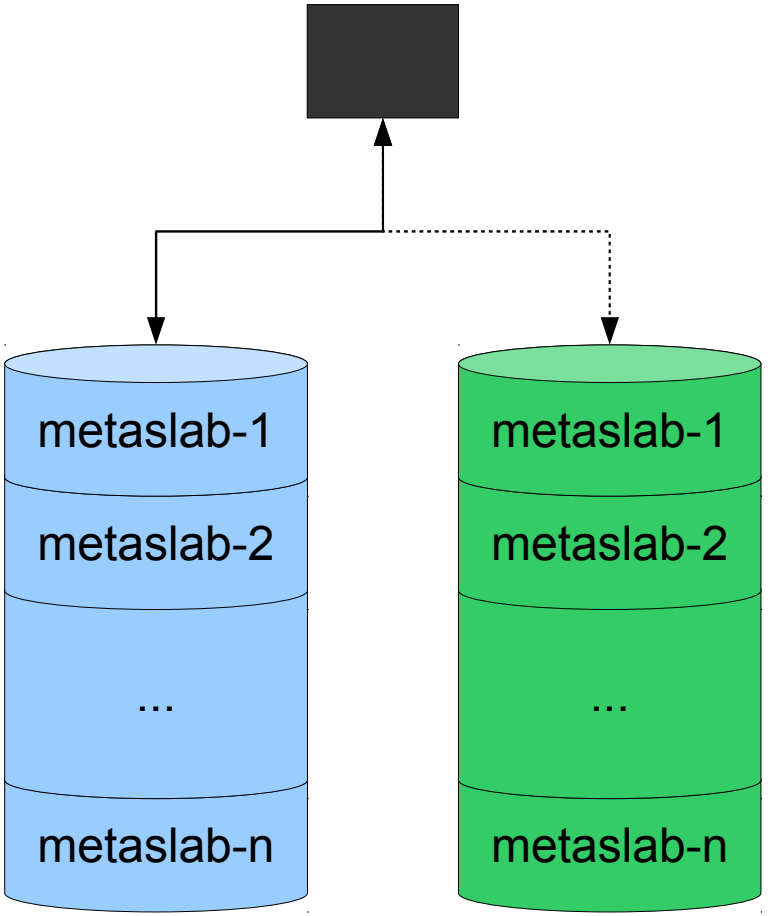
- Реализуют работу с логическими устройствами: RAID
  - Зеркалирование (mirror)
  - Четность (RAID-Z). 1, 2 или 3 диска с четностью.
- Корневое устройство
  - Dynamic Striping - RAID-0, но с балансированием нагрузки
- Устройства горячей подмены
- L2ARC и лог-устройства



Metaslab-аллокатор

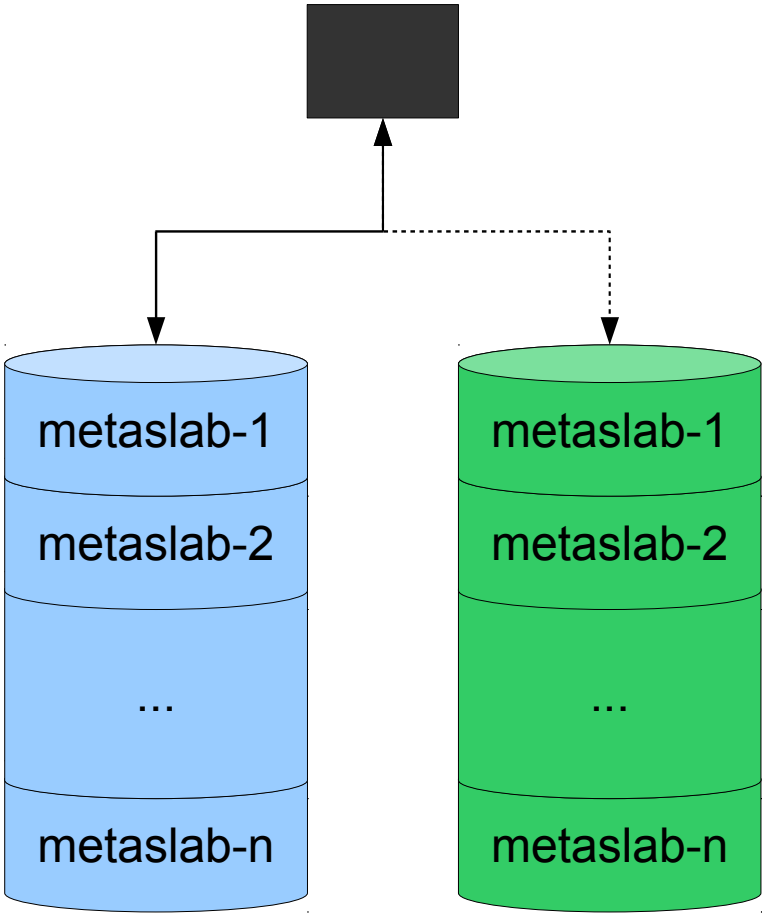
- Каждое физическое устройство разбито на metaslab (~200)
- Свободное место в metaslab организовано через spacemap — логи выделения блоков
- В сумме — AVL-дерево
- Посмотреть (OpenSolaris, Solaris 10 U9):

```
# zdb -m -vvvv tiger
```



Metaslab-аллокатор

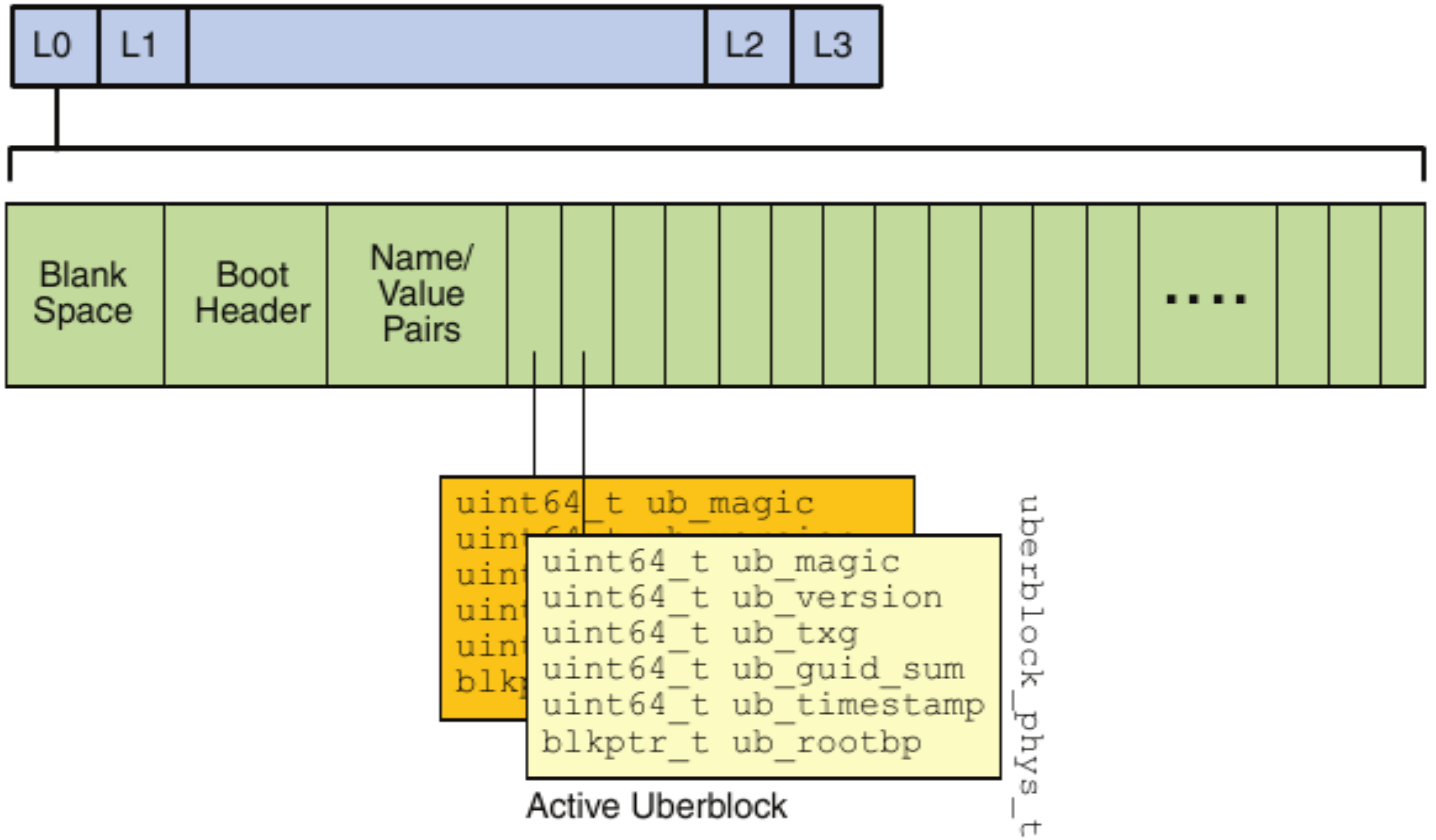
- Dynamic striping — алгоритм выбора устройства:
  - Новые устройства заполняются в первую очередь
  - Для обычных данных — устройство меняется каждые 512 Кб
  - Для служебных — поблочный Round-Robin
- Metaslab выбирается по весу: меньший LBA — большая скорость
- Внутри metaslab — первый попавшийся блок



Метки VDEV

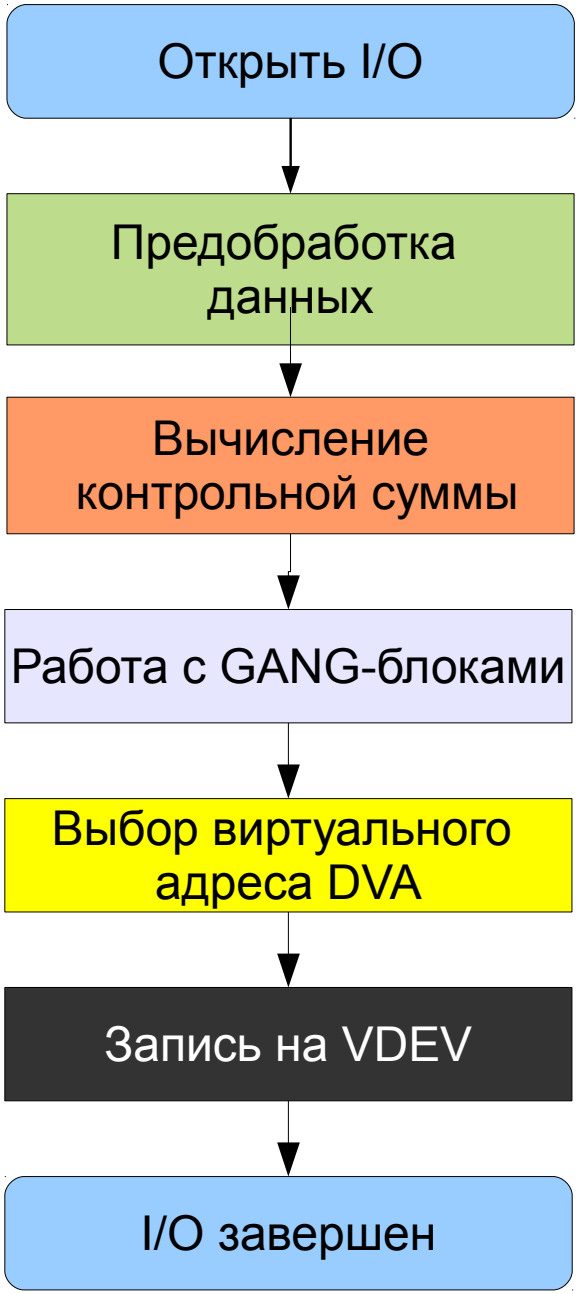
`# zdb -l /dev/dsk/c0t2d0` - метка

`# zdb -u tiger` - уберблок

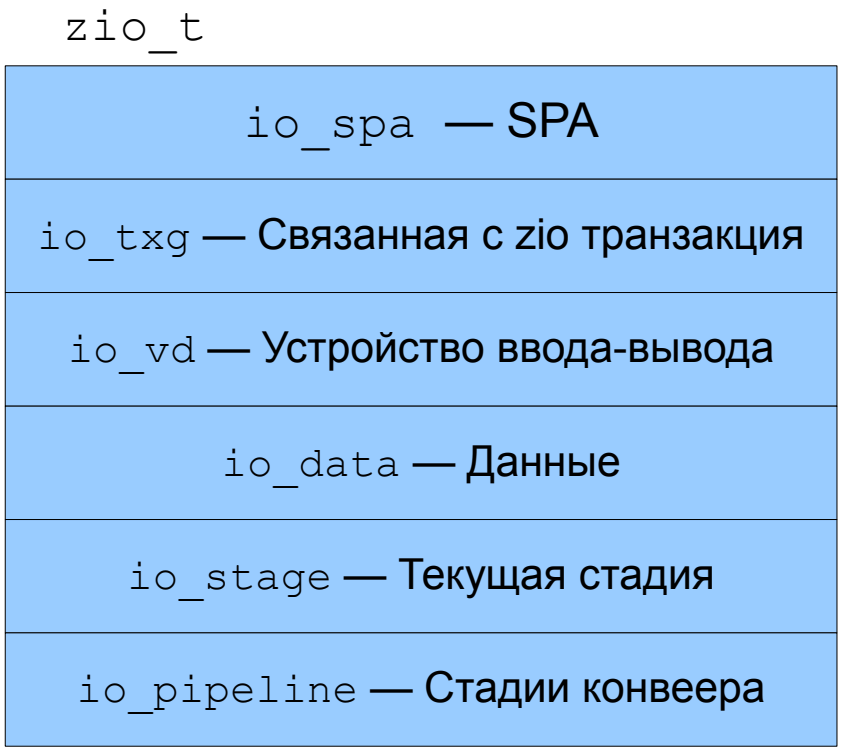


Конвейер ZIO

- 5 типов операций:
  - Read
  - Write
  - Free — освобождение блока по его DVA
  - Claim — подтвердить блок, записанный в качестве zilog, но не в рамках транзакции
  - Ioctl — сброс дисковых кешей



Конвейер ZIO



- zio_create() порождает новую транзакцию на конвеере
  - Для дочерних VDEV — порождаются отдельные транзакции
- Набор taskq диспетчеризует одну из транзакций с помощью zio_taskq_dispatch
- zio_execute выбирает следующую возможную подоперацию и выполняет
- Операции могут блокироваться (например I/O)

Конвейер ZIO

I/O Types	Pipeline Interlock	Compression	Checksum	Gang Blocks	Data Virtual Addressing	Virtual Device I/O
RWFCI	Open					
RWFCI	Wait for children ready					
-W---		Write compress				
-W---			Checksum generate			
-WFC-				Gang pipeline setup		
-WFC-				Get gang header		
-W---				Rewrite gang header		
--F--				Free gang members		
---C-				Claim gang members		
-W---					DVA allocate	
--F--					DVA free	
---C-					DVA claim	
-W---			Gang checksum generate			
RWFCI	Ready					
RW--I						I/O start
RW--I						I/O done
RW--I						I/O assess
RWFCI	Wait for children done					
R----			Checksum verify			
R----				Read gang members		
R----		Read decompress				
RWFCI	Done					

Конфигурация SPA

- Читает конфигурацию пулов при загрузке системы
- Хранит историю команд, выполняемых на пуле

```
# zpool history tiger
```

- Отслеживает ошибки на пуле и передает их FMA
- Создает и удаляет пулы
- Является посредником между ZIO и VDEV

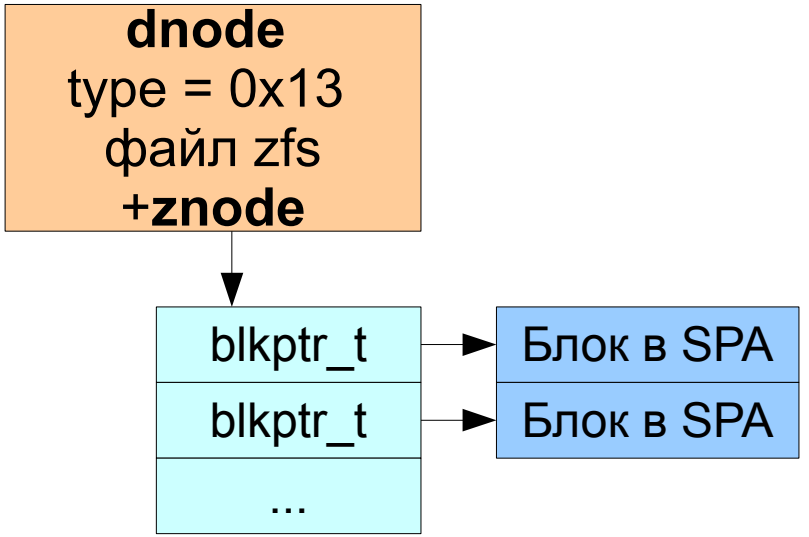
`spa_t`

`spa_name` — Имя пула
`spa_state` — Состояние пула
`spa_*_txg` — Номера транзакций
`spa_is_root` — Является ли корневым
`spa_log_state` — Состояние ZIL
`spa_root_vdev` — Корневой VDEV
`spa_meta_objset` — Meta Object Set

Transactional Object Layer

Модуль управления данными DMU

- Объект:
  - Набор блоков, выделенных в SPA
  - dnode + набор DVA, ссылающихся на блоки
  - Служебные данные (например znode, содержащий имя владельца и дату создания файла)
- dnode + znode = UFS inode

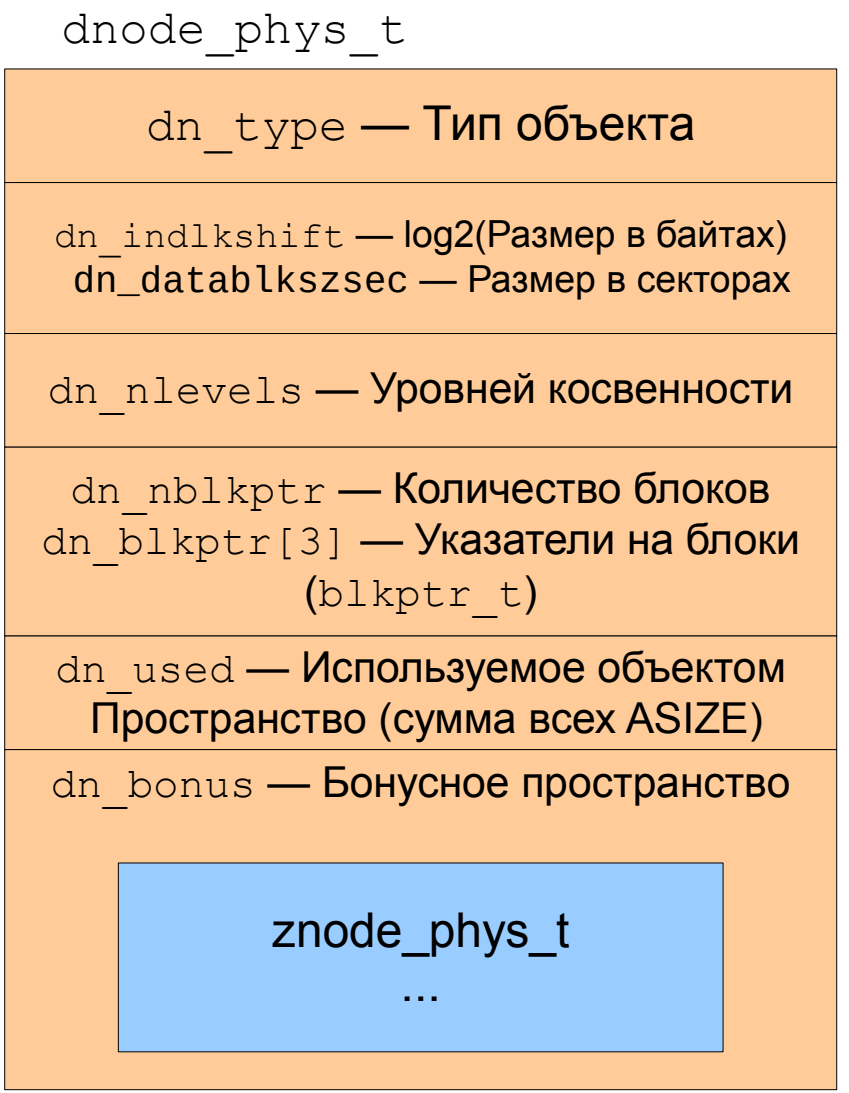


Модуль управления данными DMU

- Каждый объект описывается структурой dnode_phys_t
- Содержит набор указателей на DVA (dn_blkptr)
- Пространство для хранения дополнительной информации по объекту (владелец и время создания для файла ZPL) в области dn_bonus

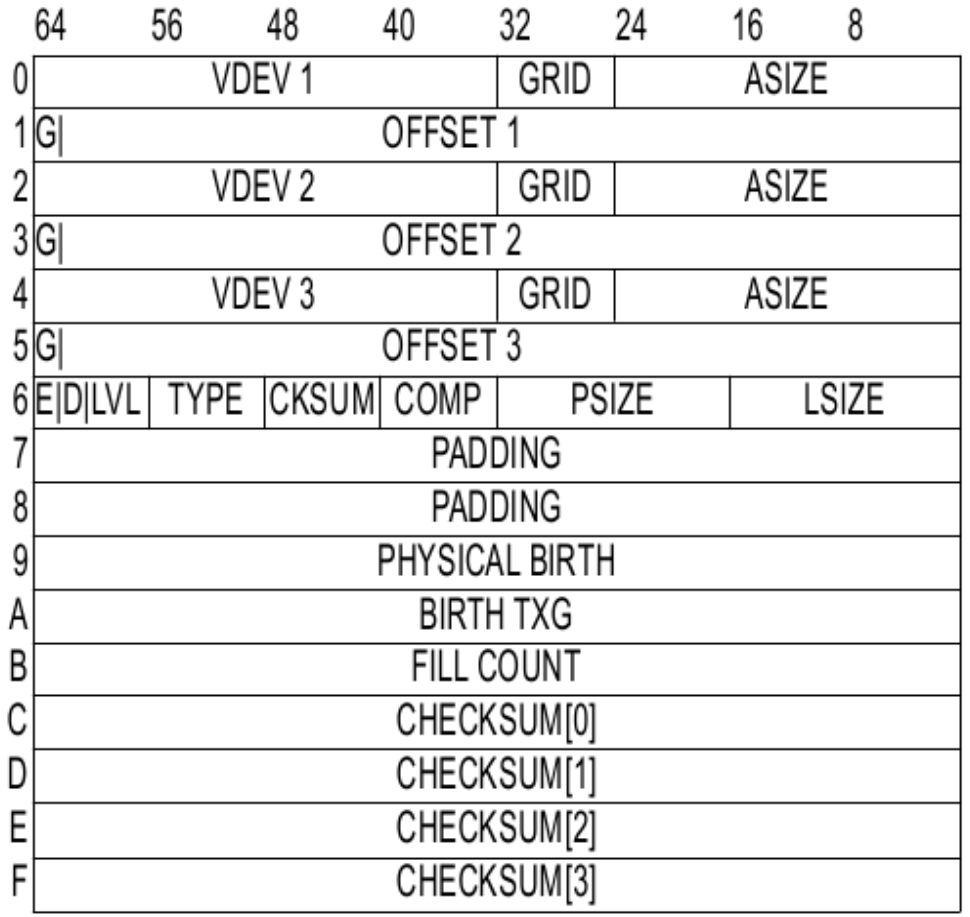
- Вывести список dnode:

```
# zdb -dd tiger
```

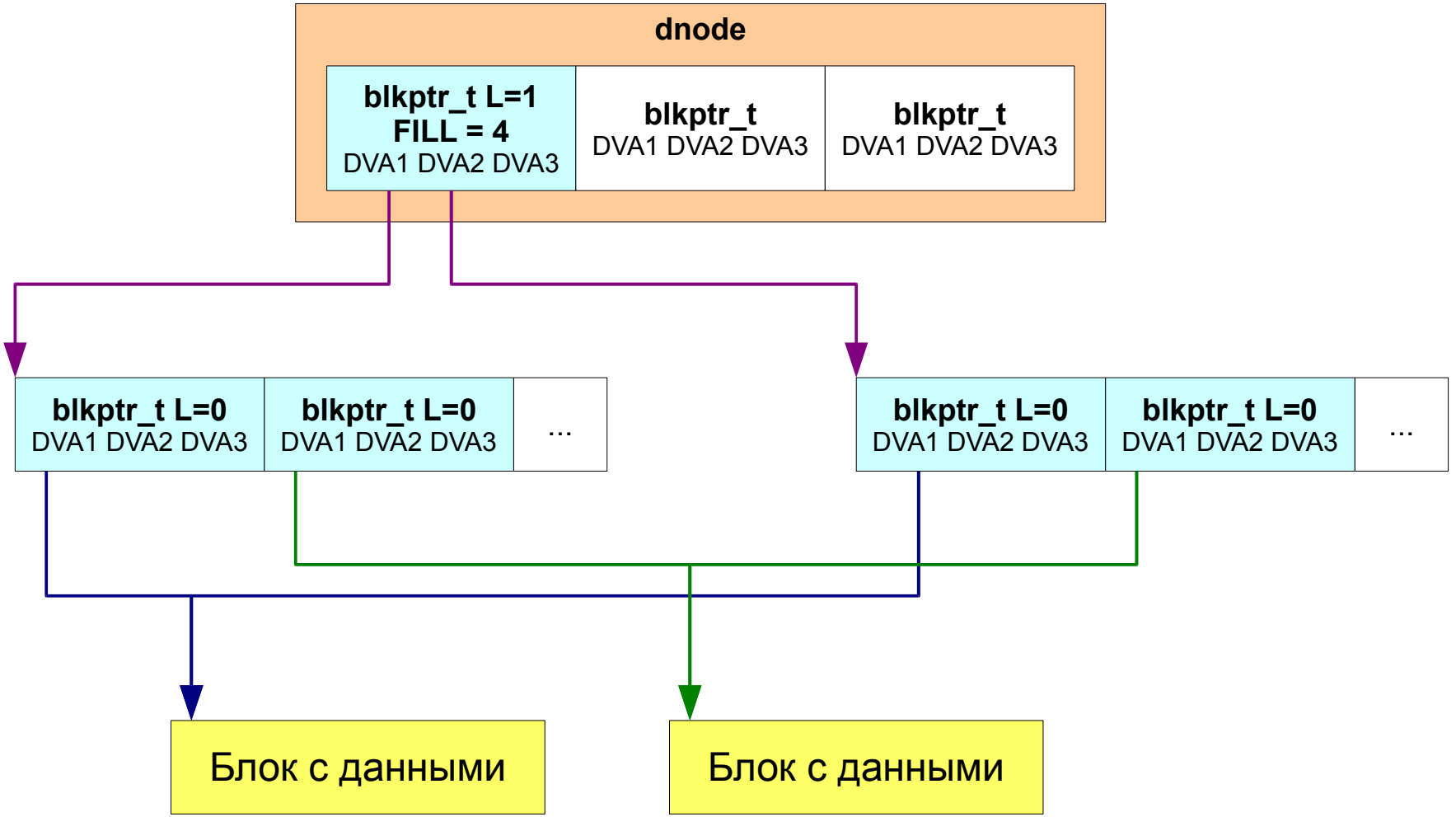


Модуль управления данными DMU

`blkptr_t`



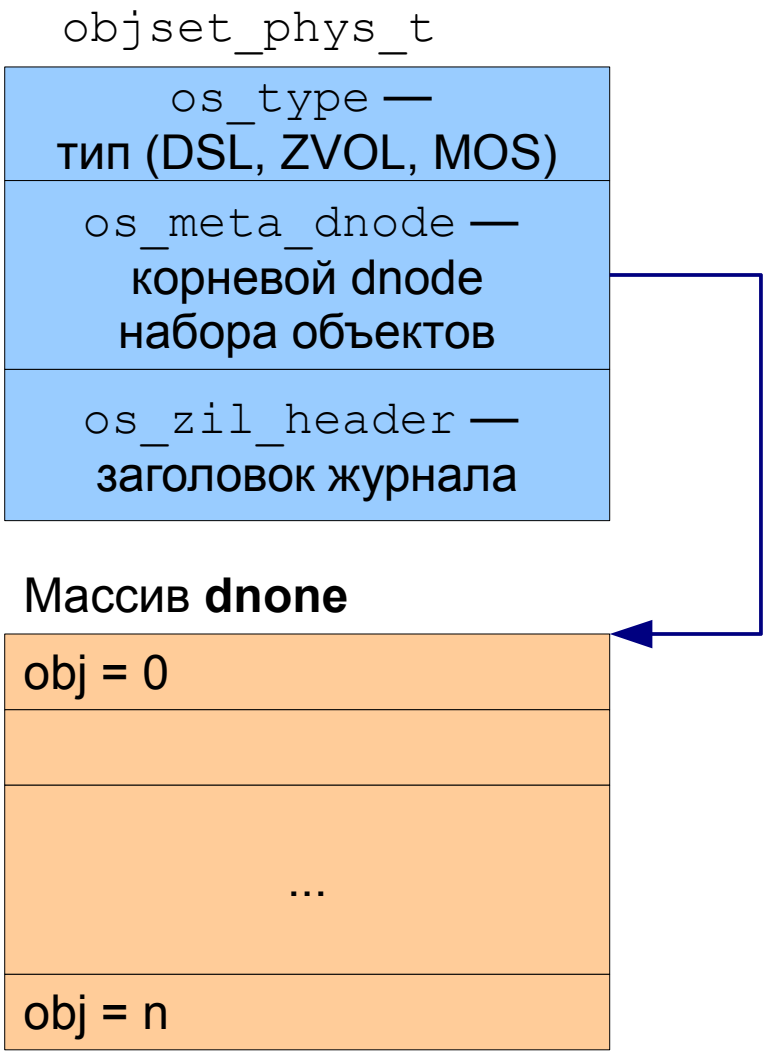
Модуль управления данными DMU



Модуль управления данными DMU

- Objset — набор связанных объектов
- Номер объекта - uint64_t
- Каждый objset это:
  - Файловая система
  - Снимок
  - Клон
  - Том (ZVOL)
  - Meta Object Set

Управляются DSL



Процессор атрибутов ZAP

- Необходимо хранить пары ключ-значение
  - Для организации директорий (имя файла-dnode/znode)
  - Для хранения свойств в DSL
  - Для хранения глобальных настроек пулов
- Два разных алгоритма:
  - MicroZAP — простейшая структура.
    - Если имя каждого атрибута не длиннее 50 символов
    - Все значения имеют тип uint64_t
    - Все пары умещаются в одном логическом блоке (128 Кб)
  - FatZAP — используется во всех остальных случаях
- MicroZAP автоматически преобразуется в FatZAP

Уровень наборов данных и снимков DSL

- Dataset = Object Set + Иерархия
- Директория dsl_dir_phys_t определяют иерархию файловых систем и ZFS volume
- Датасеты dsl_dataset_phys_t определяют сам датасет и его снапшоты

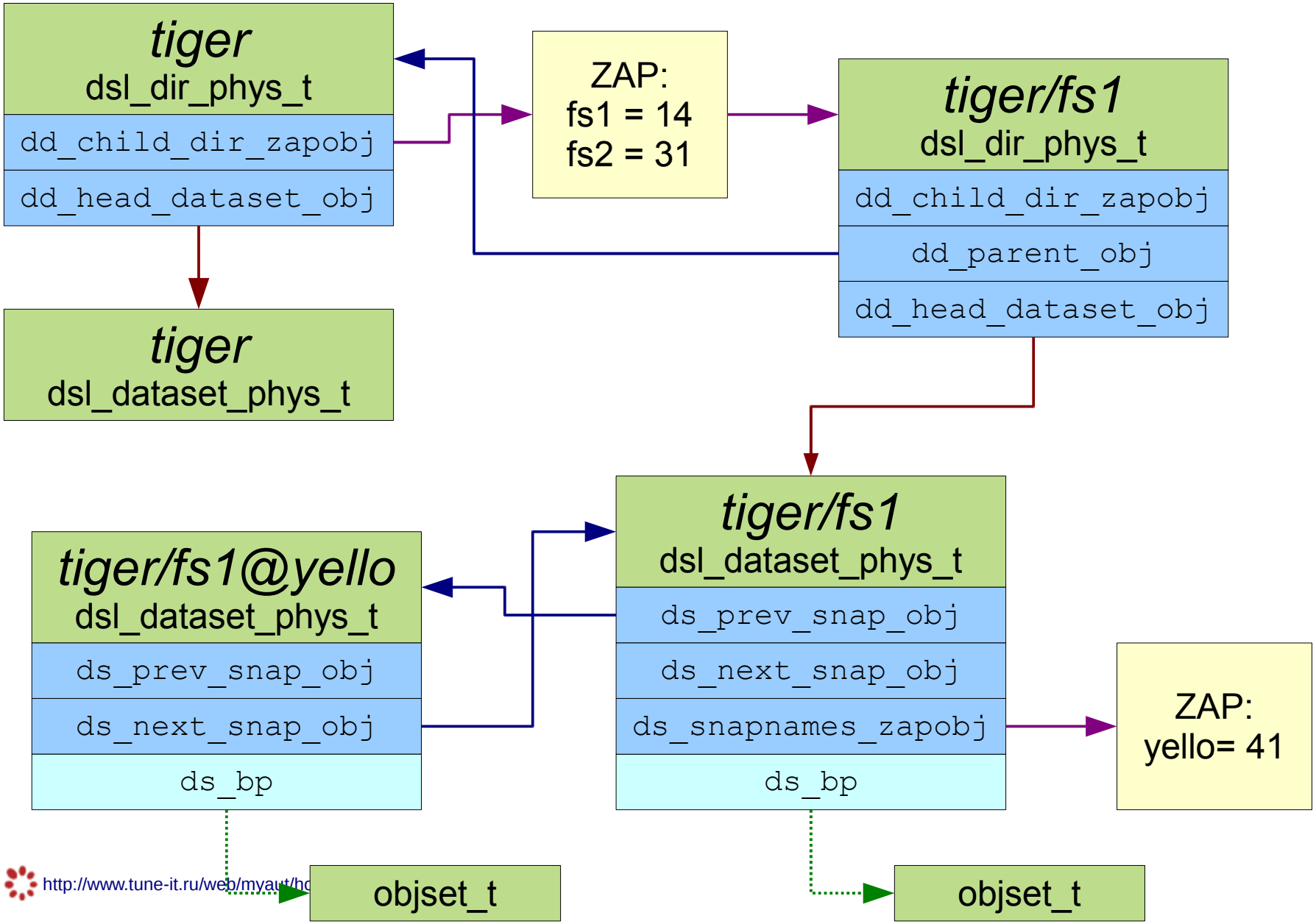
`dsl_dataset_phys_t`

`ds_dir_obj` — Номер объекта родительского каталога
`ds_prev_snap_obj` — предыдущий снапшот `ds_next_snap_obj` — свежий снапшот
`ds_snapnames_zapobj` — Карта соответствий имен и объектов снимков
`ds_creation_time` — Время создания по UTC
`ds_creation_txg` — Номер транзакции рождения набора данных

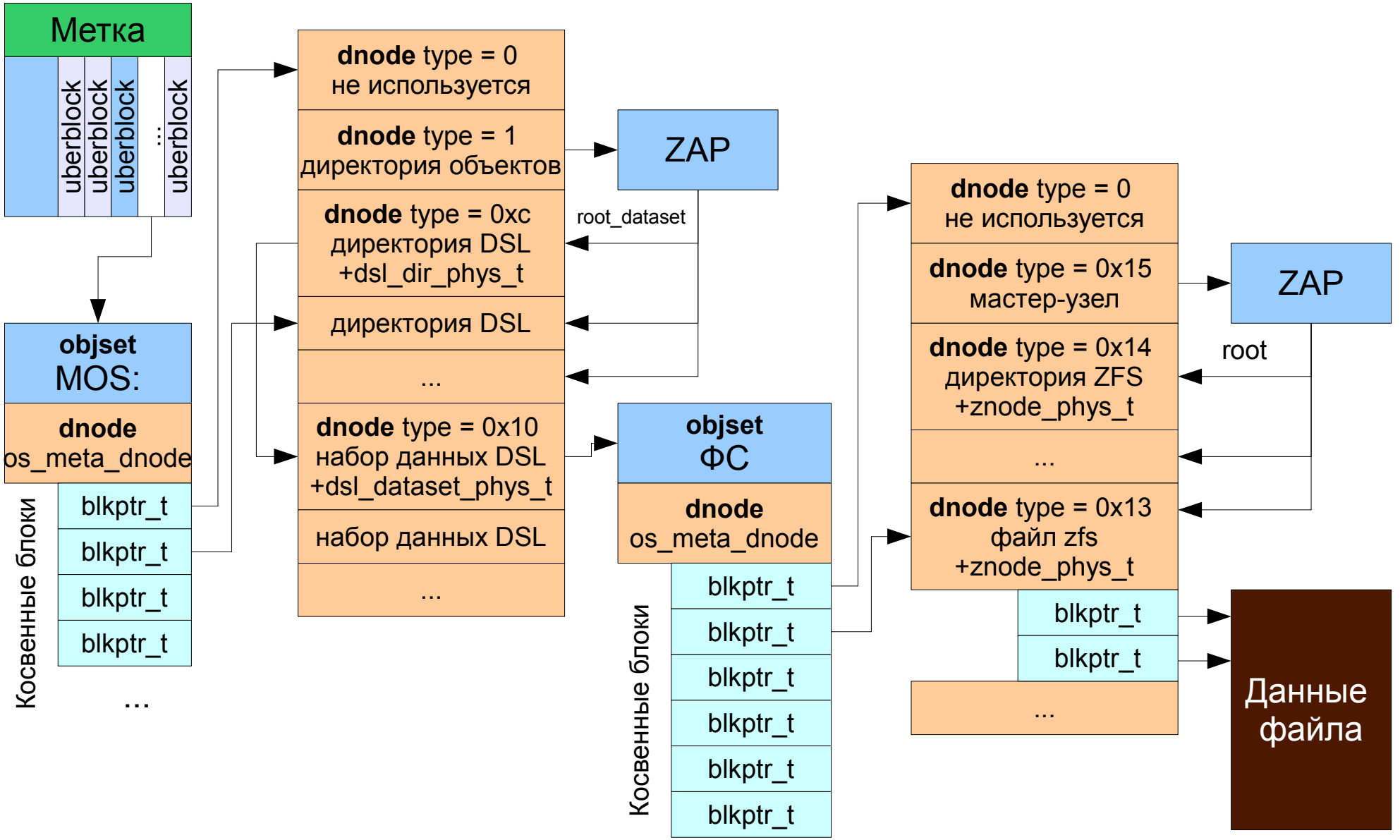
`ds_used_bytes`
`ds_compressed_bytes`
`ds_uncompressed_bytes`
`ds_unique_bytes`

`ds_props_obj` — Свойства набора данных

Уровень наборов данных и снимков DSL



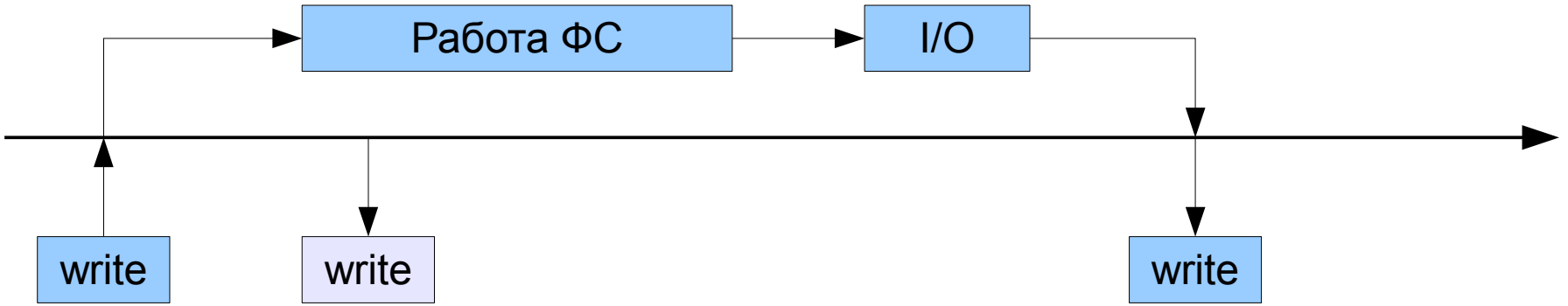
Уровень наборов данных и снимков DSL

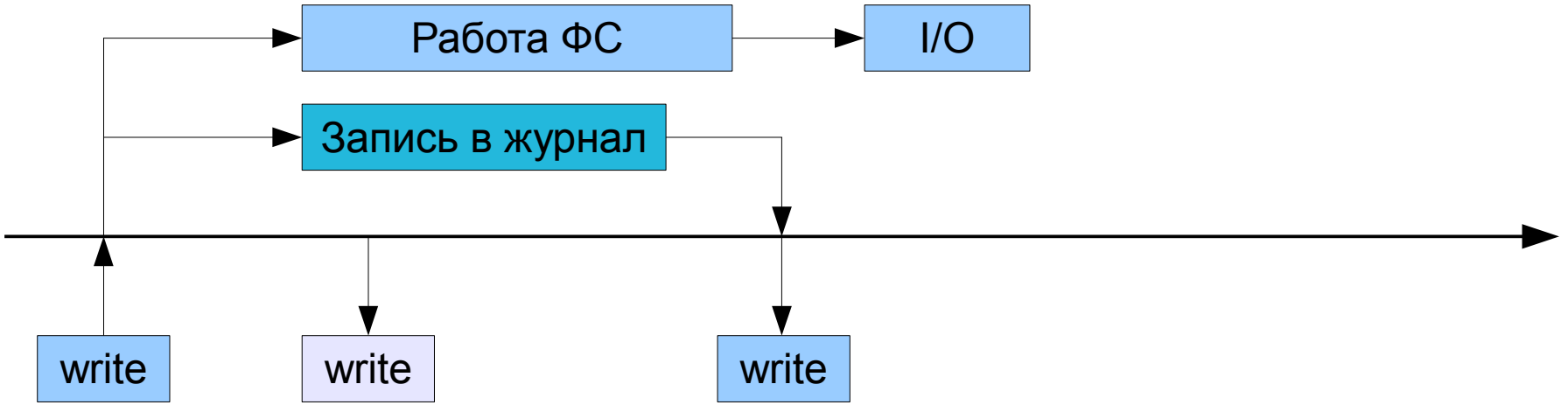


ZFS Observability, ARC, ZIL и Interface Layer

Лог намерений ZIL

- Используется для синхронной записи (O_DSYNC, fsync)





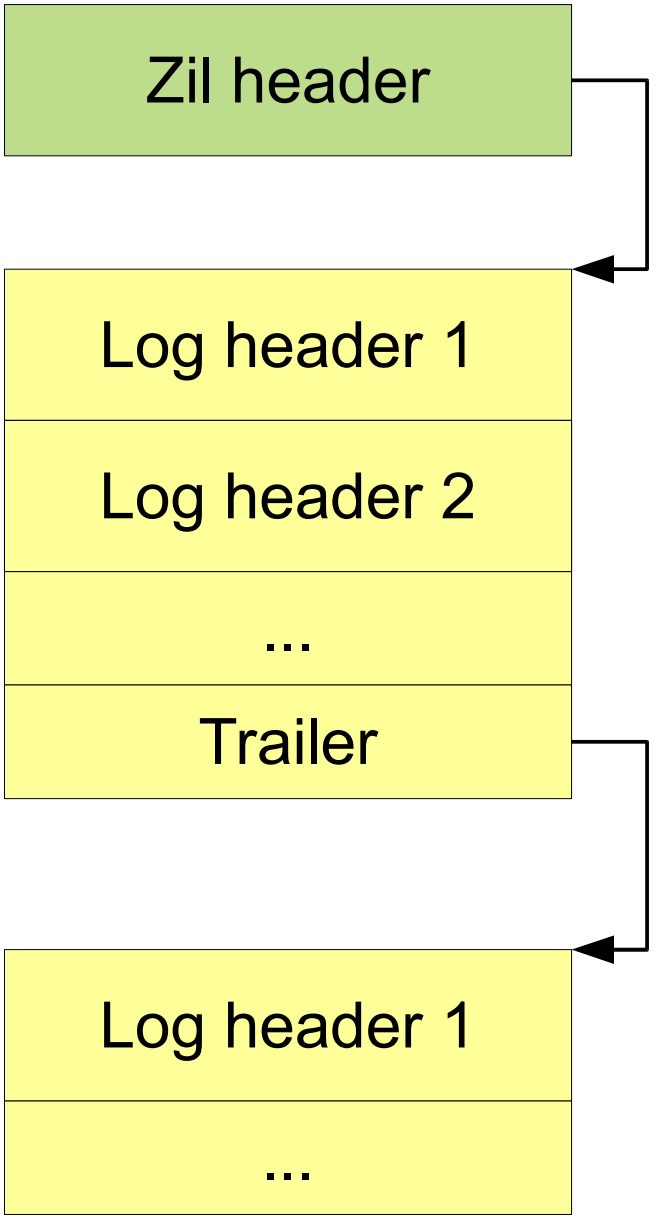
Лог намерений ZIL

- Записываются на SPA или выделенное устройство (log device)
- Zilog содержит достаточно информации для воспроизведения транзакции При фиксации транзакции zilog освобождается, а блок подтверждается
- Можно отключить по zil_disable, или с помощью свойства sync

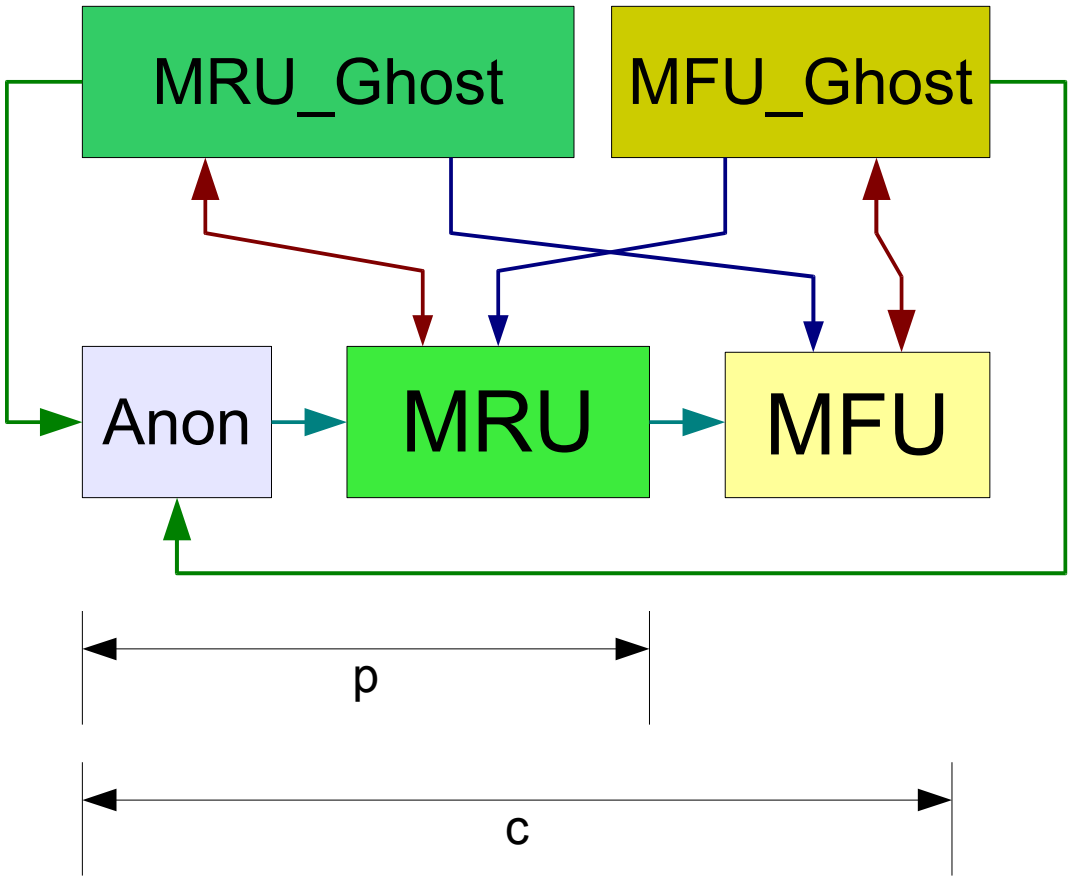
```
# zfs set sync=disabled
tiger/nsync
```

- Тюнинг

```
# zfs set logbias=(latency|
throughput) tiger/test
```

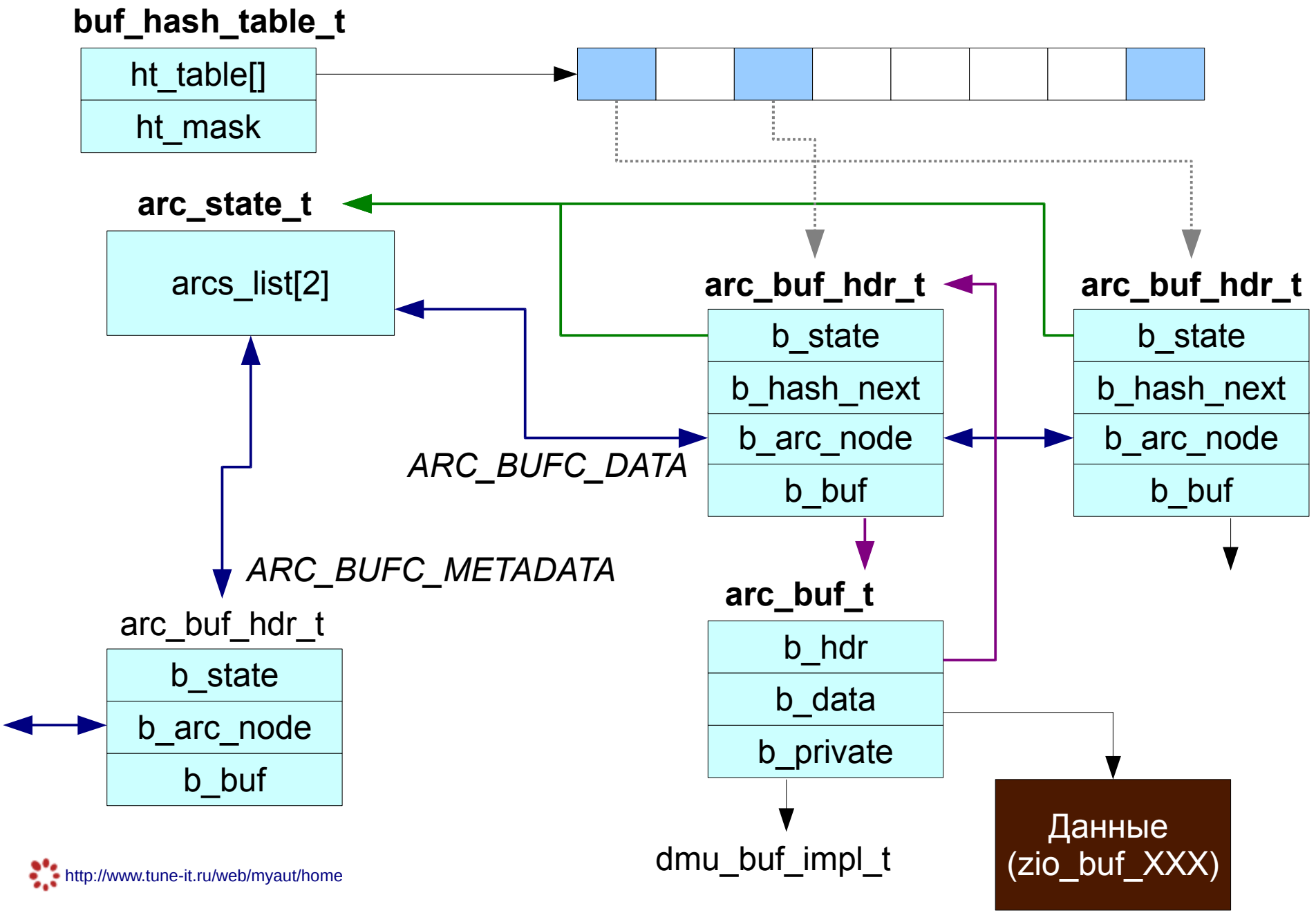


Кеш с адаптивным замещением ARC



- Адаптация (Adapting) — увеличение целевого размера кеша
- Приспособление — выталкивание (arc_evict) неактивных записей
- Reclaim — возвращение памяти ядру
  - arc_no_grow = 1
  - ARC shrink — уменьшение c на MAX(1/32 c, needfree) до c_min
  - kmem_cache_reap_now()

Кеш с адаптивным замещением ARC



Интерфейсный уровень

- ZPL представляет VFS интерфейс для объектов DMU
  - vnode ↔ znode, zfs_vsops
  - Директории организуются в виде ZAP (имя файла, номер объекта)
  - Реализует ACL в ZFS
- ZVOL реализует тома, как классический менеджер томов
  - Динамический размер тома
- libzfs представляет административный интерфейс к ZFS
  - Front-end: zpool, zfs
- JNI — Java-библиотека для администрирования ZFS

ZFS Observability

Производительность

- dtrace + fbt
- iostat теряет смысл, только zpool iostat
- kstat (ZFS arcstats, и т. п.)

Формат на диске

- zdb
- Модифицированный mdb (см. ZFS On-Disk Data Walk)