ORACLE

Elastic Data

Harvey Raja
Principal Member Technical Staff
Oracle Coherence

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

Agenda – Elastic Data

- Inception
- High Level Objectives
- Implementation Details
- Configuration
- More Details
- Collector

Traditional Stores

Backing Map Implementations

- LocalCache
  - In-memory binary map
  - Eviction / Expiry support

// MFU

// Allocations on heap

- External scheme
  - BDB
  - NIO [file | memory]
  - LH

// Sparse Usage

// Avoid LH

- ReadWriteBackingMap

// Coordinator

Local Cache

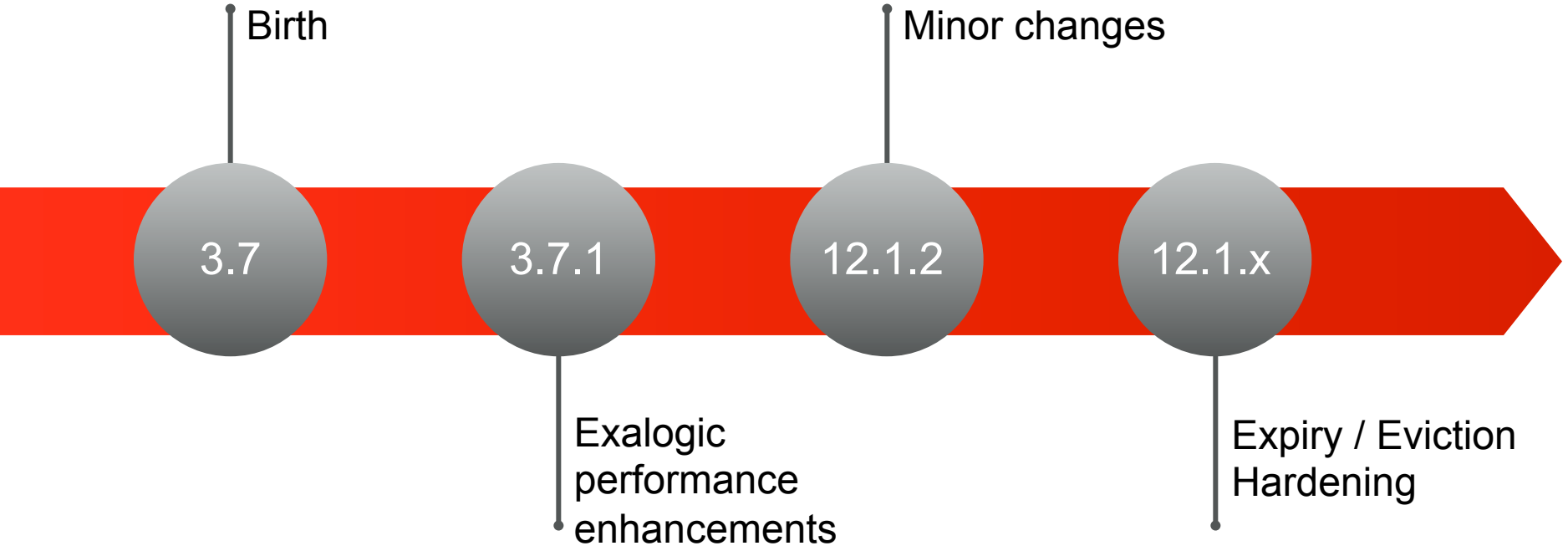
~~Disadvantages~~ Imperfections

- Allocations made on heap
  - Objects treated the same as any other allocations
- High storage in a backing map == Large heap
- Unable to isolate scratch space from application storage

Elastic Data: High Level Objectives

- Both RAM and Disk (off heap) based stores
- Retain keys in memory with tickets
- Journaling – append only
- Overflow from RAM to Disk
- Big Data – we manage the (de)allocations thus compaction

Elastic Data Over Time



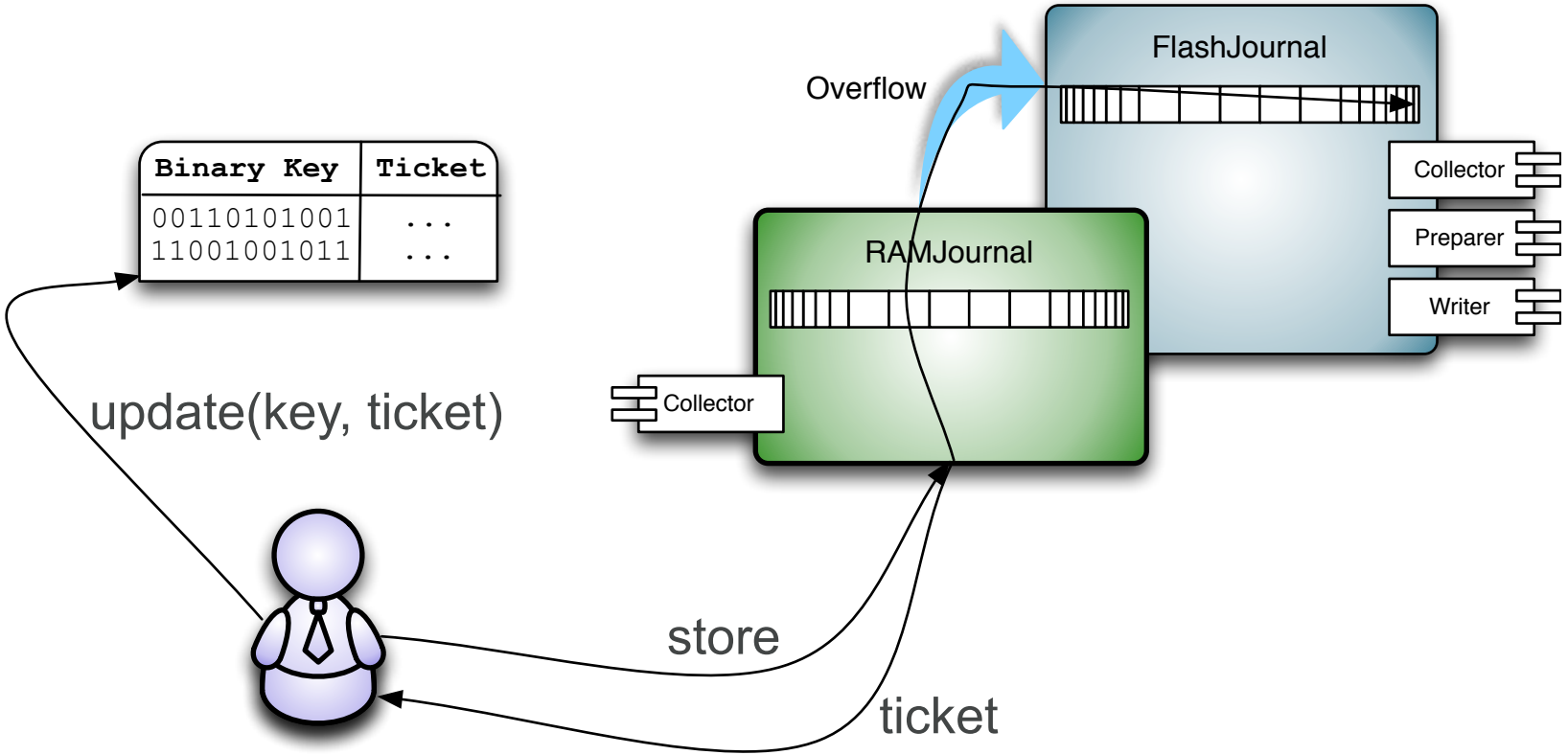
Implementation Details

- Resource Managers (RamJournalRM, FlashJournalRM)
  - Scoped to member
  - Manage resources (Journals)
- Hold in-memory structure of keys to tickets
- Ticket returned from the journal after a successful store
- Ticket used to locate binary thus is passed to Journal.read

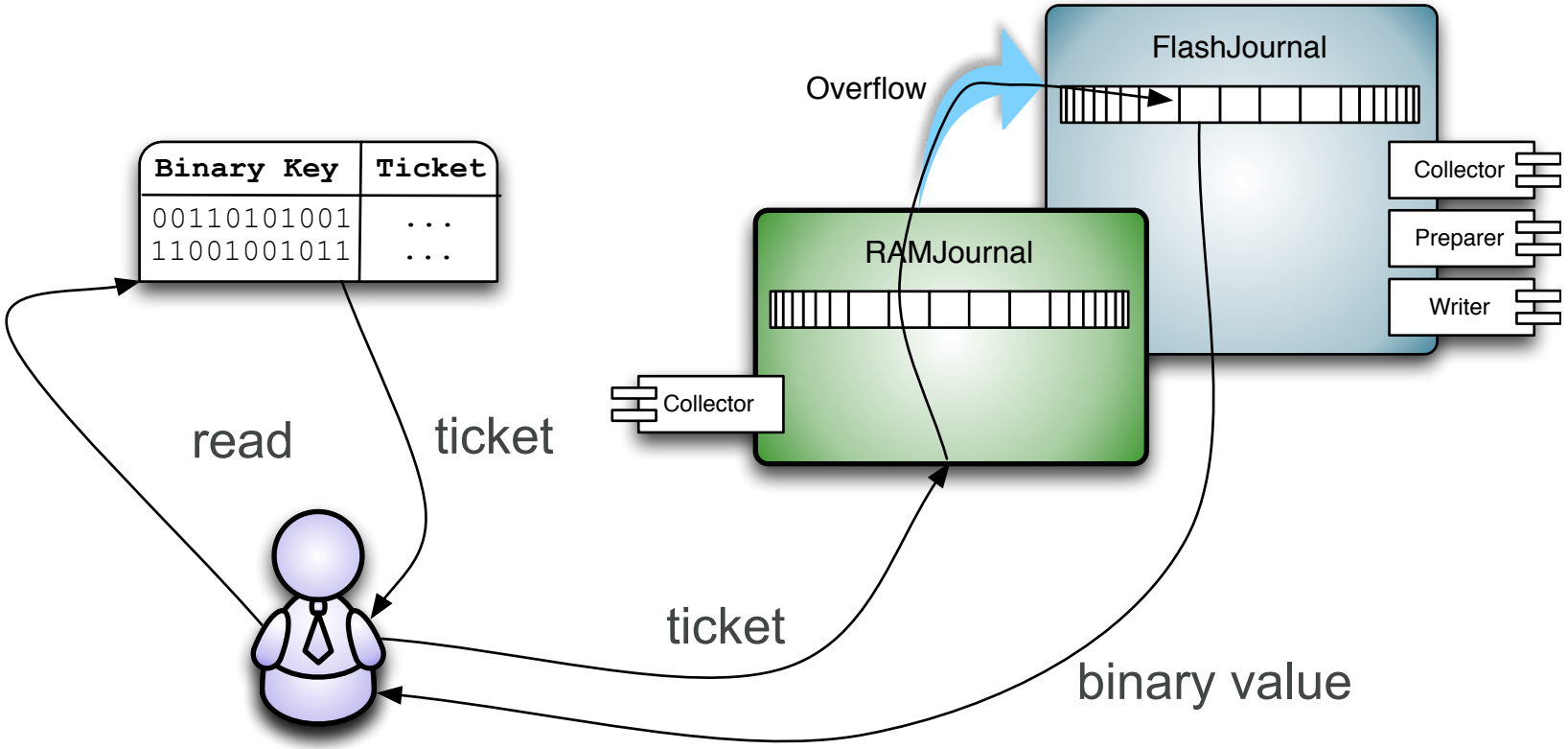
Implementation Details

- Update to an existing entry is an append and release
  - Journaling
- All ResourceManagers require a collector
  - Responsible for garbage collection and compaction (evacuation)
- RAM is always used in conjunction with Disk (Flash)
  - When RAM reaches capacity we overflow to Flash
- Minimal locking; CAS operations where possible

Store flow



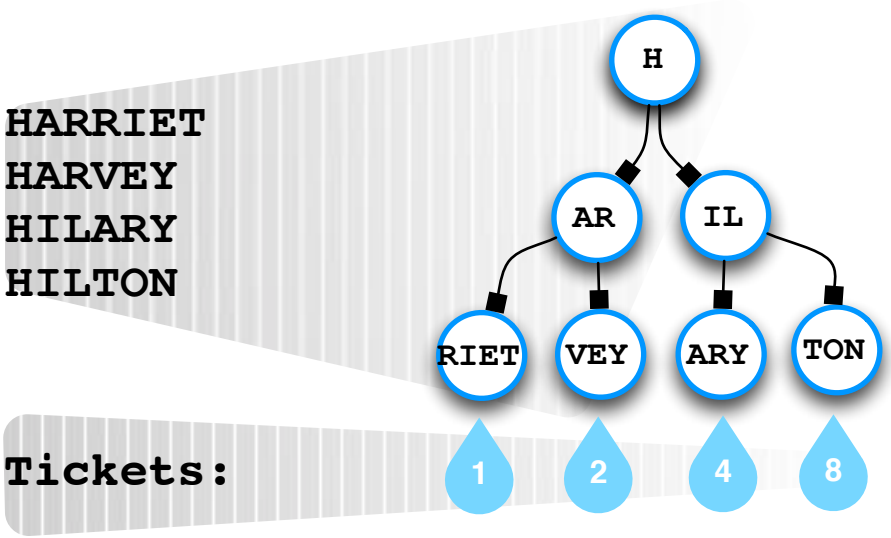
Load flow



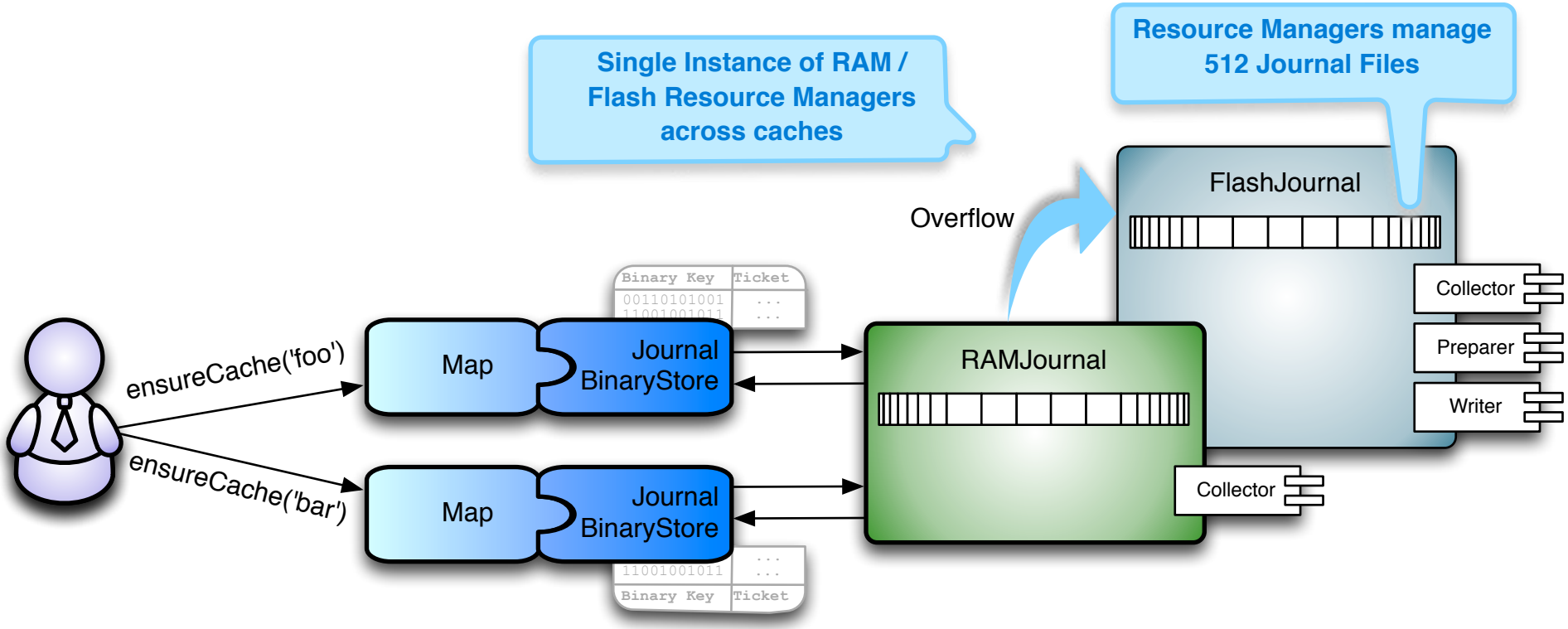
Binary Radix Tree

Keys and Tickets

- Keys are stored in memory but in compact form
- Binary Radix tree allows sharing of common denominators
- Tree instance per partition and cache
  - Increase in (cache & partition) density likely to yield more benefit
  - Specifically common segments in binary values, however tricky to measure



Cache ResourceManagers



Configuration

Overview

- Operational configuration
  - Defines member scoped resource managers
- Cache configuration
  - Defines cache storage implementation
  - No child elements (3.7.1)

```
<coherence>
  <cluster-config>
    <journaling-config>
      <ramjournal-manager>
          ...
      </ramjournal-manager>
      <flashjournal-manager>
          ...
      </flashjournal-manager>
    </journaling-config>
  </cluster-config>
</coherence>
```

```
<cache-config>
  ...
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>dist-ram</scheme-name>
      <backing-map-scheme>
        <ramjournal-scheme/>
      </backing-map-scheme>
    </distributed-scheme>
    ...
  </caching-schemes>
</cache-config>
```

Configuration

RAM Journal

- How utilized a file is to be eligible for compaction
- Maximum amount of storage per file
- Maximum size for each binary value
- Maximum size in total
- Whether NIO should be used opposed to storing on heap

```
<coherence>
  <cluster-config>
    <journaling-config>
      <ramjournal-manager>
        <minimum-load-factor>0.84</minimum-load-factor>
        <maximum-file-size>512K</maximum-file-size>
        <maximum-value-size>128K</maximum-value-size>
        <maximum-size>25%</maximum-size>
        <off-heap>false</off-heap>
      </ramjournal-manager>
      ...
    </journaling-config>
  </cluster-config>
</coherence>
```

Configuration

RAM Journal Validation

- Maximum Size can be percentage of heap
- Maximum File Size == Maximum Size / 512
- Maximum Value Size <= Maximum File Size / 2
- Log statements if values are adjusted due to conflicts with other values

Configuration

Flash Journal

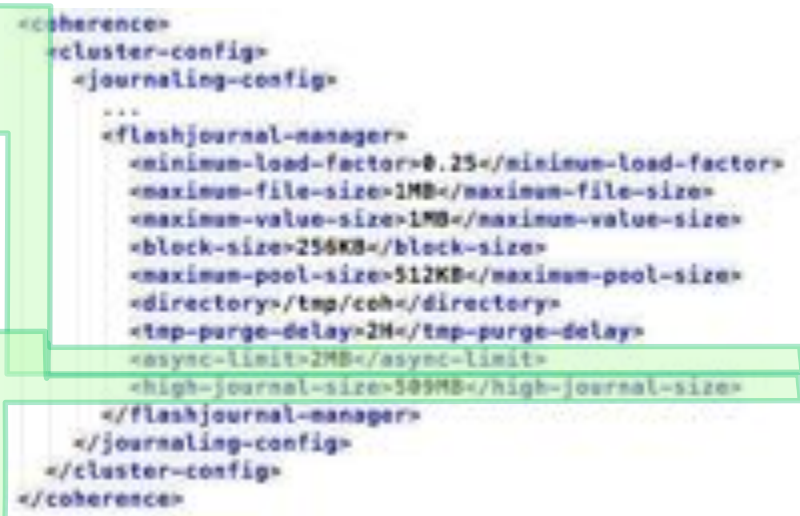
- Batch size to perform writes to file
- Maximum size of buffer prior to writing to disk
  - max-pool-size % block-size == 0
- Mount point for device writes
- Upon startup which files should be eligible for purge

```
<coherence>
  <cluster-config>
    <journaling-config>
      ...
      <flashjournal-manager>
        <minimum-load-factor>0.25</minimum-load-factor>
        <maximum-file-size>1MB</maximum-file-size>
        <maximum-value-size>1MB</maximum-value-size>
        <block-size>256KB</block-size>
        <maximum-pool-size>512KB</maximum-pool-size>
        <directory>/tmp/coh</directory>
        <tmp-purge-delay>2M</tmp-purge-delay>
        <async-limit>2MB</async-limit>
        <high-journal-size>509MB</high-journal-size>
      </flashjournal-manager>
    </journaling-config>
  </cluster-config>
</coherence>
```

Configuration

Flash Journal

- Maximum amount of storage that can be pending to be written to device
  - Will stop further writes until writing to the device catches up
- How much memory should be used to enter an aggressive collection mode



Configuration

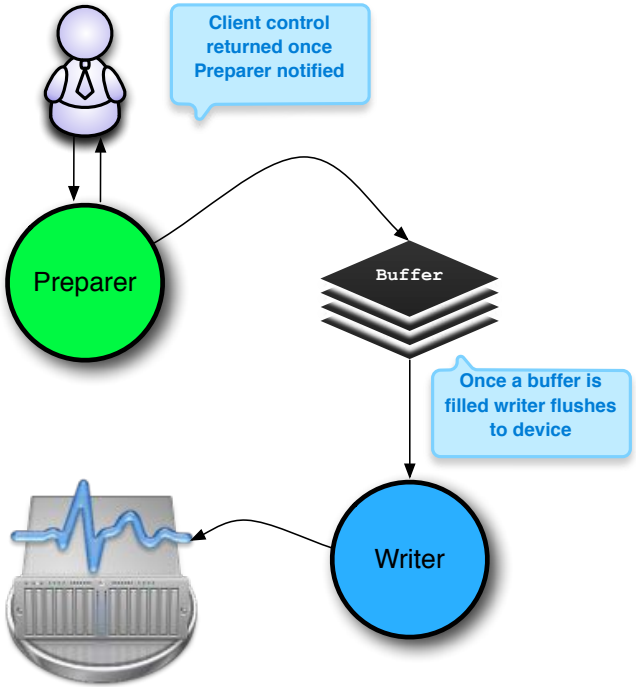
Flash Journal Validation

- Block size must be a 2^x
- Pool Size % Block Size == 0
- 4K <= Async Limit <= 1GB
- High Journal Size > Max File Size

Flash Journal

- Writes performed asynchronous to store
- A couple of threads co-ordinate to perform writing
- Inbuilt flow control mechanism to ensure writer is not overwhelmed

Flash Journal




```
$ kill -3 pid
Full thread dump Java HotSpot(TM) 64-Bit Server VM (20.14-b01-447 mixed mode):
...
"Journal-Writer" daemon prio=5 tid=7f8dfb844000 nid=0x10d86b000 runnable [10d86a0
   java.lang.Thread.State: RUNNABLE
            at java.lang.Thread.currentThread(Native Method)
            at java.nio.channels.spi.AbstractInterruptibleChannel.blockedOn(Abst
            at java.nio.channels.spi.AbstractInterruptibleChannel.end(AbstractIr
            at sun.nio.ch.FileChannelImpl.write(FileChannelImpl.java:203)
            - locked <7bf64fc68> (a java.lang.Object)
            at com.tangosol.io.journal.FlashJournalRM$WriterDaemon$PendingWriteT
            at com.tangosol.io.journal.FlashJournalRM$WriterDaemon.run(FlashJour
            at com.tangosol.util.Daemon$DaemonWorker.run(Daemon.java:803)
            at java.lang.Thread.run(Thread.java:680)

"Journal-Preparer" daemon prio=5 tid=7f8dfb843800 nid=0x10d768000 in Object.wait(
   java.lang.Thread.State: WAITING (on object monitor)
            at java.lang.Object.wait(Native Method)
            at java.lang.Object.wait(Object.java:485)
            at com.tangosol.util.SingleWaiterMultiNotifier.await(SingleWaiterMul
            - locked <7c4b36590> (a java.lang.Object)
            at com.tangosol.io.journal.FlashJournalRM$PreparerDaemon.run(FlashJo
            at com.tangosol.util.Daemon$DaemonWorker.run(Daemon.java:803)
            at com.tangosol.io.journal.FlashJournalRM$PreparerDaemon$1.run(Flash
            at java.lang.Thread.run(Thread.java:680)
```

Collector

- Responsibility: To reclaim memory from Journal Files
  - Dispose of files only with garbage
  - Compacts allocated memory thus reclaiming inaccessible memory segments
- Initial interval is 30s but adjusts based on Journal usage and predicting benefit of an execution
- Each Resource Manager has its own Collector

Collector

Continued

- RAM Journal executes against multiple Journal files as it is relatively cheap to compact
- Flash Journal compacts a single file per execution to avoid overwhelming the block device
- Configuring High Journal Size (3.7.1) allows configuring when the collector should become aggressive in collection

Q & A

Hardware and Software

Engineered to Work Together


ORACLE®