APACHE SLING & FRIENDS TECH MEETUP
BERLIN, 25-27 SEPTEMBER 2017

TarMK: Facts and Figures

Michael Dürig / Valentin Olteanu, Adobe

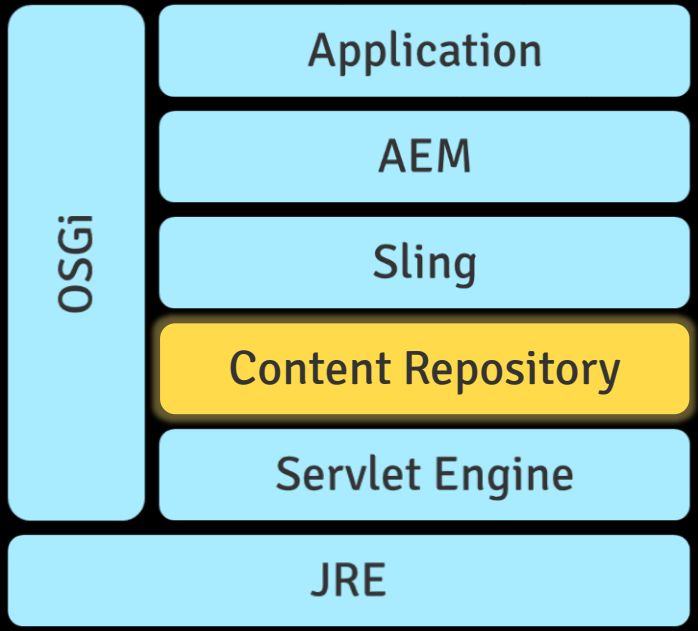
adaptTo()

Slooooow

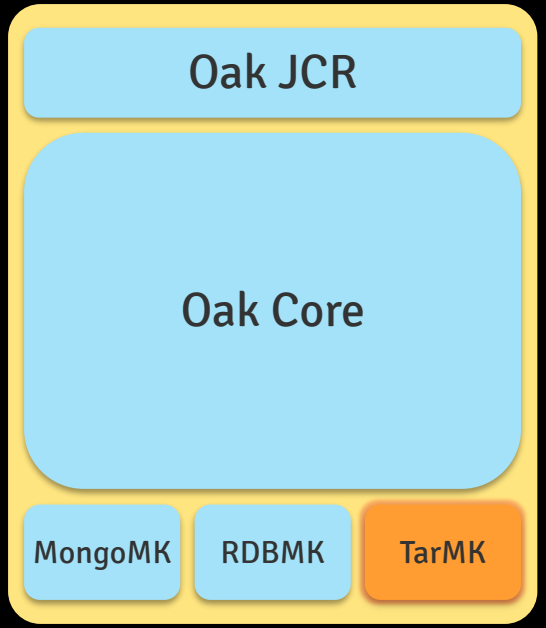
Agenda

- AEM, Oak and the TarMK
- System Resources
- Problems and Symptoms
- Outlook

Introducing the TarMK

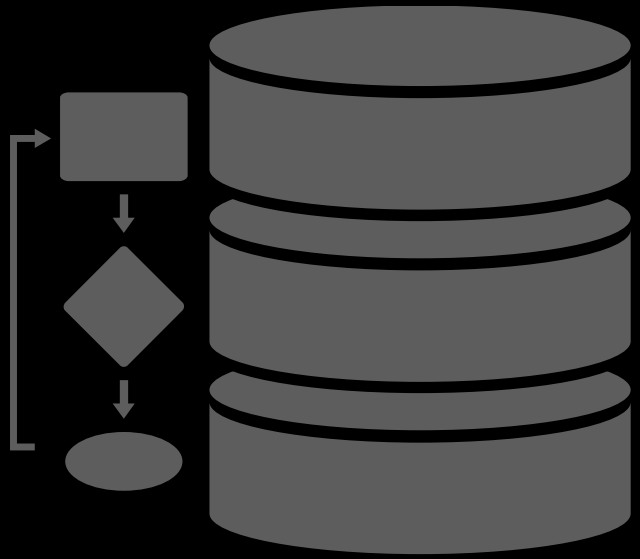


Introducing the TarMK

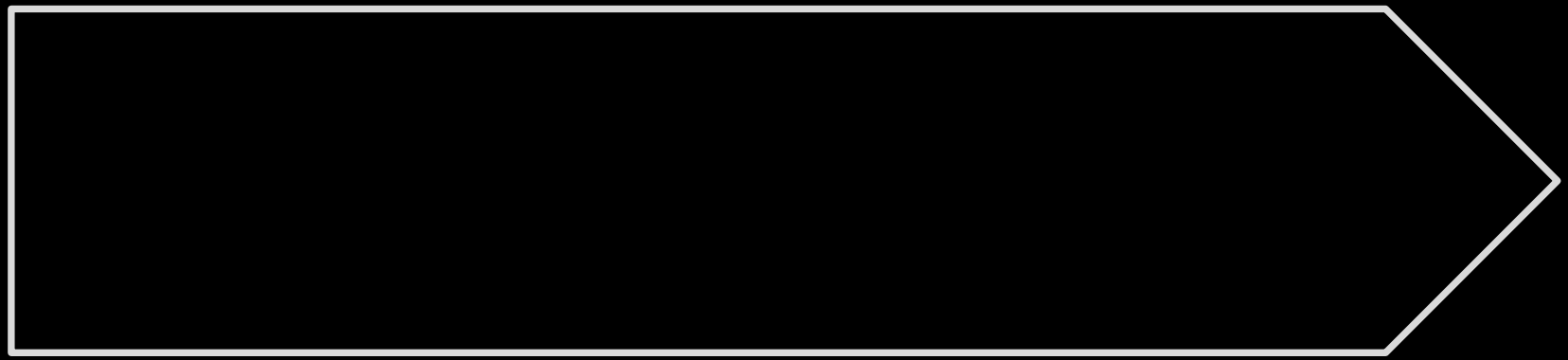


Features of the TarMK

- Embedded Database
  - Hierarchical
  - Fast / Small
  - Vertical scalability
  - MVCC / append only

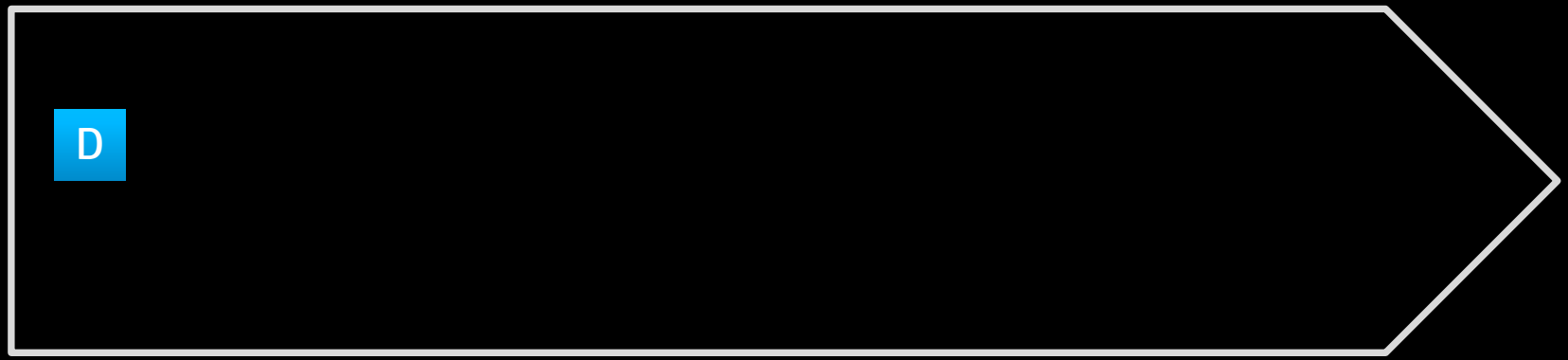


Records and Segments



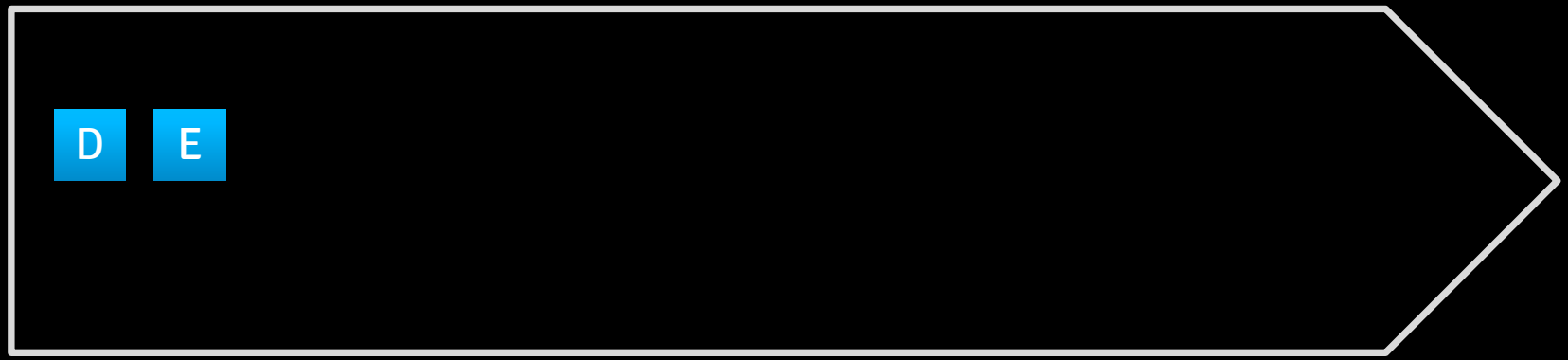
adaptTo()

Records and Segments



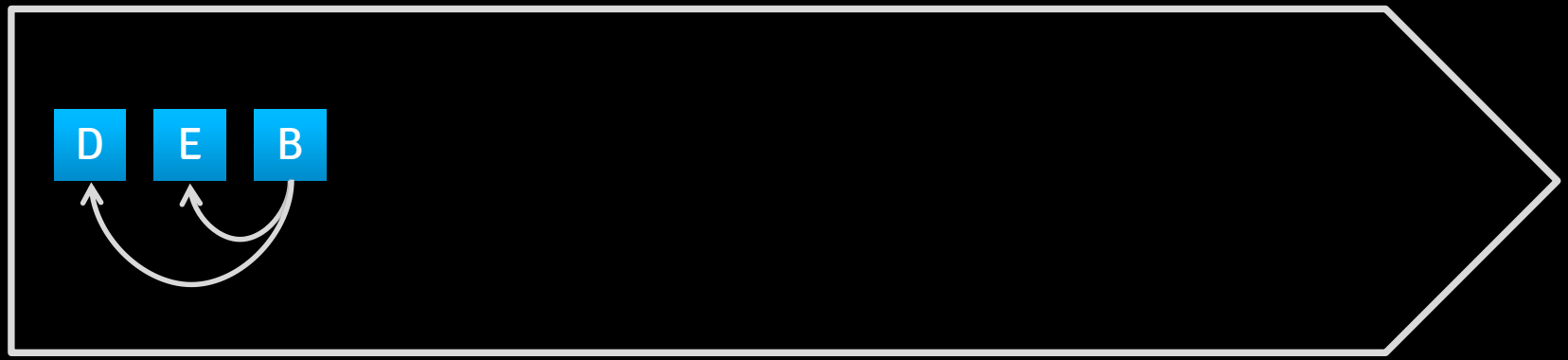
adaptTo()

Records and Segments



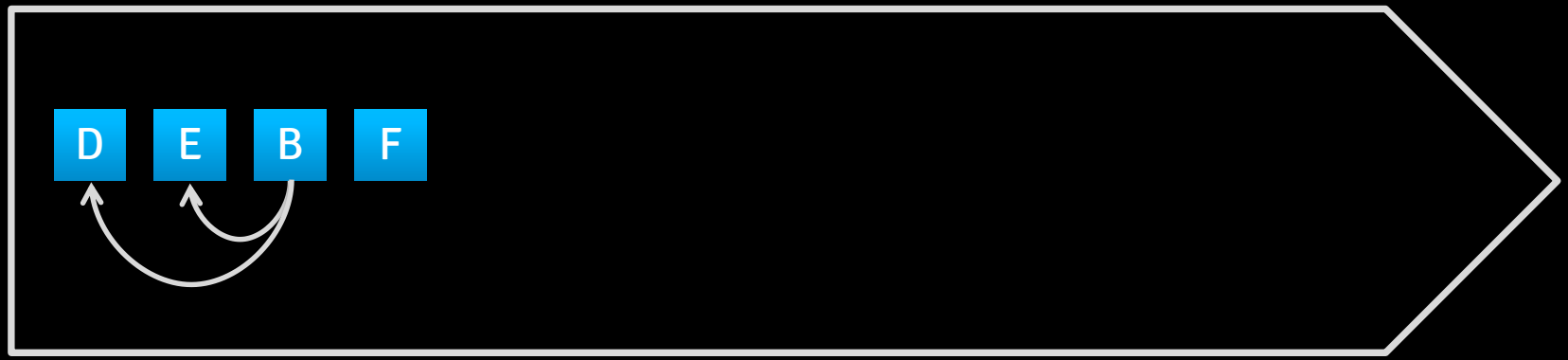
adaptTo()

Records and Segments

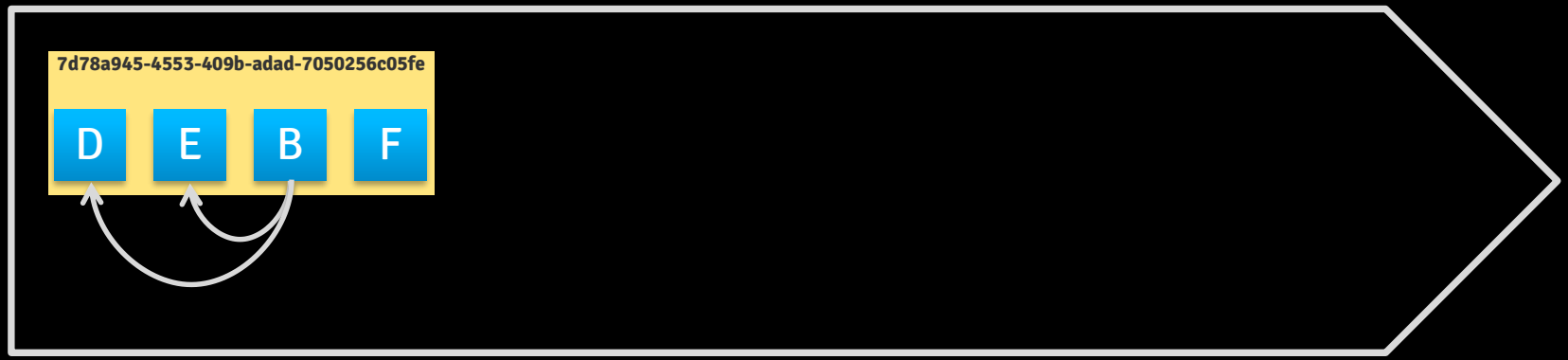


adaptTo()

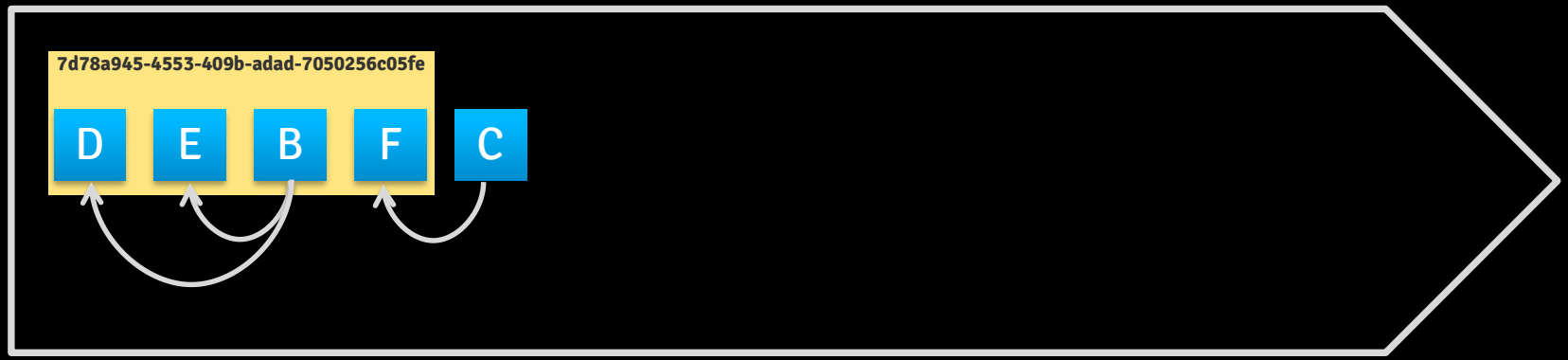
Records and Segments



Records and Segments

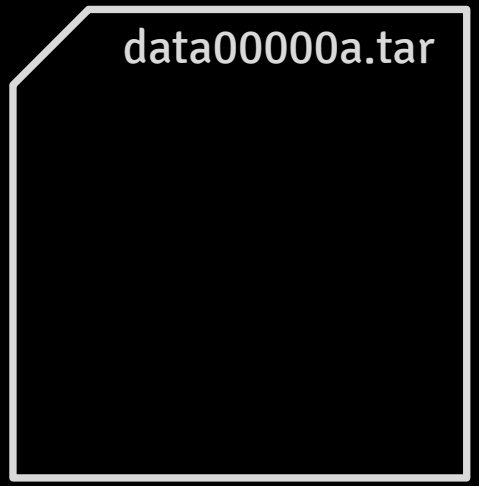


Records and Segments

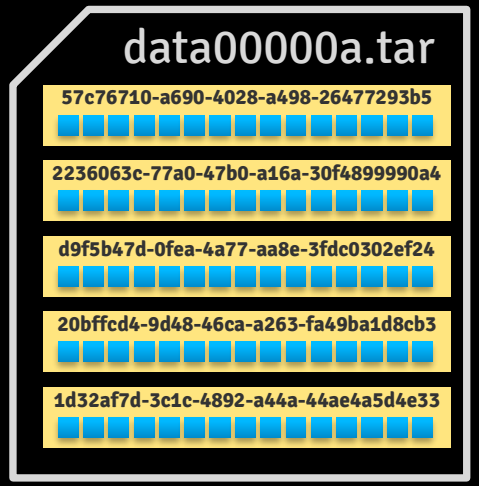


Records and Segments

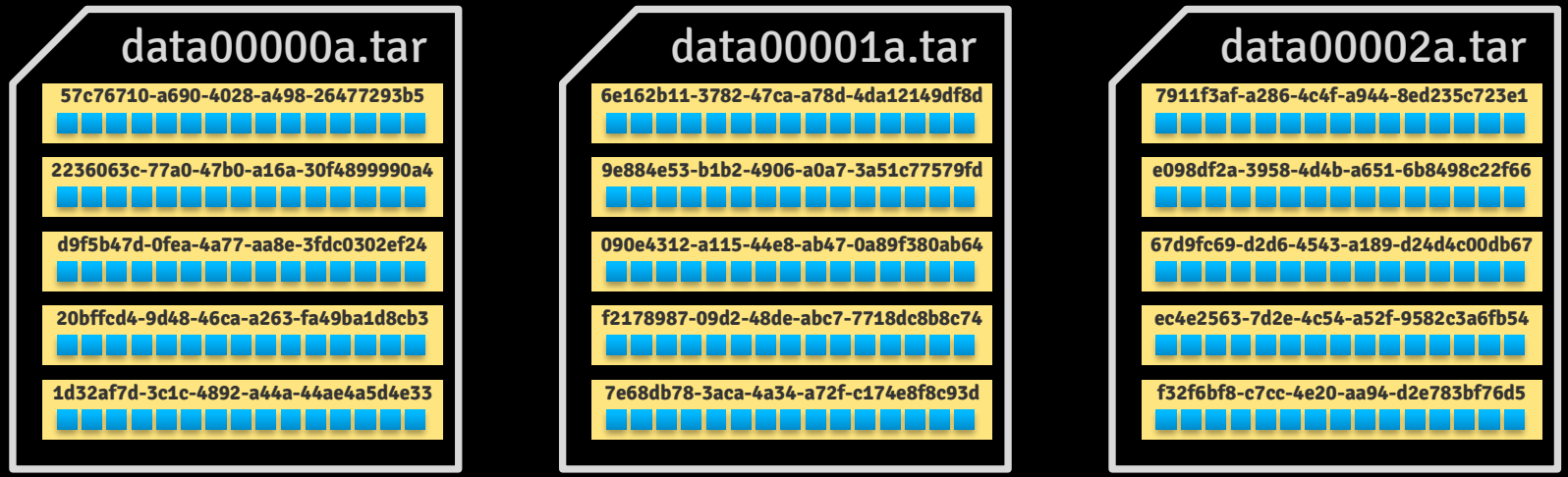
Segments and Tar Files



Segments and Tar Files

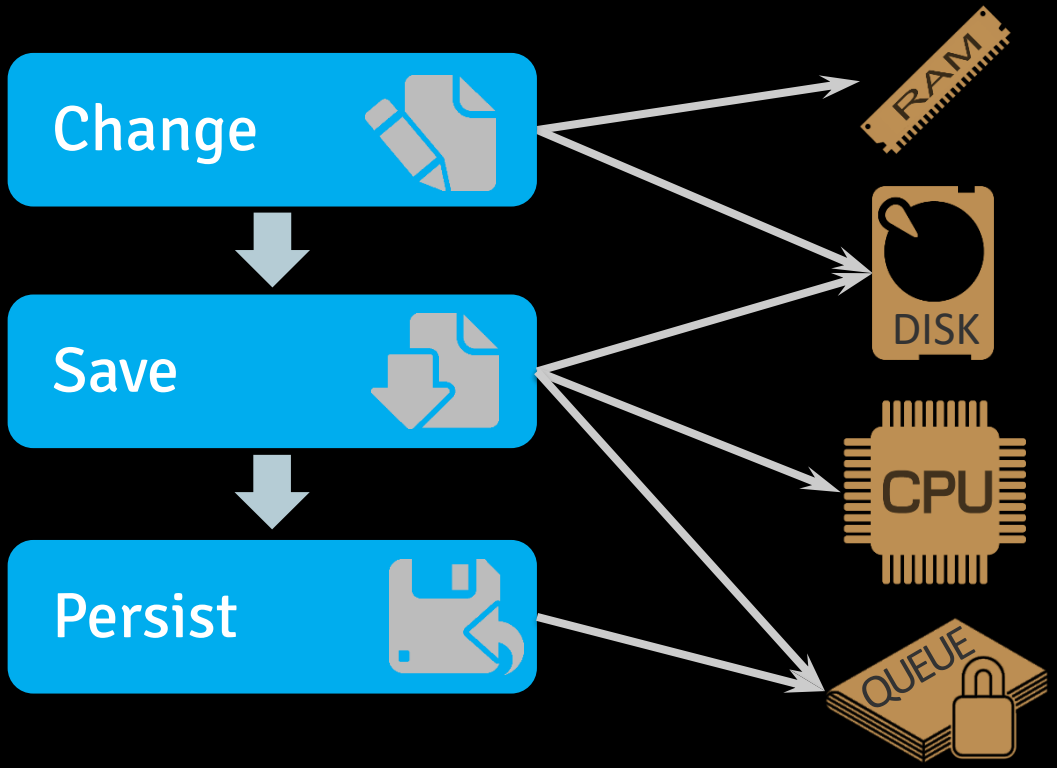


Segments and Tar Files



System Resources

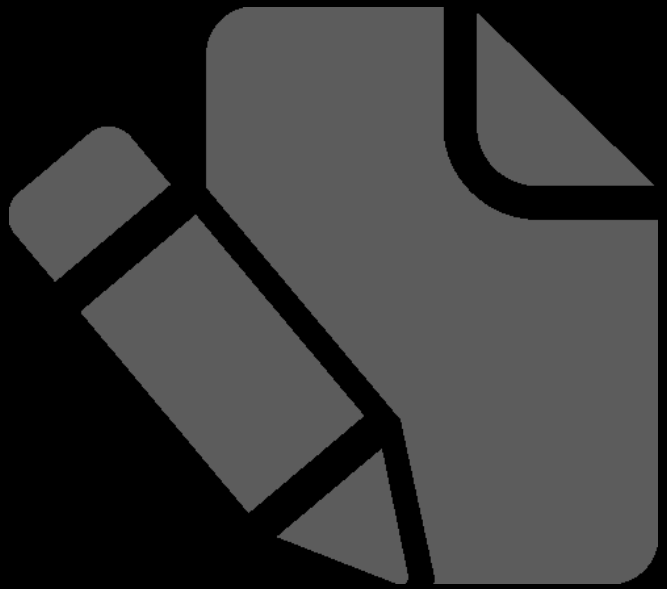
Write Operation



adaptTo()

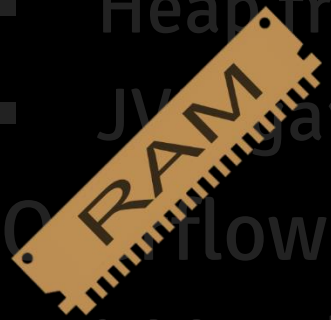
Change

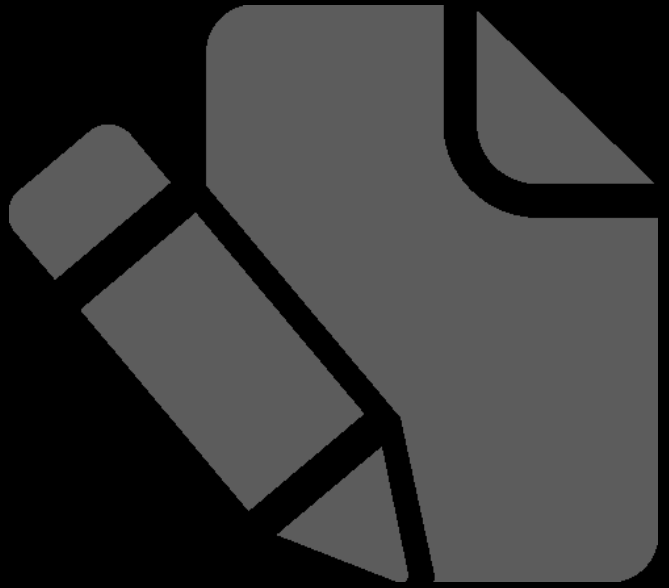
- Transient on heap
  - Heap fragmentation
  - JVM garbage collection
- Overflowed to disk
  - Write ahead
  - Segment fragmentation



Change

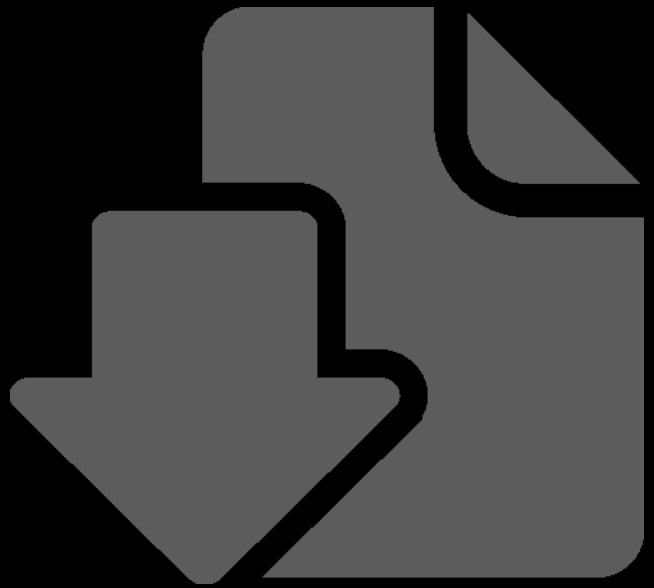
- Transient on heap
  - Heap fragmentation
  - JVM garbage collection
- Overflowed to disk
  - Write ahead
  - Segment fragmentation





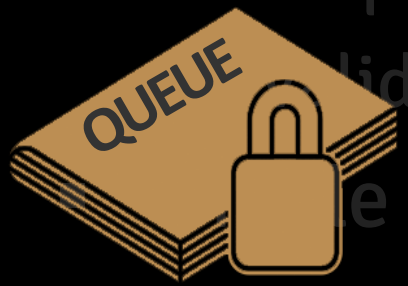
Save

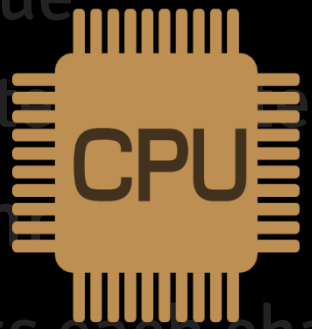
- Process Changes
  - Enqueue
  - Validate, update
- Single thread
  - Process each change, O(n)
  - Discarded sessions cause segment fragmentation

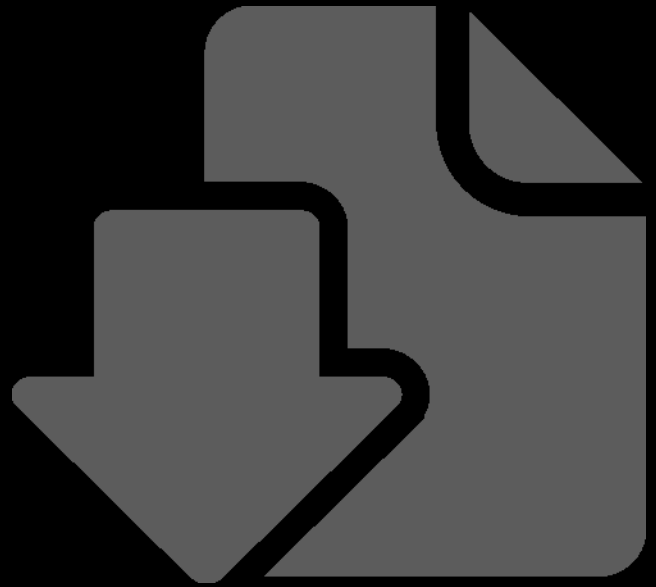


Save

- Process Changes
  - Enqueue
  - Validate
  - Save the
  - Process each change, O(n)
  - Discarded sessions cause segment fragmentation

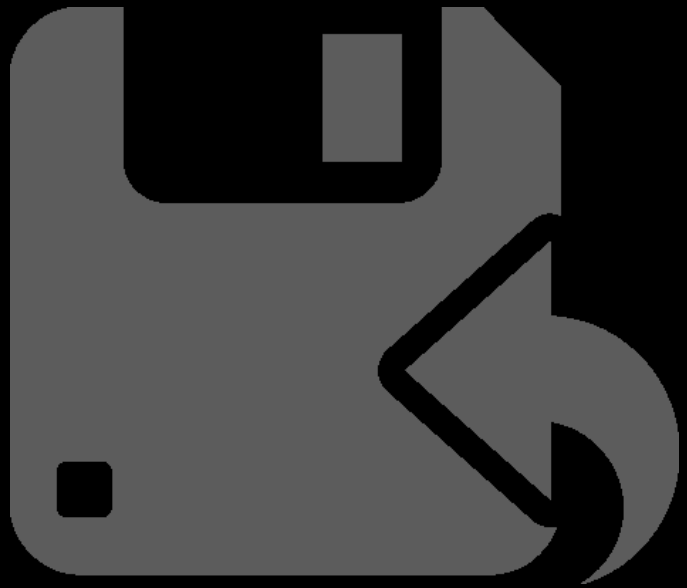






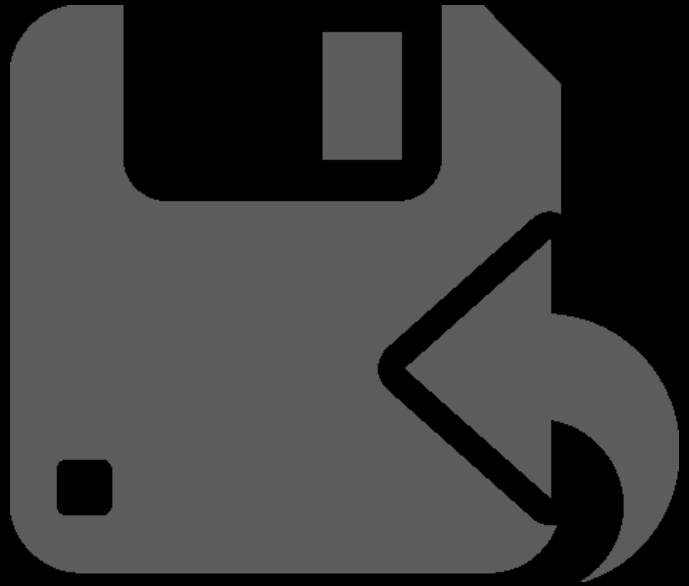
Persist

- Persist
  - Update journal
  - Dequeue
- Fan-out
  - Asynchronous indexes
  - Workflows, Assets, Rendition
  - Replication



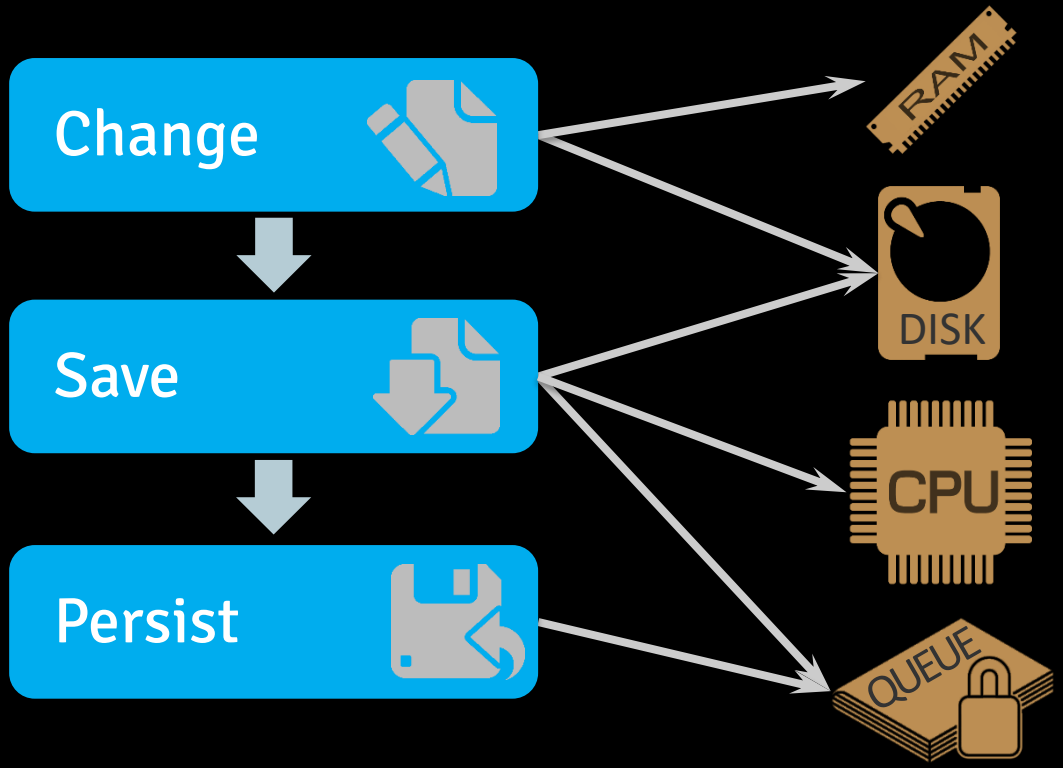
Persist

- Persist
  - Update journal
  - Dequeue
- Fan-out
  - Asynchronous indexes
  - Workflows, Assets, Rendition
  - Replication



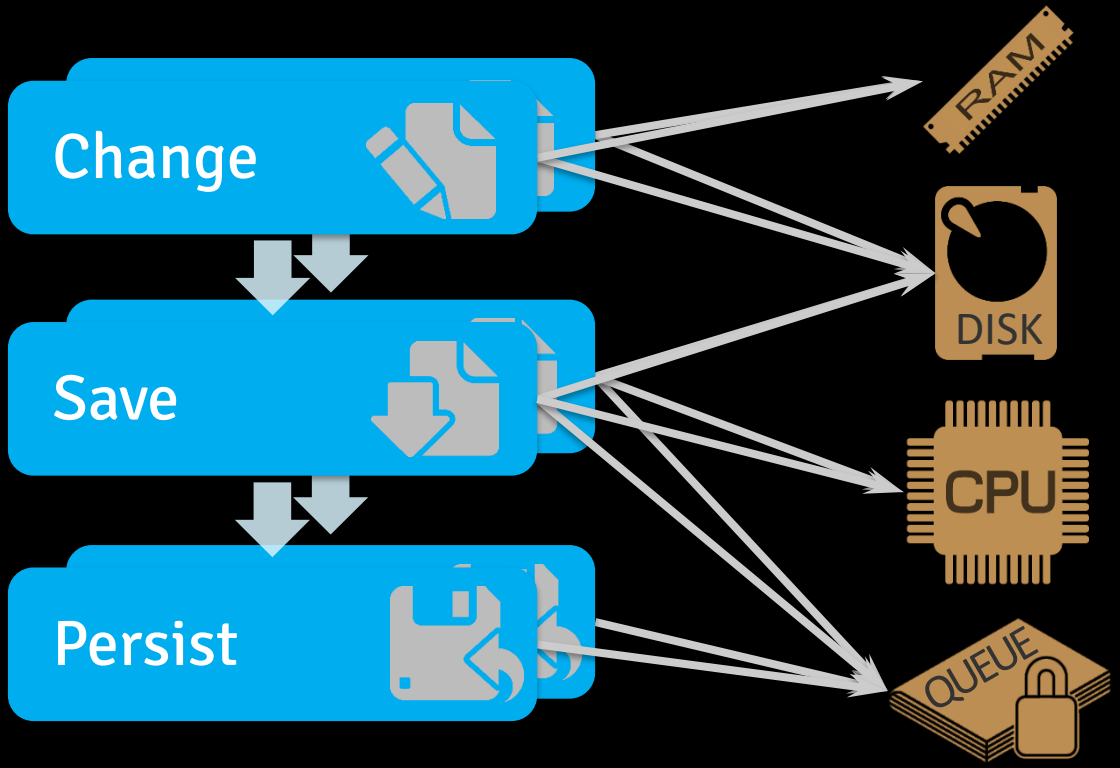
adaptTo()

Concurrent Changes

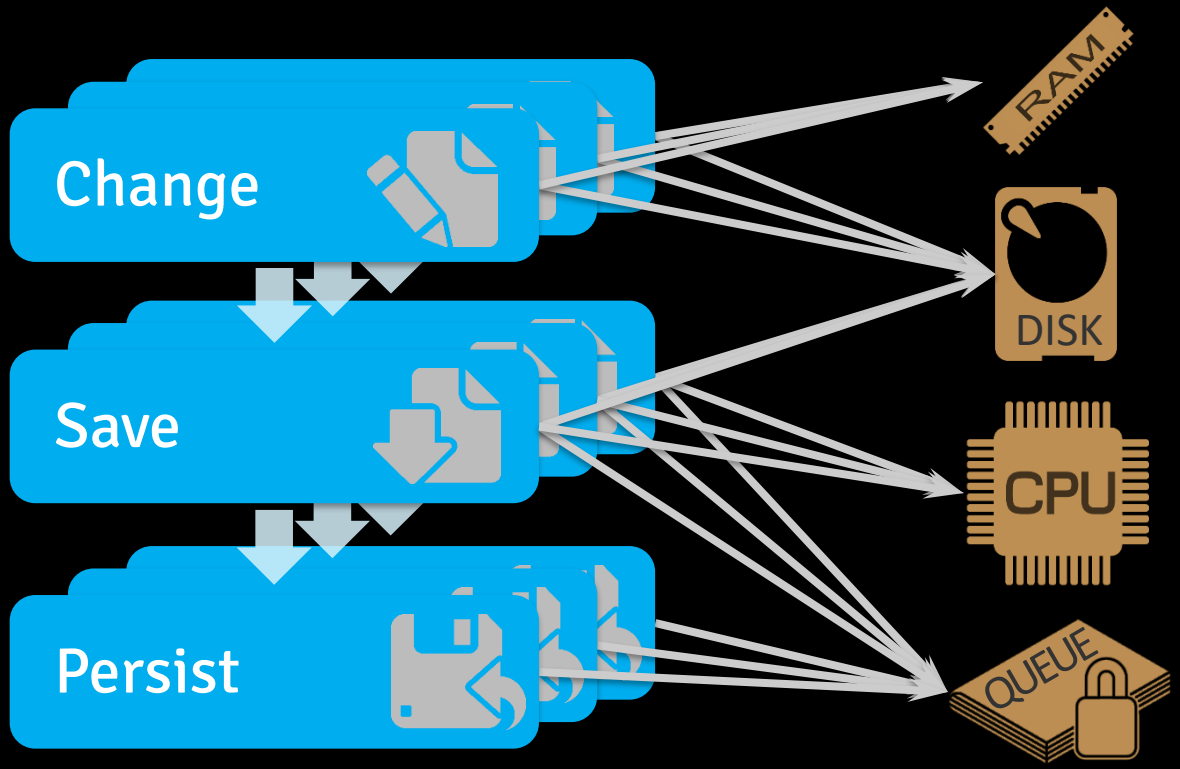


adaptTo()

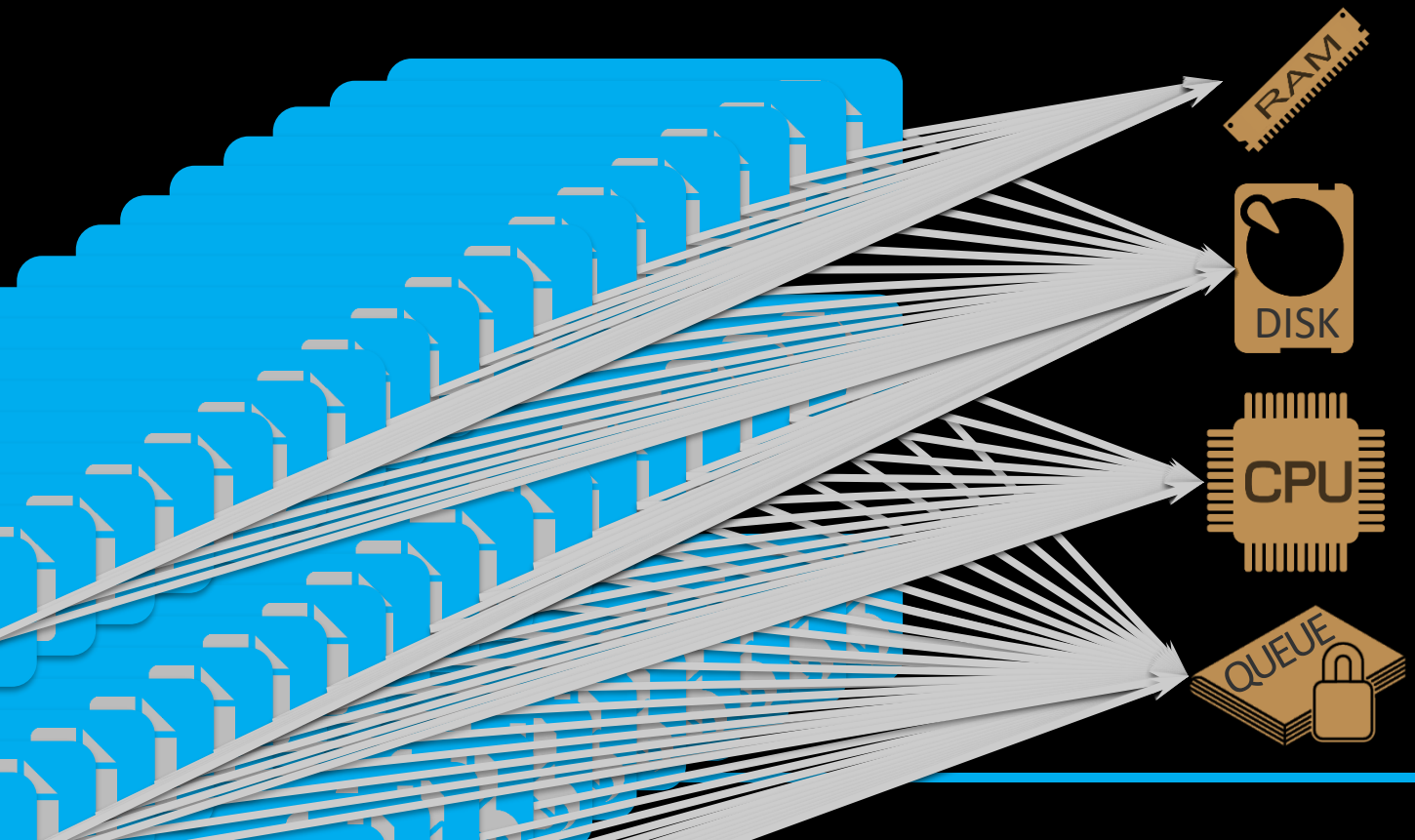
Concurrent Changes



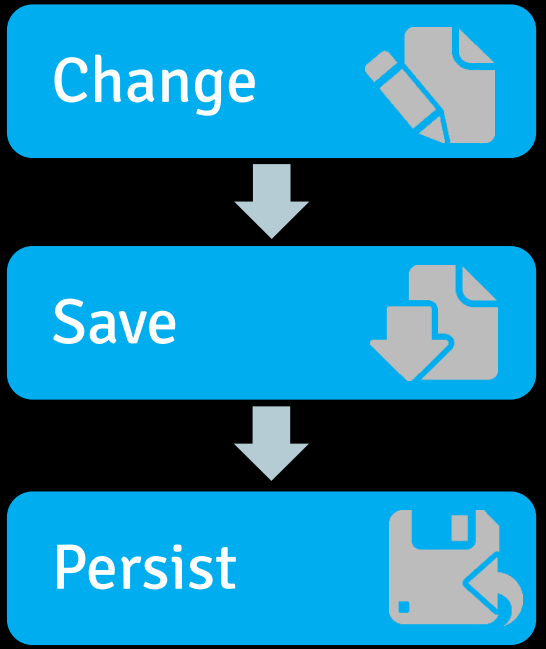
Concurrent Changes



Concurrent Changes

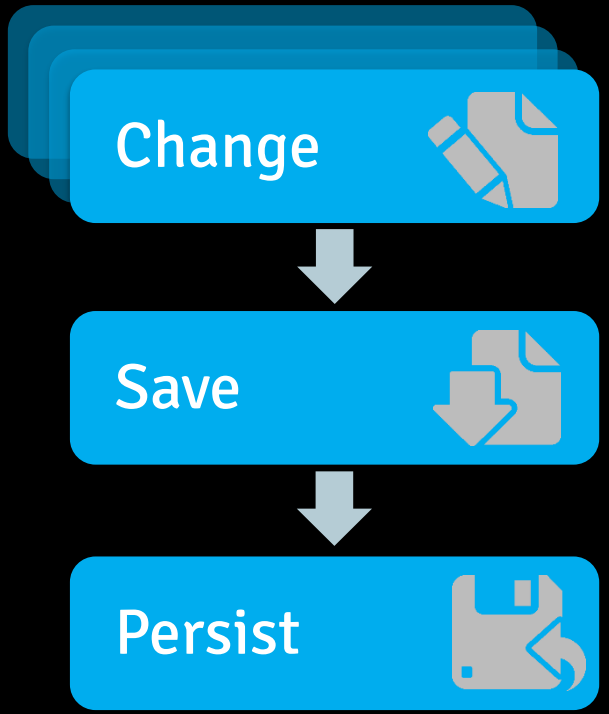


Concurrent Changes

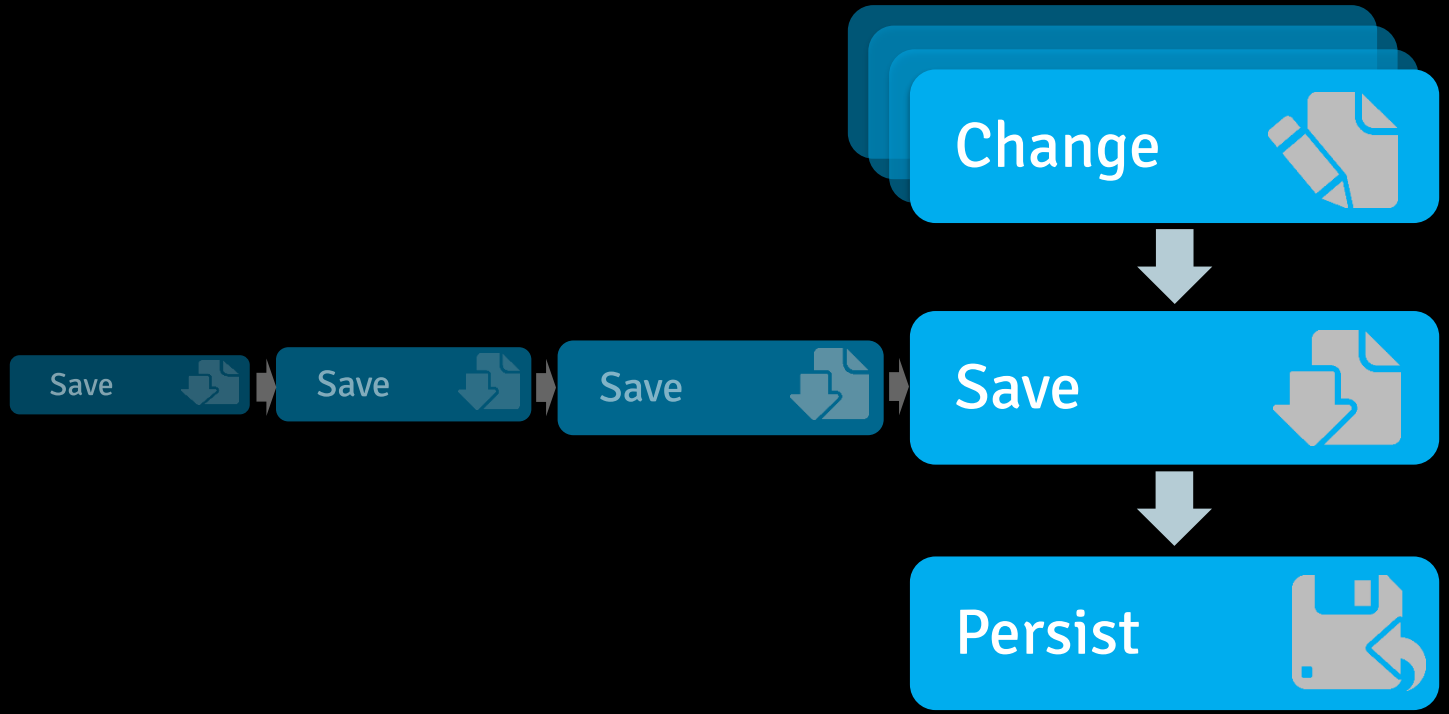


adaptTo()

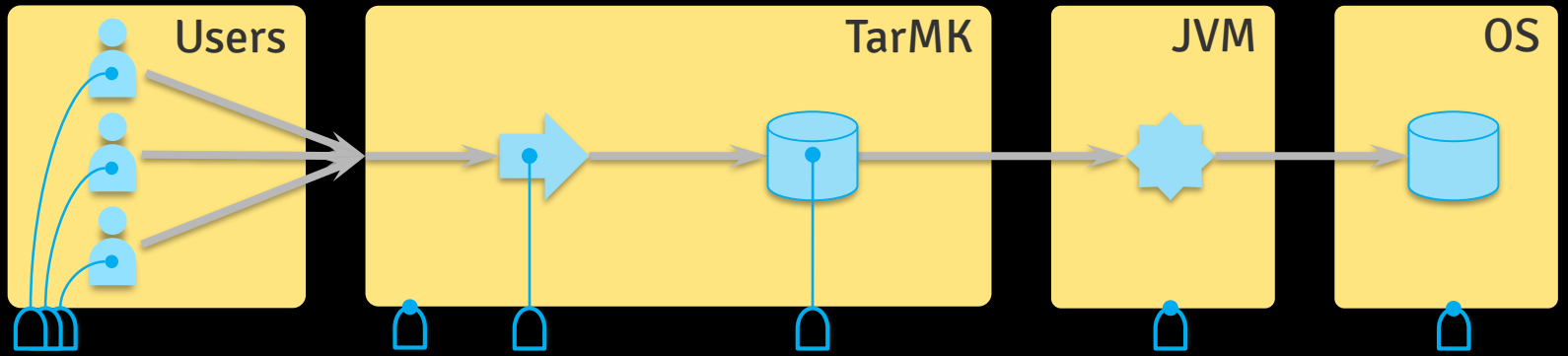
Concurrent Changes



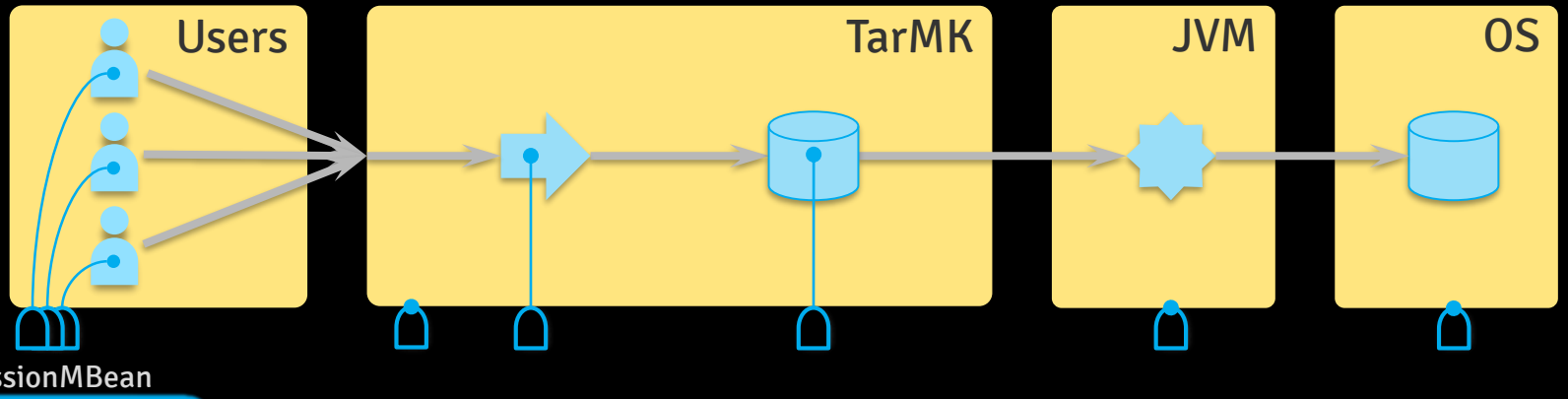
Concurrent Changes



Monitoring

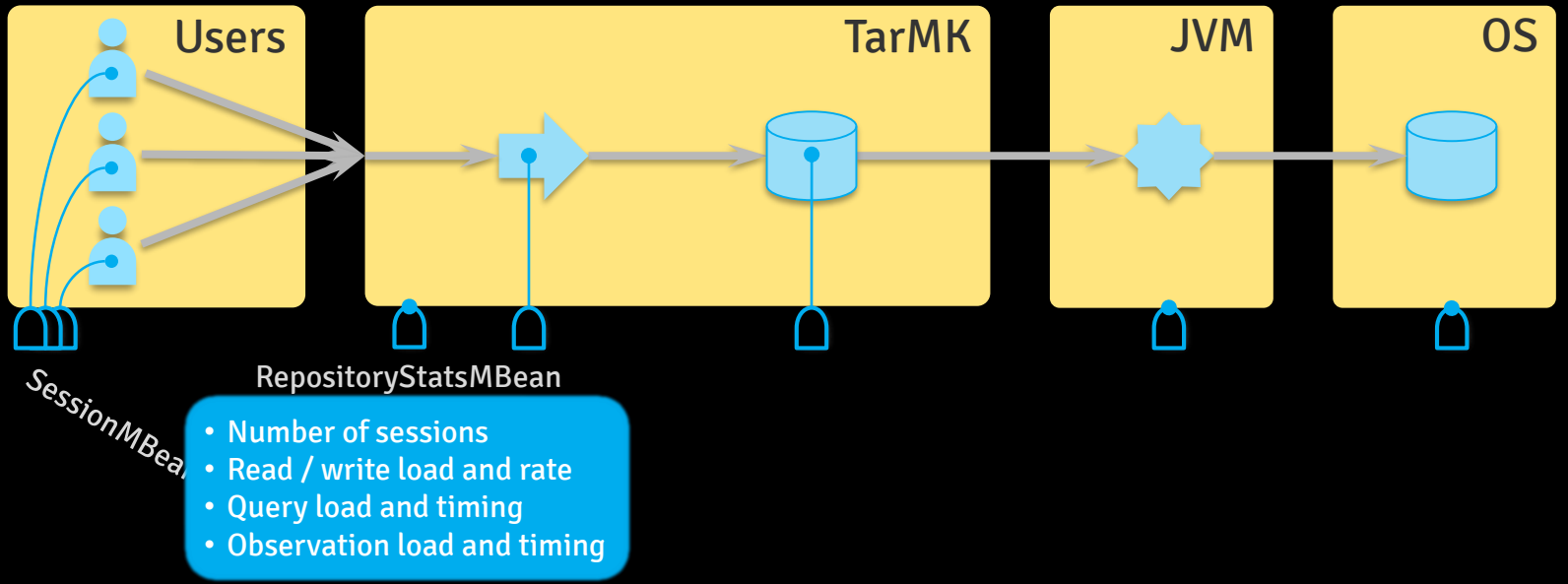


Monitoring

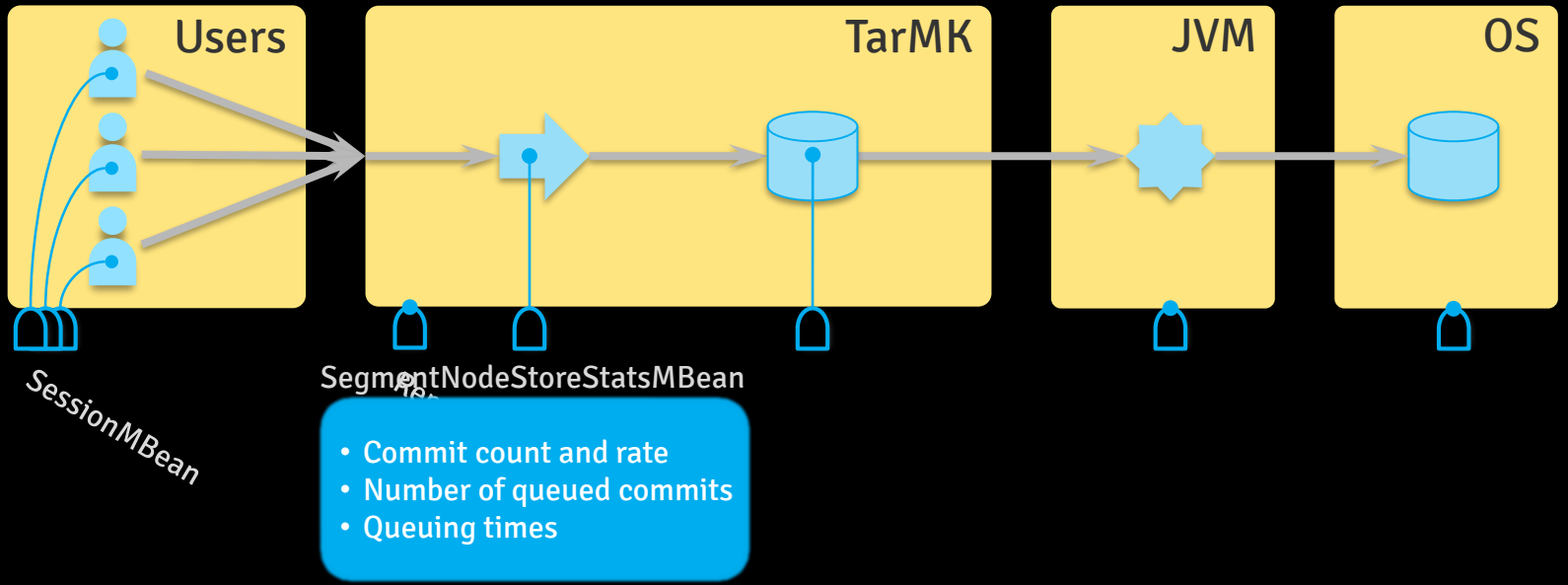


- Read count / rate
- Write count / rate
- Age

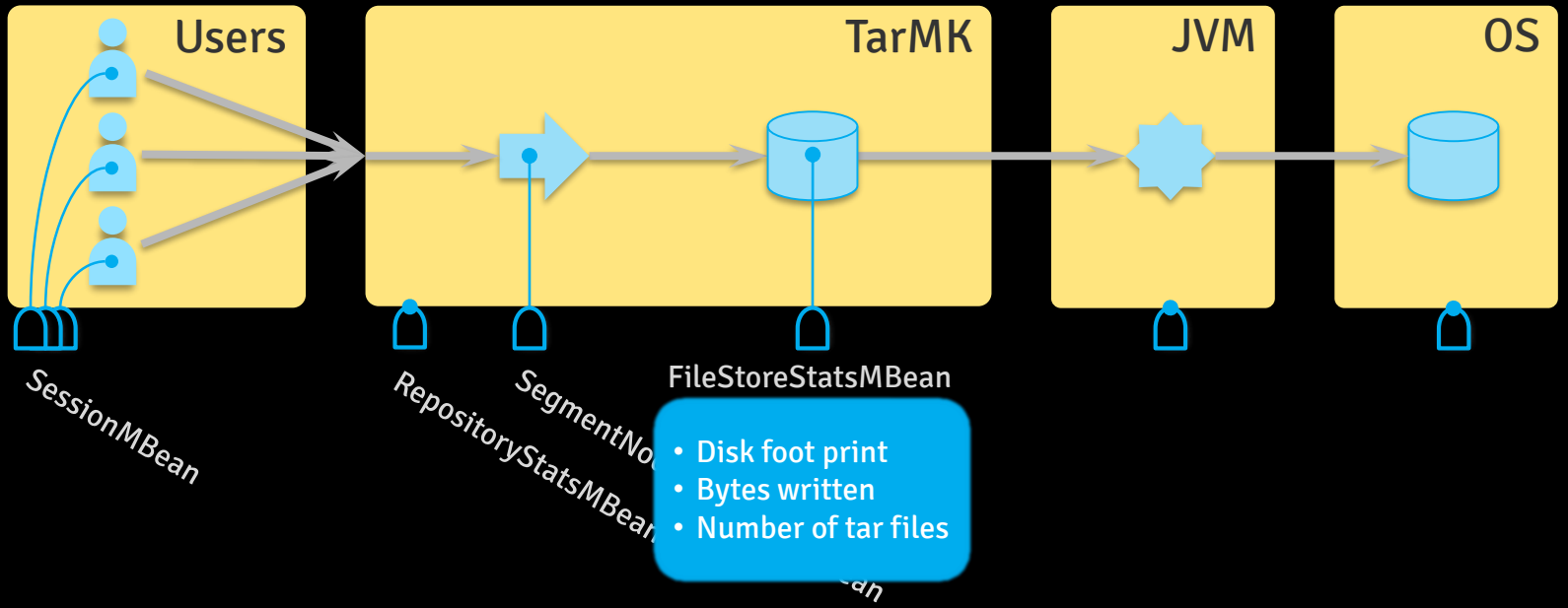
Monitoring


Users
TarMK
JVM
OS
SessionMBean
RepositoryStatsMBean
• Number of sessions
• Read / write load and rate
• Query load and timing
• Observation load and timing

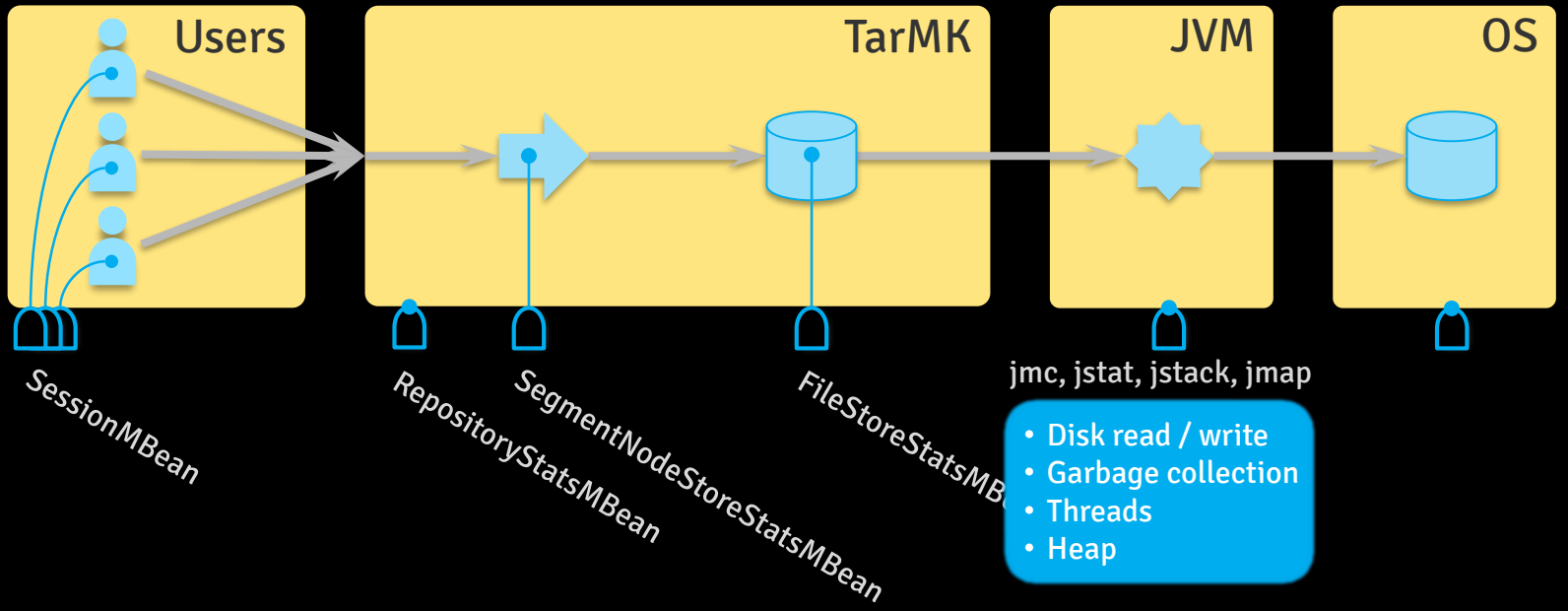
Monitoring


Users
TarMK
JVM
OS
SessionMBean
SegmentNodeStoreStatsMBean
• Commit count and rate
• Number of queued commits
• Queuing times

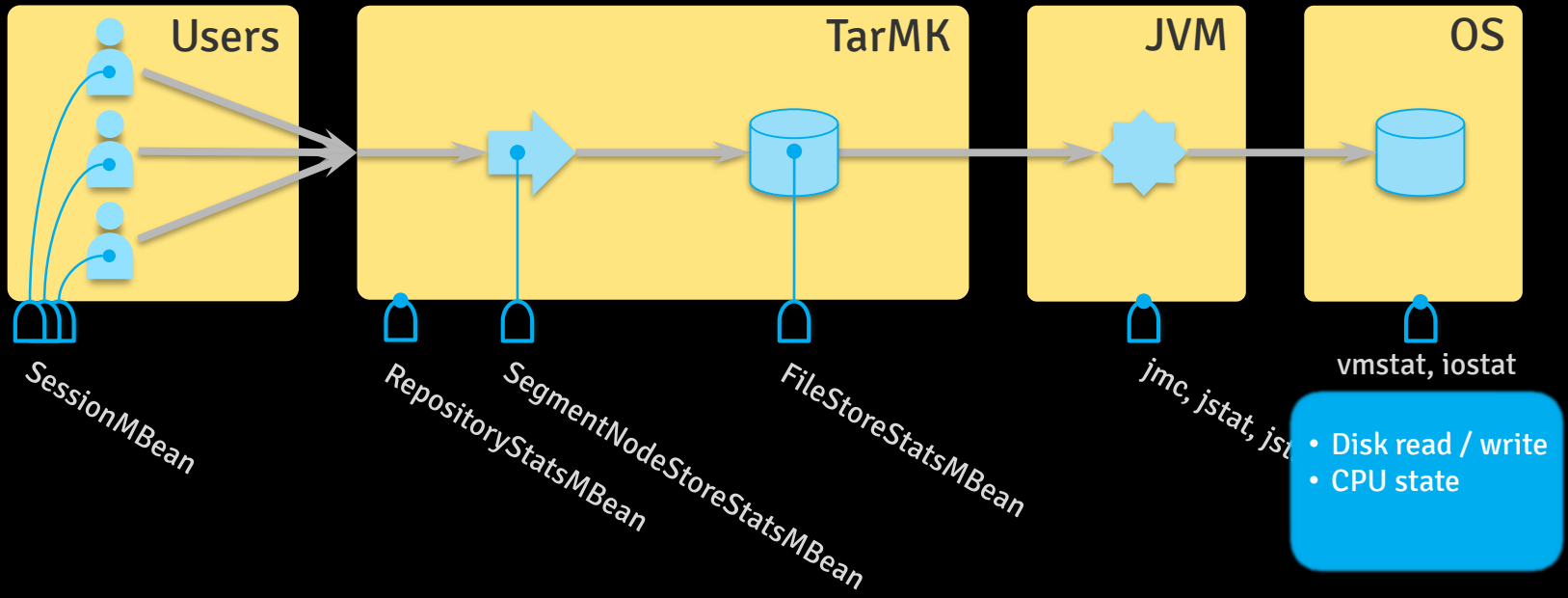
Monitoring


Users
TarMK
JVM
OS
SessionMBean
RepositoryStatsMBean
SegmentNo
FileStoreStatsMBean
• Disk foot print
• Bytes written
• Number of tar files

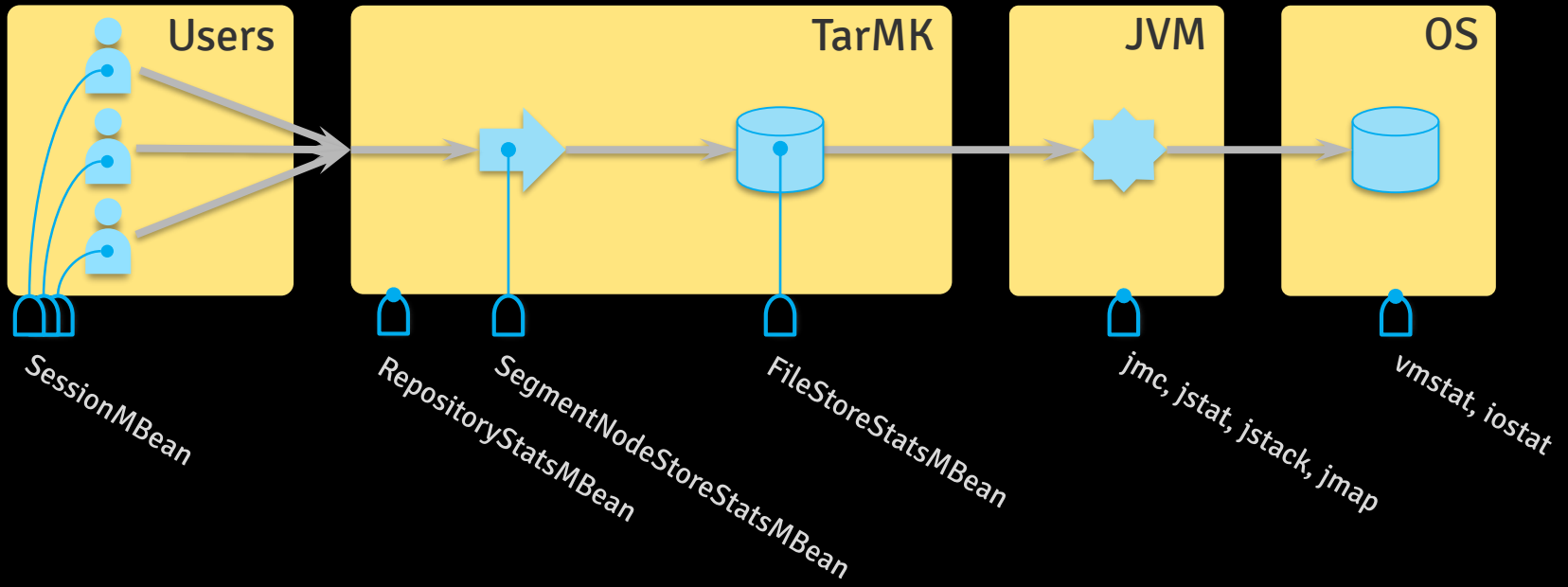
Monitoring


Users
TarMK
JVM
OS
SessionMBean
RepositoryStatsMBean
SegmentNodeStoreStatsMBean
FileStoreStatsMB
jmc, jstat, jstack, jmap
• Disk read / write
• Garbage collection
• Threads
• Heap

Monitoring


Users
TarMK
JVM
OS
SessionMBean
RepositoryStatsMBean
SegmentNodeStoreStatsMBean
FileStoreStatsMBean
jmc, jstat, jst
vmstat, iostat
• Disk read / write
• CPU state

Monitoring


Users
TarMK
JVM
OS
SessionMBean
RepositoryStatsMBean
SegmentNodeStoreStatsMBean
FileStoreStatsMBean
jmc, jstat, jstack, jmap
vmstat, iostat

Case Study: Thrashing

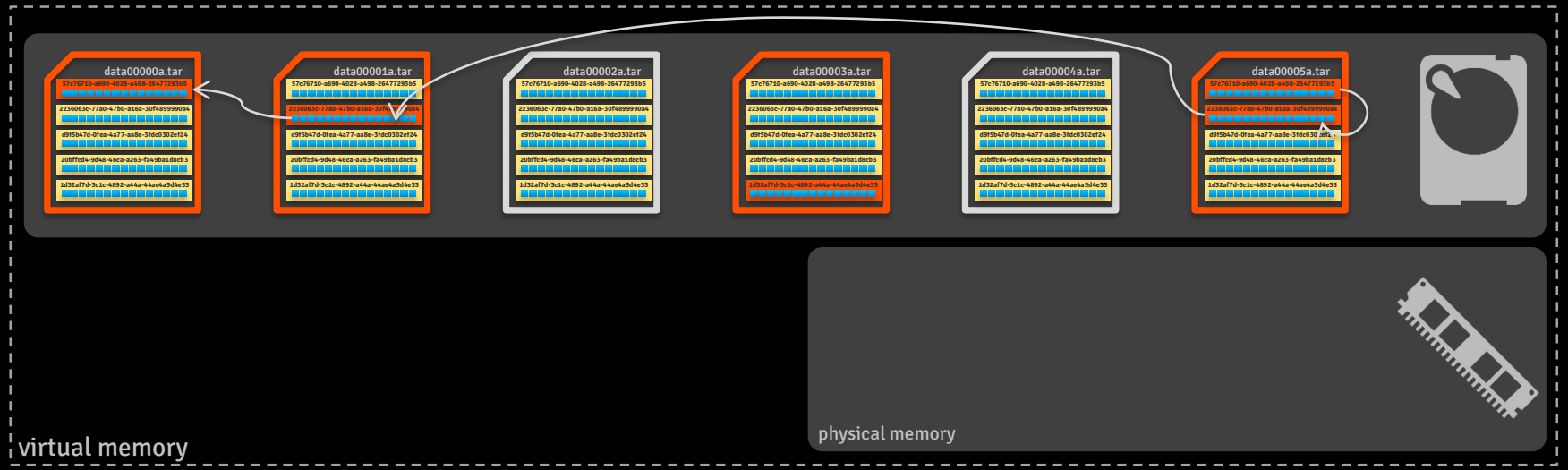
Thrashing

In computer science, **thrashing** occurs when a computer's virtual memory subsystem is in a **constant state of paging**, rapidly exchanging data in memory for data on disk, to the exclusion of most application-level processing. This causes the performance of the computer to **degrade or collapse**.

https://en.wikipedia.org/wiki/Thrashing_(computer_science)

Thrashing in the TarMK

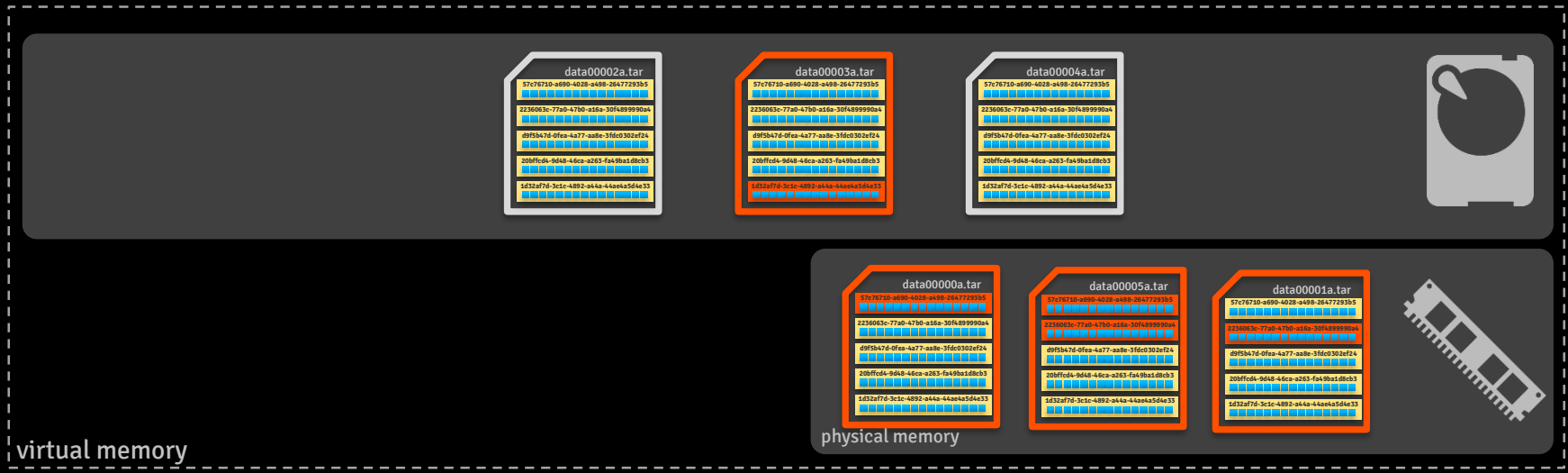
In the TarMK, **thrashing** occurs when the working set of tar files does not fit into system **memory**, so every repository operation leads to **disk access**.



working set

Thrashing in the TarMK

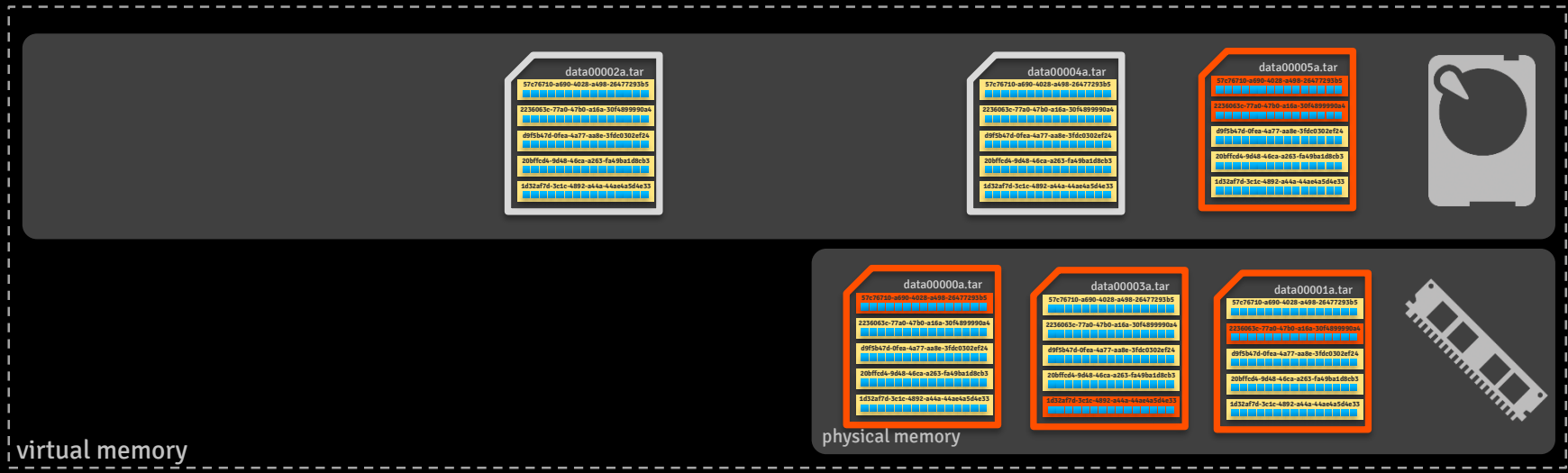
In the TarMK, **thrashing** occurs when the working set of tar files does not fit into system **memory**, so every repository operation leads to **disk access**.



■ working set

Thrashing in the TarMK

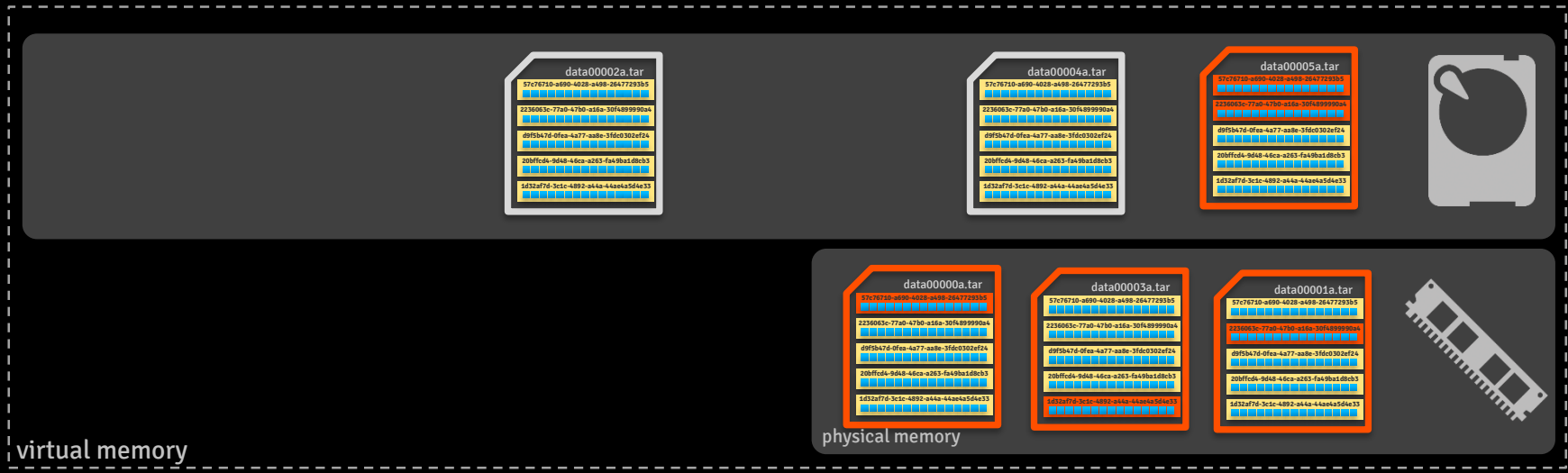
In the TarMK, **thrashing** occurs when the working set of tar files does not fit into system **memory**, so every repository operation leads to **disk access**.



■ working set

Thrashing in the TarMK

In the TarMK, **thrashing** occurs when the working set of tar files does not fit into system **memory**, so every repository operation leads to **disk access**.



■ working set

Test Setup

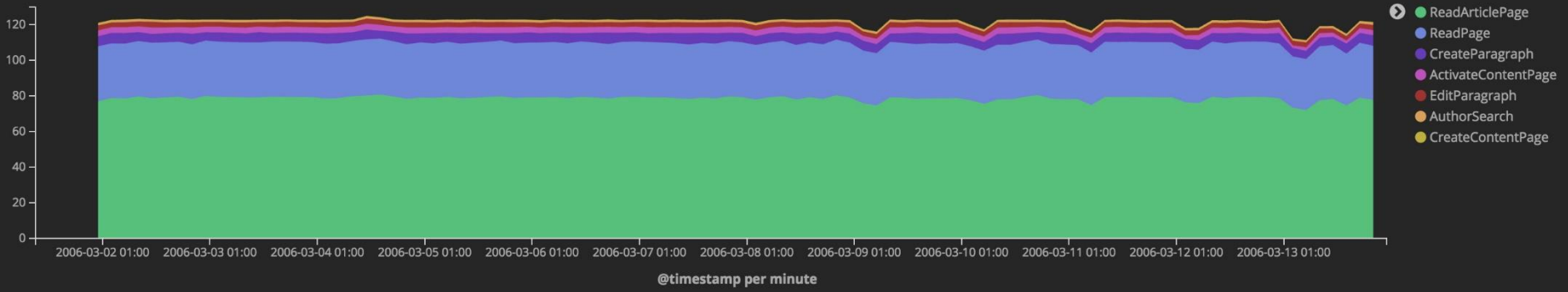
Hardware specs

2 vCPUs

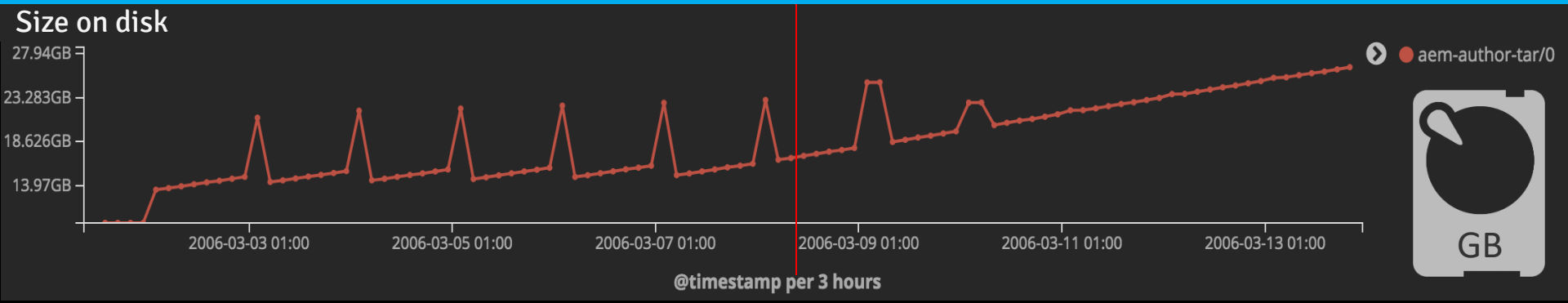
8 GB (4GB Heap)

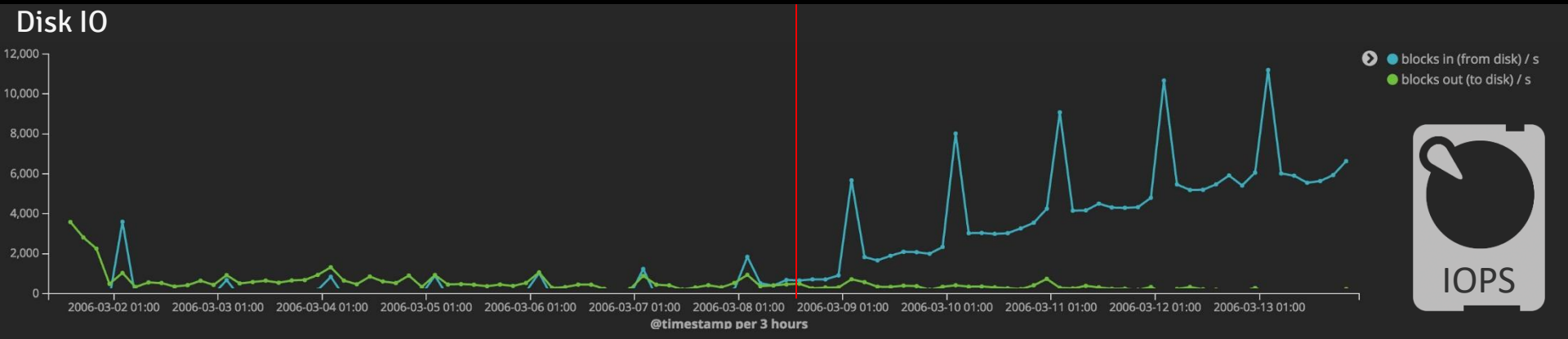
500 GB Magnetic Disk (EBS)

Requests throughput

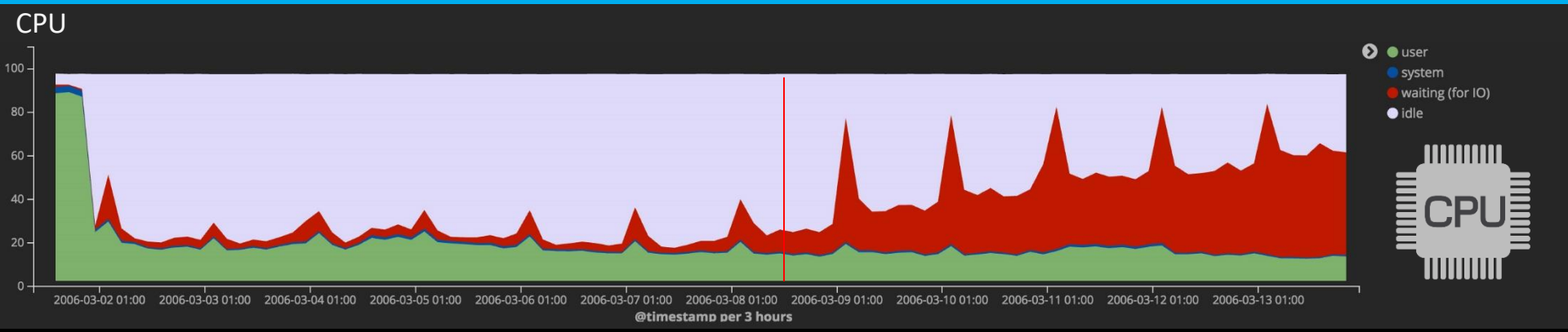


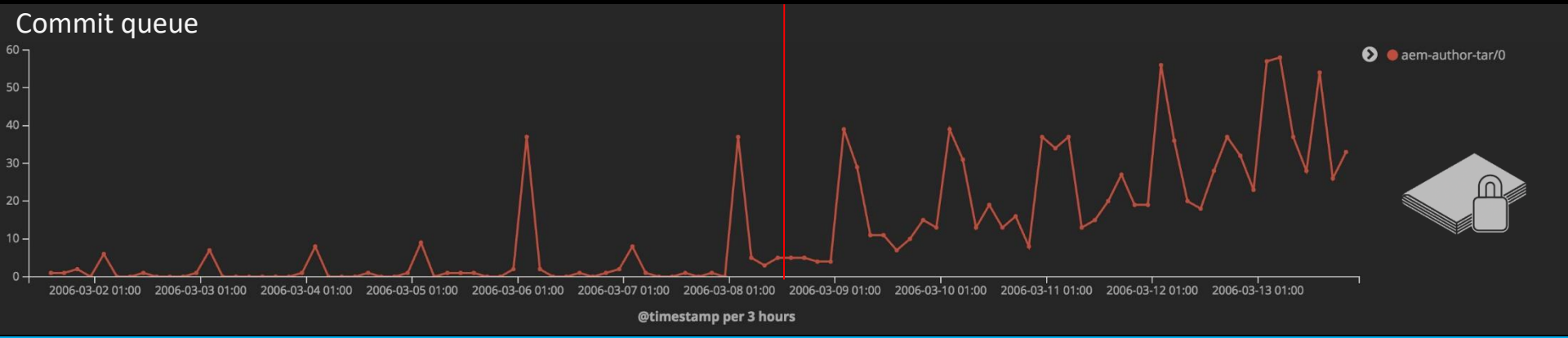
Disk


Size on disk
27.94GB
23.283GB
18.626GB
13.97GB
2006-03-03 01:00
2006-03-05 01:00
2006-03-07 01:00
2006-03-09 01:00
2006-03-11 01:00
2006-03-13 01:00
@timestamp per 3 hours
aem-author-tar/0
GB

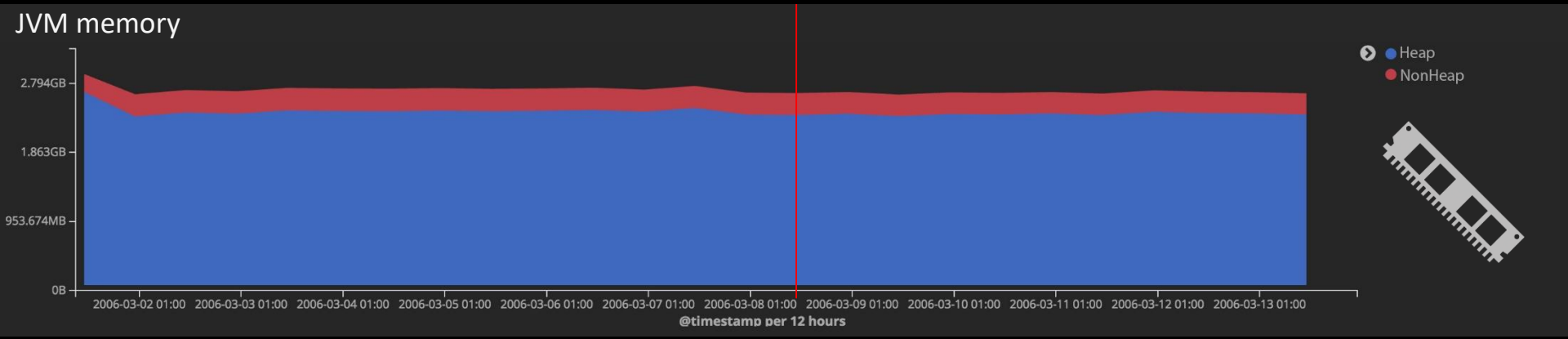

Disk IO
12,000
10,000
8,000
6,000
4,000
2,000
0
2006-03-02 01:00
2006-03-03 01:00
2006-03-04 01:00
2006-03-05 01:00
2006-03-06 01:00
2006-03-07 01:00
2006-03-08 01:00
2006-03-09 01:00
2006-03-10 01:00
2006-03-11 01:00
2006-03-12 01:00
2006-03-13 01:00
@timestamp per 3 hours
blocks in (from disk) / s
blocks out (to disk) / s
IOPS

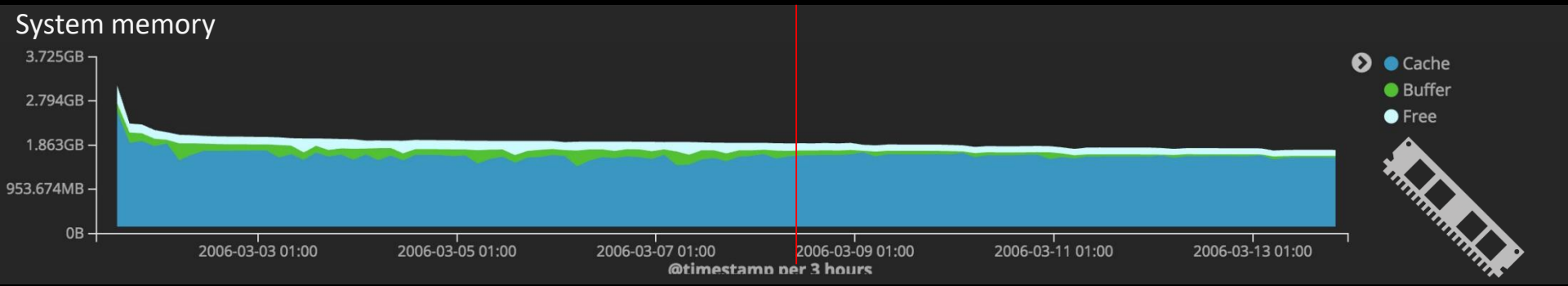
CPU and Commits



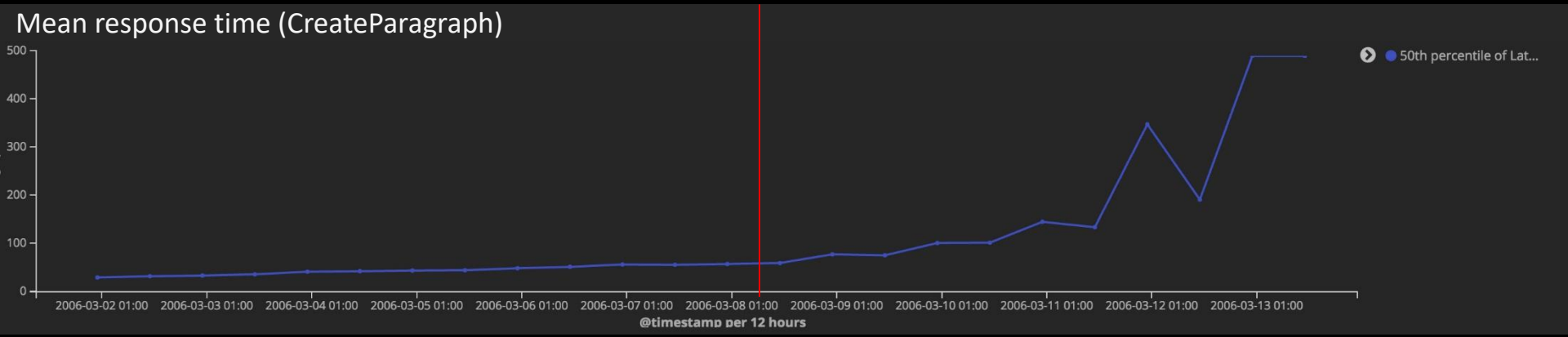


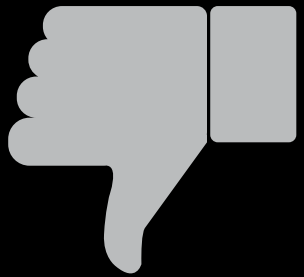
Memory


JVM memory
2.794GB
1.863GB
953.674MB
0B
2006-03-02 01:00
2006-03-03 01:00
2006-03-04 01:00
2006-03-05 01:00
2006-03-06 01:00
2006-03-07 01:00
2006-03-08 01:00
2006-03-09 01:00
2006-03-10 01:00
2006-03-11 01:00
2006-03-12 01:00
2006-03-13 01:00
@timestamp per 12 hours
Heap
NonHeap


System memory
3.725GB
2.794GB
1.863GB
953.674MB
0B
2006-03-03 01:00
2006-03-05 01:00
2006-03-07 01:00
2006-03-09 01:00
2006-03-11 01:00
2006-03-13 01:00
@timestamp per 3 hours
Cache
Buffer
Free

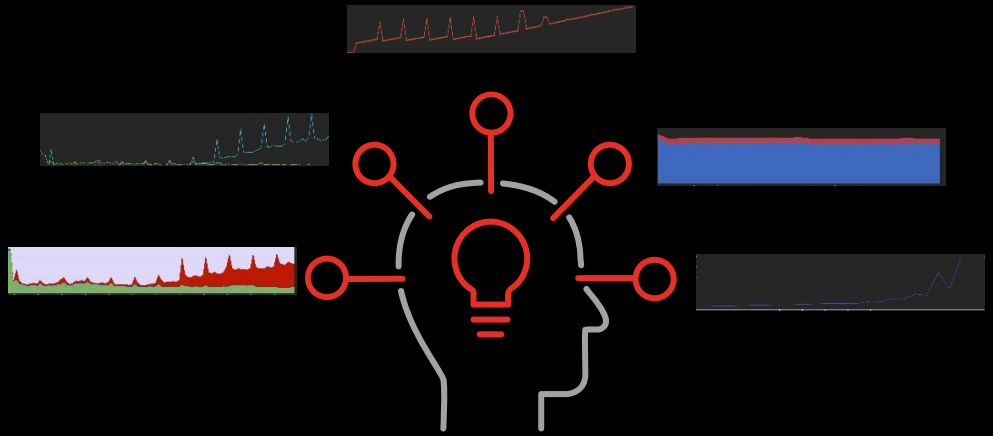
Response Times





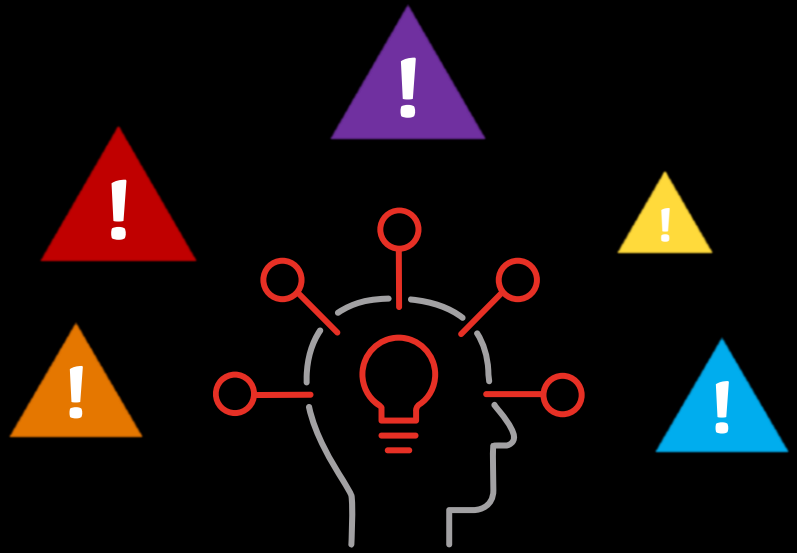
adaptTo()

Now What?



adaptTo()

Now What?



Now What?

NOW WHAAAAAAAT?!

Now What?

1.

NOW WHAAAAAAAT?!

Now What?

NOW WHAAAAAAAT?!

1. 

2. Qualify the problem

Now What?

NOW WHAAAAAAAT?!

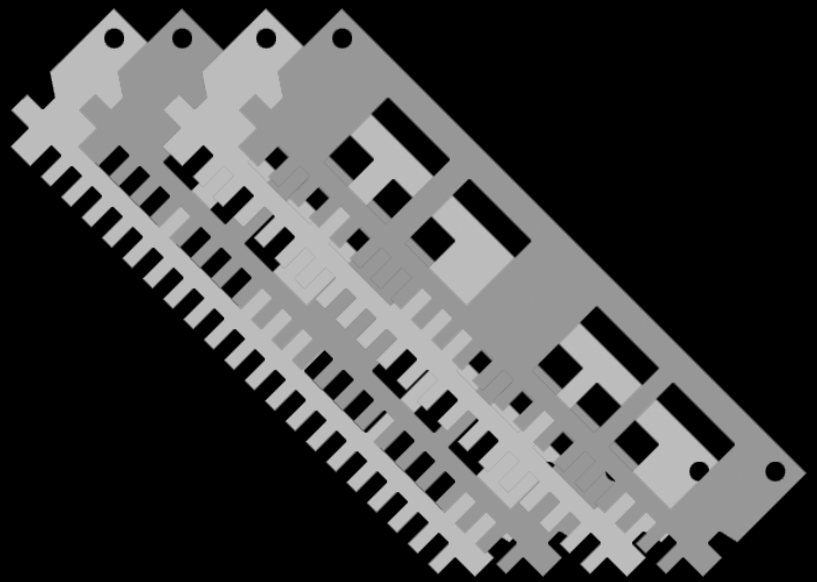
1. 

2. Qualify the problem

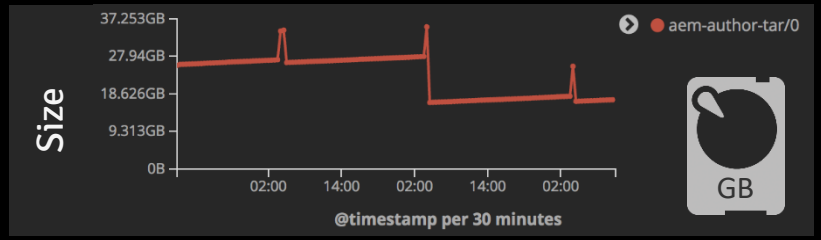
3. Take prompt actions

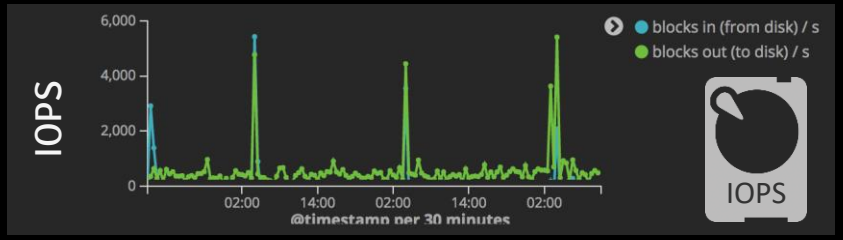
Potential actions (1)

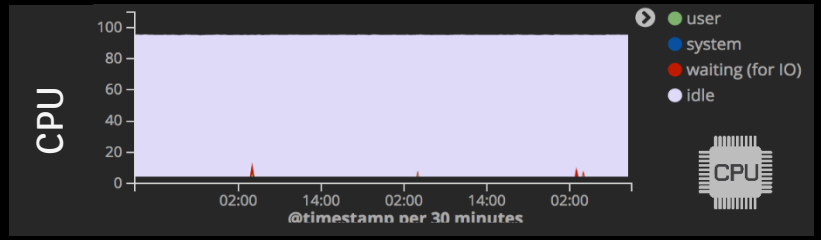
1. Upgrade hardware
   - Add RAM
   - Optimize IO

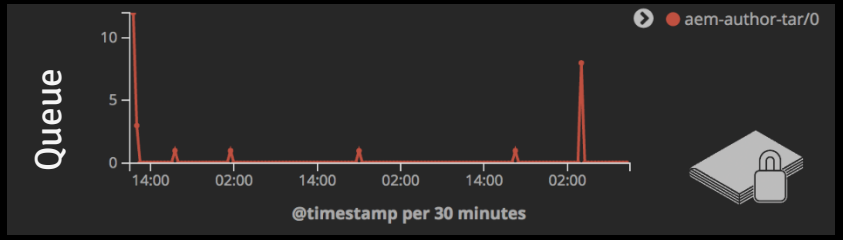


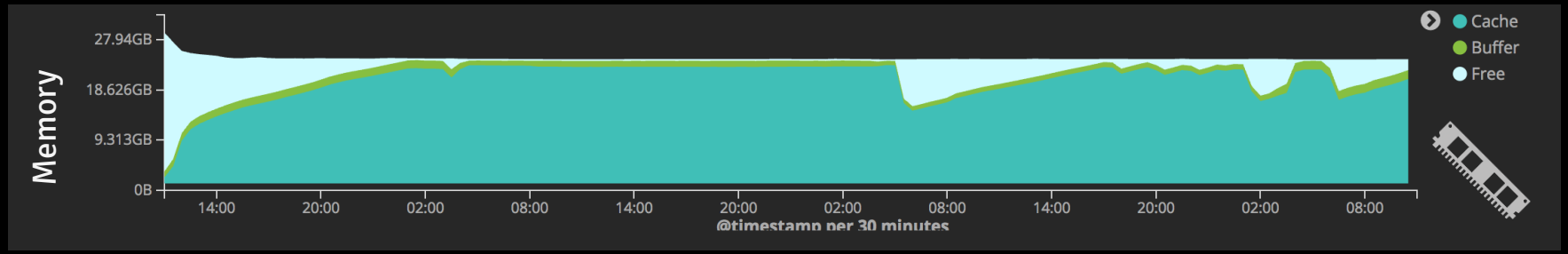
A1: Upgrade (increase RAM to 32GB)









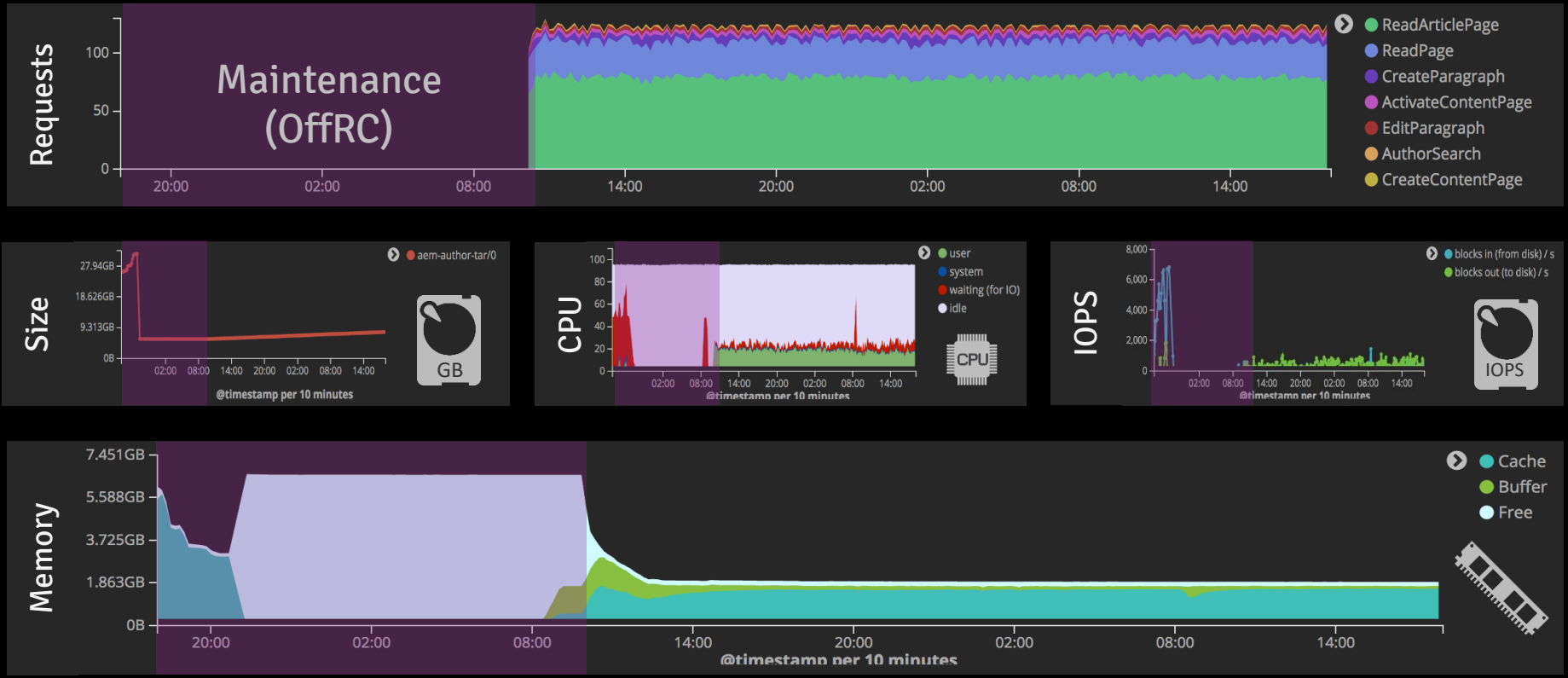


Potential actions (2)

2. Reduce repository

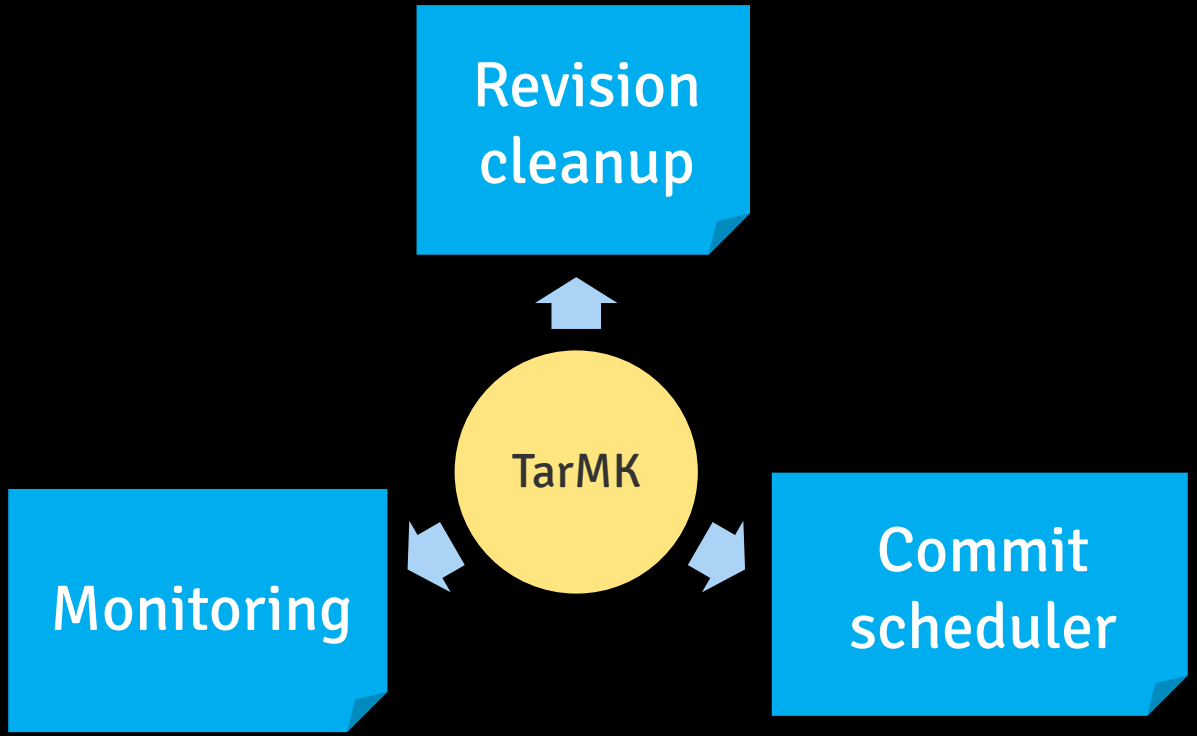
- Use a blob store
- Manage inactive content (Content hygiene)
- Optimize indexes

A2: Cleanup content (and offline revision cleanup)



Outlook

Areas of Improvement



Thank you

Questions ?

Appendix

Typical Segment Store Composition