Core Architecture

Sakai Kernel Progress towards Sakai 2.6 and Performance Improvements

Dr Ian Boston
CTO,
CARET, University of Cambridge

A Technical Presentation

- The Sakai Kernel
- Performance enhancements in 2.5
- Improvements for 2.6
- Longer term
- Questions

The Sakai Kernel

Why Architecture Matters

- Creates Conflict
- Good Architecture makes a community work and attracts developers
- Modular architecture is vital to a communities growth

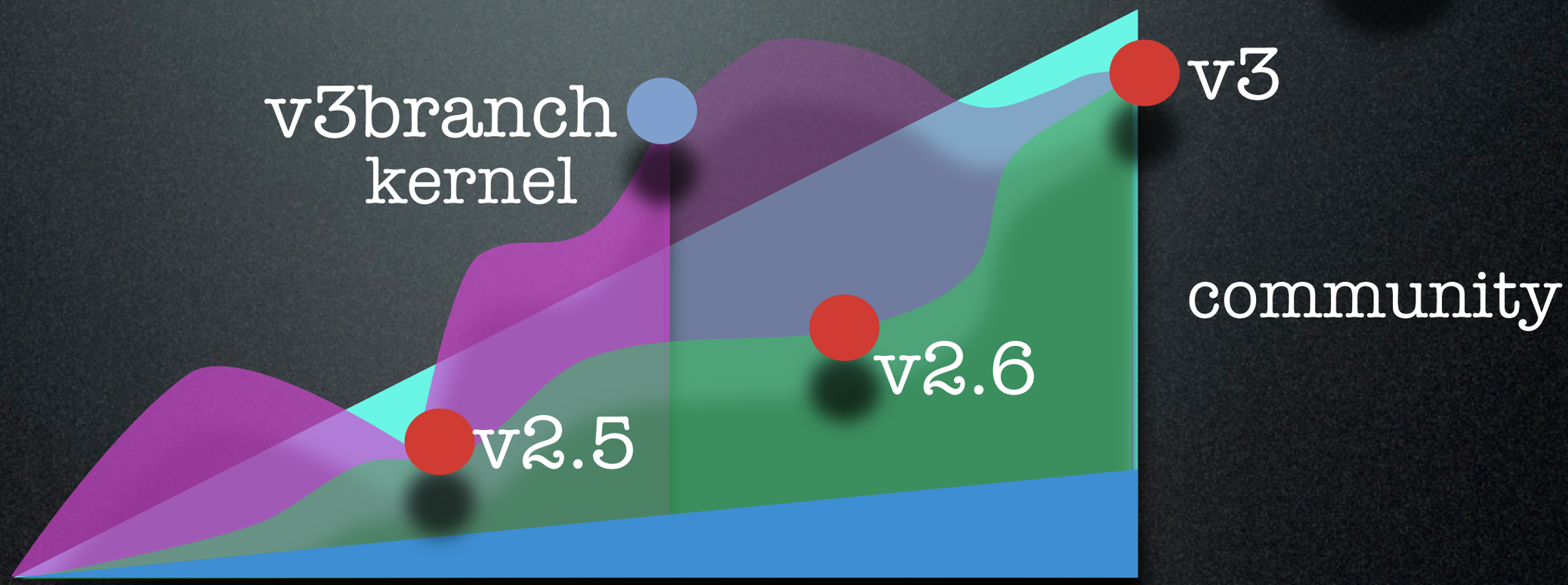
“architecture is the eternal flame. Nobody is every happy with the architecture; and since it tends to be an immovable object complaining about it is a low risk activity. One is unlikely to be called on it; e.g. "put up or shut up "” Ben Hyde, MIT

If its not broken,

- Modular Architecture
- Areas of Concern
- Volume of Code
- Standards
- Scalability

"Sakai has a reasonably modular architecture that has allowed teams of developers to work independently, its not perfect, it has problems, so lets not fix what is not broken?"

Next Version Development



Loosing the community

What is 

- A target to aim for.
- Achievable

- 4000 concurrent users per 32bit app server. 2Gb Heap
- Mobile Sessions
- Easier to reload applications
- Testable Applications
- Sensible URL Space
- no tool iframes

The “Kernel” now

Tools

resources alias realms portal site user

access login reference courier site man master

API

content alias authz portal site user

import event email courier site man jcr

jobschedule tool db component cluster cm

Pack

content alias authz portal site user

import event email courier site man velocity

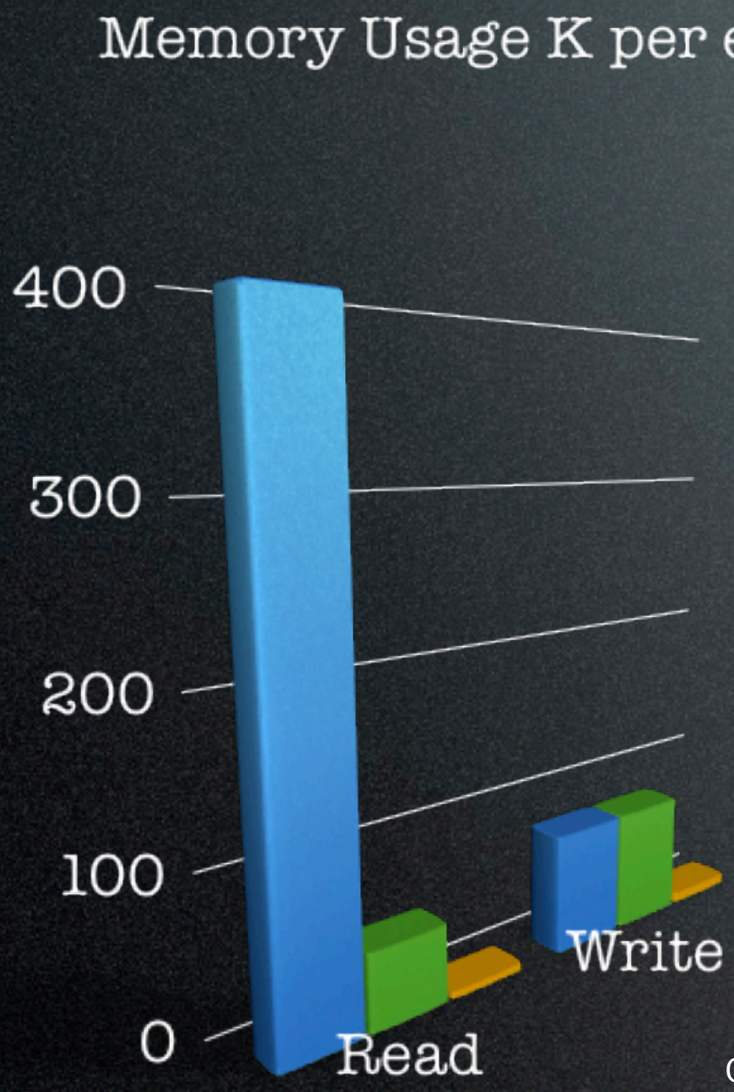
jobschedule tool db component cluster util

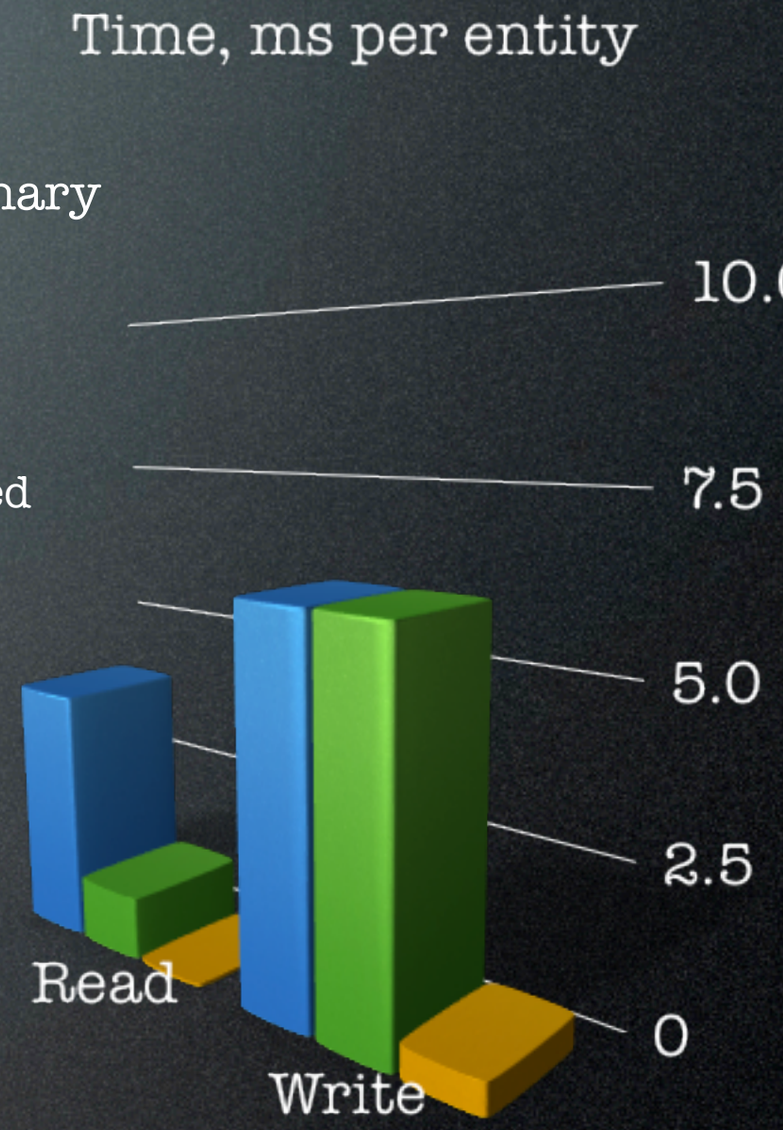
Performance enhancements in 2.5/2.4x

Bottlenecks

- Serialization
- Queries

Serialization





DOM SAX Binary

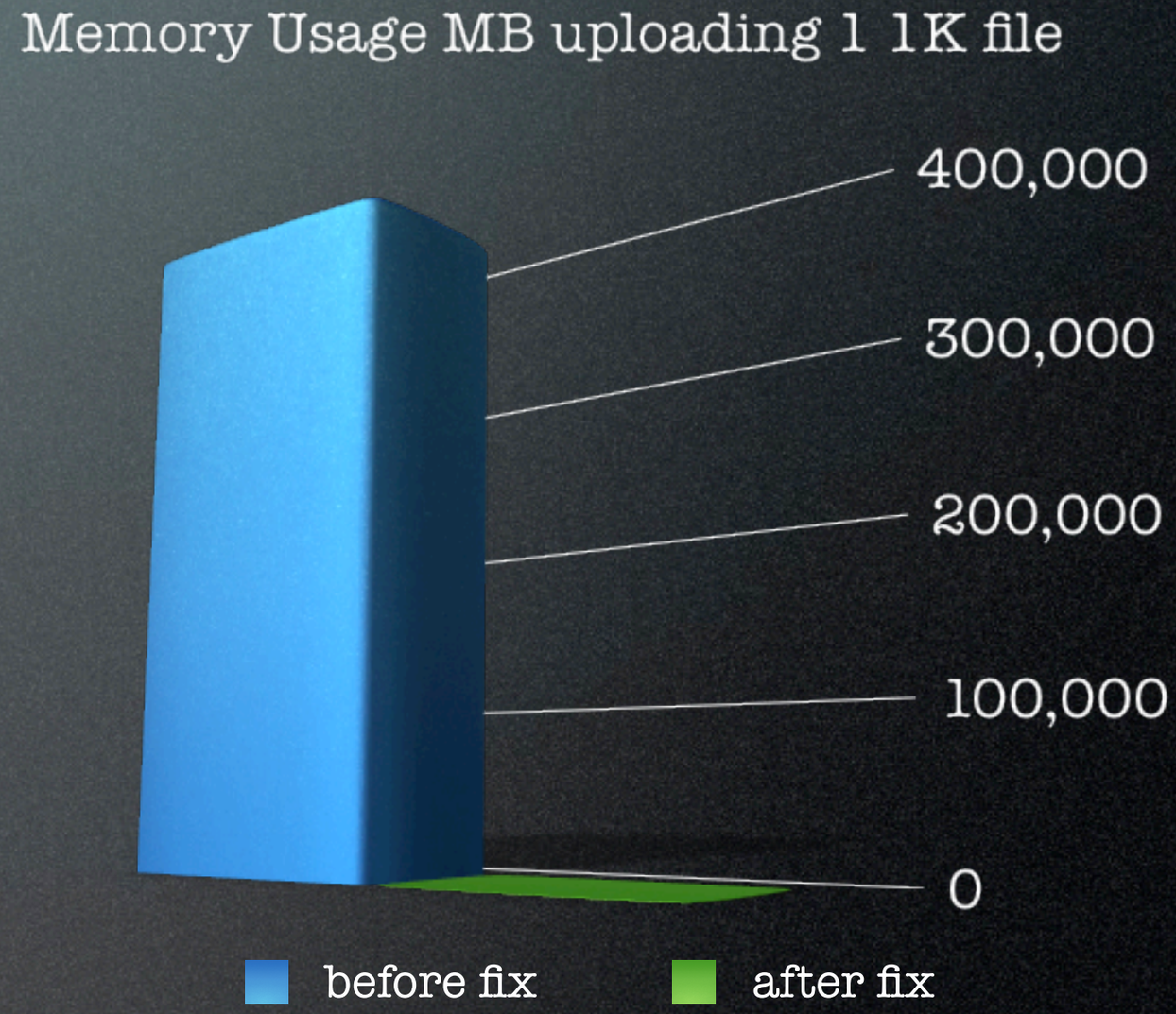
- Core parts of Sakai use XML
- pre-2.5 serialization DOM based
- Quota calculation scan.
- 1 1k upload == 400MB GC.
- Better Serialization

Original DOM: 3-4ms per entity, 400K heap
SAX: 1-2ms per entity, 50K heap
New parser: 0.0630ms 4.681 K,

For writing to the DB we have only DOM and the New parser.
DOM: 6ms 80K
New Parser
0.0493ms 6.768 K

Quota Calculation

- To calculate the quota, load all resources in the site, add up.
  - load == parse xml with DOM
- moved size into column and sum in the DB.



Before 400MB, After 400K, Still using DOM with binary 4K

Calendar

- Loaded all events in a site
- DOM Serialization
- Added Range Query to limit number loaded
- Converted to SAX Serialization

Repent

- These problems have existed since 1.0

- Why didn't anyone fix them earlier ?

"architecture is the eternal flame. Nobody is every happy with the architecture; and since it tends to be an immovable object complaining about it is a low risk activity. One is unlikely to be called on it; e.g. "put up or shut up ""

- Why didn't I submit a patch ?

Improvements for 2.6

2.6 Kernel Roadmap

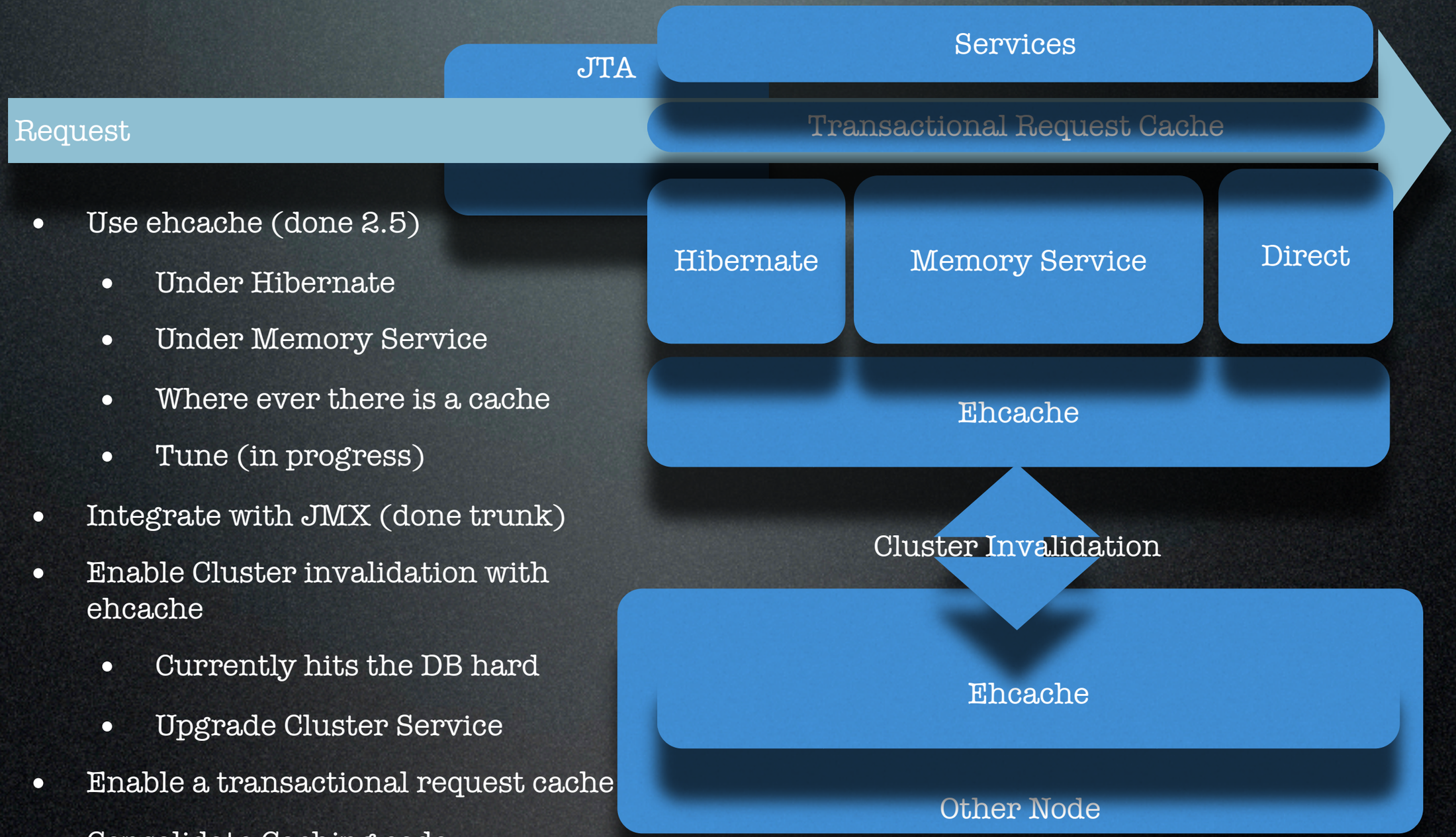
- Fix other Serialization issues
- Caching
- Events
- Tuning
- Memory Footprint
- Portal

Serialization Issues

- Convert to Binary
- Convert to Flat ie Columns

Caching

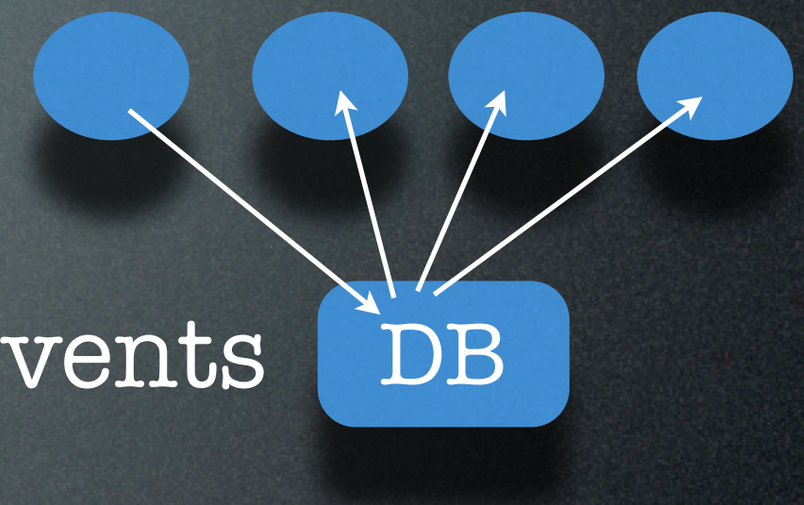
- Use ehcache (done 2.5)
  - Under Hibernate
  - Under Memory Service
  - Where ever there is a cache
  - Tune (in progress)
- Integrate with JMX (done trunk)
- Enable Cluster invalidation with ehcache
  - Currently hits the DB hard
  - Upgrade Cluster Service
- Enable a transactional request cache
- Consolidate Caching code



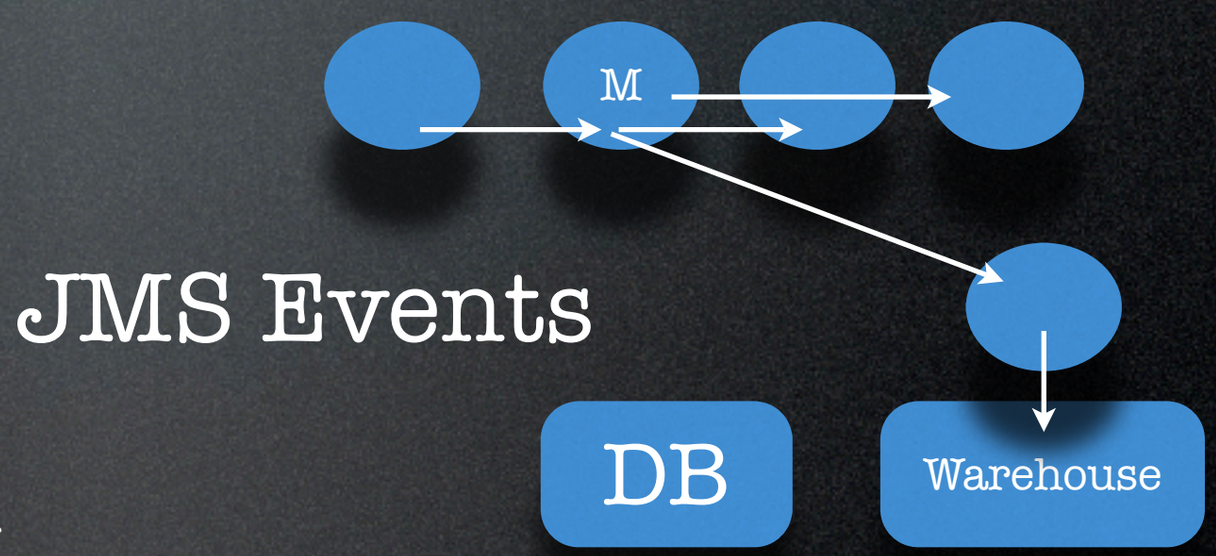
JSR-107 the caching API was never ratified and is not widely supported. ehcache does support it, but we probably wont bind to it.

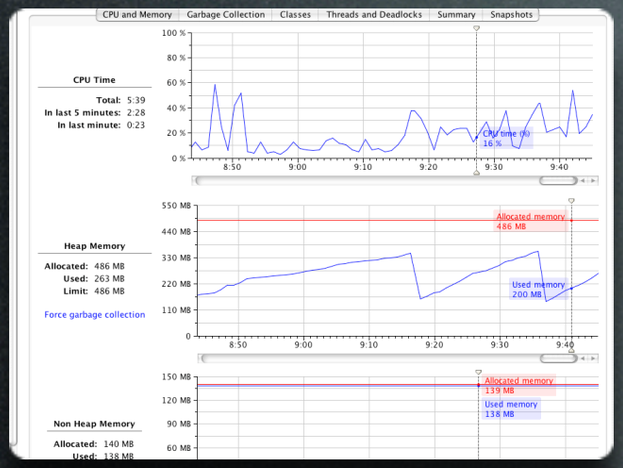
Events

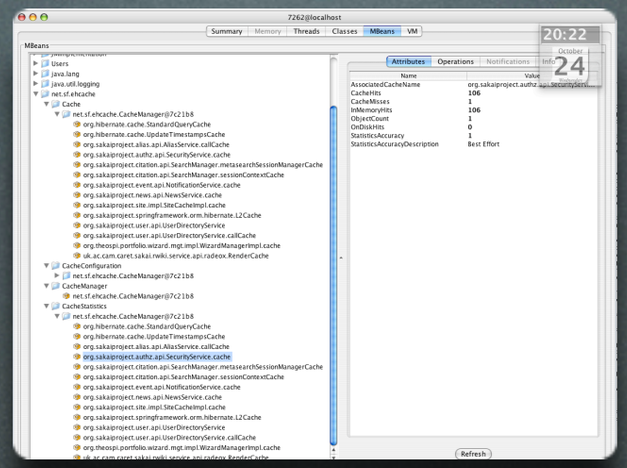
- Events table is expensive.
- Events are not bound to transactions correctly
- Propagation requires central DB scaling

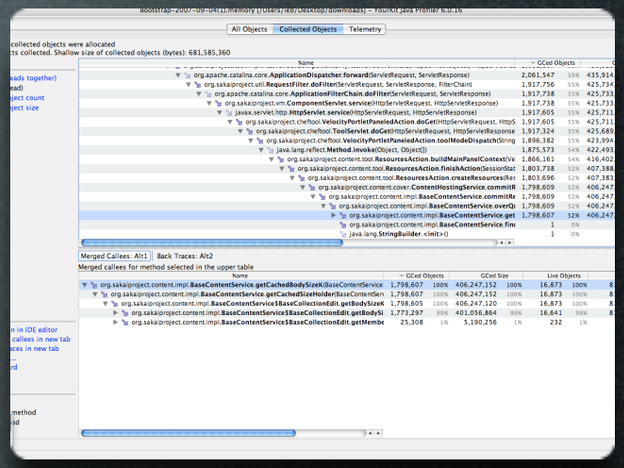


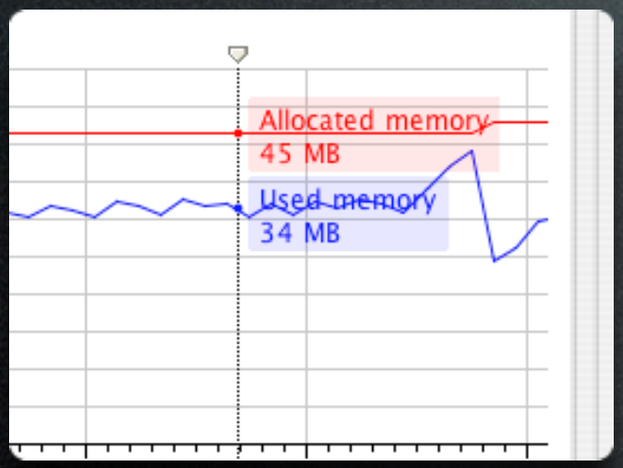
- Replace with JMS
  - ActiveMQ
  - Use local brokers with local storage in a master election cluster.
  - One subscriber to warehouse events if required.
  - Bind to JTA
    - may require replacement of current transaction manager.

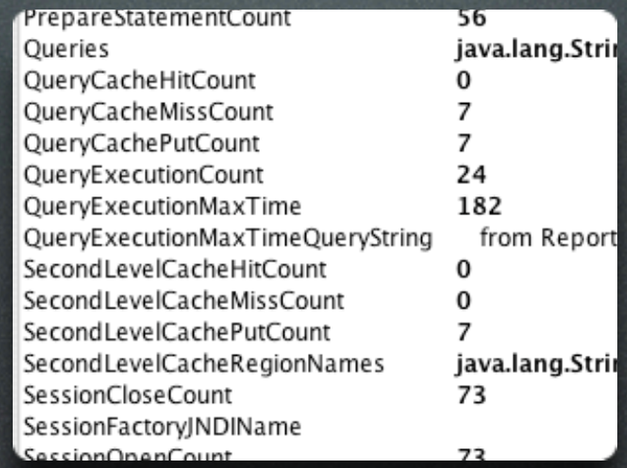


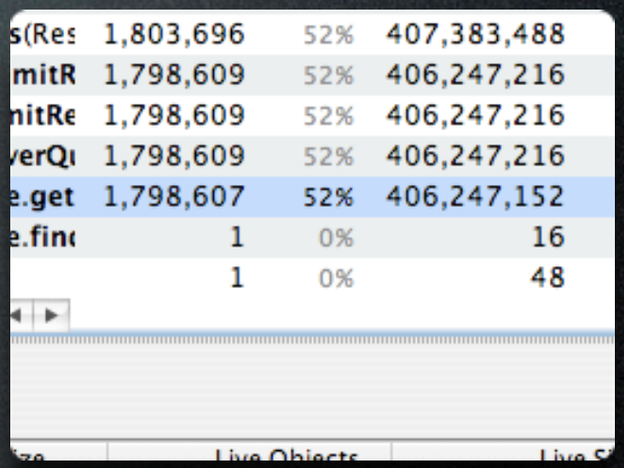












Tuning and Monitoring

More Profiling, Monitoring, load testing and Forensic analysis. Measurable evidence is required to make improvements.

Memory

- Session Footprint
  - 1.4G/4000 = 350K/User
- Request Footprint
  - Requests now 100M GC
  - Should be 1M
- Base Requirements
  - Perm Space
    - too many jars, too many copies.
  - Heap
- Bloatware ?

- Session Footprint
  - Investigate Session usage
- Request Footprint
  - Profile and reduce, caching
- Base Requirements
  - Investigate classloader structure and component manager
- We should not make 64bit a requirement to run Sakai in production.

"In 1999 tuning an early non trivial webapp, we tuned the application and got 1000 users in 512MB on a PIII 366MhZ Linux box, we don't need 64bit to service users"

Portal

- Make Sakai behave like a normal web application.
    - Make Back button work
    - Make URL's make sense
    - deprecate iframes for tools.
- Cleaner Entity Broker URL's

"Strictly, the idea of a uniform syntax for global identifiers of network-retrievable documents was the core idea of the World Wide Web. In the early times, these identifiers were variously called "document names", "Web addresses" and "Uniform Resource Locators". These names were misleading, however, because not all identifiers were locators, and even for those that were, this was not their defining characteristic. Nevertheless, by the time the RFC 1630 formally defined the term "URI" as a generic term best suited to the concept, the term "URL" had gained widespread popularity, which has continued to this day" Wikipedia

Integrate JSR-170

- Generalized Content Store for storing anything that is content.
- Jackrabbit.

Startup/Testing/ Developing

- Improve Component Manager.
- Remove sensitivity to startup order.
- Make better use of classloaders
  - Investigate JSR-277, Classworlds, Plexus

Longer term

Mobile Sessions

- Reduce Session Size
- Standardize Session
- Make it mobile
  - eg Terracotta

Scale Up DB

- Tune Cluster Caches
  - Store more in the app server (a reason to go to 64bit)
  - Reduce Load on the DB, fewer unnecessary low performance reads
  - Enable Read only snapshots

Portal

- Make everything URL addressable
- All sorts of Accessibility fixes. Input from Fluid

Thank you and Questions