Infinispan

Galder Zamarreño
Infinispan Core Developer
JBoss by Red Hat

jboss.org

Who am I?

- Core developer for Infinispan and JBoss Cache
- Contributor and committer on JBoss AS, JGroups, Hibernate, JBoss Portal, etc.

jboss.org

Agenda

- What is Infinispan?
- Relationship with JBoss Cache
- New features
- Demo

jboss.org

What is Infinispan?

- Highly scalable data grid platform
  - 100% open source licensed (LGPL)
  - Based on some JBoss Cache code
- JBoss Cache = Tree structured cache
  - Replicated using JGroups
  - Supports JTA, evictions, cache stores, etc.
- New JSR-107(JCACHE) compatible API
  - Cache extends Map
  - Tree adapter API available for legacy apps

jboss.org

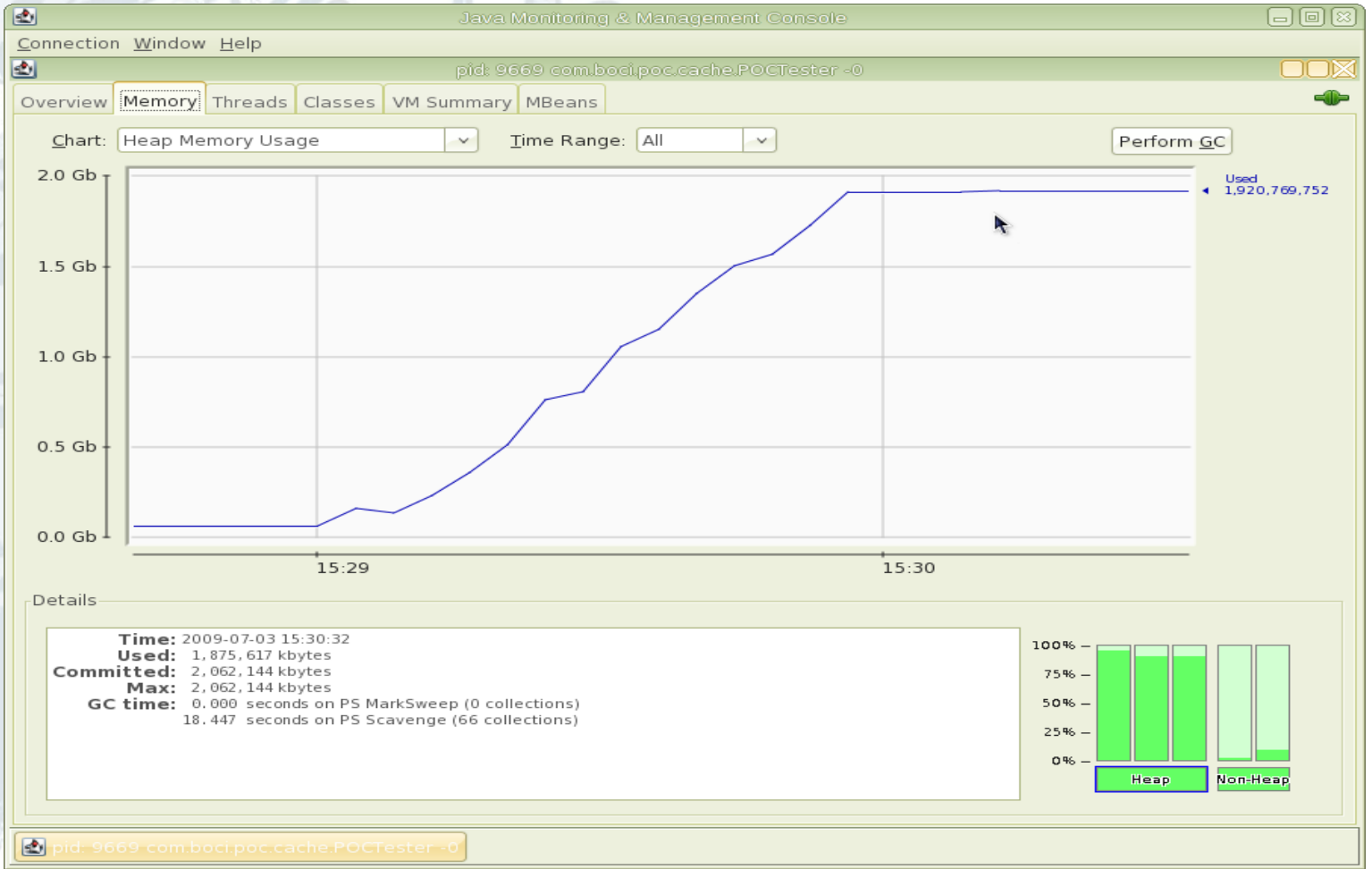
More scalable than JBoss Cache

- Internal structures more memory efficient
  - Tree --> Flat concurrent map
  - Eviction queue --> Ordered container
- Marshalling based on JBoss Marshalling
  - Smaller payloads + Poolable streams
- Early benchmarks
  - Significant performance improvements

jboss.org

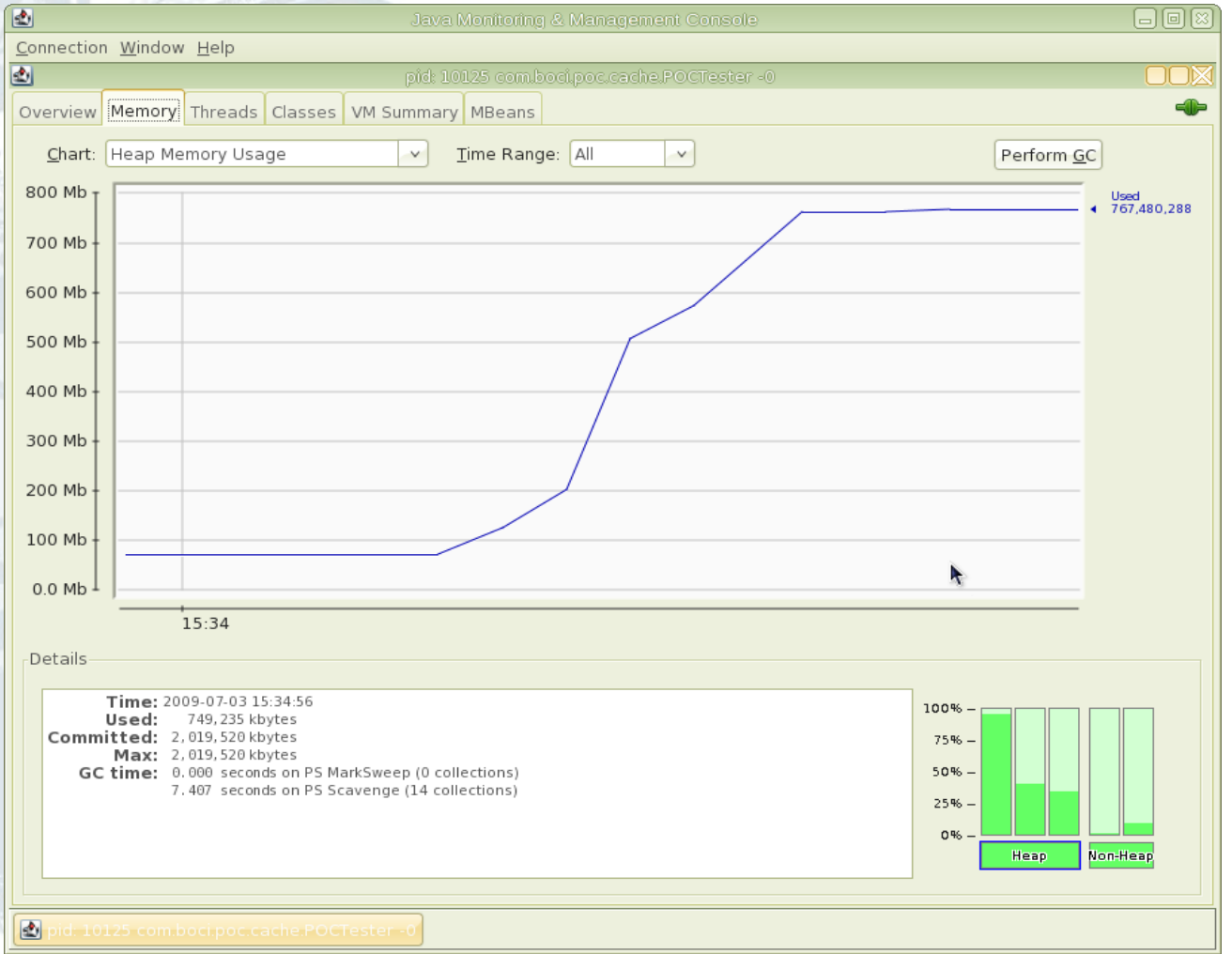
Memory Consumption Comparison

- Test: Put 2 million serial objs into cache
- JBoss Cache 3.1: With 2gb, 1 million objs



Memory Consumption Comparison (2)

- Infinispan 4.0.0.Alpha4: With 700mb, 2 million objs



...borrowing best bits from JBoss Cache

- Multiversion Concurrency Control (MVCC)
  - New locking strategy in JBoss Cache 3.0
  - Readers never locked!
  - Writers work on copy of cache entry
- Non-blocking state transfer
  - Senders generate state without stopping
  - Crucial when state is large

jboss.org

New features - distributed cache

- Consistent hash based distribution
  - Will allow us to scale to bigger clusters
- Lightweight, L1 cache for efficient reads
  - On writes, L1 invalidated
- Dynamic rebalancing
- Pluggable consistent hashing algorithms
- Already available in 4.0.0.Alpha5!

jboss.org

New features - asynchronous API

- putAsync(), putIfAbsentAsync()
  - Do not block, return a j.u.c.Future
  - Future.get() blocks till call completes
- Best of both sync and async worlds
  - Future.get() provides sync guarantees
  - Greater parallelism
- Already available in 4.0.0.Alpha5!

jboss.org

New features - Eager locking

- By default: locks acquired at commit time
  - Problematic if updating a shared counter
- New: Acquiring locks eagerly in cluster
  - Explicit: via API
    - `cache.lock(k) // acquire cluster wide lock on k`
  - Implicit: via configuration
    - Each modification implicitly acquires cluster wide lock if not already held.
- Already available in 4.0.0.Alpha5!

jboss.org

New features - client/server module

- Server module = cache wrapper over TCP
- Client module = cache proxy
- Highly pluggable!
  - Transport: XNIO, Netty, etc.
  - Protocols: memcached, custom, etc.
- Failover and load balancing
- Usable with current memcached clients
- Drop-in replacement memcached servers

New features - fine-grained model

- Successor to POJO Cache
- JPA-like interface: persist, find, remove...
- Will not rely on AOP, javassist...etc
  - More robust and easier to use/debug

jboss.org

New features - Others

- Query module
  - Execute Lucene queries against cache
  - Based on JBoss Cache - Searchable
- Distributed executors
  - Runnable/Callable executed on data set
  - Moves code, not data, around cluster

jboss.org

The End

- Demo
- Questions?

jboss.org