A memcached implementation in Java

Bela Ban
JBoss
2340

JAZOON09
THE INTERNATIONAL CONFERENCE ON JAVA TECHNOLOGY
JUNE 22–25, 2009 ZURICH

AGENDA

- Introduction
- memcached
- memcached in Java
- Improving memcached
- Infinispan
- Demo

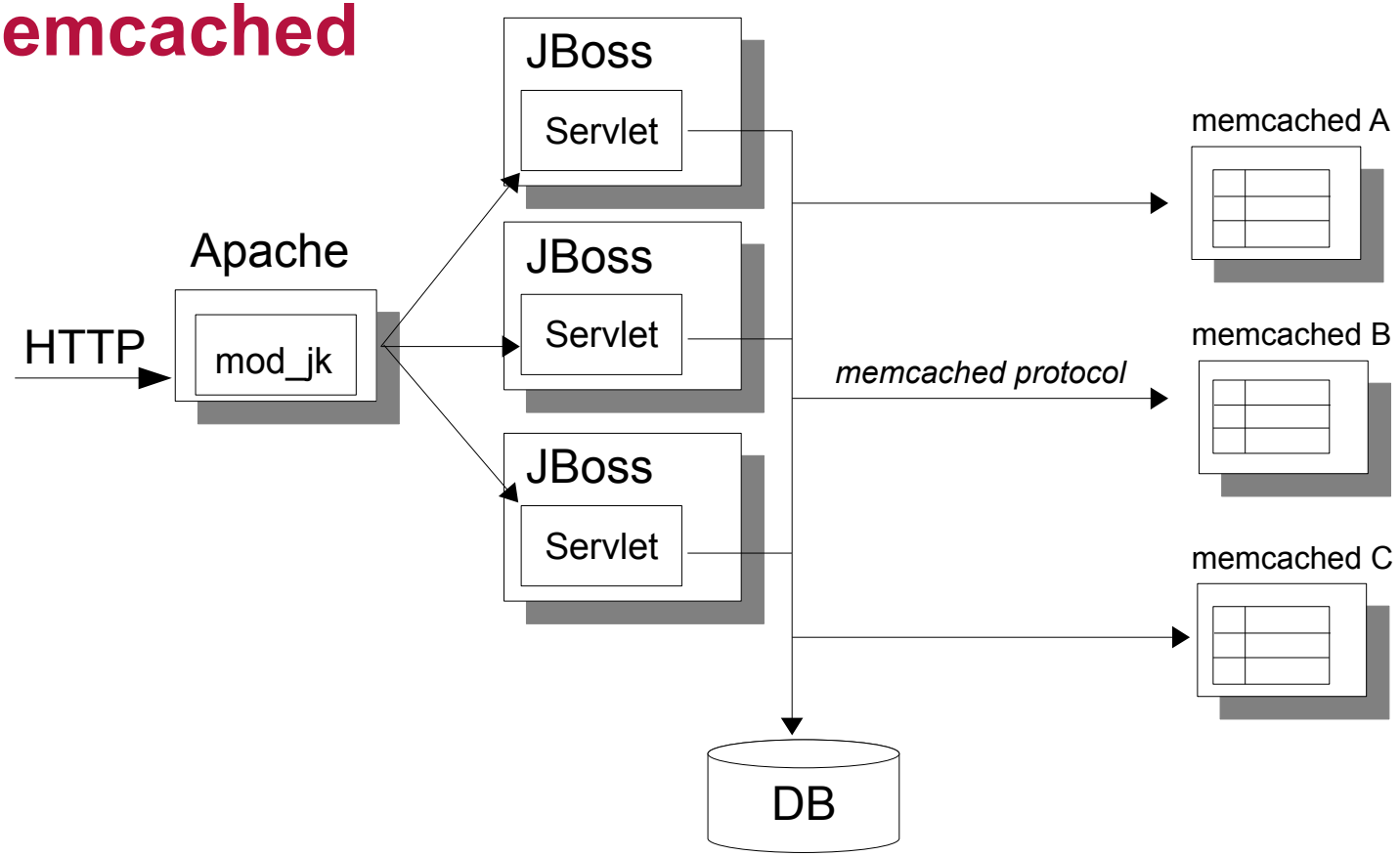
Introduction

- We want to store all of our data in memory
  - "Memory is the new disk, disk is the new tape" (Tim Bray)
  - Reducing DB access speeds up the application
    - ➔ A network round trip is at least an order of magnitude faster than accessing the disk
    - ➔ The DB becomes the bottleneck
- However, we'll likely exceed the memory of a single host
- Therefore, we spread our data across multiple hosts
- The aggregated memories of all hosts serves as our (large virtual) disk
  - Given enough data redundancy, we can temporarily afford to lose hosts due to crashes, updates etc

What is memcached ?

- Server (daemon) process written in C, providing an in-memory hashmap
- Clients can be written in any language
  - They talk to memcached servers via the memcached protocol
- Simple API: put(key,val), get(key), remove(key)
- A key always maps to the same memcached server (consistent hashing)
- Clients pick the right memcached server based on consistentHash(K)
- Given a good consistent hash function, keys are evenly distributed across memcached servers
- Typical use cases
  - PUT(K, V): write K,V to DB and put K,V into the cache
  - GET(K): get from cache, return V if found. Else fetch from DB, insert into cache and then return V

memcached


JBoss
Servlet
JBoss
Servlet
JBoss
Servlet
Apache
HTTP
mod_jk
memcached protocol
memcached A
memcached B
memcached C
DB

Issues with memcached

- Client have to use the memcached protocol to talk to a server
  - Protocol is ASCII over HTTP, inefficient, parsing of headers
- No L1 cache (some memcached clients cache locally though)
- memcached servers don't know about each other
- When a memcached server leaves, or a new server is started, some keys may now map to a different server ... which doesn't have them → DB lookup
- Keys are not redundantly stored on multiple servers
  - Server crash (or graceful shutdown) causes keys not to be found → DB lookup
  - On shutdown of a server, keys are not relocated to a different server

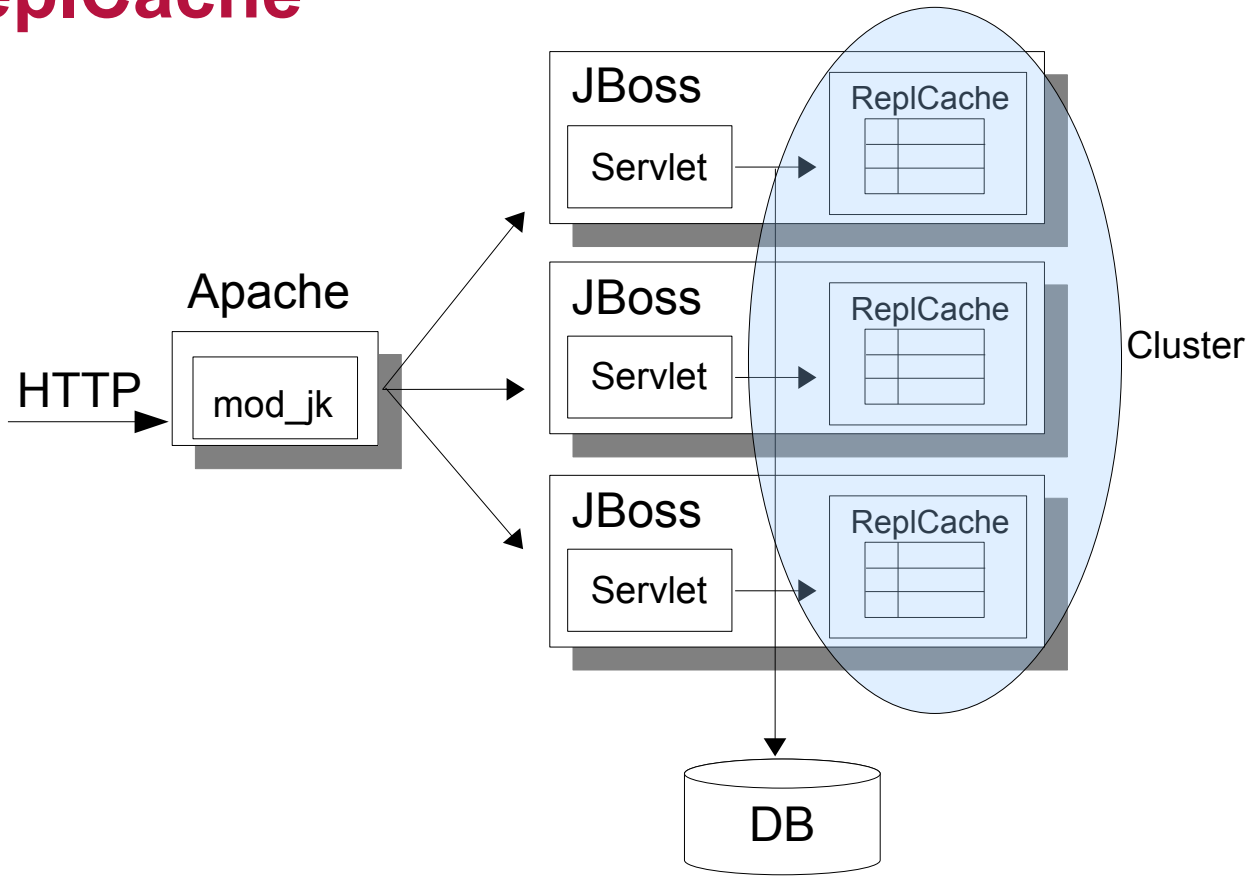
memcached in Java

- L1 caches
  - Clients run in the same address space as ReplCache
  - Access to data is fast when in L1 cache, no marshalling overhead
- All ReplCaches know about each other (in the same cluster)
  - Data can be migrated when shutting down
  - On rehashing, data is rebalanced automatically
    - ➔ Avoids a DB access
- Fast binary protocol (JGroups)
  - Can be adapted (compression, encryption, batching) via XML config
- Fewer network round trips

ReplCache: memcached improved

- memcached has no redundancy
  - When a server crashes, the keys on that server need to be re-read from the DB
- With ReplCache, every key has a *replication count*
  - Defines how many times the key should be stored in the cluster
  - -1: store key on every server
  - N: store key on N servers
    - ➔ Example: put(key,val, 3) stores {key,val} on 3 servers
    - ➔ When one of the 3 servers crashes, {key,val} will be copied to another server
- Advantage:
  - We can define redundancy *per data element* !
  - This allows us to use more of the available aggregated memory

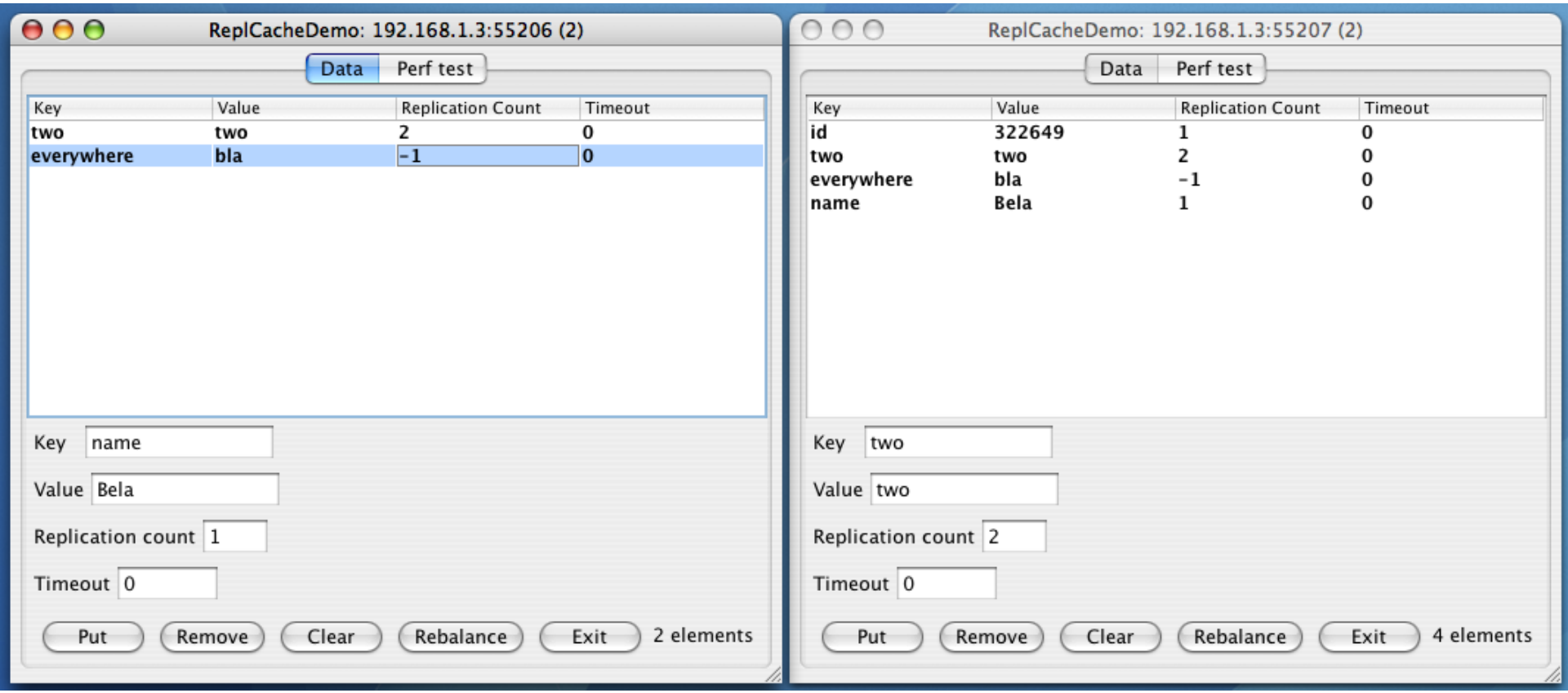
ReplCache


JBoss
Servlet
ReplCache
JBoss
Servlet
ReplCache
JBoss
Servlet
ReplCache
Cluster
Apache
HTTP
mod_jk
DB

ReplCache: memcached improved

- Data that can be easily re-read from the DB might use repl-count=1
- Data that isn't available in the DB needs to have a repl-count of -1
- Data where we can tolerate 3 concurrent crashes has repl-count=3
- Think of ReplCache as *Dynamic RAID*

Demo


ReplCacheDemo: 192.168.1.3:55206 (2)
Data
Perf test
Key
Value
Replication Count
Timeout
two
two
2
0
everywhere
bla
-1
0
Key
name
Value
Bela
Replication count
1
Timeout
0
Put
Remove
Clear
Rebalance
Exit
2 elements
ReplCacheDemo: 192.168.1.3:55207 (2)
Data
Perf test
Key
Value
Replication Count
Timeout
id
322649
1
0
two
two
2
0
everywhere
bla
-1
0
name
Bela
1
0
Key
two
Value
two
Replication count
2
Timeout
0
Put
Remove
Clear
Rebalance
Exit
4 elements

Infinispan

- Open Source Data Grid Platform
- Simple JSR-107 interface
  - Cache extends Map
- Content distribution based on Consistent Hashing
  - Ability to address massive heap
  - 100 nodes with 2GB each, DIST configured with 1 copy => 100GB addressable space from anywhere!
- JTA compliant, support for persistence to cache stores, eviction to prevent OOMs
- JMX reporting, (upcoming) mgmt console, migration tools
- Extremely high performance core container to support high concurrency
  - State of the art algorithms with minimal use of mutual exclusion

Infinispan

- memcached server module
  - Coming soon in Infinispan
- Speaks both memcached text protocol + custom binary protocol
- Able to use any memcached client
  - Not just Java! C#, C++, Python, PHP
- Custom Java client with ability to load-balance/fail-over
  - Makes use of binary protocol
  - We expect more clients like this for other systems (C++, etc) to be contributed
- Drop-in replacement for memcached farms
  - With all the added benefits of Infinispan

Infinispan

- Cool new AsyncAPI
  - Future<V> putAsync(K key, V value)
  - Allows you to fire off a number of such operations which operate in parallel
    - ➔ Operations can be synchronous
  - And wait for responses using Future.get()
- Other upcoming features include
  - Querying API
  - JPA-style interface for “POJO Caching”
  - Distributed executors
    - ➔ Runnables moved to the data, not the other way around
    - ➔ Map/reduce on your data!

Outlook

- Implementation of file system (java.io) interfaces to use the grid as a large in-memory file system
- JDBC: use the grid as data store rather than the disk ?
- Hibernate on a grid

Links

- memcached: http://www.danga.com/memcached
- ReplCache: http://www.jgroups.org/memcached.html and http://www.jgroups.org/replcache.html
- ReplCache WebStart demo: http://www.jgroups.org/jnlp/replcache.jnlp
- ReplCache Flash demo: http://www.jgroups.org/movies/ReplCache.swf
- Infinispan: http://www.infinispan.org

Bela Ban
JBoss / Red Hat

http://www.jboss.com
bela@jboss.com