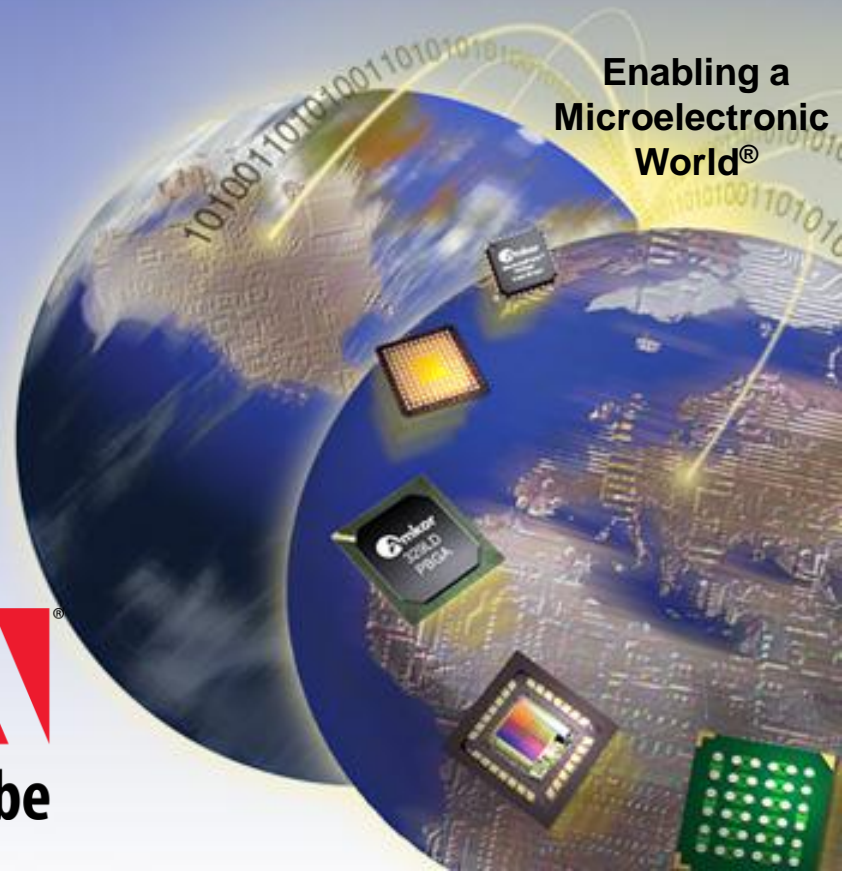


Greater Memory Management with ColdFusion 9 & Ehcache 2.4

Rob Brooks-Bilson

June, 2011

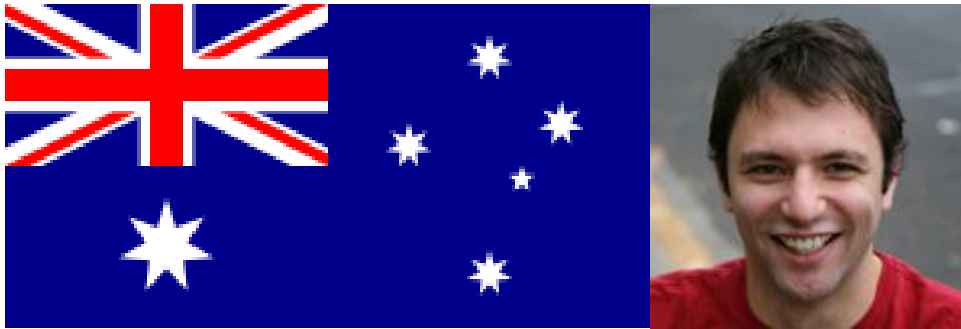
Enabling a
Microelectronic
World®



What we'll Cover

- Ehcache Overview
- What's new in ColdFusion 9.0.1 and Ehcache 2.0
- Ehcache Enhancements Post 2.0
- Upgrading Ehcache
- Cache Search
- Cache Replication
- Distributed Caching
- Big Memory
- Cache Monitoring
- Q&A

Ehcache Overview



“EE H Cayche”



“EE H Cash”

Written as Ehcache or ehcache, not ehCache or EhCache

Ehcache Overview

- De facto cache for Enterprise Java and ColdFusion
- Over 500,000 implementations
- Can be configured to run:
 - Local: In-process
 - Replicated: In-process
 - Distributed: In-process and Out-of-process
- Cached items are stored in memory as key/value pairs
- Supports disk stores for cache overflow
- Implemented as ColdFusion 9's caching provider for fragment, object and 2nd level Hibernate caching
- Also available in Railo as of 3.1.2

- Ehcache is fast
 - In-process caches are generally faster than out-of-process caches
 - No serialization required for objects written to memory
 - Order of magnitude faster than caching with Memcached or a NoSQL database
- Supports LRU, LFU, FIFO and Expiration Timeouts
- Cache-level operations are thread safe
- Ehcache can scale both vertically and horizontally
 - By default Ehcache runs within CF's JVM
 - Cache size is limited by the amount of memory available to CF's single jvm
 - You can cluster ehcache servers, but data is replicated among them
 - Ehcache 2.0 adds distributed caching via Terracotta
 - Open Source version allows for one active and one backup node
- Ehcache Enterprise
 - Built-in distributed caching – Terabyte Scale
 - Commercial support
 - Added enterprise features for production operation

Ehcache vs. Persistent Variable Scopes

- Why use Ehcache when ColdFusion already lets you store data in persistent scopes and the query cache?
- Both session/application variables, the ColdFusion query cache and Ehcache implement a HashMap
- In simple cases, performance across all three caching methods may be relatively equal
- Ehcache advantages:
 - Performs well regardless of load
 - Easily replicated/distributed
 - Flexible eviction policies
 - Self-managing
 - Comprehensive monitoring and statistics
 - Search
 - Big Memory

Ehcache vs. NoSQL

- Use cases are different
 - Distributed cache vs. Big Data persistence
 - Low latency/speed vs. Durability
- Ehcache is always faster than NoSQL
 - Hot data is always kept in-process in Ehcache
 - All NoSQL solutions as well as Memcached are out-of-process
 - In-process is always faster than out-of-process
 - Ehcache in-process access is $< 1 \mu\text{s}$
 - Ehcache out-of-process (Terracotta) access is $< 2\text{ms}$

Ehcache Editions



| | Open Source | DX | EX | FX |
|---|-------------|-------------|-------------|-------------|
| Ehcache Open Source Features and Modules | X | X | X | X |
| Query, search and analysis for distributed cache | Single Node | Single Node | Distributed | Distributed |
| Server array with striping for linear scale | | | | X |
| High availability | | | X | X |
| High performance data persistence | | | X | X |
| In-memory performance as you scale | | | X | X |
| Wide-area network (WAN) replication (add-on module) | | | X | X |
| Comprehensive cache management | | | X | X |
| Visual cache tuning | | X | X | X |
| Operations console monitor 3 rd party monitoring integration | X | X | X | X |
| Enterprise-class 24x7 production and developer support | | X | X | X |
| Certified software, updates, and patches | | X | X | X |
| Commercial license and legal indemnification | | X | X | X |

ColdFusion Caching – What's New

CF 9.0

- Page fragment caching
- Caching objects and data
- ORM caching

Ehcache 1.6

Core cache

NEW

NEW

CF 9.0.1

- New and enhanced functions: cacheGetSession, cacheGetMetadata
- Support for new configuration properties
- User-defined caches

Ehcache 2.0

- Improved Hibernate plug-in, management, XA transactions, write-behind, bulk load, ...
- Snap-in support for distributed caching, via easy upgrade to Enterprise Ehcache



Ehcache Enhancements Post 2.0



- Dozens of bug fixes
- Faster performance
- New features:
 - Cache Search
 - copyOnRead and copyOnWrite
 - Explicit locking API
 - Nonstop Cache
 - New consistency modes: Strong and Eventual
 - Local and XA transactions
 - XA Transactions for non clustered caches

Upgrading Ehcache

Add 5 grilled prawns to any steak

extra £1.49

Add pepper sauce on the side

extra 79p

**UPGRADE YOUR
CHIPS TO WEDGES
FOR AN EXTRA 49p**

Bal

Mix

bur

flat

Add

MC

Clas

Chi

Len



Caution!

Upgrade at Your Own Risk

Upgrading to Newer Versions of Ehcache

1. Download the latest version of Ehcache
2. Extract ehcache-core-2.4.2.jar into your ColdFusion /lib directory
3. Rename your existing ehcache.jar file to ehcache.original
4. Restart you ColdFusion server

Cache Search



- Available starting in Ehcache 2.4.0
- Single node for open source Ehcache
 - Multi-node requires Enterprise Ehcache and Enterprise Terracotta
- Support for simple and compound expressions:
 - Additional clauses automatically *and*

Search Operators

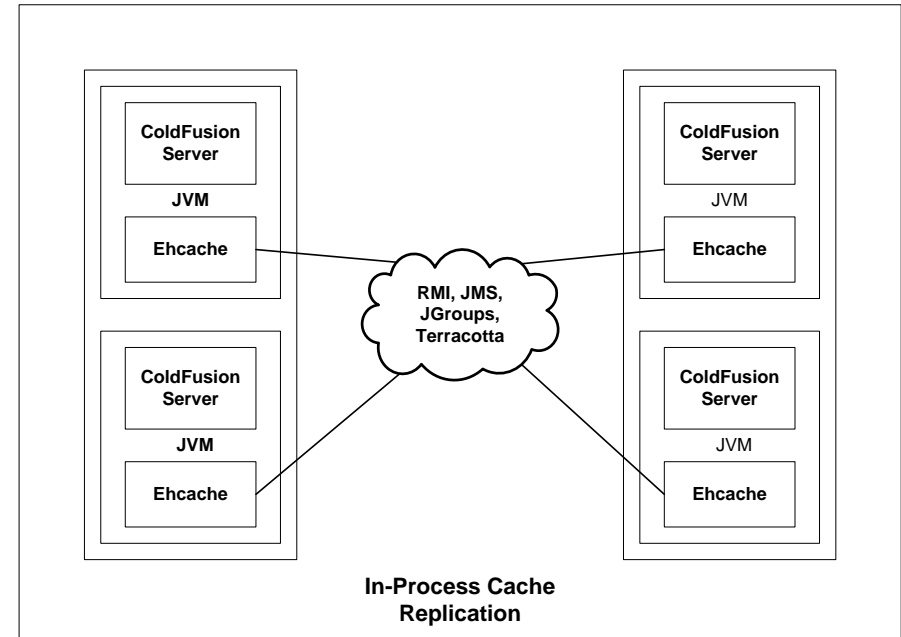
| Shorthand | Criteria Class | Description |
|-----------|--------------------|---|
| and | And | The Boolean AND logical operator |
| between | Between | A comparison operator meaning between two values |
| eq | EqualTo | A comparison operator meaning Java "equals to" condition |
| gt | GreaterThan | A comparison operator meaning greater than. |
| ge | GreaterThanOrEqual | A comparison operator meaning greater than or equal to. |
| in | InCollection | A comparison operator meaning in the collection given as an argument |
| lt | LessThan | A comparison operator meaning less than. |
| le | LessThanOrEqual | A comparison operator meaning less than or equal to |
| ilike | ILike | A regular expression matcher. '?' and "*" may be used. Note that placing a wildcard in front of the expression will cause a table scan. ILike is always case insensitive. |
| not | Not | The Boolean NOT logical operator |
| ne | NotEqualTo | A comparison operator meaning not the Java "equals to" condition |
| or | Or | The Boolean OR logical operator |

Cache Replication



Cache Replication in ColdFusion 9

- Object, Template and Hibernate caches can be replicated
- Simple configuration via ehcache.xml file
- Cache replication via RMI, JMS, JGroups or Terracotta
- Replication can be synchronous or asynchronous



Potential Replication Gotchas

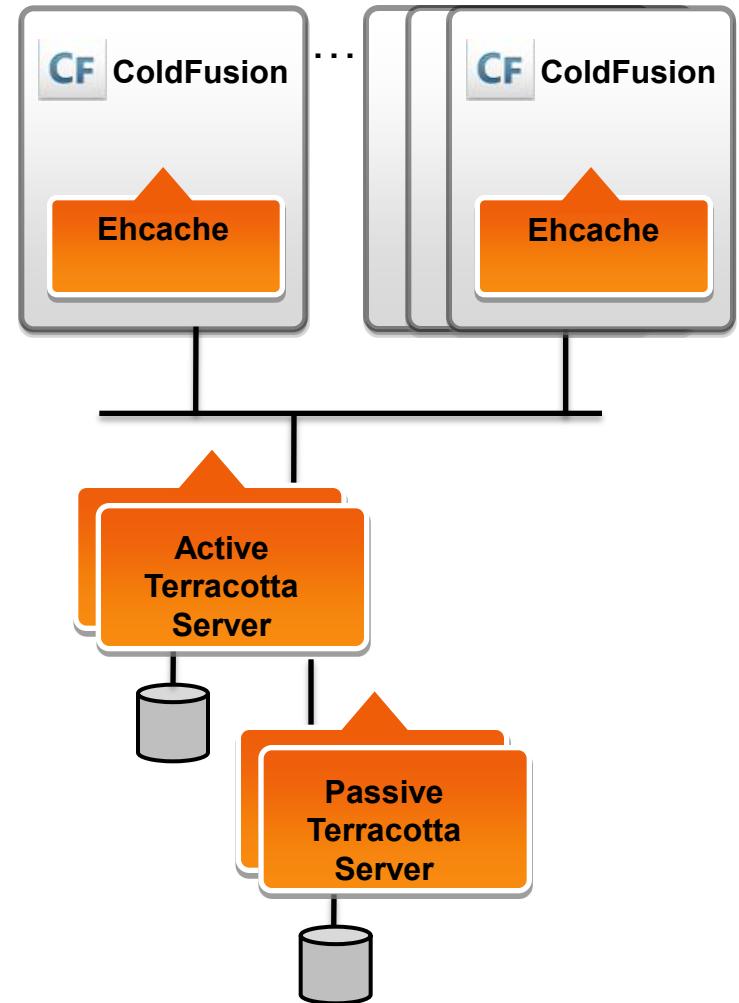
- Synchronous vs. Asynchronous delivery
 - Asynchronous replication is the fastest method
 - Because it's asynchronous the caller returns immediately
 - Messages are placed in a queue and batched via RMI as they are processed
 - Potential for data inconsistency exists
 - Synchronous
 - Removes potential for data inconsistency
 - Slower operation as caller waits for replication to complete before returning
- Time To Idle
 - Inconsistent with replicated caching
 - Data on some nodes will live longer than on others due to cache usage patterns
 - Do not use unless you don't care about inconsistent data across cache nodes

Distributed Caching

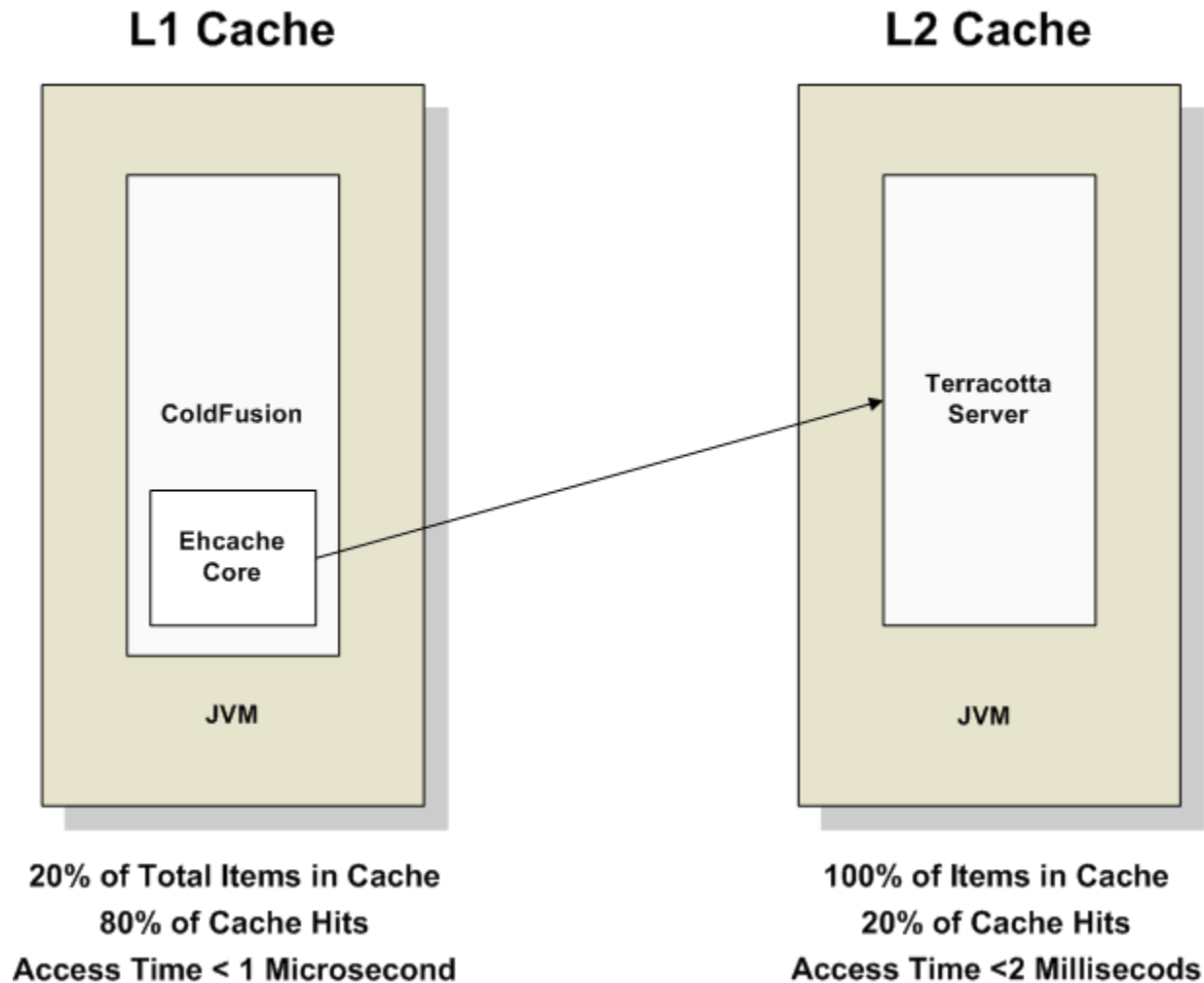


Distributed Caching: Ehcache Core with Terracotta Server

- Open source license
- Terracotta runs out-of-process
- Single active node + passive backup
- Works with single node ehcache or replicated ehcache
- Simple config via ehcache.xml
- Supports template, object & hibernate caches

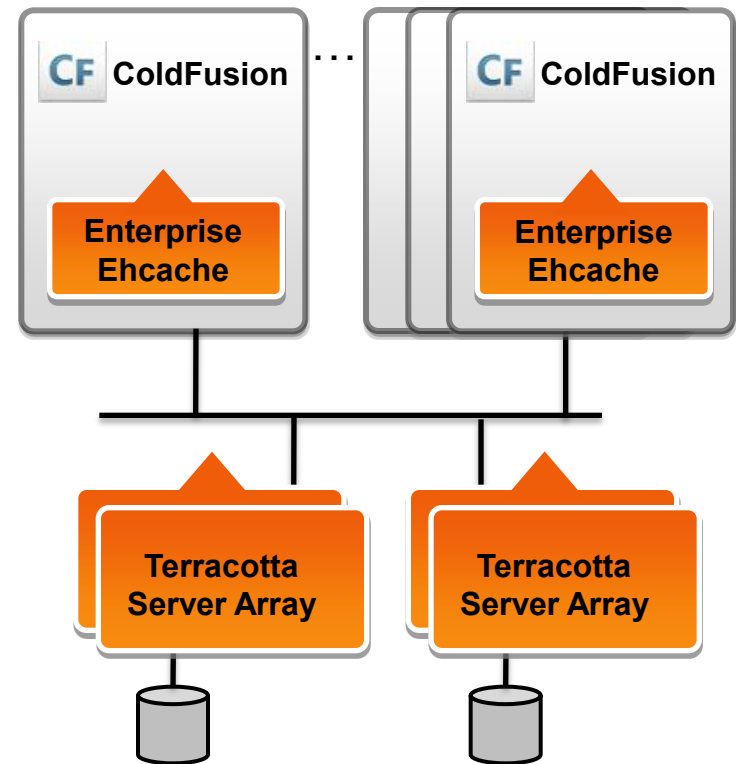


Ehcache/Terracotta Tiered Caching Architecture



Distributed Caching: Enterprise Ehcache with Terracotta Server Array

- Commercial license
- “Snap-in scale”
- Same API
- High data capacity: 1TB+
- Highly available
- Coherent—i.e., drift-free data consistency between machines
- Flexible—you set the cache semantics based on business requirements



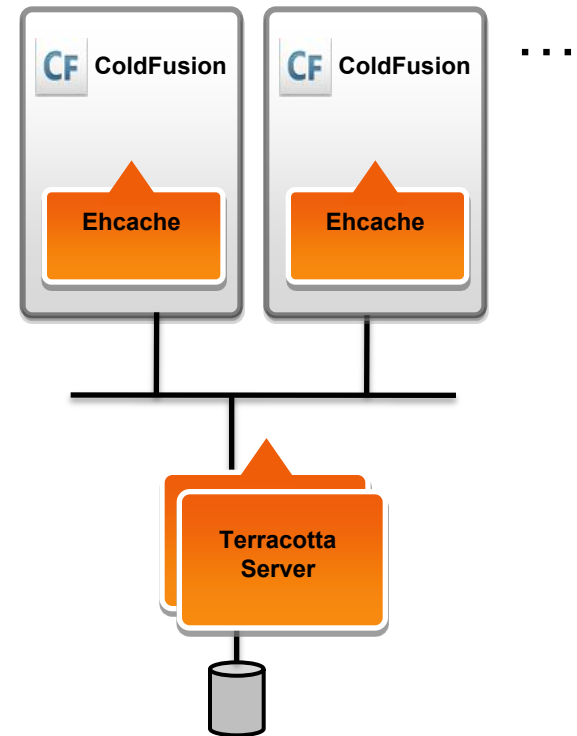
Distributed Caching in ColdFusion

3 Easy Steps, 2 Lines of Config

1. Add the following jar files for Terracotta 3.5.1 / Ehcache 2.4.2:

ehcache-terracotta-2.4.2.jar
terracotta-toolkit-1.2-runtime-3.1.0.jar
slf4j-api-1.6.1.jar
slf4j-log4j12-1.6.1.jar
slf4j-jdk14-1.6.1.jar

*You'll be updating some of ColdFusion's
existing slf4j files



Distributed Caching in ColdFusion

3 Easy Steps, 2 Lines of Config

2. Edit your ehcache.xml

```
<ehcache>
```

```
  <terracottaConfig url="someserver:9510"/>
```

```
  <defaultCache
```

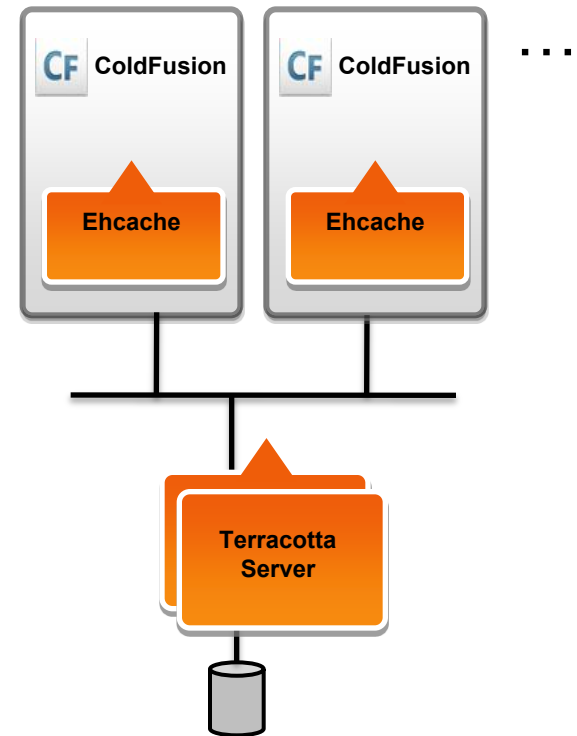
```
    maxElementsInMemory="10000"
```

```
    timeToLiveSeconds="120">
```

```
    <terracotta clustered="true">
```

```
  </defaultCache>
```

```
</ehcache>
```



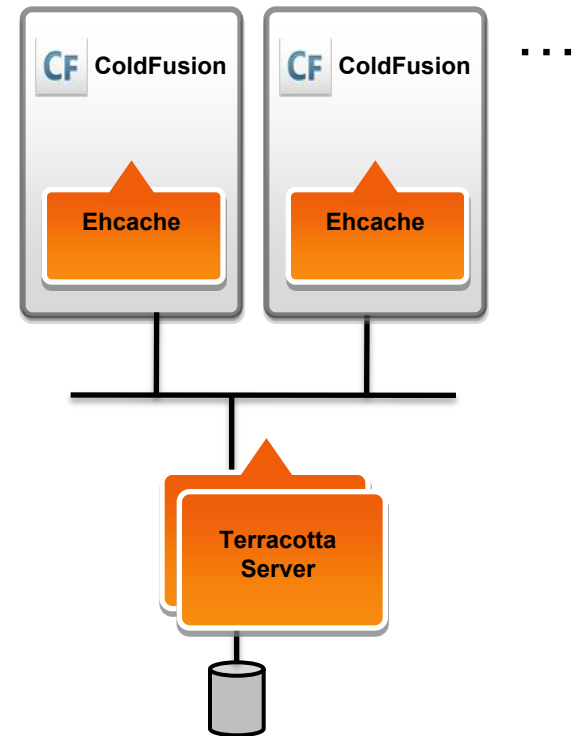
Distributed Caching in ColdFusion

3 Easy Steps, 2 Lines of Config

3. Start the Terracotta server, then restart ColdFusion

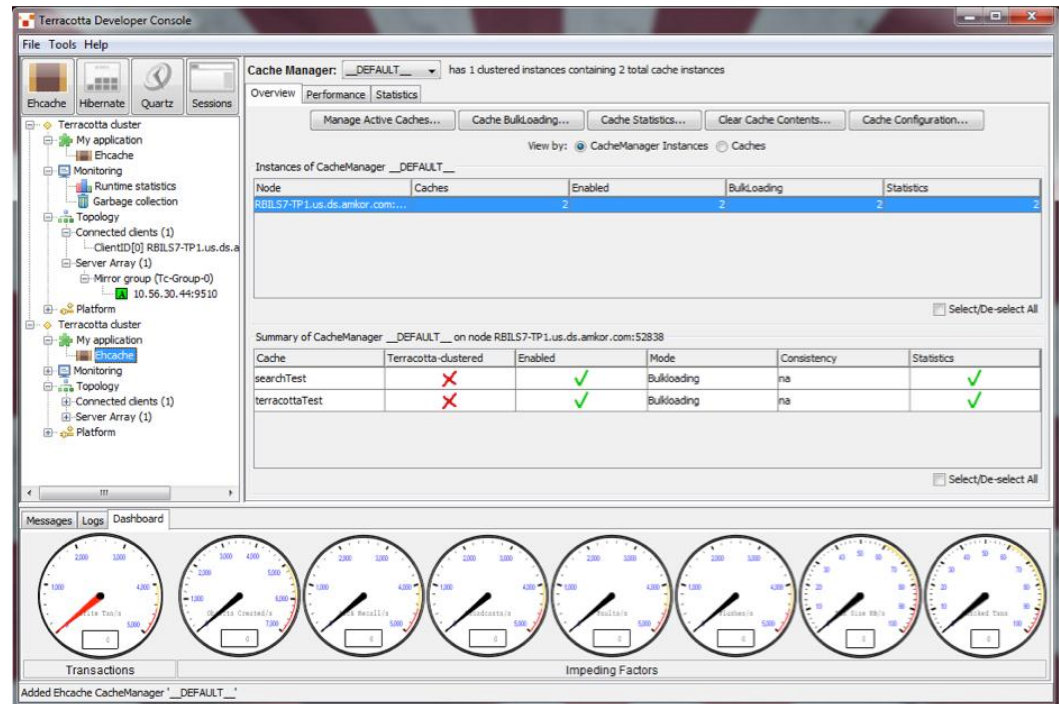
`bin/start-tc-server.sh`

*Note that the Terracotta server must be running before you bring up your ColdFusion server otherwise ColdFusion will hang when Ehcache can't get a connection to the Terracotta server.



Developer Console

- Client app to monitor Ehcache, Hibernate Cache, Web Sessions and Quartz Scheduler
- For Ehcache:
 - Caches
 - Statistics
 - Config
- For Hibernate:
 - Hibernate cache view
 - Hibernate cache stats



Terracotta Editions



| Terracotta Software | Open Source | Commercial |
|---|-------------|------------|
| Linear unlimited Terabyte scale and Terracotta Server Array data striping | | X |
| Enterprise-class operations and management capabilities | | X |
| Authentication, authorization and security features | | X |
| Search for distributed cache | | X |
| Quartz Where cluster locality API | | X |
| Enterprise-class 24x7 support | | X |
| Certified software, updates, patches, & legal indemnification | | X |
| Industry-standard Java cache (Ehcache) | X | X |
| Industry-standard Hibernate cache (Ehcache for Hibernate) | X | X |
| Industry-standard Java scheduler (Quartz) | X | X |
| High-performance web cache and coherent, distributed sessions cache | X | X |
| Performance, reliability, and scalability for Spring applications | X | X |
| Terracotta server to provide coherent scale and HA for all technologies | X | X |

BigMemory

Mercury Delay Line Memory Tank

United States

Mercury delay lines served as the main memory units for many early computers. Sound waves were sent through a tube of mercury, detected, and sent back through the tube. A tube one meter long could contain about 1,000 characters, and each one millionth of a second to re-circulate these signals.

The tank for the UNIVAC-1 had 70 tubes, each holding ten 12 character words. The tank could be accessed in approximately 222 microseconds. With ten tanks, the total memory in readers units was about 20,000 bytes.

Gift of William Agnes, 2002.00



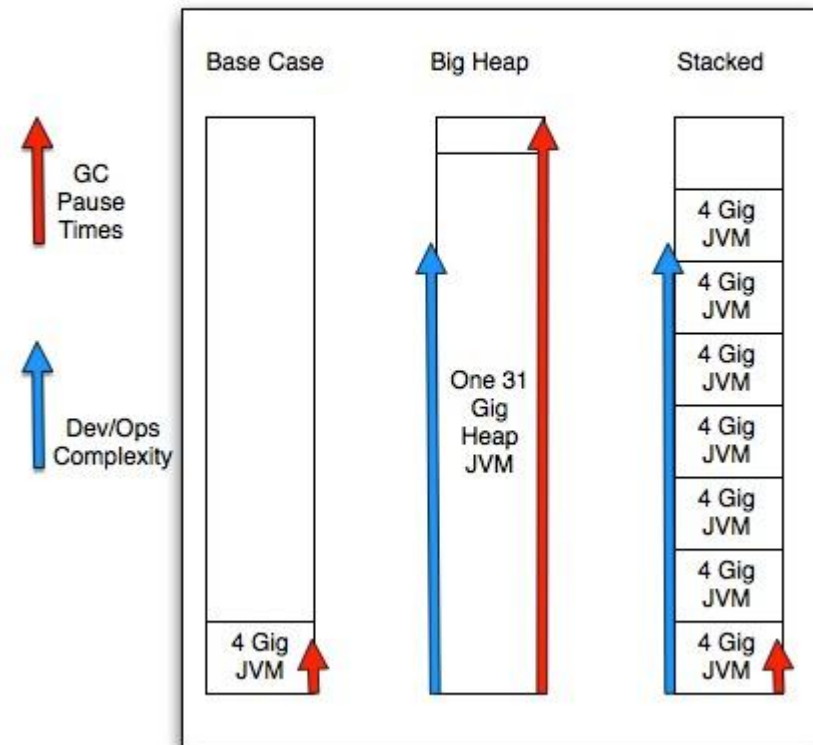
"640K ought to be enough for anybody."

Attributed to Bill Gates, although he denies it

Problems with Application Scalability on the JVM

- Slow Applications lead to caching
- Large caches lead to latency from Garbage Collection pauses
- GC pauses necessitate JVM tuning
- Application usage and data grows
- More caching is needed
- More JVM tuning is needed
- As cache size increases, more heap memory is used
- More heap memory usage results in unpredictable and longer GC pauses
- Eventually you hit a wall
- Add more JVMs
- Increased deployment and management complexity
- Tune! Tune! Tune!

Today's 32 Gig 16 Core Servers Using Java



The Problem with Memory and the JVM

- RAM is outpacing the JVM
 - 32GB is now fairly standard on most servers
 - Amazon EC2 allows for up to 68.4GB
 - Most of that memory is used inefficiently if it's used at all
- Cached data ages differently than standard business objects which can confuse the Java garbage collector
 - The cache expiration determines when the data becomes garbage
- And then there's tuning...

GC Tuning – It's a Black Art



GC Tuning –Ninja Skills?



GC Tuning – It Can Be Painful



GC Tuning – Frustrating!



GC Tuning – Why Torture Yourself?

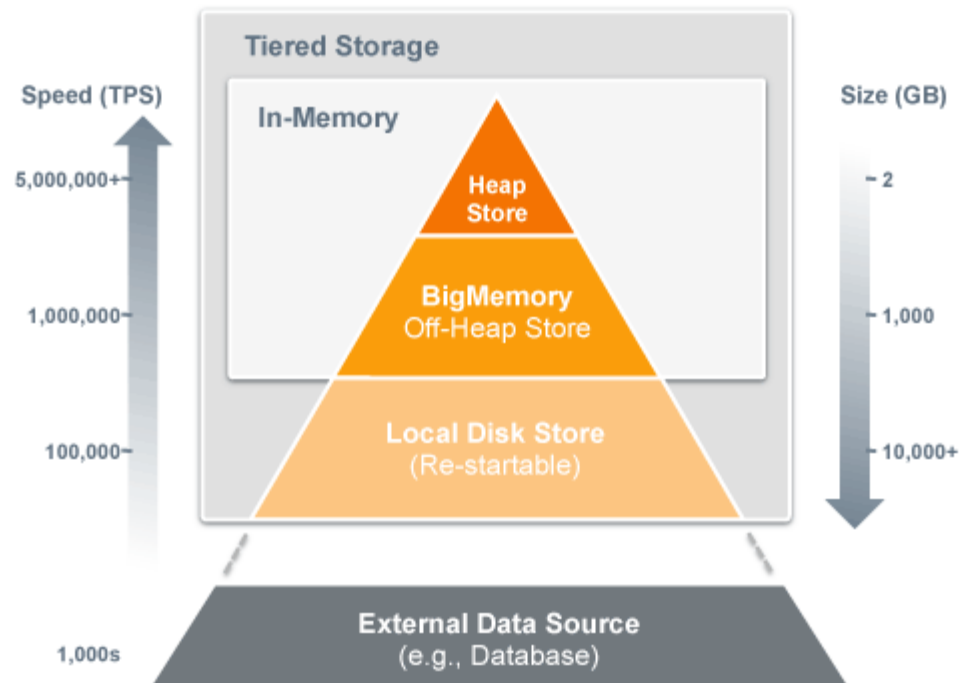


BigMemory to the Rescue



- Cache huge amounts of data with no GC pauses
- Easy to implement
 - Pure Java implementation
 - Works with all JVMs
 - Works with both standalone and distributed caches
 - No application code changes necessary to implement
- Avoid GC because of pauses
 - Off heap store on direct memory buffers
 - 1 million puts per second
 - Tested up to 350GB
 - 1 line of config to use it
 - JVM doesn't have to search for garbage because we already know when a cache item needs to be thrown out (cache expiry)
- Commercial product

Tiered Storage



BigMemory Basic Configuration

ehcache.xml:

```
<cache name="sample-offheap-cache"  
  maxElementsInMemory="10000"  
  eternal="true"  
  memoryStoreEvictionPolicy="LRU"  
  overflowToOffHeap="true"  
  maxMemoryOffHeap="1G"/>
```

JVM Config:

```
-XX:MaxDirectMemorySize=2G
```

Ehcache Monitor



- Free for development, commercial license for production
- Monitor multiple cache servers from a single web console
- Two components:
 - Probe
 - ehcache-probe-1.0.2.jar
 - Install in same directory as your ehcache.jar file
 - Server
 - Standalone server
 - May be installed local or remote
- Simple config
 - Add a cacheManagerPeerListenerFactory to ehcache.xml
 - Uncomment the server and port
- Stats are transmitted via XML over HTTP

- Ehcache: <http://ehcache.org/>
- ColdFusion 9 Documentation
- Using Ehcache with ColdFusion:
<http://ehcache.org/documentation/coldfusion.html>
- My Blog Series on Caching in ColdFusion 9:
<http://www.brooks-bilson.com/blogs/rob/index.cfm/Caching>
- High Scalability: <http://highscalability.com/>
- Building High Performance Applications with ColdFusion 9 and Ehcache 2.4: <http://java.dzone.com/articles/building-high-performance>

Rob Brooks-Bilson

*Director of Architecture
Amkor Technology*

rbils@amkor.com

Twitter: @styggiti

Questions about Terracotta

sales@terracottatech.com

Web: www.terracotta.org

Questions about Adobe

**Web:
www.adobe.com/products/coldfusion**

Visit www.terracotta.org/webcasts to register for our upcoming webcast on June 29th.
Learn more about implementing ColdFusion and Ehcache from Full Sail University.

SNAP IN

SPEED UP

SCALE OUT