Clustering a Grails Application for Scalability and Availability

Groovy & Grails eXchange

9th December 2009

Burt Beckwith

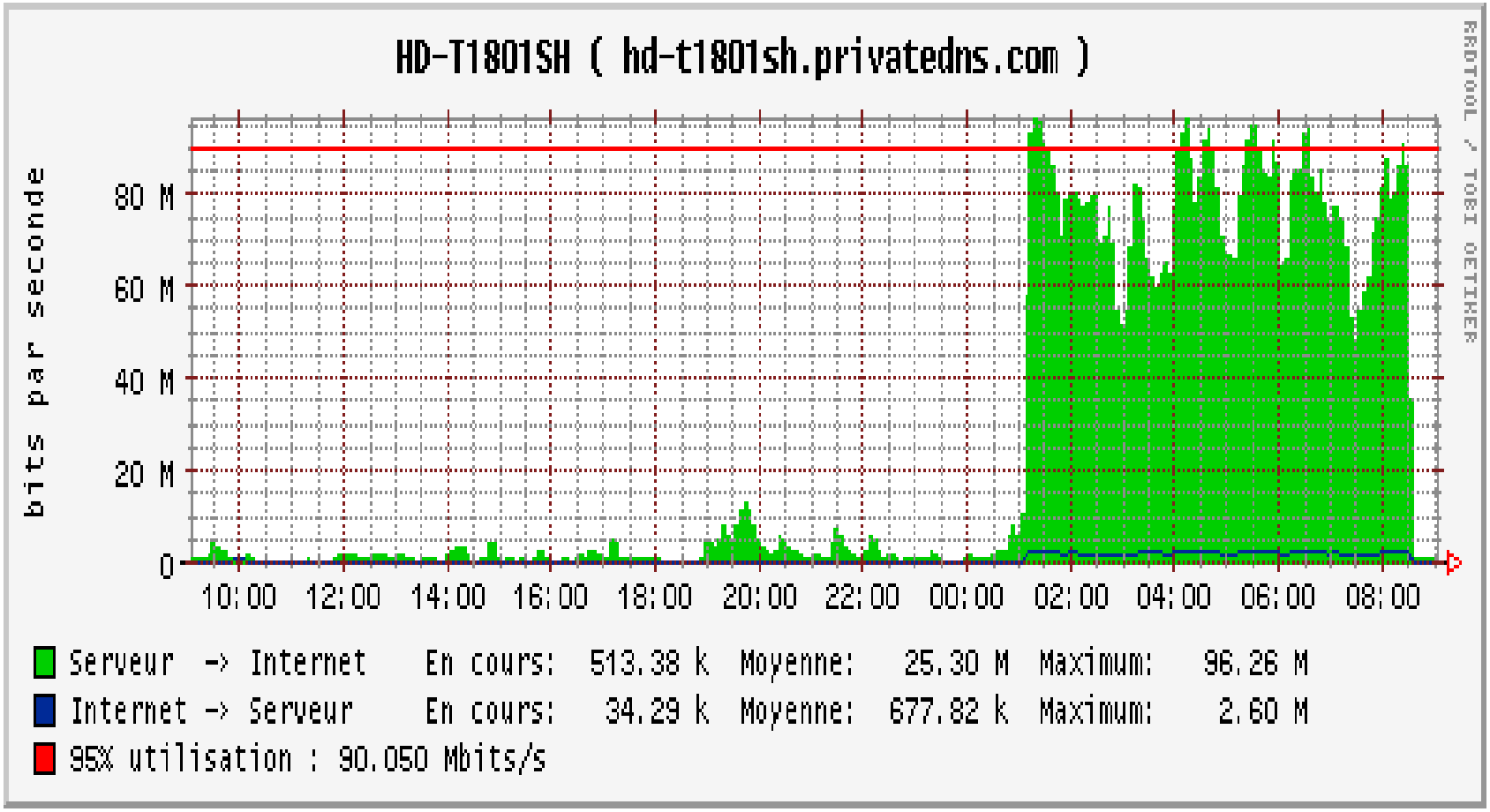
My Background

- Java Developer for over 10 years
- Background in Spring, Hibernate, Spring Security
- Full-time Grails developer since February 2008, now an independent consultant
- Regular contributor on the Grails User mailing list
- Primary developer of Spring Security (Acegi) Grails plugin
- Created UI Performance, Datasources, Twitter, Spring MVC, and CodeNarc Grails plugins
- Technical Editor of Grails in Action
- http://burtbeckwith.com/blog/
- http://twitter.com/burtbeckwith

Overview

- General strategies for scaling and availability
- Biased towards Tomcat/MySQL/Ehcache
- These are not the only options for scaling, more of an introduction and overview
- Not necessarily better performance, aim is to handle more load with the same response time per action (linear scale)

Goal: Handle being Slashdotted



What to Cluster

- Web sessions
  - Replicated HTTP sessions in at least one other server instance allows failover after crash
- Hibernate 2rd-level cache
  - If you use a read-write cache you must update other servers' caches to avoid stale data
- Quartz
  - Use the database to load-balance, as a mutex to avoid duplicate job execution, for failover
    - *Note*: Terracotta is working on non-database clustered Quartz
- Database?

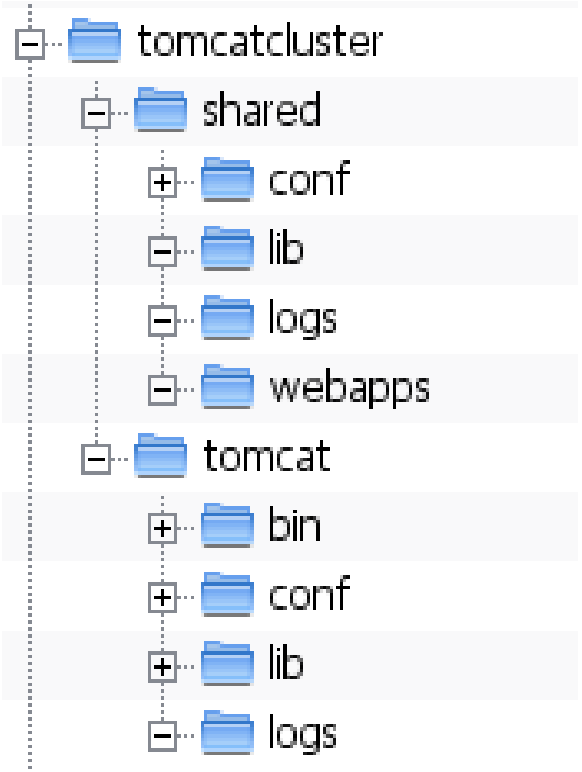
Tomcat Clustering

- Enable and configure clustering in server.xml
- For standard usage it's sufficient to uncomment the existing cluster configuration
- Change multicast port and/or address to avoid conflicts:

```
<Membership className='org.apache.catalina.tribes.membership.McastService'
        address='228.0.0.4' port='45564' frequency='500' dropTime='3000'/>
```

Tomcat Cluster Directory Layout

- `tomcat` directory with `bin` and `lib` directories – separate from shared to make upgrade easier
- `shared` directory with deployed war(s), shared logs, shared `server.xml`
- multiple `instance_N` directories with one cluster node containing node-specific `bin/setenv.sh` and `conf/catalina.properties`, and `logs`, `temp`, and `work` directories

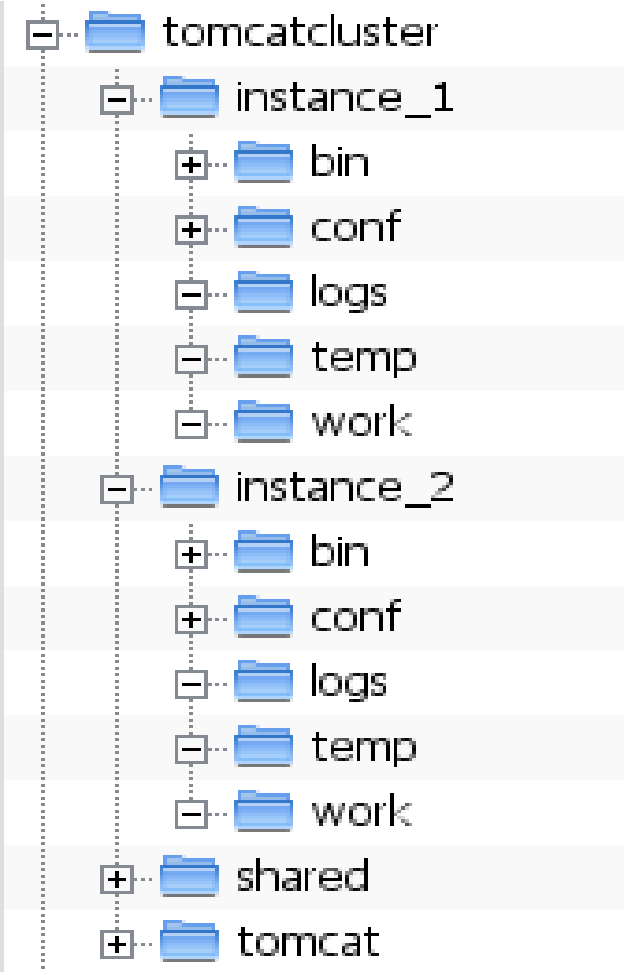


Create the Cluster

- All the scripts need the cluster root so it's best to set it as an environment variable:
  - `export CR=/usr/local/tomcatcluster`
- On server 1
  - `createCluster.sh $CR`

Create the Cluster

- Then create nodes:
  - `createInstance.sh 1 1 $CR`
  - `createInstance.sh 1 2 $CR`



Create the Cluster

Each node has

- `bin/setenv.sh` with memory, JMX params
- `conf/catalina.properties`
  - ports, server and instance number
- `logs`, `temp`, and `work` directory

Create the Cluster

Repeat for server 2, 3, etc.

- `createCluster.sh $CR`
- `createInstance.sh 2 1 $CR`
- `createInstance.sh 2 2 $CR`

- `createCluster.sh $CR`
- `createInstance.sh 3 1 $CR`
- `createInstance.sh 3 2 $CR`

Deployment

- Build as usual:
  - `grails clean && grails war`
- Use the deploy script:
  - `deploy.sh ../clustered-0.1.war $CR`
  - Deletes old war, copies and expands to `$CR/shared/webapps/ROOT/`
- All nodes on a server share one war

Running

- On each server, for each node:
  - `run.sh start 1 $CR`
  - `run.sh start 2 $CR`
- Shutdown using
  - `run.sh stop 1 $CR`
  - `run.sh stop 2 $CR`
- Between runs, cleanup using
  - `cleanup.sh $CR`
  - for each node, deletes logs, temp files, work dir contents
  - deletes shared logs

Web Cluster Load Balancing

- Ideally, use hardware load balancers
  - Coyote Point
  - F5
  - Cisco
- Can also use Apache HTTPD (JK/mod_proxy)
- Can also use a dedicated Tomcat instance using Balancer web app (not in 6.0?)

Quartz

- Running embedded in-memory instance per web server will run duplicate jobs
- A single instance is a point of failure
- Use JDBC support to store jobs in database, use database locking to prevent duplicates, load balance, support failover
- Need to create tables and do minor configuration

Quartz Configuration

- Fix quartzScheduler in resources.groovy, see GRAILSPLUGINS-1207

```
beans = {
   if (Environment.PRODUCTION == Environment.current) {
      quartzScheduler(SchedulerFactoryBean) {
         autoStartup = false
         dataSource = ref('dataSource')
         transactionManager = ref('transactionManager')
         configLocation = 'classpath:quartz.properties'
         jobFactory = ref('quartzJobFactory')
         jobListeners = [ref("${SessionBinderJobListener.NAME}")]
         globalJobListeners = [ref("${ExceptionPrinterJobListener.NAME}")]
      }
   }
}
```

Quartz Configuration

- Also patch QuartzGrailsPlugin.groovy, see GRAILSPLUGINS-1420:

```
if (scheduler.getTrigger(trigger.triggerAttributes.name,
                         trigger.triggerAttributes.group)) {
  scheduler.rescheduleJob(trigger.triggerAttributes.name,
                          trigger.triggerAttributes.group,
                          ctx.getBean("${key}Trigger"))
}
else {
  scheduler.scheduleJob(ctx.getBean("${key}Trigger"))
}
```

Quartz Configuration

- conf/QuartzConfig.groovy:

```
quartz {
   jdbcStore = true
   autoStartup = true
}
environments {
   test {
      quartz {
         jdbcStore = false
         autoStartup = false
      }
   }
}
```

Quartz Configuration

quartz.properties:

```
org.quartz.scheduler.instanceName clusterdemo_quartz
org.quartz.scheduler.instanceId AUTO

org.quartz.threadPool.class org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount 5
org.quartz.threadPool.threadPriority 5

org.quartz.jobStore.misfireThreshold 60000

org.quartz.jobStore.class org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass org.quartz.impl.jdbcjobstore.StdJDBCDelegate

org.quartz.jobStore.useProperties false
org.quartz.jobStore.tablePrefix QRTZ_
org.quartz.jobStore.isClustered true
org.quartz.jobStore.clusterCheckinInterval 5000

org.quartz.plugin.shutdownhook.class org.quartz.plugins.management.ShutdownHookPlugin
org.quartz.plugin.shutdownhook.cleanShutdown true
```

Quartz Configuration

grails-app/conf/hibernate/hibernate.cfg.xml

```
<hibernate-configuration>
   <session-factory>
      <mapping resource='Quartz.mysql.innodb.hbm.xml'/>
   </session-factory>
</hibernate-configuration>
```

Quartz Configuration

Quartz.mysql.innodb.hbm.xml (see plugins/quartz-0.4.1/src/templates/sql)

```
<hibernate-mapping>
  <database-object>
    <create>
CREATE TABLE QRTZ_PAUSED_TRIGGER_GRPS (
      TRIGGER_GROUP VARCHAR(80) NOT NULL,
      PRIMARY KEY (TRIGGER_GROUP)
) ENGINE=InnoDB
    </create>
    <drop>DROP TABLE IF EXISTS QRTZ_PAUSED_TRIGGER_GRPS</drop>
    <dialect-scope name='org.hibernate.dialect.MySQL5InnoDBDialect' />
  </database-object>
...
```

Hibernate 2rd-Level Cache

DataSource.groovy:

```
dataSource {
   pooled = true
   driverClassName = 'com.mysql.jdbc.Driver'
   url = ...
   username = ...
   password = ...
   dialect = org.hibernate.dialect.MySQL5InnoDBDialect
}

hibernate {
   cache.use_second_level_cache = true
   cache.use_query_cache = true
   cache.provider_class = 'org.hibernate.cache.EhCacheProvider'
}
```

Mapping in Domain Classes

```
class Book {
   …
   static mapping = { cache true }
}
```

```
class Country {
   …
   static mapping = { cache usage: 'read-only' }
}
```

```
class Author {
   …
   static hasMany = [books:Book]
   static mapping = { books cache: true }
}
```

Query Cache

- Criteria queries:

```
def criteria = DomainClass.createCriteria()
def results = criteria.list {
   cacheable(true)
}
```

- HQL queries:

```
DomainClass.withSession { session ->
   return session.createQuery(
      "select ... from … where ...")
      .setCacheable(true).list()
   }
}
```

- In dynamic finders (new in 1.1)

```
def person = Person.findByFirstName("Fred", [cache:true])
```

Hibernate query cache considered harmful?

- Most queries are not good candidates for caching; must be same query and same parameters
- Updates to domain classes will pessimistically flush all potentially affected cached results
- DomainClass.list() is a decent candidate if there aren't any (or many) updates and the total number isn't huge
- Great blog post by Alex Miller (of Terracotta) http://tech.puredanger.com/2009/07/10/hibernate-query-cache/

Usage Notes

- The 1st-level cache is the Hibernate Session
- Can significantly reduce database load by keeping instances and query results in memory
- "cache true" creates a read-write cache, best for read-mostly objects since frequently-updated objects will result in excessive cache invalidation (and network traffic when distributed)
- "cache usage: 'read-only'" creates a read-only cache, best for lookup data (e.g. Countries, States, Zip Codes, Roles, etc.) that never change

Usage Notes

- DomainClass.get() always uses the 2nd-level cache
- By default nothing else always uses the cache but can be overridden
- Grails switched from Ehcache as default to OSCache but only because it handles dev mode restarts better – switch back!
  - *Note*: In Grails 1.2 Ehcache 1.7 is used which handles restarts well, so it's the default again
- Create ehcache.xml in root of classpath to customize; will use defaults from jar otherwise
- In dev mode you'll want to run standalone, but when deployed will be clustered, so there's a build step

ehcache.xml (Development)

```
<ehcache>
  <diskStore path='java.io.tmpdir' />

  <defaultCache maxElementsInMemory='10000' eternal='false' timeToIdleSeconds='120'
    timeToLiveSeconds='120' overflowToDisk='true' maxElementsOnDisk='10000000'
    diskPersistent='false' diskExpiryThreadIntervalSeconds='120'
    memoryStoreEvictionPolicy='LRU' />

  <cache name='com.burtbeckwith.clusterdemo.Role'
     maxElementsInMemory='1000' eternal='true' maxElementsOnDisk='0' />

  <cache name='com.burtbeckwith.clusterdemo.User.authorities'
     maxElementsInMemory='10000' eternal='false' />

  <cache name='org.hibernate.cache.StandardQueryCache'
     maxElementsInMemory='50' eternal='false' timeToLiveSeconds='120'
    maxElementsOnDisk='0' />

  <cache name='org.hibernate.cache.UpdateTimestampsCache'
     MaxElementsInMemory='5000' eternal='true' maxElementsOnDisk='0' />

</ehcache>
```

ehcache.xml (Deployed)

```
<ehcache xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xsi:noNamespaceSchemaLocation='ehcache.xsd'>
   <diskStore path='java.io.tmpdir' />

   <cacheManagerPeerProviderFactory
     class='net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory'
     properties='peerDiscovery=automatic,multicastGroupAddress=237.0.0.2,
                 multicastGroupPort=5557,timeToLive=1'
     propertySeparator=',' />

   <cacheManagerPeerListenerFactory
        class='net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory' />

   <defaultCache maxElementsInMemory='10000' eternal='false'
     timeToIdleSeconds='120' timeToLiveSeconds='120' overflowToDisk='true'
     MaxElementsOnDisk='10000000' diskPersistent='false'
     diskExpiryThreadIntervalSeconds='120' memoryStoreEvictionPolicy='LRU'/>
```

ehcache.xml (Deployed)

```
<cache name='com.burtbeckwith.clusterdemo.Role'
  maxElementsInMemory='1000' eternal='true' maxElementsOnDisk='0'>
  <cacheEventListenerFactory
    class='net.sf.ehcache.distribution.RMICacheReplicatorFactory'
    properties='replicateAsynchronously=false'
  />
</cache>
<cache name='com.burtbeckwith.clusterdemo.User.authorities'
  maxElementsInMemory='10000' eternal='false'>
  <cacheEventListenerFactory
    class='net.sf.ehcache.distribution.RMICacheReplicatorFactory'
    properties='replicateAsynchronously=false'
  />
</cache>
```

ehcache.xml (Deployed)

```
  <cache name='org.hibernate.cache.StandardQueryCache'
    maxElementsInMemory='50' eternal='false' timeToLiveSeconds='120'
    maxElementsOnDisk='0'>
    <cacheEventListenerFactory
      class='net.sf.ehcache.distribution.RMICacheReplicatorFactory'
      properties='replicateAsynchronously=false'
    />
  </cache>

  <cache name='org.hibernate.cache.UpdateTimestampsCache'
    maxElementsInMemory='5000' eternal='true' maxElementsOnDisk='0'>
    <cacheEventListenerFactory
      class='net.sf.ehcache.distribution.RMICacheReplicatorFactory'
      properties='replicateAsynchronously=false'
    />
  </cache>
</ehcache>
```

Sample App

- Uses Spring Security plugin (to demonstrate login failover and provide cacheable domain classes)
- Uses Quartz plugin
- MySQL database

Demo

General Configuration Steps

Tomcat

- Run `grails install-templates` and add `<distributable/>` to `src/templates/war/web.xml`
- Configure `server.xml`
- Configure cluster elements
- Access log valve

General Configuration Steps

Quartz

- `QuartzConfig.groovy`
- `hibernate.cfg.xml` and hbm file(s) for JDBC store
- `quartz.properties`
- Fix bugs

General Configuration Steps

Hibernate 2nd-level cache

- Change `cache.provider_class` in `DataSource.groovy` to `org.hibernate.cache.EhCacheProvider`
- Make cached domain classes Serializable - remember that closures aren't serializable
- Configure domain class cache in mapping closure
- Configure queries for query cache where appropriate
- Create `ehcache.xml` in `grails-app/conf` or `src/java`, and a distributed version to be used in production (`_Events.groovy`)

General Configuration Steps

Misc

- Configure `grails.config.locations` in Config.groovy for externalized configuration
- Configure log file logging using `ServerAwareFileAppender`

Database

- Oracle RAC $$$
- MySQL Cluster – not a general-purpose solution, joins are expensive, etc.
- C-JDBC, etc.
- com.mysql.jdbc.ReplicationDriver

MySQL ReplicationDriver

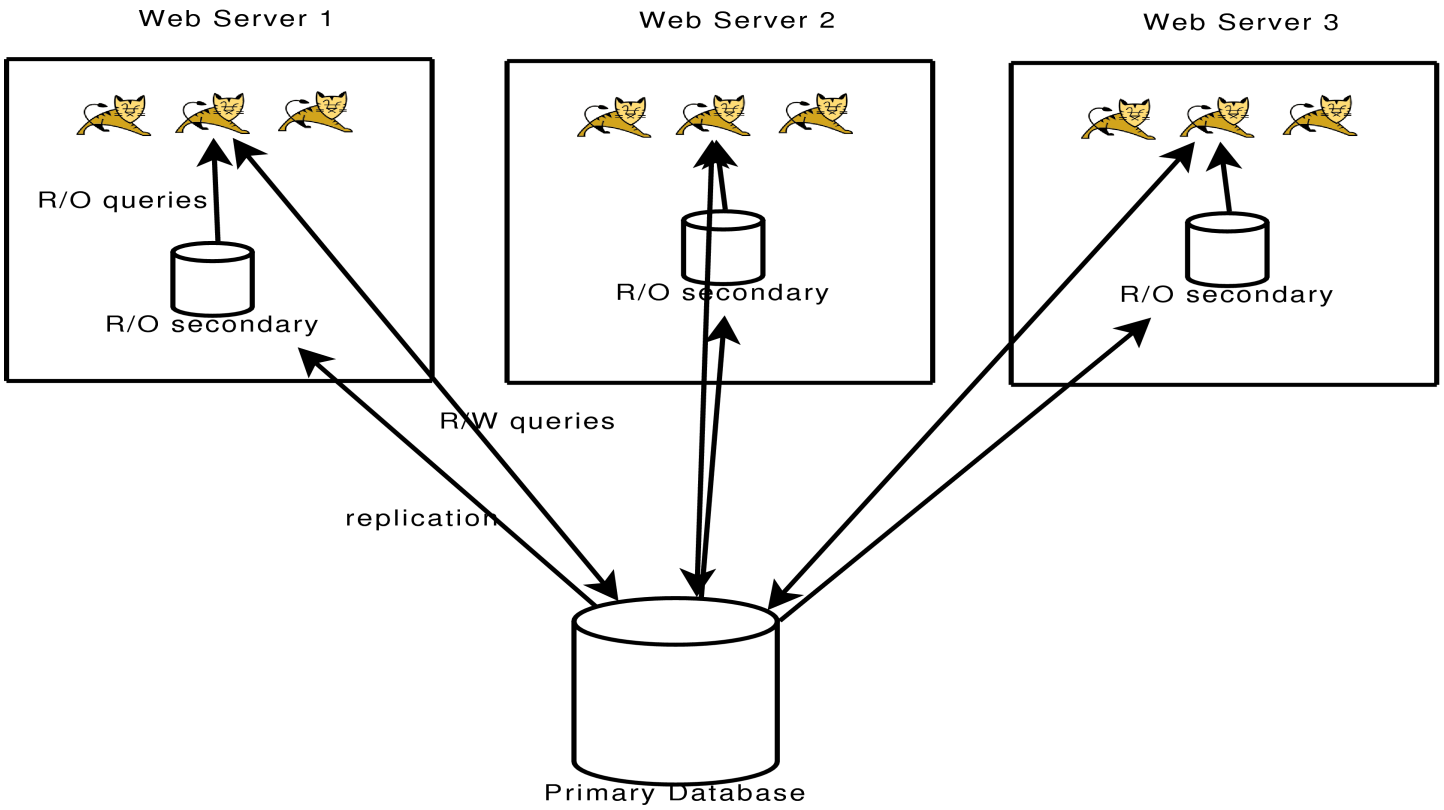
- Looks promising, included with standard MySQL JDBC driver
- Change JDBC URL to primary plus list of read-only replicated servers plus database name, e.g.
- `jdbc:mysql://server1:3306,server2:3306,server3:3306/db`
- Change driver class to com.mysql.jdbc.ReplicationDriver

MySQL ReplicationDriver

- Driver opens connection from primary and from random read-only and uses a connection proxy
- Defaults to read/write primary but switch to read-only by calling `connection.setReadOnly(true)`
- Non-intrusive usage in Grails would require a filter that has controller action mappings to determine if read-only is safe

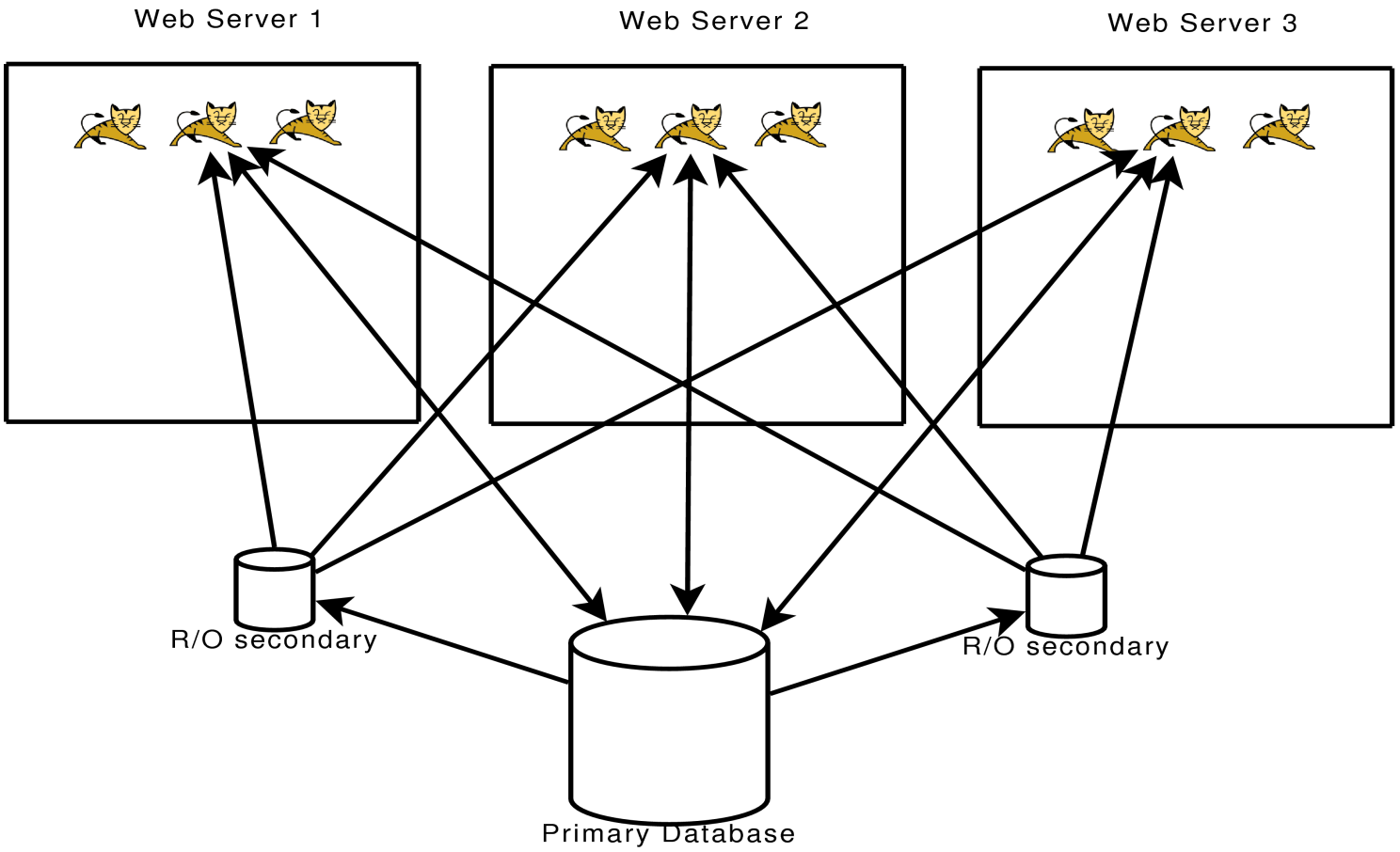
MySQL ReplicationDriver

Read-only replication DB per web server



MySQL ReplicationDriver

Shared read-only replication databases



Q&A