



Tuning Enterprise Applications with JBoss

Phillip Thurmond
QA Performance Engineer

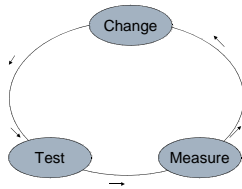


What is Performance?

- **Define the performance goals for your project.**
 - High Throughput
 - Low Latency
 - Specific Memory requirements
- **Sometimes tradeoffs must be made.**



Tuning Methodology



Helpful Hints During Testing

- **Measure and record as much as possible during tests.**
- **Make one change at a time.**
- **Track configuration changes.**



Recommended Tools

- **VCS to record configuration changes.**
- **DB to store results**
- **Collection**
 - Profiler
 - OS tools (vmstat, top, sar, etc.)
 - JMX (JBoss and JVM)
- **Load generator**
- **Automation**
 - Scripts
 - Smart Frog



New JBoss Performance Features

- **JBoss Serialization**
- **Reduced Contention**
- **JBoss Web Server**
 - New threading model
 - Better static performance
- **JBoss Messaging**
 - Enterprise-level messaging
 - Much improved persistence



Tomcat/JBoss Web

- **web.xml**
 - Turn off development mode
 - Set `genStrAsCharArray` to true
 - Set `trimspaces` to true
- **server.xml**
 - Pool sizes
 - Connector options

Tomcat Thread Pool

- **Tomcat**
 - Pure Java.
 - One thread per connection.
 - Does not scale well.

```
<-Connector
.....
maxThreads="250" strategy="ms"
maxHttpHeaderSize="8192"
acceptCount="100"
connectionTimeout="20000"
.....
/>
```

JBoss Web Thread Pool

- **JBoss Web**
 - Based on Tomcat, but interfaces with the native APR library.
 - Worker threads and Polling threads.
 - Scales much better because of reduced thread contention and resource usage.

```
<-Connector
.....
maxThreads="250"
pollerSize="2000"
pollTime="2000"
firstReadTimeout="-1"
.....
/>
```

JVM

- **Use JDK 1.5!**
 - New GC options
 - Much smarter defaults
- **Use Large Pages if it's supported on your platform.**
- **Use all available RAM.**
- **Reduce the stack size.**
 - 1 Mb by default on 64-bit platforms!
 - Large stack size limits the number of threads.
 - Most applications will be fine with 128k or 256k.
- **Increase permanent generation.**

Garbage Collection

- **Select the right GC model**
 - Concurrent
 - Parallel
- **Test using verbose GC logging.**
- **Understanding GC is worth the effort.**

Example JAVA_OPTS

```
JAVA_OPTS="
-Xms4g -Xmx4g
-XX:-DisableExplicitGC
-XX:ParallelGCThreads=8
-XX:+UseLargePages
-XX:+UseParallelOldGC
-verbosegc
"
```

DB

- DB is the bottleneck for most applications.
- Too many DB connections could be as bad as too few.
- Look at JDBC driver performance.
- Use caching.
- Use prepared statements

MySQL

- Use innodb.
- Adjust `innodb_thread_concurrency`.
- Use the latest JDBC driver.
 - Frequent releases with great speed improvements.
 - New major version 5 recently released.

Example my.cnf

```
max_connections=700
table_cache = 1024
sort_buffer_size = 2M
read_buffer_size = 2M
thread_cache = 8
innodb_buffer_pool_size = 5000m
innodb_log_file_size = 1600M
innodb_log_buffer_size = 8M
innodb_flush_log_at_trx_commit = 1
innodb_thread_concurrency = 1000
```

Example MySQL Datasource

```
<local-tx-datasource>
...
<min-pool-size>20</min-pool-size>
<max-pool-size>20</max-pool-size>

<prepared-statement-cache-size>50</prepared-statement-cache-size>

<blocking-timeout-millis>60000</blocking-timeout-millis>
<idle-timeout-minutes>90</idle-timeout-minutes>

<transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
</local-tx-datasource>
```

Clustering

- Use sticky sessions
- New in 4.0.5 – Buddy replication
- Load balancer
 - Apache/mod_jk
 - Software proxy
 - hardware

EJB3

- Turn on caching in `persistence.xml`
- Watch the generated SQL for optimization opportunities.
- Look for the right flush mode for your application.