



Tomcat Tuning

Mark Thomas

April 2009

Who am I?

- Apache Tomcat committer
- Resolved 1,500+ Tomcat bugs
- Apache Tomcat PMC member
- Member of the Apache Software Foundation
- Member of the ASF security committee
- Created the Tomcat security pages
- Senior Software Engineer and Consultant at SpringSource

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - connectors
 - content cache
 - JVM
- Scaling Tomcat
- Hints and tips

Agenda

- **The optimisation / tuning process**
- Tomcat tuning options
 - logging
 - connectors
 - content cache
 - JVM
- Scaling Tomcat
- Hints and tips

The process

- Understand the system architecture
- Stabilise the system
- Set the performance target(s)
- Measure current performance
- Identify the current bottleneck
- Fix the root cause of the bottleneck
- Repeat until you meet the target

Common errors

- Optimising code that doesn't need it
- Insufficient testing
 - realistic data volumes
 - realistic user load
- Lack of clear performance targets
- Guessing where the bottleneck is
- Fixing the symptom rather than the cause

Agenda

- The optimisation / tuning process
- **Tomcat tuning options**
 - logging
 - connectors
 - content cache
 - JVM
- Scaling Tomcat
- Hints and tips

Tomcat tuning

- Applications typically account for >80% of request processing time
- Remember the tuning process
 - Focus your efforts on the bottlenecks

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - **logging**
 - connectors
 - content cache
 - JVM
- Scaling Tomcat
- Hints and tips

Production logging

- Default configuration is generic
- Some settings not ideal for production
 - catch-all logger logs to file and stdout
 - no overflow protection
 - logging is synchronised

Production logging

- Remove duplicate logging (logging.properties)

```
.handlers = 1catalina.org.apache.juli.FileHandler,  
            java.util.logging.ConsoleHandler
```

- becomes

```
.handlers = 1catalina.org.apache.juli.FileHandler
```

- To add rotation

```
1catalina.java.util.logging.FileHandler.pattern =  
    ${catalina.base}/logs/catalina.%g.log  
1catalina.java.util.logging.FileHandler.limit = 20000000  
1catalina.java.util.logging.FileHandler.count = 5
```

Production logging

- Synchronous logging:
 - can become a bottleneck
 - don't want disk IO to become the limiting factor
- Asynchronous logging:
 - log queue uses memory
 - limit queue size to avoid out of memory errors
 - fall back to synchronised logging and/or drop some log messages

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - **connectors**
 - content cache
 - JVM
- Scaling Tomcat
- Hints and tips

Connector tuning

- Need to understand
 - your application usage patterns
 - your network
 - TCP connections
 - HTTP transactions
 - HTTP Keep-Alive
 - SSL
- Additional considerations for load balancing
 - Layer 4 or Layer 7
 - Connection pools

Which connector?

- Java Blocking IO
 - Oldest – most stable
 - JSSE based SSL
- Native (APR)
 - Non-blocking
 - Uses OpenSSL
- Java Non-blocking IO
 - JSSE based SSL

Which connector?

Requirement	Connectors in preference order		
Stability	BIO	APR/NIO	
SSL	APR	NIO	BIO
Low concurrency	BIO	APR	NIO
High concurrency No Keep-Alive	BIO	APR	NIO
High concurrency Keep-Alive	APR	NIO	BIO

Which connector?

- Why would you use the NIO connector?
- The Native (APR) connector is unstable on Solaris
- NIO is a pure Java solution
- It is easy to switch between NIO and BIO with SSL

Connector tuning

- `maxThreads`
 - maximum number of concurrent requests
 - for BIO, maximum number of open/active connections
 - typical values 200 to 800
 - 400 is a good starting value
 - heavy CPU usage → decrease
 - light CPU usage → increase

Connector tuning

- `maxKeepAliveRequests`
 - typical values 1, 100
 - maximum number of HTTP requests per TCP connection
 - set to 1 to disable keep alive
 - disable for BIO with very high concurrency, layer 4 load balancer, no SSL
 - enable for SSL, APR/NIO, layer 7 load balancer
 - Note BIO connector automatically disables keep alive when concurrent connections reach 75% of `maxThreads`

Connector tuning

- connectionTimeout
 - typical value 3000
 - default of 20000 is too high for production use
 - also used for keep alive time-out
 - increase for slow clients
 - increase for layer 7 load balancer with connection pool and keep alive on
 - decrease for faster time-outs

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - connectors
 - **content cache**
 - JVM
- Scaling Tomcat
- Hints and tips

Content cache tuning

- Dynamic content is not cached
- Static content is cached
- Configured using the `<Context .../>` element
- `cacheMaxSize`
 - 10240
- `cacheTTL`
 - 5000
- `CacheMaxFileSize`
 - 512
 - from 6.0.19 onwards
- NIO/APR can use `SEND_FILE`

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - connectors
 - content cache
 - **JVM**
- Scaling Tomcat
- Hints and tips

JVM tuning

- Two key areas
 - Memory
 - Garbage collection
- They are related
- Remember to follow the tuning process

JVM tuning

- Java heap (Xmx, Xms) is not the same as the process heap
- Process heap includes
 - Java Heap
 - Permanent Generation
 - Thread stacks
 - Native code
 - Directly allocated memory
 - Code generation
 - Garbage collection
 - TCP buffers
- Read OutOfMemory exception messages carefully

JVM tuning: memory

- **-Xms/-Xmx**
 - Used to define size of Java heap
 - Aim to set as low as possible
 - Setting too high can cause wasted memory and long GC cycles
- **-XX:NewSize/-XX:NewRatio**
 - Set to 25-33% of total Java heap
 - Setting too high or too low leads to inefficient GC

JVM tuning: ideal garbage collection

- Short lived objects never reach the Old Generation
- Short lived objects cleaned up by short minor garbage collections
- Long lived objects promoted to Old Generation
- Long lived objects cleaned up by (rare) full garbage collection

JVM tuning: garbage collection

- GC pauses the application
 - Regardless of GC algorithm
- Pause can range from milliseconds to seconds
- The pause will impact your response time
 - How much does this matter?
- `-XX:MaxGCPauseMillis` `-XX:MaxGCMinorPauseMillis`
 - Set GC pause time goals
 - More frequent GC, shorter pauses

JVM tuning: garbage collection

- There are many more options
- Useful reference
 - <http://blogs.sun.com/watt/resource/jvm-options-list.html>
- Newer GC algorithms may not behave the way you expect
- Concurrent Mark Sweep
 - `-XX:+UseConcMarkSweepGC`
 - Does not use survivor spaces
 - Can be forced to; not recommended

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - connectors
 - content cache
 - JVM
- **Scaling Tomcat**
- Hints and tips

Scaling Tomcat

- Load balancing
 - Routing requests to multiple Tomcat instances
- Clustering
 - Sharing state between Tomcat instances for fail-over

Scaling Tomcat

- Simplest configuration
 - 1 * httpd
 - 2 * Tomcat instances
 - mod_proxy_http
- Considerations
 - state management
 - fail over

Scaling Tomcat

- Stateless
 - Requests routed to Tomcat instances based purely on load balancing algorithm
 - HTTP sessions will not work
- Adding HTTP session support
 - Tomcat instance maintains HTTP session state
 - 'Sticky sessions'
 - All requests for a session routed to same Tomcat instance

Scaling Tomcat

- Fail over
 - Add session replication - clustering
 - Asynchronous by default so usually used with sticky sessions
 - Single line configuration for defaults
 - Uses multicast for node discovery
 - Will need additional configuration for production use

Agenda

- The optimisation / tuning process
- Tomcat tuning options
 - logging
 - connectors
 - content cache
 - JVM
- Scaling Tomcat
- **Hints and tips**

Hints and tips

- Load balancing / clustering
 - use a minimum of 3 Tomcat instances
 - use load balancing and clustering in your development environment
- Redeployment can expose memory leaks
 - include this in your testing
- Remember to follow the process

Questions?

- markt@apache.org
- users@tomcat.apache.org
- <http://people.apache.org/~markt/presentations>

- mark.thomas@springsource.com
- <http://www.springsource.com/webinars>