Optimising and Tuning Apache Tomcat

Mark Thomas
Senior Software Engineer
6 August 2008

Welcome

- Use the chat window to ask questions
- I'll answer them at the end
- Questions needing longer answers I'll put on my blog:

  http://blog.springsource.com/main/
- If you have a problem, ask in the chat window – Adam is on hand to help you

Tomcat versions

- Focussed on Tomcat 6.0.x
- 5.5.x – bugs and security
- 4.1.x – occasional bugs and security
- 3.x, 4.0.x, 5.0.x are unsupported
- 7.x is on the horizon but no work as yet

About me

- Contributing since 2003
- Tomcat committer, PMC member and Tomcat 4 release manager
- Apache Software Foundation member
- Serve on the Apache Security Committee
- Help maintain the ASF's Bugzillas
- M.Eng. in Electrical & Electronic Engineering

Agenda

- The optimisation / tuning process
- Making the process easier
  - Tomcat configuration
  - Other tools
- Reliability
- Scalability

The process

- Understand the system architecture
- Stabilise the system
- Set performance target
- 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

Understand the architecture

- Major components
- How they interact
  - May help explain random performance issues
- Caching
  - What is cached when?
  - Can it be disabled?

Stabilise the system

- It's harder to work with a moving target
- The fewer changes, the better
- Stop uncoordinated tuning attempts
- Avoid testing on production systems if possible
- Development / Integration test systems need realistic data volumes

Agree the target(s)

- Doesn't matter what they are as long as they are:
  - realistic
  - clear
  - unambiguous
- Should include:
  - data volume
  - concurrency

Measure performance

- Usually means generate requests and time them
- Tools
  - Apache JMeter
  - Numerous other tools
- Live monitoring
  - Access logs – uncomment in server.xml
  - http://localhost:8080/manager/status
  - http://www.lambdaprobe.org/

Identify the bottleneck

- Application logging
- Profiler
- Tomcat logging
- Logs from other components
  - Front-end web server
  - Databases
  - Web services

Application logging

- Avoid using the log() methods in the Servlet API
- Use your favourite logging framework
  - It **must** be class loader aware
- Apache log4j
  - log4j.properties in /WEB-INF/classes
  - log4j-x.y.z.jar in /WEB-INF/lib
- java.util.logging – JULI
  - logging.properties in /WEB-INF/classes

Application logging

- Takes several iterations of configuration to home in on bottleneck
- A small delay in a frequently used method can be harder to spot
- Changes will require a reload
- Only as good as the log statements in the code
  - If you using Spring, AOP makes this easy
- Lots of debug logging will be slow

Profiler

- Usually requires a Tomcat restart to get maximum benefit
- No need to reload application
- Can spot small delays in frequently called methods
- Collecting detailed information can impact performance

Other information sources

- Tomcat logging
  - memory issues
  - thread pool issues
  - other error or warning messages
- Other components
  - check the logs / alerts
  - long running database queries

Fix the root cause

- Re-factor slow code
- Change configuration
- Allocate additional processing threads
- Allocate extra memory
- New hardware

- Need to deploy updated application
- Might need to restart Tomcat
- How to avoid downtime?

Review deployment options

- http://localhost:8080/docs/config/context.html
- WAR or directory
- Inside or outside of appBase
- Optional context.xml file

Application update options

- Should all be seamless
- Small possibility of issues (e.g. bug 43683)
- Updated WAR
  - replace the old WAR
- Update directory
  - replace old files
  - reload application
- Update context.xml
  - replace old file

server.xml update options

- Can perform some actions via JMX
  - save changes / keep server.xml in sync
  - saving changes deletes comments
- Restart Tomcat
  - down time

Avoiding down time

- Clustering
- Multiple Tomcat instances
- Load balancer
  - httpd, IIS, hardware, etc.

- Simple two node cluster
  - httpd 2.2
  - 2 Tomcat 6
  - mod_proxy_http

Cluster considerations

- How stateful is your application?
- Do you really need sessions?
  - If yes, use sticky sessions
- Can a client afford to lose a session on node failure / shut down?
  - It not, use session replication
- How to connect front-end to Tomcat?
  - mod_proxy_http recommended

The stateless approach

- Sessions will not work
- Requests will alternate between Tomcat instances
- Multiple Tomcat instances
  - same machine (must use different IP / port)
  - different machines
- Required httpd modules:
  - mod_proxy, mod_proxy_balancer, mod_proxy_http

The stateless approach

- httpd.conf

```
# Cluster definition
<Proxy balancer://devcluster>
  BalancerMember http://192.168.0.31:8080 disablereuse=On
  BalancerMember http://192.168.0.32:8080 disablereuse=On
</Proxy>
# Pass all requests except the manager to the cluster
ProxyPass /balancer-manager !
ProxyPass / balancer://devcluster/
# Configure the manager
<Location /balancer-manager>
  SetHandler balancer-manager
  Order Deny,Allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

Add sticky session support

- Tomcat configuration
  - server.xml
  - Set <Engine jvmRoute="tc01"... />
  - jvmRoute must be unique for each instance

Add sticky session support

- httpd.conf

```
# Cluster definition
<Proxy balancer://devcluster>
  BalancerMember http://192.168.0.31:8080 route=tc01
  disablereuse=On
  BalancerMember http://192.168.0.32:8080 route=tc02
  disablereuse=On
</Proxy>

# Pass all requests except the manager to the cluster
ProxyPass /balancer-manager !
ProxyPass / balancer://devcluster/
        nofailover=On stickysession=JSESSIONID|jsessionid

# Balancer as before
```

Add session replication

- Application configuration
  - WEB-INF/web.xml
  - Add the <distributable/> element
- Keep the session as small as possible
- Session attributes must implement Serializable

Add session replication

- Tomcat configuration - server.xml
  - Uncomment <Cluster ... /> element under <Engine ... >
- The same <Cluster .../> element can also be used under <Host ...
- Defaults to get you started
- Overview:
  - /docs/cluster-howto.html
- Details:
  - /docs/config/cluster.html

Add session replication

- httpd.conf

```
# Cluster definition
<Proxy balancer://devcluster>
  BalancerMember http://192.168.0.31:8080 route=tc01
  disablereuse=On
  BalancerMember http://192.168.0.32:8080 route=tc02
  disablereuse=On
</Proxy>

# Pass all requests except the manager to the cluster
ProxyPass /balancer-manager !
ProxyPass / balancer://devcluster/
        nofailover=Off stickysession=JSESSIONID|jsessionid

# Balancer as before
```

How to avoid down time

- Use manager to disable route to tc01
- Wait for current requests to finish
- Upgrade / reconfigure instance tc01
- Use manager to enable route to tc01
- Repeat for tc02

Reliability

- Redeployment can cause memory leaks
- Caused by static references to the class loader
  - shared libraries
- Include redeployment in testing

Scalability

- Design with this in mind
- Make sure the application works in a cluster
- To increase capacity, add new Tomcat instances
- Use interfaces at a few key points to provide options to off-load processing

Concurrent requests

- Tomcat maintains a thread pool
- Incoming request is assigned to a free thread
- If all threads are busy, request is queued
- If queue is full, request is rejected
- Running out of threads is **usually** a symptom of an application issue

Concurrent requests

- If you do need more threads – server.xml
  - maxThreads
- For highly concurrent environments
  - maxKeepAliveRequests=”1”
  - connectionTimeout=”3000”

Questions?

Mark Thomas
mark.thomas@springsource.com

http://springsource.com

Thank You for Attending

Find out more at:

http://tomcat.apache.org

SpringSource Enterprise support:

http://springsource.com/support

insidesales@springsource.com

+1 800-444-1935