



University of Southern California Shibboleth High Availability with Terracotta

Russell Beall
Senior Programmer/Analyst
Information Technology Services
University of Southern California
Los Angeles, California, USA
beall@usc.edu

Overview

- Intro to HA architecture with Terracotta
- Benefits
- Drawbacks
- Shibboleth and Terracotta at USC
- Monitoring
- Issues Resolved
- Tracking Down Issues

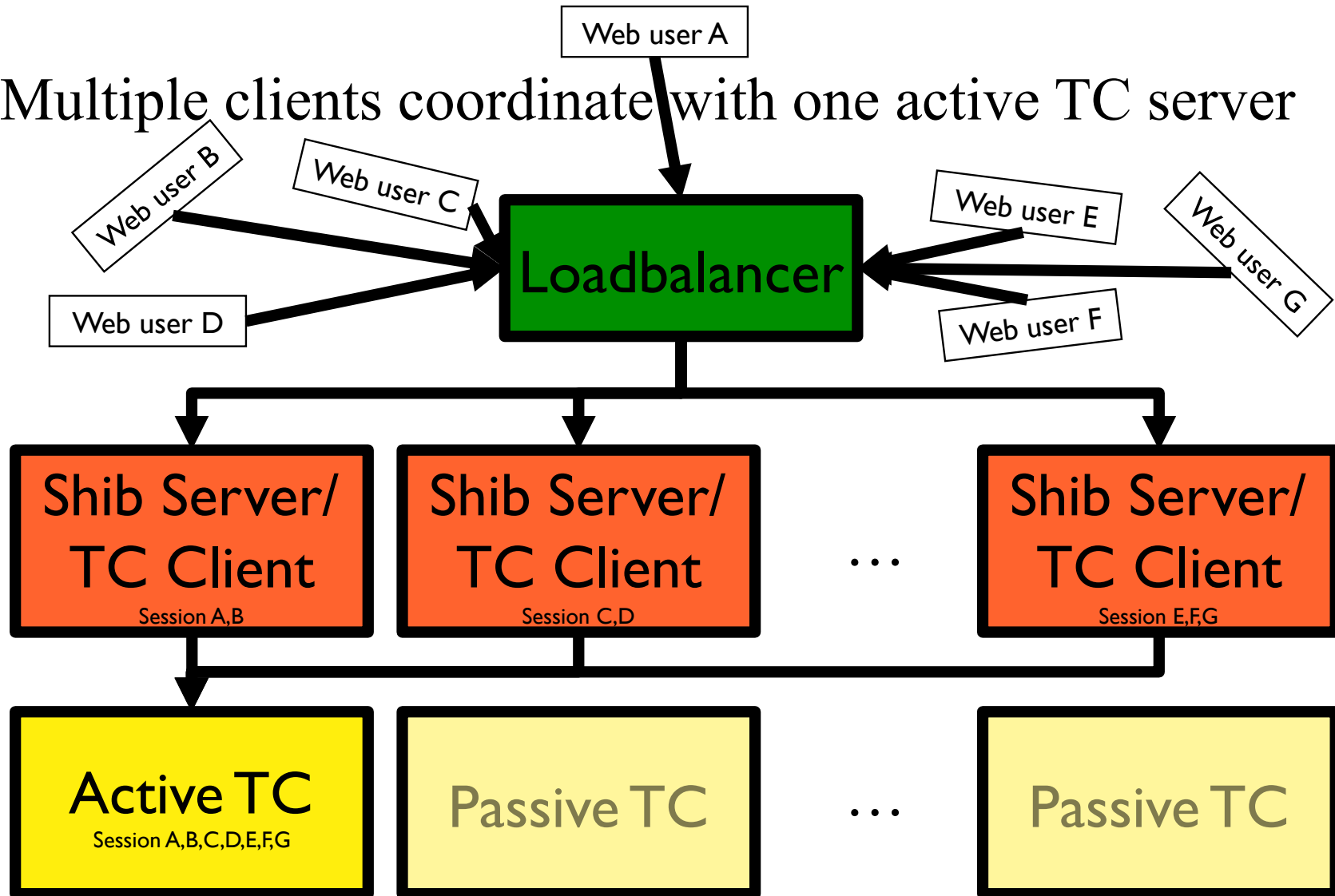
Introduction to HA with Terracotta

Terracotta is designed to assist:

- High availability with failure protection
- Distribution of client load amongst several machines
- Sharing of session data between clients as needed
- Preserving session data on disk (can survive cluster-wide full restarts if done right)
- Automatic failover in the event of node failure

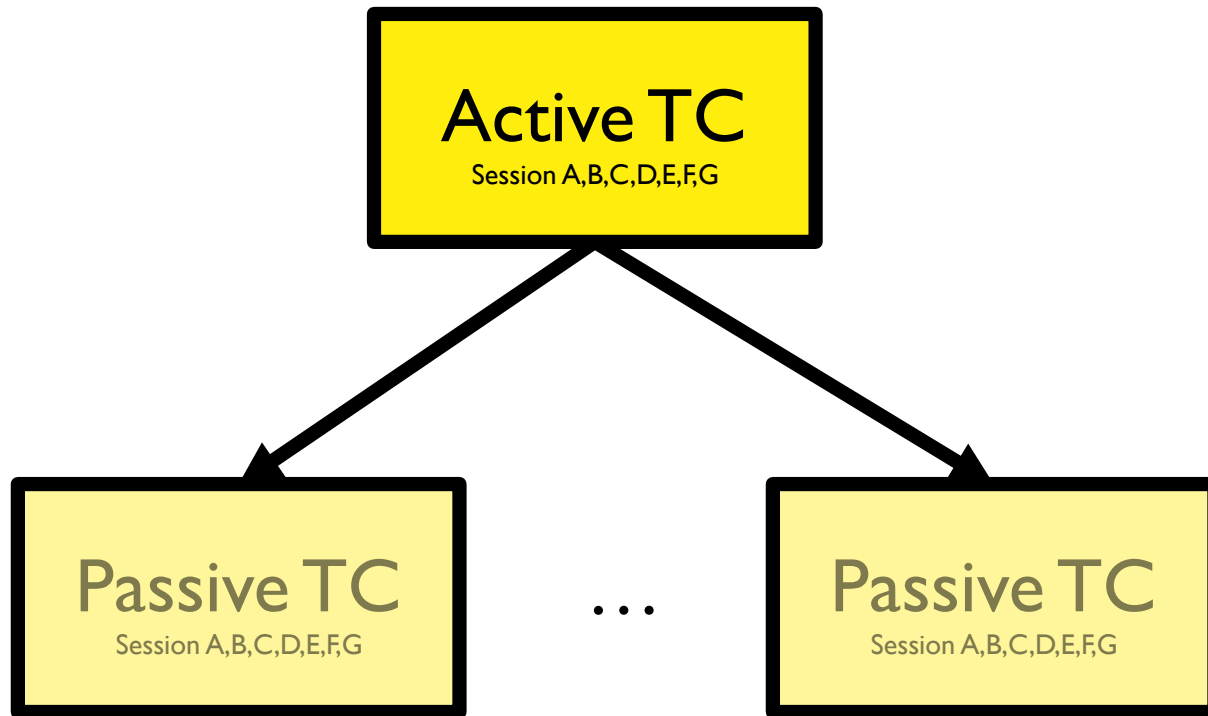
Terracotta Basics

Multiple clients coordinate with one active TC server



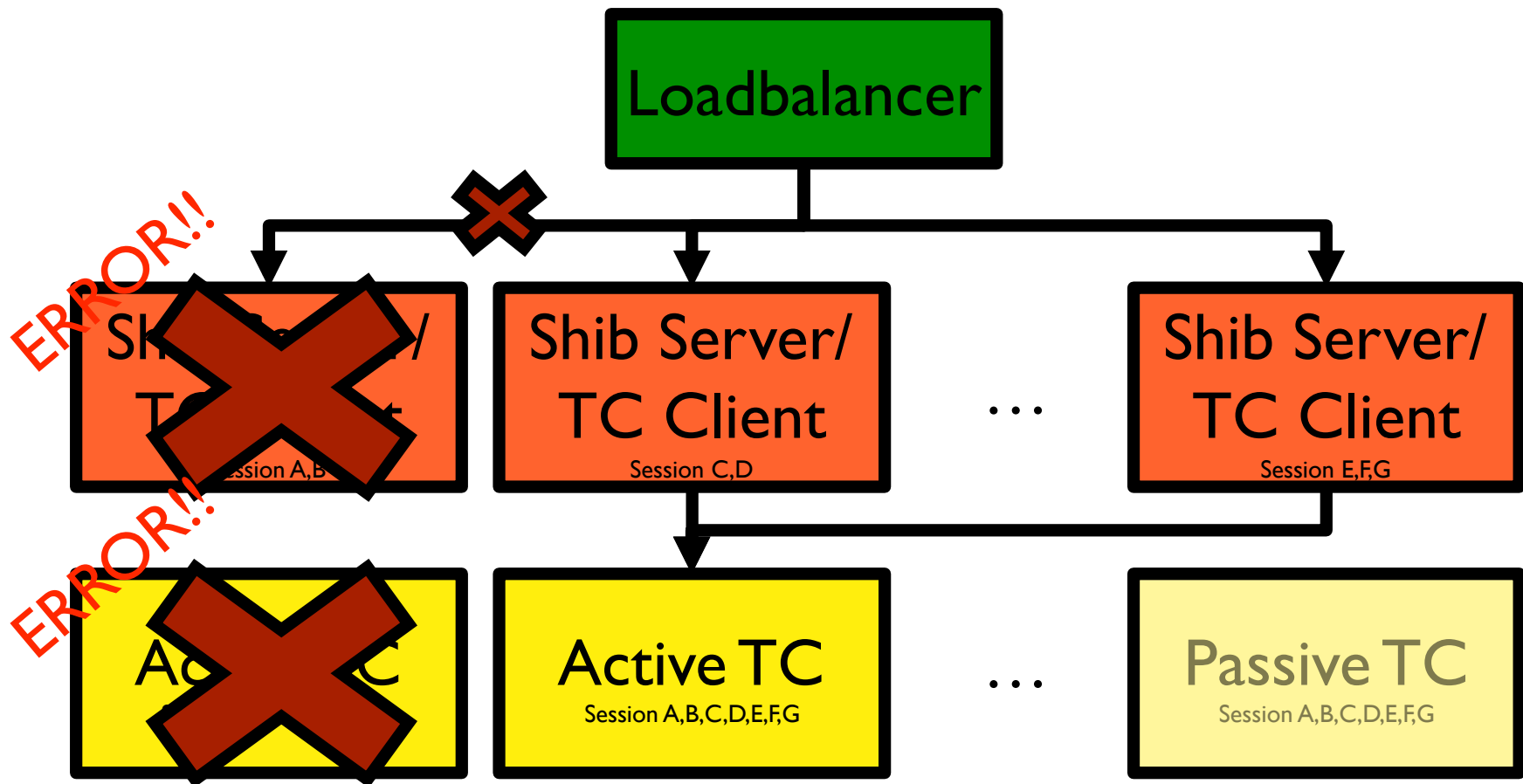
Terracotta Basics

Active server distributes session data to passive



Terracotta Basics

In the event of failure, a new Active server is automatically selected. Clients automatically fail over.



Terracotta Basics

- Terracotta servers can reside on hardware independent of the clients
 - recommended by Terracotta docs
 - good when many smaller machines are available
- Terracotta servers can also reside on the same hardware
 - recommended by Shibboleth docs
 - good where fewer, but larger machines, with more cores and more memory are available

Terracotta Basics

- Proper application failover requires:
 - Load balancing software or hardware with failure detection
 - Redundant nodes (the more the merrier)
 - Sized to handle the load on a subset of nodes
 - ❖ Users should not notice that one or perhaps multiple nodes break (ideally)

Benefits

- Automatic failover and retention of services, with the help of a loadbalancer, in the event of service failure
- Enables services to be taken down and restarted as needed without affecting users
- Preserves session data on restarts
- Additional clients are easily configured and added
- And more...

Benefits

- Decreased memory requirements for application server (Tomcat/JBoss) on each machine:
 - session data will be split between multiple boxes and will not be loaded on another box unless the user is re-routed or an attribute query hits a different box
 - session objects which otherwise might have pushed memory limits to the max will be shunted off by the LFU algorithm when memory usage exceeds configurable thresholds – client shunts to TC service, TC service shunts to disk

Drawbacks

- Increased processing overhead
 - Active TC server consumes nearly as much CPU time as the Tomcat server on that box
 - Increased memory requirements to run TC server on each box – somewhat balanced by decreased memory for tomcat
- Garbage collection in TC server may produce delays in addition to client JVM GC (subject to tuning)
- Potential issues on network disconnects requiring manual service restarts

Drawbacks

- Authentication modules other than LDAP JAAS may require significant Terracotta configuration to enable.
- Benefits are questionable if not using AttributeQuery or other data services requiring session objects to be accessible from each server.
- If TC goes down, all clients block, even the stop/start scripts for the Tomcat servers will block.

Benefit/Drawback Comparison

- Drawbacks boil down mainly to increased overhead
 - Easily solved by adding a bit of hardware.
- Persistence of user sessions with transparency through restarts is very desirable and facilitates configuration changes.
- Terracotta will be required for redundancy where session sharing is used.
- Setup of Terracotta has proven easy enough and reliable enough to justify its use.

Shibboleth and Terracotta at USC

- Current configuration
 - Two Sun v240s – 2 core, 8 GB ram (soon to be retired)
 - ❖ Processes 30K+ logins per day on each machine
 - ❖ Load average is regularly 4 or greater during peak times
 - ❖ If one were to fail, the other could get overloaded
 - ❖ Not much room to grow
 - ❖ Easily handles IdP traffic without TC, but can get rather loaded when TC server also competes
 - Citrix NetScaler loadbalancing
 - ❖ Distributes load evenly
 - ❖ Maintains client affinity

Shibboleth and Terracotta at USC

- Future configuration

- Two Sun v440s – 4 cores, 16 GB ram

- ❖ Additional memory for increased heap sizes

- ❖ increased time between garbage collection cycles

- ❖ decrease or eliminate the occasional failure to promote from the New Generation when using the Concurrent Mark Sweep collector (results in a stop-the-world RAM defrag, 10-20 seconds).

- ❖ Enough CPU (hopefully) to turn AttributeQuery back on for 1.3 SPs and Encryption for 2.x SPs

- ❖ Extra processors will reduce contention, especially GC*

Monitoring

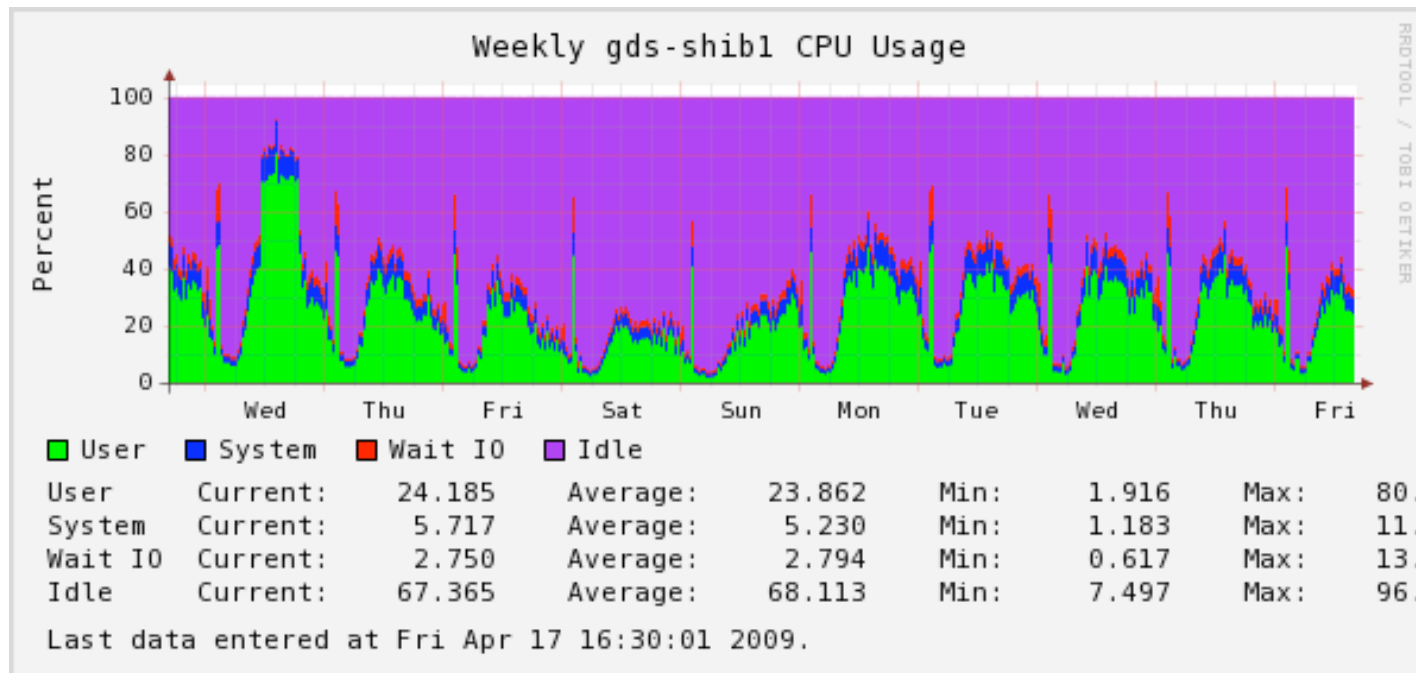
- <https://shibboleth.usc.edu/idp/profile/Status>
 - Standard monitoring profile handler – not used
 - Returns the string “ok”
- <https://shibboleth.usc.edu/shibboleth-idp/Status>
 - Custom jsp script – monitors Tomcat only (for now)
 - Returns the string “AVAILABLE”
 - Used by loadbalancer and Nagios
 - “ok” was considered too common and likely to occur in web communication – even potentially in an error case

Monitoring

- Orca

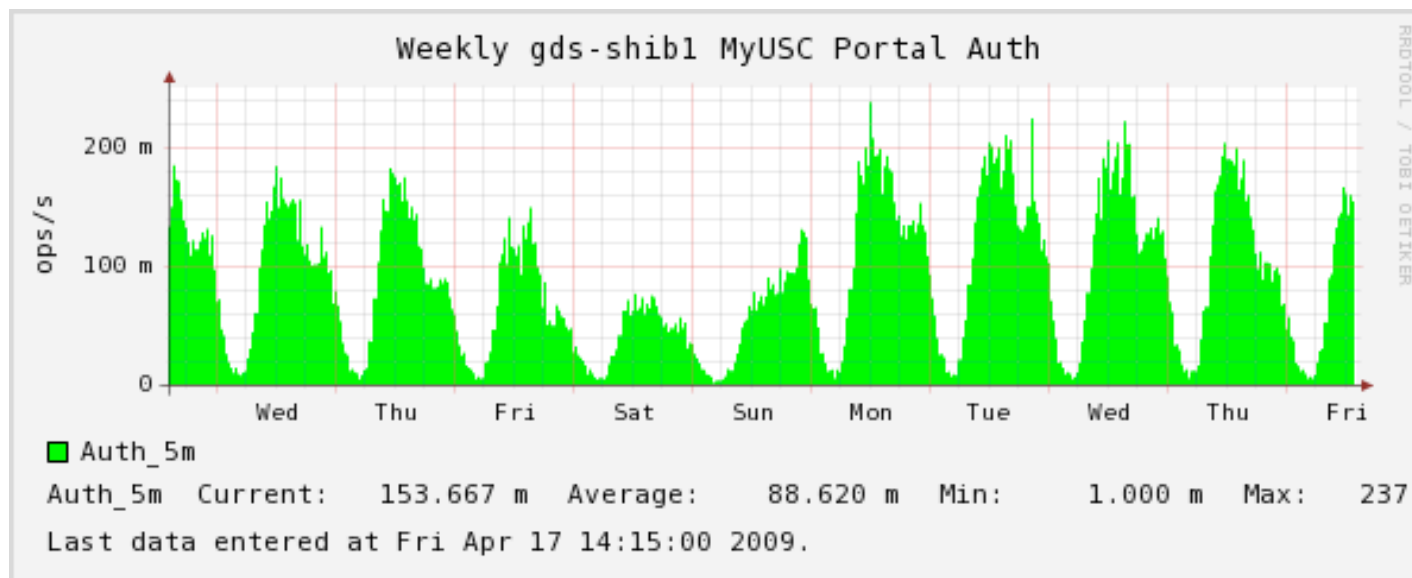
- <https://gds-admin.usc.edu/orca/>

- example:



Monitoring

- shlook.pl
 - custom plugin to Orca to monitor Shibboleth specific stats
 - available on same page as CPU stats
 - example:



Monitoring

- Terracotta admin console:



Monitoring

- Unix Terminal (I tend to watch behaviors with these):
 - top
 - follow the idp-process.log
 - catalina.out for garbage collector stats
 - `while [[true]]; do ps -auwwx|grep -c httpd; sleep 5; done;`
- Other logs:
 - terracotta-server.log
 - terracotta-client.log

Issues Resolved

- Disk Garbage Collector
 - By default, runs every hour
 - Should be turned off and triggered by script at night
- Log space
 - Many unimportant warnings printed to catalina.out
 - Remedied by
 - ❖ cronolog rolling the catalina.out file
 - ❖ swallowOutput tomcat option plus logging.xml config
 - ❖ May not be an issue with recently released TC 3.0

Issues Resolved

- Memory Management
 - With TC, memory turnover is high
 - May be remedied by ConcurrentMarkSweep collector (if you have enough heap space)
- Network Connectivity
 - Network hiccups may disconnect servers
 - TC servers auto-recover, clients do not
 - May be remedied by configuring reconnect windows

Tracking Down Issues

- `idp-process.log`
 - Where access messages build up but no authentication messages resolve the request, TC service may be down
- `terracotta-client.log/terracotta-server.log`
 - Indicates when connectivity is available or in error
 - Shows heap usage stats
 - TC memory monitor messages and alerts, e.g.:
 - ❖ long garbage collection cycles
 - ❖ long times spent on LFU to shunt active objects out of memory

(Zen and the art of)

Tracking Down Issues

- Just look
 - Examine logs
 - ❖ Get a feel for normal behavior to then be able to spot the abnormal
 - ❖ Trace time stamps and correlate between logs
 - Research
 - ❖ Learn the meanings of log messages and configuration options
 - ❖ Read the documentation of the products
 - ❖ Open source opens opportunities – you can look at the code
- Persist

Links

- USC:
 - <http://www.usc.edu>
- Terracotta Integration and Streamlining
 - <https://spaces.internet2.edu/display/SHIB2/IdPCluster>
- This presentation
 - <http://its.usc.edu/~beall/Shibboleth-Terracotta-Spring09I2.ppt>
- Russell Beall
 - beall@usc.edu