# **Apache Tomcat Clustering**

Mark Thomas, Staff Engineer



## Agenda

- Introductions
- Terminology
- When to cluster
- Components
- Configuration choices
- Debugging
- Questions



# Introductions



## Introductions

- Mark Thomas
- Apache Tomcat committer (markt)
- Other ASF
  - Infrastructure team
  - Security
  - Commons
  - Member
- Staff Engineer at VMware
  - Tomcat
  - Security
  - tc Server
  - support

# Terminology

## Terminology

### Clustering

- Has lots of meanings / uses
- In Tomcat, we mean replicating HTTP session data between multiple nodes

## Load-balancing

• Using a reverse proxy to route traffic for a site to more than one Tomcat instance

## Sticky sessions

 Configuring the load-balancer so requests associated with a session are always routed to the same node

## When to cluster



## When to cluster

#### Ideally, never

- Adds configuration complexity
- Requires additional processing
- Debugging is lot harder

#### What do you really need?

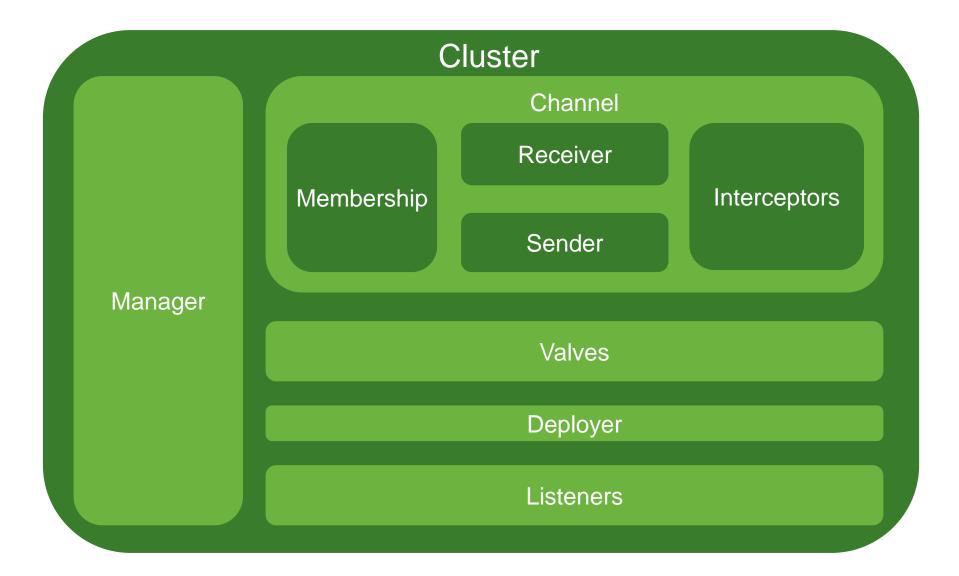
- Load-balancing plus sticky sessions
- If a node fails, sessions will be lost

#### Clustering should be the last resort

## Components



## Components





# **Configuration choices**



#### Two options

- Delta manager
- Backup manager

## Delta manager

- Default
- Replicates every change to every node
  - Maximum reliability
- Network traffic proportional to the square of the number of nodes
  - Doesn't scale to large numbers of nodes
- Fail-over can be to any node

#### Backup manager

- Sessions have a primary node and a backup node
  - Need to use sticky sessions
- Backup node selected on a round-robin basis from all other nodes
- There is NOT a single backup node
- Every node knows the primary node and backup node for every session
- Network traffic proportional to the number of nodes
- Failover is more complicated

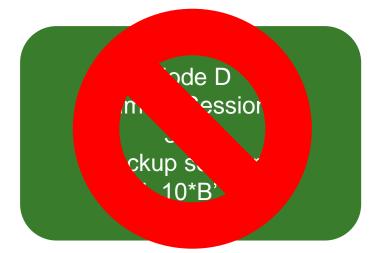
Node A Primary Sessions: 30\*A Backup sessions: <u>10\*B', 10\*</u>C', 10\*D'

Node B Primary Sessions: 30\*B Backup sessions: 10\*A', 10\*C', 10\*D'

Node C Primary Sessions: 30\*C Backup sessions: 10\*A', 10\*B', 10\*D' Node D Primary Sessions: 30\*D Backup sessions: 10\*A', 10\*B', 10\*C' Node A Primary Sessions: 30\*A Backup sessions: 10\*B', 10\*C', 10\*D'

Node B Primary Sessions: 30\*B Backup sessions: 10\*A', 10\*C', 10\*D'

Node C Primary Sessions: 30\*C Backup sessions: 10\*A', 10\*B', 10\*D'



## **Configuration choices: Manager**

Node D fails

#### Sessions will be distributed to other nodes

As soon as node failure is detected

#### If new node was the backup

- It becomes the primary
- A new backup node is selected
- Session is copied to new backup node

#### If new node was not the backup

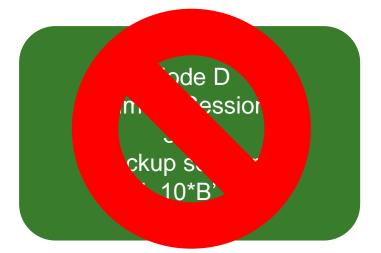
- It becomes the primary
- The backup node remains the same
- Session is copied from the backup node

#### Sessions are re-distributed amongst remaining nodes

Node A Primary Sessions: 40\*A Backup sessions: 20\*B', 20\*C'

Node B Primary Sessions: 40\*B Backup sessions: 20\*A', 20\*C'

Node C Primary Sessions: 40\*C Backup sessions: 20\*A', 20\*B'



#### Two options

- Multicast
- Static

#### Multicast membership

- Requires multicast to be enabled on the network
- Can be difficult to debug problems
- Scales more easily

#### Static

- Simple to debug
- Adding nodes gets time consuming as cluster grows

## **Configuration choices: sendOptions**

#### Delta manager

channelSendOptions on Cluster

#### Backup manager

- mapSendOptions on Manager
- Synchronous or asynchronous

## **Configuration choices: sendOptions**

#### Synchronous

- Request processing does not complete until session data has been sent
- What is meant by sent?
  - On the TCP stack
  - Received by the other node
  - Processed by the other node
- Next request to a different node will see updated sessions

#### Asynchronous

- Request processing does not have to wait for session data to replicate
- Next request to a different node may not see updated sessions

## **Configuration choices: Summary**

## Manager

- Delta or Backup
- Sticky sessions
  - Yes or no

## Membership

Multicast or static

## Send options

Synchronous or asynchronous



# Debugging



- Need to know
  - Session ID
  - Current route
  - Which node handled the request
- I use a simple JSP page that shows all of the above
- Quickly test behaviour is as expected
  - Is the route correct for the current node
  - Is load-balancing happening as expected
  - Is fail-over happening as expected
- Keep in mind how reverse proxy handles failed nodes

## **Debugging: Application problems**

- Just like trying to debug any other application problem
  - But harder
- Can the issue be replicated in a non-clustered environment?
- Approach depends a lot on the application
- Network / failover issues
  - Look at the access logs (need session IDs)
  - Look at error logs
  - May need to look at network traffic

### Application issues

- Logging, logging and more logging
- Need to be able to fine tune logging

## Questions

