JBoss® WORLD

ORLANDO 2008

PRESENTED BY RED HAT

Orlando, Florida

February 13-15, 2008

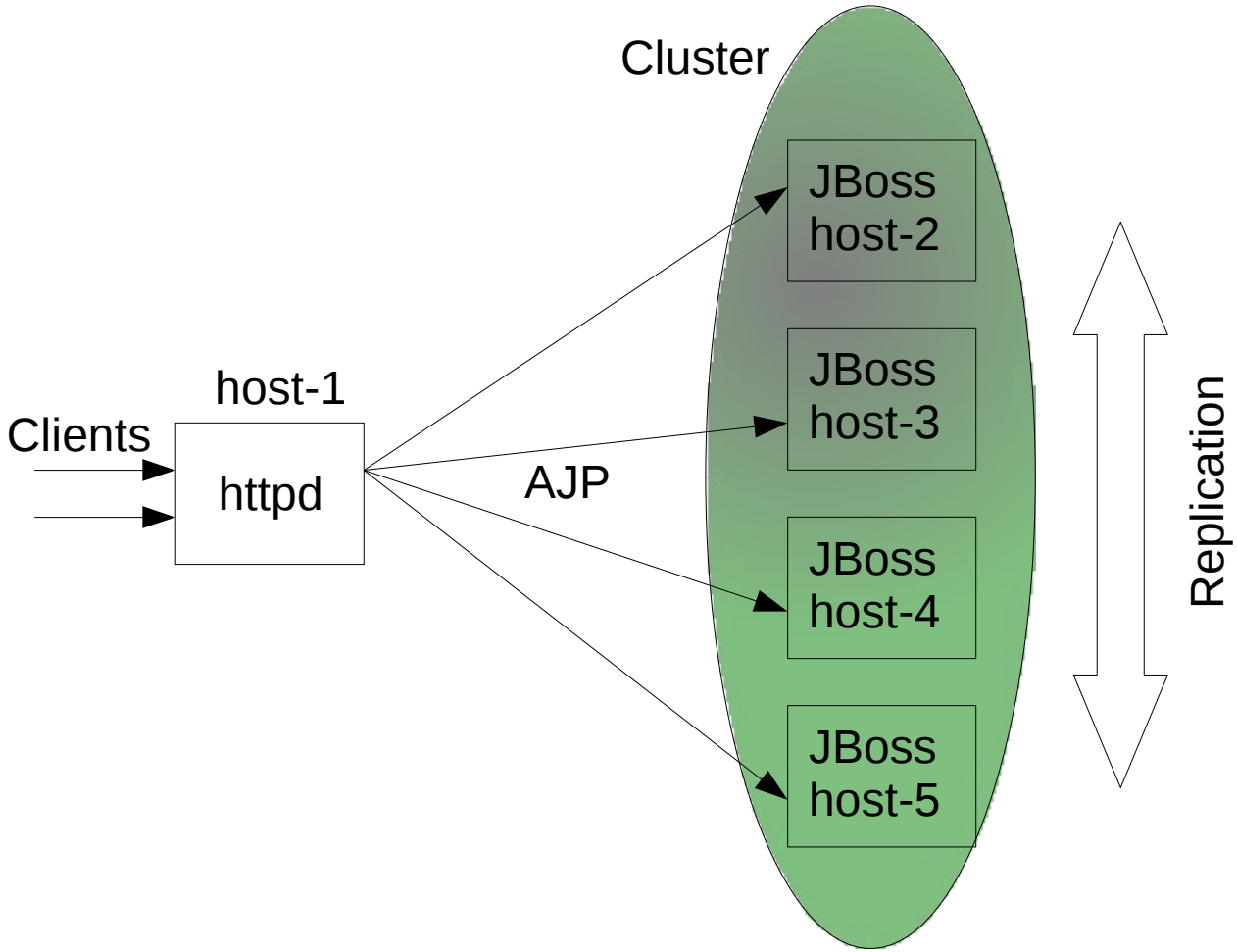
Cluster Tuning:
Getting the best performance out of your JBoss cluster

Bela Ban, Lead JGroups
Brian Stansberry, Lead JBoss AS Clustering

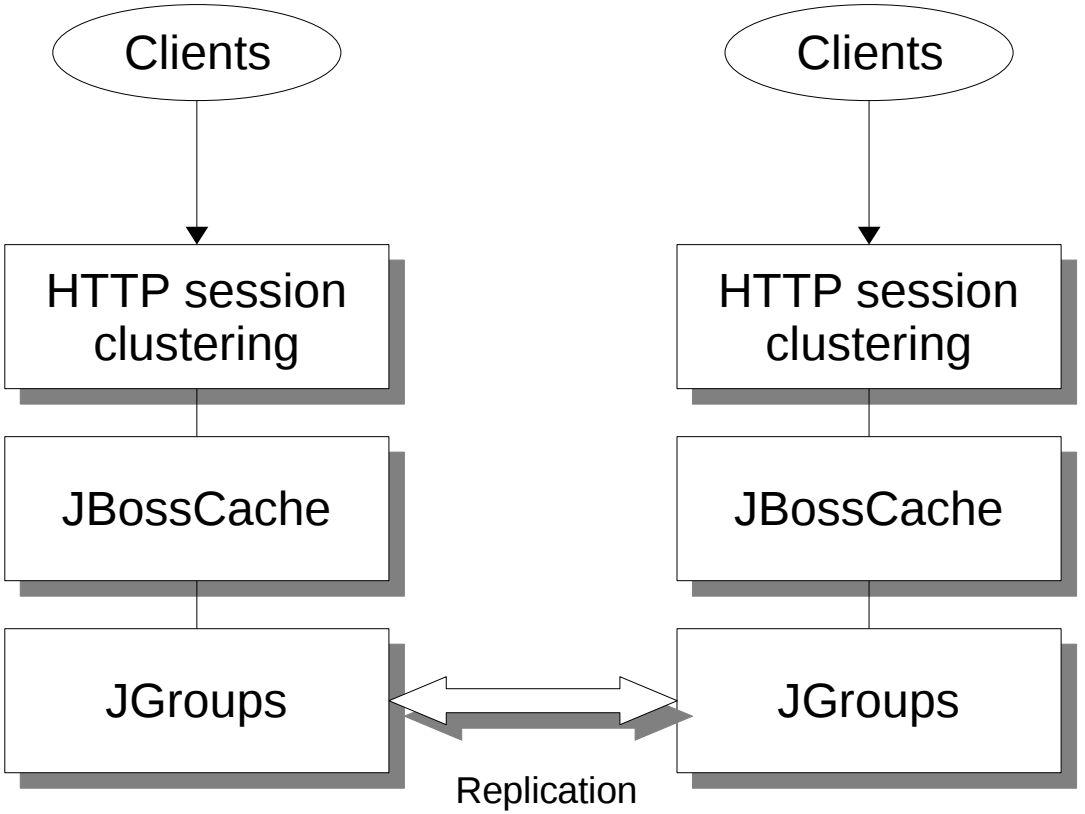
Goals

- Short overview of clustering and the test setup
- Which knobs can you turn and what's the effect on performance
  - Focus is on “end to end performance”
  - Not on replication performance
- Additional tips & tricks

Topology


Cluster
JBoss
host-2
JBoss
host-3
JBoss
host-4
JBoss
host-5
host-1
Clients
httpd
AJP
Replication

HTTP session clustering - architecture



Setup

- Client simulation with java.net.URLConnection
  - We can run any number of threads (clients)
  - Apache
    - mod-jk, workers.properties, urimapping, modjk.conf
  - JBossWeb
    - jvmRoute=”node1” (jboss-web.deployer/META-INF/server.xml)
    - UseJK=”true” (jboss-web.deployer/META-INF/jboss-service.xml)

Configuration

- web.xml: add <distributable/>
- jboss-web.xml
  - Replication granularity (what is replicated ?)
    - SESSION (default): replicate entire session after request
    - ATTRIBUTE: replicate only modified attribute(s)
    - FIELD: replicate modified fields of attribute values

Configuration

- jboss-web.xml
  - Replication trigger (when is data replicated ?)
    - SET
      - When setAttribute() is called
      - getAttribute() does **not** replicate !
    - SET_AND_GET
      - getAttribute() **or** setAttribute()
    - SET_AND_NON_PRIMITIVE_GET (default)
      - setAttribute(), and getAttribute() which returns non primitive types, e.g:
        - Collections, arrays, Pojos
  - Only one replication per request (at end of request)

Replication mode

- Synchronous
  - The HTTP response blocks until the changes to the session have been replicated through the cluster and acknowledgments have been received
- Asynchronous
  - The HTTP response blocks only until the replication message is put on the wire
  - This is faster, but on failover all the changes may not have been received yet by everyone
- Configuration
  - REPL_SYNC or REPL_ASYNC in the JBossCache config (deploy/jboss-web-cluster.sar/META-INF/jboss-service.xml)

Buddy vs. Total Replication

- Total Replication (default)
  - Session is replicated to all nodes of cluster
    - N cluster nodes == each node uses memory, CPU, network resources to provide backup for N-1 nodes
    - Doesn't scale (data wise)
- Buddy Replication
  - Session is replicated to a configurable number of backup “buddies” (default is 1)
- Configuration
  - Set “buddyReplicationEnabled” to “true” in the JBossCache config (deploy/jboss-web-cluster.sar/META-INF/jboss-service.xml)

Perf application

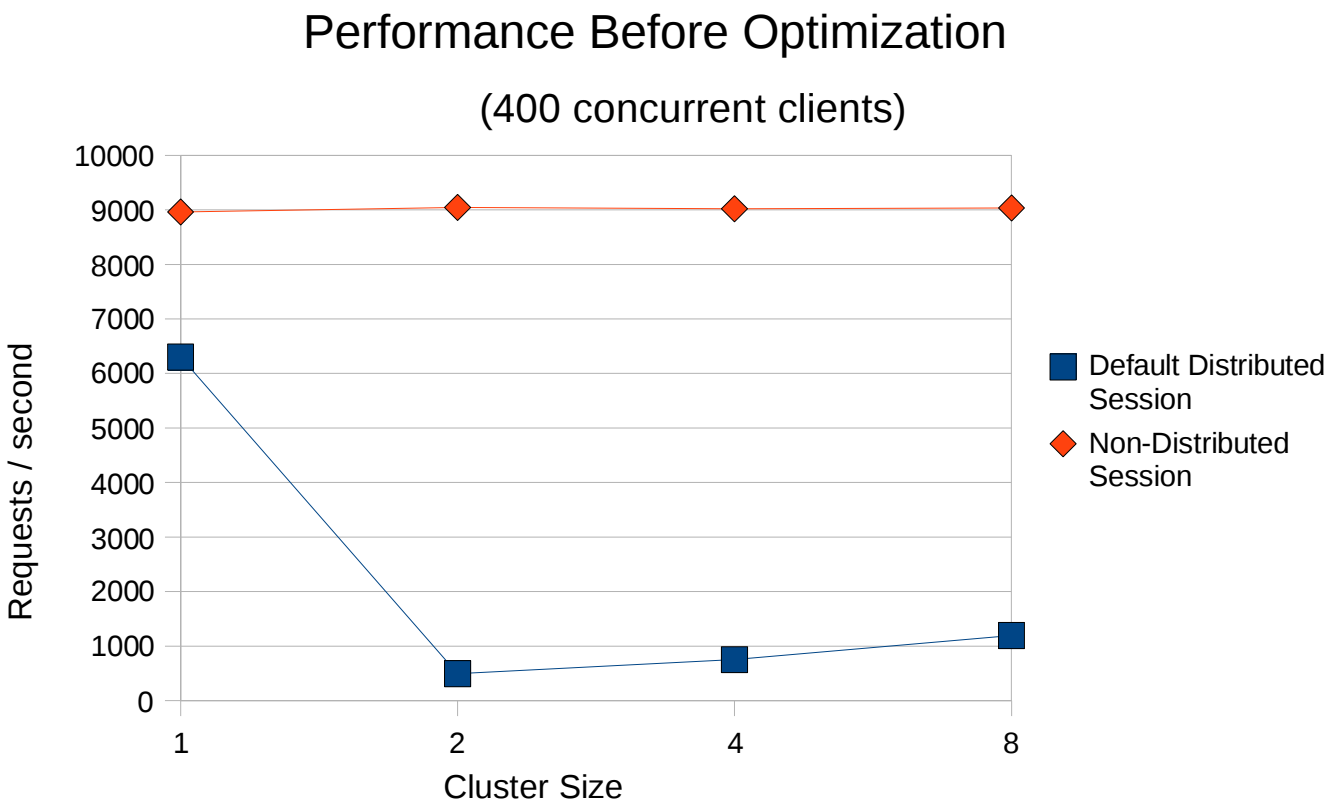
- HTTP sessions have ints as keys and byte[] buffers as values
- Each client
  - Creates HTTP session with `num_attrs` (10) attributes with `size` (2500) byte[] buffers
  - Starts timer
  - Loops X times
    - With 10% chance, writes a random key, **or**
    - With 90% chance, reads a random key
  - Stops timer
  - Destroys HTTP session

Quiz

- Each session has 10 attributes, each attribute key is an int (1-10); each attribute value is a byte[2500] buffer
- What happens on session.getAttribute(“5”) with default granularity (SESSION) and trigger (SET_AND_NON_PRIMITIVE_GET) ?
  - A: nothing is replicated
  - B: approx. 2'500 bytes are replicated
  - C: approx. 25'000 bytes are replicated
  - D: none of the above

Unoptimized Numbers

- Replication carries a heavy cost
  - Need to optimize!



First Tip: Use ATTRIBUTE Granularity

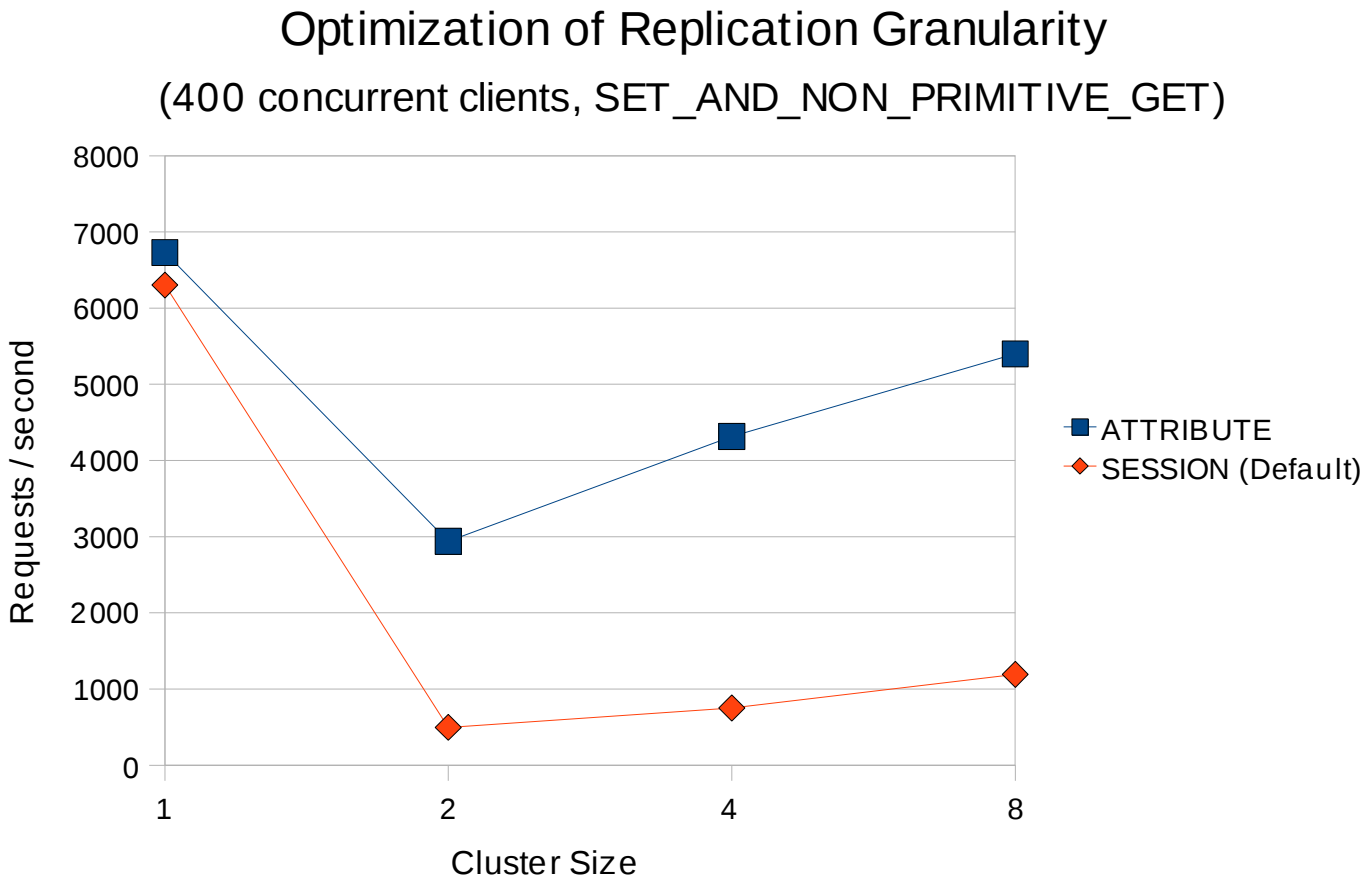
- Reduce amount of data replicated
- Session in our test app has 10 attributes, 2.5KB in each attribute
  - We access one attribute per request
  - SESSION: we replicate ~ 25KB per request
  - ATTRIBUTE: we replicate ~ 2.5KB per request

jboss-web.xml

```
<replication-config>
   <replication-granularity>ATTRIBUTE</replication-granularity>
</replication-config>
```

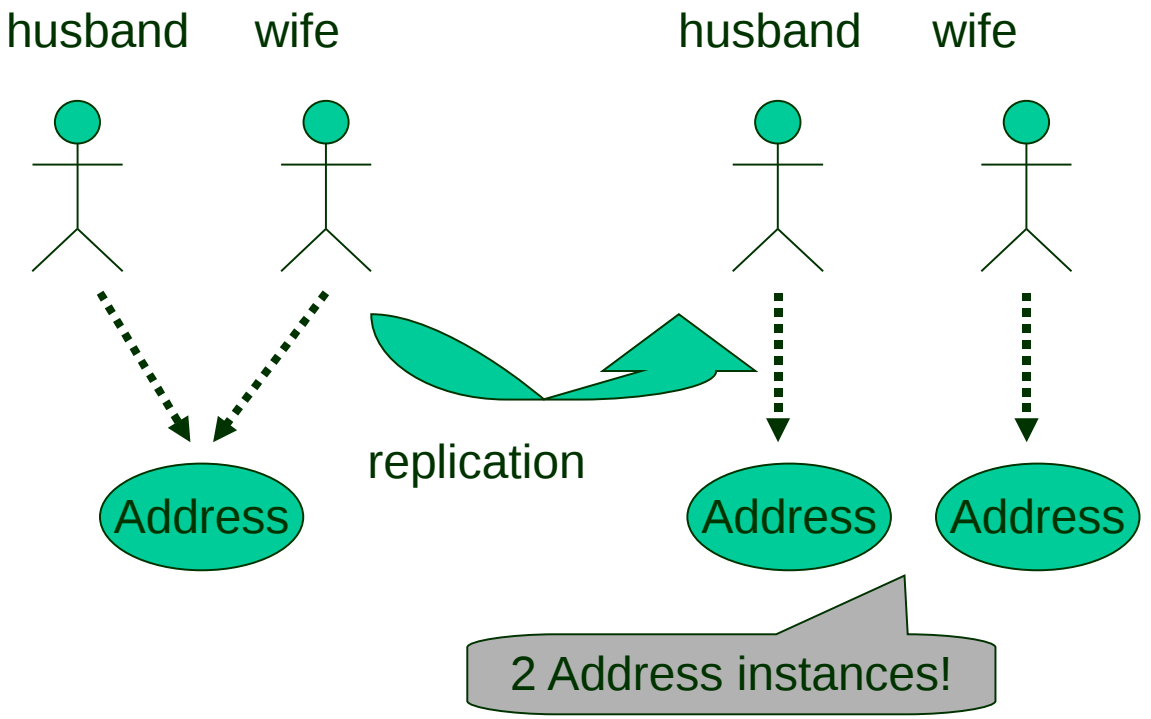
Effect of Using ATTRIBUTE

- Approximately a 5x improvement



Caveat: Object relationships

- Be careful with shared object refs between attributes
  - With ATTRIBUTE they are separately serialized
  - On remote nodes, refs will no longer be shared!



Next Tip: Use SET

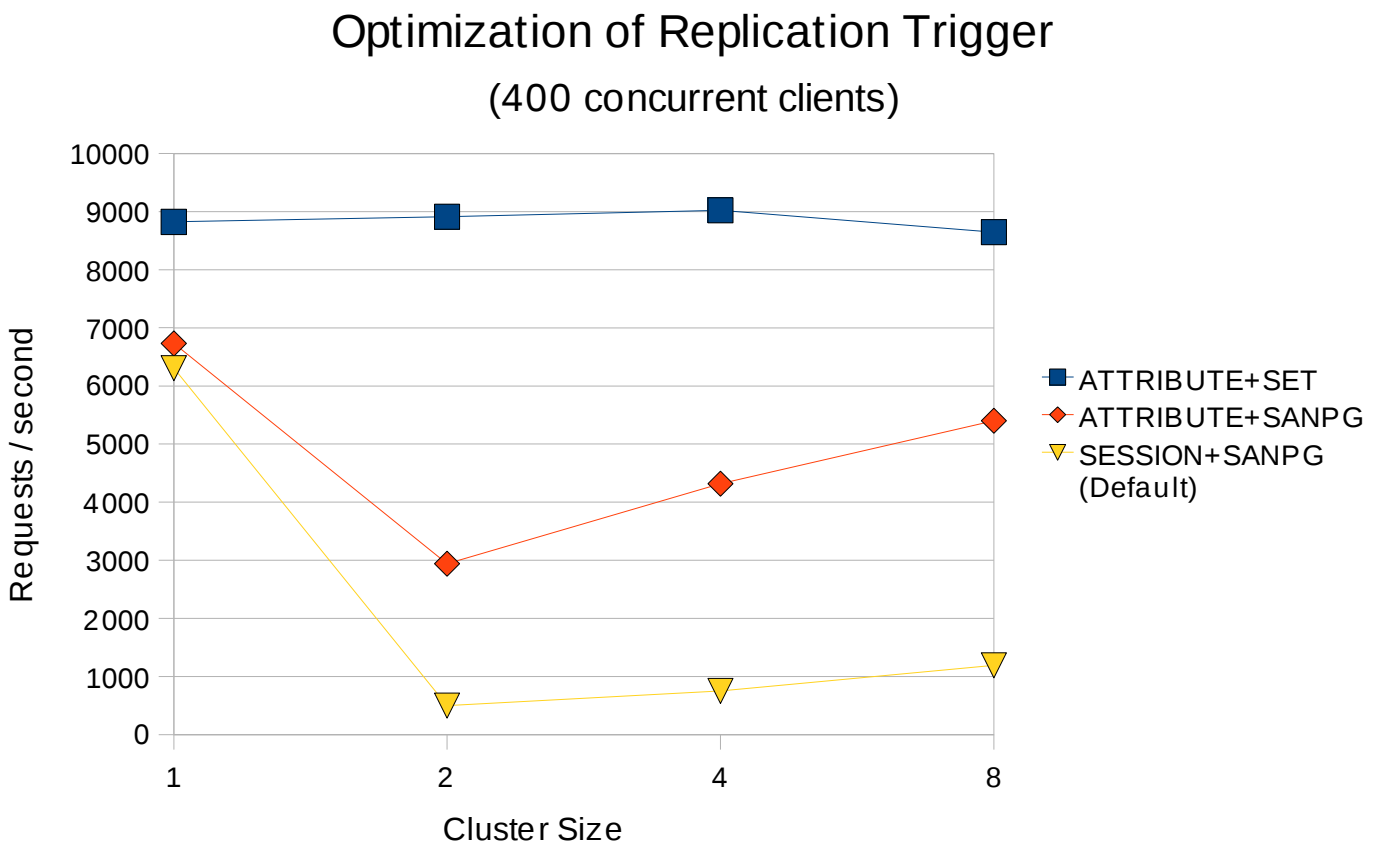
- Avoid unnecessary replication
- In test only 10% of requests modify the session
  - 90% just read an attribute of type byte[]
  - JBoss doesn't know if code modifies the byte[] after reading, so by default we replicate
    - 100% of requests replicate when only 10% need to!
  - Use replication-trigger SET to give you control

jboss-web.xml

```
<replication-config>
   <replication-trigger>SET</replication-trigger>
</replication-config>
```

Effect of Using SET

- Performance close to non-distributed sessions
  - In combination with ATTRIBUTE



Next Tip: Buddy Replication

- Replicate to N backups instead of all nodes

jboss-web-cluster.sar/META-INF/jboss-service.xml

```
<buddyReplicationEnabled>true</buddyReplicationEnabled>
<buddyLocatorProperties>
    numBuddies = 1
.....
```

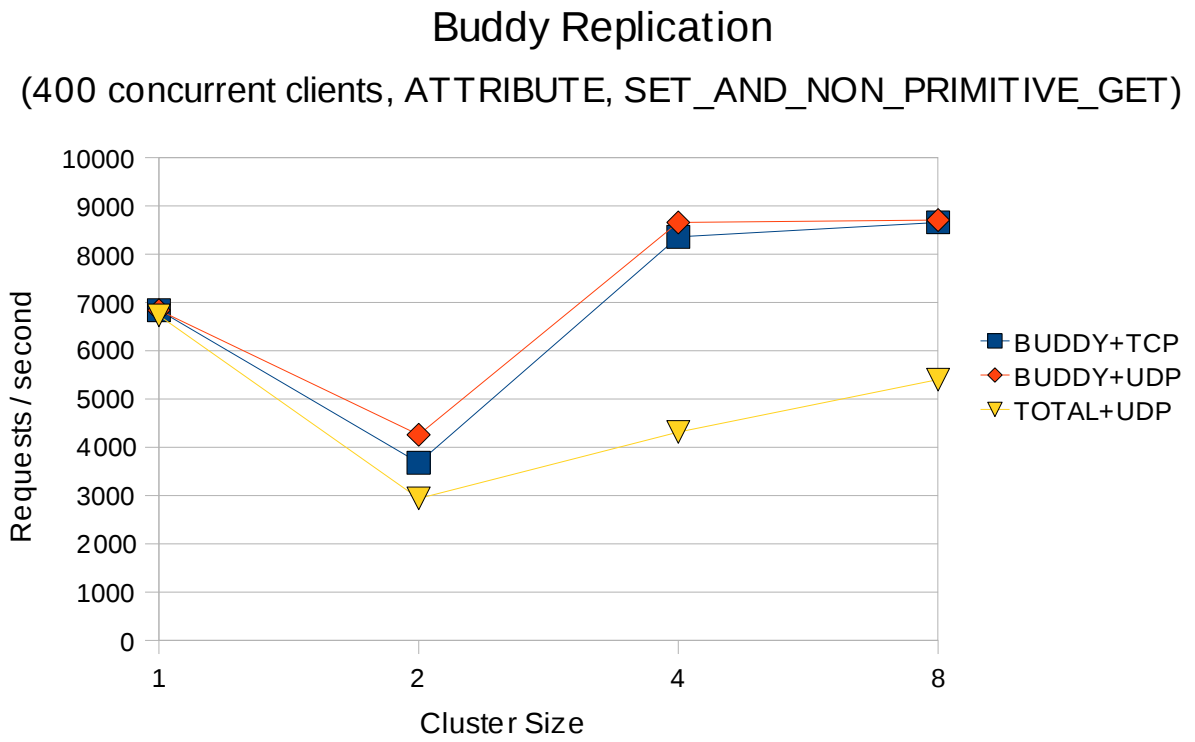
- Use of TCP for intra-cluster comm

jboss-web-cluster.sar/META-INF/jboss-service.xml

```
<attribute name="ClusterConfig">
  <config>
    <TCP start_port="7810" loopback="true"
      .....
```

Effect of Buddy Replication

- We didn't use SET here as we wanted to push more data, better to show effect of buddy repl
  - Each request replicates ~ 2.5KB



Caveat

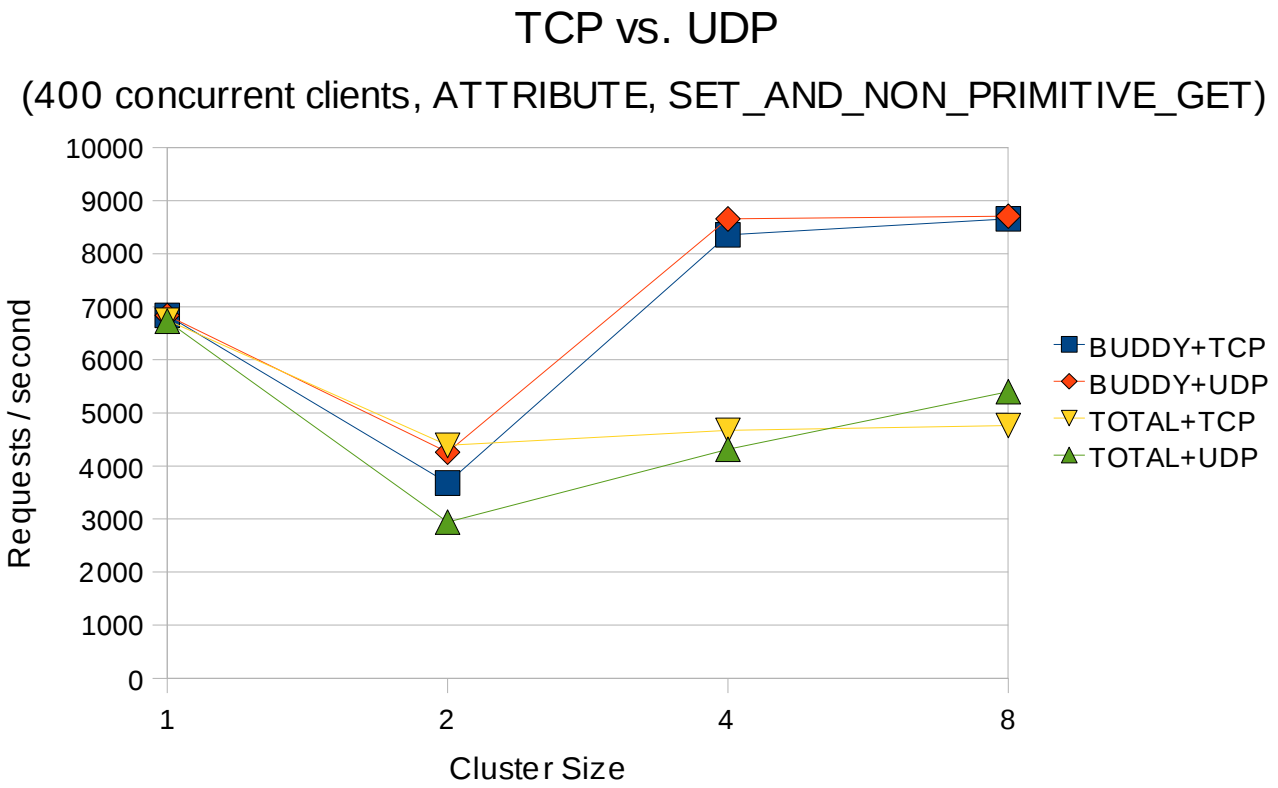
- With buddy replication, if a node fails or is shutdown, other nodes need to pick new buddy
  - Need to transmit all session state to new buddy
  - Failover requests need to pull session's state over from failed node's buddy
  - Adds stress to system already under stress
- Total replication doesn't need to do this
  - Everyone already has all state
  - KISS principle: if total replication meets your needs, it's simpler

Next Tip: TCP vs. UDP

- By default, intra-cluster comm channel uses UDP and multicast
  - Logical for total replication
    - Send one multicast message, all peers receive it
  - For buddy replication, UDP unicast is used
- You can configure the channel to use TCP
  - If group has N members, N – 1 TCP unicasts are sent with total replication
  - jboss-web-cluster.sar/META-INF/jboss-service.xml

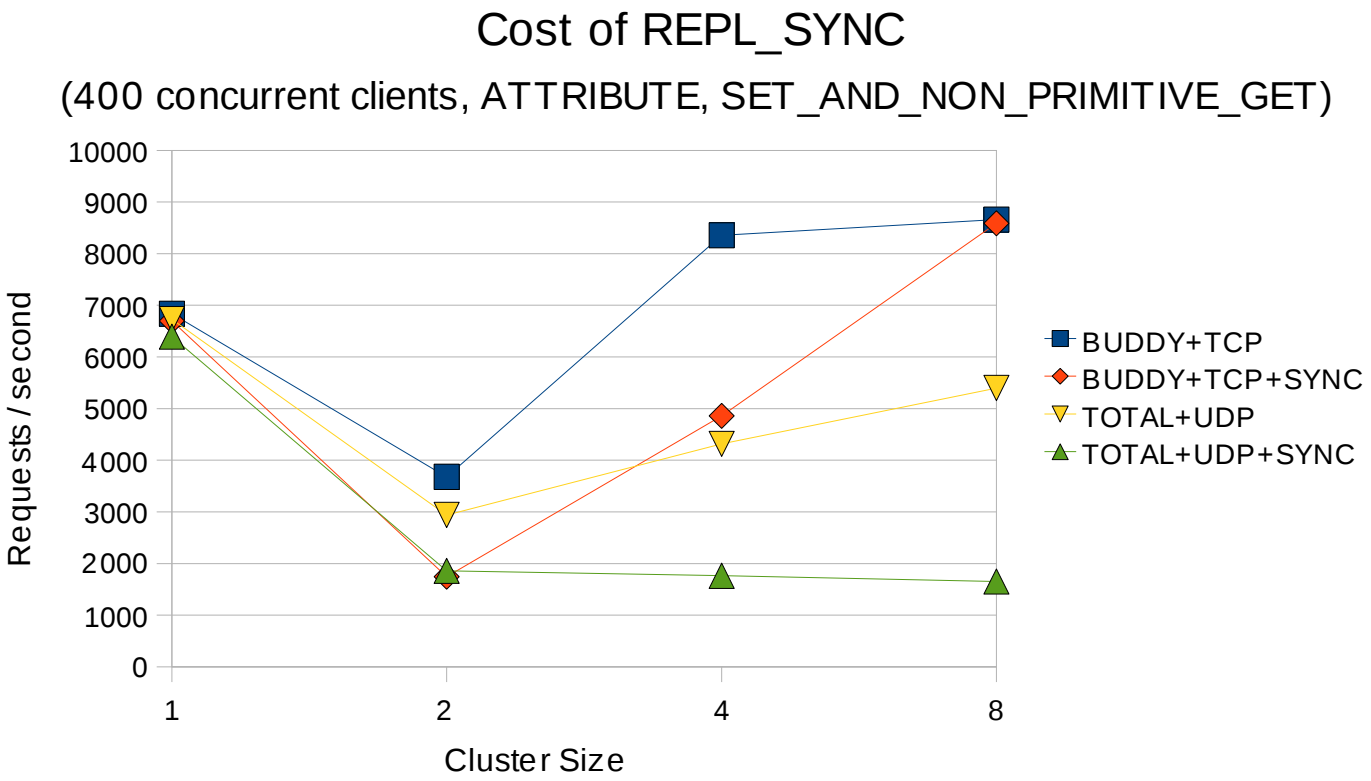
Effect of TCP vs. UDP

- TCP is a valid choice if number of peers is low
- UDP unicast performs well



Cost of REPL_SYNC

- With TOTAL, cost increases with cluster size



Additional Tips & Tricks

- Enable KeepAlive in httpd.conf!
- Connection pool sizes must be big enough to handle the max # of concurrent connections
  - Remember, nodes can fail, adding load to survivors
  - Defaul AJP Connector pool size is 40
    - Too small to run these tests

httpd.conf

```
<IfModule prefork.c>
StartServers          100
MinSpareServers        25
MaxSpareServers       100
ServerLimit           500
MaxClients            500
MaxRequestsPerChild  4000
</IfModule>
```

server.xml

```
<Connector port="8009"
   address="${jboss.bind.address}"
   protocol="AJP/1.3" emptySessionPath="true"
   enableLookups="false" redirectPort="8443"
   maxThreads=”500”
   connectionTimeout=”600000” />
```

Additional tips & tricks

- JBossWeb: use native APR lib (better scaling)
- Use EAPs if possible
  - Heavily tested and optimized
  - All subsystems are certified to work together
- JMX
  - listThreadCpuUtilization(), listThreadDump()
  - TomcatCluster MBean: look at contents of tree
- Unused HTTP sessions take up space
  - Set HTTP session timeout
  - Call invalidate() when done with a session

Logging

- Make sure logging is tuned!
  - JBoss by default logs at DEBUG
- Useful for development, debugging
- Turn it down (WARN) in production
  - Modify conf/jboss-log4j.xml
- JBoss logging can be changed at runtime
- Apache: access_log, modjk.log: might be big
  - error_log should be enabled

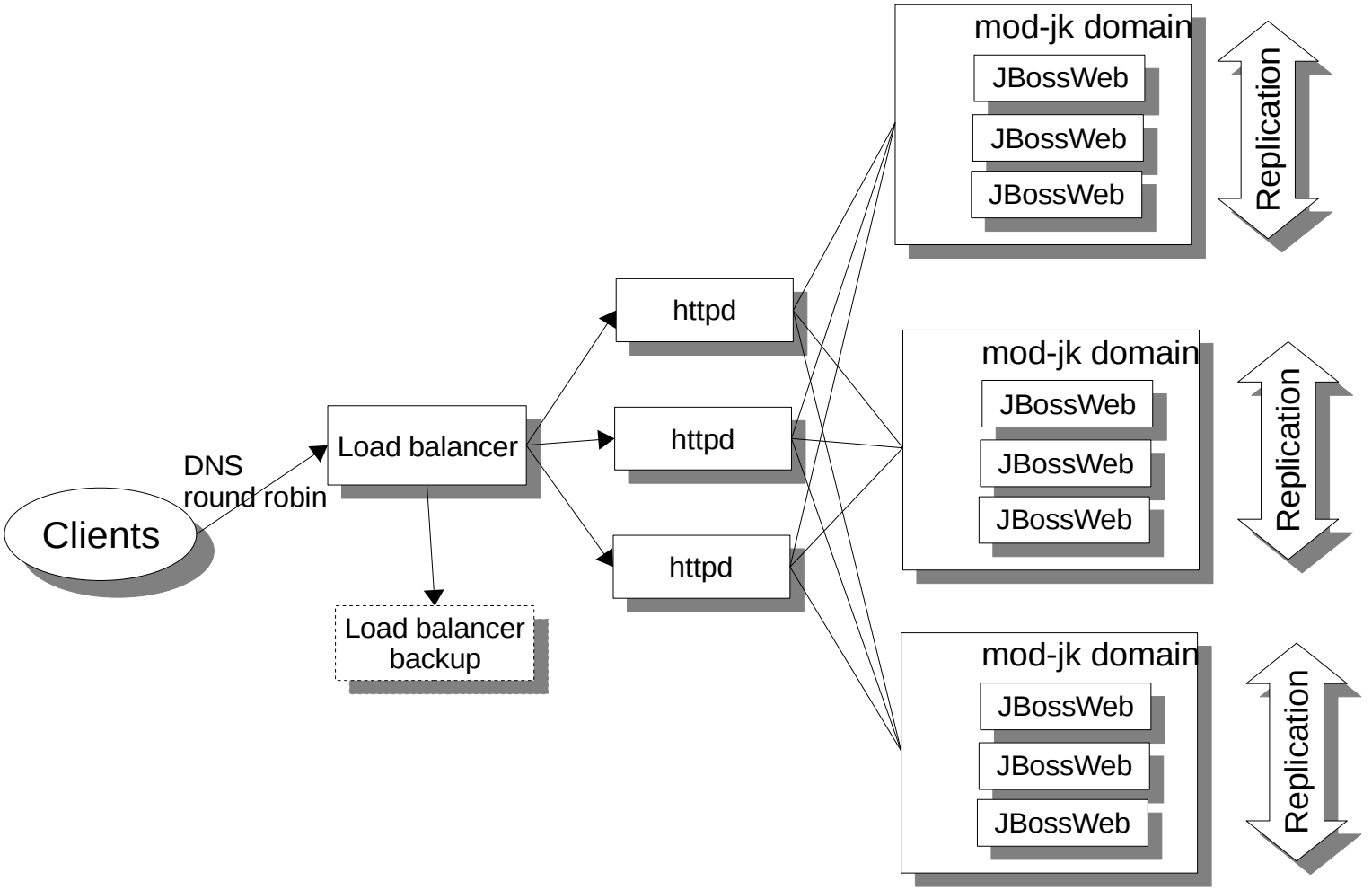
Losing some fat

- Remove unneeded stuff
- From 'all' to 'trimmed'
  - Starting: 1m07s 'all', 14s 'trimmed' (on a quad-core box !)
  - From 4 clusters to 1 cluster (session repl)
  - From 699 MBeans down to 203
  - From 106 threads down to 32

Tips & Tricks

- Separate networks for client, AJP and replication traffic, e.g.
  - Client requests come in through eth0 (switch-1)
  - AJP uses eth1 (switch-2)
  - Replication traffic uses eth2 (switch-3)
  - Otherwise client traffic and repl share bandwidth !
- Increase bandwidth or decrease sharing
  - Multiple httpds
  - Port trunking
  - Mod-jk domains

Mod-jk domains



Mod-jk domains

- Independent bouncing of domains
- Graceful draining of sessions
  - Apache 'status' application to gracefully take a domain down (drains HTTP sessions)
- With or without total replication
- Less contention
- Mod-jk: standby servers
  - Take part in replication, but no requests are sent
  - Can be activated on more load, are fully hot
  - Only works with total replication

Outlook: mod-cluster

- Dynamic discovery (no workers.props anymore)
- Session creation based on actual load
  - Load computation pluggable
    - CPU, number of HTTP sessions, total number of attrs over all sessions, bytes accessed and so on...
- JBossCache partitioning
  - Splitting huge sessions across cluster

Links and Q&A

- References
  - AS Clustering: http://labs.jboss.com/jbossclustering/
  - JBoss Cache: http://labs.jboss.com/jbosscache/
  - JGroups: http://labs.jboss.com/jgroups/
  - http://www.jboss.org/wiki/Wiki.jsp?page=UsingMod_jk1.2WithJBoss

- Questions?