Benchmarking and tuning JBoss EAP session clustering

Bela Ban
JBoss
06.29.2012
What's this talk about?

- Performance of a clustered webapp on EAP 6
  - Default configuration --> performance
  - Tuned configuration --> performance
- Along the way
  - How to configure an EAP 6 cluster
  - Get some tuning advice, best practices
Architecture of an EAP 6 cluster
Architecture of an EAP cluster

Infinispan

JGroups

HTTP session clustering

JBoss instance

Replication

JBoss instance

HTTP session clustering

Infinispan

JGroups
Let's take a look at the clustering modes: replication and distribution
Mode = replication

- All the data is stored on all cluster nodes
- Writes are sent to all nodes
  - Every node updates its local cache
- Reads are always local
- New nodes acquire the initial state from the oldest node
- Clients can access any node for reading or writing
- Scalability is limited by cluster size and data size
  - 10 nodes with 100MB state each: every node needs 1GB
Replication mode: write

Clients

mod-cluster

Replication

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>

SUMMIT JBoss WORLD
PRESENTED BY RED HAT
Replication mode: read

Clients

mod-cluster

Replication

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K1</th>
<th>Val1</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2</td>
<td>Val2</td>
</tr>
<tr>
<td>K3</td>
<td>Val3</td>
</tr>
<tr>
<td>K4</td>
<td>Val4</td>
</tr>
</tbody>
</table>
Mode = distribution

- Data is only stored on N cluster nodes (say N=2)
- A consistent hash on a key “id” determines the 2 servers for “id”
  - Example: cluster is \{A,B,C,D,E,F\}
  - Hash(“id”) = 8; 8 MOD 6 = 2
    - --> Primary owner = B, backup owner = C
  - Crash of B, new view is \{A,C,D,E,F\}
    - --> Primary owner = D, backup owner = E
    - --> C needs to transfer “id” to D and E and remove it locally
- Knowing the key, we always find the right server(s)
Mode = distribution

- A write updates the N owners of a key
- A read asks the primary owner
  - Possibly stores the result in its L1 cache
    - L1 caches require invalidation on updates
- Distribution uses less space than replication
- If all owner nodes of a given key crash at the same time, the data is lost
- The cost of shipping the modifications is constant
  - Good for large clusters
Distribution mode: write

write(K2)

Clients

mod-cluster

Replication
Distribution mode: read

Clients

mod-cluster

K1 | Val1
---|---
K2 | Val2
K4 | Val4
K5 | Val5

K1 | Val1
K2 | Val2
K4 | Val4
K5 | Val5

K2 | Val2
K3 | Val3
K5 | Val5

K3 | Val3

Replication
Mode=distribution and session clustering

- JBoss always stores a session on the current node
  - Requires session stickiness, or else we have a lot of remote reads (enable L1 in this case)!
- To do so, it grabs a key whose consistent hash maps to the current node
- That key acts as the jessionId
- Advantages
  - No remote reads
  - One of the writes is local
  - No need for an L1 cache (no invalidation)
Sync versus async

- Sync blocks the caller (Session.setAttribute()) until the changes have been applied
  - Confirmation using acks
- Async ships the changes in the background, the caller returns immediately
- Async is faster than sync
  - Recommendation: use sticky sessions
  - Sync always involves network round trips
- 4 combinations:
  - repl-sync, repl-async, dist-sync, dist-async
Configuration of an EAP cluster
MaxRequestsPerChild 0
ServerLimit 20
ThreadsPerChild 50
MaxClients 1000

<IfModule manager_module>
  Listen 8000
  <VirtualHost *:8000>
    <Location />
    Order deny,allow
    Allow from all
  </Location>
  EnableMCPMReceive
  <Location /mod_cluster_manager>
    SetHandler mod_cluster-manager
    Order deny,allow
    Allow from all
  </Location>
  </VirtualHost>
</IfModule>
EAP configuration (standalone-ha.xml)

- Address of httpd / mod-cluster:

  ```xml
  <subsystem xmlns="urn:jboss:domain:modcluster:1.1">
    <mod-cluster-config proxy-list="cluster09:8000" advertise="false"/>
  </subsystem>
  ```

- Cache configuration:

  ```xml
  <subsystem xmlns="urn:jboss:domain:infinispan:1.2">
    <cache-container name="web" default-cache="repl-sync">
      <transport stack="udp"/>
      <replicated-cache name="repl-async" mode="ASYNC"/>
      <replicated-cache name="repl-sync" mode="SYNC"/>
      <distributed-cache name="dist-async" owners="2" mode="ASYNC"/>
      <distributed-cache name="dist-sync" owners="2" mode="SYNC"/>
    </cache-container>
  </subsystem>
  ```
EAP configuration (standalone-ha.xml)

- Transport:

```xml
<subsystem xmlns="urn:jboss:domain:jgroups:1.1" default-stack="udp">
  <stack name="udp">
    <transport type="UDP" socket-binding="jgroups-udp"/>
    <protocol type="PING"/>
    <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
    <protocol type="pbcast.NAKACK"/>
    <protocol type="UNICAST2"/>
    <protocol type="pbcast.STABLE"/>
    <protocol type="pbcast.GMS"/>
    <protocol type="FRAG2"/>
  </stack>
  <stack name="tcp">
    <transport type="TCP" socket-binding="jgroups-tcp"/>
    <protocol type="MPING" socket-binding="jgroups-mping"/>
    ...
  </stack>
</subsystem>
```
Webapp configuration (web.war)

- WEB-INF/web.xml:

```
...<distributable/>
...```

- WEB-INF/jboss-web.xml

```
<replication-config>
  <cache-name>web.repl-async</cache-name>
</replication-config>`

Performance numbers
Test environment

- HP DL360 G7 dual socket, 4 core E5640 Nehalem, 2.66 GHz, 32GB RAM
- 1GB network
- JVM: 1.6.0_31
  - Options: -Xms256m -Xmx1g -server
- 8 physical boxes for EAP instances
  - When benching 12 and 16 instances, we run 2 EAP instances on some (12) or all (16) boxes
    - Competing for CPU / memory / network access
Perf test setup
Test driver

- Each 'client' is a thread in the (same) test driver
- 400 clients execute 100'000 HTTP requests
- 90% of the requests are reads, 10% are writes
- Each client does the following (num-reqs=100000):
  - Create a session (10 attributes, each has a 1K byte[] buffer)
  - while num-reqs > 0:
    - with a 90% chance: read a random attribute (1-10), or,
    - with a 10% chance: write a random attribute
    - decrement num-reqs
  - Destroy the session
Base line

Non-replicated

Nodes

Requests / sec

- Static: index.html
- Static: web/static.jsp
- Local (not replicated)
Comments

- Clustering adds some overhead, so we won't get higher than these numbers
- Ca. 30'000 requests / sec is what apache httpd can handle for a single test driver (400 threads)
- We could run multiple httpds and multiple test drivers to increase total throughput, but that's not what we want to measure... we want to measure the difference between local and clustered sessions
TCP versus UDP

TCP non-optimized

Requests / sec

Nodes

- Non-replicated
- repl-sync
- repl-async
- dist-sync
- dist-async
Comments

- With increasing cluster size, DIST outperforms REPL
  - On a modification, DIST changes the local cache and then sends a modification message to the backup owner
  - REPL has to send the modification to all cluster nodes
  - For 10K messages, we can get a theoretical max of roughly 12'500 reqs/sec (at the receivers)
- DIST_SYNC is not much slower than DIST-ASYNC
  - The network latency is low; it is apparently not the limiting factor
  - The limiting factor is client-httpd interaction and (replication) bandwidth
Comments

- With mode=REPL, TCP is faster than UDP initially, but perf degrades with increasing cluster size
  - In a 16 node cluster, a 'multicast' involves sending the same message to 15 nodes
  - TCP gets slower than UDP at 8-12 nodes
- With DIST, at 8+ nodes, TCP and UDP perform the same
Changing the replication granularity (jboss-web.xml)

```
<replication-config>
  <cache-name>web.repl-async</cache-name>
  <replication-granularity>ATTRIBUTE</replication-granularity>
</replication-config>
```

- SESSION is the default
  - A modification serializes the entire session
  - 10 attrs of 1K each --> 10K
- ATTRIBUTE only serializes the modified attribute
  - Serialized state --> 1K
Effect of changing the replication granularity

replication-granularity=ATTRIBUTE

Requests/sec

Nodes

Non-replicated
repl-sync
repl-async
dist-sync
dist-async
Comments

- REPL-ASYNC gets much faster, as it only has to replicate 1K instead of 10K
- REPL-SYNC is still bound by the latency; it needs to receive acks from all cluster nodes
Changing the replication trigger

```xml
<replication-config>
  <cache-name>web.repl-async</cache-name>
  <replication-granularity>ATTRIBUTE</replication-granularity>
  <replication-trigger>SET</replication-trigger>
</replication-config>
```

- SET_AND_NON_PRIMITIVE_GET is default
  - Since a byte[] buffer is non-primitive, even a read (getAttribute()) marks a session as dirty (triggering replication)
- SET only marks a session as dirty on a write (setAttribute())
- Now the 90% reads won't trigger replication!
Effect of changing the replication trigger

Replication-trigger=SET

Nodes

Requests / sec

- Non-replicated
- repl-sync
- repl-async
- dist-sync
- dist-async
- EAP 4.x (2008 test)
Comments

- Now we're only triggering replication in 10% of all requests (writes, size=1K)
  - This doesn't stress the clustering subsystem enough
- All reads are local
  - DIST pins the sessionId to the local node
  - REPL has all of the data, so the read is local as well
- Comparison to 2008 (EAP 4.x)
  - Almost comparing apples and oranges
  - Different hw/sw (same test though)
More tuning tips
Infinispan

- Use of repl queue for async replication seems to be counter-productive!
  - With: 18'000, without: 30'000 reqs/s (repl-async, 2 nodes)
  - Generates bursts rather than steady stream
  - Memory increase if generation rate > send rate
  - JGroups already performs batching
- Turn L1 cache off for sessions
  - The session is always in the local cache
  - Generates invalidation traffic
  - Memory overhead
Network

- Enabling 802.3 flow control can make a big difference with mode=REPL (in high traffic scenarios)
  - [https://community.jboss.org/wiki/PerfTuning](https://community.jboss.org/wiki/PerfTuning)
- Going from 1GB to 10GB makes a big diff as well
httpd

- **Do not** use Maxsessionid in httpd.conf!
- Install httpd in a local file system
  - NFS kills performance
    - Example: moving from NFS to /tmp increased local webapp access from 17'000 to 31'000 requests/sec
- Decrease logging verbosity
  - Remove or curtail access_log (can become big)
  - Log rotation is your friend
Conclusion
Distribution or replication?

- ATTRIBUTE and SET are best in most cases

- Use replication when
  - Cluster size is small and data is small
  - Data * cluster_size < allocated heap
  - Reads are fast, no rebalancing

- Use distribution when
  - The data is big or the cluster is large
  - Access cost is constant
  - Async when we have (session) stickiness
  - Sync when we don't
Links

- EAP 6: http://www.redhat.com/products/jbossenterprisemiddleware/application-platform
- mod-cluster: http://www.jboss.org/mod_cluster
- Infinispan: http://www.infinispan.org
- JGroups: http://www.jgroups.org
- Perf test: https://github.com/belaban/SessionPerf
LIKE US ON FACEBOOK
www.facebook.com/redhatinc

FOLLOW US ON TWITTER
www.twitter.com/redhatsummit

TWEET ABOUT IT
#redhat

READ THE BLOG
summitblog.redhat.com

GIVE US FEEDBACK
www.redhat.com/summit/survey