

0



**Table of ContenTs**

[**1 Recommendation Overview**](#_gjdgxs)

[**2 Document Conventions**](#_30j0zll)

[**2.1 Relative Locations**](#_1fob9te)

[**3 Scaling Recommendations**](#_3znysh7)

[**3.1 Operating System (Node) Optimizations**](#_2et92p0)

[3.1.1 Supported OS](#_tyjcwt)

[3.1.2 Patched OS](#_1t3h5sf)

[3.1.3 Tuning Configurations](#_2s8eyo1)

[3.1.4 Native (Library) Optimizations](#_35nkun2)

[**3.2 JVM Optimizations**](#_2jxsxqh)

[3.2.1 JVM Supportabliity](#_z337ya)

[3.2.2 Memory 6](#_1y810tw)

[3.2.3 Processors](#_2bn6wsx)

[3.2.4 Garbage Collection](#_1pxezwc)

[**3.3 AppServer Customization & Optimization**](#_23ckvvd)

[3.3.1 JBoss Support & Patching](#_ihv636)

[3.3.2 Profile Configuration](#_41mghml)

[3.3.3 Logging](#_3fwokq0)

[3.3.4 Container Resources](#_4k668n3)

[**4 High Availability Recommendations**](#_kgcv8k)

[**4.1 Load Balancers 4**](#_34g0dwd)

[4.1.1 HLB - Hardware Load Balancer](#_1jlao46)

[4.1.2 SLB - Software Load Balancer](#_43ky6rz)

[**4.2 Application Server Clustering**](#_1x0gk37)

[4.2.1 Verify JGroups clustering dependencies are installed, L2 Generally Recommend](#_4h042r0)

[4.2.2 Verify HTTP Session Clustering Configuration, L3 Strongly Recommend](#_2w5ecyt)

[4.2.3 Verify Distributable Servlet Development & Deployment, L2 Generally Recommend](#_1baon6m)

[4.2.4 Verify EJB Clustering, L1 Noted](#_3vac5uf)

[4.2.5 Verify JNDI Clustering, L1 Noted](#_2afmg28)

[4.2.6 Isolate JBoss Clustering networks, L3 Strongly Recommend](#_pkwqa1)

[**4.3 JBoss Messaging (HornetQ)**](#_39kk8xu)

[**5 DevOps Recommendations**](#_1opuj5n)

[**5.1 JBoss Rolling Deployments**](#_48pi1tg)

[**5.2 JBoss Classloaders and Classloader Isolation**](#_2nusc19)

[5.2.1 Verify deployment classloader isolation, L1 Noted](#_1302m92)

[5.2.2 JBoss Classloader - Modular](#_3mzq4wv)

[**5.3 Monitoring Webserver and AppServer Performance**](#_2250f4o)

[5.3.1 Metrics to Monitor](#_haapch)

[5.3.2 Monitoring Systems](#_319y80a)

[5.3.3 Apache Monitoring](#_40ew0vw)

[5.3.4 Tools for JVM Analysis](#_2fk6b3p)

[5.3.5 Security](#_184mhaj)

[**6 References**](#_meukdy)

[**7 Addendum: SLB Configuration**](#_36ei31r)

[**7.1 Configuration Generalizations**](#_1ljsd9k)

[7.1.1 Default MPM MaxClients formulas:](#_2koq656)

[7.1.2 Default Web Connector formulas:](#_zu0gcz)

[7.1.3 Explanation of Configuration Options](#_3jtnz0s)

[**7.2 Apache PreFork MPM**](#_1yyy98l)

[7.2.1 Validate extra capacity, L3 Strongly Recommend :](#_4iylrwe)

[7.2.2 Remove Worker MPM configuration, L2 Generally Recommend](#_2y3w247)

[**7.3 Apache mod\_jk**](#_1d96cc0)

[7.3.1 Verify URI Workmap, L2 Generally Recommend:](#_3x8tuzt)

[7.3.2 Alternative: Set JKMount for multiple Virtual Host paths](#_2ce457m)

[7.3.3 Verify mod\_jk workers.properties cachesize, L2 Generally Recommend:](#_rjefff)

[7.3.4 Add URI mapper mod\_jk uriworkermap.properties, L2 Generally Recommend](#_3bj1y38)

[7.3.5 Consider Alternative Load Balancer Connector (mod\_cluster), L2 Generally Recommend](#_1qoc8b1)

[7.3.6 HTTP Timeouts](#_4anzqyu)

[7.3.7 Verify Apache Timeout, L2 Generally Recommend](#_2pta16n)

[7.3.8 Maintain Apache Installation, L1 Noted](#_14ykbeg)

[7.3.9 Maintain SLB Compatibility, L1 Noted](#_3oy7u29)

[**7.4 JBoss**](#_243i4a2)

[7.4.1 JBoss Worker Node Setup](#_j8sehv)

[7.4.2 Application Sessions and Stickiness](#_338fx5o)

[7.4.3 JBoss EAP and Apache HTTPD Tuning](#_1idq7dh)

[7.4.4 JBoss Worker Threads](#_2hio093)

[7.4.5 Load Balancer Configuration Templates](#_wnyagw)

[**7.5 Ideal Settings & Conclusions**](#_3gnlt4p)

presented to CUSTOMER

by RH

* 1 [Recommendation Overview](#_gjdgxs)
* 2 [Document Conventions](#_30j0zll)
  + 2.1 [Relative Locations](#_1fob9te)
* 3 [Scaling Recommendations](#_3znysh7)
  + 3.1 [Operating System (Node) Optimizations](#_2et92p0)
    - 3.1.1 [Supported OS](#_tyjcwt)
      * 3.1.1.1 [Maintain AppServer & WebServer Nodes are running RHEL 6.6 x86\_64 on Rackspace, L1 Noted](#_3dy6vkm)
    - 3.1.2 [Patched OS](#_1t3h5sf)
      * 3.1.2.1 [Resolve RHN Registration, L3 Highly Recommend](#_4d34og8)
    - 3.1.3 [Tuning Configurations](#_2s8eyo1)
      * 3.1.3.1 [Ensure adequate OS userspace limits to the JBoss service acount. L3, Strongly Recommend](#_17dp8vu)
      * 3.1.3.2 [Tune network buffers for JBoss. L2, Generally Recommend](#_3rdcrjn)
      * 3.1.3.3 [Enable Large Page Memory. L1, Noted](#_26in1rg)
      * 3.1.3.4 [Set CGroup affinity, L1 Noted](#_lnxbz9)
    - 3.1.4 [Native (Library) Optimizations](#_35nkun2)
      * 3.1.4.1 [Install and enable native libraries (APR). L3, Strongly Recommend](#_1ksv4uv)
        + 3.1.4.1.1 [Installation and verification for RHEL 6 and JBoss EAP 6](#_44sinio)
  + 3.2 [JVM Optimizations](#_2jxsxqh)
    - 3.2.1 [JVM Supportabliity](#_z337ya)
      * 3.2.1.1 [Switch to Oracle Hotspot JVM (Java 7 or 8), L2 Generally Recommend](#_3j2qqm3)
    - 3.2.2 [Memory](#_1y810tw)
      * 3.2.2.1 [Set JVM/profile maximum heap size to minimum heap size, L1 Noted](#_4i7ojhp)
      * 3.2.2.2 [Memory Footprint Formula:](#_2xcytpi)
      * 3.2.2.3 [Rightsize Node's guest OS memory, L2 Generally Recommend](#_1ci93xb)
      * 3.2.2.4 [Memory Footprint Samples:](#_3whwml4)
    - 3.2.3 [Processors](#_2bn6wsx)
      * 3.2.3.1 [Rightsize Node's guest OS processing cores, L2 Generally Recommend](#_qsh70q)
      * 3.2.3.2 [Rightsize Threads available to GC, L2 Generally Recommend](#_3as4poj)
    - 3.2.4 [Garbage Collection](#_1pxezwc)
      * 3.2.4.1 [Enable GC logging, L3 Strongly Recommend](#_49x2ik5)
      * 3.2.4.2 [Set CMS GC for frontend app containers, L2 Generally Recommend](#_2p2csry)
      * 3.2.4.3 [Set Parallel GC for webservice app containers, L2 Generally Recommend](#_147n2zr)
      * 3.2.4.4 [Disable application GC management, L2 Generally Recommend](#_3o7alnk)
  + 3.3 [AppServer Customization & Optimization](#_23ckvvd)
    - 3.3.1 [JBoss Support & Patching](#_ihv636)
      * 3.3.1.1 [Right size JBoss EAP subscriptions from RedHat, L2 Generally Recommend](#_32hioqz)
      * 3.3.1.2 Update EAP to CP 01 or 02 (EAP v6.4.1+). L3, Strongly Recommend
    - 3.3.2 [Profile Configuration](#_41mghml)
      * 3.3.2.1  [Remove unused Server Group from domain.xml, L2 Generally Recommend:](#_2grqrue)
      * 3.3.2.2 [Remove unused profile definitions from domain.xml, L2 Generally Recommend:](#_vx1227)
    - 3.3.3 [Logging](#_3fwokq0)
      * 3.3.3.1.1 [Use global logging facility, L2 Generally Recommend](#_1v1yuxt)
    - 3.3.3.2 [Key JBoss Subsystem Categories](#_4f1mdlm)
      * 3.3.3.2.1 [Access Logger for HTTP requests (in EAP):](#_19c6y18)
    - 3.3.3.3 [Logfile Rotator](#_3tbugp1)
      * 3.3.3.3.1 [60413046\_Customer%28CUSTOMER%29JBossArchitectureReview-](#_28h4qwu)
      * 3.3.3.3.2 [Example Classloading Logger](#_nmf14n)
      * 3.3.3.3.3 [Application / Unmanaged Logging](#_37m2jsg)
      * 3.3.3.3.4 [Example Logger for Hibernate filter:](#_1mrcu09)
      * 3.3.3.3.5 [Remove explicit DEBUG or TRACE logs from applications (if loglevel check missing). L2 Generally Recommend](#_46r0co2)
      * 3.3.3.3.6 [Disable CONSOLE logging in PROD. L2 Generally Recommend](#_2lwamvv)
      * 3.3.3.3.7 [Disable DEBUG and TRACE level logs in PROD. L3 Strongly Recommend](#_111kx3o)
    - 3.3.3.4 [JPA/2LC Entity Cache](#_3l18frh)
      * 3.3.3.4.1 [Enable 2LC for Entity persistence, L2 Generally Recommend](#_206ipza)
  + 3.3.4 [Container Resources](#_4k668n3)
    - 3.3.4.1 [Web Container (HTTP/AJP)](#_2zbgiuw)
      * 3.3.4.1.1 [Verify HTTP/AJP maxThreads pool, L2 Generally Recommend](#_1egqt2p)
      * 3.3.4.1.2 [Web container thread pool formulas:](#_3ygebqi)
      * 3.3.4.1.3 [Verify reasonable limits on Web resources, L3 Strongly Recommend](#_2dlolyb)
    - 3.3.4.2 [Datasources (JCA)](#_sqyw64)
      * 3.3.4.2.1 [Stabilize JCA cached connection pool, L2 Generally Recommend](#_3cqmetx)
      * 3.3.4.2.2 [Verify Cached Connection Manager, L2 Generally Recommend](#_1rvwp1q)
      * 3.3.4.2.3 [Verify Oracle JDBC, L2 Generally Recommend](#_4bvk7pj)
      * 3.3.4.2.4 [Ensure supportability of persistence mapping (ORM), L1 Noted](#_2r0uhxc)
      * 3.3.4.2.5 [Optimize JDBC operations, L1 Noted](#_1664s55)
      * 3.3.4.2.6 [Remove test H2 JDBC, L3 Strongly Recommend](#_3q5sasy)
    - 3.3.4.3 [Miscellaneous Resources](#_25b2l0r)
* 4 [High Availability Recommendations](#_kgcv8k)
  + 4.1 [Load Balancers](#_34g0dwd)
    - 4.1.1 [HLB - Hardware Load Balancer](#_1jlao46)
    - 4.1.2 [SLB - Software Load Balancer](#_43ky6rz)
      * 4.1.2.1 [Optimized SLB with Apache mod\_jk tuning, L3 Strongly Recommend](#_2iq8gzs)
      * 4.1.2.2 [Reconfigure SLB with Apache mod\_cluster tuning, L2 Generally Recommend](#_xvir7l)
      * 4.1.2.3 [Maintain SLB on dedicated, DMZed WebServer Nodes. L3 Strongly Recommended](#_3hv69ve)
  + 4.2 [Application Server Clustering](#_1x0gk37)
    - 4.2.1 [Verify JGroups clustering dependencies are installed, L2 Generally Recommend](#_4h042r0)
    - 4.2.2 [Verify HTTP Session Clustering Configuration, L3 Strongly Recommend](#_2w5ecyt)
    - 4.2.3 [Verify Distributable Servlet Development & Deployment, L2 Generally Recommend](#_1baon6m)
    - 4.2.4 [Verify EJB Clustering, L1 Noted](#_3vac5uf)
    - 4.2.5 [Verify JNDI Clustering, L1 Noted](#_2afmg28)
    - 4.2.6 [Isolate JBoss Clustering networks, L3 Strongly Recommend](#_pkwqa1)
  + 4.3 [JBoss Messaging (HornetQ)](#_39kk8xu)
* 5 [DevOps Recommendations](#_1opuj5n)
  + 5.1 [JBoss Rolling Deployments](#_48pi1tg)
  + 5.2 [JBoss Classloaders and Classloader Isolation](#_2nusc19)
    - 5.2.1.1 [Verify deployment classloader isolation, L1 Noted](#_1302m92)
  + 5.2.2 [JBoss Classloader - Modular](#_3mzq4wv)
* 5.3 [Monitoring Webserver and AppServer Performance](#_2250f4o)
  + 5.3.1 [Metrics to Monitor](#_haapch)
  + 5.3.2 [Monitoring Systems](#_319y80a)
    - 5.3.2.1 [Implement monitoring solution, L3 Strongly Recommend](#_1gf8i83)
  + 5.3.3 [Apache Monitoring](#_40ew0vw)
  + 5.3.4 [Tools for JVM Analysis](#_2fk6b3p)
    - 5.3.4.1 [Heap Analysis Tools](#_upglbi)
    - 5.3.4.2 [Thread Analysis Tools](#_3ep43zb)
    - 5.3.4.3 [JBoss Snapshot/Point-in-Time Monitoring](#_1tuee74)
      * 5.3.4.3.1 [JConsole](#_4du1wux)
      * 5.3.4.3.2 [Verify environments for troubleshooting tools, L2 Generally Recommend](#_2szc72q)
  + 5.3.5 [Security](#_184mhaj)
    - 5.3.5.1 [Isolate Runtime Daemon, L3 Strongly Recommend](#_3s49zyc)
    - 5.3.5.2 [Enable JMX & Web management security, L3 Strongly Recommend](#_279ka65)
* 6 [References](#_meukdy)

1. **Recommendation Overview**

This document describes the recommendations for enhancing CUSTOMER's enterprise application server environments using RedHat JBoss Enteprise Application Platform (EAP). It includes technical changes to the CUSTOMER JBoss reference architecture.

RH recommends that CUSTOMER enhance the CUSTOMER JBoss reference architecture based on our recent review of the current JBoss EAP environments. Typical problems that we've noticed in JBoss environments generally include: excessive GC in JVMs, unexpected sticky or cookie affinity, exhaustion of load balancer or appserver resources, and insecure default configurations. Beyond typical problems, general performance and cost expectations are often not met because both vertical and horizontal scaling (i.e., scaling out and scaling up) are poorly implemented. Standard, fundamental pros and cons of both scaling models should be noted:

* horizontal scaling increases application fault tolerance but usually increases infrastructure costs
* vertical scaling maximizes infrastructure resources and investment but at the risk of resource exhaustion

Continued high availability while scaling infrastructure resources is dependent on simultaneously enhancing the fault tolerance and load distribution of application server clusters. The success of vertical scaling is dependent on optimizing the compute resources available to nodes that are dedicated to serving JBoss and its applications, while horizontal scaling is dependent upon dynamic load distribution of additional infrastructure resources. Optimizations we recommend include: tuning operating system (OS) and userspace, optimizing local OS resources for the application server, tuning Java Virtual Machines (JVM) including Garbage Collectors (GC), tuning application server containers including Web container and persistence container, and tuning Apache software load balancers (SLB). The success highly available and performant applications assumes these recommendations are tested in a production-like environment, reviewed, and implemented if meeting business and technical expectations..

1. **Document Conventions**

It is assumed that this document's audience is familiar with the current CUSTOMER JBoss environments, including terms used to describe those environments that will not be defined, including "Node", "frontend", "webservice backend", etc.

So technical and business teams can set priorities on recommendations we have labelled the changes described in this proposal with an expected level of impact for both scaling and availability goals. The impact scale ranges from low (L1, Note) to high (L3, Strongly Recommend), and each recommendation will be highlighted in this document, for example:

*Ensure adequate OS userspace limits to the JBoss service account. L3, Strongly Recommend*

We've attempted to elaborate on how to technically implement L2 and L3 recommendations while generally describing L1. Technical and business teams can determine the impact of these recommendations and decide whether to commit the change after functional and performance testing.

This document was updated early 2016 with details specific to the latest Red Hat supported release, JBoss EAP 6.4.x.

* 1. **Relative Locations**

Unless an absolute path is given, assume these relative paths:

* **Apache HTTPD**
  + Base Directory: /etc/httpd/
  + Conf Directory: /etc/httpd/conf.d
  + HTTPD Logging: /var/log/httpd/
* **JBoss EAP**
  + Base Directory: /usr/local/applmgr/
  + JBOSS\_HOME: /usr/local/applmgr/server/jboss
  + JBoss Domain Config: /usr/local/applmgr/server/jboss/domain/configuration
  + Logging:
    - JBoss Host/Master Logger Config: /usr/local/applmgr/server/jboss/domain/configuration/logging.properties
    - JBoss Server/Slave Logger Config: /usr/local/applmgr/server/jboss/domain/servers/server-\*/data/logging.properties
    - JBoss Console Logging: /usr/local/applmgr/logs/server
    - JBoss Host / Process Logs: /usr/local/applmgr/server/jboss/domain/log
    - JBoss Server Logging: /usr/local/applmgr/server/jboss/domain/servers/server-\*/log
    - CUSTOMER Application Logging: /usr/local/applmgr/logs/apps
* **Zabbix**

1. **Scaling Recommendations**
   1. **Operating System (Node) Optimizations**

Operating system optimization includes tuning resources of each Node in a JBoss environments to maximize resources available to JBoss EAP. These compute resources include: memory, filehandles, threads, network buffer/window, and native library interfaces. Since CUSTOMER JBoss nodes are dedicated as application servers, it is acceptable to tune system-wide parameters for server specific loads. Some OS optimizations are only available for RHEL so RH strongly recommends keeping the current Node OS.

### Supported OS

#### Maintain AppServer & WebServer Nodes are running RHEL 6.6 x86\_64 on Rackspace, L1 Noted

|  |
| --- |
| $ cat /etc/redhat-release  Red Hat Enterprise Linux Server release 6.6 (Santiago) |

### Patched OS

#### Resolve RHN Registration, L3 Highly Recommend

Apache HTTP Nodes are not registered to recieved Red Hat patches via RHN (Red Hat Network). From MOTD:

|  |
| --- |
| This machine has failed to register with RHN. Support  should attempt to register this machine manually. |

### Tuning Configurations

#### Ensure adequate OS userspace limits to the JBoss service acount. L3, Strongly Recommend

In RHEL, it may be best to set ulimits globally on a guest since these Nodes are dedicated to JBoss but either global or local is fine. Below are key userspace resources that should be configured but hard numbers are not given because their values must be determined by load testing. For example, the maximum number of threads used by all JVMs on a Node will be reflected in the total number of filehandles opened on the OS. It should be noted that unbounded/linear thread growth resulting in eventual OOM can be caused from a resource limit at the OS level (when the JVM running JBoss hits its nproc userspace limit).

Set limits.conf or limits.d/\* or PAM or jboss user profile for:

|  |
| --- |
| jboss hard nofile  jboss hard nproc  jboss hard memlock |

See RedHat's documentation for details on daemonized userspace limits, including changes to RHEL6: <https://access.redhat.com/site/solutions/341963>

#### Tune network buffers for JBoss. L2, Generally Recommend

All virtual NICs in RHEL should be reconfigured for optimal networking since the Nodes are dedicated to JBoss. To maximize the network buffers, reset the network TCP window to values that reduce fragmentation and queue size for typical JBoss network traffic. Your current kernel supports tuning these parameters by configuration.

|  |
| --- |
| echo 655360 > /proc/sys/net/core/wmem\_max echo 26214400 > /proc/sys/net/core/rmem\_max |

#### Enable Large Page Memory. L1, Noted

For heaps that get "big" (meaning >4GB generally) RedHat recommends large page memory optimization. This option is both an OS *and* JVM optimization. Since current JVMs do not have heap sizes greater than 4GB, this option should be placed in long term planning notes.

#### Set CGroup affinity, L1 Noted

Allow the RHEL OS to allocate compute resources to all running JVM/JBoss using its default scheduler. If resource contention is detected (by high processor load, aka. a cpu "hog", and swapping, aka. memory "hog"), set up non-exclusive minimum compute resource allocations to isolate processes. CGroups in RHEL with CPU and Memory ceilings set for JBoss runtime instances will isolate resource over-committment. RHEL Container Groups (CGroups) example setup for Oracle (where JBoss can be a drop-in replacement, ex: replace service account "oracle" with "jboss"): [cgroup\_resource\_mgmt\_v1.pdf](https://wiki-citytech.ctmsp.com/download/attachments/60413046/cgroup_resource_mgmt_v1.pdf?version=1&modificationDate=1438889413000&api=v2). See RHEL footnote: <https://access.redhat.com/site/solutions/231383>

### Native (Library) Optimizations

#### Install and enable native libraries (APR). L3, Strongly Recommend

The Native Components package is an optional component for JBoss EAP that incorporates native operating system components and connectors for web servers, including JBoss Native and LibAIO Native running on RHEL. Installing JBoss Native results in higher server performance as native and optimized operating system functionality becomes available to the JVM. These Native package includes:

* Apache Portable Runtime (APR), OpenSSL and Tomcat Native (TC-native);
  + Apache Portable Runtime (APR) provides superior scalability, performance, and improved integration with native server technologies. It enables access to advanced, non-blocking IO functionality (for example: sendfile, epoll and OpenSSL), Operating System level functionality (for example: random number generation and system status), and native process handling (shared memory and Unix sockets).
  + OpenSSL implements the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols and includes an enhanced cryptographic library. This is critical because of typical latency introduced by encrypted communication and algorithmic optimizations available to the OS.
  + Tomcat Native (TC-Native) is a Java Native Interface (JNI) that provides much of Tomcat's core functionality in native code rather than Java. This allows for an overall increase in the speed of a server.
* LibAIO is used as a bridge between JBoss and Linux I/O.

##### Installation and verification for RHEL 6 and JBoss EAP 6

Native libraries can be installed via RHN repository or after download. Note: ensure the correct library versions are used. CUSTOMER's Nodes should use the Linux 64-bit libraries.

See Red Hat's installation guide: <https://access.redhat.com/solutions/222023>, plus these additional guidelines:

1. Verify installation: After installing the Native components you can verify the components have been installed by looking for the following in your server.log

INFO [AprLifecycleListener] Loaded Apache Tomcat Native library <version>. INFO [AprLifecycleListener] APR capabilities: IPv6 [true], sendfile [true], accept filters [false], random [true].   
  
If component dependencies are missing, you might see the following log message:

DEBUG [org.apache.catalina.core.AprLifecycleListener] (main) The Apache Tomcat Native library which allows optimal performance in production environments was not found ... 129 more

*Tune APR, L3 Strongly Recommend*

APR will alter the default runtime parameters of JBoss so re-turning should be done. Specifically, the SLB recommendations for JBoss in the [CUSTOMER Addendum: SLB Configuration](#_36ei31r) should be revisited.

* 1. **JVM Optimizations**

JVM optimizations include tuning memory , multi-processor and multi-threaded use, and tuning garbage collection based on application profiling. JVM supportability, both for the application builds and the application server runtimes, are reviewed here.

### JVM Supportabliity

JBoss Nodes currently run Java 8 in OpenJDK JVMs.

|  |
| --- |
| $ java -version openjdk version "1.8.0\_45" OpenJDK Runtime Environment (build 1.8.0\_45-b13) OpenJDK 64-Bit Server VM (build 25.45-b02, mixed mode) |

Red Hat does not support Java 8 OpenJDK for the current EAP runtime. Red Hat supports Oracle Hotspot JVMs for Java 8 (and Java 7) for the current EAP runtime. See Red Hat's [Supported Configuration Matrices](https://access.redhat.com/articles/111663).

#### Switch to Oracle Hotspot JVM (Java 7 or 8), L2 Generally Recommend

Oracle's JVM supports a variety of reliable tuning options so RH recommends switching the Oracle Hotspot JVM and setting basic optimizations of JVM memory, processors, and garbage collectors. There are intuitive algorithms (introduced as supplements to ergonomics) since Java 6 that can dynamically optimize the runtime based on predefined goals – such as inputting a SLA based on throughout of application time, say max 1% time in GC. For either OpenJDK or Oracle Hotspot, additional testing is required to verify any tuning.

### Memory

#### Set JVM/profile maximum heap size to minimum heap size, L1 Noted

CUSTOMER has set the JVM parameters for the min/max memory size to the same values. Ensure this is *standardized across all environments* to minimize the time re-sizing the heaps during GC events. In XML for each profile or globally via domain.conf:

|  |
| --- |
| java -D[Process Controller] -server -Xms2048m -Xmx2048m -XX:MaxPermSize=1024m |

64-bit JRE in Server mode does appear global.

Note: There used to be debate about slow performance on 64bit and slimming the app into smaller deployables under 32-bit mode but there's consistent tests that show there’s no great benefit from running 32-bit with today's JVMs. This is especially true for Oracle's Hotspot since update 23 enables 32-bit optimizations by default (+CompressedOops).

Any deployments of the Oracle JVM will default to the recommended Server Mode on 64-bit OSes.

#### Memory Footprint Formula:

#### Rightsize Node's guest OS memory, L2 Generally Recommend

It is critical that memory not be overcomitted within the guest OS hosting JBoss. *Total heap allocation for all JVMs on a single Node should not exceed real available memory in the RHEL guest*. **In other words, swapping must be avoided**; because virtual disk latency will significantly impact GC times and could introduce problems with JVM resiliency (such as the JVM going OOM because the GC response time exceeds ergonomic expectations).

To rightsize memory allocation for Nodes, use the following formula:

JVM Xmx + RHEL OS reserved memory < RHEL Guest OS Memory Allocation

#### Memory Footprint Samples:

The JBoss EAP 6 memory footprint is important to consider in vertical scaling because it consumes the same memory resource as applications. JBoss consumes a relatively small amount. Usually the largest consumers of memory are application dependencies (such as Spring) and applications themselves but additional profiling would be needed to sample exact sizes. The current maximum heap sizes (Xmx) sampled indicate the current Nodes for CUSTOMER JBoss are at capacity. CUSTOMER JBoss Node guests are allocated 11GiB from Rackspace Hypervisors, from which memory usage is:

|  |  |
| --- | --- |
| Runtime | Xmx |
| JBoss Host Controller | 2048m |
| JBoss Process Controller | 2048m |
| JBoss Domain Server 1 | 2048m |
| JBoss Domain Server 2 | 2048m |

Average heap per JVM = 2 GiB

Using the memory footprint formula above, both Node1 and Node2 samples leave 2GiB for RHEL:

8GiB + N < 8 GiB, where N ~2GiB

Assuming these memory footprints accurately provide resources for application load, and assuming future applications will require, on average, the same amount of heap sizing, then vertical scaling of will require additional guest OS memory allocation. For example, a "Big Box" Node with 8 will require the following memory allocations:

average JVM heap size \* number of + RHEL reserved memory = guest OS memory allocation

1.5 GiB \* 8 + 2GiB = 14GiB "Big Box" Node

### Processors

#### Rightsize Node's guest OS processing cores, L2 Generally Recommend

Typically, the largest consumers of processing resources for Java environments are: 1) application "think time", 2) encryption/decryption, and 3) GC events. Case #3 and #2 are discussed in depth elsewhere (See GC tuning below and OpenSSL native optimization above) and case #1 is determined by application specific load testing.

vCPU overcommitment within the guest OS is acceptable in a virtual environment like CUSTOMER JBoss after release certification testing. These tests will determine whether a dispersed, constant, or spikey load requires rightsizing processing resources. The RHEL process scheduler will allocated cores and threads to each JVM/profile based on demand. Oracle HotSpot ergonomics will maximize GC threads based on total vCPUs allocated to the guest OS by default so tuning these parameters (like GCThreads, etc.) should not be necessary.

When switching GC algorithms, it’s well known that some require more processing but there are few hard recommendations for the quantity of processing power. For example, Oracle suggests a minimum of 2 CPUs per JVM if CMS is enabled but they backtrack and say 1 CPU is fine if another option is passed to CMS (Incremental options, see below). More than 2 CPUs are recommended for ParallelGC to maximize multithreaded GC with that algorithm but with the advent of hyperthreaded cores this is also a soft CPU recommendation. See the Garbage Collection section, set a baseline, and adjust core allocations based on results.

*Cores Available: Web Node = 4*

*Cores Available: JBoss Node = 2*

|  |
| --- |
| cat /proc/cpuinfo | egrep 'processor|cores' processor : 0 cpu cores : 1 processor : 1 cpu cores : 1 processor : 2 cpu cores : 1 processor : 3 cpu cores : 1 |

#### Rightsize Threads available to GC, L2 Generally Recommend

Most JVMs will maximize multiprocessor systems ability to perform symmetric multithreading. The formula for threads allocated to the Parallel GC is:

*N* processors = *N* GC threads

In the case of CUSTOMER, this means that each JVM (with ParallelGC) could request 4 threads from the JBoss Node during a simultaneous GC events:

2 processors per JVM = 2 GC threads

Without resource isolation (such as RHEL CGroups), each JVM can consume all compute resources on a Node, so the theoretical GC threads that could simultaneously is:

2 processors per JVM \* 2 JVMs = 4 GC threads

Vertical scaling will compound compute resource requests (by a factor of 2), so any additional scaling should only be done after thread and heap tuning to minimize JVM "stop-world" events.

### Garbage Collection

#### Enable GC logging, L3 Strongly Recommend

GC logging is vital to analyzing JVM internals in regards heap usage, collection intervals, etc. In each profile's XML or in JVM runtime, include:

|  |
| --- |
| -verbose:gc -Xloggc:gc.log.`date +%Y%m%d%H%M%S` -XX:+PrintGCDetails -XX:+PrintGCTimeStamps |

This configuration can be setup centrally via each JBoss Host, in host.xml:

|  |
| --- |
| <jvm name="server-two-jvm" debug-enabled="false">  <jvm-options>  <option value="-verbose:gc"/>  <option value="-Xloggc:/var/log/jboss/server-two-gc.log"/>  <option value="-XX:+PrintGCDetails"/>  <option value="-XX:+PrintGCTimeStamps"/>  </jvm-options>  </jvm> |

Note: the loggc should be relative to the PWD (present working directory) for the JBoss runtime. This defaults to:

-jboss-home /usr/local/applmgr/server/jboss

#### Set CMS GC for frontend app containers, L2 Generally Recommend

#### Set Parallel GC for webservice app containers, L2 Generally Recommend

RH usually recommends CMS switched on any frontend/UI and Parallel for any backend/webservice. Red Hat still argues for the ParallelOld GC in some JBoss installations based on loads. RH's internal testing on JBoss products suggests CMS for concurrency, so low latency, and Parallel/Old for parallelism, so maximizing processing power. Some parts of Oracle's CMS are not multi-threaded so more processing power will not always help. We’re recommending CUSTOMER keep JBoss standard, rather than exceptions, so the frontend/UI apps will benefit most from CMS and the backend from Parallel.

For each JVM, the following test iterations are in sequence:

1. Basline (see Memory recommendation above):

-Xmx<heapsize> -Xms<heapsize> -XX:PermSize=1024M -verbose:gc -Xloggc:gc.log.`date +%Y%m%d%H%M%S` -XX:+PrintGCDetails -XX:+PrintGCTimeStamps

1. **frontend/UI :**
   1. Round 1: Concurrent Test

-XX:+UseConcMarkSweepGC

* 1. Round 2: Concurrent Incremental Test

-XX:+UseConcMarkSweepGC -XX:CMSIncrementalMode

1. **backend/webservice :**
   1. Round 1: Parallel Test

-XX:UseParallelGC

* 1. Round 2: Parallel Major&Minor Test

-XX:+UseParallelGC -XX:UseParallelOldGC

...

PermSize and MaxPermSize are not options suggested by Red Hat. Oracle also discourages fixing the size of individual generations or setting them via remote calls (RMI) based on fixed time internals.

There's typically suspicion of PermGen space. Frankly, PermGen should (ideally) stabilize and have little impact on GCs after all modules are loaded by JBoss and all applications deploy to containers. If there’s a lot of churn in PermGen or GC logs reveal gradual growth in this space, then there’s a bigger problem to trace out. Of course, if GC intervals or durations cause timeouts, then RMI GC frequency and MaxPermSize can be added back and retested.

#### Disable application GC management, L2 Generally Recommend

For "inexplicable" GC events that happen even after telling the JVM when to do a full GC, there's the Achilles heel of System.gc. Without scraping all source code, there may be unexpected or legacy applications making this call. A simple code check could verify whether it's not just the JVM itself that is controlling GCs, or you can just trump the applications.

In each profile's XML or JVM runtime:

|  |
| --- |
| -XX:+DisableExplicitGC |

* 1. **AppServer Customization & Optimization**

JBoss optimization and tuning changes include: profile slimming, log tuning, cache tuning (including JPA container/Hibernate), Web container thread pools, bean container tuning, and JCA container (DS) connection pools. The recommendation starts with updating JBoss binaries.

### JBoss Support & Patching

#### Right size JBoss EAP subscriptions from RedHat, L2 Generally Recommend

If vCPUs are increased in a CUSTOMER JBoss environment, then ensure that there are enough remaining cores in the purchased RedHat subscription. Total vCPU from Rackspace in all CUSTOMER JBoss environments should be <= the total core band available from RedHat. See RHN recommendation above.

#### Update EAP to CP 01 or 02 (EAP v6.4.1+). L3, Strongly Recommend

Cumulative Patches (CP) are a collection of minor version updates that include both functionality patches and security errata that Red Hat strongly recommends for JBoss EAP. CUSTOMER JBoss currently runs on CP 6.4.0. Red Hat does not support EAP running this version within an OpenJDK v1.8 JVM. See Red Hat's [Supported Configuration Matrices](https://access.redhat.com/articles/111663). From server.log:

|  |
| --- |
| server.log:02:46:35,097 INFO [org.jboss.as] (MSC service thread 1-1) JBAS015899: JBoss EAP 6.4.0.GA (AS 7.5.0.Final-redhat-21) starting |

Note: RH generally recommends switching to Oracle Hotspot JVM instead of using OpenJDK.

Full CP details:

For steps and example upgrade, see:

### Profile Configuration

The current in CUSTOMER JBoss environments are based off of the out-of-box, template shipped with JBoss*.* Basic application functionality, including call-by-value and deployment isolation, are enabled in the default.

The environment/realm should be based off of the high availability "ha" profile template shipped with JBoss. We *strongly recommend verifying all dependency and configuration requirements for a JBoss cluster*. See the Clustering recommendations below.

#### Remove unused Server Group from domain.xml, L2 Generally Recommend:

|  |
| --- |
| <server-group name="other-server-group" profile="full-ha">  <jvm name="default">  <heap size="1000m" max-size="1000m"/>  <permgen max-size="256m"/>  </jvm>  <socket-binding-group ref="full-ha-sockets"/>  </server-group> |

#### Remove unused profile definitions from domain.xml, L2 Generally Recommend:

|  |
| --- |
| $ grep profile domain.xml  <profiles>  <profile name="default">  </profile>  <profile name="ha">  </profile>  <profile name="full">  </profile>  <profile name="full-ha">  </profile>  </profiles> |

### Logging

#### Use global logging facility, L2 Generally Recommend

A single, global logging facility configuration is easiest to maintain. Red Hat has a guide for JBoss/JVM logging recommendations: <https://access.redhat.com/solutions/18178>

#### Key JBoss Subsystem Categories

Some of the key subsystem category names are given in the following table. These are just the top level category names. Generally you can specify much more specific category names to enable very targeted logging.

**Table 10.1. JBoss SubSystem Categories**

|  |  |
| --- | --- |
| SubSystem | Category |
| Cache | org.jboss.cache |
| CMP | org.jboss.ejb.plugins.cmp |
| Core Service | org.jboss.system |
| Cluster | org.jboss.ha |
| EJB | org.jboss.ejb |
| JCA | org.jboss.resource |
| JMX | org.jboss.mx |
| JMS | org.jboss.mq |
| JTA | org.jboss.tm |
| MDB | org.jboss.ejb.plugins.jms, org.jboss.jms |
| Security | org.jboss.security |
| Tomcat | org.jboss.web, org.apache.catalina |
| Apache | org.apache |
| JGroups | org.jgroups |

##### Access Logger for HTTP requests (in EAP):

Note the pattern formater in the example below. It will include both timestamp and thread details. Basically you’re extending the HTTP component of EAP out-of-the-box for the web host alias and telling EAP to output to the directory below your current JBoss log dir.

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:web:1.1" … >  <connector name="http" …>  <virtual-server name="default-host" enable-welcome-root="true">  <alias name="localhost"/>  <access-log pattern="%t %a %H %p %U %s %S %I %D” rotate="true">  <directory path=“./access-logs/" relative-to="jboss.server.log.dir"/>  </access-log>  </virtual-server> .. |

The official document about this is <https://access.redhat.com/site/solutions/16101> (although I couldn’t find the official list of all the log pattern modifiers — the %I etc. -- but there’s several sources of them out on the Net)

#### Logfile Rotator

To create a JBoss managed rotating logfile, extend the domain.xml configuration like this:

|  |
| --- |
| <periodic-rotating-file-handler name="FILE"> … <file relative-to="jboss.server.log.dir" path="server.log"/> |

To change the target destination of logfiles:

|  |
| --- |
| <periodic-rotating-file-handler name="FILE" autoflush="true">  <formatter>  <named-formatter name="PATTERN"/>  </formatter>  <file path="/var/log/jboss/${jboss.server.name}.log"/> |

##### 

##### Example Classloading Logger

To add additional loggers to JBoss, both an <appender> and a <category> should be setup in the central configuration. This can be used to make application specific loggers based on Java naming space (.com.Customer.\*). To be able to trace back deadlock/stuck threads, change the pattern formatter of the server log to include thread details. The following format will print the thread name/id in the server logs. For any ConversionPattern in log4j file, change from any [%c] to [%t %c]

|  |
| --- |
| <appender name="THREADING\_LOG" class="org.jboss.logging.appender.DailyRollingFileAppender">  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>  <param name="File" value="${jboss.server.log.dir}/threading.log"/>  <param name="Append" value="true"/>  <param name="Threshold" value="WARN"/>   <!-- Rollover at the top of each hour -->  <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>   <layout class="org.apache.log4j.PatternLayout">  <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message\n -->  <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>  <!-- explicit pattern   <pattern-formatter pattern="%d{dd MMM yyyy HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/> -->  </layout> </appender> |

Or standard catch all for threads:

|  |
| --- |
| <!-- Enable threading logging --> <category name="java.lang.Thread">  <priority value="WARN"/>  <appender-ref ref="THREADING\_LOG"/> </category> |

##### Application / Unmanaged Logging

If not all logging can be configured centrally and globally, then you can include customized logging at the application level. RH noticed that CUSTOMER is deployable log4j enabled applications. RH recommends excluding log4J libraries in your application's WAR since these may conflict with the JBoss library and introduce classloader isolation issues (just make sure library versions are compatible with functionality). To exclude the library in a WAR yet expose the customized logging for the application, create a classloader exclusion. Note that you cannot modify your configuration during runtime so a restart is needed. Application level logging facilities are configured within the deployable itself. Apache Log4j, for example, is reconfigured via log4j.xml or log4j.properties in your deployment.

Here's an example of application deployment XML:

|  |
| --- |
| <jboss-deployment-structure>  <deployment>  <exclusions>  <module name="org.apache.log4j" />  </exclusions>  </deployment>  </jboss-deployment-structure> |

##### Example Logger for Hibernate filter:

|  |
| --- |
| <logger category="org.hibernate">  <handlers>  <handler name="HIBERNATE\_LOG"/>  </handlers> </logger> <periodic-rotating-file-handler name="HIBERNATE\_LOG">  <formatter>  <pattern-formatter pattern="%d{dd MMM yyyy HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>  </formatter>  <file relative-to="jboss.server.log.dir" path="hibernate.log"/>  <suffix value=".yyyy-MM-dd"/>  <append value="true"/> </periodic-rotating-file-handler> |

##### Remove explicit DEBUG or TRACE logs from applications (if loglevel check missing). L2 Generally Recommend

Scrape code for explicit logs that are written at high priority levels that are production unfriendly for performance. Sample code indicates that this is already being done:

|  |
| --- |
| if (logger.isDebugEnabled())  {  logger.debug("queuing report with client id: " + request.getClientId() + "; app id: " + request.getAppId() + "; requestor id: " + request.getRequestorId()); |

##### Disable CONSOLE logging in PROD. L2 Generally Recommend

Remove the CONSOLE appender block, <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">, from PROD deployments as it typically is too generic and consumes resources.

##### Disable DEBUG and TRACE level logs in PROD. L3 Strongly Recommend

Filter the level of loggers with the THRESHOLD option in the Appender that cause excessive logging, such as DEBUG and TRACE, by downgrading the loglevels. Ex: to only include WARN and above: <param name="Threshold" value="WARN"/>

#### JPA/2LC Entity Cache

##### Enable 2LC for Entity persistence, L2 Generally Recommend

Red Hat generally recommends enabling out-of-the-box Second Level Cache (2LC) for all entities. In EAP v6 these cache implementations have been standardized by Hibernate and Infinispan. Enabling and developing a 2LC is out of the scope of this document however general guides are available since RH noted that CUSTOMER is deploying Oracle and EHCache enabled applications. Generally speaking, entity caches like 2LC would not be clustered (local only), be of type invalidation, and use an LIRS eviction strategy.

In any applications with persistence configuration, such as META-INF/persistence.xml, can include a JPA compliant datasource that will be reference by the application via JNDI:

|  |
| --- |
| <persistence-unit name="jpaDS">  <jta-data-source>java:jboss/datasources/jpaDS</jta-datasource>  <shared-cache-mode>DISABLE\_SELECTIVE</shared-cache-mode>  <properties>  <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>  <property name="hibernate.hbm2ddl.auto" value="update" />  <property name="hibernate.cache.use\_second\_level\_cache" value="true"/>  </properties> </persistence-unit> |

This persistence configuration means that Infinispan will be enabled by default as a 2LC for Hibernate enabled entities, via JPA compliant beans, as implied by the annotation: @Cacheable

### Container Resources

#### Web Container (HTTP/AJP)

Web container resources will be consumed by 2 critical users: 1) web clients, and 2) web services. Web clients will consume HTTP resources inherited by AJP on the front layer. Web services will consume HTTP resources directly on the webservice/backend. Load tests of concurrent web users of both types should reveal the ideal parameter of Web container resources. The web services user, however, is asynchronous by definition and its requests can be queued by the container, therefore the queue must be long enough for the total concurrent requests. It is strongly recommended to set max-connections / maxThreads in Web containers to the maximum concurrent sessions sustainable in production loads to prevent failed client connections.

##### Verify HTTP/AJP maxThreads pool, L2 Generally Recommend

##### Web container thread pool formulas:

The default thread pool size formula is:

512 \* Processors

And the ideal thread pool sizing formula:

maximum threads = maximum, keep-alive sessions

In the case of CUSTOMER JBoss Nodes, this means the current default max-connections / maxThreads pool size for the Web containers are:

512 \* 4 vCPUs = 2048 threads

If maximum sessions under production load are greater than this default size, then the max-connections / maxThreads can be increased in Connector settings. See the [CUSTOMER Addendum: SLB Configuration](#_36ei31r) for more details.

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:web:2.2" default-virtual-server="default-host" native="false">  <connector name="ajp" protocol="AJP/1.3" scheme="http" socket-binding="ajp" max-connections="1024" ></connector>  <virtual-server name="default-host" enable-welcome-root="false"> |

##### Verify reasonable limits on Web resources, L3 Strongly Recommend

Once the maxThreads limit is reached in the Web container, no further HTTP connections are accepted. This was done to prevent a Denial Of Service (DOS) attack from web clients and services. The general idea is to serve the keepAlive connections and reject new ones once the limit is reached. The maxThreads value should be reasonably aligned with normal, maximum session loads. See the SLB Addenum section for relevant details.

See the Monitoring section below for verifying the current session counts.

#### Datasources (JCA)

##### Stabilize JCA cached connection pool, L2 Generally Recommend

The CachedConnectionManager can be setup for JCA with the ability optimize database connections. Increase your connection pool size for datasources with high volume concurrency. In each \*-ds.xml or datasource configuration, add:

<min-pool-size>10</min-pool-size><max-pool-size>10</max-pool-size> <blocking-timeout-millis>15000</blocking-timeout-millis>

Note: this blocks only while waiting for a permit for a connection, and will never throw an exception if creating a new connection takes an inordinately long time. The default is 30000 (30 seconds).

##### Verify Cached Connection Manager, L2 Generally Recommend

All managed connections should by verified or validated before pooled. This is accomplished in the domain configuration datasource subsystem. Following example is for Oracle datasources using JDBC:

|  |
| --- |
| <validation>  <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>  <check-valid-connection-sql>SELECT 1 FROM dual</check-valid-connection-sql>  <validate-on-match>true</validate-on-match>  <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"/>  </validation>  <timeout>  <idle-timeout-minutes>1</idle-timeout-minutes>  <query-timeout>120</query-timeout>  </timeout> |

JCA can expose the calling datasource invoker to determine those calls that don't close connections and point to resource exhaustion from open/stale connections. We've not closely analyzed transaction connections so follow-up is needed before provided a technical recommendation. In addition, the JMX Console can expose the current connection pool usage (see MaxConnectionsInUse)

Set logging to will expose JDBC connections not closing or timing out. General recommendations follow:

in datasource of domain/\*-ds.xml:

|  |
| --- |
| <track-statements>true</track-statements> </local-tx-datasource>  ...  <track-statements>true</track-statements> </xa-datasource> |

Generally, for any datasource managed by the JBoss Connection Manager, increase logging via root-logger:

|  |
| --- |
| <logger category="jboss.jdbc.spy">  <level name="WARN"/>  </logger>  <logger category="org.jboss.jca.core.connectionmanager">  <level name="WARN"/>  </logger> |

For Oracle, in your logging facility:

|  |
| --- |
| <logger category="oracle.jdbc">  <level name="TRACE"/> </logger> |

And specifically for cached connections under the jca subsystem:

|  |
| --- |
| <cached-connection-manager debug="false" error="true"/> |

See Red Hat logging for Oracle guide: <https://access.redhat.com/solutions/294933>

##### Verify Oracle JDBC, L2 Generally Recommend

CUSTOMER appears to be using Oracle RAC for backend persistence of applications deployed to JBoss. This backend has specific requirements for both Oracle DBs and JBoss appservers. See Red Hat's guide on configuration: <https://access.redhat.com/solutions/22274>

##### Ensure supportability of persistence mapping (ORM), L1 Noted

Hibernate is supported an a standard ORM. Please consider portability for development strategy.

##### Optimize JDBC operations, L1 Noted

JDBC driver for Oracle has options available to optimize database transactions, including: prepared statements, batch inserts, fetch sizing. See Oracle's documentation for details about features available with their driver.

##### Remove test H2 JDBC, L3 Strongly Recommend

The JBoss testing database drive, h2, should not be deployed to Production because it poses both stability and security weaknesses. From domain.xml:

|  |
| --- |
| <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true" use-java-context="true">  <connection-url>jdbc:h2:mem:test;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE</connection-url>  <driver>h2</driver>  <security>  <user-name>sa</user-name>  <password>sa</password>  </security>  </datasource> |

|  |
| --- |
| <driver name="h2" module="com.h2database.h2">  <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class> </driver> |

#### Miscellaneous Resources

The general JBoss Basic/System Thread Pool can be leveraged by development implementations that directly manage threading. See<https://access.redhat.com/site/solutions/694993> if your applications manage multi-threading.

1. **High Availability Recommendations**
   1. **Load Balancers**

CUSTOMER JBoss environments have two load balancers: 1) a network load balancer (NLB) using , and 2) a software load balancer (SLB) using Apache mod\_\*. Both types of load balancers allow high availability of web requests.

### HLB - Hardware Load Balancer

The current Network/Hardware load balancer could supplement load distribution by using either of the more intelligent methods: a) number of users (sessions), or b) least amount of traffic. The Apache mod\_\* SLBs already intelligently distributing load for JBoss's Web container resources from becoming overcommitted.

### SLB - Software Load Balancer

#### Optimized SLB with Apache mod\_jk tuning, L3 Strongly Recommend

CUSTOMER is currently configured with mod\_jk as SLB connector. RH has verified RedHat's latest recommendation for using Apache's Prefork Multi-Processing Module (MPM) for CUSTOMER JBoss . See the [SLB Addendum.](#_36ei31r)This connector is setup with the following characteristics: static cluster list (worker nodes), round robin, binary communication (AJP), sticky sessions. See the [SLB Configuration](#_36ei31r) for details.

#### Reconfigure SLB with Apache mod\_cluster tuning, L2 Generally Recommend

Red Hat supports various types of SLB's for JBoss clusters. See a full comparison here: <https://access.redhat.com/solutions/18495>, and <https://access.redhat.com/solutions/101793>

mod\_cluster enables JBoss to use a dedicated connection to transmit server-side load balance factors and lifecycle events back to Apache HTTPD via a custom set of HTTP methods (the Mod-Cluster Management Protocol (MCMP). This additional feedback channel allows mod\_cluster to offer a level of intelligence and granularity not found in other load balancing solutions.

Remove mod\_cluster extension/module from JBoss:

|  |
| --- |
| <extension module="org.jboss.as.modcluster"/> |

#### Maintain SLB on dedicated, DMZed WebServer Nodes. L3 Strongly Recommended

Architecture inclusive of an SLB is both an organizational decision and a technical decision. The Web Layer can be managed by a different group organizationally, the use of a Web Server can provide this group more control without dealing with change control processes, especially in production operations. The SLB adds intelligent awareness of the JBoss cluster as an Apache module that enables routing and distribution (timeouts and connection retries/handling). Generally, a dedicated native SLB will isolate resource usage to ensure scalability for large concurrency volume and ensure secure communication. As the amount of increase for Big Box JBoss Nodes, the Apache mod\_\* connector frontend/UI(on the layer) will need more resources to respond to requests and maintain sessions. Resources that will be eventually overcommitted with a Big Box design hosting both web server and application server are: 1) threads, and 2) memory. See the Memory Section above for detailed explanation. In addition, the web server (Apache) in general acts as a proxy server to the JBoss servers so it can secure access to the applicat servers, speed up access to resources using caching (optional), apply access policy to network services (optional), and provides auditing/logging and content filtering (optional). The SLB itself should remain in a DMZ as part of CUSTOMER's network security solution.

An Active/Standby configuration can also be setup for the SLB so that multiple Apache Web Servers can be used for high availability.

* 1. **Application Server Clustering**

The horizontal scale of the CUSTOMER JBoss environments should meet fault tolerance and load distribution expectations. The current scale is: 4 JBoss application server Nodes per environment. To enable production ready fault tolerance and load balancing the clustering ability can be selectively added to each profile. Scalability should be verified by add/removing Nodes.

### Verify JGroups clustering dependencies are installed, L2 Generally Recommend

Multicast clustering is disabled. mod\_jk does not support this feature. Unicast communication to the JBoss cluster must be setup. Please refer to <https://access.redhat.com/solutions/1235953>

### Verify HTTP Session Clustering Configuration, L3 Strongly Recommend

Sticky sessions from a web app deployed as distributed will already go to the same servlet instance for the duration of the session. Cookie based sessions can be properly replicated out-of-the-box (<https://access.redhat.com/site/solutions/509743>). When there’s a fault, the session payload is ready to be served by another JBoss instance in the cluster (HTTP session replication is enabled by default in JBoss clusters using mod\_cluster on Apache). Apache should be configured to honor the JSESSION cookie based stickiness (<https://access.redhat.com/site/solutions/265733>) . The affinity to JBoss appservers can be traced via parameters (instance-id) and already defaults to the JBoss server’s name ([jboss.node.name](http://jboss.node.name)). During testing, make sure the correct delimiter is used to specify a session (<https://access.redhat.com/site/solutions/880393>). Example: <http://localhost:8080/app/page;jsessionid=value>

All web applications must be deployed as distributable to enable caching of managed sessions in the Web container across JBoss nodes. JBoss provides a Cache Manager out-of-the-box for caching any distributable web application: Infinispan. Enabling and developing a caching layer is out of the scope of this document however general guides are available since RH noted that CUSTOMER is also deploying EHCache enabled applications.

The Infinispan web cache configuration is in domain.xml.

Local web cache (performance optimization, no fault tolerance):

|  |
| --- |
| <cache-container name="web" aliases="standard-session-cache" default-cache="local-web" module="org.jboss.as.clustering.web.infinispan">  <local-cache name="local-web" batching="true">  <file-store passivation="false" purge="false"/>  </local-cache> </cache-container> |

Replicated web cache (fault tolerance):

|  |
| --- |
| <cache-container name="web" aliases="standard-session-cache" default-cache="repl" module="org.jboss.as.clustering.web.infinispan">  <transport lock-timeout="60000"/>  <replicated-cache name="repl" mode="ASYNC" batching="true">  <file-store/>  </replicated-cache>  <replicated-cache name="sso" mode="SYNC" batching="true"/>  <distributed-cache name="dist" l1-lifespan="0" mode="ASYNC" batching="true">  <file-store/>  </distributed-cache> </cache-container> |

### Verify Distributable Servlet Development & Deployment, L2 Generally Recommend

Ensure servlets are configured using a standard web application deployment descriptor, provided as WEB-INF/web.xml in the WAR file. The URL pattern of “\*” allows the servlet to respond to any HTTP request where the root context matches that of the application (/clusterApp). See example:

|  |
| --- |
| <servlet-mapping> <servlet-name>ClusteredServlet</servlet-name> <url-pattern>/\*</url-pattern> </servlet-mapping> |

By designating the web application as DISTRIBUTABLE, we instruct the container to allow replication of the HTTP session data, as configured in a WAR:

|  |
| --- |
| <distributable /> |

### Verify EJB Clustering, L1 Noted

All Beans implemented with clustering should be deployed to that enable the JBoss CacheManager to manage sessions, caching, and passivation for EJBs. For each profile, the configuration is:

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:ejb3:1.5">  <caches>  ...  <passivation-stores>  ... |

### Verify JNDI Clustering, L1 Noted

All HA components within the cluster should be deployed to that enable the JBoss HA JNDI to for distributed directory lookups.

### Isolate JBoss Clustering networks, L3 Strongly Recommend

RH has noticed that JBoss clusters deployed to same datacenter but "different" networks are improperly isolated. JBoss clusters default to connectionless (UDP) multicast addresses for internode communication. CUSTOMER's networking team should verify that these multicast address spaces are unique to JBoss environments (Production cannot communicate with Staging, etc.) and verify that no cross-talk is occurring over wire.

From domain.xml, the relevant default addresses are:

|  |
| --- |
| <socket-binding-group name="ha-sockets" default-interface="public">  <!-- Needed for server groups using the 'ha' profile -->  <socket-binding name="ajp" port="8009"/>  <socket-binding name="http" port="8080"/>  <socket-binding name="https" port="8443"/>  <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>  <socket-binding name="jgroups-tcp" port="7600"/>  <socket-binding name="jgroups-tcp-fd" port="57600"/>  <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>  <socket-binding name="jgroups-udp-fd" port="54200"/>  <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-port="23364"/> |

See Red Hat's guide for isolating clusters: <https://access.redhat.com/solutions/274263>

* 1. **JBoss Messaging (HornetQ)**

Not enabled in HA profile.

1. **DevOps Recommendations**

The following recommendations are intended to address both development and operational concerns with components not directly related to vertical scaling or high availability. There are are L1 (*Noted*) recommendations.

* 1. **JBoss Rolling Deployments**

WAR/EAR deployments will target JBoss Server Groups for rolling ability. Deployments will target main-server-group / cluster1 for testing. See the [Deployment SOP](https://wiki-citytech.ctmsp.com/display/SAPPHIRIUM/Sony+Deployments+SOP) for pulling a server group out of rotation for Blue/Green style QA/testing.

* 1. **JBoss Classloaders and Classloader Isolation**

The JBoss EAP 6 classloader is modular and uses dependency injection. WAR application should be isolated by default.

### Verify deployment classloader isolation, L1 Noted

### JBoss Classloader - Modular

**1. Boot classpath**

JVM Bootstrap class loader (not related to JBoss) that loads the classes which are required to start a JVM, ex: rt.jar

**2. System classpath**

All jars specified by -classpath/cp at runtime (run.sh etc). Ex: monitoring agent libraries are placed at this level so they can see both the JVM and JBoss.

**3. JBoss modular classloader**

**4. Isolated application classpath:**

Classes and Jars in applications that have an isolated classloader

* 1. **Monitoring Webserver and AppServer Performance**

There are several monitoring and management solutions that are also opensource but almost all Application Performance Monitoring (APM) solutions are commercial products. The selection comes down to being committed to SLAs for your customer satisfaction and the ease of integrating the monitoring solution into the JBosss environments.

### Metrics to Monitor

RH recommends the following metrics be monitored continuously for application stability and capacity analysis:

1. Threading
   1. Count
   2. Processing Time
2. Servlet Response Time
3. HTTP Connections (by Apache for & JBoss for webservice layer)
   1. Request Count
   2. Session Lifecycle
4. Memory Footprints
   1. Apache process
   2. JVM Heap Spaces
5. JVM Full Garbage Collections
   1. Frequency
   2. Duration
6. Transactions
7. Beans
   1. SLSB Pool

### Monitoring Systems

#### Implement monitoring solution, L3 Strongly Recommend

Application Performance Monitoring (APM) solutions and appserver monitoring systems allow analysis of trends, including profile metrics and capacity analysis, and operational response, including alerts and automated recovery, that are specific to JBoss and Apache. Solutions are listed with RH's pros and cons.

1. Zabbix
   1. Zabbix is RH's goto monitoring and management solution for middleware and Java platforms. Zabbix agents ensure application SLAs are met and imposes governance, reduces application downtime and proactively addresses issues by automated operations, and exposes appserver and webserver metrics for analysis.
2. AppDynamics
   1. CUSTOMER has already decided to enable AppDynamics monitoring given its unique profiling and tracing abilities, and RH concurs.

### Apache Monitoring

Out-of-the-box Apache includes mod\_status for basic metrics about HTTPD. Simple enable a Location for publishing those metrics:

httpd.conf:

|  |
| --- |
| ExtendedStatus On <Location /server-status>  SetHandler server-status  Order deny,allow  Deny from all  Allow from all </Location> |

### Tools for JVM Analysis

These tools and utilities allow developers and middleware operations teams to analyze technical data collected by the JVM.

#### Heap Analysis Tools

1. jmap - for getting a heap dump. Example:

-bash-4.1$ jmap -dump:file=/tmp/heap.dump 2706  
Dumping heap to /tmp/heap.dump ...  
Heap dump file created

1. Eclipse MAT - for analyzing the heap dump.

The JVM does pause the application until the dump to disk was complete so that's one caveat. To be clear, IBM has a "Memory Analyzer" too but Eclipse's is different.

#### Thread Analysis Tools

1. jstack - for getting thread dumps. This allows you to take a snapshot of the running threads in a JVM and will show their stack traces
2. jvmtop - this tool allows you to watch the busiest threads and gives you thread IDs. jvmtop can also tell you what threads are blocking others. JVMs must be configured to allow JMX connections for this tool to work

RedHat has additional details on obtaining and analyzing these dumps. See: <https://access.redhat.com/site/node/18178>

#### JBoss Snapshot/Point-in-Time Monitoring

These monitoring tools provide point-in-time snapshots of the JBoss runtime. They are not intended for long term analysis of trends.

##### JConsole

JConsole is a JMX compliant monitoring tool for Oracle JDKs. It is has a GUI that can be used traverse any MBeans. (JConsole is an Oracle tool and v6 of it's user guide is at:<http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html>)

Here are the MBeans exposed for JBoss and viewable by JConsole:



The following is screenshot of pulling the currentThreadsBusy counter from the ThreadPool MBean via JConsole:



##### Verify environments for troubleshooting tools, L2 Generally Recommend

To connect to a JBoss JVM/profile, enable remote/native management interfaces:

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:remoting:1.2">  <connector name="remoting-connector" socket-binding="remoting" security-realm="ApplicationRealm"/>  </subsystem> |

Add ApplicationRealm user.

JConsole Connection Strings:

* JConsole URI for local connect to the appserver (by bouncing GUI back to workstation)
  + service:jmx:remoting-jmx://{ipaddr}:9999

* JConsole URI for remote connect the appserver
  + service:jmx:remoting-jmx://<HOSTNAME>:9999

NOTE: if there is a firewall between the appserver and the workstation running JConsole, additional steps must be taken to compensate for ephemeral ports. See:

* + <https://access.redhat.com/site/solutions/39310>

### Security

#### Isolate Runtime Daemon, L3 Strongly Recommend

Both load balancer and application server runtimes should be secured by local accounts with limited system access. These local accounts should a) not have superuser privileges (including sudo),

|  |
| --- |
| $ pstree ├─domain.sh───java─┬─java───42\*[{java}]   │ ├─java───102\*[{java}]   │ ├─java───79\*[{java}]   │ └─26\*[{java}] $ ps auxwww | grep java applmgr 18625 0.0 0.8 4655612 104696 ? Sl Aug07 1:07 java |

|  |
| --- |
| $ pstree  ├─httpd───6\*[httpd] $ ps auxwww | grep httpd root 23826 0.0 0.0 177976 4676 ? Ss Jul29 0:25 /usr/sbin/httpd apache 34109 0.0 0.0 178108 3564 ? S 08:38 0:00 /usr/sbin/httpd apache 34623 0.0 0.0 178108 3568 ? S 08:47 0:00 /usr/sbin/httpd apache 34637 0.0 0.0 178108 3536 ? S 08:49 0:00 /usr/sbin/httpd apache 36328 0.0 0.0 178108 3560 ? S 09:19 0:00 /usr/sbin/httpd apache 38386 0.0 0.0 178108 3564 ? S 09:56 0:00 /usr/sbin/httpd apache 38405 0.0 0.0 178108 3556 ? S 09:59 0:00 /usr/sbin/httpd |

#### Enable JMX & Web management security, L3 Strongly Recommend

Administration and management interfaces should be secured instead of removed. RH cautions against disabling the Web connector for HTTP (port 8080) for remote management web consoles. Technicians will be unable to go directly to a JBoss instance’s web management console (or bypass the SLB) and any management that uses HTTP (such as web service calls) will fail. It is less intrusive and still secure to limit access to the HTTP port to internal networks only.

Configure credentials for these interfaces for granular access control in domain.xml:

|  |
| --- |
| <management>  <access-control provider="simple">  <role-mapping>  <role name="SuperUser">  <include>  <user name="$local"/>  </include>  </role>  </role-mapping>  </access-control>  </management> |

Verify with local account access to JBoss CLI as well:

|  |
| --- |
| $ ./jboss-cli.sh -c |

1. **References**

JBoss EAP 6 Reference Architecture: <http://www.redhat.com/en/resources/jboss-eap-6-clustering>

[Oracle Hotspot 7 Troubleshooting & Tools:](http://www.redhat.com/en/resources/jboss-eap-6-clustering) <https://docs.oracle.com/javase/7/docs/webnotes/tsg/TSG-VM/html/docinfo.html>

1. **Addendum: SLB Configuration – mod\_jk**

Table of Contents

* Addendum: SLB Configuration
  + [Configuration Generalizations](#_1ljsd9k)
    - [Default MPM MaxClients formulas:](#_2koq656)
    - [Default Web Connector formulas:](#_zu0gcz)
    - [Explanation of Configuration Options](#_3jtnz0s)
  + [Apache PreFork MPM](#_1yyy98l) 
    - [Validate extra capacity, L3 Strongly Recommend:](#_4iylrwe)
    - [Remove Worker MPM configuration, L2 Generally Recommend](#_2y3w247)
  + [Apache mod\_jk](#_1d96cc0)
    - [Verify URI Workmap, L2 Generally Recommend:](#_3x8tuzt)
    - [Alternative: Set JKMount for multiple Virtual Host paths](#_2ce457m)
    - [Verify mod\_jk workers.properties cachesize, L2 Generally Recommend:](#_rjefff)
    - [Add URI mapper mod\_jk uriworkermap.properties, L2 Generally Recommend](#_3bj1y38)
  + [Consider Alternative Load Balancer Connector (mod\_cluster), L2 Generally Recommend](#_1qoc8b1)
  + [HTTP Timeouts](#_4anzqyu)
  + [Verify Apache Timeout, L2 Generally Recommend](#_2pta16n)
  + [Maintain Apache Installation, L1 Noted](#_14ykbeg)
  + [Maintain SLB Compatibility, L1 Noted](#_3oy7u29)
* [JBoss](#_243i4a2)
  + [JBoss Worker Node Setup](#_j8sehv)
  + [Application Sessions and Stickiness](#_338fx5o)
  + [JBoss EAP and Apache HTTPD Tuning](#_1idq7dh)
    - [Verify Connection Pool in Web Container, L3 Strongly Recommend](#_42ddq1a)
  + [JBoss Worker Threads](#_2hio093)
  + [Load Balancer Configuration Templates](#_wnyagw)
* [Ideal Settings & Conclusions](#_3gnlt4p)
  1. **Configuration Generalizations**

The following configuration is recommended as a baseline for SLB (Software Load Balancing) functionality before production certification.

These configurations assume Apache HTTPD Webserver Nodes are configured as load balancing proxies for a horizontally scalable (2+ Node) JBoss EAP cluster. These recommendations highlight common solutions and problems yet they ultimately depend on CUSTOMER load testing. **It is critical to determine the maximal, operational load on both Apache and JBoss to ensure the current infrastructure and configuration meet SLAs**. Web clients have the ability to send multiple requests and the web applications have numerous workflows that alter resource requirements, such as thread pools (used for MaxClients and max-connections settings discussed below) and processor "think time"; and infrastructure faults do occur. Load testing and fault tolerance / risk analysis should ultimately determine any configuration certified for production.

Configurations baselines are derived from CUSTOMER Staging Environment, using supported formulas for the Prefork MPM and HTTP/AJP Java connectors; and assume CUSTOMER JBoss environments have dedicated WebServer Nodes with 4 cores and dedicated AppServer Nodes with 2 cores.

### Default MPM MaxClients formulas:

Prefork: 200 \* cores

Worker: 300 \* cores

### Default Web Connector formulas:

512 \* availableProcessors

### Explanation of Configuration Options

Because Apache and JBoss run on dedicated Nodes, these general formulas apply. Assuming the above MPM formulas and horizontal scale, these estimated optimal settings are included in the following configurations.

For further notes on configuration options and ideals, see: <https://access.redhat.com/site/solutions/94773>, <https://access.redhat.com/site/solutions/770923>

* 1. **Apache PreFork MPM**

Given the above MPM formulas, the recommended maximum for Prefork MaxClients configuration is 800 (200 \* 4 cores). **MaxClients** can be set with apache restarts or reloads whereas **Serverlimit** requires a full stop of apache to change. There is no obvious reason to have these at different settings. **Maxrequestsperchild** has some debate on the ideal setting. Some argue that having this set to 0 lowers the strain on the parent apache process by removing the need to open and close child processes. Others argue that having a number set here (other than 0) makes sure that the apache server doesn't run into any memory leaks. There will be some time before 4000 requests are served so we are not recycling too often and we are making sure that the processes are fresh.

### Validate extra capacity, L3 Strongly Recommend :

* SeverLimit/MaxClients exceeds recommended maximum (800)
* StartServers > default (5)
* MaxRequestsChild > default (0)

httpd.conf, or sourced configuration

**Tabelle 1 MPM module configuration**

|  |
| --- |
| <IfModule prefork.c>  StartServers 8  MinSpareServers 5  MaxSpareServers 20  ServerLimit 1200  MaxClients 1200  MaxRequestsPerChild 4000 </IfModule> |

### Remove Worker MPM configuration, L2 Generally Recommend

This should be unnecessary as this MPM isn't used. httpd.conf, or sourced configuration:

|  |
| --- |
| <IfModule worker.c> StartServers 4 MaxClients 300 MinSpareThreads 25 MaxSpareThreads 75 ThreadsPerChild 25 MaxRequestsPerChild 0 </IfModule> |

* 1. **Apache mod\_jk**

CUSTOMER is currently configured with mod\_jk as SLB connector. This connector is setup with the following characteristics:

* static cluster list (worker nodes/JBoss cluster)
* round robin
* binary communication to worker nodes (AJP)
* sticky sessions

### Verify URI Workmap, L2 Generally Recommend:

Append to httpd.conf or sourced configuration:

JkMountFile conf/uriworkermap.properties

JkWatchdogInterval 60

Remediate httpd.conf or sourced configuration:

Allow from all

**Tabelle 2 httpd.conf / mod\_cluster configuration**

|  |
| --- |
| LoadModule jk\_module modules/mod\_jk.so  JkWorkersFile conf/workers.properties JkLogFile logs/mod\_jk.log JkLogLevel info JkLogStampFormat "[%a %b %d %H:%M:%S %Y]" # For mod\_rewrite compatibility, use +ForwardURIProxy (default since 1.2.24) JkOptions +ForwardKeySize +ForwardURICompatUnparsed -ForwardDirectories JkRequestLogFormat "%w %V %T" JkMountFile conf/uriworkermap.properties JkShmFile run/jk.shm JkWatchdogInterval 60  <Location /jkstatus>  JkMount status  Order deny,allow  #Deny from all  Allow from all </Location> |

### Alternative: Set JKMount for multiple Virtual Host paths

|  |
| --- |
| JkMount /sample/\* web1-loadbalancer |

### Verify mod\_jk workers.properties cachesize, L2 Generally Recommend:

worker.server-one.cachesize=10

Alternative configuration: switch to work templates:

worker.template.port=8009  
worker.template.type=ajp13  
worker.template.ping\_mode=A  
worker.template.socket\_connect\_timeout=10000  
worker.template.connection\_pool\_timeout=600

|  |
| --- |
| worker.list=cluster1-loadbalancer,cluster1-status  worker.server-one.port=8009 worker.server-one.host=192.168.110.135 worker.server-one.type=ajp13 worker.server-one.lbfactor=1 worker.server-one.cachesize=10 ...  worker.cluster1-loadbalancer.balance\_workers=server-one,server-two worker.cluster1-loadbalancer.type=lb worker.cluster1-status.type=status |

### Add URI mapper mod\_jk uriworkermap.properties, L2 Generally Recommend

|  |
| --- |
| # Map application to balancer /myapp=loadbalancer /myapp/\*=loadbalancer |

See Red Hat's guide for all recommended options: <https://access.redhat.com/solutions/395943>

### Consider Alternative Load Balancer Connector (mod\_cluster), L2 Generally Recommend

See Red Hat's comparison between mod\_cluster and mod\_jk: <https://access.redhat.com/solutions/101793>

### HTTP Timeouts

Apache will wait a period of time for web requests to be fulfilled by the web server. This setting should be adjusted to notify the web client that its request is long running. The Apache setting should match the JBoss setting (connectionTimeout="600000") for the AJP connector.

### Verify Apache Timeout, L2 Generally Recommend

In httpd.conf:

Timeout 60

### Maintain Apache Installation, L1 Noted

Apache supportability good.

|  |
| --- |
| $ apachectl -version Server version: Apache/2.2.15 (Unix) Server built: Aug 15 2014 03:02:07   $ httpd -V Server version: Apache/2.2.15 (Unix) Server built: Aug 15 2014 03:02:07 Server's Module Magic Number: 20051115:25 Server loaded: APR 1.3.9, APR-Util 1.3.9 Compiled using: APR 1.3.9, APR-Util 1.3.9 Architecture: 64-bit Server MPM: Prefork  threaded: no  forked: yes (variable process count) Server compiled with....  -D APACHE\_MPM\_DIR="server/mpm/prefork"  -D APR\_HAS\_SENDFILE  -D APR\_HAS\_MMAP  -D APR\_HAVE\_IPV6 (IPv4-mapped addresses enabled)  -D APR\_USE\_SYSVSEM\_SERIALIZE  -D APR\_USE\_PTHREAD\_SERIALIZE  -D SINGLE\_LISTEN\_UNSERIALIZED\_ACCEPT  -D APR\_HAS\_OTHER\_CHILD  -D AP\_HAVE\_RELIABLE\_PIPED\_LOGS  -D DYNAMIC\_MODULE\_LIMIT=128  -D HTTPD\_ROOT="/etc/httpd"  -D SUEXEC\_BIN="/usr/sbin/suexec"  -D DEFAULT\_PIDLOG="run/httpd.pid"  -D DEFAULT\_SCOREBOARD="logs/apache\_runtime\_status"  -D DEFAULT\_LOCKFILE="logs/accept.lock"  -D DEFAULT\_ERRORLOG="logs/error\_log"  -D AP\_TYPES\_CONFIG\_FILE="conf/mime.types"  -D SERVER\_CONFIG\_FILE="conf/httpd.conf" |

Use the following utility to test configurations in apache without restarting:

apachectl configtest

Apache module supportability good, including MPM and mod\_jk:

|  |
| --- |
| $ apachectl -t -D DUMP\_MODULES | egrep 'jk|mpm' Syntax OK  mpm\_prefork\_module (static)  jk\_module (shared) |

|  |
| --- |
| $ strings /etc/httpd/modules/mod\_jk.so | grep mod\_jk  mod\_jk/1.2.40 |

Full compatibility matrix maintained by Red Hat: <https://access.redhat.com/articles/111663>

### Maintain SLB Compatibility, L1 Noted

Red Hat supports and releases SLBs via RHN. Ensure that all WebServer Nodes are registered with RHN for running supportable SLBs over time. For details on installing mod\_jk specifically, see: <https://access.redhat.com/solutions/31521>

* 1. **JBoss**

Essentially, the JBoss Web Container follows recommendations for any application server with a web server connector. Red Hat has commented on these recommendations for both JBoss and Tomcat appservers generally <https://access.redhat.com/node/15786> and for EAP v6 specifically at <https://access.redhat.com/solutions/389513>. Attention should be paid to supported tuning options available for meeting the desired operating conditions or load.

### JBoss Worker Node Setup

JBoss application servers are setup as worker nodes of the mod\_jk extension to Apache via jvmRoute. This is out-of-the-box going to round robin load balance between all worker nodes of mod\_jk. To ensure sticky sessions are possible, aka. stickiness of JSESSIONID, the jvmRoute must be unique for any mod\_jk load balancer. By default, the jvmRoute of every work node of mod\_jk is the server name. Since JBoss server names will default to their hostname but multi-tenant JBoss instances on a single host are common, we should explicitly set unique server names.

domain.xml:

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:web:2.2" default-virtual-server="default-host" instance-id="${jboss.server.name}" native="false"> |

host.xml:

|  |
| --- |
| <server name="server-one" group="main-server-group">   <system-properties>  <property name="jvmRoute" value="server-one"/>  </system-properties> |

### Application Sessions and Stickiness

To enable sessions to stick to a worker node (JBoss server), enable sticky sessions. Also, to enable Java Application Sessions, JSESSIONS, ensure that Apache recognizes Sessions via standard Cookie mechanism. JSESSIONID key/value pairs are the default but you can customize this, so for verbosity's sake see below.

|  |
| --- |
| worker.web1-loadbalancer.sticky\_session=true worker.web1-loadbalancer.session\_cookie=JSESSIONID |

### JBoss EAP and Apache HTTPD Tuning

Red Hat created a matrix for identifying which options are actually effective when connecting JBoss EAP and Apache HTTPD. See <https://access.redhat.com/solutions/389513>.

CUSTOMER is currently running an HTTP/AJP Java connector to EAPv6.x so given the Web Connector formula above and the RH matrix of supported options, the default JBoss Web Connector will set a maximum of 1024 connections (512 \* 2 cores). This setting may be overridden by the JBoss max-connections parameter.

#### Verify Connection Pool in Web Container, L3 Strongly Recommend

in domain.xml:

|  |
| --- |
| <subsystem xmlns="urn:jboss:domain:web:2.2" default-virtual-server="default-host" native="false">  <connector name="ajp" protocol="AJP/1.3" scheme="http" socket-binding="ajp" max-connections="1024" ></connector>  <virtual-server name="default-host" enable-welcome-root="false"> |

### JBoss Worker Threads

The worker thread pool can be used instead of the above mentioned max-connections parameter but we do not recommend this and only mention it for clarity's sake. Threads are created on demand by the OS only when needed by Apache or JBoss resources. minSpareThreads, for example, is typically not needed for Linux OSes because the latest threading libraries (NPTL) already respond efficiently.

These lines would go directly after the </extensions> tag in domain.xml instead of the max-connections setting above:

|  |
| --- |
| <system-properties>  <property name="org.apache.tomcat.util.net.MAX\_THREADS" |

### Load Balancer Configuration Templates

Red Hat can generate baseline SLB configurations using their template generator. To regenerate the templates, goto: <https://access.redhat.com/labs/lbconfig>

* 1. **Ideal Settings & Conclusions**

Ultimately, the maximum amount of load balanced web requests should match the maximum amount of threads that can respond, with additional capacity for fault tolerance. For web applications, this means that the total Apache MaxClients should match the total JBoss max-connections / MaxThreads. CUSTOMER should rationalize any mismatch between Apache and JBoss tuning. This general formula for web applications is ideal:

JBoss max-connections / MaxThreads = Apache HTTPD MaxClients

Obviously, if some clients are not load balanced – and make direct connections to applications deployed to JBoss – then the general formula for (frontend) web applications doesn't apply.

The current JBoss cluster (of 4 worker Nodes) has a total max-connections of 4096 (4 Nodes \* 1024 max-connections), while Apache load balancing (of 2 Nodes) has a total MaxClients of 2400 (2 Nodes \* 1200 MaxClients). *This implies an underutilization of JBoss Nodes or the expected fault tolerance of about 2 JBoss Nodes, assuming all other variables are equal. Assuming the frontend MaxClients is required for SLAs, there is little extra capacity in the SLB for fault tolerance (additional Apache Nodes to assume load distribution).*

See Red Hat's guide on mod\_jk optimization for further details: <https://access.redhat.com/solutions/18415>