JBoss Messaging 2.0 User's Guide

Setting the Standard for High Performance Messaging

Table of Contents

1

About JBoss Messaging 2.0

The goal of JBoss Messaging 2.0 is simple and uncompromising- to bring unrivalled levels of performance and reliability to messaging, and to be the fastest, best featured and most scalable open source messaging system.

We are not in the business of making unfalsifiable statements about performance so we ship this release with a set of basic performance measurements against other open source messaging systems, so you can decide for yourself. All performance results are fully reproducible by you, using the tools shipped with this distribution. Let the facts speak for themselves. If you can't wait, the performance results are provided in Chapter 8, Performance.

The procedure of installing and configuring JBoss Messaging 2.0 is detailed in this guide, along with a set of runnable examples. This guide will be extended, before the general availability (GA) release of JBoss Messaging 2.0

And remember, this is an alpha release! This release is not designed for production use.

Enjoy!

Permanent team: Tim Fox (Lead), Jeff Mesnil, Andy Taylor, Clebert Suconic

2

Introduction

JBoss Messaging 2.0 alpha, is a bare-bones messaging system. This release is designed to show case the elegant architecture and high performance transport and persistence. Many other features, including state of the art clustering will be added before the final general availability (GA) release

JBoss Messaging builds upon the solid performance of JBoss Messaging 1.4 to bring unrivalled levels of performance and scalability

This release contains the following features:

- Class beating, ultra high performance journal based persistence

  We have designed and implemented a fast append only journal which handles the persistence for JBoss Messaging 2.0. The journal is written 95% in Java for portability, and can run in one of two modes:

  a) Using pure Java NIO. This provides great performance and runs on any operating system the JVM runs on.

  b) Using Linux asynchronous IO (aio). This provides even better performance, impossible via Java alone. This is implemented via a thin C++ layer which the journal tasks to via JNI. This option is only available when running on Linux

- JBoss Messaging 2.0 has a new high performance network transport which leverages Apache MINA [http://mina.apache.org/] to provide high performance and high scalability at network layer with an asynchronous API via Java NIO.

  The JBoss Messaging team work closely with Trustin Lee, the lead of MINA, to ensure it's smooth integration.

- Standalone and embedded versions.

  JBoss Messaging can be embedded in your own system without the requirement of any servers at all. Just instantiate a few POJOS (Plain Old Java Objects) and you have a Messaging System running.

- Full JMS 1.1 support

  JBoss Messaging provides a full JMS 1.1 API

- JMS agnostic messaging core

  JBoss Messaging core is actually 100% JMS agnostic. It provides its own non JMS API and fully supports transactional (including XA), reliable, persistent messaging without JMS. The JMS API is actually provided as a thin facade on the client side which translates calls to and from the messaging core API. Abstracting out messaging functionality into a general purpose messaging core makes it easier for us to support other messaging protocols in the future, like AMQP.

3

Download Software

The official JBoss Messaging project page is http://www.jboss.org/jbossmessaging/.

The download location is the JBoss Labs Messaging Project download zone: http://www.jboss.org/jbossmessaging/downloads/

3.1. SVN Access

If you want to experiment with the latest developments you may checkout the latest code from the Messaging SVN trunk. Be aware that the information provided in this manual might then not be accurate. For the latest instructions on how to check out and build source code, please go to Messaging Development wiki [http://wiki.jboss.org/wiki/JBossMessagingDevelopment], and specifically "Building and Running JBoss Messaging" [http://wiki.jboss.org/wiki/JBossMessagingBuildInstructions] page.

4

JBoss Messaging Installation

This section describes how to install JBoss Messaging 2.0.

4.1. Prerequisites

Note

JBoss Messaging only runs with Java 5 or later. We recommend Java 6 for the best performance.

By default, JBoss Messaging server is run with 1GB of memory. If your computer has less memory, modify the value in `bin/run.sh` accordingly.

Ant [http://ant.apache.org/] is required to build and run the examples.

4.2. Installing JBoss Messaging standalone

After downloading the distribution unzip it into your chosen directory. At this point it should be possible to run straight out of the box, the following describes the directory structure that should be seen

```
         |-- bin
         |-- config
         |-- lib
         |-- docs
         | |-- api
         | `-- userguide
         `-- examples
```

- bin

  This contains the binaries and scripts needed to run JBM.

- config

  This contains configuration files needed to configure JBM. Refer to the configuration chapter for details on how to do this.

- lib

  This contains jars needed to run JBM.

- docs

  This contains this user guide and the JBM Javadocs..

- examples

  This contains a set of examples. Refer to the 'running examples' chapter for details on how to run them.

To run JBM, open up a shell or command prompt and navigate into the 'bin' directory. Then execute './run.sh' (or 'run.bat' on windows) and you should see the following output

```
java -Xmx512M -Dorg.jboss.logging.Logger.pluginClass=org.jboss.messaging.core
.logging.JBMLoggerPlugin -Djava.library.path=. -classpath ../lib/xercesImpl.j
ar:../lib/trove.jar:../lib/slf4j-log4j12.jar:../lib/slf4j-api-1.4.3.jar:../li
b/mina-core-2.0.0-M2-20080418.144850.jar:../lib/log4j.jar:../lib/jnpserver.ja
r:../lib/jgroups.jar:../lib/jboss-xml-binding.jar:../lib/jbossts-common.jar:.
./lib/jboss-transaction-spi.jar:../lib/jbosssx-client.jar:../lib/jboss-securi
ty-spi.jar:../lib/jboss-messaging.jar:../lib/jboss-messaging-client.jar:../li
b/jboss-kernel.jar:../lib/jbossjta.jar:../lib/jbossjta-integration.jar:../lib
/jboss-javaee.jar:../lib/jboss-jaspi-api.jar:../lib/jboss-dependency.jar:../l
ib/jboss-container.jar:../lib/jboss-common-logging-spi.jar:../lib/jboss-commo
n-core.jar:../lib/jboss-aop-mc-int.jar:../lib/jboss-aop-jdk50.jar:../lib/java
ssist.jar:../lib/concurrent.jar:../lib/commons-logging.jar:../config/ org.jbo
ss.jms.server.microcontainer.JBMBootstrapServer jbm-standalone-beans.xml
10:25:57,225 INFO @main [JournalStorageManager] Directory /home/andy/jbm-tes
t/data/bindings does not already exists
10:25:57,226 INFO @main [JournalStorageManager] Creating it
10:25:57,270 INFO @main [JournalStorageManager] Directory /home/andy/jbm-tes
t/data/journal does not already exists
10:25:57,271 INFO @main [JournalStorageManager] Creating it
10:25:57,276 INFO @main [JournalStorageManager] AIO loaded successfully
10:25:57,689 INFO @main [MinaService] Registering:tcp://localhost:5400
10:25:57,707 INFO @main [FileDeploymentManager] Deploying org.jboss.messagin
g.core.deployers.impl.SecurityDeployer@fe0fd9 with urlfile:/home/andy/project
s/jBossMessaging/build/messaging-2.0.0.alpha1/config/queues.xml
10:25:57,758 INFO @main [XmlDeployer] deploying topicjms.testTopic
10:25:57,761 INFO @main [XmlDeployer] deploying topicjms.securedTopic
10:25:57,761 INFO @main [XmlDeployer] deploying topicjms.testDurableTopic
10:25:57,762 INFO @main [XmlDeployer] deploying queuejms.testQueue
10:25:57,762 INFO @main [XmlDeployer] deploying queuejms.NoSuchQueue
10:25:57,763 INFO @main [XmlDeployer] deploying topicjms.NoSuchTopic
10:25:57,763 INFO @main [XmlDeployer] deploying queuetempjms.*
10:25:57,764 INFO @main [XmlDeployer] deploying topictempjms.*
10:25:57,764 INFO @main [XmlDeployer] deploying *
10:25:57,765 INFO @main [FileDeploymentManager] Deploying org.jboss.messagin
g.core.deployers.impl.QueueSettingsDeployer@1220b36 with urlfile:/home/andy/p
rojects/jBossMessaging/build/messaging-2.0.0.alpha1/config/queues.xml
10:25:57,783 INFO @main [XmlDeployer] deploying queuejms.QueueWithOwnDLQAndE
xpiryQueue
10:25:57,784 INFO @main [XmlDeployer] deploying topicjms.TopicWithOwnDLQAndE
xpiryQueue
10:25:57,784 INFO @main [XmlDeployer] deploying queuejms.QueueWithOwnRedeliv
eryDelay
10:25:57,784 INFO @main [XmlDeployer] deploying topicjms.TopicWithOwnRedeliv
eryDelay
10:25:57,785 INFO @main [XmlDeployer] deploying queuejms.testDistributedQueu
e
10:25:57,785 INFO @main [XmlDeployer] deploying topicjms.testDistributedTopi
c
10:25:57,785 INFO @main [XmlDeployer] deploying queuejms.testPerfQueue
10:25:57,785 INFO @main [XmlDeployer] deploying *
10:26:02,824 INFO @main [FileDeploymentManager] Deploying org.jboss.messagin
g.core.deployers.impl.SecurityManagerDeployer@1a1ff9 with urlfile:/home/andy/
```

```
         projects/jBossMessaging/build/messaging-2.0.0.alpha1/config/jbm-security.xml
         10:26:02,831 INFO @main [XmlDeployer] deploying guest
         10:26:02,991 INFO @main [FileDeploymentManager] Deploying org.jboss.messagin
         g.jms.server.impl.JMSServerDeployer@d6c07 with urlfile:/home/andy/projects/jB
         ossMessaging/build/messaging-2.0.0.alpha1/config/jbm-jndi.xml
         10:26:03,005 INFO @main [XmlDeployer] deploying DLQ
         10:26:03,035 INFO @main [XmlDeployer] deploying ExpiryQueue
         10:26:03,038 INFO @main [XmlDeployer] deploying testQueue
         10:26:03,044 INFO @main [XmlDeployer] deploying testPerfQueue
         10:26:03,046 INFO @main [XmlDeployer] deploying A
         10:26:03,048 INFO @main [XmlDeployer] deploying B
         10:26:03,050 INFO @main [XmlDeployer] deploying C
         10:26:03,051 INFO @main [XmlDeployer] deploying D
         10:26:03,072 INFO @main [XmlDeployer] deploying ex
         10:26:03,075 INFO @main [XmlDeployer] deploying PrivateDLQ
         10:26:03,077 INFO @main [XmlDeployer] deploying PrivateExpiryQueue
         10:26:03,078 INFO @main [XmlDeployer] deploying QueueWithOwnDLQAndExpiryQueu
         e
         10:26:03,080 INFO @main [XmlDeployer] deploying QueueWithOwnRedeliveryDelay
         10:26:03,081 INFO @main [XmlDeployer] deploying testDistributedQueue
         10:26:03,083 INFO @main [XmlDeployer] deploying testTopic
         10:26:03,086 INFO @main [XmlDeployer] deploying securedTopic
         10:26:03,087 INFO @main [XmlDeployer] deploying testDurableTopic
         10:26:03,088 INFO @main [XmlDeployer] deploying TopicWithOwnDLQAndExpiryQueu
         e
         10:26:03,089 INFO @main [XmlDeployer] deploying TopicWithOwnRedeliveryDelay
         10:26:03,090 INFO @main [XmlDeployer] deploying testDistributedTopic
         10:26:03,091 INFO @main [XmlDeployer] deploying testConnectionFactory
         10:26:03,091 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,101 INFO @main [XmlDeployer] deploying ConnectionFactory
         10:26:03,101 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,102 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,103 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,103 WARN @main [JMSServerManagerImpl] Binding for java:/ConnectionF
         actory already exists
         10:26:03,103 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,104 WARN @main [JMSServerManagerImpl] Binding for java:/XAConnectio
         nFactory already exists
         10:26:03,104 INFO @main [XmlDeployer] deploying ClusteredConnectionFactory
         10:26:03,104 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,105 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,106 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,106 WARN @main [JMSServerManagerImpl] Binding for java:/ClusteredCo
         nnectionFactory already exists
         10:26:03,106 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,106 WARN @main [JMSServerManagerImpl] Binding for java:/ClusteredXA
         ConnectionFactory already exists
         10:26:03,107 INFO @main [XmlDeployer] deploying MyExampleConnectionFactory
         10:26:03,107 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,118 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,130 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,131 INFO @main [JMSServerManagerImpl] Creating cf ** with ws:1000
         10:26:03,133 INFO @main [JBMBootstrapServer] JBM Server Started
```

JBoss Messaging is now running. If any errors are seen, refer to the troubleshooting guide for help

4.3. Installing JBoss Messaging in JBoss AS 5

At this point JBoss Messaging 2 Alpha is a technology preview and we only support the standalone or embedded versions.

5

Running the Examples

In the directory`examples`, you will find 2 sets of examples:

- a set of JMS examples
- a set of non-JMS examples that demonstrate how to use the JBoss Messaging core API

The examples will be expanded on before JBoss Messaging 2.0 GA release

It is highly recommended that you familiarise yourself with the examples.

Make sure you start JBoss Messaging before running the examples!

5.1. The JMS examples

The following JMS examples are provided. To run these you will first need to start the server as shown in the installation chapter.

For each example, you can always override the default ports it will try to connect to by editing jndi.properties in the config directory

To run a specific example open up a shell or command prompt and navigate into the `examples/jms` directory and run the command ant followed by the example name, as follows

```
ant queueExample
```

The output should be similar to the following

```
Buildfile: build.xml

init:
[mkdir] Created dir: /home/andy/projects/jBossMessaging/docs/examples/jms/build

compile:
[javac] Compiling 5 source files to /home/andy/projects/jBossMessaging/docs/
examples/jms/build

queueExample:
[java] 10:59:02,124 INFO @main [QueueExample] sending message to queue
[java] 10:59:02,187 INFO @main [QueueExample] message received from queue
[java] 10:59:02,187 INFO @main [QueueExample] message = This is a text message!

BUILD SUCCESSFUL
```

```
        Total time: 3 seconds
```

The following examples are available

- queueExample

  This example shows a simple send and receive to a remote queue using a JMS client

- topicExample

  This example shows a simple send and receive to a remote topic using a JMS client

- durSubExample

  This example shows the use of a durable subscriber.

- perfSender

  This example will run a basic performance test. It sends messages to a destination according to the specified parameters. This needs to be used in conjunction with the perfListener example. The number of messages, delivery mode etc can be configured as follows:

```
                       ant -Dmessage.count=20000 -Ddelivery.mode=PERSISTENT perfSender
```

  The following parameters can be configured for the sender

  - message.count

    The number of messages to send.

  - delivery.mode

    The delivery mode to use, PERSISTENT or NON_PERSISTENT.

  - message.warmup.count

    How many messages to warm up for. Because of the JIT compiler maximum throughput will take a little while to kick in.

  - message.size

    The size of message to send, in bytes

  - sess.trans

    Whether or not the session is transacted.

  - sess.trans.size

  If the session is transacted the batch size to commit.

  - queue.lookup

    The name of the queue to use.

  - cf.lookup

    The name of the connection factory to use.

- perfListener

  This example will run a basic performance test. It will consume messages from a destination according to the parameters specified. Before running start the example and wait for it to start, you will see `READY!!!` when the listener has started. The number of messages, delivery mode etc can be configured as follows:

  ```
  ant -Dmessage.count=20000 -Ddelivery.mode=PERSISTENT perfSender
  ```

  If running the sender and listener seperately make sure to run the listener with the parameter `drain.queue` set to false

  The following parameters can be configured:

  - message.count

    The number of messages to consume.

  - message.warmup.count

    How many messages to warm up for. Because of the JIT compiler maximum throughput will take a little while to kick in.

  - sess.trans

    Whether or not the session is transacted.

  - sess.trans.size

    If the session is transacted the batch size to commit.

  - sess.ackmode

    The acknowledge mode to use, DUPS_OK or AUTO_ACK. Ignored if the session is transacted

  - drain.queue

    Whether or not the listener will empty the queue before starting.

  - queue.lookup

  The name of the queue to use.

- cf.lookup

  The name of the connection factory to use.

There are also some ant targets for running the perf sender and listener in different modes:

```
                  perfNonTransactionalSender
                  perfTransactionalSender
                  perfAutoAckListener
                  perfDupsOKListener
                  perfDupsOKListener
```

5.2. The Messaging examples

The messaging examples demonstrate the use of the messaging core API and also how to create and run an embedded instance of JBM. The following examples are available:

to run a specific example open up a shell or command prompt and navigate into the `examples/messaging` directory and run the command ant followed by the example name, as follows

```
         ant simpleClient
```

- SimpleClient

  This example shows a simple send and receive to a remote queue using a core messaging client. The server will need to be running for this example.

- SSLClient

  This example shows a simple send and receive to a remote queue using SS. The server will need to be running and configured to use SSL for this example. Refer to the configuration chapter for details on how to do this.

- simpleExample

  This example shows how to create an embedded JBM server using the core API. The server must not have been started before running this example.

6

Configuration

In this chapter, we discuss how to configure JBoss Messaging

JBoss Messaging configuration is spread among several configuration files:

- `jbm-configuration.xml`
- `jbm-security.xml`
- `queues.xml`
- `jbm-jndi.xml`
- `jbm-standalone-beans.xml` or `jbm-beans.xml`

The next sections explain each configuration file in detail.

6.1. jbm-configuration.xml

This configuration file is the core configuration for the JBM server. The following is an example of a typical configuration:

```
<deployment>
   <configuration>
      <clustered>false</clustered>

      <scheduled-executor-max-pool-size>30</scheduled-executor-max-pool-size>

      <require-destinations>true</require-destinations>

      <!-- Remoting configuration -->

      <!-- one of: TCP, INVM -->
      <!-- INVM: the server is accessible only by clients in the same VM
      (no sockets are opened) -->
      <remoting-transport>TCP</remoting-transport>

      <remoting-bind-address>5400</remoting-bind-address>

      <remoting-host>localhost</remoting-host>

      <!--  timeout in milliseconds -->
      <remoting-timeout>5000</remoting-timeout>

      <!-- true to disable invm communication when the client and the server are in the
      same JVM.     -->
```

```
          <!-- it is not allowed to disable invm communication when the remoting-transport
          is set to INVM -->
          <remoting-disable-invm>false</remoting-disable-invm>

          <!-- Enable/Disable Nagle's Algorithm (resp. true/false) -->
          <!-- This setting is taken into account only when remoting-transport is set to
          TCP -->
          <remoting-tcp-nodelay>false</remoting-tcp-nodelay>

          <!-- Set the TCP Receive Buffer size (SO_RCVBUF). -->
          <!-- Set it to -1 if you want to use the value hinted by the Operating System -->
          <!-- This setting is taken into account only when remoting-transport is set to
          TCP -->
          <remoting-tcp-receive-buffer-size>-1</remoting-tcp-receive-buffer-size>

          <!-- Set the TCP Send Buffer size (SO_SNDBUF).-->
          <!-- Set it to -1 if you want to use the value hinted by the Operating System-->
          <!-- This setting is taken into account only when remoting-transport is set to
          TCP -->
          <remoting-tcp-send-buffer-size>-1</remoting-tcp-send-buffer-size>

          <!-- The interval to send a ping message to send to the client/server to make sure
          it is still alive.-->
          <!-- Set to 0 if you want to disable this functionality-->
          <remoting-keep-alive-interval>10000</remoting-keep-alive-interval>

          <!--How long to wait for a returning pong after sending a ping message to a
          client/server.-->
          <!-- If no pong is received after this time resources are cleaned up-->
          <remoting-keep-alive-timeout>5000</remoting-keep-alive-timeout>

          <!--  if ssl is enabled, all remoting-ssl-* properties must be set -->
          <remoting-enable-ssl>false</remoting-enable-ssl>

          <remoting-ssl-keystore-path>messaging.keystore</remoting-ssl-keystore-path>

          <remoting-ssl-keystore-password>secureexample</remoting-ssl-keystore-password>

          <remoting-ssl-truststore-path>messaging.truststore</remoting-ssl-truststore-path>

          <remoting-ssl-truststore-password>secureexample</remoting-ssl-truststore-password>

          <!-- Storage configuration -->

          <bindings-directory>${user.home}/jbm-test/data/bindings</bindings-directory>

          <create-bindings-dir>true</create-bindings-dir>

          <journal-directory>${user.home}/jbm-test/data/journal</journal-directory>

          <create-journal-dir>true</create-journal-dir>

          <journal-type>asyncio</journal-type>

          <!-- Does the journal sync to disk on each transaction commit,
          prepare or rollback? -->
          <journal-sync-transactional>true</journal-sync-transactional>

          <!-- Does the journal sync to disk for every non transactional
          persistent operation? -->
          <journal-sync-non-transactional>false</journal-sync-non-transactional>

          <journal-file-size>10485760</journal-file-size>

          <journal-min-files>10</journal-min-files>
```

```
        <!-- Maximum simultaneous asynchronous writes accepted by the native layer.
             (parameter ignored on NIO) -->
        <journal-max-aio>9000</journal-max-aio>

        <!--  Maximum time in milliseconds an AIO operation could take.
              This includes:
              - closing Asynchronous files
              - Transaction awaits
              - Awaits on non transactional writes
         -->
        <journal-aio-timeout>90000</journal-aio-timeout>

        <journal-task-period>5000</journal-task-period>

        <security-enabled>true</security-enabled>

     </configuration>

  </deployment>
```

The available configuration attributes are:

- scheduled-executor-max-pool-size

  The maximum number of threads available for scheduling delivery of scheduled messages

- require-destinations

  Whether or not a destination needs to pre-exist when delivering a message

- remoting-transport

  Type of transport to use, currently there is only TCP

- remoting-bind-address

  The port that JBM will bind to.

- remoting-host

  The name of the host that JBM will bind to

- remoting-timeout

  The timeout setting, in milliseconds, for both client and server connections

- remoting-disable-invm

  not used at present

- remoting-tcp-nodelay

  Sets the TCP nodelay setting when a TCP transport is used

- remoting-tcp-receive-buffer-size

sets the TCP receive buffer size, -1 will set it to the value that the OS hints at to use. This is only used when TCP transport is configured

- remoting-tcp-send-buffer-size

  sets the TCP send buffer size, -1 will set it to the value that the OS hints at to use. This is only used when TCP transport is configured

- remoting-keep-alive-interval

  The interval, in milliseconds, at which a ping message will be sent to the client/server to make sure it is still alive. Setting to 0 will disable pinging

- remoting-keep-alive-timeout

  The time, in milliseconds, to wait for a pong after a ping has ben sent to a client/server. If the pong isn't received after this timeout then the resources are cleaned up.

- remoting-enable-ssl

  Whether SSL is enabled for this server. If this is true then next 4 SSL properties need to be set

- remoting-ssl-keystore-path>messaging.keystore

  The location of the SSL keystore

- remoting-ssl-keystore-password

  The password for the SSL keystore

- remoting-ssl-truststore-path

  The location of the SSL truststore

- remoting-ssl-truststore-password

  The password for the truststore

- bindings-directory

  The directory to create the bindings persistence files in.

- create-bindings-dir

  Whether to create the bindings directory if it doesnt exist.

- journal-type

  The type of journal to use, valid configurations are 'asyncio','nio' and 'jdbc'. refere to the 'The journal based persistence approach' chapter for more detailed information

- journal-sync-transactional

Whether or not to synch to disk for a transaction on commit, prepare or rollback. false will only write to the OS buffers and lets the OS deal with synching

- journal-sync-non-transactional

  Whether or not to synch to disk for every non transaction persistent operation. false will only write to the OS buffers and lets the OS deal with synching

- journal-file-size

  The size of the data file to create, these files are pre-allocated and filled as needed.

- journal-min-files

  Minimum number of created files to start with

- journal-max-aio

  Maximum pending asynchronous writes accepted by the native layer per opened file. There is a limit and the total max AIO can't be higher than /proc/sys/fs/aio-max-nr. If you are combining the usage of JBoss Messaging with other systems that are using libaio (e.g. Oracle) you might need to increase this value on the OS. This parameter is only available on AIO which is only available on Linux at the moment.

- journal-aio-timeout

  Maximum amount of time any asynchronous operation can take in milliseconds. If any operation takes more than this amount a timeout exception will be logged. This parameter is only available on AIO which is only available on Linux.

- journal-task-period

  How frequently to reclaim unused journal data files, in milliseconds.

- security-enabled

  Whether security is enabled, if false no security checks are made.

6.2. jbm-security.xml

This configuration file is used to configure users and roles when JBM is running in standalone mode using the JBM Security Manager. The Security manager used is a pluggable component whose implementation can be changed by configuring the appropriate beans configuration file. Refer to the beans configuration section on how to do this. A typical jbm-security.xml config looks like:

```
<deployment>
   <user name="guest" password="guest">
      <role name="guest"/>
   </user>
</deployment>
```

The available configuration attributes are:

- user

  The user to add to the security manager. This must have the attribute 'name' and 'password' set.

  - role

    A role that the user has, a user may have multiple roles configured.

6.3. queues.xml

This configuration file is used to configure the security and settings for destinations. These are matched against a destination using an hierarchical style match that supports both wild cards ('*') and word replacement ('^')

For instance a destination withname 'queuejms.aqueue.myQueue' would match against 'queuejms.*', 'queuejms.aqueue.^', 'queuejms.^.myQueue' and obviously 'queuejms.aqueue.myQueue'. If a destination has multiple matches then the most precise match is used

```
<deployment>

   <security match="topicjms.testTopic">
      <permission type="create" roles="durpublisher"/>
      <permission type="read" roles="guest,publisher,durpublisher"/>
      <permission type="write" roles="guest,publisher,durpublisher"/>
   </security>

   <security match="topicjms.securedTopic">
      <permission type="write" roles="publisher"/>
      <permission type="read" roles="publisher"/>
   </security>

   <security match="topicjms.testDurableTopic">
      <permission type="create" roles="durpublisher"/>
      <permission type="read" roles="guest,publisher,durpublisher"/>
      <permission type="write" roles="guest,publisher,durpublisher"/>
   </security>

   <security match="queuejms.testQueue">
      <permission type="read" roles="guest,publisher"/>
      <permission type="write" roles="guest,publisher"/>
   </security>

   <security match="queuejms.NoSuchQueue">
      <permission type="read" roles="guest,publisher"/>
      <permission type="write" roles="guest,publisher"/>
   </security>

   <security match="topicjms.NoSuchTopic">
      <permission type="read" roles="guest,publisher"/>
      <permission type="write" roles="guest,publisher"/>
   </security>
```

```
      <security match="queuetempjms.*">
         <permission type="create" roles="guest,def"/>
         <permission type="read" roles="guest,def"/>
         <permission type="write" roles="guest,def"/>
      </security>

      <security match="topictempjms.*">
         <permission type="create" roles="guest,def"/>
         <permission type="read" roles="guest,def"/>
         <permission type="write" roles="guest,def"/>
      </security>

      <!--this will catch any word i.e. queuejms.anything-->
      <!--<security match="queuejms.^">
         <permission type="read" roles="guest,publisher"/>
         <permission type="write" roles="guest,publisher"/>
      </security>-->

      <!--this will catch any word i.e. queuejms.anything-->
      <!--<security match="topicjms.^">
         <permission type="read" roles="guest,publisher"/>
         <permission type="write" roles="guest,publisher"/>
      </security>-->

      <!--default security to catch all-->
      <security match="*">
         <permission type="create" roles="guest,def"/>
         <permission type="read" roles="guest,def"/>
         <permission type="write" roles="guest,def"/>
      </security>

      <queue-settings match="queuejms.QueueWithOwnDLQAndExpiryQueue">
         <dlq>PrivateDLQ</dlq>
         <expiry-queue>queuejms.PrivateExpiryQueue</expiry-queue>
      </queue-settings>

      <queue-settings match="topicjms.TopicWithOwnDLQAndExpiryQueue">
         <dlq>PrivateDLQ</dlq>
         <expiry-queue>queuejms.PrivateExpiryQueue</expiry-queue>
      </queue-settings>

      <queue-settings match="queuejms.QueueWithOwnRedeliveryDelay">
         <redelivery-delay>5000</redelivery-delay>
      </queue-settings>

      <queue-settings match="topicjms.TopicWithOwnRedeliveryDelay">
         <redelivery-delay>5000</redelivery-delay>
      </queue-settings>

      <queue-settings match="queuejms.testDistributedQueue">
         <clustered>true</clustered>
      </queue-settings>

      <queue-settings match="topicjms.testDistributedTopic">
         <clustered>true</clustered>
      </queue-settings>

      <queue-settings match="queuejms.testPerfQueue">
         <clustered>false</clustered>
      </queue-settings>

      <!--default for catch all-->
      <queue-settings match="*">
         <clustered>false</clustered>
         <dlq>DLQ</dlq>
         <expiry-queue>queuejms.ExpiryQueue</expiry-queue>
         <redelivery-delay>0</redelivery-delay>
         <max-size>-1</max-size>
         <distribution-policy-class>
            org.jboss.messaging.core.server.impl.RoundRobinDistributionPolicy
         </distribution-policy-class>
         <message-counter-history-day-limit>10</message-counter-history-day-limit>
      </queue-settings>

   </deployment>
```

The available configuration attributes are:

- security

  The securitysettings to use when clients access a destination.

  - permission

    This describes the permissions a user must have to perform certain tasks. The permission type can be 'create','write' or 'read' and the roles are a comma seperated list of roles.

- queue-settings

  These are the settings applied to a queue its creation.

  - clustered

    Whether or not this queue is clustered

  - dlq

    The name of the Dead Letter Queue to use for this queue

  - expiry-queue

    The name of the Expiry Queue to use for this queue

  - redelivery-delay

    How long to wait, in milliseconds, before trying to redeliver a message.

  - max-size

    The maximum number of messages a queue can hold before rejecting. -1 means unlimited which is the default

  - distribution-policy-class

    The distribution policy class to use when multiple consumers are registered with a single queue. A round robin policy is used by default.

6.4. jbm-jndi.xml

This configuration file is used to create destinations and Connection Factories and make them available in JNDI. Note that this is the only configuration file that exposes JMS functionality, the rest of the configuration is 100% JMS agnostic.

A typical jbm-jndi.xml config looks like:

```
<deployment>

<connection-factory name="testConnectionFactory">
   <entry name="testConnectionFactory"/>
</connection-factory>

<connection-factory name="ConnectionFactory">
   <entry name="/ConnectionFactory"/>
   <entry name="/XAConnectionFactory"/>
   <entry name="java:/ConnectionFactory"/>
   <entry name="java:/XAConnectionFactory"/>
</connection-factory>

<connection-factory name="ClusteredConnectionFactory">
   <entry name="/ClusteredConnectionFactory"/>
   <entry name="/ClusteredXAConnectionFactory"/>
   <entry name="java:/ClusteredConnectionFactory"/>
   <entry name="java:/ClusteredXAConnectionFactory"/>
   <supports-failover>true</supports-failover>
   <supports-load-balancing>true</supports-load-balancing>
</connection-factory>

<connection-factory name="MyExampleConnectionFactory">
   <entry name="/MyExampleConnectionFactory"/>
   <entry name="/acme/MyExampleConnectionFactoryDupe"/>
   <entry name="java:/xyz/CF1"/>
   <entry name="java:/connectionfactories/acme/connection_factory"/>
   <!-- You can specify the default Client ID to use for connections created using
   this factory -->
   <client-id>MyClientID</client-id>
   <!-- The batch size to use when using the DUPS_OK_ACKNOWLEDGE acknowledgement mode -->
   <dups-ok-batch-size>5000</dups-ok-batch-size>-size>
   <!-- This is the window size in number of messages to use when using producer window
   based flow control -->
   <producer-window-size>1000</producer-window-size>
   <!-- This is the maximum producer send rate that will be applied when using rate
   based producer flow control -->
   <producer-max-rate>100</producer-max-rate>
    <!-- This is the window size in number of messages to use when using consumer window
    based flow control -->
   <consumer-window-size>1000</consumer-window-size>
   <!-- This is the maximum producer send rate that will be applied when using rate
   based consumer flow control -->
   <consumer-max-rate>5000</consumer-max-rate>
   <!--Whether or not we use a blocking call when acknowledging a message-->
   <block-on-acknowledge>false</block-on-acknowledge>
   <!--Whether we send non persistent messages synchronously-->
   <send-np-messages-synchronously>true</send-np-messages-synchronously>
   <!--Whether we send persistent messages synchronously-->
   <send-p-messages-synchronously>true</send-p-messages-synchronously>

</connection-factory>
```

```
   <queue name="DLQ">
      <entry name="/queue/DLQ"/>
   </queue>
   <queue name="ExpiryQueue">
      <entry name="/queue/ExpiryQueue"/>
   </queue>
   <topic name="testTopic">
      <entry name="/topic/testTopic"/>
   </topic>
   <topic name="securedTopic">
      <entry name="/topic/securedTopic"/>
   </topic>
   <topic name="testDurableTopic">
      <entry name="/topic/testDurableTopic"/>
   </topic>
   <queue name="testQueue">
      <entry name="/queue/testQueue"/>
   </queue>
   <queue name="testPerfQueue">
      <entry name="/queue/testPerfQueue"/>
   </queue>
   <queue name="A">
      <entry name="/queue/A"/>
   </queue>
   <queue name="B">
      <entry name="/queue/B"/>
   </queue>
   <queue name="C">
      <entry name="/queue/C"/>
   </queue>
   <queue name="D">
      <entry name="/queue/D"/>
   </queue>
   <queue name="ex">
      <entry name="/queue/ex"/>
   </queue>
   <queue name="PrivateDLQ">
      <entry name="/queue/PrivateDLQ"/>
   </queue>
   <queue name="PrivateExpiryQueue">
      <entry name="/queue/PrivateExpiryQueue"/>
   </queue>
   <queue name="QueueWithOwnDLQAndExpiryQueue">
      <entry name="/queue/QueueWithOwnDLQAndExpiryQueue"/>
   </queue>
   <topic name="TopicWithOwnDLQAndExpiryQueue">
      <entry name="/topic/QueueWithOwnDLQAndExpiryQueue"/>
   </topic>
   <queue name="QueueWithOwnRedeliveryDelay">
      <entry name="/queue/QueueWithOwnRedeliveryDelay"/>
   </queue>
   <topic name="TopicWithOwnRedeliveryDelay">
      <entry name="/queue/TopicWithOwnRedeliveryDelay"/>
   </topic>
   <queue name="testDistributedQueue">
      <entry name="/topic/testDistributedQueue"/>
   </queue>
   <topic name="testDistributedTopic">
      <entry name="/topic/testDistributedTopic"/>
   </topic>

</deployment>
```

The available configuration attributes are:

- connection-factory

  The connection factory to create with a unique name

  - entry

    The name to store the Connection Factory object in JNDI with. Multiple JNDI entries can be added.

  - client-id

    The client id for connections created using this Connection Factory

  - dups-ok-batch-size

    The number of acks to batch up when DUPS_OK_ACKNOWLEDGE acknowledgement mode is used

  - producer-window-size

    This is the window size in number of messages to use when using producer window based flow control

  - producer-max-rate

    This is the maximum producer send rate that will be applied when using rate based producer flow control.

  - consumer-window-size

    This is the window size in number of messages to use when using consumer window based flow control

  - consumer-max-rate

    This is the maximum producer send rate that will be applied when using rate based consumer flow control.

  - block-on-acknowledge

    Whether or not we use a blocking call when acknowledging a message

  - send-np-messages-synchronously

    Whether we send non persistent messages synchronously.

  - send-p-messages-synchronously

    Whether we send persistent messages synchronously.

- queue

  The queue to create with a unique name.

  - entry

The name to store the Connection Factory object in JNDI with. Multiple JNDI entries can be added.

- topic

  The queue to create with a unique name.

  - entry

    The name to store the Connection Factory object in JNDI with. Multiple JNDI entries can be added.

6.5. The beans deployment file

This beans deployment file, usually `jbm-beans.xml` or`jbm-standalone-beans.xml`, is used by the JBoss Microcontainer [http://www.jboss.org/jbossmc/] to bootstrap all the components needed to run a JBoss Messaging Server. For the purposes of configuring JBM it is sufficient to know that the implementation details of pluggable components are configured here.

The following explains each component in more detail

- The naming Service

  This is only found in the standalone version of the beans file. When running within the App Server this is not needed since it is available as its own service. This is also where you can change the ports used.

  It is possible to replace this with any Naming Service however only the JBoss naming provider has been tested. If you do provide your own implementation remember to edit the file jndi.properties

```
<bean name="Naming" class="org.jnp.server.NamingBeanImpl"/>

<bean name="Main" class="org.jnp.server.Main">
   <property name="namingInfo"><inject bean="Naming"/> </property>
   <property name="port">1099</property>
   <property name="bindAddress"><inject bean="Configuration" property="host"/></property>
   <property name="rmiPort">1098</property>
   <property name="rmiBindAddress"><inject bean="Configuration" property="host"/></property>
</bean>
```

- The Configuration component

```
   <bean name="Configuration"
      class="org.jboss.messaging.core.config.impl.FileConfiguration"/>
```

  The Configuration component is used to configure the JBM Server and transports. The default implementation, `FileConfiguration` reads in the configuration from the file`jbm-configuration.xml`. To replace this component your class must implement following interface:

```
org.jboss.messaging.core.config.Configuration
```

- The Security Manager

  There are 2 Security Manager implementations available to use. In standalone the default is the following

```


   <bean name="JBMSecurityManager"
      class="org.jboss.messaging.core.security.impl.JBMSecurityManagerImpl">
      <constructor>
         <parameter>false</parameter>
      </constructor>
   </bean>


```

  This uses a simple security manager that also needs the following bean deployed which will read the security configuration from the file `jbm-security.xml`

```


   <bean name="SecurityManagerDeployer"
      class="org.jboss.messaging.core.deployers.impl.SecurityManagerDeployer">
      <property name="jbmSecurityManager">
         <inject bean="JBMSecurityManager"/>
      </property>
      <property name="messagingServer">
         <inject bean="MessagingServer"/>
      </property>
   </bean>


```

  The second is used when deployed in the JBoss App Server and will make use of JAAS:

```


      <bean name="JBMSecurityManager"
      class="org.jboss.messaging.core.security.impl.JAASSecurityManager"/>


```

  To replace the Security Manager implement the following interface:

```
org.jboss.messaging.core.security.JBMSecurityManager
```

- Messaging Server Management

  This exposes Server management operations via JMX. This can be removed if this functionality is not needed

```


   <bean name="MessagingServerManagement"
   class="org.jboss.messaging.core.management.impl.MessagingServerManagementImpl">
```

```
      <annotation>
         @org.jboss.aop.microcontainer.aspects.jmx.JMX(
         name="jboss.messaging:service=MessagingServerManagement",
         exposedInterface=org.jboss.messaging.core.management.MessagingServerManagement.class
         )</annotation>
      <property name="messagingServer">
         <inject bean="MessagingServer"/>
      </property>
   </bean>
```

- The Messaging Server

  This must never be changed as it is the core messaging server

```


   <bean name="MessagingServer"
   class="org.jboss.messaging.core.server.impl.MessagingServerImpl">
      <property name="storageManager">
         <inject bean="StorageManager"/>
      </property>
      <property name="remotingService">
         <inject bean="RemotingService"/>
      </property>
      <property name="configuration">
         <inject bean="Configuration"/>
      </property>
      <property name="securityManager">
         <inject bean="JBMSecurityManager"/>
      </property>
   </bean>
```

- The Storage Manager

  The Storage manager deals with the persistence of messages and bindings. For more information on this refer to the chapter 'The journal based persistence approach'.

```


   <bean name="StorageManager"
      class="org.jboss.messaging.core.persistence.impl.journal.JournalStorageManager">
      <constructor>
         <parameter>
            <inject bean="Configuration"/>
         </parameter>
      </constructor>
   </bean>
```

  To replace this pluggable component implement the following interface:

```
org.jboss.messaging.core.persistence.StorageManager
```

- The Remoting Service

The Remoting Service is the transport used by the JBM server

```
<bean name="RemotingService"
   class="org.jboss.messaging.core.remoting.impl.mina.MinaService">
   <constructor>
      <parameter>
         <inject bean="Configuration"/>
      </parameter>
   </constructor>
</bean>
```

To replace this pluggable component implement the following interface:

```
org.jboss.messaging.core.remoting.RemotingService
```

- The JMS Server Manager

  The JMS Server Manager exposes JMS operations via JMX. This can be removed if not needed.

```
<bean name="JMSServerManager"
class="org.jboss.messaging.jms.server.impl.JMSServerManagerImpl">
   <annotation>
      @org.jboss.aop.microcontainer.aspects.jmx.JMX(
         name="jboss.messaging:service=JMSServerManager",
         exposedInterface=org.jboss.messaging.jms.server.JMSServerManager.class)
         </annotation>
   <property name="messagingServerManagement">
      <inject bean="MessagingServerManagement"/>
   </property>
</bean>
```

- The JMS Server Deployer

  The JMS Server Deployer takes care of deploying Destinations and Connection Factories into JNDi via the file`jbm-jndi.xml`. This can be removed if no objects are needed in JNDI or if only core messaging is being used.

```
<bean name="JMSServerDeployer"
class="org.jboss.messaging.jms.server.impl.JMSServerDeployer">
   <property name="jmsServerManager">
      <inject bean="JMSServerManager"/>
   </property>
   <property name="messagingServer">
      <inject bean="MessagingServer"/>
   </property>
</bean>
```

7

The journal based persistence approach

7.1. ASYNCIO

If you are using JBoss Messaging 2.0 on a Linux system, you can take full advantage of this feature. All you have to do is to make sure libaio is installed and you are using an ext3 or ext2 file system, and kernel version 2.6 or later.

To install libaio, run the command `sudo yum install libaio1` on Fedora or Red Hat or `sudo apt-get install libaio1` on Ubuntu or Debian. For other OS's refer to the OS manual.

Using ther JBoss Messaging journal we provide unrivalled persistence performance. Instead of issuing a sync on every commit required by the journal, we submit writes directly to the kernel and we get callbacks when the information is stored on the hard drive. With that we maximize performance by isolating the persistence of one transaction from another and also by using Direct Memory Access between the Journal and the Kernel. With AIO you can have high rate transactions even when you commit several small transactions, and that's done without any loss of the reliability guarantee - your data is 100% guaranteed persisted to disk. We are planning migrating this native library to other platforms besides Linux, as other OS's will have different options for Asynchronous IO.

7.2. NIO

If AIO is not available JBM will automatically fall back to using NIO. Both NIO and AIO journals share a lot of its implementation at an abstract level, and the NIO journal also gives excellent performance.

7.3. JDBC

JDBC mapping is not supported for the Alpha release of JBoss Messaging 2.

It is planned for the Beta release for users that require a database for management purposes. JDBC access will be optimized but it is not expected to reach the same level of performance for persistent messages than the ASYNCIO and NIO implementations.

8

Performance Tests

8.1. How to execute our performance tests

JBoss Messaging 2.0 ships with a set of basic performance measurements comparing our performance against a selection of other open source messaging systems. We would like to provide performance figures against proprietary systems too, but unfortunately, this is usually prohibited by the other messaging system's licence terms

> **Note**
>
> It should be stressed that these performance figures are far from exhaustive and provide a basic view for how JBoss Messaging 2.0 performs against the other systems in a selection of simple and classic JMS use cases. All the tests use the standard JMS 1.1 API. Please remember JBoss Messaging 2.0 is only an alpha release so don't expect this release to be perfect in all scenarios, although it is interesting to observe that even at this early stage JBoss Messaging 2.0 appears to provide better performance than the other messaging systems. JBoss Messaging 2.0 is still largely un-optimised - we have more performance to squeeze out still. More in depth performance figures will be provided for the final GA release.

In the spirit of open-ness and not wanting to make performance claims we cannot substantiate, all these performance figures can easily be reproduced using just the tools available in this distribution, assuming you have installed the other messaging systems. We have used the JMS Examples to produce these numbers, and we used perfSender and perfListener with different scenarios. You can replicate those scenarios using these instructions provided below:

All messages used in the tests are BytesMessage instances with 1K bodies. Measurement time is taken from the time of the first message sent after the warmup period to the time of the last message consumed. Message throughput rates are measured in messages / sec. In most tests 200000 messages were sent, although this was reduced with some providers due to the provider running out of memory. In all tests we first start a consumer listening on the queue, then start a producer sending to the queue. Broker was running on the server machine, and both the producer and the consumer were running on the client machine.

The tests were performed on very basic commodity hardware. In the near future we will be obtaining the use of a large performance lab with serious hardware, on which we look forward to obtaining more results

- Test 1. Send non persistent, non transactional messages. Consume messages non transactional with ack mode DUPS_OK_ACKNOWLEDGE

```
ant perfListener
ant perfSender
```

- Test 2. Send non persistent, non transactional, messages. Consume mesages non transactional with ack mode AUTO_ACKNOWLEDGE

```
ant perfListener -Dsess.ackmode=AUTO_ACK

ant perfSender
```

- Test 3. Send persistent (blocking *) non transactional, messages. Consume with ack mode DUPS_OK_ACKNOWLEDGE

```
ant perfListener

ant perfSender -Ddelivery.mode=PERSISTENT
```

- Test 4. Send persistent (non blocking *), non transactional, messages. Consume with ack mode DUPS_OK_ACKNOWLEDGE

```
ant perfListener

ant perfSender -Ddelivery.mode=PERSISTENT
```

- Test 5. Send persistent (blocking *), non transactional, messages. Consume with ack mode AUTO_ACKNOWLEDGE

```
ant perfListener -Dsess.ackmode=AUTO_ACK

ant perfSender -Ddelivery.mode=PERSISTENT
```

- Test 6. Send persistent (non blocking *), non transactional, messages. Consume with ack mode AUTO_ACKNOWLEDGE

```
ant perfListener -Dsess.ackmode=AUTO_ACK


ant perfSender -Ddelivery.mode=PERSISTENT
```

- Test 7. Large transaction test. Send persistent, transactional, messages with transaction size = 1000. Consume persistent transactional, transaction size = 1000

```
ant perfListener -Dsess.trans=true -Dsess.trans.size=1000


ant perfSender -Dsess.trans=true -Dsess.trans.size=1000 -Ddelivery.mode=PERSISTENT
```

- Test 8. Small transaction test. Send persistent, transactional, messages with transaction size = 2. Consume persistent transactional, transaction size = 2

```
ant perfListener -Dsess.trans=true -Dsess.trans.size=2


ant perfSender -Dsess.trans=true -Dsess.trans.size=2 -Ddelivery.mode=PERSISTENT
```

* Some messaging systems send persistent messages by default synchronously, and others send them by default asynchronously. Some supports both modes and others only support one. To avoid confusion we consider sending persistent messages synchronously or asynchronously separately. Configuring a particular system to send synchronously or asynchronously is specific to the system. For instance in Apache QPID we can specify use blocking mode by providing the system property "sync_persistence" with the value "true" to the client.

8.2. Systems Used

All the tests were executed on the same hardware, operating system and JDK configuration. All messaging system configuration was with out of the box defaults unless otherwise stated.

- Client:

Hardware: Dell Latitude D820, dual core 2 x 2GHz Intel CoreDuo, 2GB RAM, 1GB ethernet card.

Operating system: Linux, 32 bit, kernel version 2.6.22-14-generic

JDK: Sun JDK 1.6.0_03-b05

- Server:

  Dell Precision 470 workstation, dual cpu 2 X 2.8 GHz Intel Xeon 64 bit, 2GB RAM, 1GB ethernet card.

  Operating system: Linux, 64 bit, kernel version 2.6.22-14-generic

  JDK: Sun JDK 1.6.0_03-b05 64bit

- Network:

  1 GB ethernet

  Netgear GS105 Gigabit switch

8.3. Performance Results

All the tests were executed against these following messaging systems:

- 1. JBM 2.0 alpha. Out of the box config used.
- 2. JBM 1.4.0.SP3_CP02 - this is the production version used in JBoss Enterprise Application Platform 4.3 GA CP01. Out of the box config used. Used with MySQL 5.0.45 colocated with server and InnoDB tables with innodb_flush_log_at_trx_commit=1 (enable sync at tx commit)
- 3. ActiveMQ 5.1. This is the latest production version of ActiveMQ. Out of the box config was used with two changes: 1) syncOnWrite was set to true in the persistence config (otherwise ActiveMQ won't sync to disc on tx boundaries) 2) The upper queue memory limit was extended to 100MB otherwise the tests would block as queues became full.
- 4. JBoss MQ in JBoss AS 4.2.2.GA. This is JBoss' legacy JMS provider - now superceded by JBoss Messaging. Used with MySQL 5.0.45 colocated with server and Innodb tables with innodb_flush_log_at_trx_commit=1 (enable sync at tx commit)

Table 8.1. Performance Results (all results in messages/sec)

Test	JBM 2.0_Alpha	JBM 1.4.0.SP3_CP02	Apache ActiveMQ 5.1	JBoss MQ
Test 1. NP/Dups	18,836	13,401	12,963	881
Test 2. NP/AutoAck	14,143	3,889	9,813	672
Test 3. Persist/ Blocking/	1,372	442	312	622

Test	**JBM 2.0_Alpha**	**JBM 1.4.0.SP3_CP02**	**Apache ActiveMQ 5.1**	**JBoss MQ**
NonTX/DupsOk				
Test 4. Persist/Non Blocking/ NonTX/DupsOk	14,977	JBM 1.x does not support non blocking persistent message sends	ActiveMQ 5.1 does not support non blocking persistent message sends	JBoss MQ does not support non blocking persistent message sends
Test 5. Persist/ Blocking/ NonTX/AutoAck	1,265	421	524	637
Test 6. Persist/Non Blocking/ NonTX/AutoAck	12,056	JBM 1.x does not support non blocking persistent message sends	ActiveMQ 5.1 does not support non blocking persistent message sends	JBoss MQ does not support non blocking persistent message sends
Test 7. Large transactions. Persistent/ Transacted size=1000	9,607	1,355	1,576	805
Test 8. Small transactions. Persistent/ Transacted size=2	1,818	546	396	523

8.4. Performance conclusions.

JBoss Messaging 2.0 provides the highest throughput in all tests, and in particular shows ground breaking persistent performance for both transactional and non transactional operations.

JBoss Messaging 1.4.0.SP3_CP02 and ActiveMQ 5.1 are good all-round performers and show similar and respectable performance.

JBoss MQ has reasonable persistent performance, but is let down by poor non persistent results.

9

Troubleshooting

- Message on logs: AIO wasn't located on this platform

  Possible causes are:

  - Linux is not your platform

    Just ignore the message as NIO will be selected automatically or change the journal type to NIO on jbm-configuration.xml

  - The JBoss Messaging JNI wrapper is not on library.path

    Solution: Make sure you have libJBMLibAIO32.so or libJBMLibAIO64.so as provided on the download package.

  - libaio is not installed

    Make sure you have libaio installed at your Linux distribution. This could be done as yum install libaio1 on Fedora or apt-get install libaio1 on Debian. (Refer to your manual to how to upgrade or install packages)

  - libaio is very old

    We have been using and testing libaio 0.3.106 and Kernel 2.6. If you have older versions you might need to upgrade or change the journal type to NIO on jbm-configuration.xml

- Low response time even though the CPU and IO seems ok

  Possible causes are:

  - You are using short transactions in your system and you don't have TCPNoDelay set

    Make sure you have remoting-tcp-nodelay set to true on jbm-configuration.xml

  - The journal directory is a NFS

    You shouldn't use the journal over a NFS. Make sure you have direct access to the disk device.

  - You are using Asynchronous IO on Linux and you don't have an ext3 or ext2 file system

    If using AIO, make sure the journal folder is on an ext2 or ext3 file system.