# JBossESBPerformanceTuning

## Objective

The performance of JBossESB depends on multiple factors, ranging from environmental to programmatic (application-specific). The information presented here will focus on the environmental side, with particular attention paid to configuration of JBossESB as well as things to consider from an underlying application server, database and JVM perspective.

A general rule to follow, however, is to never depend solely on tuning guides, but instead to use them for points to consider. In the end, you will need to have specific performance goals, and take the appropriate, documented, reproducible steps to reach those goals. In most cases, this will mean employing java profiling tools, operating system, database and sometimes network monitoring tools, and the like.

http://community.jboss.org/wiki/MonitoringManagementAndProfilingTools

Specific operating system, database and other tuning information will **NOT** be provided here.

Finally, you should realize that as with any system, there is always a trade-off to be made between performance and reliability. The out-of-the-box configuration for JBossESB is tuned for maximum reliability and resiliency, which may have an adverse affect on performance in certain circumstances.

## JBossESB Configuration

# InVM Transport

InVM transport means that the Service can be invoked (via ServiceInvoker) from within the JVM with minimal overhead (i.e. without incurring any networking or message serialization overhead). Please note, due to the volatility aspect of the InVM queue, you may not be able to achieve all of the ACID semantics, particularly when used in conjunction with other transactional resources such as databases. For details, please refer to the "InVM Transport" section in Programmers Guide.

It also should be noted that there are pitfalls to using pass by reference semantics, especially if the message contains classes provided from different classloaders. Any reference held within the message will prevent those classloaders from being garbage collected until the message has been processed and discarded.

The code snippet below shows how to configure your service using InVm transport:

```
<service category="..." name="..." description="..." invmScope="GLOBAL">
   <listeners>
      ...
   </listeners>
   <actions>
      ...
   </actions>
</service>
```

# Max Threads for Message-Aware Listeners

You can change the maximum number of threads used by Message-Aware Listeners in your .esb's META-INF/jboss-esb.xml file. The default value is 1. The following example shows how to set the max threads to 100:

```
<service category="..." name="..." description="..." invmScope="...">
   <property name="maxThreads">100</property>
   <listeners>
      <...-listener name="..." busidref="..." is-gateway="true" maxThreads="100" />
   </listeners>
   <actions>
      ...
   </actions>
</service>
```

# Max Threads for JBoss Remoting Listener

JBoss Remoting also has its own maxThreads property.  The default value is 50.  The following example shows how to set its max threads to 100:

```
<service category="..." name="..." description="..." invmScope="...">
   <property name="maxThreads">100</property>
   <listeners>
      <jbr-listener name="..." busidref="..." is-gateway="true" maxThreads="100">
         <property name="jbr-maxThreads" value="100" />
      </jbr-listener>
   </listeners>
   <actions>
      ...
   </actions>
</service>
```

# HTTP Gateway vs. HTTP via JBoss Remoting

Using the new HTTPGateway can provide better performance than using HTTP via JBoss Remoting.  When using the HTTPGateway, the actual embedded Tomcat included in JBoss AS runs an instance of HTTPGatewayServlet for each unique declaration of the gateway. Then, any Tomcat performance tuning you are familiar with can be applied.  For example, the setting of maxThreads in the Connector element of server.xml:

```
<Connector maxThreads="250" ... />
```

For JBossESB deployments in JBoss AS 4.2.x, you will find server.xml here:

    <server_home>/server/<server_config>/deploy/jboss-web.deployer/server.xml

For JBossESB deployments in JBoss AS 5.1.x, you will find server.xml here:

    <server_home>/server/<server_config>/deploy/jbossweb.sar/server.xml

For more information on using the HTTPGateway, please see here:

http://community.jboss.org/wiki/HTTPGateway

## Tuning for JMS Listeners

The "org.jboss.soa.esb.jms.connectionPool" property in the "transports" properties section of <server_home>/server/<server_config/deploy/jbossesb.sar/jbossesb-properties.xml controls the consumption from the JMS server:

```
<properties name="transports" depends="core">
   <property name="org.jboss.soa.esb.jms.connectionPool" value="20"/>
</properties>
```

The JBossESB-Internal/DeadLetterService service's jms-listener can have its maxThreads property configured (defining the number of threads consuming messages from the queue) in <server_home>/server/<server_config>/deploy/jbossesb.esb/META-INF/jboss-esb.xml:

```
<service category="JBossESB-Internal" name="DeadLetterService" ...>
```

```
   <listeners>
      <jms-listener name="JMS-DLQListener" busidref="DeadMessageQueue" maxThreads="1" />
   </listeners>
   <actions>
      ...
   </actions>
</service>
```

## Message Filters

Message filters are used to dynamically augment ESB messages. (For example, adding transaction or security information when the message flows through the ESB.) Depending on the particular message filters that have been configured in your ESB, performance may be impacted. These filters are defined in the "filters" properties section of <server_home>/server/<server_config/deploy/jbossesb.sar/jbossesb-properties.xml:

```
<properties name="filters">
   <property name="org.jboss.soa.esb.filter.1 value="org.jboss.internal.soa.esb.message.filter.ABCFilter"/>
   <property name="org.jboss.soa.esb.filter.2 value="com.mycompany.XYZFilter"/>
   <property name="org.jboss.soa.esb.filter.3 value="..."/>
</properties>
```

For further information regarding message filters please refer to the "Meta-data and Filters" section in the Programmers Guide.

## CachingRegistryInterceptor

Recent changes in JBossESB make this change less impactful, but you can configure the ESB to cache registry results by enabling the CachingRegistryInterceptor as part of the "org.jboss.soa.esb.registry.interceptors" property of the "registery" properties section of <server_home>/server/<server_config/deploy/jbossesb.sar/jbossesb-properties.xml:

```
<properties name="registry">
   <property name="org.jboss.soa.esb.registry.interceptors" value="org.jboss.internal.soa.esb.services.registry.InVMRegis
</properties>
```

## Service Deployments

If you do not require the capabilities of certain ESB-provided services, you can simply remove the associated ESB archives (.esb extension).  For example, if you are not leveraging jBPM for service orchestration, you can remove <server_home>/server/<server_config/deploy/jbpm.esb.  This not only makes for a trimmer heap, but also will keep a jbpm-specific daemon thread (which looks in the databases for processes to execute) from running.

# Application Server Configuration

## Logging

Unnecessary logging can be an incredible performance-killer.  Edit <sever_home>/server/<server_config/conf/jboss-log4j.xml appropriately.

If you don't want ANY I/O done (unlikely but sometimes useful when doing certain performance tests), you can simply comment out the FILE appender:

```
<!--
<appender-ref ref="FILE"/>
-->
```

Even if you do the above, and no matter what, you will want to change limit the JBoss logging categories to suite your needs.  For example, if you don't **UNCOMMENT** this section:

```
<!-- Limit JBoss categories
-->
<category name="org.jboss">
   <priority value="INFO"/>
</category>
```

, the default is **DEBUG**!  What this means is that even if you're not doing any I/O, **ALL** the work to build up debug Strings will still be performed unnecessarily.

## Database

JBossESB on JBoss AS comes with the Hypersonic database by default.  You will **DEFINITELY** want to switch from this to an enterprise-grade database, like PostgreSQL or Oracle.  Information on how to do this is available here:

http://community.jboss.org/wiki/JBossESBSwitchDatabase

## Deployment Scanner

JBossESB relies upon the application server's service and application hot-deployment capabilities.  This is done via a deployment scanner, which by default is configured to run frequently (every 5 seconds).  Whereas this is useful for development, it is mostly likely too high for your production environment.

In the <server_home>/server/<server_config>/conf/jboss-service.xml file, you can change the ScanPeriod to a larger number.  For example, from every 5 seconds to every 60 seconds:

```
<attribute name="ScanPeriod">60000</attribute>
```

If you wanted to, you could also turn off scanning completely:

```
<attribute name="ScanEnabled">false</attribute>
```

# JVM

## Memory

You should change your JVM memory flags to acceptable values for your unique purposes, tuned based on performance testing and environmental factors.  The changes can be made in <server_home>/bin/run.conf.  Here is an **EXAMPLE** (*not* a one-size-fits-all suggestion):

```
JAVA_OPTS="-Xms1024m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -Djava.awt.headless=true -Dsun.rmi.dgc.cl
```

## Java 6

If you are manually deploying JBossESB into JBoss AS, and would like to leverage JVM performance improvements made between Java 5 (Sun's now EOL) and Java 6, certain jars are required to be available in <server_home>/lib/endorsed/.  They are:

- jaxb-api.jar
- jaxb-impl.jar
- jaxb-xjc.jar
- jbossws-native-jaxrpc.jar
- jbossws-native-jaxws-.jar
- jbossws-native-jaxws-ext.jar
- jbossws-native-saaj.jar

In JBoss AS 5.1.0, they can originally be found in client/.  If you are a subscriber and using the JBoss Enterprise SOA Platform 5.0, this step has already been done for you.

# And much, MUCH more...

The above is just a start.  Please visit the respective documentation for JBoss AS, Tomcat, JBoss Messaging, jBPM, etc. for related performance tuning information.  Obviously the applicability will be gauged depending on which components you leverage from JBossESB. For example, if you use FTP and File listeners in the ESB, but not JMS, you will probably not care too much about JBoss Messaging tuning.  The same goes for jBPM, the web tier, and the rest.