WebLogic JMS System Best Practices

Daniel Joray
Trivadis AG
Bern

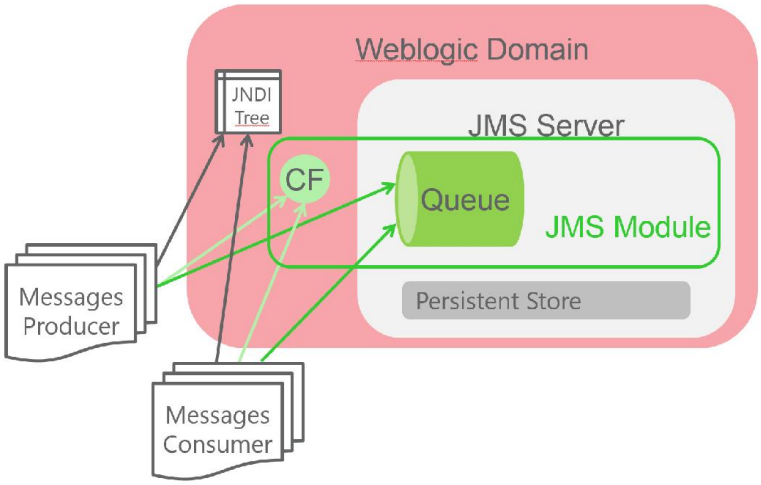
Keywords
Weblogic, JMS, Performance, J2EE

Introduction

In many J2EE project the Java Message Service JMS is for exchange of information of various application components used and in each projects the same questions about the configuration of JMS systems are asked. Questions like:

- Which components of JMS are best suited for my application?
- How to configure JMS for optimal performance?
- What happens if too many messages are being produced?
- And many more…

Weblogic JMS System



The major components of the WebLogic JMS Server architecture are:

- JMS servers that can host a defined set of modules and any associated persistent storage which reside on a WebLogic Server instance.
- Weblogic persistent storage (file store or JDBC-accessible) for storing persistent message data.
- JMS modules contains configuration resources (such as queues, topics, and connections factories).
- Client JMS applications that either produce messages to destinations or consume messages from destinations.
- JNDI (Java Naming and Directory Interface), which provides a resource lookup facility.

JMS servers
A JMS server is an environment-related configuration entity that acts as management container for JMS queue and topic resources defined within JMS modules that are targeted to specific that JMS server.
A JMS server's primary responsibility for its targeted destinations is to maintain information on what persistent store is used for any persistent messages that arrive on the destinations, and to maintain the states of durable subscribers created on the destinations.

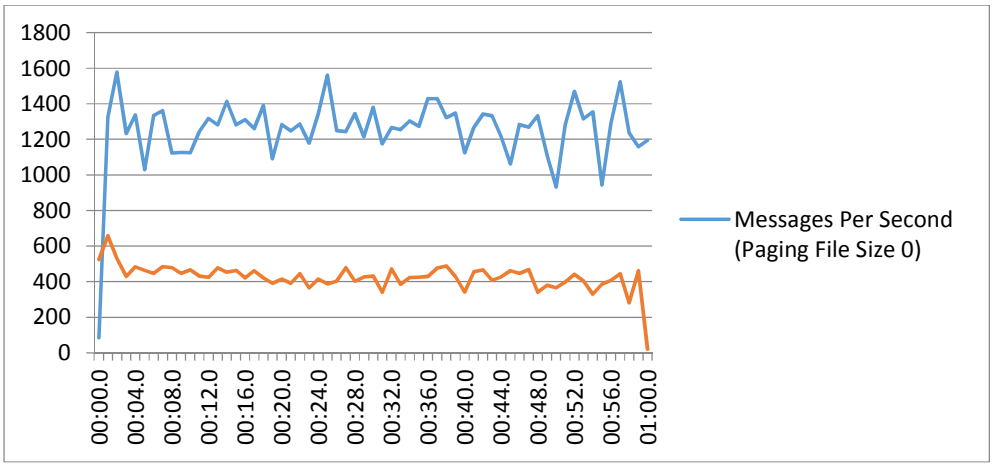
You can target a JMS server to either an independent Weblogic Server instance or to a migratable target server where it will be deployed.

JMS servers automatically attempt to free up virtual memory during peak message load periods. JMS message paging saves memory for persistent messages, as even persistent messages cache their data in memory.

The Message Buffer Size JMS server attribute specifies the amount of memory that will be used to store message bodies in memory before they are paged out to disk. The default value is approximately one-third of the maximum heap size for the JVM or 512Mb, whichever is larger.

Paged-out message does not free all the memory that it consumes, because the message header remains in memory for use with searching, sorting, and filtering

The Paging cost can be very expensive depending of the size of pagefile. A restart from the manage server will not resize the pagefile or the message file.



Persistent Stores
A persistent store provides a built-in, high-performance storage solution for Weblogic Server subsystems and services that require persistence,The persistent store supports persistence to a file-based store (File Store) or to a JDBC-enabled database (JDBC Store).

To create a File Store the path set in parameter Directory muss exist. For highest availability, use either a Storage Area Network (SAN) or a dual-ported SCSI disk.
With the parameter Synchronous Write Policy you can define how hard a file store will try to flush records to the disk. The available values are Direct-Write (default), Cache-Flush, and Disabled and are depending of or disk storage.

For subsystems (JMS Server) that share the same server instance, share one store between multiple subsystems rather than using a store per subsystem. Sharing a store is more efficient for the following reasons

- A single store batches concurrent requests into single I/Os which reduces overall disk usage.
- Transactions in which only one resource participates are lightweight one-phase transactions. Conversely, transactions in which multiple stores participate become heavier weight two-phase transactions.

To create a JDBC Store you need a Datasource. You cannot specify a JDBC data source that is configured to support global (XA) transactions. The specified JDBC data source must use a non-XA JDBC driver. You cannot enable Logging Last Resource or Emulate Two-Phase Commit in the data source
This limitation does not remove the XA capabilities of layered subsystems that use JDBC stores Weblogic JMS is fully XA-capable regardless of whether it uses a file store or any JDBC store

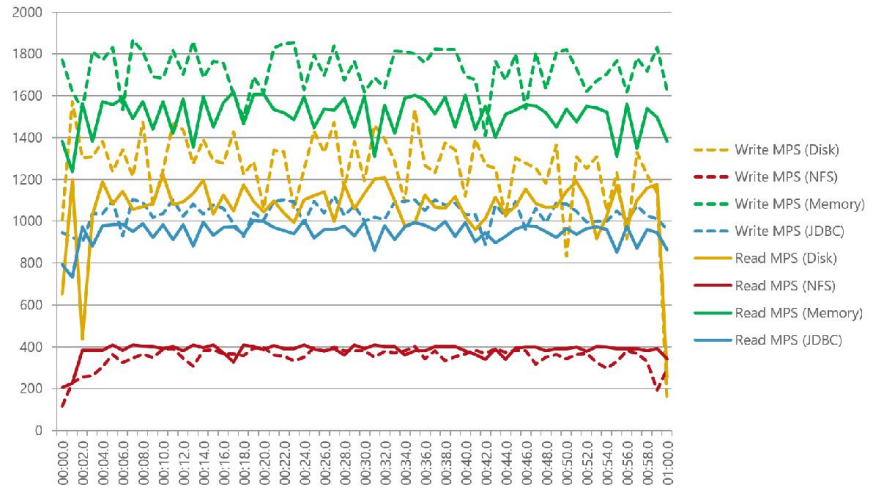
Weblogic will automatically attempt to create the required table if

- The table is not already present
- The data source credentials have “Create” rights
- The data source vendor type is not “Other”

A matching DDL file is found for the data source type and you can chose the prefix from the table [prefixWLStore]. The Store DDL files for standard vendors are found within the archive
`<WL_HOME>/modules/com.bea.core.store.jdbc_xxx.jar`

For Oracle databases, the default DDL used is oracle.ddl, this DLL create a table who the record is save in a LONG RAW Colum. You can change the default DDL whit Create Table from DDL File field. oracle_blob.ddl is available that uses a BLOB data type instead of the default LONG RAW type.

Not all stores have the same performance this is depending from your system or your configuration



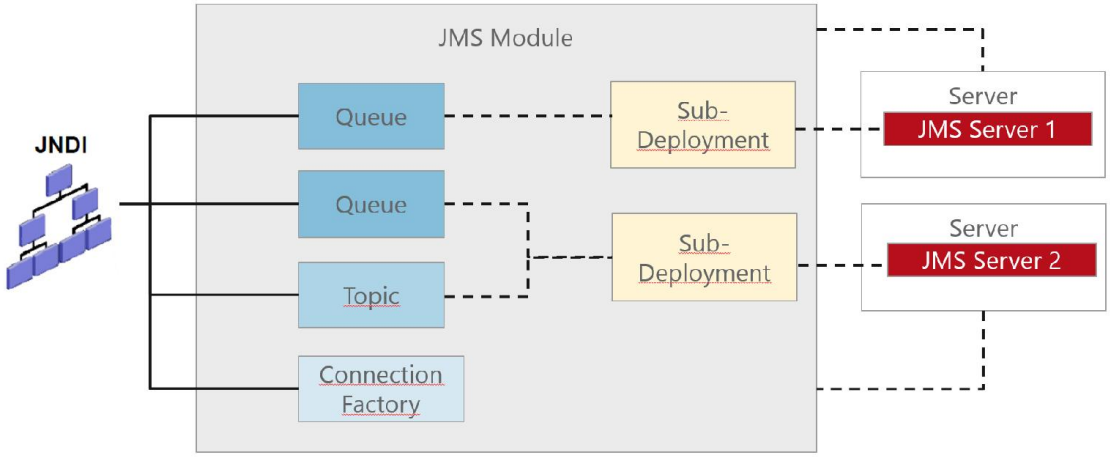
In this case, nfs storage is slower as local storage and the DB storage but this is not always the case.

JMS modules
JMS modules are application-related definitions that are independent of the domain environment. The JMS modules are globally available for targeting to servers and clusters configured in the domain, and therefore are available to all applications deployed on the same targets and to client applications.

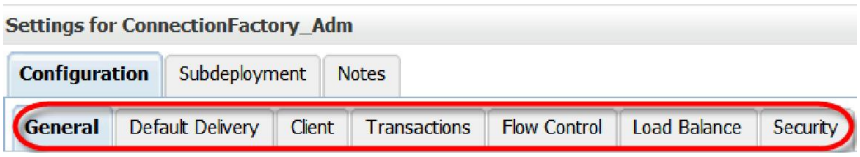
The following configuration resources are defined as part of a system module or an application module

- Connection factories
- Queue and topic destinations
- Templates
- Destination keys
- Quota
- Distributed destinations (Queue or Topic)
- Foreign servers
- JMS store-and-forward (SAF) configuration items
-

Not all JMS Resource in a JMS Module have the same target. In this case you can use Sub-Deployment to define a specific physical target to a JMS Resource



JMS Resources
A Connection Factory encapsulates connection configuration information, and enables JMS applications to create a Connection. You can use the preconfigured default connection factories provided by Weblogic JMS, or you can configure one or more connection factories to create connections with predefined attributes that suit your application. The Connection Factory allow you to configure parameters like timeout, delivery mode, transaction management and load balancing management.



Destination Key defines a unique sort order that destinations can apply to arriving messages. As messages arrive on a specific destination, by default they are sorted in FIFO (first-in, first-out) order, which sorts ascending based on each message's unique JMSMessageID. You can use a destination key to configure a different sorting scheme for a destination, such as LIFO (last-in, first-out).

Quota controls the allotment of system resources available to destinations like Bytes Maximum, Messages Maximum and Policy**.**

Queue defines a point-to-point destination type, which are used for asynchronous peer communications. A message delivered to a queue is distributed to only one consumer

Topic defines a publish/subscribe destination type, which are used for asynchronous peer communications. A message delivered to a topic is distributed to all topic consumers.

Ditributed Queue defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS client. Two types of distributed queues are available. **Uniform Distributed Queue**, when queue members are created uniformly from a common configuration and **Distributed Queue** when queue members are created and weighted individually to fine tune performance

New Delete Showing 1 to 5 of 5 Previous | Next

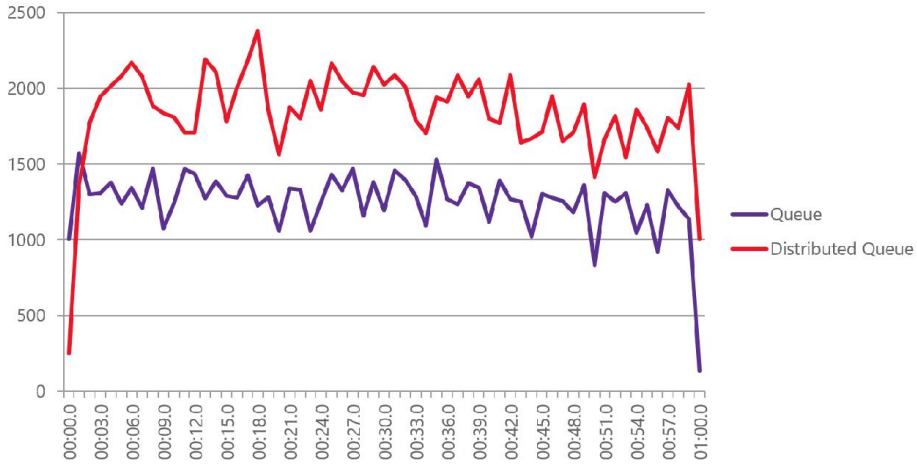
Name	Type	JNDI Name	Subdeployment	Targets
cf_jmsCluster	Connection Factory	jms/cf_jmsCluster	Default Targetting	mycluster
MyDistributedQueue	Distributed Queue	jms/MyDistributedQueue	N/A	N/A
MyUniformDistributedQueue	Uniform Distributed Queue	jms/MyUniformDistributedQueue	MyUniformDistributedQueue	JMSServer_srv2, JMSServer_srv3
Queue-srv2	Queue		Queue-srv2	JMSServer_srv2
Queue-srv3	Queue		Queue-srv3	JMSServer_srv3

Name	Queue	Weight
Queue-srv2	Queue-srv2	1
Queue-srv3	Queue-srv3	1

Ditributed Topic defines a set of topics that are distributed on multiple JMS servers, but which are accessible as a single, logical topic to JMS clients.

JMS high availability
Each JMS server is deployed on exact one Weblogic Server instance and handles requests for a set of destinations. During the configuration phase, the system administrator defines load balancing by specifying targets for JMS servers. If your JMS resource should be high available, you have to migrate your JMS Server. Weblogic support whole server migration and service migration (automatic and manual)

JMS loadbalancing
Distribution of application load across multiple JMS servers through connection factories, thus reducing the load on any single JMS server and enabling session concentration by routing connections to specific servers.



Best practices

To choice the right configuration, you need to find what is important for your Application

- Producer insertion
- Consumer reliability
- Throughput
- Message Order
- Persistence

And you should answer those questions for each JMS resource

To tune your JMS system you need to:

- Monitor CPU, disk I/O, memory, network interfaces from all involved systems
- At the beginning of each test, stop the Weblogic systems and clean-up or remove persistent store, paging files, truncate store tables
- Use a load generator (Jmeter, LoadUi, Grinder, Java function)
- Document each test case, at the beginning, use only short test case, when you have reached your targets with short test, you can make long-term test
- Create a custom store on each Weblogic server, which will host a JMS server.
- It is recommended to always target to migratable targets when available
- Configure message count quotas on each JMS server. There is no default quota, so configuring one helps protect against out-of-memory conditions.
- Configure Redelivery Limit, the number of redelivery tries a message can have before it is moved to the error
- Configure Error Destination, name of the target error destination for messages that have expired or reached their redelivery limit. If no error destination is configured, then such messages are simply dropped.
- Although JMS paging is enabled by default, verify that the default behavior is valid for your environment.
- Specify a Message Paging Directory. The default value is $MW_HOME\user_projects\domains\domainname\servers\servername\tmp
- Message Paging Directory must not be placed on the same disk as persistent store.

- Tuning the Message Buffer Size Option, to specify the amount of memory that will be used to store message bodies in memory before they are paged out to disk.
- Configure the distributed queue to enable forwarding. Distributed queue forwarding automatically internally forwards messages that have been idled on a member destination without consumers to a member that has consumers
- It is highly recommended to always configure message count quotas. Quotas help prevent large message backlogs from causing out-of-memory errors, and Weblogic JMS does not set quotas by default
- Configure Messages Expiration Policies
- Configure Flow Control
- Configure One-Way Message Send and Windows Size
- *One-way message send* can greatly improve the performance of applications that are bottle-necked by senders but do so at the risk of introducing a lower quality-of-service
- When active, the associated producers can send messages without internally waiting for a response from the target destination's host JMS server

Conclusion

The tuning task of Weblogic JMS System is complex, in this task You need to verify each component of the system, like JMS Servers, the persistent stores, connection factories, and the JMS component (queue, topic, distribute queue, etc.)

It is not possible to predict the “JMS performances” of a system without testing and you should never see a JMS Resource like a container. The physical implementation can impact the application. Not only the performances but the workflow of application.
You can only choose the right JMS resource if you know the application.

Contact:
Daniel Joray
Trivadis AG
Weltpoststrasse 5
CH-3015 Bern

Telefon: +41 (0) 58 459 50 26
E-Mail Daniel.Joray@trivadis.com
Internet: www.trivadis.com