IBM Maximo Asset Management Solutions Version 7.6.x

Best Practices for System Performance

Document version 1.0

CONTENTS

LIST OF FIGURES

LIST OF TABLES

REVISION HISTORY

Date	Version	Comments
December 2015	1.0	Initial Version for Release 7.6.x
October 2016	1.1	Updated Anywhere section to indicate that the latest release of MobileFirst supports compression by default. Added missing link to Linux kernel settings for Oracle.

1 Introduction

This paper provides best practices guidelines to improve the performance of IBM® Maximo® Asset Management version 7.6.x and related products, including multi-tenant support. This paper will refer to these products collectively as Maximo.

While these guidelines provide optimal performance in the lab test environment, your implementation may require different settings. Use the settings in this paper as guidelines or as a starting point, and then monitor and tune to your specific environment.

Maximo has a long and successful history in the world marketplace. Over the years, Maximo has incorporated many new features and grown in complexity, and Maximo integrates with other complex software systems.

Small, medium, and large organizations implement Maximo in increasingly complex ways. For many customers, Maximo is now a global, enterprise-wide implementation that is in use by thousands of users.

The larger and more complex the deployment of Maximo is, the more challenging it is for you to keep Maximo performing well for your users. Because some of the greatest challenges are faced by those who deploy these products across large, global enterprises, this document has a special focus on improving performance in advanced enterprise configurations.

Note: The guidelines in this document are based on results from non-virtualized environments. Deployments in virtualized environments, such as VMware®, will likely not see the same performance benefits from the recommendations made in this document. Using shared resources in virtualized environments is not recommended for optimal performance. Use dedicated CPU and memory resources instead. See Appendix B: VMware Considerations for more information.

Always check the Service Management Connect Asset Management Group Files section to make sure that you have the latest version of this document. You can provide direct feedback to this document by adding a comment using the *Add a Comment…* link located in the document description.

1.1 Best Practices as a cooperative effort

This paper is the result of a cooperative effort between IBM employees and a core group of enterprise customers. This paper was developed both in response to customers and in concert with them. The cooperative effort is ongoing.

System performance information that has been developed both within IBM and in actual deployments of Maximo is presented here. The goal of this paper is to provide you with information that can help you to improve the experience of system performance for your users.

1.2 Quality Assurance and testing

With each release of Maximo, IBM provides new features for your end users. We also strive to increase performance, reliability, and speed with each release. One way that we work to enhance performance is to provide tools and features that are specifically designed to improve system performance.

The IBM Quality Assurance team does extensive testing of Maximo. The Quality Assurance team tests during the development cycle, and again, before the product is released. Testing continues after Maximo is released.

Application and system functionality testing is performed, as is system performance testing. The Quality Assurance team tries to emulate many of the kinds of system setups that you

might use. However, a laboratory-testing environment can never fully anticipate all of the scenarios that the product is put through in your environments.

The testing environment undergoes constant review to determine how best it can be scaled to meet the demands of large enterprise deployments. Customer involvement and feedback is vital to improving our quality assurance testing.

1.3 Factors in system performance

System performance depends on more than the applications and the database. The network architecture affects performance. Application server configuration can hurt or improve performance. The way that you deploy Maximo across servers affects the way the products perform. Many other factors come into play in providing the end-user experience of system performance.

Subsequent sections in this paper address the following topics:

- System architecture setup
- Application server configuration
- Scheduled tasks (cron tasks)
- Reporting
- Integration with other systems using the integration framework
- Network issues
- Bandwidth
- Load balancing
- Database tuning
- SQL tuning
- Client workstation configuration
- Considerations for Maximo Anywhere
- Miscellaneous performance improvement tips
- Troubleshooting

1.4 Maximo infrastructure

Maximo can be seen as sitting at the apex of a pyramid that consists of network, hardware, and software services.

The pyramid comprises the infrastructure within which Maximo operates, as follows:

- The entire system infrastructure relies on the network architecture.
- Hardware servers operate within the network.
- Software services such as application servers, report servers, and database servers, operate on the hardware servers.
- Maximo runs within the sphere of the software services.

For Maximo to perform well, the entire system must be sufficiently robust and well-tuned.

The figure that follows illustrates the interdependence of Maximo and other elements that make up the entire system infrastructure.

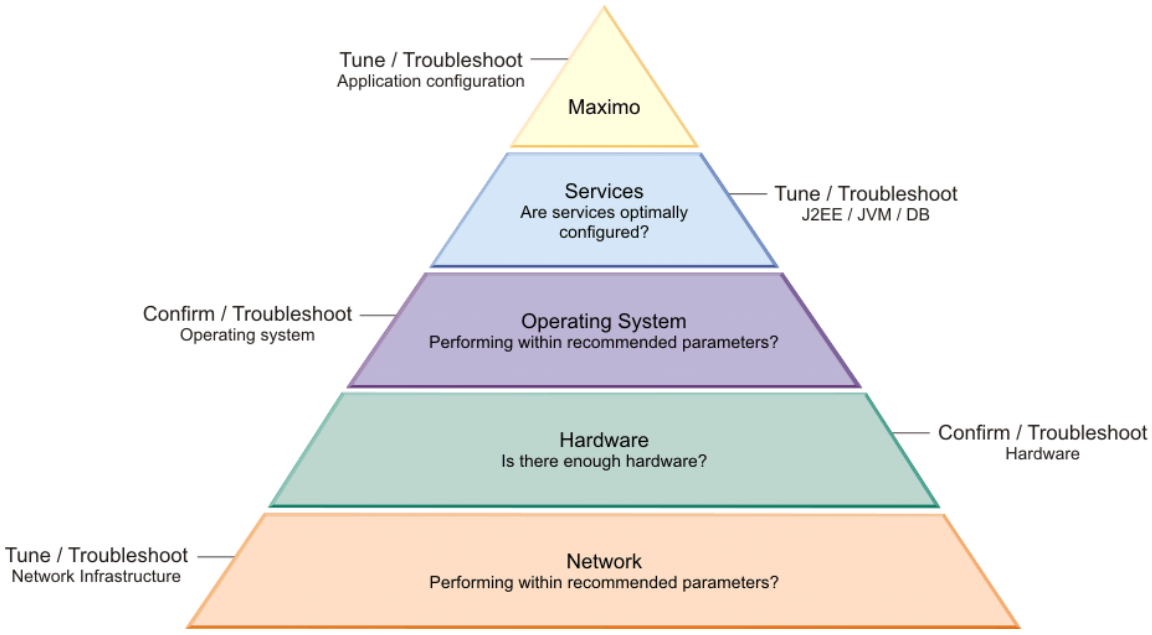


Figure 1: Maximo infrastructure pyramid

Improving system performance generally requires attention to all elements of the infrastructure. Your databases can be tuned perfectly, but if the network is slow, you may still have system performance issues. You may have a fast local area network, but if client workstations lack memory and processing power, users experience poor performance.

Customers report that there is normally more than one issue to address. Improving performance involves adjusting many things. Together, these changes can combine to yield significant performance improvement.

Making improvements in one or two areas can boost performance. For example, upgrading end-user workstations can often greatly improve the application experience, even if other problems exist in the overall deployed environment. As you continue to improve different areas of the environment, system performance improves incrementally.

2 System architecture and application server configuration

This section addresses ways to maximize system performance by configuring your system architecture to account for your resource needs.

2.1 Scalability factors

Many factors can affect the scalability of Maximo.

The type of hardware used can determine how many active users per JVM you will be able to achieve.

The average workload that is applied to the system can also have a large impact on the system. For example, you can support more users per JVM if those users are producing five transactions per hour as opposed to ten transactions per hour.

In addition, heavily customizing Maximo can result in fewer users per JVM due to the overhead in processing custom code.

Lastly, the architecture used in deploying Maximo has a significant impact on the number of active users per JVM. Some clients achieve only 35 users per JVM when all transactions (both front-end and back-end) are running in the JVMs, as described in the Basic system configuration section. Creating separate JVMs for user interface transactions allows for more active users per JVM, as described in the Advanced system configuration section. In an advanced system configuration, clients report the ability to support 50 to 75 users per user interface JVM.

2.2 Basic system configuration

A basic configuration consists of Maximo running on a single application server. The application server connects to a single instance of the database that is available on a database server. The application server can also connect to a report server.

If the integration framework is configured for deployment, then you must set up additional messaging queues. Maximo use additional queues to send data to external systems, and to receive data from external systems.

While your test environment must closely simulate your eventual production environment, the basic configuration is appropriate as a development or small production configuration, shown in the figure that follows:

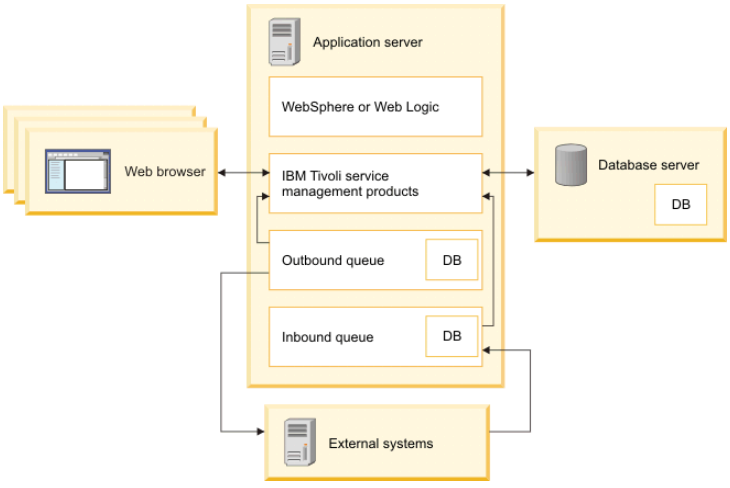


Figure 2: Basic system configuration

2.3 Advanced system configuration

Deploying Maximo using horizontal and/or vertical clustering is an effective way to distribute user load and improve the end-user experience of system performance. When you use clustering, implement the cluster configuration so that the system can scale well.

You can set up multiple clusters, each consisting of multiple JVMs. The number of JVMs depends in part on the overall hardware and software limitations of the environment, as well as on the expected number of concurrent users. You can tune the setup based on traffic and performance numbers.

A system of separate clusters, with multiple JVMs as cluster members, provides advantages in system administration. For your system to perform well, isolate the user- interactive applications to one cluster. Isolate the other processes, such as queue, cron task, and reporting processes, to other clusters. Hardware load balancers, such as F5 or Cisco, can be used in enterprise implementations to balance the load between multiple web servers.

The following figure illustrates an example of this type of complex enterprise architecture for Maximo Asset Management:

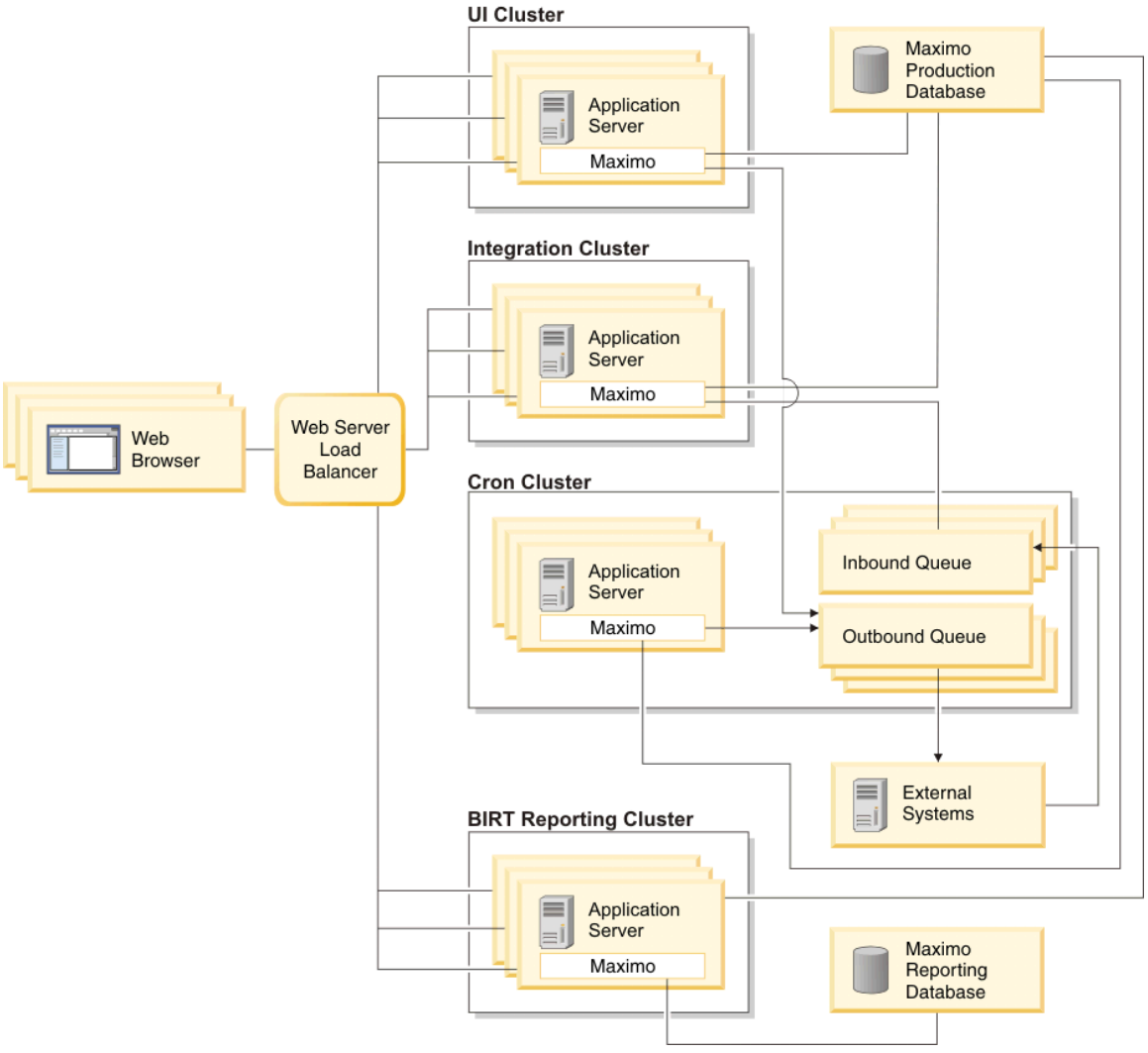


Figure 3: Maximo and SCCD advanced system configuration

When Maximo Anywhere is included in your enterprise architecture, deploying IBM MobileFirst Server and Maximo Mobile OSLC Server should use similar horizontal and/or vertical clustering to distribute user load and improve the end-user experience of system performance. The following figure illustrates an example of this type of complex enterprise architecture:

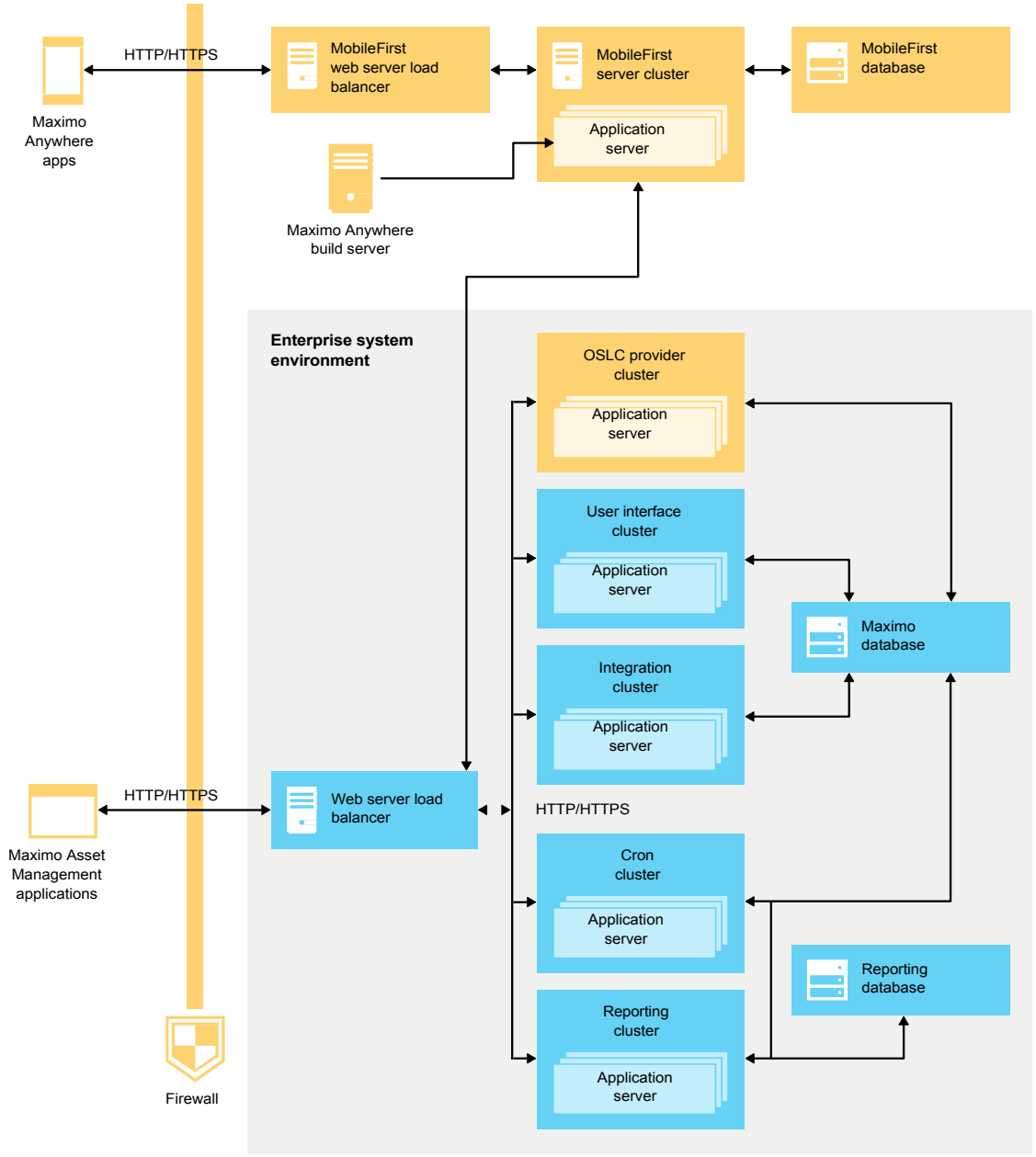


Figure 4: Maximo Anywhere advanced system configuration

For more information on clustering Maximo, see Cluster and Integration Framework Configurations Links.

3 IBM WebSphere Application Server tuning

This section discusses settings that can optimize IBM WebSphere® Application Server (WAS) for Maximo. Maximo, version 7.6 now requires WebSphere Application Server version 8.5 or higher with JDK version 7.

For further information on tuning WAS, see WebSphere Application Server Performance It is important to tune the application server at the operating system level. More information on tuning operating system parameters for WAS can be found in the Server operating system configuration section of this document.

While the settings below provide optimal performance in test environments, your environment may require different settings. You can use the settings below as a guideline or as a starting point, and then customize the settings to your specific environment requirements.

3.1 JVM tuning guidelines

You can tune the JVM settings in WAS using the WAS Administrative Console. After logging into the console, select *Servers → Server Types → WebSphere application servers → <server_name> → Process definition → Java Virtual Machine*.

For additional information on tuning the JVM, see Tuning the IBM virtual machine for Java in the WAS documentation.

3.1.1 Initial / maximum heap size values

You can use the following heap size values to start tuning:
Initial: 6144m / Maximum: 6144m (Recommended)
Initial: 4096m / Maximum: 4096m (Minimum)

Set the initial and maximum heap sizes to the same values for maximum performance.

Note: When planning for overall system memory consumption, include additional available RAM for the Operating System to use. An additional 4GB of RAM is a good target. Additional available RAM also will be needed for the Java process and stack. A general rule of thumb is to include an additional 30% - 40% of the allocated heap size to account for this overhead.

JVM heap size parameters directly influence garbage collection behavior. Increasing the heap size permits more objects to be created before it triggers a garbage collection. However, a larger heap size means a larger amount of time is needed to find and process objects that need to be garbage collected. Consequently, JVM heap size tuning often involves a balance between the interval between garbage collections and the pause time needed to perform the garbage collection.

To tune the JVM heap size, enable verbose GC. When enabled, the JVM prints out useful information at each garbage collection, such as the amount of free and used bytes in the heap, the interval between garbage collections, and the pause time. This information is logged to the native_stderr.log file that can then be used to analyze the heap usage. Once tuning is complete, disable verbose GC and delete the native_stderr.log file, since the log file can grow quite large while verbose GC is enabled. See the Displaying garbage collection statistics on the server section of this document for information on enabling verbose GC.

In addition, Maximo can log heap memory utilization in the SystemOut.log file. You can analyze the SystemOut.log file to ensure that no out of memory conditions are occurring. For information on setting up logging to obtain this data, see Using debug properties to monitor and troubleshoot performance.

Using the guidelines above, you can analyze heap memory usage and garbage collection data to determine the appropriate value for a particular deployment that minimizes garbage collection times while providing enough memory to the application. Your particular deployment may require different settings to produce optimal performance.

3.1.2 Generic JVM arguments

Gencon garbage collection policy

-Xgcpolicy:gencon

The gencon garbage collection policy places objects in separate areas of the heap based on their lifetime. The heap is split into a nursery, where objects are created, and a tenured area, where objects are promoted after having survived a certain number of garbage collections. The nursery can then be garbage collected frequently without the overhead of collecting the entire heap, which keeps garbage collection pauses to a minimum.

Since Maximo creates many, short-lived objects, use the gencon policy.

Nursery size

25% of maximum heap size:

-Xmn1536m (assuming a 6G heap size)
-Xmn1024m (assuming a 4G heap size)

As mentioned above, when using the gencon garbage collection policy, the heap is split into the nursery and the tenured space. Setting the size of the nursery when using this policy can be very important to optimize performance.

Maximo recommends the nursery size be set to 25% of the heap size. If you further tune your heap sizes based on the verbose garbage collection and available heap memory, remember to adjust the nursery size accordingly.

Garbage collector threads

-Xgcthreads4

This option sets the number of threads that the Garbage Collector uses for parallel operations.

Java RMI settings

-Dsun.rmi.dgc.ackTimeout=10000

The value of this property represents the length of time (in milliseconds) that the server- side Java RMI runtime strongly refers to a remote object (or a reference to a remote object). This time is returned from the current virtual machine as part of the result of a remote method call. The default value is 300000 (five minutes). Setting the value too low increases the risk of a remote object being prematurely garbage collected, when the only known reference to the remote object is the one in transit as part of the remote method call result. Setting the value too high prevents the remote objects being garbage collected in time when they are not actually being referenced, which can cause a larger memory footprint on the server.

Since Maximo uses RMI, and RMI allocates a large quantity of short-lived remote objects, setting *sun.rmi.dgc.ackTimeout* to the default of five minutes prevents garbage collection from collecting the objects on the server fast enough, which can cause an out of memory problem. To prevent an out of memory problem, set this value to 10000 (10 seconds).

Disable explicit garbage collection

-Xdisableexplicitgc (Recommended)

Disabling explicit garbage collection disables any System.gc() calls from triggering garbage collections. For optimal performance, disable explicit garbage collection.

Note: Some customers with workloads requiring large, frequent memory requests have reported out of memory issues when disabling explicit garbage collection. In those rare cases, make the following changes:

- Remove the **–Xdisableexplicitgc** flag from the generic arguments
- Add the following flags to the generic arguments:
  - -Dsun.rmi.dgc.server.gcInterval=900000
  - -Dsun.rmi.dgc.client.gcInterval=900000

Prefer IPv4 Stack

-Djava.net.preferIPv4Stack=true

If IPv6 is available on the operating system the underlying native socket will be, by default, an IPv6 socket which lets applications connect to, and accept connections from, both IPv4 and IPv6 hosts. For performance reasons, Maximo recommends to set this property to true. The implication is that it will not be possible for the application to communicate with IPv6 only hosts. Therefore, if you have IPv6 only hosts in your environment, then you should omit this property to use the default value of false.

Code Cache

Code cache: **-Xcodecache32m**

This option sets the size of each block of memory that is allocated to store the native code of compiled Java™ methods

Permanent Generation

-XX:PermSize=320m
-XX:MaxPermSize=320m

The permanent generation is used to hold reflective data of the VM itself such as class objects and method objects. These reflective objects are allocated directly into the permanent generation, and it is sized independently from the other generations. Some customers with issues running reports resolve this issue by adding theses generic JVM arguments.

3.1.3 Thread pool tuning recommendations

You can tune the JVM settings in WAS using the WAS Administrative Console. After logging into the console, select *Servers → Server Types → WebSphere application servers → <server_name> → Thread pools*.

Thread pools enable components of the server to reuse threads, eliminating the need to create new threads at run time to service each new request.

The *Default* thread pool is used when requests come in for a message driven bean or if a particular transport chain has not been defined to a specific thread pool. The *WebContainer* thread pool is used when requests come in over HTTP.

The *minimum size* specifies the minimum number of threads to maintain in the pool.

The *maximum size* specifies the maximum number of threads to maintain in the pool.

The *thread inactivity timeout* specifies the amount of inactivity (in milliseconds) that can elapse before a thread is reclaimed.

Allow thread allocation beyond maximum thread size specifies whether the number of threads can increase beyond the maximum size configured for the thread pool. Using this setting can

ensure that you always have enough threads to handle requests; however, each thread will consume memory, so using this setting could result in heap memory issues, especially if a problem occurs that causes thread creation to spiral upward.

The following options and values can be configured for thread pools:

Default thread pool:

- Minimum: 20
- Maximum: 50
- Timeout: 30000

Webcontainer thread pool:

- Minimum: 120
- Maximum: 150
- Timeout: 60000

You can use Tivoli Performance Viewer (TPV) to monitor thread usage to ensure you have set the values appropriately. If TPV shows the Percent Maxed metric remaining consistently in the double digits, consider increasing the maximum size.

For additional information on configuring thread pool settings, see Thread pool settings in the WAS documentation.

4 IBM HTTP Server and Plugin tuning

You can use the settings below to optimize IBM HTTP Server Version 8.5 or later for Maximo. While these settings provide optimal performance in the test environment, your environment may require different settings. You can use the settings as a guideline or as a starting point and then monitor and tune the settings to your specific environment.

See IBM HTTP Server Performance Tuning for additional information on how to tune IBM HTTP Server.

TimeOut

Value: **900**

The TimeOut directive sets the amount of time the server waits for certain events before failing a request.

KeepAliveTimeOut

High Network Bandwidth: **10**
Low Network Bandwidth: **60**

The KeepAliveTimeout directive sets the amount of time the server waits for subsequent requests on a persistent connection. Set this value to 60 for those environments that may have high network latency, such as when users are located in areas that are geographically different from the location of the servers.

MaxKeepAliveRequests

Value: **0**

The MaxKeepAliveRequests directive sets the number of requests allowed on a persistent connection. Setting this value to zero allows unlimited requests.

MaxRequestsPerChild

Value: **0**

The MaxRequestsPerChild directive sets the limit on the number of requests that an individual child server handles during its life. Once this limit is reached, the child process dies. Setting this value to zero ensures that the process never expires.

ThreadLimit

Value for Windows environments: **2400**

The ThreadLimit directive sets the upper limit on the configurable number of threads per child process.

This setting is optimized and tested for the Windows® environment. While the settings is also tested on an AIX environment, it is not optimized for UNIX®/Linux®.

ThreadsPerChild

Value for Windows environments: **2400**

The ThreadsPerChild directive sets the number of threads created by each child process.

In a Windows environment where there is only one child process, this value must be set to a high enough value to handle the entire load of the server. In UNIX/Linux environments, threads and processes are allocated differently than in Windows. You would not normally set the ThreadLimit to such a high value.

The ThreadsPerChild and ThreadLimit directives should be set to the same value.

4.1 HTTP compression

HTTP compression is a capability that can be built into web servers, such as IBM HTTP Server and Apache HTTP Server 2.x and above, and web browsers to better use available bandwidth, and to provide faster transmission speeds.

See the section on HTTP compression of this document for more information on this topic.

4.2 Load balancing

Load balancing is the distribution of the task load across multiple instances of an application. User load comes from users who are logged in to the system and use the interface to perform tasks. Non-user load comes from items such as scheduled jobs (cron tasks) and integration framework incoming transactions.

You can distribute user load and non-user load across different application servers or clusters. The IBM HTTP Server for WebSphere Application Server plug-in acts as an “agent” that redirects HTTP requests from the web server to the application server using the HTTP protocol.

You can improve the performance of IBM HTTP Server (with the WebSphere web server plug-in) by modifying the load balancing option that the plug-in uses when sending requests to the application servers associated with that web server.

The default load balance option, Round Robin, provides an even distribution of work across cluster members. However, the Random option yields a more even distribution of work across the cluster. You must test the options to determine which option works better for your deployment of Maximo. You can configure Maximo to log the number of users per JVM to assist in determining which option provides the best load balancing. For information on setting up logging to obtain this information, see Using debug properties to monitor and troubleshoot performance.

5 Database server tuning and maintenance

The database is central to the functionality of Maximo. This database stores all data that is collected and calculated by the applications. This database also stores metadata for configuring and maintaining the environment.

The database server processes all transactions from the applications. The integrated reporting function accesses the data in the database to generate documents, such as work orders and purchase orders. Reporting also generates resource-intensive management reports.

Because all functionality is based on database performance, the database can be a key focus for performance tuning.

5.1 IBM DB2 database tuning

This section discusses best practices for tuning your DB2 version 10.1 or above database for Maximo.

For additional information on configuring DB2 for Maximo, see Manually configuring DB2 10.5. For assistance in performing DB2 administration tasks, see the DB2 Knowledge Center.

5.1.1 DB2 registry settings

For convenience, IBM DB2 provides an environment variable that can be used to optimize DB2 instances for use with a variety of products, including Maximo. Set DB2_WORKLOAD=MAXIMO. This accomplishes the same thing as if you set the following registry keys individually, using db2set commands:

```
db2set DB2_SKIPINSERTED=ON
db2set DB2_INLIST_TO_NLJN=YES
db2set DB2_MINIMIZE_LISTPREFETCH=YES
db2set DB2_EVALUNCOMMITTED=YES
db2set DB2_SKIPDELETED=ON
```

After setting DB2_WORKLOAD=MAXIMO, stop and start the database to apply the changes.

In addition to the DB2_WORKLOAD=MAXIMO settings, use the following DB2 registry setting:

```
db2set DB2_FMP_COMM_HEAPSZ=65536
```

5.1.2 DB2 database configuration settings

For optimal performance, use the following DB2 database configuration settings:

```
db2 update db cfg for <dbname> using CHNGPGS_THRESH 40
db2 update db cfg for <dbname> using DFT_QUERYOPT 5
db2 update db cfg for <dbname> using LOGBUFSZ 1024
db2 update db cfg for <dbname> using LOGFILSIZ 8192
db2 update db cfg for <dbname> using LOGPRIMARY 20
db2 update db cfg for <dbname> using LOGSECOND 100
db2 update db cfg for <dbname> using LOCKLIST AUTOMATIC
db2 update db cfg for <dbname> using LOCKTIMEOUT 300
db2 update db cfg for <dbname> using MAXFILOP 61440 (UNIX / Linux)
db2 update db cfg for <dbname> using MAXFILOP 65535 (Windows)
db2 update db cfg for <dbname> using NUM_IOCLEANERS AUTOMATIC
db2 update db cfg for <dbname> using NUM_IOSERVERS AUTOMATIC
db2 update db cfg for <dbname> using SOFTMAX 1000
```

```
db2 update db cfg for <dbname> using STMTHEAP 20000
db2 update db cfg for <dbname> using CUR_COMMIT ON
db2 update db cfg for <dbname> using AUTO_REVAL DEFERRED
db2 update db cfg for <dbname> using DEC_TO_CHAR_FMT NEW
db2 update db cfg for <dbname> using STMT_CONC LITERALS
```

While the recommended setting for SOFTMAX has a positive impact on runtime performance, crash recovery time will increase. If you use HADR, this would not be cause for concern. In addition, if you use online backup, there may be a spike in I/O while data is being flushed to disk when the backup starts.

Use the following database configuration changes to take advantage of the self-tuning aspects of the database, and to reduce the overall management cost for the installation. You must have enough system resources for automatic tuning to work properly. If you are constrained on system resources, tune the settings manually:

```
db2 update db cfg for <dbname> using DATABASE_MEMORY AUTOMATIC
db2 update db cfg for <dbname> using PCKCACHESZ AUTOMATIC
db2 update db cfg for <dbname> using DBHEAP AUTOMATIC
db2 update db cfg for <dbname> using STAT_HEAP_SZ AUTOMATIC
```

In general, Maximo does not recommend turning on automatic maintenance settings, such as AUTO_MAINT, AUTO_TBL_MAINT, AUTO_RUNSTATS, and AUTO_STMT_STATS. Instead, schedule database maintenance activities during regular maintenance windows to avoid adversely affecting end user performance.

5.1.3 DB2 database manager configuration settings

For optimal performance, use the following DB2 database manager settings:

```
db2 update dbm cfg using AGENT_STACK_SZ 1024 (UNIX / Linux)
db2 update dbm cfg using AGENT_STACK_SZ 1000 (Windows)
db2 update dbm cfg using RQRIOBLK 65535
db2 update dbm cfg using HEALTH_MON OFF
```

Set the following database manager configuration parameter to avoid potentially encountering a DB2 heap exception (com.ibm.db2.jcc.c.SqlException: Not enough storage is available in the "MON_HEAP_SZ" heap to process the statement):

```
db2 update dbm cfg using MON_HEAP_SZ AUTOMATIC
```

Set the following parameters when using fenced processes:

```
db2 update dbm cfg using KEEPFENCED YES
db2 update dbm cfg using FENCED_POOL=200
```

5.1.4 DB2 Statement Concentrator Considerations

REOPT ONCE bind option

Using the concentrator (STMT_CONC=LITERALS) changes statement texts, and this causes less information to be available to the optimizer, which can cause deficiencies in plans. To help the query optimizer select an efficient access plan you must also specify the REOPT(ONCE) bind option when you run queries. When the REOPT(ONCE) bind option is used, the query optimizer selects the access plan the first time that the query is run. Each subsequent time that the query is run, the access plan is reused.

To specify the REOPT(ONCE) bind option, run the following command on DB2:

- `db2 bind '<db2home>/bnd/db2clipk.bnd' collection NULLIDR1`

Next, set the following system property value in Maximo:

- `mxe.db.DB2jdbcCollection='NULLIDR1'`

Text Search

Enabling statement concentrator results in sub-optimal plan generation for Text Search queries. If the performance of text search queries is unacceptable with statement concentrator enabled, then statement concentrator should be disabled. Refer to DB2 Text Search performance degradation when STMT_CONC is set to LITERALS for more information.

Note: DB2 10.5 Fixpack 3 introduces improvements to performance for text search with the statement concentrator set to literals. However, internal testing shows that text search queries still perform better when STMT_CONC=OFF.

Database Migration

Maximo recommends enabling the statement concentrator during database migrations from prior releases as this has a positive impact on migration performance.

For more information on the statement concentrator and the REOPT(ONCE) bind option, refer to Statement Concentrator Performance Benefits, Performance Tip: REOPT(ONCE) when using DB2 Statement Concentrator and Enabling the REOPT(ONCE) bind option.

5.1.5 DB2 HADR synchronization considerations

HADR overhead on primary database logging varies, depending on the HADR sync mode chosen. Although a stronger sync mode provides more HA and DR protection, hardware and network speed may limit your choice. The rule of thumb is to use SYNC or NEARSYNC mode on LAN, and to use ASYNC or SUPERASYNC on WAN. With the new HADR tool kit, you can do quantitative analysis of your workload, disk, and network to make an informed choice. The step-by-step TCP and sync mode configuration procedure is recommended during HADR planning, after HADR deployment (if it was not done during planning), or any time when a performance problem is suspected (since workload characteristics can change over time). See the DB2 HADR Performance Guide by Yuke Zhuge and High availability disaster recovery (HADR) synchronization mode for more information.

5.1.6 DB2 History Retention

When using DB2, make sure the history file (db2rhist.asc) is not large. When the file size is greater than 30MB, you can face performance issues with DB2.

To prevent this, you can set the REC_HIS_RETENTN parameter to 30:

- `update db cfg for <DB NAME> using REC_HIS_RETENTN 30`

To prune the history files, use the following procedure:

- Backup the current history files:
  - `cp db2rhist.asc hist.old`
  - `cp db2rhist.bak hist.old.bak`
- Prune the history file:
  - `db2 connect to dbname`
  - `db2 prune history <timestamp> with force option`

5.2 Oracle database tuning

For assistance in performing Oracle database tuning, see the Database Performance Tuning Guide on the Oracle website. For additional information on configuring Oracle for Maximo, see Manually Configuring Oracle 12c.

Maximo recommends using the following initialization parameters:

```
CURSOR_SHARING=FORCE
NLS_LENGTH_SEMANTICS=CHAR
WORKAREA_SIZE_POLICY=AUTO
PROCESSES=4000
OPEN_CURSORS=3000
SESSIONS=4000
TRANSACTIONS=2425
SESSION_CACHED_CURSORS=400
SESSION_MAX_OPEN_FILES=300
```

Calculated Parameters

In addition to the above parameters, the following values should be determined and set for your environment:

Ensure that the `OPTIMIZER_FEATURES_ENABLE` is set to match your current version of Oracle.

A larger log buffer will result in fewer and larger writes to the redo log, which is slightly more efficient. You can check V$SYSSTAT for redo buffer allocation retries and increase the value for `LOG_BUFFER` until retries are near zero. At this point, processes do not have to wait for space in the redo log buffer. However, a high value for `LOG_BUFFER` will result in more memory usage.

The value for `DB_FILE_MULTIBLOCK_READ_COUNT` may be calculated using the following formula:

```
max I/O chunk size
------------------
  db_block_size
```

To obtain the value for max I/O chunk size, download and execute http://www.ixora.com.au/scripts/sql/multiblock_read_test.sql. This script conducts a test and samples actual I/O chunk sizes on your server to aid in setting `DB_FILE_MULTIBLOCK_READ_COUNT`.

The value for `DB_WRITER_PROCESSES` should be set to `1` or `CPU_COUNT/8`, whichever is greater. If the number of processor groups is less than 36 but greater than the number of DB writer processes, then the number of DB writer processes is adjusted to be a multiple of the number of processor groups. If the number of DB writer processes is greater than or equal to the number of processor groups, then there is no adjustment.

Automatic Memory Management

Automatic Memory Management (AMM) allows for automatically readjusting the sizes of the main pools (`DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE`, and `JAVA_POOL_SIZE`) based on existing workloads.

Initialization parameters that are used for enabling this feature are `SGA_TARGET` and `SGA_MAX_SIZE`.

In addition, with Automatic Shared Memory Management (ASMM), the `MEMORY_TARGET` parameter is dynamic (changeable with "alter system" commands), where RAM can be de-allocated from an instance's SGA/PGA and re-allocated to another instance. Initialization

parameters that are used for enabling this feature are `MEMORY_TARGET` and `MEMORY_MAX_TARGET`.

In general, Maximo recommends enabling AMM and ASMM for optimal performance; however, there may be valid reasons where enabling AMM is not feasible in your environment (such as when using large page sizes in Linux). Even if enabling AMM is not feasible, Maximo highly recommends enabling ASMM. Consult with your database administrator to see if you should take advantage of these features and to size these parameters.

For additional details, refer to Using Automatic Memory Management.

Dedicated Server Mode

Maximo recommends using Oracle in dedicated mode (the default) versus shared mode for optimal performance.

5.3 Microsoft SQL Server database tuning

Maximo recommends using the following settings in Microsoft SQL Server (MS SQL).

Set the following system properties:

- `mxe.db.resultsettype=TYPE_FORWARD_ONLY`
- `mxe.db.sqlserverPrefetchRows=200`

For more information, see SQL Server out of memory and performance problems.

Configure the database to allow read committed isolation:

- `ALTER DATABASE <dbname>`
  `SET ALLOW_SNAPSHOT_ISOLATION ON`
- `ALTER DATABASE <dbname>`
  `SET READ_COMMITTED_SNAPSHOT ON`

Refer to Snapshot Isolation in SQL Server for more information.

Edit the `<maximo_root>\applications\maximo\properties\maximo.properties` file and make sure the `mxe.db.url` property is in the below format:

1) For Maximo before 7.6.1

```
jdbc:sqlserver://<hostname>:<port>;databaseName=<dbname>;integratedSecurity=false;sendStringParametersAsUnicode=false;selectMethod=cursor;responseBuffering=adaptive
```

2) For Maximo 7.6.1 or newer version

```
jdbc:sqlserver://<hostname>:<port>;databaseName=<dbname>;integratedSecurity=false;sendStringParametersAsUnicode=false;responseBuffering=adaptive
```

Please note, from Maximo 7.6.1, we removed the ‘selectMethod=cursor’ from above JDBC URL, which can make Maximo overall performance much better.

Before Maximo 7.6.1, we keep the ‘selectMethod=cursor’ due to a defect, which makes Maximo get out of memory if there is no ‘selectMethod=cursor’.

And the defect has been fixed in Maximo 7.6.1, Maximo won't get out of memory when there is no 'selectMethod=cursor'. So we removed 'selectMethod=cursor' from Maximo 7.6.1 for better performance.

If customers who use Maximo before 7.6.1 want to get better performance, they can do below manual fix of the defect first, then remove the 'selectMethod=cursor' from JDBC URL.

Manual fix steps:

a) Go to Application Designer. Find WOTRACK application. Export wotrack.xml. Save it.

b) Search for mboname="LOCATIONS".

Replace the 2nd mboname="LOCATIONS" with parentdatasrc="MAINRECORD" relationship="LOCATION" in BIMVIEWER area.

Before the fix:

```
<section border="false" id="model_viewer" mboname="LOCATIONS"
beanclass="psdi.webclient.beans.bim.viewer.WOModelLocBean" width="100%"
sigoption="BIMVIEWER">
```

After the fix:

```
<section border="false" id="model_viewer" parentdatasrc="MAINRECORD"
relationship="LOCATION"
beanclass="psdi.webclient.beans.bim.viewer.WOModelLocBean"  width="100%"
sigoption="BIMVIEWER">
```

c) Save it. Then import wotrack.xml

After editing `maximo.properties` file as above, rebuild the Maximo ear file and redeploy on the application server(s).

Note: Internal testing has shown that Microsoft SQL Server may not support as many concurrent users as either DB2 or Oracle. In addition, measured response times with MS SQL are overall higher than those same transactions using DB2 or Oracle. However, many smaller deployments of Maximo do successfully use MS SQL in production. If your performance with MSSQL is not acceptable, you may want to investigate using DB2 or Oracle instead.

5.4 General database tuning

You can apply standard database-tuning techniques to Maximo. Periodically monitor a production database during peak load. You can use any appropriate monitoring tools or techniques. If necessary, adjust parameters to resolve the bottlenecks reported by the monitoring tools.

5.4.1 Indexing

Indexing a database requires a good understanding of the data, the user functions, and how databases use indexes. Indexes use key parts of data from a table in a binary structure to

enhance searching capability. Each record of data in the table must have associated data in the index.

Indexing can greatly increase search speeds. However, a drawback of indexes is that for each insert, update, or delete, the index must also be updated. Database administrators often apply many indexes to a table to enhance searching, and then sometimes find that other activities have slowed. Review the indexes to ensure that you have the right balance for searching and for updating tables.

Special index types

Some special index types are available on each database platform that are not available in the Database Configuration application. These index types can be created and maintained from the back end, and they can improve performance in specific cases. For example, on Oracle and DB2, you might create a bitmap index or a function-based index if you determine that these indexes would improve certain queries.

If you use special index types, the system administrator must remember to remove any special indexes before configuring database changes. After the database is configured, the special indexes must be replaced.

Customization and indexes

Customizing Maximo can change the way you select information from the database. Some customizations include additional tables and columns. If you have customized Maximo, carefully compare indexes to the user functions that use them by taking DB2 snapshots or Oracle AWR reports to look for candidate statements for indexing. Ensure that you implemented the right balance of indexes.

See DB2 Query Tuning and Index Creation and Tuning and Indexing Oracle Databases for more information.

5.4.2 Tablespaces

During installation, you have the option to separate indexes from the data and place them into different tablespaces. This is highly recommended for optimal performance. You can also do this post installation, but requires much more work by using the Database Configuration application to move the indexes.

Also, separate large tables, like Assets and Work orders, into their own tablespaces for optimum performance.

5.4.3 Sequences

For improved performance, use sequence caching in Oracle and IBM DB2® platforms. Use cache size **50** for all sequences, with the exception of the `maxseq` sequence, which is used for rowstamps. The size for this sequence is **500**.

To enable or modify sequence caching, perform the following steps:

Run the following command to generate a script file that contains the SQL statements that set the sequence cache size:

```
db2 "select 'alter sequence maximo.' || sequencename || ' cache 50;'
from maximo.maxsequence" > change_seq_cache.sql
```

1. Edit the `change_seq_cache.sql` file to change the sequence cache size for the maxseq sequence to **500**.
2. Next, remove any entries from the file that match the following sequence names:

ASSETATTRIBUTESEQ	DPADISKSEQ	DPAMSWSUITECOMPSEQ

CDMCITYPESSEQ	DPADISPLAYSEQ	DPAMSWSUITESEQ
CLASSANCESTORUSEQ	DPAFILESEQ	DPAMSWUSAGERANGESEQ
CALSSSPECSEQ	DPAIMAGEDEVICESEQ	DPAMSWUSAGESEQ
CLASSSPECUSEWITHSEQ	DPAIPXSEQ	DPAMSWVARIANTSEQ
CLASSSTRUCTURESEQ	DPALOGICALDRIVESEQ	DPANETADAPTERSEQ
OMPSEQ	DPAMADAPTERSEQ	DPANETDEVCARDSEQ
RELATIONRULESSEQ	DPAMADPTVARIANTSEQ	DPANETDEVICESEQ
RELATIONSEQ	DPAMEDIAADAPTERSEQ	DPANETPRINTERSEQ
ACTCIRELATIONSEQ	DPAMMANUFACTURERSEQ	DPAOSSEQ
ACTCISEQ	DPAMMANUVARIANTSEQ	DPASOFTWARESEQ
ACTCISPECSEQ	DPAMOSSEQ	DPASWSUITESEQ
DEPLOYEDASSETSEQ	DPAMOSVARIANTSEQ	DPATCPIPSEQ
DPACOMMDEVICESEQ	DPAMPROCESSORSEQ	DPAUSERINFOSEQ
DPACOMPUTERSEQ	DPAMPROCVARIANTSEQ	OMPCIRLNSEQ
DPACPUSEQ	DPAMSOFTWARESEQ	

Table 1: Sequences to remove from `change_seq_cache.sql` file

3. Run the SQL script on the database to change the sequence cache values:
   db2 -tvf change_seq_cache.sql

The example above is for DB2. The steps for Oracle are identical with the exception that you use the *sqlplus* command instead of the *db2* command.

5.4.4 Query tuning

Having optimized SQL queries is critical for good performance in Maximo. Ongoing query tuning is essential in order to prevent non-optimal queries from having negative effects on performance for the entire system.

Tips for creating well-tuned SQL queries

- Eliminate the use of the leading % wildcard character. Using this character will result in a full table scan. The best way to accomplish this is to change most search types from wildcard to exact. Refer to Setting the appropriate search type for more information.
- Order index columns so that the number of rows returned will be reduced earlier. For instance using a Boolean column type in the first position of the index would only be able to eliminate about half the rows, while using a column such as ticketuid in the first position would give you a much smaller result set.
- Replace the use of the join with the SYNONYMDOMAIN table by the values that are returned from the query of the SYNONYMDOMAIN table. Refer to Queries with SYNONYMDOMAIN joins for more information.
- Using “not in” or “!=” will result in a full table scan. Replace with “in” or “not exists”

Tips for finding non-optimal custom queries

When looking for custom queries that need to be tuned, you can start by looking in the following places:

- QUERY table – clause column

- CONDITION table – expression column
- ESCALATION table – condition column
- WFCONDITION table – condition column
- MAXRELATIONSHIP table – whereclause column
- MAXAPPS table – restrictions and orderby columns
- MAXTABLEDOMAIN table – validtnwhereclause, listwhereclause columns

List panel order by queries

Most of the WHERE clauses in queries are generated by individual users on the list tabs of applications. While this is a powerful feature of Maximo, it can produce inefficient SQL.

Queries with order by clauses can be expensive, especially list panel queries since these are heavily used. You can edit the presentation XMLs and remove order by clauses on these queries.

Saved user queries

Users usually have a well-defined set of columns that they want to query in each application. You can identify these columns in user team interviews during implementation, or later, by examining reports of slow-running SELECT statements. You can then index these columns to improve system performance.

Users can create and save their own queries, and can share queries with other users. Saved queries are stored in a table named QUERY. You can periodically review these saved queries for inefficient conditions and use of columns that are not indexed, and to remove order by clauses.

Someone with SQL expertise can help create special-purpose queries (for example, return all PM work orders created since Monday of this week). Using these queries can save end users the effort of querying larger sets and sorting and scrolling through them.

You also can set up users with an efficient default query for the most-used applications so that they see their preferred record set when they enter the application. For example, in Work Order Tracking, you might specify a default query for a supervisor named “Smith” so that the query only shows work orders with “SMITH” in the Supervisor field.

You want to prevent users from performing queries that retrieve thousands of records. These types of queries adversely affect database performance, causing slower user response times. To reduce the likelihood of performance robbing queries, define default queries for your users, or educate users about how to use efficient queries.

A tool that can be used to identify poorly performing queries the Performance Analysis Suite. The Performance Analysis Suite is capable of parsing DB2 snapshots and Oracle AWR reports to identify poor performing SQL. We recommend that you monitor query performance at least once per month by analyzing a snapshot or AWR report for the busiest hour of the busiest day of the week. This is especially important after adding new functionality or a new user base to the system.

Start Center result set and pre-defined queries

The Start Center is a heavily used component of Maximo. As such, it is important to ensure that any start center result set queries are well tuned to avoid performance issues. In general, all pre-defined queries must be well tuned for optimal performance across the system.

Doclinks queries

The doclinks query runs whenever an application is opened or a new record is inserted into an application to determine if there are any documents attached to the record. This query turns on the paperclip icon in the application to indicate there are attached documents.

This query is quite complex if there are many associated records since the paperclip is turned on when there are attachments to other related records as well. For instance, if you have assigned an asset to your incident, then any attachments to the asset are also available to be viewed from the incident document link.

While it is always possible to turn off the doclinks query by using the system property `enabledoclinkonload=false` as described in the Disable Doclinks on data loading processes section of this document, most customers prefer not to turn off the query as their users find it useful to know if there are attached documents.

Another solution is to simplify the query. All the doclinks queries can be found by querying the `maxrelationship` table searching for all relationships with the name `doclinks`:

```
SELECT * FROM MAXIMO."MAXRELATIONSHIP" where name ='DOCLINKS';
```

The parent field gives you the name of the table from which relationship originates and is usually the main table under the application.

Let's look at a typical doclinks query. This is the where clause from the doclinks query that is run from the Incident application:

```
(ownertable = 'INCIDENT' and ownerid = :ticketuid) or (
ownertable='SR' and ownerid in (select ticketuid from sr where
ticketid=:origrecordid and class=:origrecordclass) ) or (
ownertable='PROBLEM' and ownerid in (select ticketuid from problem
where ticketid=:origrecordid and class=:origrecordclass )) or (
ownertable='WOCHANGE' and ownerid in (select workorderid from
wochange where wonum=:origrecordid and woclass=:origrecordclass) )
or ( ownertable='WORELEASE' and ownerid in (select workorderid
from worelease where wonum=:origrecordid and
woclass=:origrecordclass) ) or (ownertable='SOLUTION' and ownerid
in (select solutionid from solution where solution=:solution)) or
(ownertable='ASSET' and ownerid in (select assetuid from asset
where assetnum=:assetnum)) or (ownertable='LOCATIONS' and ownerid
in (select locationsid from locations where location=:location))
or (ownertable='WOACTIVITY' and ownerid in (select workorderid
from woactivity where origrecordid=:ticketid and
origrecordclass=:class)) or (ownertable='JOBPLAN' and ownerid in
(select jobplanid from jobplan where jpnum in (select jpnum from
woactivity where origrecordid=:ticketid and
origrecordclass=:class))) or (ownertable='COMMLOG' and ownerid in
(select commloguid from commlog where ownertable='INCIDENT' and
ownerid=:ticketuid))
```

The first thing to do is identify all the associated tables. This can be done by looking at all of the subqueries. If we separate the query into its subcomponents, we will see the following:

```
(ownertable = 'INCIDENT' and ownerid = :ticketuid) or


(ownertable='SR' and ownerid in (select ticketuid from sr where
ticketid=:origrecordid and class=:origrecordclass) ) or
```

```
(ownertable='PROBLEM' and ownerid in (select ticketuid from
problem where ticketid=:origrecordid and class=:origrecordclass ))
or

(ownertable='WOCHANGE' and ownerid in (select workorderid from
wochange where wonum=:origrecordid and woclass=:origrecordclass) )
or

(ownertable='WORELEASE' and ownerid in (select workorderid from
worelease where wonum=:origrecordid and woclass=:origrecordclass)
) or

(ownertable='SOLUTION' and ownerid in (select solutionid from
solution where solution=:solution)) or

(ownertable='ASSET' and ownerid in (select assetuid from asset
where assetnum=:assetnum)) or

(ownertable='LOCATIONS' and ownerid in (select locationsid from
locations where location=:location)) or

(ownertable='WOACTIVITY' and ownerid in (select workorderid from
woactivity where origrecordid=:ticketid and
origrecordclass=:class))or

(ownertable='JOBPLAN' and ownerid in (select jobplanid from
jobplan where jpnum in (select jpnum from woactivity where
origrecordid=:ticketid and origrecordclass=:class))) or

(ownertable='COMMLOG' and ownerid in (select commloguid from
commlog where ownertable='INCIDENT' and ownerid=:ticketuid))
```

The `ownertable` identifies the parent table to which the document is attached. For example, we see `ownertable = 'SOLUTION'`; this brings all the documents attached to the solution associated with this record to the paperclip on incident. Therefore, for `incident`, all the attached documents would come `from incident, sr, problem, wochange, worelease, solution, asset, locations, woactivity, jobplan`, and `commlog` records associated to this particular incident.

The next step is to work with the user community and identify all the associated documents that are not required to be viewed from the incident record. For instance, if `solution` is not being used in your implementation, then it would not be necessary to see the documents attached to solutions.

To simplify the relationship, remove the section that is not needed. For example, to remove the link to the solutions documents, remove the section shown in bold text from the where clause in the `maxrelationship` table:

```
(ownertable='SOLUTION' and ownerid in (select solutionid from
solution where solution=:solution)) or
```

Make sure you remove the "or"; otherwise, the query is not syntactically correct.

This can be repeated for all the doclinks queries. If you are able to significantly simplify this query, it improves your performance whenever you open a record.

Queries with SYNONYMDOMAIN joins

Maximo uses synonym domains for the purpose of "constant lookup" or translation; hence, it is very common to see queries containing multiple joins with table "SYNONYMDOMAIN" (or in the form of "in" or "exist" predicates). It is also common to see expensive queries having one or multiple joins between large tables and table "SYNONYMDOMAIN".

Take the following inefficient query as an example:

```
select * from sr where
  (status in (select value from synonymdomain where
              domainid = 'SRSTATUS' and maxvalue not in 'CLOSED')
   and historyflag =0)
and
  ((affectedperson=(select personid from maxuser where userid=:L0)
     or
     reportedby=(select personid from maxuser where userid=:L0) )
    and pmcomtype is null
    and status not in (select value from synonymdomain where
     maxvalue='DRAFT' and domainid='SRSTATUS'))
for read only
```

Counting the number of joins in this query shows that there are five joins in this query, of which three are with "SYNOYMDOMAIN" table. It might not be obvious where the fifth join is.

"SR" is a view, within which there is one more join between tables "TICKET" and "SYNONYMDOMAIN":

```
create view SR as select distinct
```

... (irrelevant parts omitted here)

```
from ticket where class in (
 select value
 from synonymdomain
 where domainid='TKCLASS' and maxvalue='SR');
```

Relational database servers sometimes have performance problems with queries having multiple join operations, especially when table data is not evenly distributed. A simple strategy for addressing join performance problems is to reduce the number of joins. In this case, you can modify the "SR" view first, since it is relatively easy to modify a view definition in a deployed environment without changing any code or any configuration.

Essentially the join with "SYNONYMDOMAIN" here is used to look up the "*value*" corresponding to internal maxvalue "SR" of synonym domain "TKCLASS". You can replace the sub-select of the "IN" clause with the actual "*value*" queried from table "SYNONYMDOMAIN" (which is 'SR' in this case). Go to **Database Configuration** and modify the object "SR" view where clause from:

```
class in (select value from synonymdomain where domainid='TKCLASS'
and maxvalue='SR')
```

to:

```
class in ('SR')
```

This revision produces exactly the same result, except for the performance difference. In this case, in the test environment, response times are reduced by 95% from the original query.

When you have slow queries joining large tables with "SYNONYMDOMAIN", you can use the above technique to improve performance. The basic principle here is easy: the fewer joins you have, the more likely database servers are able to find the most optimized access plan. Although you cannot add or delete synonym domains, Maximo does allow adding new "*value*" for existing synonym domains. When you take this performance tip to manually replace "SYNONYMDOMAIN" joins in views or queries with fixed values, remember to refresh your changes (in views or queries) whenever you modify (add or remove) synonym domain values. Otherwise, you might get incorrect results. Fortunately, that rarely happens and is usually only during the system development phase.

Restricting querying in applications

You can control or restrict user access to query features and user ability to query on specific columns by using a combination of methods:

Application Designer

You can use the **Application Designer** to customize an application by adding or removing columns from the List tab. You can then ensure that the columns to be queried are all indexed.

Application Cloning

You can clone an application and then use the **Application Designer** to create an alternate version of an application with a restricted number of columns that can be queried.

Security Groups

After you clone applications, you can use **Security Groups** to assign users to specific application clones.

You can also use security groups to prohibit access to the *More Search Fields* and where clause *Advanced Query Options*. By prohibiting access to those options, you limit users to querying on the **List** tab of the application.

Independent security groups allow you to configure site-specific access to a set of applications, options or controls. For performance reasons, you should use independent security groups whenever possible. Refer to The Independence of Independent security groups for more information.

Resetting the DATETIME data type

When you search on fields whose data type is DATETIME, the resulting database query uses TO_TIMESTAMP in the WHERE clause. Queries that use TO_TIMESTAMP are time- consuming.

You can edit the WHERE clause manually and replace TO_TIMESTAMP with TO_DATE. Queries that use TO_DATE in the WHERE clause yield a faster search. However, you must edit the query each time you search on a DATETIME field.

In many cases, a better approach is to change the data type of routinely queried DATETIME fields to the data type DATE. Fields with a data type of DATE use TO_DATE in the WHERE clause.

Use the **Database Configuration** application to change the data type of a column.

ALN field types

When queried through normal filtering (Query by Example), which is case insensitive, the ALN type fields use the UPPER function in the query; thus the index is not used. For any ALN type fields where case is not important, Maximo recommends to change those field types to UPPER.

Making this change allows an index to be used resulting in a significant performance improvement.

5.4.5 Setting the appropriate search type

You can improve the ease of use and convenience for users by setting the appropriate search types for database columns. Using appropriate search types reduces the load on the database. You can change the default search type of WILDCARD to TEXT or EXACT. TEXT and EXACT searches can use indexes. You specify the search type for a database column in the Database Configuration application. You also can change the search type for groups of columns. See Removing Automatic Wildcard Searching in Maximo for more information. Changing the search type from WILDCARD to EXACT results in a significant performance improvement.

EXACT search type

When an end user does a search, the default method of searching on many database fields is to use wildcards (SEARCHTYPE = WILDCARD). The WILDCARD search type causes Maximo to construct a condition of the following form when the user enters "value" in a field on the **List** tab:

```
column like '%value%'
```

In wildcard searching, the database engine cannot use indexes. Searching without indexes will result in slower search times, especially on tables with many rows. This also affects overall database performance and will impact all users.

You can specify a search type of EXACT when word searching is not needed. For example, key fields (such as Work Order and Purchase Order) and value list fields (such as Work Order Status) can benefit from the indexing that is used in EXACT searches. EXACT searches use wildcarding only if a user explicitly enters wildcard characters on the List tab or in the WHERE clause.

TEXT search type

Most tables have one or longer character columns for descriptions, memos, or remarks. You can provide a search type of TEXT, and a corresponding Oracle Text index or SQL Server full text catalog, for columns that have a lot of text. Text indexing puts some load on the database because of the constant need for background processing to keep the indexes synchronized. However, text indexing produces efficient word searching of description fields.

Specify a search type of TEXT on description fields that word searching is required. Use TEXT on description fields of tables with large numbers of rows (tens of thousands, for example).

The text search engine takes some time to refresh the indexes, so new records may not be found until the text search index refreshes itself.

On Oracle, you can modify the procedure `maximo_ts_job` to change the schedule of the synchronization process to any interval that you want. In addition, schedule the `ctx_ddl.optimize_index` procedure to run during your regularly scheduled database maintenance, such as when `runstats` or `reorgs` are run, to keep the indexes defragmented.

On SQL Server, set and modify the population schedule for the Full Text Catalog. Use SQL Server 2000 Enterprise Manager or SQL Server 2005 Management Studio.

IBM DB2 provides text search; however, it may not be installed by default. Refer to Enabling DB2 database for Text Search with Maximo for more information on setting up text search with DB2.

NONE search type

The NONE search type prevents a column from being used in a WHERE clause, and still allows the display of the returned values on the List tab.

5.4.6 Base language install

If you are installing for a single language, make sure to specify that language at install time. For example, do not install using English and then add French if French will be the only language or primary language used by end users. Installations with more than a single language have additional SQL query overhead to determine values for languages that are installed, whereas, installations with a single language can avoid this overhead.

5.4.7 Database server virtualization

Internal testing has shown that putting the Maximo database in a virtualized environment such as VMWare can have significant performance impacts. These impacts increase rapidly with larger workloads. While many customers run their Maximo database successfully in a virtual machine, we recommend dedicated hardware for optimal performance.

5.5 Database maintenance

Database maintenance consists of tasks such as reorganizing tables and indexes, and archiving and deleting historical data.

5.5.1 Statistics

Statistics must be periodically generated and updated on all the tables and indexes, including text search indexes. This is necessary to ensure the best database performance possible. Updating statistics is especially important when major changes to the number of rows in the tables is occurring. This usually happens during the first 12 months of a new system or when a large number of new users or assets are added to an existing system. For a stable production system, updating statistics weekly or monthly is sufficient. In the **Database Configuration** application, use the **Select Action** menu action to run **Update Statistics**. You can also update index statistics using database commands:

DB2 runstats

To ensure that DB2 is using the optimal path to access data, runstats must be executed periodically as part of your database maintenance activities.

Use the following runstats command:

```
RUNSTATS ON TABLE <TABSCHEMA>.<TABNAME> WITH DISTRIBUTION AND
DETAILED INDEXES ALL ALLOW WRITE ACCESS
```

Executing the `RUNSTATS` command for indexes with the `DETAILED` clause collects statistical information about indexes that allows the optimizer to estimate how many data page fetches are required, based on various buffer-pool sizes. This additional information helps the optimizer make better estimates of the cost of accessing a table through an index.

Using the `WITH DISTRIBUTION` clause allows the optimizer to choose an optimal access path if data values are very different from one another or if data values are clustered together in some spots, or if many duplicate data values are encountered. It can also help queries with predicates that do not have parameter markers or host variables.

Note: It is also possible to use a table iteration approach to do a massive runstats across all Maximo tables and indexes as shown below:

```
select 'runstats on table maximo.' || tabname || ' on all columns and
detailed indexes all allow write access;' from syscat.tables where
tabschema ='MAXIMO'
```

Oracle DBMS_STATS.GATHER_TABLE_STATS

On an Oracle database, you can execute `DBMS_STATS.GATHER_TABLE_STATS(ownname => schema_owner, cascade => true, estimate_percent => dbms_stats.auto_sample_size)` to update the statistics for all tables owned by that schema owner.

MS SQL Server sp_updatestats

In Microsoft SQL Server, you run the stored procedure `sp_updatestats` to update statistics more frequently than the default updates.

5.5.2 Reorganizing tables and indexes

You may want to perform reorganizations to reclaim space in a table. When a large number of updates are made on a table, the space can become fragmented. If no further growth is expected in the table, then the space can be reclaimed. However, if further inserts are expected, then the database should be effectively reusing the space within the table. Reorganization must not be done soon after a previous reorganization as additional inserts will be done to the table. The action of shrinking and growing the table is detrimental to performance.

Another case where reorganizations can be helpful is to improve performance. When the data is not aligned on an index, performance can become degraded.

Some questions to consider before doing reorganization are:

- Is fragmented space causing a performance problem?
    - Will the records in the table be updated or will new records inserted within a short period of time?
    - Is fragmented space resulting in inefficient storage utilization?
- Is the order of the table data aligned with a particular index?
- Has performance degraded over time or has it happened at a specific time?
- What changes could have caused the incident?
- How does the performance correlate to system CPU and I/O utilization?
- Have access plans changed?
- What does the reorganization check say?
- If table reorganization is done, what is the expected benefit? Running reorganization affects data availability, consumes system resources and takes time to complete.

In DB2, use the *reorgchk* and *reorg* tools to optimize tables and indexes.

For Oracle and Microsoft SQL Server, it is not as important to reorganize tables; however, there may be times when rebuilding indexes can be beneficial for performance. In those cases, you would use the following commands:

- `ALTER INDEX <idxname> REBUILD;` (Oracle)
- `ALTER INDEX <idxname> ON <tablename> REORGANIZE;` (MS SQL)

Consult with your Oracle or Microsoft SQL Server DBA to understand when rebuilding / reorganizing indexes would be beneficial.

General considerations for reorganizations are:

- Only run reorganization against a table when careful analysis has been done to determine that reorganization corrects the performance problem.

- Only reorganize tables that have had many deletes and updates to reclaim space when it has been determined that no further inserts will be done in the near term.
- If it has been determined that a reorganization improves a performance issue with the table/index, then careful consideration must be given on whether the reorganization should be done online or offline while the system is down. System resources are required to do reorganizations. Reorganization can slow down the system. As well, consider that users will have to wait for locks during an online reorganization.
- Note that the larger the table is, the longer a reorganization for that table takes. If a table is especially large, you might want to consider using the database's compression feature to improve performance with database utilities.
- Carefully maintain the size of the database by archiving and trimming tables where appropriate to reduce the time needed for reorganization.

DB2 considerations for reorganizations are:

- A reorganization check must be run on the table to determine if the table needs to be reorganized before performing the reorganization; however, this should not be the only indicator of whether reorganization needs to be done. For example, reorgchk flags a table for reorganization if the table size in bytes is not more than 70% of the total space allocated for the table. Running reorganization in this case does not resolve the issue. For instance, a table using 56% of the space allocated could be improved to 66%, but that still does not meet the 70% requirement, so it is again flagged for reorganization.

- If excessive locking is occurring, verify the following DB2 lock parameters:
  - Maxlocks (maximum percentage of lock list before escalation)
  - Locklist (amount of memory allocated for locks)
  - Highly experienced DB2 database administrators may want to consider the following two types of reorganizations, each with advantages and disadvantages:
    - Inplace reorganization – Run while the database is active and users are doing work.

      An inplace reorganization requires an instant super exclusive (Z) lock. When a table holds a Z lock, no concurrent application can read or update data in that table. However, a Z lock cannot be obtained when a pre-existing intent share (IS) lock is active. If the IS lock is never released, the reorganization process for the table never starts. If the IS lock is released, inplace reorganization starts and subsequent IS locks that are obtained do not halt the reorganization until the truncation phase. At the truncation phase, the inplace reorganization acquires another Z lock that prevents users from setting IS locks on tables to do work. Any requests from the user must wait for the release of the Z lock, resulting in users perceiving system slowdown. Inplace reorganizations also require log space to store temporary data, which can grow quite large, when run against an index.
    - Offline reorganization – Run when the database is not active.

      The length of time to run the reorganization depends on the database size. Tables are readable up until the copy phase of the reorganization. Monitoring can tell you what phase the reorganization is in. Any reads started before the copy phase can continue and the reorganization waits for the read to end before table access is blocked. New reads cannot start until the copy phase completes.

Note: You usually want to determine tables needed a reorg using updated statistics. On DB2, you can update the statistics for all tables in a database by using the "reorgchk" command:

```
reorgchk update statistics on table all
```

The output from the reorg command will have a per table indicator of which tables require a reorg. Any table or index that has a "*" value on the REORG column means this table or index need a reorg. You can then issue the appropriate reorg command, e.g.:

```
reorg table <table name> allow read access
reorg indexes all for table <table name> allow read access
```

It is also possible to use a table iteration approach to do a massive runstats across all Maximo tables as shown below:

```
select 'reorg table maximo.' || tabname || ' allow read access;' from
syscat.tables where tabschema ='MAXIMO'
```

Check with your DBA for any similar considerations for Oracle or Microsoft SQL Server.

5.5.3 Pruning workflow tables

Workflow tables such as WFTRANSACTION and WFINSTANCE can grow to very large sizes over time, which can cause performance issues. You should periodically prune the entries that are no longer needed by following the guidance in Script To Delete Unused Workflow Data.

5.5.4 Archiving and deleting historical data

Deleting unnecessary data from the database helps to optimize its performance. *IBM Maximo Archiving with InfoSphere Optim Data Growth Solution V7.6* provides a way to archive and delete historical data from your database.

Archiving historical data can also facilitate upgrading to a new version of Maximo because the historical data is not critical to day-to-day operations and might not need to be included in the new system.

For more information, refer to Maximo Archiving with InfoSphere Optim Data Growth Solution.

5.5.5 Key Performance Indicators (KPIs)

Key performance indicators (KPIs) display the state of systems and processes in Maximo. Because KPIs can be user-defined, monitor them for efficiency.

Live KPIs on Start Centers

When a KPI is defined on a Start Center, you can choose a method of retrieving information. You can run an immediate query to get the information, or you can retrieve the information from a table that is updated by a cron task.

The immediate query runs every time the Start Center opens. This approach can cause a long delay in opening the Start Center, and it puts a load on the database.

KPI queries

Periodically review queries that you write for KPIs. Check for SQL efficiency and index usage.

KPI cron task frequency

How up-to-date does your KPI information need to be? 5-minute intervals might sound like a good idea initially, but 60-minute or 120-minute intervals might be as useful to the people who want to see the information. Longer KPI cron task intervals reduce the load on the database from KPIs and can improve system performance.

In general, use longer rather than shorter KPI cron task frequency intervals when the value of the data to the end user is essentially the same at longer intervals.

KPI best practices

- Set all KPIs to retrieve their data from the KPI table.
- Use cron tasks to run KPI queries at reasonable intervals.

6 Network considerations

Clients connect to the application over the network. The application also communicates with its various parts (application server, database, report server) over the network. If any segment of the network performs poorly, the end user experiences a system that is slow and hard to navigate.

Maximo is a web-based product that operates on a request and response basis. If the requests and responses are delivered slowly, Maximo has no control over response time.

Optimum network configurations for Maximo should include the ability to produce **50 ms** or faster round-trip packet response between the client and the server. Users may begin to experience performance degradation if the network does not operate within these parameters.

Data bandwidth requirements are difficult to determine since it is highly dependent upon the workload characteristics. Some transactions require more data than others, with work order processing being one of the heavier transaction types.

One way to estimate bandwidth requirements for your Maximo implementation would be to capture data throughput while performing pre-production performance testing as well as the length of the test and use those values to calculate the average network bandwidth used.

6.1 Using Citrix or Windows® Terminal Server

Other options for resolving low bandwidth or high-latency network performance issues include services such as Windows® Terminal Server and Citrix. These services can help provide maximum performance between the Windows Terminal Server or the Citrix client and the application server with a minimum of traffic between the Windows Terminal Server or the Citrix server and the end user.

A benefit of running Maximo through Citrix is that Citrix traffic is treated as “business traffic”. In some customer environments, business traffic can take priority over non-business traffic.

Some customers have found considerable improvements in network performance when they use a Citrix or Windows Terminal Server. Note that IBM does not test or certify Citrix or other bandwidth tools.

You can use network caching, acceleration, and compression utilities to improve network performance.

6.2 Using compression techniques to improve performance

Maximo provides performance improvement capabilities in a variety of areas.

Sites that have bandwidth or latency issues can use one of the following techniques to improve performance:

- HTTP Compression
- Hardware Compression
- Gzip Compression

Each of these compression options is mutually exclusive of the others. That is, if you are using hardware compression, then you should not also use HTTP or Gzip compression.

6.2.1 HTTP compression

HTTP compression is a capability that can be built into web servers, such as IBM HTTP Server and Apache HTTP Server 2.x and above, and web browsers to make better use of available bandwidth, and provide faster transmission speeds.

HTTP compression affects all servers in a cluster. HTTP data is compressed before it is sent from the server. Note that hardware load balancers that bypass the HTTP Server cannot employ this method.

Compression-compliant browsers announce, to the server, what compression methods the browser supports, so the server can provide the compressed data in the correct format.

Browsers that do not support compression simply download uncompressed data.

Data can be compressed by using a compression module such as Apache's mod_deflate. The compression scenario is dictated by what the server software supports.

The compression scenario that follows has been tested and has been found to be a good solution for low bandwidth locations.

IBM HTTP Server: Use Apache mod_deflate and set DeflateCompressionLevel to 3 or 6 to improve response time in environments that have low bandwidth and high latency.

See Data compression on IBM HTTP Server for information about configuring mod_deflate.

6.2.2 Hardware compression using network appliances

Network appliances, such as Juniper and Riverbed, provide compression and caching features. Network appliances can help compress data and optimize bandwidth. Customers report that network appliances can prove to be beneficial to system performance, especially in a high-latency and/or a low bandwidth environment.

Hardware compression affects all servers in the configuration.

6.2.3 Runtime gzip compression

The Maximo distribution contains the gzip utility that you can configure to compress files before sending them to browser users. This reduces bandwidth usage, but it also increases CPU and memory consumption. HTTP server compression is a more efficient method.

For information about how to set up to use gzip, see Network Performance via Response Compression & Browser Cache GZip & MaxAge.

6.3 Image and JavaScript browser caching (MaxAge)

The MaxAge browser file caching filter is enabled by default on the application server. Browser images, CSS files, and JavaScript files are stored locally.

Storing images and files locally benefits performance in the following ways:

- Images and files do not need to be constantly requested from the server.
- Because images are not downloaded as often, less bandwidth is required.

For Maximo, keep browser file caching enabled. For information about how to ensure that this feature is enabled, see Network Performance via Response Compression & Browser Cache GZip & MaxAge.

Ensure that the number of days until refresh is enabled to 32 days or longer. If required, modify the web.xml file and set the max-age=<seconds> value to 2764800 or higher. 2764800 seconds is 32 days.

7 Maximo tuning

There are several ways to tune Maximo, such as by tuning system properties and connection pool settings.

7.1 Application settings

System properties can be used to tune performance in Maximo. For more information on the system properties that can be configured, see System properties overview.

7.1.1 Disable Doclinks on data loading processes

Doclinks provide information about attachments that are linked to individual records in the database. To associate doclinks with records, the database must evaluate conditions to make sure that there are attachments available for a particular record. These evaluations consume resources and degrade performance of the data loading process.

To improve performance, you can disable doclink processing so the evaluations are not performed when records are loaded into the database by setting the `enabledoclinkonload` property in the *webclient.properties* file.

With `enabledoclinkonload=false`, the paperclip for every record appears as highlighted regardless of whether or not an attached document record is associated with the record. Use this setting to enable the performance improvement.

With `enabledoclinkonload=true`, the query is run to see whether the record has an attached document associated with it. Only those records with attached documents associated with them appear with a highlighted paperclip icon. This is the default value of the property.

After setting the property, you must save, rebuild, and redeploy the maximo.ear file for the changes to take effect.

If you want to try to tune the doclinks queries instead of disabling the feature, see Doclinks queries.

7.1.2 Database connection pool

Maximo uses proprietary connection pooling. You can tune the connection pool settings for better management of connections:

Initial connections: **mxe.db.initialConnections**

Default Value: **6**

When the server starts up, it immediately creates six database connections and puts them in the connection pool, ready to be used. You can opt for a larger number only if large numbers of users log in immediately after the server starts up. Otherwise, this number must not exceed the maximum free connection, because extra connections are freed if not used.

Maximum free connections: **mxe.db.maxFreeConnections**

Default Value: **8**

As transactions are finished for the user, connections are freed back to the pool. If the connections put back are greater in number than the connections being requested, the number of free connections increases. Once the server detects that the number of free connections exceeds this setting, the server closes the extra connections to keep the number of free connections within the range.

Minimum free connections: **mxe.db.minFreeConnections**

Default Value: **4**

When a connection is requested by a user operation, a free connection in the pool is assigned to this user. If more free connections are taken from the pool than the connections freed back to the pool, the free connection number drops. Once the server detects the free connection number is below this setting, it creates new connections to fill the pool so the connections are ready for the new requests.

New connections: **mxe.db.newConnectionCount**

Default Value: **3**

When the connection number drops below the threshold of minimum free connections, new connections are created in the batch of this setting to fill the pool.

To reduce the overhead associated with allocating and freeing database connections, increase the value in the properties `mxe.db.initialConnections` and `mxe.db.maxFreeConnections`. These properties are set in the *System Properties* application.

If you have a small number of cluster members or if your database resources are abundant, increase the number of free connections to provide faster access to the database.

7.1.3 Limiting query times

Maximo provides configurable system properties that determine how long queries can run. A control is provided in the user interface so you can cancel your own queries before a configured timeout property cancels them. These configurable attributes are set in the **System Properties** application.

Query timeout: **mxe.db.QueryTimeout**

Value: **300**

This attribute determines how long an SQL query is allowed to run without producing a result set. If an SQL query has not completed in the timeout interval, the SQL query is stopped.

Result set timeout: **webclient.ResultSetQueryTimeout**

Value: **300**

This attribute determines how long Query by Example (QBE) queries can run without producing a result set. These queries are constructed from user input that is specified on forms and fields in the user interface. You do not specify the SQL commands or verbs needed to run the query. Instead, QBE queries are translated into SQL queries by a QBE program that processes form input.

This attribute also controls how long filter queries can run, such as when you filter table rows or users, or when you ask to see all records for an object type, such as all assets. If you do not click the control to cancel a query, the query stops when this timer expires.

Maximum Lookup Count: **mxe.db.lookupmaxrow**

Value: **2000**

This attribute controls how many records are returned by a lookup. Because unfiltered lookups can take a long time to perform, this property can be used to prevent database degradation by limiting the number of records being processed by a lookup.

7.1.4 Fetch stop limits

Use these properties to set a limit on the number of objects that can be fetched from the database, and then constructed on the server into a single set:

mxe.db.fetchStopLimitEnabled

Value: **0** to disable, **1** to enable. The default is **1.**

This property enables or disables checking of the fetch stop limit.

mxe.db.fetchResultStopLimit

Value: **5000**

The fetch stop limit used when checking of the fetch stop limit is enabled. The limit applies to all objects for which a specific fetch stop limit property is not specified.

mxe.db.fetchResultStopLimit.*OBJECTNAME*

Value: Any integer value. **-1** means no fetch stop limit for this object.

The fetch stop limit for the specified object. If this property exists, its value is used as the fetch stop limit for this object only. To disable the fetch stop limit for a specific object, create this property and set its value to **-1**.

mxe.db.fetchStopExclusion

Value: Comma-separated object name list, as object names appear in the MAXOBJECT table.

If an object name is in the list, the fetch stop limit check is disabled for the object. If an object name is in the list and the same object is specified in an mxe.db.fetchResultStopLimit.*OBJECTNAME* property, the exclusion overrides the other property.

For additional details on these settings, see Using fetch stop limits to prevent out-of-memory errors.

7.1.5 Asynchronous and client side validations

For optimal performance, Maximo recommends to enable both asynchronous validation and client side validation. By default, asynchronous validation is enabled; however, client side validation is not automatically enabled. To enable both features, set the following system properties:

- `mxe.webclient.async=1`
- `mxe.webclient.ClientDataValidation=1`

7.1.6 Control queries in left hand navigation

You can limit the number of queries that display for an application in the Side Navigation bar under "Available Queries" by using the `mxe.webclient.maxNavbarQueryLimit` system property. Limiting the number of queries that display improves application load time. For more information on this property, refer to Limiting the number of queries that display on the Side Navigation Bar in Maximo 7.6.

7.2 Cron tasks

A cron task is a task that is scheduled to run at a specified frequency, such as every 10 minutes or on a certain day or date at a certain time. The *Reorder* routine and *Preventative Maintenance* record generations are two examples of activities that use cron tasks. Cron tasks run in the background and require no direct end-user action after they have been configured.

If there are large numbers of cron tasks running at the same time there is a significant end-user load, response time for the end user can decline. Running cron tasks on separate servers from the user interface improves throughput and can help improve fault tolerance. You can run cron tasks on several nodes within a cluster or even on several nodes in different clusters. If any one of the nodes fails, the surviving nodes can still run the cron tasks.

7.2.1 Determining where to run cron tasks

To prevent cron tasks from running inside the end-user environment, refer to Disabling cron tasks on an application server.

By using mxe.crontask.donotrun and mxe.crontask.dorun instance properties, you can control which servers run (or do not run) specific cron tasks. By combining these properties with multiple cron task servers (JVMs), it is possible to balance your cron tasks as well as provide failover

should a server fail. Refer to Load Balancing / High Availability of Maximo 7.6 Cron Tasks using DoNotRun and NOW DoRun Properties for more information.

Give each cron task JVM a different name by adding `-Dmxe.name=<server name>` to the JVM generic argument list in each JVM. You can then use these JVM names as instance variables when setting the properties.

7.2.2 Configuring cron tasks

For the same cron task definition, you can create multiple cron task instances to increase performance. Ensure that the instances do not interfere with each other. For example, you can set up *reorder* cron task instances so that each instance handles a different site.

You must also determine and set the appropriate cron task frequencies. Cron tasks can consume considerable resources; running a job can put a drain on system resources and affect performance.

When you define multiple instances on the same server, set them up so that they do not all start and run at the same time. A simultaneous start or run of multiple cron tasks can adversely affect the performance of the server.

7.3 Escalations

Escalations are similar to cron tasks, except that escalations are configurable. However, escalations are very resource intensive when performing tasks. For more information on escalations, see Managing escalations.

Efficiency and frequency

An escalation selects a set of records and performs a list of one or more actions on the result set. The columns in the WHERE clause of the selection must be indexed efficiently.

Set the frequency or schedule of an escalation according to its importance, as follows:

- An escalation that dispatches emergency work orders for critical safety issues might warrant a frequency of five minutes.
- An escalation that ensures that service requests from executives are dealt with promptly might need to run every 15 minutes.
- An escalation that closes work orders that were completed 90 or more days ago might need to run only once a week at an off-peak time.

7.4 Maximo Integration Framework

The Maximo integration framework is (MIF) used to integrate Maximo with other systems.

Incorporation of both multi-threading and clustering greatly increases throughput in an integrated environment. Maximo employs a multi-threaded integration model. This integration framework model is set up to process multiple inbound transactions at once.

The following figure gives an overview of the integration framework.

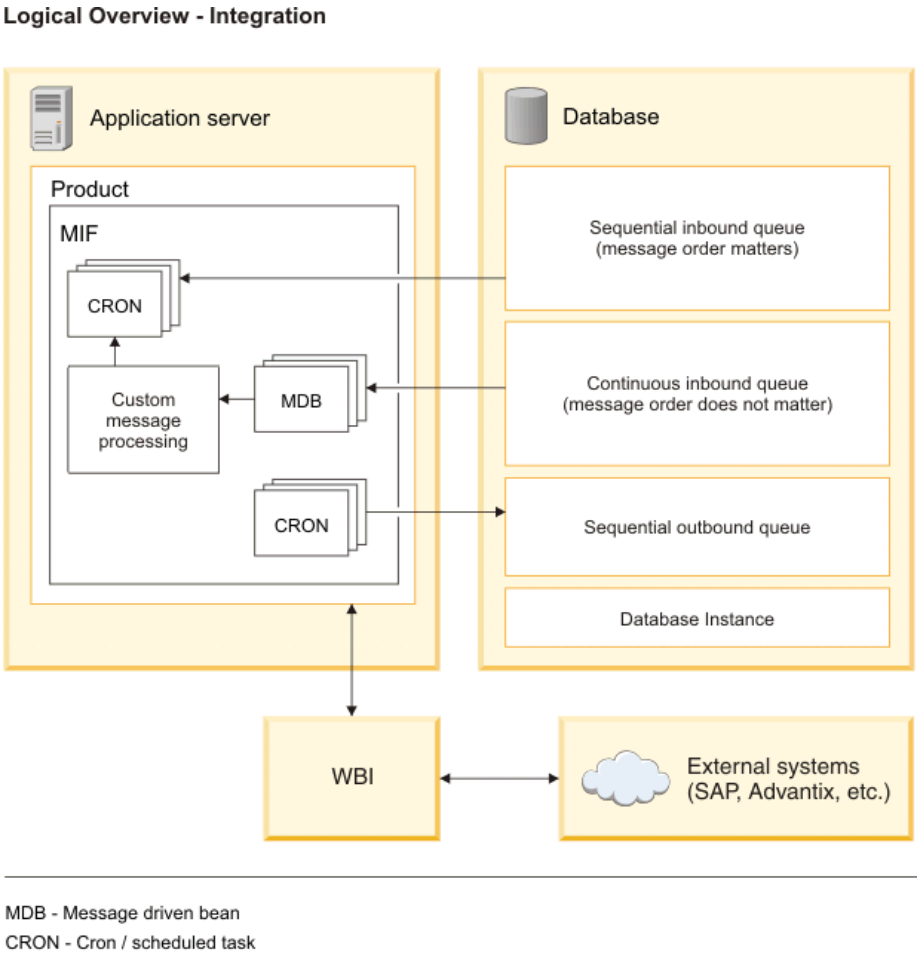


Figure 5: Integration framework overview

In an integrated environment, one of the main causes of falling performance is the demand of non-user-load processes by means of the integration framework.

The continuous queue functionality that the integration framework uses is resource-intensive when there is a large amount of queued data. Queuing can require significant memory and CPU cycles.

In a large, integrated system, you must tune the application and the overall environment parameters, including the parameters for the application server.

For more information on the integration framework, see Integrating data with external applications.

7.4.1 Cluster support

When you integrate Maximo with external systems, it is a best practice to set up multiple clusters to provide failover and load sharing. If a server instance (a cluster member) that is running processes or an application server fails, other server instances can continue to service requests and deliver responses. Distributing processing functions across multiple server instances (spanning one or many clusters), can increase performance.

In a typical implementation, Maximo is connected with multiple external systems through JMS queues. You configure the queues using the integration framework.

The purpose of clustering is to separate the inbound queue processing to a different cluster from the interactive users. Separating the queue this way prevents inbound traffic from affecting the user interface performance.

Set up separate sequential inbound and sequential outbound queues for every system with which Maximo communicates. Using multiple queues prevents a problem in a system from affecting traffic in other systems. If you also set up each sequential queue with its own cron task, administering the queues is easier.

While setting up a clustered environment with Maximo, configure the clusters so that the server that responds to customer requests, made by means of the user interface, runs in its own cluster. Configure other processes, such as JMS queue processing and other cron task operations, like VMMSYNC / LDAPSYNC, so these processes occur in a different cluster. An outage of the non-UI cluster does not immediately affect customers who are using the interface.

7.4.2 Integration load

If you are using the integration framework to work with other systems, from IBM or from other vendors, the activity generated by the integrated systems places an additional load on system resources. This additional load is over and above the load imposed by users accessing Maximo using the interface. To ensure that human users receive optimal performance, you can take steps to limit the resources consumed by the integrated systems.

To do this, you can limit the number of message-driven beans (MDBs) on the continuous queue of the integration framework. You can also improve performance by processing inbound integration traffic on a different server or cluster.

A default installation of Maximo does not have a specific number of MDBs in the deployment descriptor file. The Enterprise Java Bean container manages the number of MDBs that are created. Depending on the load on your system, you might need to reset this number. It is generally best to start with a small number, such as two MDBs.

For a small or medium-sized implementation, two or three MDBs usually provide acceptable performance. As the system load increases, you might need to gradually increase the number of MDBs. You generally must not exceed ten MDBs. More than ten MDBs does not necessarily yield higher throughput and it can significantly increase resource usage (CPU cycles in particular).

Test different numbers of MDBs in a development setting to determine an appropriate number before you establish the number of MDBs for your production system.

7.4.3 Exception-handling queues

Use exception-handling queues to prevent system performance degradation caused by bad data. Bad data in a queue can come in two forms:

- The data needs manual intervention to correct.
- The “bad” data becomes “good” data in a matter of time or sequence.

An example of bad data that can become good data is a purchase order that specifies a vendor that is not yet saved in the database. The purchase order can be processed after the vendor record is saved.

An effective way to deal with a high volume of this type of bad data is to establish an exception-handling queue. You can direct the bad messages from the continuous queue to the exception-handling queue after a preset number of retries in the continuous queue. Sidetracking bad messages prevents the continuous queue from consuming system resources to process bad data. The bad data can be processed in the exception-handling queue in a way that suits your needs.

The integration framework provides an out-of-the-box MDB that can be used for processing messages from the error queues. When error queues are implemented, the MDB section for the error queue must be uncommented in the file *ejb-jar.xml*, as well as in the file *ibm-ejb-jar-bnd.xmi* for the module *mboejb*.

7.4.4 Loop back exception queue design

One approach to addressing a high volume of continuous queue errors is to take advantage of the exception queue concept of WAS.

You can use the following design option:

- Create an exception destination named **A** for the continuous queue.
- For exception destination A, set **A** as its own exception destination,
- Then specify that **A** is the exception queue for the continuous queue.

This queue configuration causes error messages to loop back to the end of the error queue (A). A bad message is reinserted into a queue behind the message that allows the bad message to become a good message. This setup allows most errors that are based on bad timing to be resolved without manual error correction.

Setting the *mxe.int.mdbdelay* property to a value greater than N (where N>0) implies that each error message is being delayed by N milliseconds before being processed. For those N milliseconds, the MDB is holding the messages and is not doing any other work. Setting the delay to a high value can decrease error message processing performance. Delaying the error message processing helps good messages to be processed faster. This also prevents the system from being tied up in processing error messages that continue to arrive because the system is waiting for good data either from a self-correcting sequence or from a manual correction.

If you set up an exception queue to handle errors, the error XML files are saved in the continuous queue folder, not in the exception queue folder hierarchy.

7.4.5 XML/HTTP and SOAP-based web services for inbound processing

An important thing to consider while integrating your external system with Maximo is whether you can use XML/HTTP mechanisms instead of the Simple Object Access Protocol (SOAP) web services-based mechanism. XML/HTTP has less overhead and performs better under load conditions compared to SOAP. If your integration uses either XML/HTTP or SOAP, you are likely aware of the service that scales better for your environment.

7.4.6 Multi-record inbound XML message processing

Consider using a file import when importing XML documents containing multiple records. When an HTTP POST or a web service mechanism is used, the whole XML message is placed in the queue (if using the asynchronous path). The message is eventually processed by only one MDB (assuming that the continuous queue is used), which implies single-threaded processing of this large message. If, however, the file is loaded through the file import mechanism, the multi-record message is split into smaller XML messages, with each containing one record. Then, smaller messages are placed into the queue. This is more efficient because it allows multiple MDBs to pick up individual records and process them in parallel.

7.4.7 Sync operation with the Add action

Consider providing the action attribute value Add for the Sync operation while sending inbound data into Maximo. If the Add action is not used, the framework tries to find the record in the database and assumes it is an Update or Add, based on whether or not the record is found. If you provide the action as Add, the framework tries to create the objects directly, instead of searching for them. This action improves performance. Keep in mind that providing the action attribute value as Add can only be done if you are confident this data does not exist in the database.

7.4.8 Message tracking

Use message tracking to track a queue-based integration framework message. As with any reliable tracking mechanism, this comes with a performance overhead because all tracking points are logged in a database. Be aware of the performance impact before you implement message tracking.

7.4.9 Disable Retain MBOs

In the Publish Channel, there is a parameter called Retain MBO's that needs to be set to OFF to prevent performance and memory issues. With Retain MBO's enabled, the existing MBO is not discardable, so the application server will generate new MBOs for every record processed instead of using existing ones. As well as growing the MBO count, having this parameter enabled impacts java performance. Refer to Integration Framework performance tip--turning off Retain MBOs for more information.

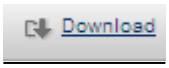
7.5 Reporting

Maximo provides an ad hoc reporting feature. This feature is available in many applications by clicking *Select Action → Run Reports*. Examine the ad hoc reports and options. Ad hoc reports allow you to create unique reports by selecting columns and choosing sorting and grouping options, as well as adding summaries to attributes and calculated fields. You can save report queries so you can use them again, without having to re-specify the query criteria. You can run reports on demand, or the reports can be scheduled. You can have the results emailed to any user and you can secure reports so only authorized users can run them. When you create a report, you choose the format of the output, by selecting from a list of popular file formats (for example, PDF, XLS, and XLSX).

Some customers customize Maximo to have applications display more information in the user interface than a standard installation displays. For example, by default, the **List** tab in the Assets application provides the following information for each asset:

- Asset (name)
- Description
- Location
- Loop Location
- Parent
- Rotating Item
- Is M&TE
- Linear
- Site

Customizations are performed by using the Application Designer, or by directly modifying the XML for an application. The usual intent of the customizations is to provide a convenient set of information that can easily be downloaded as an XLS file, by using the "Download" feature.



However, these customizations can create an unnecessary drain on system performance because every time the **List** tab is displayed, the added information must be retrieved from the database. Instead of customizing applications to display additional information, use ad hoc reports to create the information you need. Using the ad hoc reports reduces the impact on system performance because the additional information is retrieved by means of an ad hoc report only when it is needed; not each time the application is opened.

Access the Maximo Report Wiki Pages or the Maximo Report Reference Materials for more information.

Reporting replica database

Using reporting databases can reduce the load on the client's live, transactional database and can lead to performance improvements. If these reports are scheduled, additional processing time can be off-loaded from the server. These performance improvements can be especially noticeable for complex reports. This separate, external database may be a snapshot of the production, transactional database, and is sometimes referred to as an external, replicated, or reporting database.

Isolating reporting

The use of application reports can also put a large burden on the user-interactive applications. To remove the impact of reports on user interface performance, run reports that require significant system resources on a separate report server. You can also schedule reports to execute from a cluster separate from the UI cluster.

To help the overall system performance, run only simple reports from the user-interactive application.

Report queries

Review custom reports for efficient SQL and use of indexes. Most reports receive a WHERE clause from Maximo. In all these cases, improving the efficiency of user queries also improves the efficiency of reports.

Report Limits and Notifications

To minimize performance impacts of users inadvertently running reports against very large record sets, you can specify record limits. Record limits can be set at the individual report, or at the security group level for users either creating ad hoc reports or executing reports. These limits will either prevent the very large report from executing, or cancelling it when it reaches a certain value.

Additionally, you can configure notifications to be sent to your administrator when reports are processing either for a very long time, or for reports that may be hung in the queue.

For more details on these features, access the Maximo 7.6 Report Performance Considerations.

8 Server operating system configuration

This chapter discusses tuning that can be applied at the operating system (OS) level on application and on database servers to optimize performance of Maximo. While these settings are recommended for optimal performance, you should consult with your systems and network administrators before using these settings to ensure they are compatible with any standards in place for your organization.

8.1 AIX operating system settings

The sections that follow provide AIX operating system settings that you might want to use. These settings must be applied to the WebSphere userid on application servers and on the userid for the database instance owner on the database server.

8.1.1 Networking settings

Refer to TCP streaming workload tuning for guidelines specific to your network type for tuning the following parameters:

- tcp_recvspace
- tcp_sendspace
- rfc1323
- MTU path discovery
- tcp_nodelayack
- sb_max
- Adapter options, such as checksum offload and TCP Large Send

Refer to UDP tuning for guidelines on tuning

- udp_sendspace
- udp_recvspace
- UDP packet chaining
- Adapter options, like interrupt coalescing

In addition, Maximo recommends setting the following networking settings:

```
/usr/sbin/no -r -o ipqmaxlen=250
/usr/sbin/no -r -o clean_partial_conns=1
/usr/sbin/no -r -o tcp_keepidle=600
/usr/sbin/no -r -o tcp_keepintvl=10
/usr/sbin/no -r -o tcp_keepinit=40
/usr/sbin/no -r -o tcp_timewait=1
/usr/sbin/no -r -o tcp_finwait2=60
/usr/sbin/no -r -o tcp_ephemeral_low=1024
/usr/sbin/no -r -o tcp_nodelayack=1
```

Refer to TCP and UDP performance tuning for more information.

8.1.2 Resource (ulimit) settings

Use the following resource (ulimit) settings:

```
time(seconds)        unlimited
file(blocks)         unlimited
data(kbytes)         unlimited
stack(kbytes)        4194304
memory(kbytes)       unlimited
```

```
coredump(blocks)      unlimited
nofiles(descriptors) unlimited
threads(per process) unlimited
processes(per user)  unlimited
```

8.1.3 Process settings

Use the following process settings:

```
chdev -l sys0 -a maxuproc='4096'
```

8.1.4 Virtual memory settings

Use the following settings for virtual memory:

```
vmo -p -o lru_file_repage = 0
vmo -p -o maxclient% = 90
vmo -p -o maxperm%=90
vmo -p -o minperm%=5
```

8.1.5 Processor Affinity on Power7, Power7+, and Power8

Internal testing has demonstrated significant performance improvements when utilizing processor affinity on P7, P7+, and P8 processors. However, there are limitations when using affinity:

- The best use of affinity needs to consider all of the applications running on a system to optimally partition the resources among the applications
- The LPAR must have dedicated resources (cores, memory)
- You must use MCM as the memory affinity policy
- Testing used 2 cores per JVM binding

You can use the following commands to start a JVM using processor affinity:

- `export MEMORY_AFFINITY=MCM`
- `execrset -F -c <CPUlist> -m <MEMlist> -e <WAS_HOME>/AppServer/profiles/<profile_name>/bin/startServer.sh <cluster_member_name>`

You would then repeat the `execrset` command for each JVM you want to start using affinity.

8.2 RedHat Linux configuration

Default settings for tcp and udp tunables on Linux are good and recommended by Maximo. However, a few settings should be modified for best performance:

For networking, use the following settings:

```
sysctl -w net.ipv4.ip_local_port_range="1024 65535"
```

For resources (ulimits), use the following settings:

```
max user processes   (-u)  8192
open files           (-n)  8000
```

For shared memory on DB2 servers, refer to Modifying kernel parameters (Linux).

For shared memory on Oracle servers, refer to Configuring Kernel Parameters and Resource Limits.

8.3 Windows configuration

Set the following networking parameters, located under the Windows registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters:

```
TcpTimedWaitDelay           dword:0000001e (30)
StrictTimeWaitSeqCheck      dword:00000001 (1)
MaxFreeTcbs                 dword:00011940 (72000)
MaxHashTableSize            dword:0000ffff (65535)
TcpWindowSize               dword:0000ffff (65535)
EnableDynamicBacklog        dword:00000001 (1)
MinimumDynamicBacklog       dword:00000032 (20)
MaximumDynamicBacklog       dword:000003eb (1000)
DynamicBacklogGrowthDelta   dword:0000000a (10)
Interfaces\TcpAckFrequency dword:00000001 (1)
```

If some of the above parameters are not currently in the registry, they may be added by following the instructions at Add a Value to a Registry Key Entry.

For Windows Server 2008 and above, the default dynamic port range has changed. The new default start port is 49152 and the default end port is 65535. Therefore, 16384 ports are available by default (not 5000).

To view the dynamic port range, start the Command Prompt and use the netsh command, as follows:

```
netsh int ipv4 show dynamicport tcp
```

To change the dynamic port range for the maximum number of ports allowed, issue the following command:

```
netsh int ipv4 set dynamicport tcp start=1025 num=64510
```

Note that the minimum start port is 1025 and the maximum end port cannot exceed 65535.

9 Maximo Multitenancy Considerations

Deployment architecture and performance tuning recommendations from earlier sections of this document still apply to a multi-tenant Maximo system; however, compared to a typical non-multi-tenant system, a multi-tenant deployment tends to be larger in concurrent user count, transaction data volume, and database size and demands better scalability.

Concurrent user counts and user load characteristics are still the most important factors that drive the application server performance and capacity. The cumulative SQL transactions and data volume size contribute to the database performance. Therefore, the application server and database tuning practices remain the same.

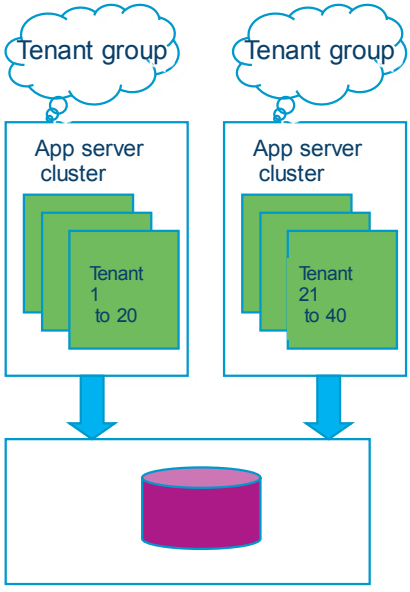
However, there are a few characteristics specific to the multi-tenant system that have an important role in planning for performance.

9.1.1 Memory footprint of serving a tenant

A given application server can serve the requests from multiple tenants. Each tenant will have a certain amount of memory footprint on the app server. The more tenants an application server will receive requests from, the bigger the footprint.

It is recommended to allocate a slightly larger heap size compared to a non-multi-tenant system in order to compensate for the tenant memory footprint if you intend to support greater than 10 tenants on one server. Benchmark testing shows that with the same amount of memory, a 10% performance degradation in terms of maximum concurrent users the application server can support is seen if requests are from 20 tenants comparing to from just one tenant. However, the footprint for a tenant will be determined by how much customization a tenant has done and the size of its local cache, such as site or organization cache, of the tenant. Therefore, the actual number may vary.

You can also group the application servers so that a group of application servers will handle user requests from only a subset of tenants in your system. This limits the memory footprint on each server, thus minimizing the performance impact, and still allows easy scaling when the user load increases or new tenants are on boarded. The grouping of user traffic can be done by reverse proxy servers:



The same approach applies to cron task servers, MIF servers, and report servers.

9.1.2 Cron tasks

For the same number of concurrent users, a multi-tenant Maximo system will have a larger number of cron tasks running in the system, collectively, compared to a non-multi-tenant system. Each tenant has its own set of basic system cron tasks and they can have their own cron task instances. Another challenge in the managing the cron tasks in a multi-tenant Maximo system is that the cron tasks are owned by the tenant but the resources to run the tasks are managed by the system provider.

It is essential to have a separate set of cron task servers allocated apart from the UI cluster. The capacity will need to be adequate, and the load-balancing feature introduced in 7.6 needs to be used so that the system can absorb the changes of cron task load by tenants. This is especially important because the system administrator who is in charge of system resources is not directly informed of tenants' changes to their cron tasks.

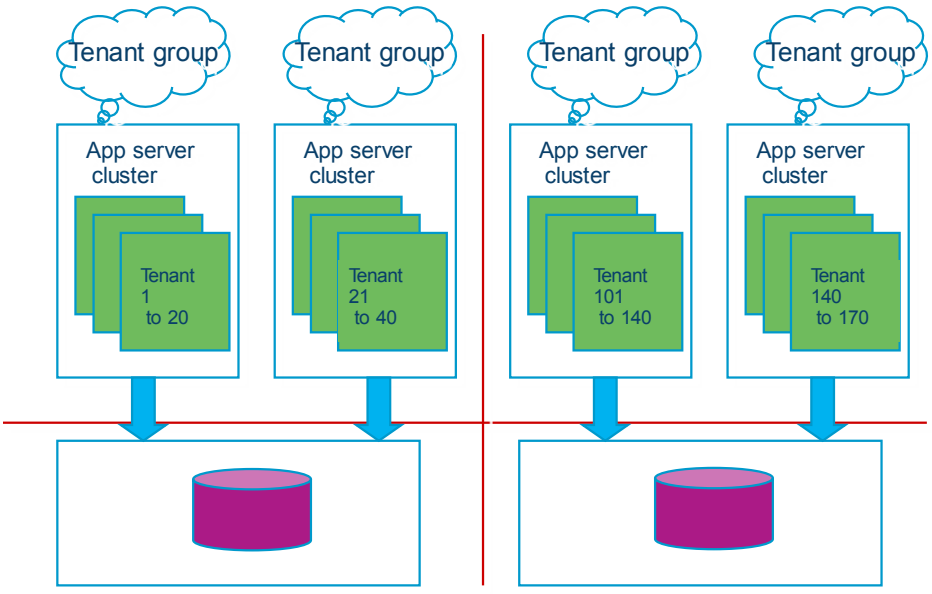
The server performance of the cron task servers need to be monitored, as well as the `CRONTASKMONITOR` table, which contains the information of all cron tasks' schedules across all tenants. It is also a good practice to advice your tenants to schedule their cron tasks at different starting times. In addition, group your tenants of different time zones onto the same set of servers. The *mxe.crontask.runtenants* property can be used to group the tenant's cron task load onto different servers.

9.1.3 Increased database requirements

Multitenancy requires the following minimum versions of DB2: 10.1 with Fixpack 5 or 10.5 with Fixpack 4.

The database in a multi-tenant system can become very large. This makes archiving and purging the data a priority.

Additionally, if the data volume or the concurrent transaction count becomes very high, the database can become the bottleneck of the system. Separating the database by grouping the tenants can be an alternative if the limit is expected. This essentially breaks the systems into multiple deployments. However since the database schema and application server set up are identical, this will still keep the administration cost low. Make sure the systems are updated at the same time and the configurations are kept identical.



Since the tenant usage of the system is not directly visible to the system provider, the system needs to be closely monitored through database performance reports and application logs

showing any long running SQL statements. The database needs to be constantly tuned by adjusting indexes and communicating to the tenants.

9.1.4 System properties

There are some system properties related to performance that should be carefully considered. They can be used as a way to prevent a tenant from causing a large memory footprint and long running SQL queries on the system. For example:

- mxe.db.QueryTimeout or webclient.ResultSetQueryTimeout
- mxe.db.fetchResultLogLimit
- mxe.usermonitor.timeout
- webclient.maxdownloadrows

The values are recommended to be set.

Many properties can be set at the tenant level. It is a good practice to use value rules of the property to allow only the tenant to increase or decrease the value in order to protect the system. For example:

- mxe.assettopology.depth
- mxe.int.doclink.maxfilesize

You can set *mxe.assettopology.depth* to DECONLY to prevent the tenant from having too deep of a hierarchy.

For information on these properties, refer to System Properties in the Knowledge Center.

10 Maximo Anywhere Considerations

This section provides best practices guidelines to improve the performance of IBM Maximo Anywhere 7.5.x or greater. While these guidelines provide optimal performance in the lab test environment, your environment may require different settings. The settings in this section can be used as a guideline or as a starting point, and then monitored and tuned to your specific environment.

10.1 MobileFirst Server tuning

This section discusses settings that can optimize Worklight on IBM WebSphere Application Server. In a production environment, Worklight server must be installed on a 64-bit operating system with all software at 64 bit. We recommend running Worklight in a clustered environment, configured similarly to the Maximo Application Server(s). For further information on tuning Worklight, see Optimization and tuning of MobileFirst Server.

Note: Starting with version 6.3 of MobileFirst Server, HTTP compression is supported and enabled by default.

10.1.1 IBM WebSphere Application Server tuning

This section discusses settings that can optimize IBM WebSphere® Application Server (WAS) Version 7 or later. For further information on tuning WAS, see WebSphere Application Server Performance. It is important to tune the application server at the operating system level. More information on tuning operating system parameters for WAS can be found in the Server operating system configuration section of this document.

While the settings below provide optimal performance in test environments, your environment may require different settings. You can use the settings below as a guideline or as a starting point, and then customize the settings to your specific environment requirements.

JVM heap size values

You can use the following heap size values to start tuning:

Initial: 4096m / Maximum: 4096m

Note: Worklight best practices recommend 8096m as the maximum heap size. Increasing the heap size may be necessary depending on your load.

Generic JVM arguments

You can tune the JVM settings in WAS using the WAS Administrative Console. After logging into the console, select *Servers → Server Types → WebSphere application servers → <server_name> → Process definition → Java Virtual Machine*.

You can use the following JVM arguments to start tuning:

-Xdisableexplicitgc -Xgcpolicy:gencon -Xmn1024m -Dsun.rmi.dgc.ackTimeout=1000 -Djava.net.preferIPv4Stack=true -Xcodecache32m

Note: In the JVM argument string above, **-Xmn1024m** is the **Nursery Size** representing 25% of maximum heap size. If you increase your heap size, remember to increase this value accordingly.

Thread pool tuning recommendations

You can tune the JVM settings in WAS using the WAS Administrative Console. After logging into the console, select *Servers → Server Types → WebSphere application servers → <server_name> → Thread pools*.

Thread Pool Name	Minimum Size	Maximum Size	Thread Inactivity Timeout
Default thread pool	20	50	30000
WebContainer thread pool	120	120	60000

Table 2: Thread pool settings

10.1.2 IBM HTTP Server Tuning

You can use the settings below to optimize IBM HTTP Server Version 8.5 or later. While these settings provide optimal performance in the test environment, your environment may require different settings. You can use the settings as a guideline or as a starting point and then monitor and tune the settings to your specific environment.

You can tune the IBM HTTP Server settings by modifying httpd.conf. You can use the following values to start tuning:

- TimeOut: 900
- KeepAliveTimeOut: 60
- MaxKeepAliveRequests: 0
- MaxRequestsPerChild: 0
- ThreadLimit: 2400
- ThreadsPerChild: 2400

10.2 Maximo Anywhere OSLC Server tuning

This section discusses settings that can optimize Maximo Mobile OSLC Server.

10.2.1 IBM WebSphere Application Server tuning

Please follow the steps in Section 3: IBM WebSphere Application Server tuning of this document.

10.2.2 IBM HTTP Server tuning

Please follow the steps in Section 4: IBM HTTP Server and Plugin tuning of this document.

10.2.3 Database tuning

Please follow the steps in Section 5: Database server tuning and maintenance of this document. If you are using MobileFirst Server for other applications in addition to Maximo Anywhere, you can refer to Optimization of MobileFirst Server project databases for more information.

10.3 Maximo Anywhere application settings

This section discusses settings that apply to restricting data downloads and user / group security settings for Maximo Anywhere Work Execution and Work Approval applications.

10.3.1 Restricting the amount of data downloaded on the device

Maximo Anywhere Asset Audit, Asset Data Manager, Inspection, Issues and Returns, Physical Count, Service Request, Transfers and Receiving, Work Approval, and Work Execution applications can download a significant amount of data on the device. The amount of data that is downloaded can be restricted using System Properties, by modifying the application's XML file, or by using the Anywhere Administration Application.

System Properties

You can configure system properties for each object structure part of the maximo OSLC implementation to restrict the amount of data that is downloaded. The system properties use the following naming convention and can be added / updated using the System Properties application:
`mxe.oslc.<objectStructureName>.fetchlimit = <number>`

Following is an example of how to set these restrictions:

- Open the browser and launch the Maximo Login page
- Login as the Maximo administrative user
- Go to the System Properties application and create the following properties. Use your own values as appropriate, for example:
  - `mxe.oslc.oslcasset.fetchlimit = 50000` (for Asset Record)
  - `mxe.oslc.oslcinvbalances.fetchlimit = 1000` (for Bin)
  - `mxe.oslc.oslcinventory.fetchlimit =1000` (for Inventory)
  - `mxe.oslc.oslclabor.fetchlimit =100` (for Labor)
  - `mxe.oslc.oslclaborcraftrate.fetchlimit = 1000` (for Labor Craft Rate)
  - `mxe.oslc.oslcoperloc.fetchlimit = 5000` (for Operation Locations)
  - `mxe.oslc.oslcperson.fetchlimit = 250` (for Person)
  - `mxe.oslc.oslcpremiumpay.fetchlimit = 1000` (for Person Premium Pay)
  - `mxe.oslc.oslcitem.fetchlimit = 5000` (for Item)
  - `mxe.oslc.oslctoolitem.fetchlimit = 500` (for Tool)
  - `mxe.oslc.oslcalndomain.fetchlimit = 5000` (for ALN Domain)
  - `mxe.oslc.oslcfailurelist.fetchlimit = 20000` (for Failure List)
  - `mxe.oslc.oslclocations.fetchlimit = 50000` (for Locations)
- Save the changes
- Select changed properties, and Click Live Refresh

Mobile Application XML (app.xml)

Mobile application XML file can be modified to restrict the amount of data downloaded (and stored in the offline cache) using queries / where clauses with "variables" to retrieve information that is specific to the current user and location. You would define the where clauses for the resources in the <data> section of the XML file.

Note: Each Maximo Anywhere application has its own app.xml.

Following is an example of a where clause for Crew Labor and Tool resource used to create follow on labor and tools:

```
<!-- Crew Labor and Tool resource used to create follow on labor and tools -->
       <resource class="application.business.CrewLaborObject"
additionalData="true" isSystem="true" pageSize="10" name="laborcrew"
describedBy="http://jazz.net/ns/ism/asset/smarter_physical_infrastructure#Amcre
w" providedBy="/oslc/sp/SmarterPhysicalInfrastructure">
              <attributes>
                     <attribute name="crewid" index="true"
describedByProperty="oslc:shortTitle"/>
              </attributes>

                     <whereClause
clause="oslc:shortTitle=${mylabor.amcrewOrEmpty} and spi:status in
[${domaincrewstatus[maxvalue=ACTIVE].value}]"/>
```

OR

```
                <queryBases>
                      <!-- list of query bases the app will rely upon for
this resource , name of the queryBases are configured on server-->
                      <queryBase name="getMyCrewLabour /> 
                </queryBases>
</resource>
```

Anywhere Administration Application

The Anywhere Administration application may be used to change the default query for an application. For more information, refer to Enabling the Anywhere Administration application and Modifying apps with the Anywhere Administration application.

10.3.2 Security and Groups

The Maximo Anywhere applications require the user to be part of the ANYWHERE_TECHNICIAN, ANYWHERE_APPROVER, ANYWHERE_PHYSICAL_COUNT, and ANYWHERE_INSPECTOR groups.

Note: Changes made to users/security groups are not committed if that user has MX sessions open. This means that if a user has existing sessions open, changes made related to his security group profile will not take place until all existing server sessions are closed.

10.3.3 Connectivity Timeout

If you see an error in the logs that looks similar to the following:

```
Caused by: java.net.SocketTimeoutException: Read timed out
at java.net.SocketInputStream.socketRead0(Native Method)
at java.net.SocketInputStream.read(SocketInputStream.java:152)
at java.net.SocketInputStream.read(SocketInputStream.java:122)
at
org.apache.http.impl.io.SessionInputBufferImpl.streamRead(SessionInputBufferImpl.ja
va:136)
at org.apache.http …
```

Your connectivity timeout should be increased to at least 60000 for this large of a scale of data.

To increase this setting, edit the *MaximoAnywhere/adapters/OSLCGenericAdapter OSLCGenericAdapter.xml* file in the Anywhere build tree and set the following:

```
<connectionTimeoutInMilliseconds>60000</connectionTimeoutInMilliseconds>
```

10.4 Device Considerations

Maximo Anywhere Release 7.5 supports Android, iOS, and Windows phone and tablet devices.

The most important factor affecting the user performance experience is the processing power of mobile device. The second most important factor will be the network speed (i.e. 3G, 4G, 4G LTE and Wi-Fi).

Before choosing an Android device, please refer to the Primatelabs Android Benchmark as a reference. For iOS devices, please refer to the Primatelabs iOS Benchmark.

We recommend running Maximo® Anywhere applications on the devices that have benchmark scores above 1300. The applications may execute, but run poorly on those devices that have a lower score.

Devices with benchmark scores between 1300 and 2000 are considered average speed devices by today's standards. Devices with benchmark scores exceeding 2000 are considered fast devices.

11 Client workstation configuration

Customers report that client workstation configuration is, initially, the most important area to focus on when users experience performance issues.

Hardware and software products constantly evolve. Maximo can run on some older hardware and software platforms. The best performance is achieved by using the newest supported operating systems and the latest hardware. Robust workstations provide better performance. For example, client workstations must have, at least, the minimum RAM required by the operating system and by all other applications that it runs, but adding more RAM can boost performance. Similarly, using workstations with multiple CPUs and higher clock speeds also boosts performance.

For more information on supported client workstations, see the Product Configuration Matrix.

Limit or prevent some workstation activities and processes

Customers report that some user activities and workstation processes can degrade performance. Check for and monitor these activities and processes, and respond as necessary.

Monitor the network for streaming audio and video

Customers report that monitoring the network to prevent users from using streaming audio and video can noticeably increase the bandwidth available to the system.

Have only one network link active

If a user has both a wireless network link and an active LAN link, it can cause system performance issues. Limit users to one active network link.

Monitor for hung processes and applications

Processes and applications that are not responding (are hanging) use memory and can affect the performance of the client workstation. A workstation can have hung processes or applications of which the user might be unaware.

Use system tools, such as the Windows Task Manager, to check for and end hung processes and applications.

12 Troubleshooting and monitoring performance

Maximo have features that log health statistics. These features and their associated data help you troubleshoot and monitor for performance problems.

12.1 Troubleshooting performance problems

You can use the troubleshooting procedures in a development or test environment for performance analysis and debugging. The procedures generally must not be used in a production environment. Use these procedures in a production environment only if you cannot isolate the problem in a test environment.

To help narrow down the issue, consider the following questions:

- Are they slow all of the time?
- Only sometimes?
- Only when doing certain functions?
- Only when accessing from certain sites?

The answers to these questions can help determine what type of issue may be occurring.

12.1.1 Setting up a standalone application server for debugging

Debugging SQL or business object problems and using detailed logging when you reproduce other problems can generate large logs and slow down the system.

You can set up a standalone application server for debugging in production. However, if issues that you are investigating cannot be replicated in a test environment, setting up standalone loggers can provide limited benefit.

Set the parameters so that the server does not execute cron tasks. The standalone application server can be on a separate computer. Using a separate computer lets you easily stop and start the server to change logging parameters and reproduce problems with a single user.

12.1.2 WebSphere Performance Monitoring Infrastructure (PMI)

When tuning the WAS for optimal performance, it is important to understand how the various run time and application resources are behaving from a performance perspective. PMI provides a comprehensive set of data that explains the runtime and application resource behavior.

Using PMI data, the performance bottlenecks in the application server can be identified and fixed. PMI data can also be used to monitor the health of the application server.

For more information on using PMI, see Enabling PMI data collection.

12.1.3 Debugging parameters

You can use debugging parameters with Maximo. You can configure the debugging parameters in the **System Properties** application, as follows:

- `mxe.db.fetchResultLogLimit=200`
- `mxe.db.logSQLTimeLimit=1000` (default)
- `mxe.mbocount=1` (default)

// NOTE: Oracle Only below - Impacts performance

- `mxe.db.logSQLPlan=true`
- `mxe.db.sqlTableScanExclude ACTION,MAXROLE,SCCONFIG,MAXUSER`

Maximo recommends setting the first three properties along with setting the root logger to INFO at all times to make performance debugging simpler with little to no overhead on the system.

For more information on using the parameters, see Using debug properties to monitor and troubleshoot performance.

12.1.4 Logger objects

Logging is performed in Maximo using a logger object. Loggers define a hierarchy and provide the run-time control to print statements or not. You make changes in the **Logging** application, located at **System Configuration → Platform Configuration**. You must be logged in as a user with administrative privileges to modify the logging, apply, and save the changes.

For Maximo, keep the `maximo root logger` and `log4j root logger` at their default out-of-the-box values (INFO) for optimal performance. Lower settings will reduce logging overhead. However, the default settings are best for performance and for obtaining enough information to monitor the health of the system.

In addition, keep the other sub-loggers to the ERROR level for optimal performance.

When debugging slow performing SQL statements, you can set the SQL logger, `log4j.logger.maximo.sql`, to the INFO level. Make sure to set it back to the ERROR level when debugging is complete as the INFO setting has performance implications.

For more information on the logging functionality in Maximo, see Logging in Maximo.

12.1.5 Displaying garbage collection statistics on the server

You can turn on garbage collection (GC) verbosity to find out how the garbage collection routine is functioning on the application server.

For WAS, specify the parameter in the administration console by selecting *Servers → Server Types → WebSphere application servers → <server_name> → Process definition → Java Virtual Machine* and placing a check mark by verbose garbage collection.

After you set garbage collection to verbose, the console or the log file displays how much time is spent for each garbage collection cycle.

Several tools are available to assist with garbage collection analysis, such as the IBM Pattern Modeling and Analysis Tool for Java Garbage Collector and the IBM Support Assistant Tool.

12.1.6 Apply the latest patches

Apply the latest available patch, fix pack, or hot fix for your release of Maximo. Patches and hot fixes contain fixes for application issues and often contain features that can improve system performance.

Patches also contain new features and parameters that help you to better monitor and debug system performance issues.

12.1.7 Staged rollout for new users

The first time that a user logs in to Maximo, there is an initial setup of the user's Start Center. This setup results in many database queries and inserts. If too many users perform this initial login simultaneously, all available database resources could be consumed by the setup operations. Therefore, in large deployments, new users can be brought on board in a staged manner to avoid this bottleneck.

12.1.8 Start Center portlets

A large number of portlets on the Start Center can contribute to slow performance. Use the Start Center tabs so that a limited number of portlets are displayed on any given tab. Some customers have suggested using default Start Centers without any portlets to help alleviate this issue.

12.1.9 Limiting use of eAudit

eAudit is a feature that is designed to support industries, such as life sciences, that are required to track changes by user, date, and time. Because eAudit creates a transaction record for every change in the system, it impacts system performance. You can weigh the benefits of enabling eAudit against the impact on performance.

For more information on eAudit, see Configuring databases in the IBM Maximo Asset Management information center.

By default, when you enable eAudit for an object, such as the Work order object, all fields are eAudited. You might want to limit the number of objects that you eAudit. For each object that is eAudited, limit the number of fields that you eAudit. Limiting the use of eAudit to only those objects and fields that require an audit trail limits the impact on system performance.

12.2 Performance problem determination

If possible, do load testing during the implementation phase to expose performance problems before you put Maximo into production.

If you have the equipment to perform load testing, you can use load testing after Maximo is in production to determine if there is any performance impact from patches or from data growth over time.

The Performance Test Best Practices With HP LoadRunner and Performance Test Best Practices for RPT papers illustrate techniques for performance testing your deployment of Maximo.

12.2.1 Problem determination techniques

Application server

WAS logs must be reviewed for any errors. For load-balanced implementations, attention should be paid to the distribution of users across the JVMs. Monitor the JVM memory utilization on the application server by enabling verbose garbage collection.

WAS logs

SystemOut.log and *SystemErr.log* – any application errors, long running SQL queries, and so on, can be found in these logs.

Native_system_err.log – verbose garbage collection information will be in this log if verbosegc is enabled.

http_plugin.log – can be reviewed for any load balancing issues in a clustered environment

WAS configuration

Ensure that the JVM heap size is set adequately and that heap exhaustion does not occur.

Ensure that thread pool sizes for the Default, and WebContainer thread pools are set to appropriate levels.

Connection Pool Watchdog (DB Connection Leaks)

Enable the Connection Pool Watchdog to monitor the database connection pool to ensure that connection leaks are not occurring by setting `log4j.logger.maximo.dbconnection` to **`INFO`**.

Web server

Web server logs must also be reviewed for errors. View the maximum connections and total connection attempts to see if your web server can use as much of the connection as it needs. Compare these numbers to memory and processor usage figures to determine whether a problem is caused by a connection, and not by some other component.

On IBM HTTP Server, the logs of interest are access.log, admin_access.log, admin_error.log, error.log, and http_plugin.log.

It may be necessary to raise the *ThreadsPerChild* setting in the *httpd.conf* file.

Database server

Analyze the `maxsession` table and the number of database connections to verify that session activity is as expected.

Database server memory and instance memory must also be monitored. Database traces and snapshots can be gathered to assist with database tuning issues. See DB2 Query Tuning and Index Creation and Tuning and Indexing Oracle Databases for more information.

Network

Monitoring the bytes per second processed by the network interface card can also give insight to potential network overload.

If you need more detailed network information to understand bandwidth requirements, bandwidth-monitoring tools provide the ability to analyze HTTP requests, the number of round trips between tiers, and TCP/IP packet information.

CPU

Monitor the CPU to ensure that all processors are being used as expected and that overall CPU utilization remains healthy.

Memory

Be careful about monitoring total memory usage. For Maximo, JVM heap size is the most important memory metric to monitor on application servers.

12.3 Problem determination tools

The following tools can be used to assist with performance problem determination of your Maximo.

Performance Analyst Suite

The IBM Performance Analysis Suite (PerfAnalyst) is a tool intended to make performance analysis of Maximo easier. Some features include:

- Importing software / middleware configuration for checking against the settings outlined in this document. Some examples include configuration of DB2, Oracle, WebSphere, operating systems, and system properties.
- Importing performance data for analysis, e.g., DB2 snapshots and Oracle AWR reports, Java verbose GC output, and Java thread dump output.
- Providing a rules engine to define alerts to detect symptoms from the data. An out-of-box pre-defined set of rules are provided, based on the performance best practices outlined in this document.

Click the link above to learn more about downloading, installing, and using the tool.

Maximo Activity Dashboard (PerfMon)

The Maximo Activity Dashboard (also called PerfMon) is a built in tool that assists in analyzing long running SQL and server side processing time. For instructions on enabling and starting the activity dashboard in version 7.6.x, see Enabling Maximo Activity Dashboard.

Heap dump, thread dump, and garbage collection utilities

The following tools can be used to debug Java code:

- HeapAnalyzer – Allows the finding of a possible Java heap leak area through its heuristic search engine and analysis of the Java heap dump in Java applications. This tool supports both IBM heap dumps and Weblogic hprof format heap dumps.
    - For information on collecting heap dump files for WAS, see Generating heap dumps manually.
- IBM Thread and Monitor Dump Analyzer for Java – Analyzes javacore files and diagnoses monitor locks and thread activities to identify the root cause of hangs, deadlocks, and resource contention, or to monitor bottlenecks.
    - For information on collecting thread dump files for WAS, see "*To force a thread dump*" in Web module or application server stops processing requests.
- ThreadLogic – Analyzes thread dumps produced by Weblogic using the Oracle JDK.
- IBM Pattern Modeling and Analysis Tool for Java Garbage Collector – Parses verbose GC trace, analyzes Java heap usage, and provides key configurations based on pattern modeling of Java heap usage.
- Eclipse Memory Analyzer – Helps you to find memory leaks and to reduce memory consumption.
- IBM Support Assistant Tool – Provides a workbench to help you with problem determination.

Application profiling utilities

The following tools can be used to profile and debug Java code:

- Health Center – a GUI-based diagnostics tool for monitoring the status of a running Java Virtual Machine (JVM).
- Performance Inspector – Contains suites of performance analysis tools to assist in understanding the performance of applications and the resources they consume.
- YourKit – A CPU and memory Java Profiler that supports J2EE/J2ME.
- OProfile – A system-wide profiler for Linux systems and can profile any running code with low overhead.

Database Utilities

Each of the database platforms contains tools that can be used to analyze database health and SQL queries to assist with any long-running SQL statements. Refer to database documentation for more details. Some of the more useful tools include:

- AWR/ADDM Reports
- IBM DB2 monitors and snapshots
- IBM Data Studio
- IBM standalone db2advis and execution plans

12.4 Monitoring the system

Ongoing monitoring of your system can prevent performance issues from arising in the first place. Have a monitoring strategy in place before you put Maximo into production.

Monitoring the IBM Maximo Platform provides an overview of how the Cloud & Smarter Infrastructure monitoring portfolio can be used to monitor your environment.

12.4.1 Monitoring tools

The following tools can also be used to monitor your Maximo.

Middleware monitoring tools

The following monitoring tools can be used to monitor the middleware components associated with Maximo:

- Monitoring Performance with Tivoli Performance Viewer – Enables monitoring of the overall health of WAS from within the administrative console.
- Database Monitoring – Each database platform contains tools that can be used to monitor the database and to provide useful information.

System resource monitoring tools

The following tools can be used to monitor system resources while performing tests:

- PerfMon – Can be used to gather performance metrics on Windows-based systems. It is part of Windows and provides access to all of the Windows performance counters.
- nmon – Can be used to gather performance statistics on AIX- or Linux-based systems. Nmon for AIX is included with AIX from 5.3 TL09, AIX 6.1 TL02, and Virtual I/O Server (VIOS) 2.1, and is installed by default. Versions of nmon for previous versions of AIX and more information on using the tool can be found on IBM developerWorks. nmon for Linux is released to open source and is available from the nmon for Linux wiki.
- rstatd – Gathers performance metrics from a UNIX system kernel. rpc.rstatd is shipped with AIX and can be downloaded from the rstatd 4 Linux site for Linux platforms.
- sysstat – A package of utilities containing the sar, sadf, iostat, mpstat, and pidstat commands for AIX and for Linux. sar provides system information related to I/O, memory, paging, processes, network, and CPU utilization. sadf formats data collected by sar. iostat gives CPU and I/O data disks and TTY devices. pidstat reports process data, and mpstat reports global and per-processor data.
- vmstat – Reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity on UNIX systems.

Bandwidth / Throughput monitoring tools

The following monitoring tools can be used to monitor network and HTTP bandwidth while performing tests:

- HttpWatch – Logs and allows inspection of HTTP and HTTPS traffic, and can be used to understand the requests/responses between a browser client and a web server for bandwidth analysis.
- Wireshark – A network protocol analyzer that can be used to capture network traffic for bandwidth analysis.
- WinPcap – A packet capture and filtering engine used by many network protocol analyzers, including Wireshark.

APPENDIX A: ORACLE WEBLOGIC CONSIDERATIONS

While this paper concentrates on tuning and monitoring and troubleshooting Maximo running on WAS, Oracle WebLogic Server 12c is also a supported application server environment.

If you are running WebLogic instead of WAS for Maximo, see Performance and Tuning for Oracle WebLogic Server provided by Oracle for general tuning guidance. Note that specific settings, such as JVM heap size recommendations, should follow Maximo guidelines, where applicable; however, you will need to refer to the tuning guide for appropriate instructions to set these values for Weblogic Server.

For HTTP Compression on WebLogic, use Apache HTTP Server 2.x and above as an external HTTP server. Use mod_deflate and set DeflateCompressionLevel to **3** to improve response time in environments that have low bandwidth and high latency.

To enable verbose garbage collection in WebLogic, set the following parameter in the start command file:

```
JAVA_OPTIONS=-verbose:gc.
```

If you do not find the `JAVA_OPTIONS` parameter, you can set the `-verbose:gc` parameter in the java command. For example:

```
java -verbose:gc.
```

APPENDIX B: VMWARE CONSIDERATIONS

If you deploy Maximo on VMware, refer to the following tuning guides available from VMware:

- Performance Best Practices for VMware vSphere 5.5
- Performance Best Practices for VMware vSphere 6.0

Note: The guidelines in this document are based on results from non-virtualized environments. Deployments in VMware will likely not see the same performance benefits from the recommendations made in this document. Using shared resources in virtualized environments is not recommended for optimal performance. Use dedicated CPU and memory resources instead.

In addition, internal testing has shown that running the Maximo database in VMWare can have significant performance impacts. These impacts increase rapidly with larger workloads. While many customers run their Maximo database successfully in a VMWare virtual machine, we recommend dedicated database server hardware for optimal performance.

GLOSSARY OF TERMS

This section provides a brief explanation of some terms used in this paper.

application server. Software that maintains and provides an infrastructure to run applications, such as Maximo. IBM WebSphere Application Server (WAS) is an example.

cluster. A group of application servers that transparently run a J2EE application as if it were a single entity. The two methods of clustering are horizontal scaling and vertical scaling.

horizontal clustering. Involves running multiple Java application servers on two or more separate physical machines. Horizontal scaling is more reliable than vertical scaling, since there are multiple machines involved in the cluster environment, as compared to only one machine.

A horizontal cluster consists of at least two nodes with a single cluster instance in each node. Each node in the cluster typically represents a unique Maximo product installation on a unique physical system. Horizontal clusters, like vertical clusters, expand the capacity and availability of the applications. Unlike vertical clusters, each horizontal cluster instance has a unique set of physical resources (processors and memory). A horizontal cluster is not affected by other horizontal instance resource requirements.

vertical clustering. Consists of multiple Java application servers on a single physical machine. With vertical scaling, the machine's processing power, CPU usage, and JVM heap memory configurations are the main factors in deciding how many server instances can be run on one machine.

A vertical cluster has more than one cluster instance within a WAS node. A node typically represents a single physical server in a managed cell. It is possible to have more than one node per physical server. You can add vertical clusters to a node using the Deployment Manager administrative console. Each additional vertical cluster instance replicates the configuration of the first instance; no additional installation or configuration is necessary.

Java Virtual Machine (JVM). A platform-independent execution environment that converts Java code to machine language and executes it. Each application server runs on a different JVM. JVMs in a cluster can run on the same physical piece of hardware or on different hardware.

logical processor or **core**. An independent processor residing within the physical processor.

physical processor. The actual hardware chip containing one or more logical processors.

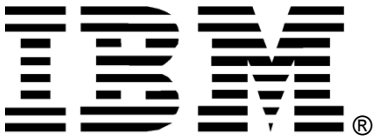
processor. The central processing unit of the computer.

processor topology. The concept of how your logical processors correlate to your physical processors. For example, a two-processor quad core system and a four-processor dual core system both have 8 logical processors. However, the systems have different processor topologies.

thread. A single execution pipeline. Some logical processors have the ability to run multiple execution threads.

virtual processor. A representation of a physical processor of a logical partition that uses shared processors.

front-end transactions and **user load** are transaction requests initiated from a browser. **Back-end transactions** and **nonuser load** are processing requests that occur on a scheduled basis, or that come in by way of the integration components.



IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PAPER “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes may be made periodically to the information herein; these changes may be incorporated in subsequent versions of the paper. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this paper at any time without notice.

Any references in this document to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
4205 South Miami Boulevard
Research Triangle Park, NC 27709 U.S.A.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Note: Other product and service names might be trademarks of IBM or other companies.