DATABASE AND INFRASTRUCTURE TUNING GUIDELINES FOR PERFORMANCE

Performance Team

TABLE OF CONTENTS

1. Introduction

This paper presents guidelines for database configurations, operating systems and other infrastructure components which could be followed to exercise optimum and acceptable performance SLAs when working with webMethods products.

2. Disclaimer

This document lists just the general guidelines which may help in improving the performance of a typical webMethods installation. These guidelines may degrade performance in different environments and use cases. These guidelines are generic and can be used on trial and error basis only.

Please contact Performance Team in Software AG for suggestions, help and clarifications if any.

3. Database Tuning

These are general high-level considerations while tuning the system:

For Oracle database:

- Put the different tablespace datafiles related to data, indexes and blobs on different physical disks
- Put the redo logfiles on different physical disk from datafiles
- Set REDO logfile size to at least 1 GB
- It is good to use a single REDO logfile if transaction data is not critical
- Database Parameters :
    - CURSOR_SHARING = SIMILAR
    - Increase the number of processes from default number to 1000
    - Increase the number of sessions from default number to 1000
    - Increase optimizer_index_caching to upto 50
    - Reduce optimizer_index_cost_adj to 50
    - If possible put recyclebin to off
    - If possible put audit_trail to NONE
- Eliminate statement parsing - add the string “MaxPooledStatements=35” in the JDBC pool URL

For DB2 database:

- By default DB2 database is created with 4K page size tablespace, but is often considered bad for performance. Create 16K/ 32K page size bufferpools and tablespaces.
- By default DB2 database is created with 4 primary and 2 secondary logfiles of 4K page size. This is also considered too less. Increase the number of primary and secondary logfiles to considerable number (say 20 each).

For SQL Server database:

- Separate Data and Log Files. Put the Data and Log files on different disks.
- Set the MAX memory setting to at least 1 to 2GB less than the total amount of memory on the server if this is a single instance. If there are multiple instances, that also have to be accounted for.
- Don’t enable shrinking Data files.

3.1. Product specific database tuning

3.1.1. My webMethods Server

From My webMethods Server (MWS) point of view, the only tables which needed tuning are Task Engine tables. Those are T_TASK_nnn, as they typically have a lot of records and are being accessed a lot in the BPMS deployments.

This change is to add the following index to drastically improve task delete performance:

```
CREATE INDEX IX_T_TASK_8 ON T_TASK (PARENT_TASK_ID)
TABLESPACE WEBMINDX;
```

Indexing all possible columns and combination of column in the T_TASK_nnn tables is not a good approach as it will slow down significantly any inserts/ updates/ deletes.

```
ALTER TABLE T_TASK_INBOX ADD CONSTRAINT
PK_T_TASK_INBOX PRIMARY KEY (SOURCE_ID, TASK_ID, TYPE, REFERENCE_ID)
USING INDEX TABLESPACE WEBMINDX;
```

These are the indexes that are added in production for improving the performance of My webMethods Server drastically.

Index creation statements:

```
CREATE INDEX WM_TM_MWS.IX_T_TASK_NEW ON WM_TM_MWS.T_TASK
(TASK_DEFINITION_ID, STATUS)COMPUTE STATISTICS;
```

Above statement adds an index on TASK_DEFINITION_ID, STATUS columns. Adding an index on both (TASK_DEFINITION_ID, STATUS) columns is good for performance.

```
CREATE INDEX WM_TM_MWS.IX_T_TASK_INBOX_NEW ON
WM_TM_MWS.T_TASK_INBOX (SOURCE_ID) COMPUTE STATISTICS;
```

To remove logging of table data:

```
alter table T_JMS_QUEUE nologging;
alter table T_JMS_EVENTS nologging;
alter table T_TASK_LOCK nologging;
```

Data is not purged on restart of My webMethods Sever because any given MWS is not aware about any other My webMethods Severs running and using these tables. Though there are possible theoretical use cases when data corruption or loss in these tables occurring due to Oracle failure may lead to operational troubles.

For example:

- MWS has a lock on task when updating it, but did not release the lock due to Oracle transaction failure.
  * solution: to clean lock records manually.
- MWS JCR index uses events which are distributed through JMS tables to maintain its consistency. So if the event is lost, the index may become inconsistent.

* solution: index automatically checks consistency on startup.

4. Infrastructure Tuning

Tuning the infrastructure for better performance of webMethods products involves monitoring the hardware usage and tuning the hardware to get maximum throughput with optimum response times.

4.1. Monitoring for Infrastructure tuning

To tune the infrastructure for various combinations of solutions and webMethods products, you have to monitor the system and gather the performance statistics over a period of time. These statistics can be analyzed to find out the optimum values which can give performance boost.

Following are a few actions to follow:

- Check hardware, understand webMethods solution and deployment architecture
- Identify the critical business scenarios and find out the required performance SLAs
- Check for the fix levels and install latest fix level
- Check similar issues or relevant fixes for the performance issues/ improvements
- Make one change at a time, test and monitor the performance test results
- Check bottlenecks for CPU, memory, network, and disk
- Monitor CPU queue/ usage, paging, network utilization, disk usage/ IOPS
- Ensure that none of the hardware resources are bottlenecks. If any, tune/ fix them first
- Check the tuning parameters specific for the webMethods products

4.2. Operating system tuning

4.2.1. Windows

Network Settings:

In Windows registry under the following path:

- MyComputer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
- MaxUserPort (REG_DWORD) = 65534 (default – 5000)

In Windows registry under the following path:

- MyComputer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\ {3856EA3E-82AB-44A6-824C-C7936ADB0189}
- TCPAllowedPorts (REG_MULTI_SZ) = 0 (default – 0)
- TcpWindowSize (REG_DWORD) = 1094713343 (default – 65534)
- UDPAllowedPorts (REG_MULTI_SZ) = 0 (default – 0)

TCPWindowSize: This value determines the maximum amount of data (in bytes) that can be outstanding on the network at any given time. It can be set to any value from 1 to 65,535 bytes by using the following registry entry:

> *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\TcpWindowSize (REG_DWORD)*

The default for a gigabit interface is set to approximately 65,535 (rounded down to the nearest multiple of full TCP packets), 16,384 for a 100 Mbps link, and 8,192 for all interfaces of lower speeds (for example, modems). This value should be set to the product of end-to-end network bandwidth (in bytes/s) and the round-trip delay (in seconds), also referred as the bandwidth delay product. This value should be set according to the amount of TCP data expected to be received by the computer.

MaxUserPort: A port is used whenever an active connection is used from a computer. Given the default value of available user mode ports (5,000 for each IP address) and TCP time-wait requirements, it may be necessary to make more ports available on the system. You can set the following registry entry to as high as 0xfffe (65534):

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\MaxUserPort
```

4.2.2. Linux

Note: Following are the recommended parameters. Since operating system tuning is trial and error method, try out these parameters for performance gains.

File Descriptors

Increase the number of file descriptors from default. Having a higher number of file descriptors ensures that the server can open sockets under high load and not abort requests coming in from clients. Start by checking system limits for file descriptors with following command:

```
cat /proc/sys/fs/file-max 8192
```

The current limit is 8192. To increase it to 65535, use the following command (as root):

```
echo "65535" > /proc/sys/fs/file-max
```

To retain this value after a system reboot, add it to /etc/sysctl.conf and specify the maximum number of open files permitted:

```
fs.file-max = 65535
```

Note: The parameter is not proc.sys.fs.file-max, as one might expect.

To list the available parameters that can be modified using sysctl:

```
sysctl –a
```

To load new values from the sysctl.conf file:

```
sysctl -p /etc/sysctl.conf
```

To check and modify limits per shell, use the following command:

```
ulimit
```

The output will look something like this:

```
cputime unlimited
filesize unlimited
datasize unlimited
stacksize 8192 kbytes
coredumpsize 0 kbytes
memoryuse unlimited
descriptors 1024
memorylocked unlimited
maxproc 8146
openfiles 1024
```

The openfiles and descriptors show a limit of 1024. To increase the limit to 65535 for all users, edit /etc/security/limits.conf as root, and modify or add he nofile setting (number of file) entries:

- soft nofile 65535
- hard nofile 65535

Virtual Memory

To change virtual memory settings, add the following to /etc/rc.local:

```
echo 100 1200 128 512 15 5000 500 1884 2 > /proc/sys/vm/bdflush
```

Network Interface

To ensure that the network interface is operating in full duplex mode, add the following entry into /etc/rc.local:

mii-tool -F 100baseTx-FD eth0 - where eth0 is the name of the network interface card (NIC).

To tune disk I/O performance for non SCSI disks

1. Test the disk speed.
   Use this command:
   ```
   /sbin/hdparm -t /dev/hdX
   ```

2. Enable direct memory access (DMA).
   Use this command:
   ```
   /sbin/hdparm -d1 /dev/hdX
   ```

Check the speed again using the hdparm command.

Given that DMA is not enabled by default, the transfer rate might have improved considerably. To do this at every reboot, add the

/sbin/hdparm -d1 /dev/hdX line to /etc/conf.d/local.start, /etc/init.d/rc.local, or to whatever the startup script is called.

TCP/IP Settings

To tune the TCP/IP settings

1. Add the following entry to /etc/rc.local

```
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 60000 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 15000 > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
```

2. Add the following to /etc/sysctl.conf

```
- Disables packet forwarding
        net.ipv4.ip_forward = 0
- Enables source route verification
        net.ipv4.conf.default.rp_filter = 1
- Disables the magic-sysrq key
        kernel.sysrq = 0
        net.ipv4.ip_local_port_range = 1204 65000
        net.core.rmem_max = 262140
        net.core.rmem_default = 262140
        net.ipv4.tcp_rmem = 4096 131072 262140
        net.ipv4.tcp_wmem = 4096 131072 262140
        net.ipv4.tcp_sack = 0
        net.ipv4.tcp_timestamps = 0
        net.ipv4.tcp_window_scaling = 0
        net.ipv4.tcp_keepalive_time = 60000
        net.ipv4.tcp_keepalive_intvl = 15000
        net.ipv4.tcp_fin_timeout = 30
```

3. Add the following as the last entry in /etc/rc.local

```
sysctl -p /etc/sysctl.conf
```

4. Reboot the system.

5. Use this command to increase the size of the transmit buffer:

```
tcp_recv_hiwat ndd /dev/tcp 8129 32768
```

6. Use this during time out issues

```
/proc/sys/net/ipv4/tcp_syn_retries = 1
```

Verify following parameters

Settings to change in Linux/ Unix machines for tuning TCP settings.

```
net.core.wmem_max=15000000
net.core.rmem_max=15000000
net.ipv4.tcp_rmem= 10240 4194304 15000000
net.ipv4.tcp_wmem= 10240 4194304 15000000
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_no_metrics_save = 1
net.core.netdev_max_backlog = 10000
```

4.2.3. HP-UX

Note: Following are the recommended parameters. Since operating system tuning is trial and error method, try these parameters for performance gains.

The system has default kernel parameters, except max_thread_proc and maxdsiz have been increased to recommended values for Java.

Disk related parameters

Create_Fastlinks
Create_fastlinks tells the system to store HFS symbolic link data in the symbolic link's inode. This reduces disk space usage and increases performance. By default, this feature is disabled for backward compatibility. All systems should have create_fastlinks enabled by setting this kernel parameter to 1.

Dbc_Max_Pct
This parameter determines the percentage of main memory that the dynamically allocated buffer cache is allowed to grow to. As the system uses as much memory as it can for buffer cache, when performing intense block I/O, this becomes the size of the buffer cache on a system that is not feeling memory pressure due to process invocations. The problem arises when there is memory stress due to process space requirements. The process space requirements require the system to start paging, at which point the system tries to reclaim buffer cache pages to allocate them to running processes. But the system is also trying to allocate as much buffer cache as it can, causing a vicious cycle of allocating and deallocating memory between buffer cache and process memory space, creating a large amount of overhead. The idea then is to keep this number reasonably low, allowing you to have the cache space but also keep the application space large enough to avoid high levels of conflict between them. The default value is 50%, but we recommend 25% to start. We have seen systems that need buffer cache to have a maximum of as little as 5%, with a minimum at 2%. This requires careful attention with appropriate modification. If this form of thrashing in main memory becomes an increasing problem, the only good fix is to purchase more physical memory.

Fs_Async
This kernel parameter controls the switch between synchronous or asynchronous writes of file system meta structures to disk. Asynchronous writes to disk can improve file system I/O performance significantly. However, synchronous writes to disk make it easier to restore file system integrity if a system crash occurs while file system meta structures are being updated on the file system. Depending on the application, you have to decide which is more important. The decision should be based on what types of applications are going to be run. You may value file system integrity more than I/O speed. If so, fs_async should be set to 0.

Memory Related parameters

Maxdsiz
Maxdsiz defines the maximum size of the data segment of an executing process. The default value of 64 MB is too small for most applications. We recommend this value be set to the maximum value of 1.9 GB. If maxdsiz is exceeded by a process, it will be terminated, usually with a SIGSEGV (segmentation violation) and you will probably see the following message:

Memory fault (coredump)

In this case, check out the values of maxdsiz, maxssiz and maxtsiz. For more information on these parameters, please see the online help section within SAM's Kernel configuration. If you need to exceed the specified maximum of 1.9 GB, there are a couple of ways (yet to be supported) to do so. Contact your Hewlett Packard technical consultant for the details. It is important to note that the maxdsiz parameter must be modified in order for these procedures to work. Maxdsiz will need to be set to 2.75 GB or 3.6 GB depending on the method chosen and/ or size required.

Maxfiles
This sets the soft limit for the number of files a process is allowed to have open. Recommended value is 200.

Maxssiz
Maxssiz defines the maximum size of the stack of a process. The default value is 8 MB. Recommended value is 79 MB.

Maxtsiz
Maxtsiz defines the maximum size of the text segment of a process. Recommended value is 1024 MB.

Ninode
Ninode sizes the incore inode table also called the inode cache. For performance, the most recently accessed inodes are kept in memory. Each open file has an inode in the table. An entry is made in the table for each "login directory", each "current directory", each mount point directory, and so on. It is recommended that ninode be set to 15,000.

maxdsize_64
double or increase the size of this parameter for performance improvement in case of memory bound tests on HP UX.

4.2.4. Solaris

Note: Following are the recommended parameters. Since operating system tuning is trial and error method, try these parameters for performance gains.

TCP Parameters

tcp_conn_req_max_q
Specifies the default maximum number of pending TCP connections for a TCP listener waiting to be accepted by accept (3SOCKET).

Default Value: 128

Range 1 to 4,294,967,296

For applications such as web servers that might receive several connection requests, the default value might be increased to match the incoming rate. Do not increase value of the parameter to a very high value. The pending TCP connections can consume excessive memory. Also, if an application cannot handle that many connection requests fast enough due to large number of pending TCP connections, new incoming requests might be denied.

Note that increasing tcp_conn_req_max_q does not mean that applications can have that many pending TCP connections. Applications can use listen (3SOCKET) to change the maximum number of pending TCP connections for each socket. This parameter's value is the maximum an application can use listen() to set the number to. Thus, even if this parameter is set to a very large value, the actual maximum number for a socket might be much less than tcp_conn_req_max_q, depending on the value used in listen().

tcp_conn_req_max_q0
Specifies the default maximum number of incomplete (three-way handshake not yet finished) pending TCP connections for a TCP listener.

Default 1024

Range 0 to 4,294,967,296

For applications such as web servers that might receive excessive connection requests, you can increase the default value to match the incoming rate. The following explains the relationship between tcp_conn_req_max_q0 and the maximum number of pending connections for each socket. When a connection request is received, TCP first checks if the number of pending TCP connections (three-way handshake is done) waiting to be accepted exceeds the maximum (N) for the listener. If the connections are excessive, the request is denied. If the number of connections is permissible, then the TCP checks if the number of incomplete pending TCP connections exceeds the sum of N and tcp_conn_req_max_q0. If it does not, the request is accepted. Otherwise, the oldest incomplete pending TCP request is dropped. Set the following TCP-related tuning parameters using the ndd command as demonstrated in the following example:

```
ndd -set /dev/tcp tcp_conn_req_max_q 16384
```

Tip: Use the netstat -s -P tcp command to view all available TCP parameters.

/etc/system File Parameters

This section lists important /etc/system file tuning parameters that when tuned, can enhance application performance. Each socket connection to the server consumes a file descriptor. To optimize socket performance, you need to configure your operating system to have the appropriate number of file descriptors. Therefore, you should change the default file descriptor limits, as well as the hash table size and other tuning parameters in the /etc/system file to the recommended values in the following list of parameters:

Note: You must reboot your machine everytime you modify /etc/system parameters.

rlim_fd_cur
Defines the “soft” limit on file descriptors which a single process can have open. A process might adjust its file descriptor limit to any value up to the “hard” limit defined by rlim_fd_max by using the setrlimit() call or by issuing the limit command in whatever shell it is running. You do not require a superuser privilege to adjust the limit to any value less than or equal to the hard limit.

Default 256

Range 1 to MAXINT

When the default number of open files for a process is not enough, increasing this value means that it might not be necessary for a program to use setrlimit to increase the maximum number of file descriptors available for it.

rlim_fd_max
It specifies the “hard” limit on file descriptors which a single process might have opened. Overriding this limit requires a superuser privilege.

Default 65,536

Range 1 to MAXINT

When the maximum number of open files for a process is not enough. Other limitations in system facilities can mean that a larger number of file descriptors is not as useful as it might be. For example:

- A 32-bit program using standard I/O is limited to 256 file descriptors. A 64-bit program using standard I/O can use up to 2 billion descriptors. Specifically, standard I/O refers to the stdio(3C) functions in libc(3LIB).
- select is by default limited to 1024 descriptors per fd_set. For more information, see select(3C). Starting with the Solaris 7 release, 32-bit application code can be recompiled with a larger fd_set size (less than or equal to 65,536). A 64-bit application uses an fd_set size of 65,536, which cannot be changed.

An alternative to changing this on a system wide basis is to use the plimit(1) command. If a parent process has its limits changed by plimit, all children inherit the increased limit. This alternative is useful for daemons such as inetd.

ip:ipcl_conn_hash_size
Controls the size of the connection hash table used by IP. The default value of 0 means that the system automatically sizes an appropriate value for this parameter at boot time, depending on the available memory.

Default 0

Range 0 to 82,500

If the system consistently has tens of thousands of TCP connections, the value can be increased accordingly. Increasing the hash table size means that more memory is wired down, thereby reducing available memory to user applications.

autoup
Along with tune_t_flushr, autoup controls the amount of memory examined for dirty pages in each invocation and frequency of file system synchronizing operations. The value of autoup is also used to control whether a buffer is written out from the free list. Buffers marked with the B_DELWRI flag (which identifies file content pages that have changed) are written whenever the buffer has been on the list for longer than autoup seconds. Increasing the value of autoup keeps the buffers in memory for a longer time.

Default 30

Range 1 to MAXINT

Here are several potential situations for changing autoup,

tune_t_fsflushr, or both:

- Systems with large amounts of memory – In this case, increasing autoup reduces the amount of memory scanned in each invocation of fsflush.
- Systems with minimal memory demand – Increasing both autoup and tune_t_fsflushr reduces the number of scans made. autoup should be increased also to maintain the current ratio of autoup /

tune_t_fsflushr.

- Systems with large numbers of transient files (for example, mail servers or software build machines) – If large numbers of files are created and then deleted, fsflush might unnecessarily write data pages for those files to disk.

tune_t_fsflushr
Specifies the number of seconds between fsflush invocations.

Default 1

Range 1 to MAXINT

Kernel parameters

Maxphys
Defines the maximum size of physical I/O requests. If a driver encounters a request larger than this size, the driver breaks the request into maxphys sized chunks. File systems can impose own limit.

Default 131,072 (sun4u or sun4v) or 57,344 (x86). The sd driver uses the value of 1,048,576 if the disk drive supports wide transfers. The ssd driver uses 1,048,576 by default.

Range Machine-specific page size to MAXINT

When doing I/O to and from raw devices in large chunks. Note that a DBMS doing OLTP operations issues large number of small I/Os. Changing maxphys does not result in any performance improvement in that case. You might also consider changing this parameter when doing I/O to and from a UFS file system where large amounts of data (greater than 64 Kbytes) are being read or written at any one time. The file system should be optimized to increase contiguity. For example, increase the size of the cylinder groups and decrease the number of inodes per cylinder group. UFS imposes an internal limit of 1 Mbyte on the maximum I/O size it transfers.

Slowscan
Defines the minimum number of pages per second that the system looks at when attempting to reclaim memory.

Default: The smaller of 1/20th of physical memory in pages and 100.

Range 1 to fastscan / 2

When more aggressive scanning of memory is preferred during periods of memory shortfall, especially when the system is subject to periods of intense memory demand.

tune_t_fsflushr
Specifies the number of seconds between fsflush invocations

Default 1

Range 1 to MAXINT

set tune_t_fsflushr=5 use autoup.

rechoose_interval
Specifies the number of clock ticks before a process is deemed to have lost all affinity for the last CPU it ran on. After this interval expires, any CPU is considered a candidate for scheduling a thread. This parameter is relevant only for threads in the timesharing class. Real-time threads are scheduled on the first available CPU.

Default 3

Range 0 to MAXINT

When caches are large, or when the system is running a critical process or a set of processes that seem to suffer from excessive cache misses not caused by data access patterns. Consider using the processor set capabilities available as of the Solaris 2.6 release or processor binding before changing this parameter. For more information, see psrset(1M) or pbind(1M).

This settings tries to run threads on the same CPU it ran before. The understanding is that the CPU cache is warm and has the instructions and data for the thread improving efficiency. The rechoose_internal variable instructs the kernel on which CPU to select to run a thread. So if a process hasn't run in rechoose_interval ticks, it will be moved to another CPU. Otherwise, it will continue to wait on the CPU it has been running on. A higher value of rechoose_interval "firms-up" the soft affinity. The down side is that you can end up with sluggish spreading out of processes on an application where a single process forks a lot of children if this value is too high.

The below statement set's it to 150 which is a fairly good value for Datawarehouse systems. However you need to test and see if it reduces your LWP thread migrations.

set rechoose_interval=150

4.2.5. AIX

Kernel Settings

1. Set the user process resource limits through the Ulimit command to the following values:

Time (seconds)	Unlimited
file (blocks)	Unlimited
data (Kbytes)	Unlimited
stack (Kbytes)	2097152
Memory (Kbytes)	Unlimited
Core dump (blocks)	2097151
nofiles(descriptors)	4096 - If possible raise to 8192

The hard limits will have to be set accordingly.

Note: The nofiles parameter limits the number of files open by a single process. Change this value to at least 4096 or make it unlimited and then tune down to an optimal value.

2. Process priority can be increased to increase the response time.

3. Kernel Debugging Flags

   o JVMAIXTHREAD_COND_DEBUG,
   o AIXTHREAD_MUTEX_DEBUG
   o AIXTHREAD_RWLOCK_DEBUG

   These flags are used for kernel debugging purposes.
   Switching them off can provide a good performance boost.

4. Tune the AIX memory. Set the following parameters to appropriate values:

```
vmo -r -o lgpg_size=value lgpg_regions=value
vmo -p -o minperm%=value
vmo -p -o maxperm%=value
vmo -p -o maxclient%=value
vmo -p -o minfree=value
vmo -p -o maxfree=value
```

   Sample set of values is shown below:

```
vmo -r -o lgpg_size=16777216 lgpg_regions=128
vmo -p -o minperm%=5
vmo -p -o maxperm%=90
vmo -p -o maxclient%=90
vmo -p -o minfree=960
vmo -p -o maxfree=1088
```

Tuning disk performance

"ioo –L" lists all the current values on the system like the current value, default value and the maximum value. You can change those parameter values which may improve the performance.

```
ioo -o j2_maxPageReadAhead=value
ioo -o j2_minPageReadAhead=value
ioo -o j2_maxRandomWrite ioo -oj2_nRandomCluster
ioo -o j2_nPagesPerWriteBehindCluster=value
ioo -o j2_nBufferPerPagerDevice=value
```

Note: Be careful while changing these parameters as they may severely impact the performance.

tcp_nodelayack network parameter must be set to '1' in AIX system to avoid performance issues with message size greater than 64KB.

LRU_FILE_REPAGE

Starting with AIX 5.2 ML4, a new page stealing option is available that should be used for any AIX server whose primary role is either an application server, database server, or other mid-tier cache like Squid or memcached.
The tuning parameter is LRU_FILE_REPAGE and by default it is set to 1, which is fine for a general purpose file server. But for an AIX server hosting an application server such as WebSphere or running a database, you should set LRU_FILE_REPAGE=0.This instructs the AIX

page stealing daemon to steal (that is page out), from the file cache portion of the memory instead of any working sets from processes like WebSphere, databases, and so on. We do not want our processes paging out to disk. If you run with the default of LRU_FILE_REPAGE=1, then the page stealing algorithm will page out from both the file cache portion of memory and working sets from processes.

5. Other Infrastructure Component Performance Aspect:

5.1. Apache httpd loadbalancer

If Apache httpd load balancer is being used and if one of the cluster nodes is down, you will see long delays because of long default connection time outs which could result into Denial of Service (DoS).
Mod proxy has a very long connection time out by default (300 seconds). If you do not set it correctly, Apache will take long time until offline nodes are detected as being in error state.
By setting a short connectiontimeout and increasing the retry, you could make it work better:

```
BalancerMember http://host1:8080 route=1 connectiontimeout=10 retry=600
```

This will ensure that failing connections are detected fairly quickly and Apache does not retry too often to reach failing servers. But, it seems Apache uses actual requests for checking the balance members and thus from time to time single requests may be slow when it tries to reach a server previously put into error state.

5.2. WAN impact

In zero latency or local network if you are getting x throughput and if you move the same solution to WAN, then throughput would be (x/latency)/2.

5.3. RAID levels

- RAID (Redundant Array of Inexpensive Disk) Striping can provide a higher effective throughput than a single disk writing to multiple drives simultaneously. Performance for specific RAID systems varies significantly, but it is possible to see a performance improvement of two to three times over a single disk. The RAID level will also have an impact on performance. Selecting the optimal RAID level is a complex process, and each has its own strengths and drawbacks. For example, RAID 0/1 offers the best performance and is best suited to data that is accessed often (making this generally the recommended choice for Broker storage), while RAID 5 is cheaper and is suitable for data that is archived and only accessed infrequently. The RAID level choice will also depend on factors such as the cost of disk drives. Disk prices are currently falling; making RAID 0/1 is a more attractive option.
- The optimal RAID solution will depend on the disk access characteristics. For Broker disk storage, RAID 0/1 is preferred over RAID 5, since RAID 0/1 is more suited to fast read and write I/O access, whereas RAID 5 favors read over write access.
- RAM-fronted disk storage (for example, EMC devices) uses a large battery-backed RAM cache in front of the disk to provide performance nearly as fast as memory-only solutions with all the reliability of disk storage. This can produce an order of magnitude performance gains, but is expensive.
- Storage area network (SAN) systems offer incredible performance as well. Performance numbers nearly as fast as memory-only solutions can be achieved with such components.

Network mounted disks, such as NFS or Windows File Sharing, are not supported by the Broker. These are very slow, generate additional network traffic, and have been known to lose data when network failures occur.

Note that disk storage is generally not a strong performance factor for the Integration Server processes, except under high loads with a lot of audit logging.

5.4. Virtualization

In general disk bound activities will have high impact on performance (up to 40%) unless data being written on the virtual machines disk (local) is not a SAN mount point. Otherwise, impact could be around 5-15% for CPU and memory bound operations. Also it is really important to make sure that network bandwidth is 1Gbps or more for each virtual machine or some of them at least. But in most of the cases, customers go with 100mbps bandwidth and run into issues later.

5.5. F5 Load balancer

Configuration of the F5 hardware load balancer can be problematic due to the limited knowledge of the device in the network support group. There can be a timeout configured (at 30 seconds) after which the F5 would resend the request. Due to the time taken to execute the BPMS steps between the human workflow tasks this timeout may sometimes be triggering. This may cause task close requests to be sent more than once, culminating in a “Completed task cannot be updated” error message. The timeout can be adjusted to be more than the longest wait when linking tasks.

Compression can also be turned on in the F5, but it may cause issues with F5 CPU consumption. The F5 compression was turned off.

5.6. Network considerations and best practices

Networks are often the weakest performance link in most integration projects. The following is a list of network-related best practices:

- Place things close together if they communicate frequently. Networks can very easily become the performance bottleneck in an integration project.
- The most complex interactions in integration projects usually occur between an adapter and the resource it is accessing. For example, inserting a customer object into Clarify can take 13 operations. Minimizing the network distance between the adapter and the resource can be extremely important for performance because it minimizes communications across the network.
- The next most complex interactions occur between an Integration Server and the Broker. Guaranteed messaging requires handshaking that causes extra traffic. It is generally preferable to keep the Broker and Integration Server on separate machines, so they do not consume each other’s system resources. However, if the guaranteed document traffic between the two is expected to be large, it might make sense to consider locating the Integration Server and the Broker on a single machine to increase the data transfer speed.

- In many cases, you cannot place (or do not want to place) source and target adapters on the same host. This means that the network must be involved. The communication between Brokers is very efficient, so it is common to create a territory with one Broker on each Integration Server host.
- Minimize network cross-traffic through judicious sub-netting. TCP/IP is a broadcast protocol, meaning all messages go to all addresses and it is up to the receiver to determine if the packet is addressed to it or not. Also, increased network traffic can lead to packet collisions, which require retransmission, thereby increasing network traffic even more. Keeping cross-talk to a minimum makes network usage more efficient.

In addition, each message sent on a network has overhead, which makes smaller messages less efficient. This means that by using a series of 1,000 byte messages, it is difficult to get more than 1 Megabyte a second across a network. For this reason, fewer but larger messages are the way to achieve maximum throughput. Grouping small documents into larger ones and using the multi-publish and multi-get features of the Broker and Integration Server are ways to be more efficient.

5.7. Disk storage

The guaranteed document lifecycle requires that the documents be stored on disk at various points. For example, when a guaranteed document is received, it is written to the guar.log file (one disk access), then committed to the guar file (another disk access). If these disks are on a RAID 5 system, each disk access could be up to 4 I/Os (2 reads + 2 writes) – a total of 8 disk accesses per guaranteed transaction. Therefore, when using guaranteed documents on a high load system, the Broker often becomes bound by the speed of the disk. Disk and controller performance can dramatically affect Broker performance.

- Using fast disks can dramatically increase the Broker throughput.
- Whenever possible, assign each Broker its own dedicated disk. Sharing a disk with the operating system or other active software can reduce Broker performance.
- If possible, giving the enterprise server its own dedicated disk controller can further enhance performance.

5.8. 32bit vs. 64bit JVM

32 bit machine cannot support more than 4GB of memory. Now 64 bit hardware will have 64 bit address bus and it can point up to 16777216 TB (theoretically supported no.). Practically such a huge memory is not supported because other hardware components of mother board don’t support it.

On 64 bit box the amount of memory you can assign to heap depends on the available memory. Leaving some amount of memory for operating system and other services to run, remaining memory can be allocated to the Integration Server.
On 32 bit things are different. Since 32 bit box supports only 4GB of memory at the same time it supports only 4GB of virtual memory. Now if you assume 500 MB memory is sufficient to run operating system and other services to run and remaining 3.5 GB of memory can be allocated to Integration Server. But this is not true. Since 4GB of virtual memory is divided into 2GB application virtual memory and 2GB kernel virtual memory. Because of this you cannot assign more than 2GB of virtual memory to Integration Server. In most cases it works well with 1600 to 1800MB. But still there are ways to increase the heap size beyond

2 GB. For this you will take out 1GB of kernel virtual memory and add it to user virtual memory space. Now you can have heap size of Integration Server close to 3GB.

5.9. Validating performance test results

The best way to validate your performance test results is to use Little's Law. It states that the fundamental long-term relationship between Work-In-Process, throughput and flow time of a production system in steady state is

$N = X * (R+TT)$

Where N – no of concurrent users
X – throughput/sec (no. of transactions completed per second)
R – Response Time
TT – Think time (This is amount of time user waits before making next request to the server. This is to make a scenario a real world scenario).

5.10. webMethods running on Oracles' UltraSPARC

Traditional processor design has long emphasized the performance of a single hardware thread of execution, and focused on providing high levels of instruction-level parallelism (ILP). These increasingly complex processor designs have been driven to very high clock rates (frequencies), often at the cost of increased power consumption and heat production. But, the impact of memory latency has impacted even the fastest single-threaded processors to spend most of the time stalled, waiting for memory. Complicating this tendency, many of today's complex commercial workloads are unable to take advantage of instruction-level parallelism, instead benefiting from thread-level parallelism (TLP). SUN's Throughput Computing initiative represents a new paradigm in system design, with a focus toward maximizing the overall throughput of key commercial workloads rather than the speed of a single thread of execution. Chip multi-threading (CMT) processor technology provides a new thread-rich environment that drives application throughput and processor resource utilization while effectively masking memory access latencies.
SUN's UltraSPARC® T2 Plus processor combines chip multiprocessing (CMP) and hardware multithreading (MT) with an efficient instruction pipeline to enable chip multithreading. The resulting processor design provides multiple physical-instruction execution pipelines and several active thread contexts per pipeline.

The reason we see better numbers on this Dell server compared to Oracle's T series machine is because of the architecture of the hardware and nothing else.

Dell server, if it is 8 core box and running with 2.87 GHz then it would spawn 8 hardware threads and each one of them will be processing requests in parallel with processing speed of 2.87 GHz. In case of Oracle's T series T5440 the processing speed of each core is 1.6GHz and because of its architecture (which is meant for web based applications) it spawns 8 threads for each core. Each thread is called as strand here. So for 8 cores there would be 64 strands and each strand acts as a core and its processing speed will be

1.6GHz/8= 200MHz. So when you run single threaded tests, you are comparing performance of 2.87GHz vs. 200 MHz. It should be 14 ((200MHz*14) = 2.8GHz) times better considering processing speed but it is not because of limitation of memory speed which is 1333MHz (max). This is the reason why Oracle came up with T series hardware to overcome memory stall problems.

5.11. Bandwidth delay product

Bandwidth delay product or BDP– a calculation to determine how much outstanding traffic you may have on your line at a given latency.

If you know your RTT latency, then this simple calculation basically tells you what you need to set your TCP window sizes at to make them big enough to accommodate all the TCP traffic that is in transit at the moment.

The higher your RTT latency, the higher you will need to set your TCP windows to compensate for as-of-yet-unpacked TCP packets.

In data communications, bandwidth-delay product refers to the product of a data link's capacity (in bits per second) and its end-to-end delay (in seconds). The result, an amount of data measured in bits (or bytes), is equivalent to the maximum amount of data on the network circuit at any given time; that is data that has been transmitted but not yet received. Sometimes it is calculated as the data link's capacity multiplied by its round trip time.

A network with a large bandwidth-delay product is commonly known as a long fat network (shortened to LFN and often pronounced "elephant"). As defined in RFC 1072, a network is considered an LFN if its bandwidth-delay product is significantly larger than 105 bits (12500 bytes).
Ultra-high speed LANs may fall into this category, where protocol tuning is critical for achieving peak throughput, on account of their extremely high bandwidth, even though their delay is not great.
An important example of a system where the bandwidth-delay product is large is that of GEO satellite connections, where end-to-end delivery time is very high and link throughput may also be high. The high end-to-end delivery time makes life difficult for stop-and-wait protocols and applications that assume rapid end-to-end response.

A high bandwidth-delay product is an important problem case in the design of protocols such as TCP in respect of performance tuning, because the protocol can only achieve optimum throughput if a sender sends a sufficiently large quantity of data before being required to stop and wait until a confirming message is received from the receiver, acknowledging successful receipt of that data. If the quantity of data sent is insufficient compared with the bandwidth-delay product, then the link is not being kept busy and the protocol is operating below peak efficiency for the link. Protocols that hope to succeed in this respect need carefully designed self-monitoring, self-tuning algorithms. The TCP window scale option may be used to solve this problem caused by insufficient window size, which is limited to 65535 bytes without scaling.

Examples

- Moderate speed satellite network: 512 kbit/s, 900 ms RTT
  B×D = 512×10^{3} b/s × 900×10^{-3} s = 460,800 b.
- Residential DSL: 2 Mbit/s, 50 ms RTT
  B×D = 2×10^{6} b/s × 50×10^{-3} s = 100×10^{3} b, or 100 kb, or 12.5 KiB.
- Mobile broadband (HSDPA): 6 Mbit/s, 100 ms RTT
  B×D = 6×10^{6} b/s × 10^{-1} s = 6×10^{5} b, or 600 kb, or 75 KiB.
- Residential ADSL2+: 20 Mbit/s (from DSLAM to residential modem), 50 ms RTT
  B×D = 20×10^{6} b/s × 50×10^{-3} s = 10^{6} b, or 1 Mb, or 125 KiB.
- High-speed terrestrial network: 1 Gbit/s, 1 ms RTT
  B×D = 10^{9} b/s × 10^{-3} s = 10^{6} b, or 1 Mb, or 125 KiB.

5.12. CPU

CPU is one of the easiest parameters to assess. Most operating systems provide a way of measuring how much work the CPUs are doing. MWS production instances should be run on not less than a 2 CPU machines. For anything demanding even reasonable throughput, a 4 CPU machine is required. The more CPUs allocated to an application, the more work the JVM can execute in parallel. This is particularly important when the application is dealing with 'slow' devices such as hard disk and network.
To measure CPU usage on UNIX use `vmstat`, on Windows systems use Task Manager.

Depending on your version of UNIX you should be able to display average CPU load, or the results for each of your CPUs individually. On Windows you can select this through Task Manager->View->CPU History.

If none of your CPUs ever hits 100%, then your performance is not constrained by CPU. This doesn't necessarily mean that more CPUs wouldn't help improve performance. In a multi-threaded environment there is overhead caused by switching between threads, but the effect of this is relatively small. Note that an array of the fastest CPUs will not help you if you are not constrained by CPU, but instead by some of the other factors below.

If your CPU is hitting 100% consider if there are any other factors causing it to do so. It may be that you are hitting memory limits and the CPU is spending too much time collecting garbage.
There is a sizing tool available. It is called wm_sizing_estimate.war and plugs into My webMethods Sever UI. It does not have many parameters to drive it and can be best used to get a ballpark figure for the number of CPUs.

5.13. Disk throughput

The operating system often has to read and write to the local disk. The worst scenario is where physical memory is full and the operating system is 'swapping' physical memory with 'virtual' memory on the hard disk.
Disk speed is one of those areas where faster is almost certainly better. Many smaller machines use 7,200 rpm disks, laptops are worst at 5,400 rpm. Enterprise class machines should be using 10,000 rpm disks, or faster. The fastest speeds are usually obtained by using a Storage Area Network. These platforms usually

stripe the data across multiple disks to reduce access times and communicate with the server over fiber (SAN) or gigabit Ethernet (NAS). Also they are usually highly redundant, allowing faulty disks to be replaced with zero down time or impact on performance.

5.14. Network performance

Unless your webMethods environment is hosted on a massively parallel machine it is likely that you will be accessing resources over the network. Network bottlenecks are less of an issue with the advent of gigabit Ethernet and the use of network switches instead of routers, but it is still worth checking if you have any. Note that a network cannot run at 100% capacity. 25% is high. 50% peaks are asking for trouble. Check for errors on the network segments you are using.
All of your webMethods environment should be running on a single subnet, including the database used to back the products. If you see a sudden and unexpected decrease in performance, check that your third party systems provider hasn’t moved a server, or worst, the database to another data center to consolidate services.