The Performance of SAP MII on Intel Xeon Hardware Analysis Summary

Applies to:

SAP MII v12.1 on SAP NetWeaver CE

SAP MII v12.0 on SAP NetWeaver 2004s (v7.01)

SAP ERP v6.0 on NetWeaver 2004s (v7.01)

Microsoft Windows Server 2003 Enterprise with MaxDB v7.7

The Grinder v3.2 (http://grinder.sourceforge.net)

SAP Plant Connectivity (PCo) v2.0 (Requires .NET Framework v3.5)

Microsoft SQL Server 2005 Developer Edition

Intel® Server Systems

Summary

When sizing the hardware for use with the MII product in your corporate environment there are various factors which can play a role in which hardware configuration best fits your needs. Since the MII product is a framework for developing manufacturing integration composites there are multiple different ways in which the product can be used. This guide provides a high level overview of the performance testing performed and the results from each test.

I would like to extend special thanks to Intel for supplying the hardware which made this initiative and sizing guide possible. I would also like to thank the Global Ecosystem & Partner Group (GEPG), Manufacturing Community of Practice (CoP) and the Perfect Plant teams for corporate introductions and validation of the various scenarios outlined in this guide.

Authors: Salvatore Castro & Ravi Hegde

Company: SAP Labs & Intel

Created on: 17 December 2009

Author Biographies

Salvatore Castro of SAP Labs has a Bachelors Degree in Computer Engineering and a Masters Degree in Computer Science both through the Rochester Institute of Technology. He is a member of the MII Product Management group under John Schaefer and came aboard SAP through the Lighthammer acquisition.

Ravi has been working as a Software Optimization Engineer at Intel and is responsible for characterizing & optimizing the performance of SAP Kernels. Before that, Ravi worked as a Software Development engineer at Intel developing mission critical automation applications for Intel's manufacturing plants that integrated into MES, SPC and other systems used on the shop floor. Ravi has a Masters degree in Computer Science.

Table of Contents

Overview

The scenarios highlighted in this guide are based on real world customer scenarios that are broken down into their basic components as integration scenarios. There are four individual scenarios and two end to end scenarios. The scenarios are designed to show the peak throughput performance of the MII application under various NetWeaver configurations and it's affect on the underlying hardware.

The first scenario highlights the interaction of MII when performing synchronous communication directly with the ECC system utilizing various communication technologies (JCo, JRA, and ES). The second scenario show cases the usage of XSL Transforms versus the MII Repeater and Assignment actions for transforming XML documents. The third scenario shows the ability of the MII application to accept asynchronous messages from the Plant Connectivity (PCo) software product. The fourth scenario shows the ability of the MII Messaging services to process an asynchronous message, LOIPRO IDOC, by writing contents of the document into a database table. The remaining two scenarios are designed to simulate an average and high volume user load along with periodic background processing over a four hour time period and measuring its impact on the system.

Each of the aforementioned scenarios highlight a variety of real world use cases for the MII application and is designed to show the hardware scaling capabilities and performance on an Intel based hardware platform. The affect on the hardware from a utilization standpoint was also measured for each scenario through the user of the built in Windows Performance Monitor. The measured performance monitor values from each of the tests are available upon request and are included in the primary document. The appendices of this document include detailed technical architecture and configuration concepts that were used in each of these tests.

Scenario 1 – Synchronous Communication

The purpose of this scenario is to identify the scalability of the MII architecture when performing synchronous communication requests to ECC. The ability to efficiently and synchronously retrieve data from the SAP ECC system is a cornerstone to providing visibility from a manufacturing facility to enterprise data. This scenario helps to provide insight into the size of hardware required to handle requests and process them back to an end user.

Configuration Details

In this scenario when performing these requests via the transaction servlet each requests generates a new thread specific for that transaction. Whereas if these requests were performed in Asynchronous mode the thread would terminate once the message was written into the JMS message queue. Once the message is written to the transaction queue a message driven bean is triggered in order to run the transaction that performs the production confirmation request to ECC. This transaction for JCo and JRA utilizes an existing connection pool which needs to be set according to your usage. In this scenario there were three different server involved one for SAP ECC, SAP MII, and Test Load machine. The scenario architecture diagrams for these tests are as follows:

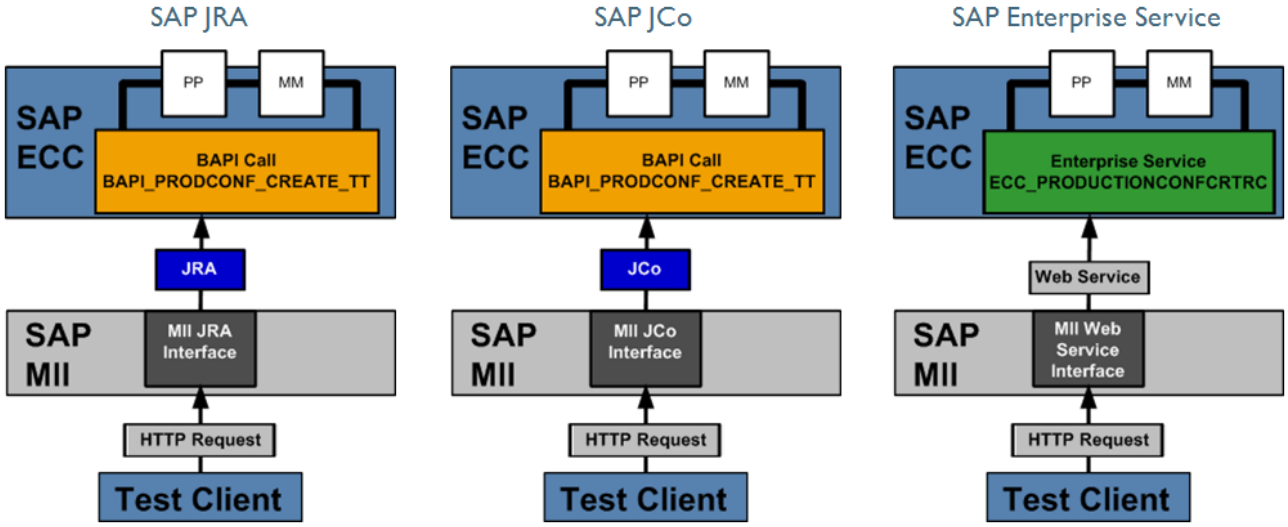


Diagram 1 - Architecture for Synchronous MII to ECC Communication

It's important to note the approach that the MII application uses to communicate to the ECC system in order to utilize each of these connection types. Both the JCo and JRA communication protocols support connection and session pooling while a web/enterprise service is designed to be stateless and not use these mechanisms. Also, the JCo connection pool is maintained within the MII application whereas the JRA connection pool is maintained in NetWeaver.

Performance Results

When measuring the performance of this setup the ECC instance was on the same network as the MII and Grinder instances but each was running on its own machine. The SAP ECC instance was running on the same hardware as the MII instance but with less RAM as it wasn't necessary for the system to handle the generated load from MII. The MII system configuration was a two socket Quad core configuration with 20GBs of RAM and for each of the scenarios the number of server nodes used and the amount of RAM assigned to each node was varied in order to illustrate it's affect on the overall performance of the application. The results from both the v12.1 and v12.0 tests are shown below side by side:

SAP NetWeaver CE with MII v12.1				SAP NetWeaver 2004s with MII v12.0	
Configuration	**Interface**	**Msgs/Sec**	**Total Msgs**	**Msgs/Sec**	**Total Msgs**
4 nodes 4GBs	ES	36.3	43631	39.6	47470
	JCo	60.8	72929	61.5	73860
	JRA	8.71	10450	60.1	72074
2 nodes 8GBs	ES	35.5	42662	34.4	41340
	JCo	66.3	79579	55.6	66727
	JRA	8.9	10687	54.1	64978
1 node 16GBs	ES	38.8	46557	26.3	31527
	JCo	67.1	80581	44.5	53420
	JRA	9.06	10875	46.2	55489

Table 1 - Scenario 1 Performance Results

The chart below illustrates the impact of varying these settings had on the system for the v12.1 test:

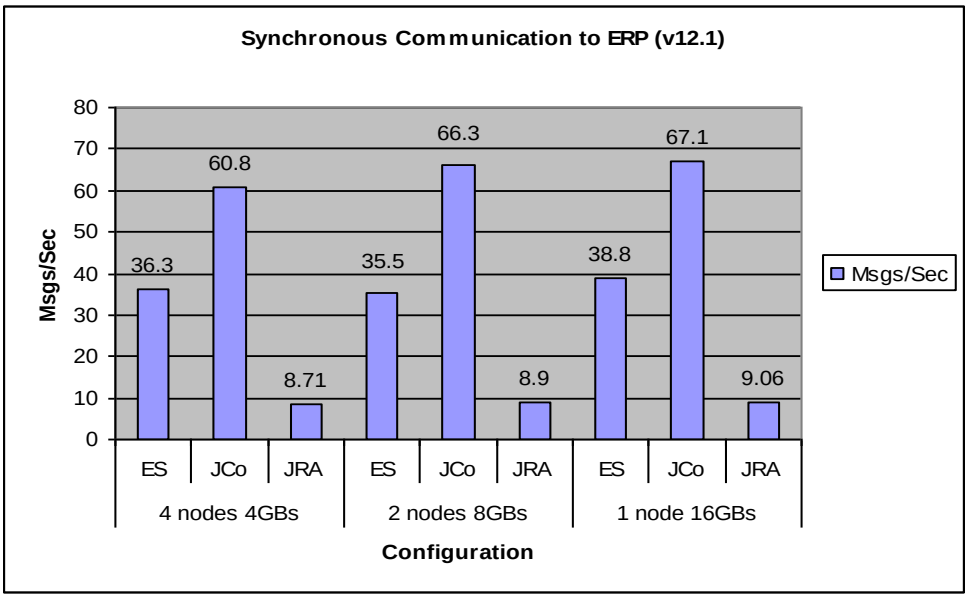


Chart 1: Scenario 1 - Performance Results by Configuration and Protocol for MII v12.1

The performance for communicating with ERP has a very clear performance bias towards JCo over both Enterprise Service and JRA requests. The primary factor for the JCo protocol to outperform Enterprise Services was due primarily to its ability to maintain an authenticated connection pool to the ERP system. The Enterprise Service request by design has to establish and authenticate a connection each time a request is made. When comparing the JCo performance to the JRA performance this yielded an unexpected

result of JRA significantly underperforming both of the protocols despite having a connection pool. When analyzing the Windows Performance Monitor logs to see the effect that this protocol has on the machine it's very clear that there's a significant amount of disk IO being performed that is limiting the throughput of the interface. This scenario was also tested using the SAP MII v12.0 product and it yielded significantly different results as shown in the chart below:

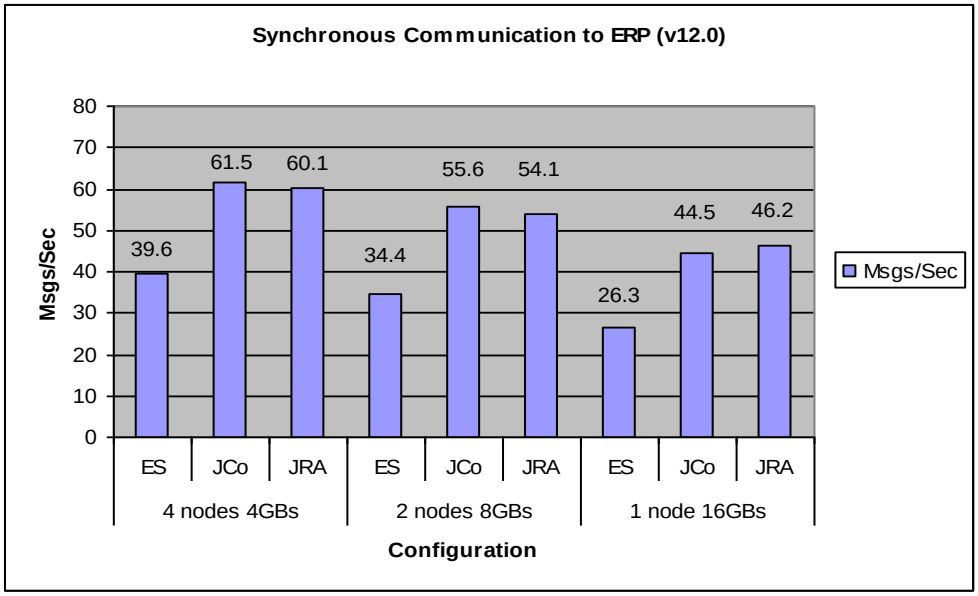


Chart 2: Scenario 1 - Performance Results by Configuration and Protocol for MII v12.0

The first major difference is the decrease in overall performance when using fewer nodes as opposed to the v12.1 configuration where performance improved. Also, the JRA protocol in v12.0 on average was able to perform with a throughput of 7x what it was able to do in v12.1.

Scenario 2 – XML Transformation

The purpose of this scenario is to help identify the performance difference between the use of an XSL transform over the MII Repeater and Assignment action blocks when transforming XML document structures. The XML document used in this scenario is very large, ~1 MB, in order to highlight the performance difference between the two approaches. Once the manipulation of the XML is completed the transaction writes an entry into the SQL DB table to indicate that the operation was successful and to help track the performance statistics.

Configuration Details

In this scenario when the test client makes a call, a new thread specific for that request is generated for a synchronous HTTP Request. There were two servers involved in this scenario one was the SAP MII Machine and the other was the Test machine that generated the test load. The configuration from a high level for this scenario is shown below:

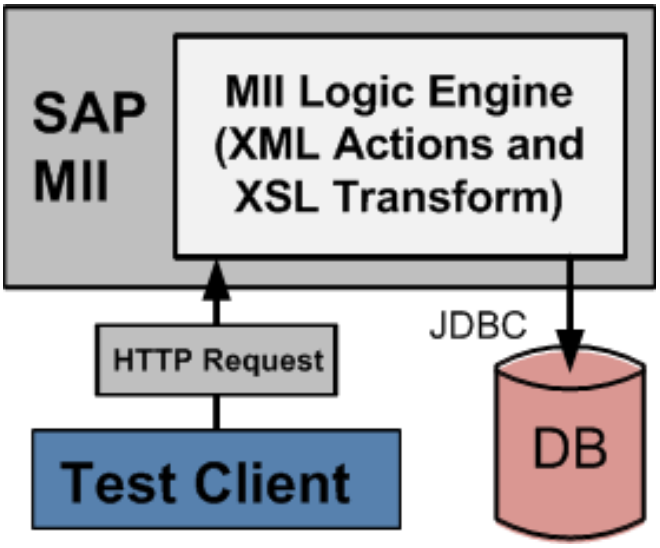


Diagram 2 – Architecture for XML and XSL Based Document Transformation

The above configuration was focused on replicating how many large XML documents can be moved through the system and their content written to a database table.

Performance Results

This was a v12.1 only test and it outlines the performance difference between various configurations of NetWeaver affect the performance of your application. The overall throughput for the two different implementation approaches was similar for each of the configurations as shown in the grid and chart below:

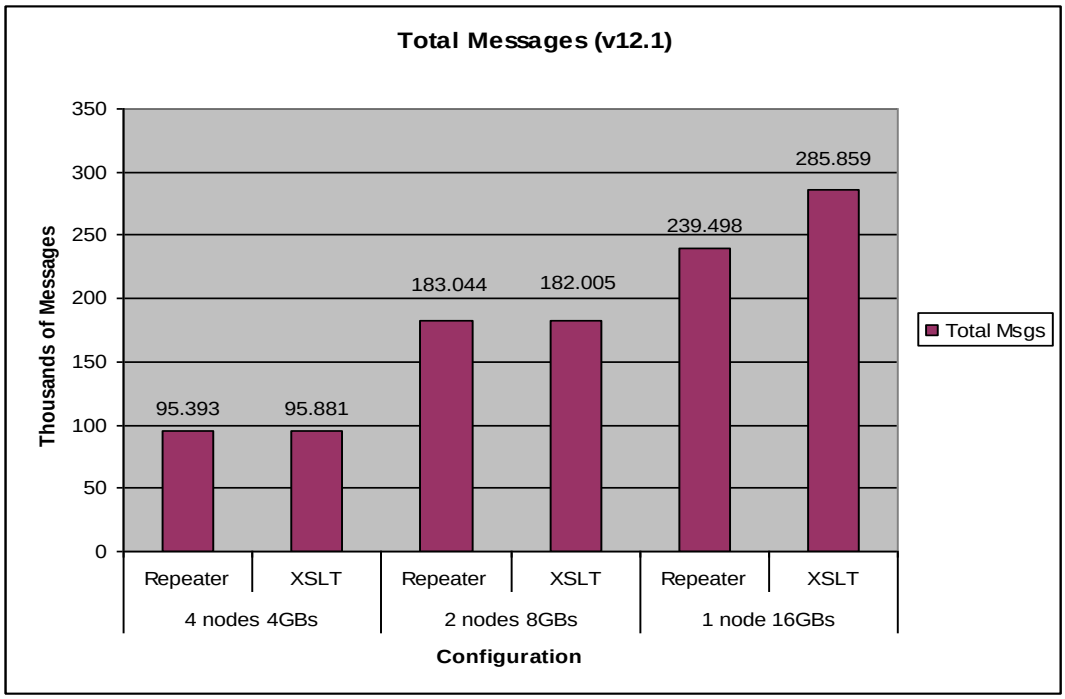


Chart 3: Scenario 2 – Performance Results by Configuration and Design for MII v12.1

However when analyzing this scenario the throughput over the two hour test duration was not as important as the impact that the approach had on the system resources. The chart below helps to highlight the impact on the amount of available system RAM during the test.

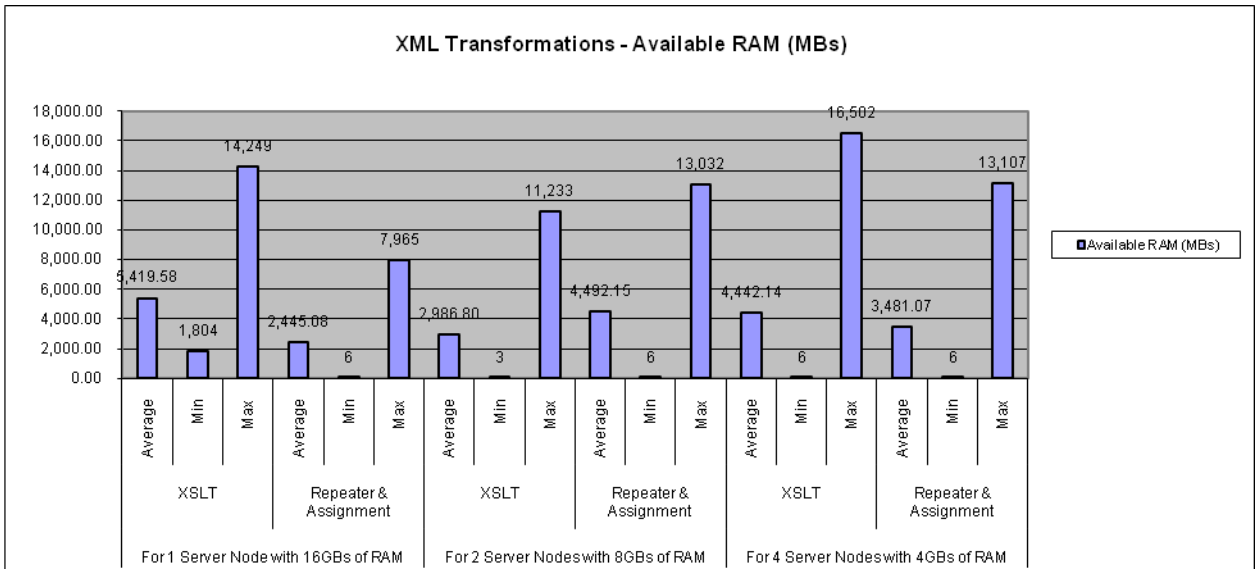


Chart 4 - XML Transformations Available RAM

From this it's important to note that on average there is more available RAM with the XSLT action approach. The next chart highlights the average CPU usage during the test and once again the XSLT action approach

demonstrates a smaller impact on the system.

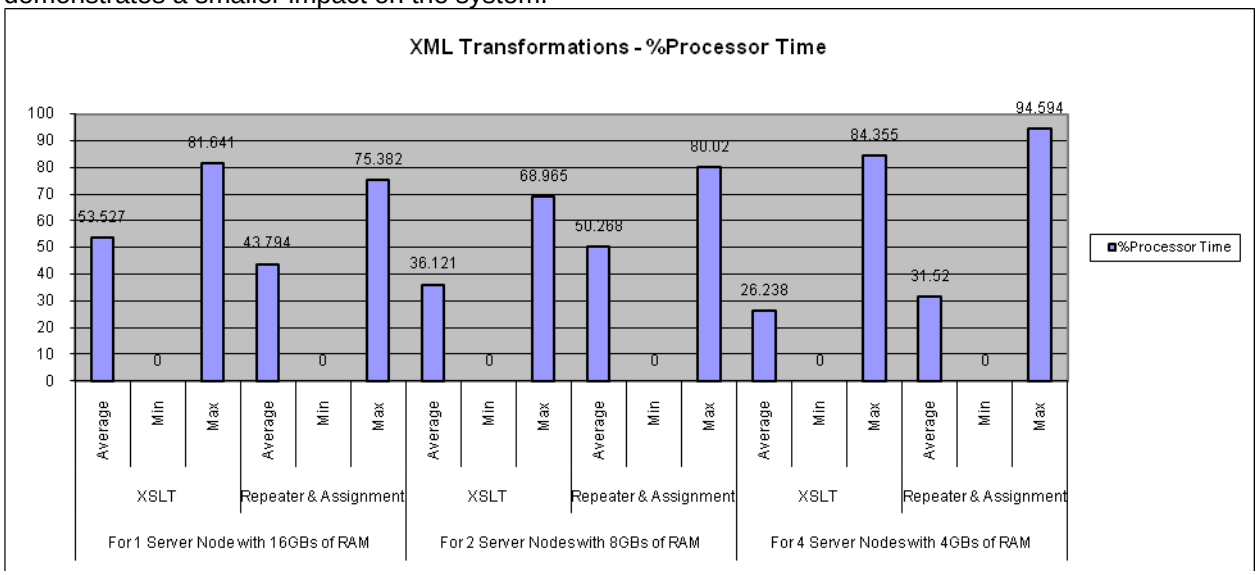


Chart 5 - XML Transformations Percent Processor Time

The primary impact on the overall performance of the two designs is shown with the smaller footprint and higher throughput with the XSLT Action approach over the Repeater with Assignment Actions approach.

Scenario3 – PCo Communication

The purpose of this scenario is to showcase the use of MII in a local or regional implementation where multiple PCo Agents are passing messages to the MII Application. The number of successful messages that can be handled successfully during peak volumes are shown. In this specific use case the amount of system overhead as it relates to RAM usage on the system for each server node is minimal as compared to the scenarios where larger XML document processing is required. As a result there is an additional test that was done with eight server nodes each with 2GBs of RAM assigned to them in order to illustrate how well the performance scales with the extra nodes.

Configuration Details

The scenario shown in the diagram below illustrates the configuration that was used in order to perform these tests. The OPC Historian and PCo instances are running on one separate machine and the SAP MII and MS SQL Database are both running on a second machine.

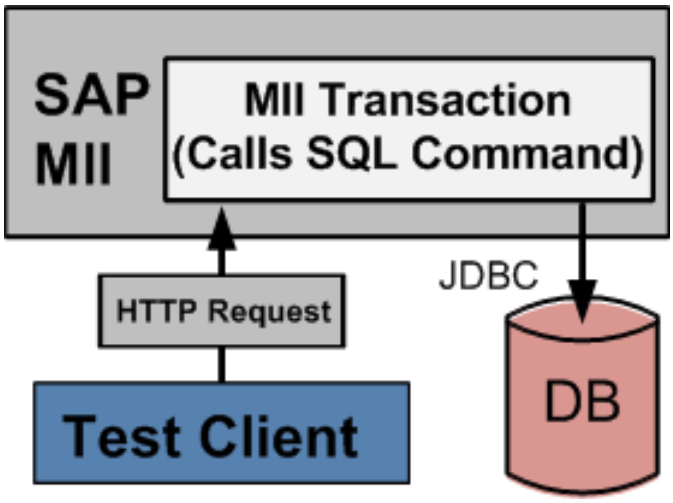


Diagram 3 - Architecture for PCo Message Processing

The performance of the PCo and Historian server were not measured since their overhead involved was not important for the scenario. The primary test for this scenario is to show how many PCo XML messages the MII server can process over approximately a two hour time period. This was done in order to help size the MII server required to process a certain number of messages from one or many PCo agent instances.

It is not possible to start all of the PCo Agent instances at the same time since each instance operates as its own windows service. As a result a windows batch script was written to start them as quickly as possible in a sequential manner and windows tasks were created to start and stop all of the services after approximately a two hour interval. The performance throughput measurements for this test do not come from the PCo Agents but instead are based on when the messages appear in the database table. The query takes the total number of records divided by the number of seconds between the first and last timestamps of all of the records.

Performance Results

The PCo test consisted of a variety of tests and scenarios that initially started with using a PCo instance and was then changed to use "The Grinder" to simulate a PCo agent request XML in order to generate and monitor a larger number of XML messages. The results from this scenario were very similar to the previous scenario results when comparing across the various configurations as shown below:

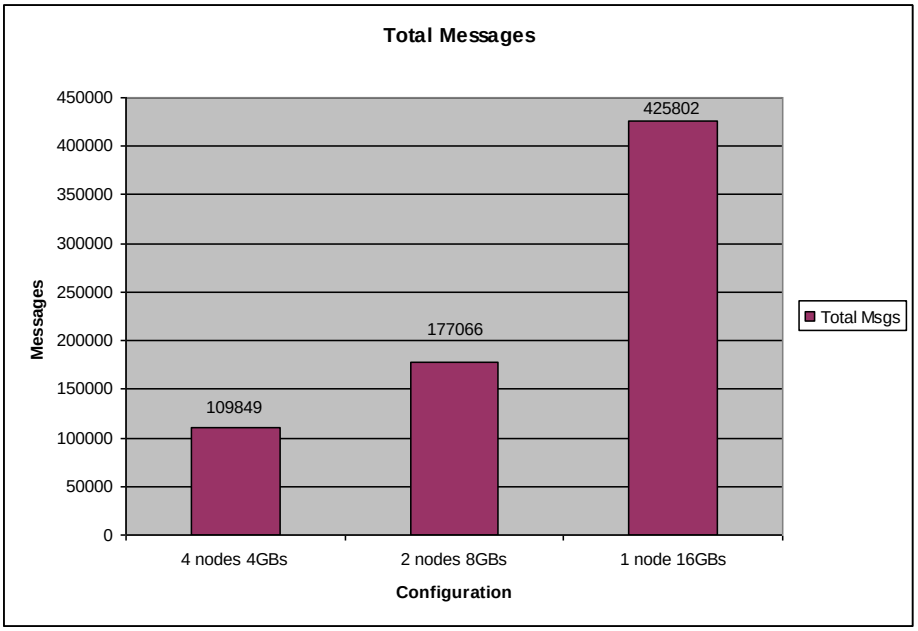


Chart 6: Scenario 3 – Performance Results by Configuration for MII v12.1

The results from this test illustrate a major performance gain between a multiple JVM node configuration and a single JVM node configuration. When reviewing the Window Performance Monitor Logs the Percent Processor Time and Available RAM both confirm the throughput results. For the Percent Processor Time results show that on average it was utilized more in the single node configuration meaning that there was less system overhead to process the various application threads. The values are shown in the chart below:

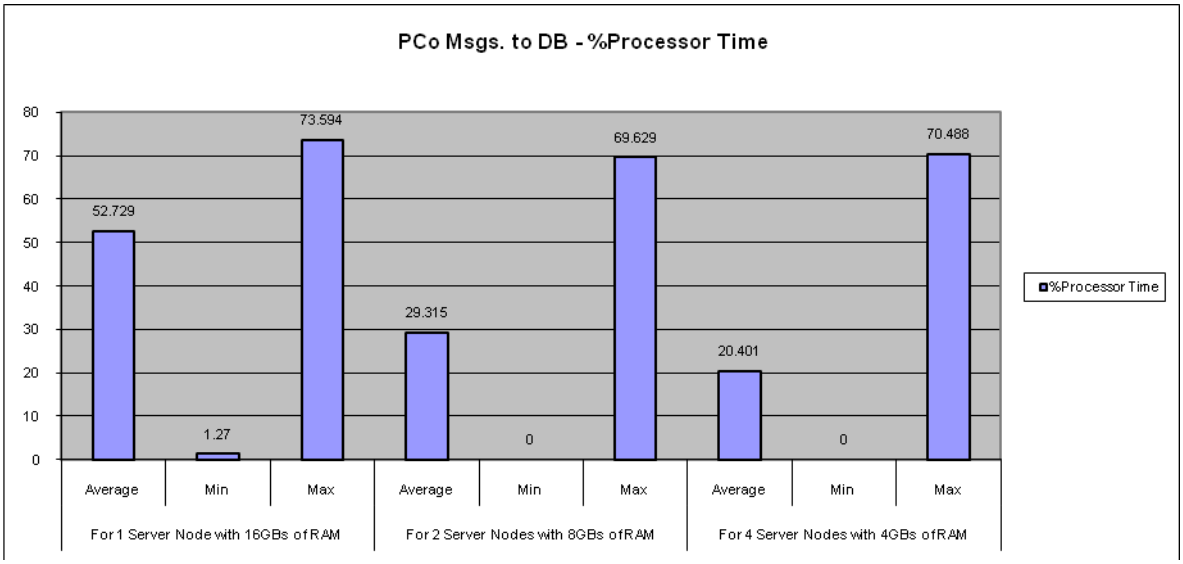
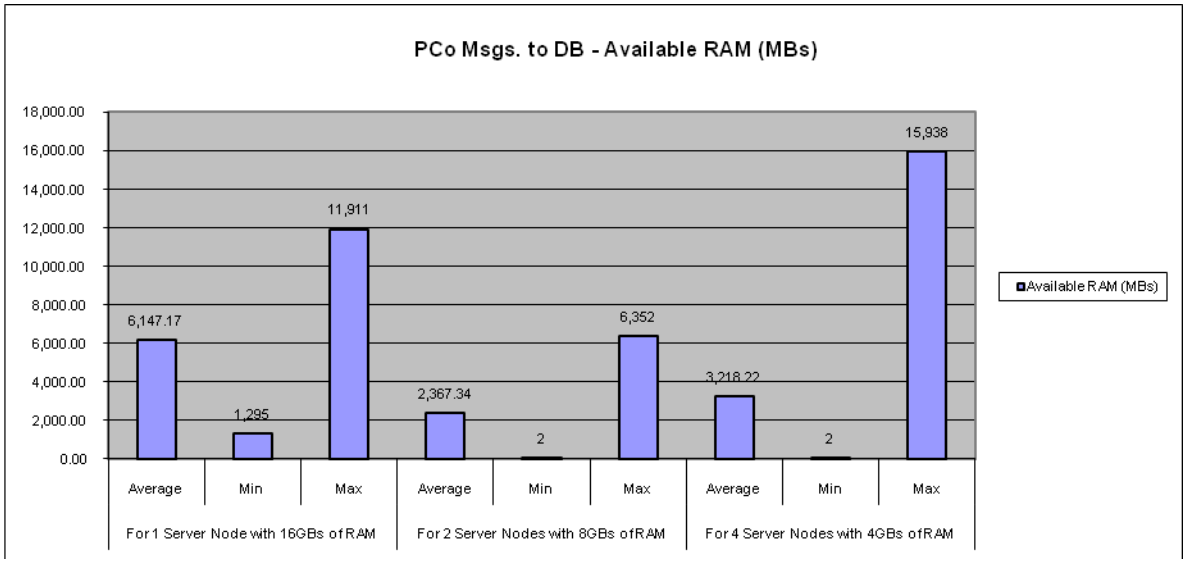


Chart 7 - PCo Messages to a Database Table Percent Processor Time

With regards to the overall amount of System RAM that was required in each of the scenarios there was a significant difference with these values as well. In order to properly read the chart the Max value represents the amount of available RAM when the NetWeaver node is first started on the system. Then as the test goes

on the Average amount is 20GBs (Total System RAM) minus what is used by the NetWeaver application. The Min value is what happends at the peak of the test before the system begins to page out information from Memory to disk and impacting the overall throughput performance. In the case of the multi node configurations this point was reached in the test where only 2MBs of free RAM was available. This is shown in the chart below:



The additional threads and handles that are generated in a multiple JVM node configuration add to the overhead on the system and under high stress negatively affect its overall throughput performance.

Scenario 4 – Asynchronous Messaging Services

The purpose of this scenario is to demonstrate the scalability of the asynchronous messaging interface and how well the MII application scales on the NetWeaver CE platform. When performing this test there are multiple different thread pools that are involved and each must be scaled properly in order to handle the load on the system.

Configuration Details

The scenario shown in the diagram below illustrates the configuration that was used in order to perform these tests. It consists of two separate machines the first with "The Grinder" on it that is used to generate the load on the SAP MII and MS SQL DB server (The JMS instance is part of the NetWeaver Platform and used behind the scenes by MII). The high level configuration of systems in this scenario is shown below:

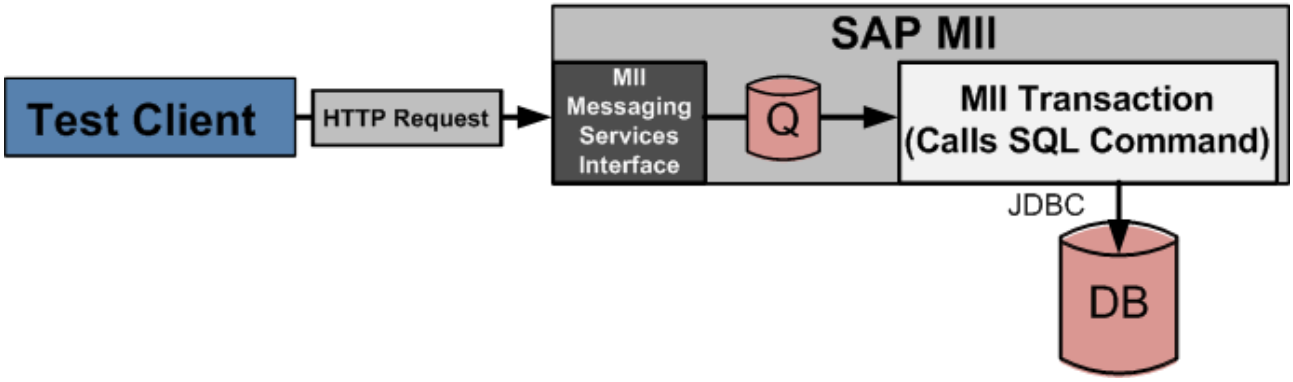


Diagram 4 - Architecture of the Asynchronous Messaging Interface

In this scenario the JMX pool plays a major role in controlling the scaling/throttling of the number of threads that get generated at any given time of the system. Since the request thread terminates once the request is written to the Queue the queue's threading model can control how many different threads are created at any given time in order to process each message without instantly overloading the system.

Performance Results

This scenario was designed to demonstrate the amount of time it would take to process 250,000 XML messages through the MII Messaging Services interface. This test was performed in both the SAP MII v12.1 and v12.0 products in order to provide a comprehensive overview of how best to configure your system and the benefits of v12.1. The performance results of v12.1 are shown in the chart below:

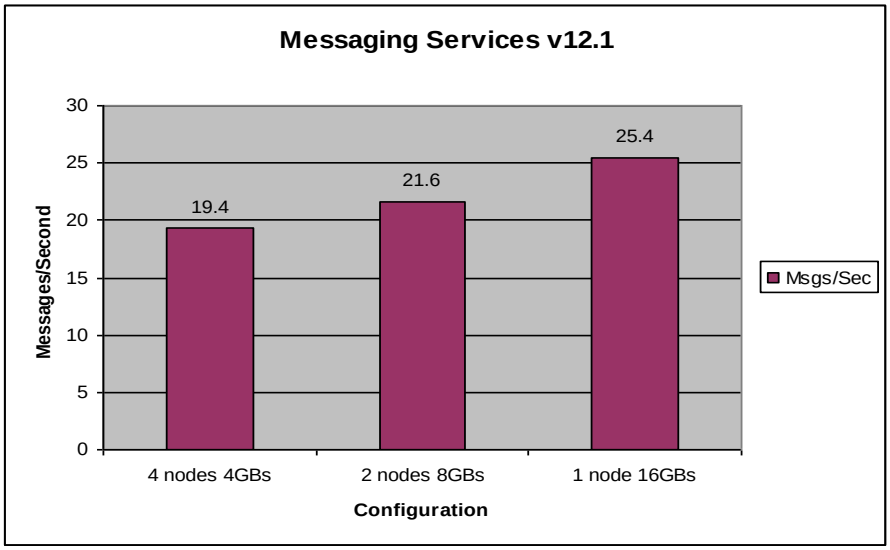


Chart 7: Scenario 4 – Performance Results by Configuration for MII v12.1

The above results demonstrate the same results as the previous scenarios of the single JVM outperforming the multiple JVM configurations. Additionally the load on the system resources in the multiple JVM configurations has a greater average memory usage which greatly limits the duration that the test would be able to continue before the system began paging. The RAM utilization in the single node configuration leveled off during the test and was able to sustain without paging for the entire duration of the test. The chart below demonstrates this by showing the Average and Min available RAM as similar values for the Single node test:

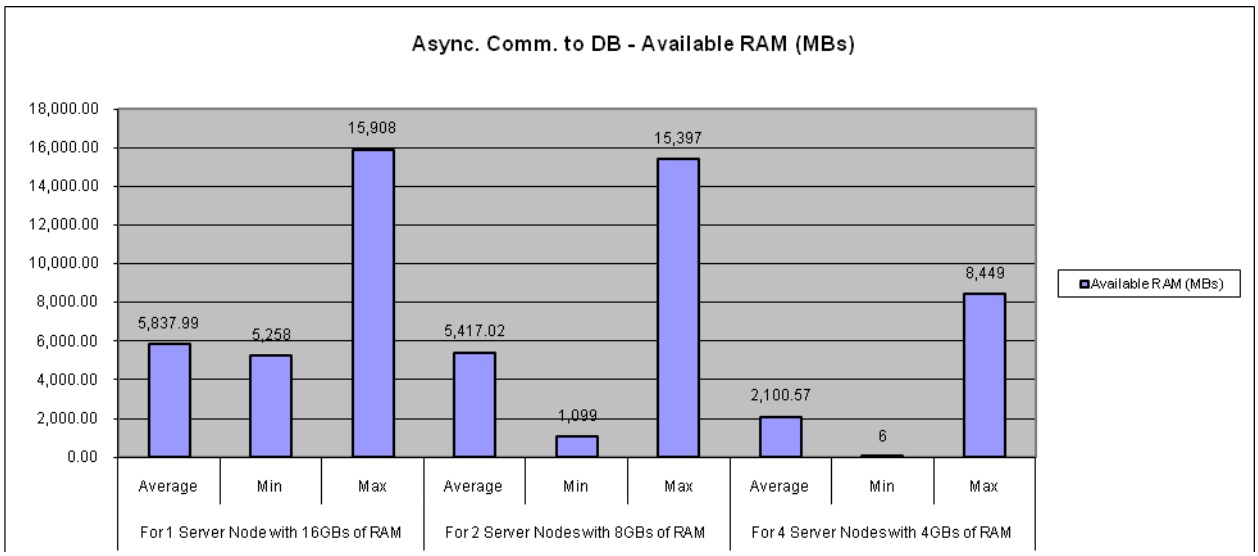


Chart 8 - Messaging Services Available RAM for v12.1

The results from the v12.0 performance test are shown below and the trend is the opposite from the results of the v12.1 test:

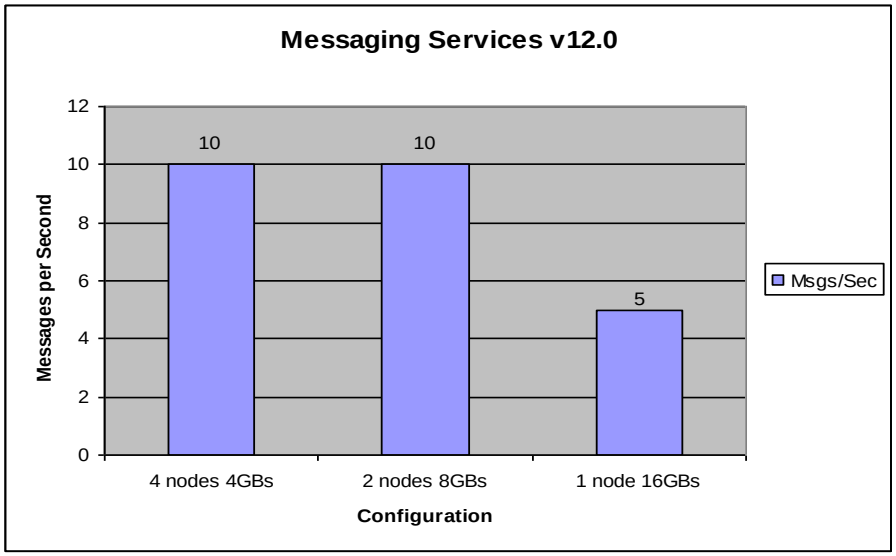


Chart 9: Scenario 4 – Performance Results by Configuration for MII v12.0

They show that in this version of the software the maximum number of messages per second to be processed through this interface was 10 compared to ~25 in v12.1. Additionally the load on the system resources in any of the JVM configurations has a greater average memory usage and similar minimum values which indicate lots of system paging (Paging results not shown but were around 30k pages/sec in each test). The chart below outlines the impact of available system RAM during the test:

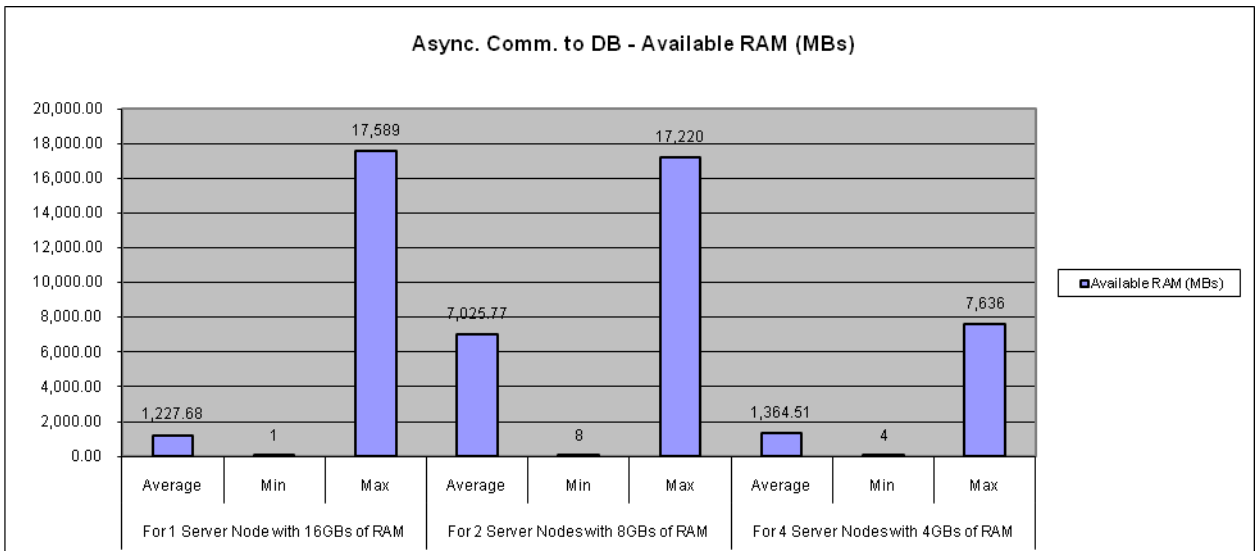


Chart 10 - Messaging Services Available RAM for v12.0

These results demonstrate a clear benefit to the MII v12.1 product performance for this interface by more than doubling the throughput capabilities for this interface.

Scenario 5 & 6 – High & Average Number of Users Accompanied by Periodic Background Messages

This scenario is designed to mimic 500 web users (Lots of Users) or 100 web users (Average Number) clicking around on a web page with approximately 5 applets for each page. The users were simulated using The Grinder and then there are IDoc transfers that occur in the background as well which are triggered from The Grinder. The formula used to estimate an average user load was (500 users * 4 Applets on a page * 1 click) / 10 seconds = 200 per second for half the users and (500 users * 4 Applets on a page * 1 click) / 30 seconds = 67 per second for the other half. The background processing load was set for 1 message every 5 seconds or 720 per hour which is on the high side of the average usage.

The applets generate three URL requests to the MII application server. Two are to retrieve the Query and Display templates and the third is to retrieve the data in binary format. Since this test was not a throughput based test each scenario configuration was run for four hours. For details on the script that was used see the Appendix of the main document.

Configuration Details

The scenario shown in the diagram below illustrates the configuration that was used in order to perform these tests. It consists of two separate machines the first with “The Grinder” on it that is used to generate the load on the SAP MII and MS SQL DB server (The JMS instance is part of the NetWeaver Platform and used behind the scenes by MII). There were two separate tests going on at different intervals and their performance was tracked by windows Performance Monitor and The Grinder.

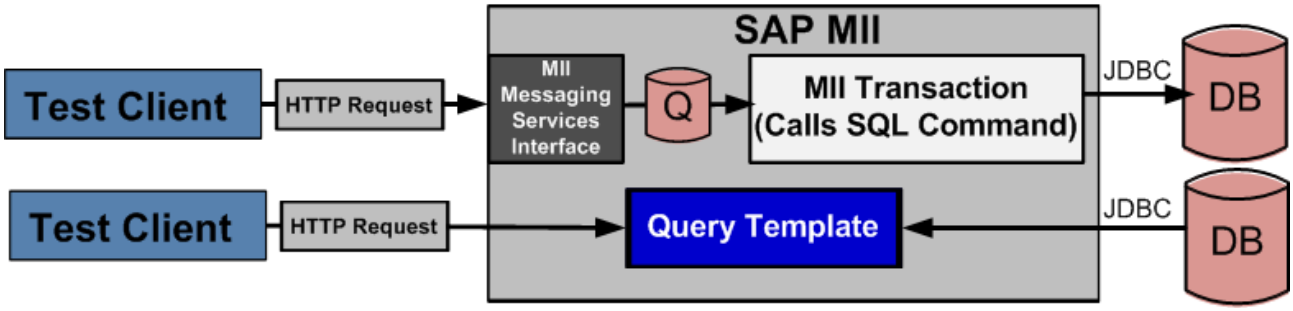


Diagram 5 - Architecture of the Real World Scenario

Performance Results

These scenarios were run in order to record the impact that a large number and an average number of simultaneous users have on the application server. The results of this test indicate that this server configuration can operate reliably under a constant load of 100 and 500 simulated users for long periods of time. The results of the high user volume test are shown below:

		Pages/Sec	Disk Queue Length	%Processor Time	Available RAM (MBs)
For 1 Node with 16GBs of RAM	**Average**	0.119	0.398	13.334	5,004.33
	Min	0	0.003	0	4,892
	Max	151.483	29.763	15.684	5,070
For 2 Nodes with 8GBs of RAM	**Average**	0.02	0.382	13.381	9,265.09
	Min	0	0.003	0	9,179
	Max	0.80	29.776	15.879	9,353
For 4 Nodes with 4GBs of RAM	**Average**	0.061	0.374	13.959	9,133.99
	Min	0	0.004	0.078	8,939
	Max	60.38	29.63	32.422	9,943

Table 2 - High Number of Users and Background Message Processing

As shown in the above table the average processor load generated from this configuration was very low and did not have any major impact on the machine. When compared to the results from the 100 user test the processor utilization is around one fifth the percentage which demonstrates that at this user load the application scales linearly. The results of the Average Number of Users test are shown in the table below:

		Pages/Sec	Disk Queue Length	%Processor Time	Available RAM (MBs)
For 1 Node with 16GBs of RAM	**Average**	0.712	0.224	3.02	15,963.45
	Min	0	0.003	0.039	15,767
	Max	309.004	30.155	13.223	16,476
For 2 Nodes with 8GBs of RAM	**Average**	0.485	0.443	59.512	10,476.34
	Min	0	0.005	0.078	10,221
	Max	1.40	20.562	76.387	10,790
For 4 Nodes with 4GBs of RAM	**Average**	0.019	0.295	3.218	8,880.85
	Min	0	0.003	0.059	8,783
	Max	0.80	30.294	4.766	8,978

Table 3 - Average Number of Users and Background Message Processing

The performance monitor logs for these tests for all of the configurations indicate a steady state of system resources throughout the entire four hour testing duration. It is important to note that for the 100 user test with 2x8GB test the processor spiked due to a system process that was kicked off during the test. This did not impact the request and response times of the test nor did it affect other aspects of the machine resources.

Closing Remarks

Whenever you're planning on putting together a hardware proposal it's always good to keep in mind that the MII software is an application framework so plan for the future as opposed to the near term. In my experience an MII implementation starts off relatively small and then continues to grow at a steady or exponential pace from there. Be sure to stay in tune with what the business roadmap is for the software in order to effectively plan and manage your hardware upgrade process. Standard IT practices for incremental environment upgrades for large corporations where software is "promoted" from a low performance server to a medium and high performance machines are common and this method applies to the MII product as well. For smaller companies where less existing hardware investment can be made it's important to get the sizing right the first time rather than have to go back to the business in a year to get budget for additional hardware. If you take the cost up front this can actually save you money by reducing the amount of incremental investment required to keep up with your applications.

The SAP MII product according to the results of these tests shows that in an Intel Xeon platform like this the system performance is based primarily on available RAM. This simple fact means that performance is memory driven and the more expansion room that you provide for RAM the better off you are. This is good news since the price of additional RAM is much cheaper than the other primary system components so be sure to size your system horsepower so that RAM is your limiting factor as well as this will allow you to scale your server in a cost effective manner.

When you scale the RAM on the system it's important to note that this will also increase the demand on the HDD subsystem due to increased server utilization which can increase the amount of paging that is required. In this configuration when the system started paging the application server began to struggle with the operating system for resources, as shown by the Performance Monitor logging files located in Appendix C of the main document.

When sizing your hardware and planning your landscape it's recommended to take the Ben Franklin approach of "A stitch in time saves nine". The onetime cost of hardware is minimal compared to the ROI that is provided back to the business by having efficiently running and a properly maintained application. A substantial amount of time and energy can be wasted simply by having people waiting behind a status bar for their daily reports to load either due to a poor network or an underpowered server. Keeping ahead of the curve while it costs money up front saves money over time.

Finally, "What gets measured gets improved", having a strong hardware and software administration team is vital in maintaining the health and performance of your investments. It's not always an underpowered server that is to blame for inadequate application performance. This can be the result of poor architecture and implementation techniques that may be lost in the implementation details. In order to keep the application development and users aware of the implications of their actions, monitoring and reporting of simple usage statistics can help to alleviate unnecessary system load. This will help to prolong the useful life of your hardware and at the same time improve the skills of your development team.

Appendix A – Detailed Hardware Configuration

Hardware Specifications

The following table shows the detailed hardware configuration of the two servers on which the MII and the ERP instances were installed.

Intel Hardware Platform	
Product Name	Intel® Server Systems SR1560SF
Processor	2 X Quad-Core Intel® Xeon® processor 5400
Processor Frequency	3.16 GHz
Number of Sockets	2
Cache	12MB L2 cache per processor
FSB Frequency	1333 MHz
Memory	20GB
Memory Type	2GB ,800MHz, FB-DIMM
Number of DIMMs used	10
Number of Disks Used	1 X 73.4GB 15K RPM SAS 1 X 450GB 10K RPM SAS
Form Factor	1U Rack Optimized Server
Operating System	Microsoft Windows Server 2003 Enterprise x64 Edition Service Pack 2
SAP Software	SAP MII v12.1.1 on NetWeaver CE EhP1 SP1 SAP MII v12.0.7 on NetWeaver 2004s SAP Plant Connectivity v2.0.1.1 SAP DB v7.7

Table 4 - Detailed List of Testing Hardware Used

The memory configuration of the Intel Xeon System was varied depending on the scenario that is being tested. This was done by varying the amount of RAM and the number of NetWeaver JVM instances created for testing each scenario. The results from this test highlight the most efficient configuration for utilizing available hardware resources to process the load put on the system during the tests.

It's important to note that each of these scenarios showcase dedicated servers connected via a closed network environment on a dedicated Gigabit switch. This was done to remove the network as a factor to accurately measure the performance of the software involved.

Software Configurations

The operating system that was used in these tests was Microsoft Windows Server 2003 Enterprise Edition and the NetWeaver CE and 2004s installations were based on the MaxDB (formerly SAPDB) database. For more information about this database product see the following URL: https://www.sdn.sap.com/irj/sdn/maxdb.

When scaling the NetWeaver platform for one JVM instance to utilize available RAM in the system for a single JVM node the following allocations for Java Maximum Heap space were used: 4096, 8192, and 16386. The second system scaling method is to create multiple JVM instances in order to create a cluster within the NetWeaver instance. This approach should allow for greater utilization of the multiple processor cores available on the system and in these tests a combination of both approaches was taken. There were also some changes that were made in the NetWeaver application server configuration to handle the high volume of messages being passed through it. These changes were made using the **ConfigTool** that comes with each NetWeaver installation. This tool is located in the following place on the NetWeaver installation <drive>:/usr/sap/<SID>/JXX/j2ee/configtool/go.bat

When interfacing to the MII Application there are four thread separate pools that are directly affected and they are the Internet Connection Monitor (ICM), Application, JMX Resource, and Data connection pools. Then depending on your system load and available resources that you can dedicate to the process you can determine the appropriate settings. The diagram below outlines the areas in the MII application that the different thread pools are utilized. This should help you to tailor your installation based on where the anticipated load will be to allow for the greatest scaling and performance of the application.

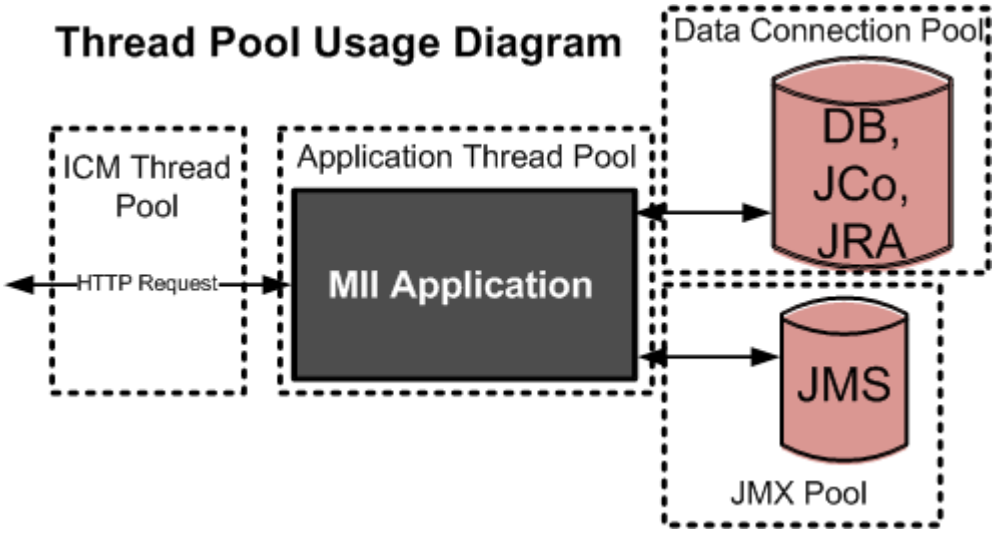


Diagram 6 - NetWeaver Platform Thread Pools Involved

The configuration tool for the instance the following parameters were modified from their default values in order to accommodate the heavy system load that was encountered during these tests.

These parameters were specific to this hardware and may vary for your environment; the main purpose of the table below is to help identify parameters that can be modified to affect scalability and performance:

Path (Instance)	Parameter	Default	Custom Value
Services -> IIOP	Parallel Requests	10	1000
Services -> IIOP	Request Queue Size	100	1000
Services -> p4	Parallel Requests	10	1000
Services -> p4	Request Queue Size	100	1000
Services -> security	Max Sessions	200	1000
Services -> cross	MaxServiceThreads	5	5
Services -> jmsconnector	Max Connections	100	500
Services -> jmx	Notification Queue Max Threads	3	30
Services -> jmx	NotificationQueueThreadThreshold	30	100
Services -> dbpool	sysDS.initialConnections	1	10
Services -> dbpool	sysDS.maximumConnections	40	50
Services -> servlet_jsp	ProductionMode	False	True
Services -> http	FCAServerThreadCount	5	50
Managers -> ApplicationThreadManager	MaxThreadCount	100	1000
Managers -> ApplicationThreadManager	PercentOfParallelismAllowed	30	50
Managers -> ThreadManager	MaxThreadCount	100	1000

Table 5 - NetWeaver CE ConfigTool Parameters

Detailed information about each of these parameters can be found in the online NetWeaver Help Documentation here: http://help.sap.com/saphelp_nwce10/helpdata/en/
The other area where parameter adjustments that are required are made in the NetWeaver profile which is located in <drive>:\usr\sap\<SID>\SYS\profile and the file is <SID>_<JXX>_<Server>. This is a plain text file so you simply need to open it in any text editor; **before modifying this file be sure to first backup this file**. The following parameters were modified/added to the profile:

```
#-----------------------------------------------------------------------
# http port configuration
#-----------------------------------------------------------------------
icm/server_port_0 = PROT=HTTP, PORT=<The NW HTTP Port To Use>, TIMEOUT=600
#-----------------------------------------------------------------------
# Number of usable threads
#-----------------------------------------------------------------------
icm/min_threads = 100
icm/max_threads = 1000
#-----------------------------------------------------------------------
# Increase the size of the MPI memory area
#-----------------------------------------------------------------------
mpi/total_size_MB = 500
#-----------------------------------------------------------------------
# Number of maximum connections, length of the wait queue
#-----------------------------------------------------------------------
icm/max_conn = 2500
icm/req_queue_len = 2500
#-----------------------------------------------------------------------
```

These parameters are referenced in the SAP NetWeaver CE Help documentation on the ICM for systems with an anticipated “High Load” as described here:
http://help.sap.com/saphelp_nwce10/helpdata/en/56/2e453cabf4ef6fe10000000a114084/content.htm
The modifications allow the NetWeaver platform to scale to the necessary level under high load volumes. In addition to modifying NetWeaver settings it is also possible to modify various Windows OS settings to perform better, however in this case it slowed down the operation of the server. As a result of this these

changes were not implemented for the tests and are only mentioned because they may affect a different configuration in a positive manner. The probable cause of this performance decrease was due to an underpowered HDD subsystem which was unable to provide the necessary initial paging space required for this configuration. However, it is possible to utilize Large Page files which will allow a more static paging approach which can benefit a dedicated server. In order to accomplish this, the "Lock pages in memory" privilege has to be set for any user that runs your application, this includes administrators. To set this up follow these steps:

1. Select Control Panel -> Administrative Tools -> Local Security Policy
2. Select Local Policies -> User Rights Assignment
3. Double click "Lock pages in memory", add users and/or groups
    a. The following were added:
        i. Administrator, daaadm, **<SID>**adm, sapadm, SAPServiceDAA, SAPService**<SID>**
4. Reboot the machine

Finally, the MII data server configuration that was used to connect to the MS SQL 2005 Enterprise Edition database for writing the XML data values in the scenarios is:

Name	Value
Connector	IDBC
ConnectorType	SQL
DatePrefix	'
DateSuffix	'
DaysRetention	7
Description	SQL Server 2005 Enterprise Edition
Enabled	T
InitCommand	
InternalDateFormat	yyyy-MM-dd HH:mm:ss
JDBCDriver	com.microsoft.sqlserver.jdbc.SQLServerDriver
MaxRetryCount	5
Name	LoadTestTop
PoolMax	250
PoolSize	100
RetryInterval	60000
ServerPackage	com.sap.xmii.Illuminator.connectors.IDBC
ServerURL	jdbc:sqlserver://localhost:1433;databaseName=LoadTest
Timeout	60
UseCount	1000
UserName	sa
ValidationQuery	SELECT GETDATE()
WaitTime	30

Table 6 - MII Data Server JDBC Connection Settings

These custom values are only suggestions and what was configured for each of the tests, your specific values may vary due to different available hardware and load levels. In general modifying these parameters will result in improved application performance by eliminating software imposed bottlenecks, most notably the number of threads allowed to be created and the size of the request queues. It is important to keep in mind that just because you can scale these values to whatever size you wish it doesn't necessarily mean that the bigger the better. Having too many threads generated on your system can lead to additional system overhead and the actual overall throughput may degrade with the number of simultaneous threads executing on your system due to an increased amount of context switching. It is possible to monitor the number of threads, handles, and context switches per second via the Microsoft Windows Performance Monitor and is a good tool for evaluating these settings for your specific needs. For details on the Performance Monitor logs see Appendix C of the main document for the counters that were captured and the recorded log results.

The Grinder Configuration Details

There is no shortage of tools available for load testing a web application but for a couple of reasons “The Grinder” was selected as the testing tool for the various scenarios. The primary reason was that it’s an open source tool that anyone can use it to replicate these scenarios in their environment for benchmarking their application. The other reason what that it was quick and easy to setup along with lots of python script examples on the website, http://grinder.sourceforge.net and included in the Appendix of the main document. It’s worthwhile to check them out since their operation is referenced in this section and will help provide insight on the details of the tests that were executed. The Grinder test tool has the ability to run threads as fast as you want or you can throttle the frequency of threads and their execution by adding in static or random wait delays. It is also possible to quickly modify the number of processes and threads along with run counts by specifying the value in a plain text properties file to gain lots of control over the generated load. Each test has its own set of scripts and properties files depending on the goal of the test and to remove any dependency on each test for quick and easy scenario replication.

Appendix B - MII Implementation Details

Overview

The MII product supports multiple methods in which to interface to the application and depending on the needs of your implementation some are better choices than others. When interfacing MII to either trigger an event or retrieve data it is important to consider whether or not the returned data from the request is important or not. This has a measurable impact on performance and also on the generated network load. Below is a diagram of how both synchronous and asynchronous HTTP requests for the SAP MII Transaction Engine each time a request is made.

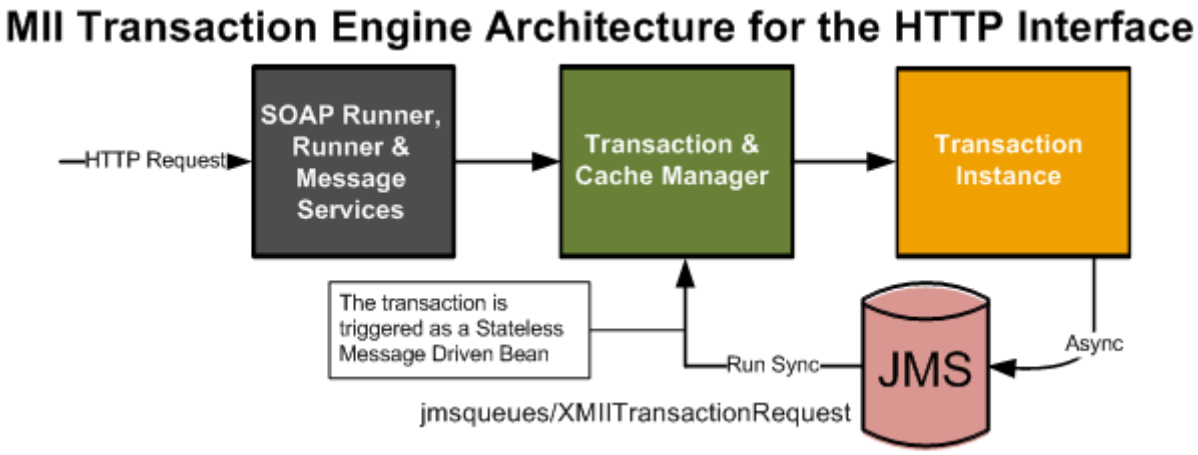


Diagram 7 - Internal Architecture of the SAP MII Transaction HTTP Interface

If the response data is important then a synchronous request to the MII Runner Servlet or Web Service interface via an HTTP request are the recommended choices. However if the response isn't necessary other than to acknowledge the receipt of the message then the Messaging Services interface or additionally in versions v12.1 and newer calling the transaction via Runner Asynchronously is the better choice. The reason being is that the calling application will not need to wait until the message is processed and can continue performing other tasks. It is important to note that the PCo Framework communicates with the MII Application via the Runner Servlet interface and The Grinder tool was used in each of the scenarios because of its ability to generate a very large load and track the results. Depending on the test that was being performed the Runner Servlet or the Messaging Services interfaces were the targets of the scripts. For more information about the scripts used and the configuration of The Grinder see Appendix A of this document.

The mechanism used to trigger the type of transaction being run along with the Persistence model of the transaction instance can also affect the performance of the transaction. The transaction persistence model is new to MII in v12.1 and allows for the end user to view the current and previous execution states for transactions that have run on your system. It provides detailed insight and overview of each transaction but is only necessary in production if there is an error during execution. As a result the Persistence parameter for all of the tests is set to "ONERROR" to limit the execution tracking overhead and to accurately represent a real-world implementation.

Plant Connectivity (PCo) Agent Configuration Details

For the PCo load testing the software was installed on a different box from the MII and ERP instances in order to better demonstrate the operation of the two software components in a real-world scenario. For the configuration of the agent instances and communication methods to MII there are a variety of settings that affect overall throughput and performance as mentioned in the previous section. When configuring an agent it is important to keep in mind the business use case and in doing so will help you understand why the settings for this scenario were chosen.

To give some background on the PCo software an event agent is designed to "subscribe" to a set of tags that exist in a data historian. The agent will monitor the tag(s) values for conditional violations and send an XML message to the MII Transaction Engine to process the message. The transaction can update a local event database or utilize some other means to persist the event so that an end user can review the current state of

their manufacturing environment. This can be very useful in detecting quality and performance anomalies that can occur during the normal manufacturing process and eliminates the need to poll the data historian which decreases the overall load on the historian system. The typical installation architecture for the PCo software is shown below:

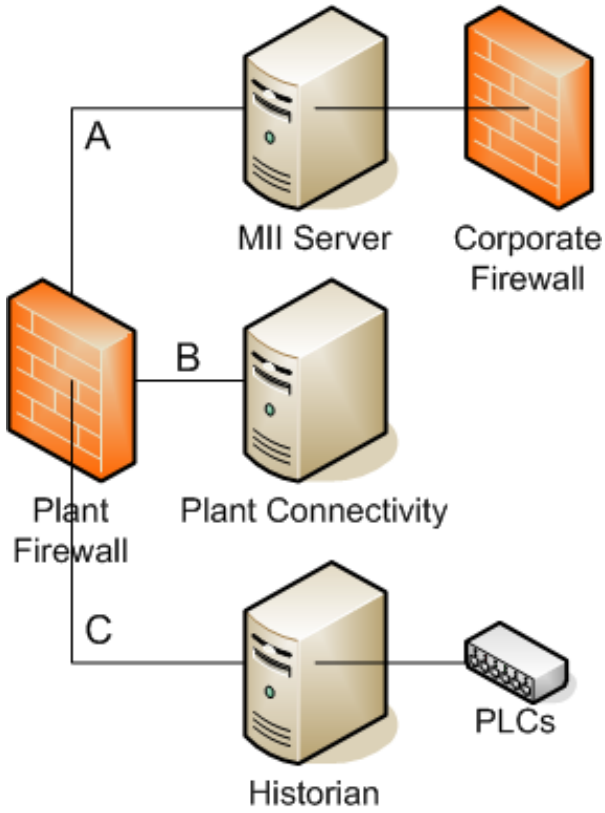


Diagram 8 - Typical Plant Installation Architecture

This installation architecture is based on the ability of the network to control what software can and cannot communicate with the Historian subnet. In the above diagram subnet 'A' can only communication with subnet 'B' and subnet 'C' can only communicate with subnet 'B'. This helps to prevent any system from gaining unauthorized access to the critical shop-floor control system. This installation architecture maintains the necessary network security and still enables the MII Application Server to access the necessary data from the control system at the same.

In the PCo testing scenario values from the received XML event messages were written to a local SQL server database as this is the most popular use for them. The design of the scenario was for the PCo software to only responsible for the delivery of the message to the MII Application which receives and processes the message. To limit the software footprint in a production environment the agent should not responsible to wait for process success or failure responses of the message from MII and should only wait for confirmation of message delivery to MII. As a result each agent was setup to call the transaction engine asynchronously. If the MII Application server is too busy to handle an inbound message then the reliability configuration of the agent determines what happens next. In this case the agents were configured they have a Reliability model of 100 retry attempts at 10 second intervals. As previously mentioned the Persistence model of the transaction does affect how well it performs and in a production environment this should be set ONERROR. The result is that the actual execution of the transaction is only stored in the MII Database tables if there's a processing error and will only appear in the MII Transaction Manager if this is the case.

These settings reduce the overhead involved in triggering a transaction and also limit the amount of information that is persisted in the database tables without loosing any messages or the ability to reprocess messages and track errors.

Related Content

SAP NetWeaver CE 7.1 SDN Homepage: https://www.sdn.sap.com/irj/sdn/nw-ce-general

SAP NetWeaver CE v7.1 Help Documentation and Release Notes:
http://help.sap.com/saphelp_nwce10/helpdata/en/46/65695eec515de4e10000000a1553f6/frameset.htm

SAP NetWeaver CE 7.1 Help Documentation for ICM Parameters:
http://help.sap.com/saphelp_nw70/helpdata/EN/95/1528d8ca4648869ec3ceafc975101c/content.htm

SAP NetWeaver CE 7.1 Help Documentation for ICM Parameter Suggestions:

http://help.sap.com/saphelp_nwce10/helpdata/en/56/2e453cabf4ef6fe10000000a114084/content.htm

SAP JVM Help Documentation:
http://help.sap.com/saphelp_nwce10/helpdata/en/47/dc90b4ef17452289f9128b8c2bbd77/frameset.htm

Sun Documentation on Performance Tuning Pointers:
http://java.sun.com/performance/reference/whitepapers/tuning.html

Sun Documentation on v1.5 garbage collection settings:
http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html

Sun Documentation on Memory Allocation: http://java.sun.com/docs/hotspot/ism.html

Sun Documentation on JVM Options: http://blogs.sun.com/watt/resource/jvm-options-list.html

Sun Documentation on the JVM Options: http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp

IBM Performance Tuning website for the JVM memory settings:
http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzamy/50/admin/prftunejmem.htm

Microsoft Reference for all of the Performance Monitor Counters:
http://technet.microsoft.com/en-us/library/cc776490(WS.10).aspx

Basic Overview of Windows Performance Monitor:
http://www.windowsnetworking.com/articles_tutorials/Windows-Server-2003-Performance-Tuning.html

Windows Server 2003 Paging Configuration:

http://stackoverflow.com/questions/39059/how-do-i-run-my-app-with-large-pages-in-windows

How to Browse, Test, Configure, and Consume and Enterprise Service in ECC:
https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/607557cd-37fc-2b10-9091-cb993c46b152

SAP MII v12.0 Installation and Configuration Guide:
https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/f09005e1-fd91-2b10-7c9c-d580a30ba59c

Relevant Notes

892486 - AS Java Reconnect Parameter Optimization

Copyright