Paper 338-2010

# Achieving High-Performance Results on z/OS with SAS® 9.2 Business Intelligence

David Fahey, SAS Institute Inc., Cary, NC

## ABSTRACT

This paper describes huge performance improvements achieved when the SAS® 9.2 Business Intelligence (BI) software runs on the z/OS platform. SAS on the z/OS system has constraints that have been surmounted, and features have been used to assist large applications.

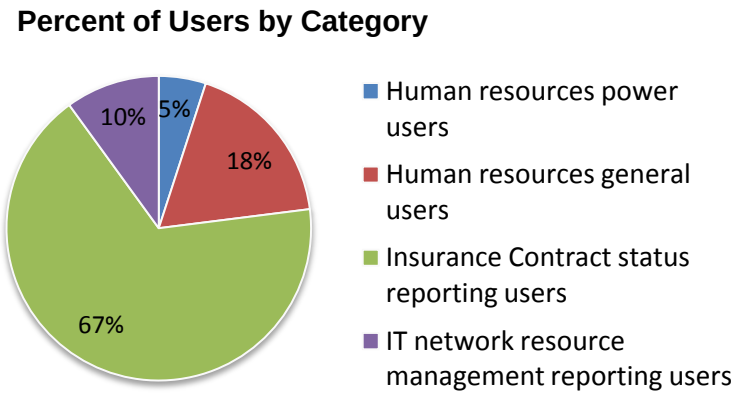
First of all, this paper describes our real-world test application (which was sourced from a customer who uses SAS® Web Report Studio and SAS® OLAP products) and also describes how various configurations were deployed on our benchmark z/OS and Windows servers. Secondly, the paper describes the monitoring tools we have used and some that have been created to monitor the benchmark systems. Finally and most importantly, this paper shows graphically the huge performance improvements that we have achieved so far and areas where we will extend our performance even further.

## INTRODUCTION

This paper discusses the black-box testing of a real customer SAS BI application, the methods used to test and measure the applications performance, improvements observed when compared to an earlier version of SAS, methods used to extend the performance, the results we have obtained so far, and directions we are and will be taking to investigate and improve performance in the future.
The source code for tools we have written to assist in monitoring our testing can be found at the following Web site
http://support.sas.com/rnd/emi/ZosRxmon92
This paper does not include techniques that measure the internal operation of the SAS BI system, nor that of the WebSphere middle-tier environment that was used. However, those topics are covered at this Web site
http://support.sas.com/rnd/emi

## THE TEST APPLICATION

The test application was supplied by a SAS customer who was planning to convert from SAS® 9.1.3 to SAS® 9.2. Their application uses SAS Web Report Studio in conjunction with SAS OLAP Server and supporting BI infrastructure in order to provide a very large group of users with a Web-based reporting system. The users and transactions included 15,000 SAS Web Report Studio users who were configured to run SAS OLAP transactions via 89 different SAS Web Report Studio reports. Four distinct categories of user were used, including: 75 Human resources power users, 270 Human resources general users, 1005 Insurance Contract status reporting users, and 150 IT network resource management reporting users.




The customer needed to make sure that the conversion to SAS 9.2 was going to provide their users with improved performance as well as the additional features they were looking for. They were able to provide a comprehensive list of the service levels expected. The customer's performance goals were as follows:

- Define15,000 users and execute with 15,000 users defined.
- Support 1,500 users concurrently logged on to SAS Web Report Studio (WRS).
- Support 150 active OLAP customer profile queries.
- Achieve expected overall response time of less than 10 seconds for Web-based queries.
- Achieve expected overall response time of less than 5 seconds for non-Web-based queries.
- CPU seconds per query used by SAS OLAP Server for average transaction should not exceed 0.6 CPU seconds on an IBM System z9.
- SAS WRS transaction response time should not exceed the following values:
    - General response time (80%) < 5 sec.
    - 10% 5 sec. < response time < 10 sec.
    - 8% 10 sec. < response < 20 sec.
    - 2% 20 sec. < response < 30 sec.
- Minimize I/O activity.

The test application structure has two parts. The simulated customer users and the customer's SAS BI applications. At SAS the performance testing of the customer SAS 9.2 BI applications was achieved using the HP Performance Center and LoadRunner tools. Earlier tests using a Java Test Tool were not able to support many more than 100 users without introducing distortions in the response time measurements. The HP tools were used to create and present online Web transactions to the customer SAS BI applications.

The customer SAS BI application systems were treated like a black box by the test tool, as shown in Figure 1 below, which shows the data flow between the major parts of the test tool. The HP Performance Center tool had to be configured to provide an adequate flow of transactions from 1,500 logged-on users. To do this required 15 Load Generator components, each capable of managing 100 user connections. The performance data logs and operating system logs were processed after the test by LoadRunner and SAS programs, then the test results were published as a Web page.

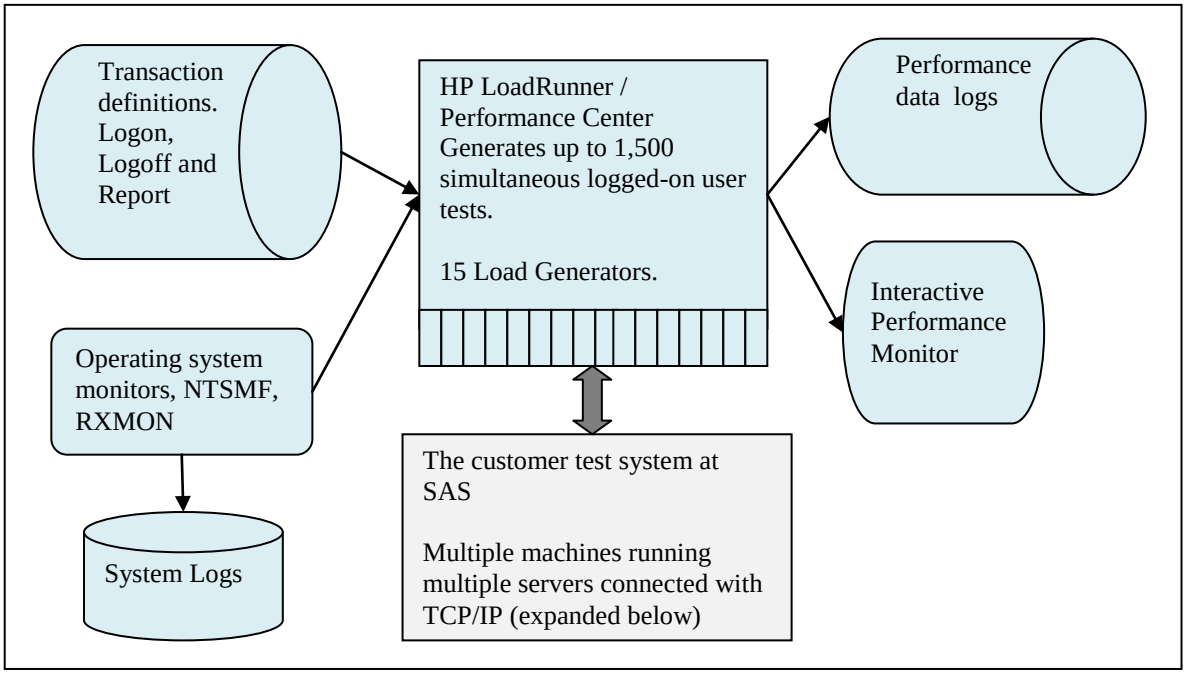



**Figure 1 Test tool and monitor configuration structure.**

Setting up the HP LoadRunner and Performance Center tools was not a trivial task. The setup involved creating LoadRunner scripts to run each of the 89 Web report studio transactions. These scripts are written in a variant of the C programming language that provided an interface to the tool. Scripts are used to define how report transactions are run and to interpret the report transaction results in order to determine if the report was successful. In all stages of our testing, having an early indication of transaction errors within the interactive performance monitor has been an

essential time saver. Our standard test runs for three and a half hours, so being able to kill it early, sort out the problem, and restart the test has been essential. Below is an example of the script language used. It is a very flexible environment, yet it is not a trivial job to become familiar with its capabilities. However, once the transactions are defined, running tests from the user interface of the Performance Center is a matter of a few clicks of the mouse.

```
Action()
  {
    int     i, l, status, TestWord_Count, ElementCnt ;
    char    *ptr, *lastptr ;
    char    ReportName[64] ;
    lr_think_time(60) ;
    ptr = lr_eval_string( "{Report_Name}" ) ;
    strcpy(ReportName,ptr) ;
    lr_message("%s user(%s): Report=\"%s\" Test=\"%s\"",User_Type,New_UserID,
      ReportName, lr_eval_string("{Report_TestWord}") ) ;
    lr_param_sprintf("Report_URL","%s%s.srx",lr_eval_string("{Report_Path}" ),
      lr_eval_string("{Report_Name}")) ;
    web_convert_param("Report_URL","SourceEncoding=PLAIN","TargetEncoding=URL",LAST) ;
    web_reg_find("Text=Begin Rendered Report",LAST) ;
    web_reg_save_param("ElementID","LB=wrsViewGetInitialReportElementContent('",
      "RB='","Ord=All","RelFrameId=1","Search=Body","IgnoreRedirections=Yes",LAST) ;
    lr_start_transaction("Report") ;
    lr_start_sub_transaction(ReportName,"Report") ;
    TestWord_Count = -1 ;             /* use -1 to indicate no test word */
    if (strlen(lr_eval_string("{Report_TestWord}")) > 0)
      TestWord_Count = 0 ;
    web_url("openRVUrl.do",
    "URL=http://{Target}/SASWebReportStudio/openRVUrl.do?rsRID={Report_URL}%28Report%29" ,
      "Resource=0","RecContentType=text/html","Mode=HTML",LAST) ;
    web_submit_data("processReportDrawStyle.do",
...
    status = lr_get_transaction_status("Report") ;
    lr_end_sub_transaction(ReportName,LR_AUTO) ;
    lr_end_transaction("Report", LR_AUTO) ;
    lr_message("%s user(%s): Report=\"%s\" Status=%s",User_Type, New_UserID,
      ReportName,status == LR_PASS ? "Succeeded" : "Failed" ) ;
    return 0 ;
  }
```

**Figure 2 Example HP Performance Center script (with some parts excluded).**

During testing, we emulated morning business activity. The number of users increased over a 50-minute period, reaching a constant level of use. Users then exited the system.

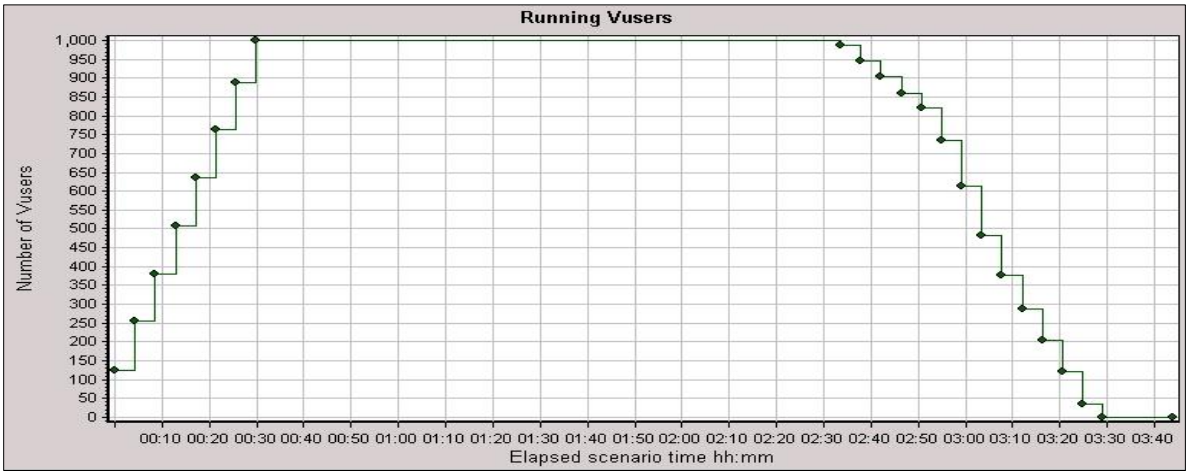



**Figure 3 Profile of virtual users allowed to run during a test.**

Users logged on to SAS Web Report Studio and consistently followed a cycle of report navigating, and launching and viewing reports for around 240 seconds. When these tasks were completed, users logged off the system. Once a user cycle was completed, a new user cycle was started. The total of 1000 users logged on was continued for one hour, and after that the numbers logged on was decreased in an orderly manner. The profile of logged on users can be seen in figure 3 above. A one hour period at the end of the test was allowed for test users to finish their work and log off the system, during that time no new users were logged on.

## TEST HARDWARE CONFIGURATIONS

Various configurations were deployed on our benchmark z/OS and Windows servers. Two configurations are described here.

At first we tested a single OLAP server system with the middle tier on Windows, and metadata and OLAP servers on z/OS. This baseline system enabled us to gauge how many OLAP servers might be needed for a given number of users. It also enabled us to compare our SAS 9.2 results with those obtained for SAS 9.1.3 and to judge the improvements gained by using some memory-tuning options created for BI servers on z/OS. The hardware used for this was as follows:

- 16 CPUs on two machines (including one System z9 with eight CPUs).
- 26 gigabytes (GBs) of RAM memory.
- Storage was an IBM System Storage DS8000.
- Network connections were all one GB/s ethernet.

The hardware organization and servers is described in figure 4 below

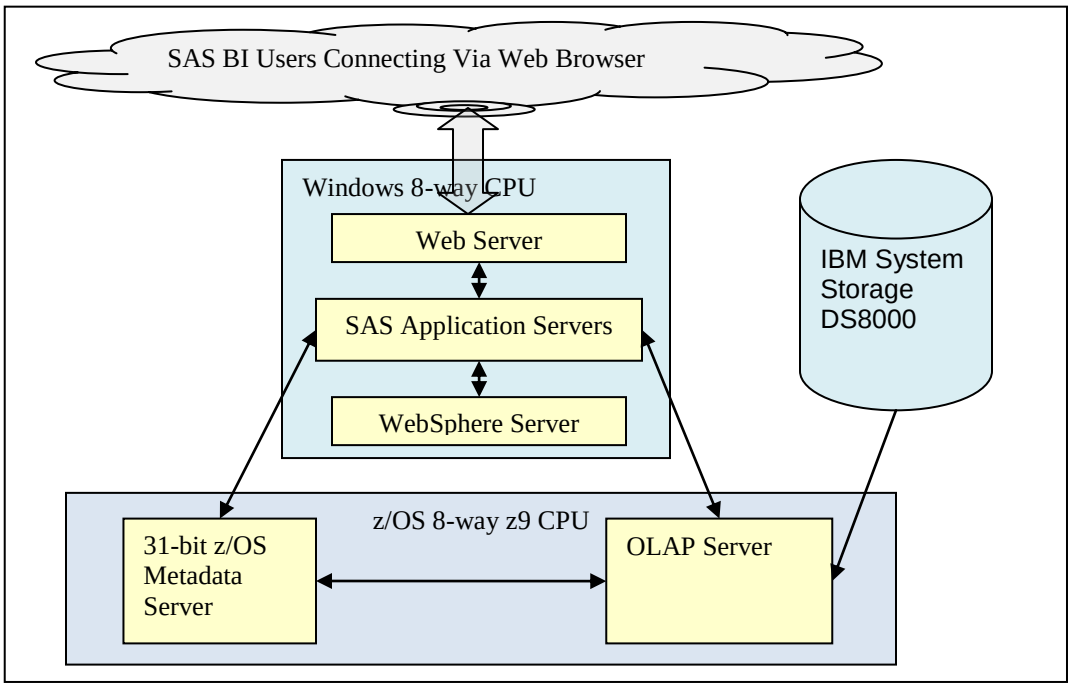



**Figure 4 Baseline test system with one OLAP server**

Secondly, we tested clustered OLAP servers and the Metadata Server on z/OS with clustered WebSphere middle-tier servers on Windows. The hardware used for this was:

- 32 CPUs on 4 machines (including 1 System z9 with eight CPUs).
- 96 gigabytes (GBs) of RAM memory.
- Storage was an IBM System Storage DS8000.
- Network connections were all one GB/s ethernet.

The hardware organization and servers is described in figure 5 below

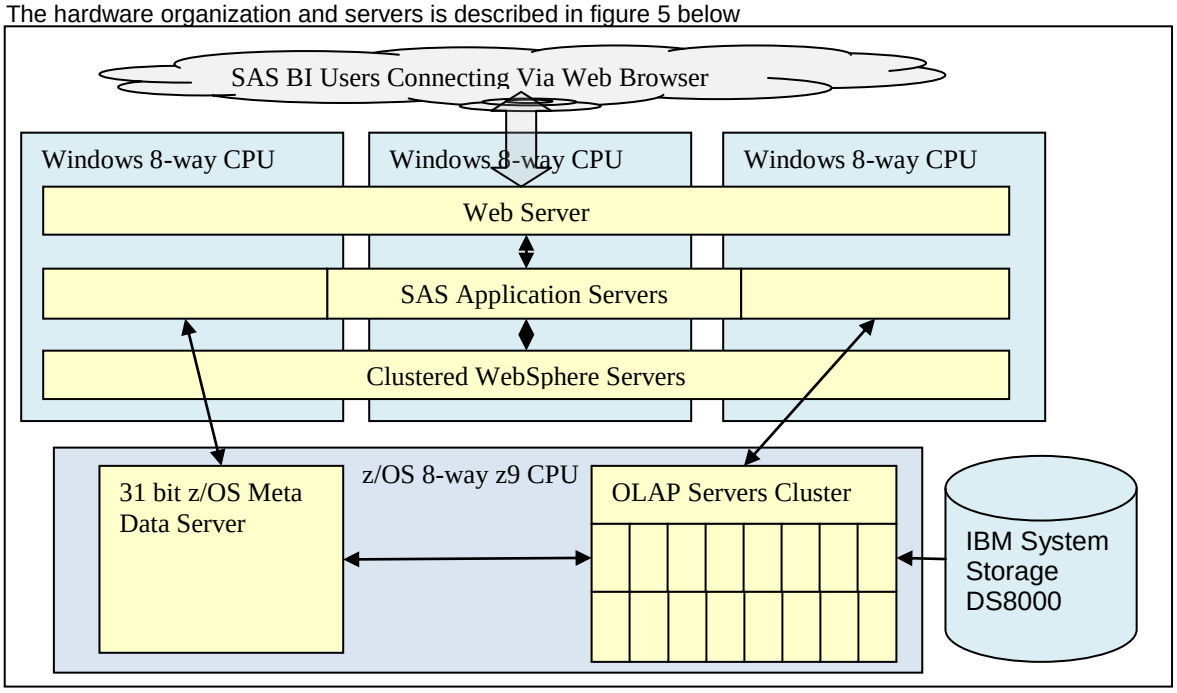



**Figure 5 Interconnection of servers with Meta Data server on 31 bit z/OS**

## MONITORING TOOLS

As well as the monitoring tools built into HP Performance Center some additional tools were built at SAS to capture information from z/OS during the test execution. These tools were required because the systems RMF tool was being used by system programmers for system-wide monitoring, and it had a monitoring interval that was too wide—15 minutes. We were interested in a much shorter interval of one minute or less. The tool we created called RXMON took only three days to create because it was created with the REXX scripting language in conjunction with the SDSF program and some handy user-written functions from the CBT tape. Figure 7 below shows how RXMON is used.

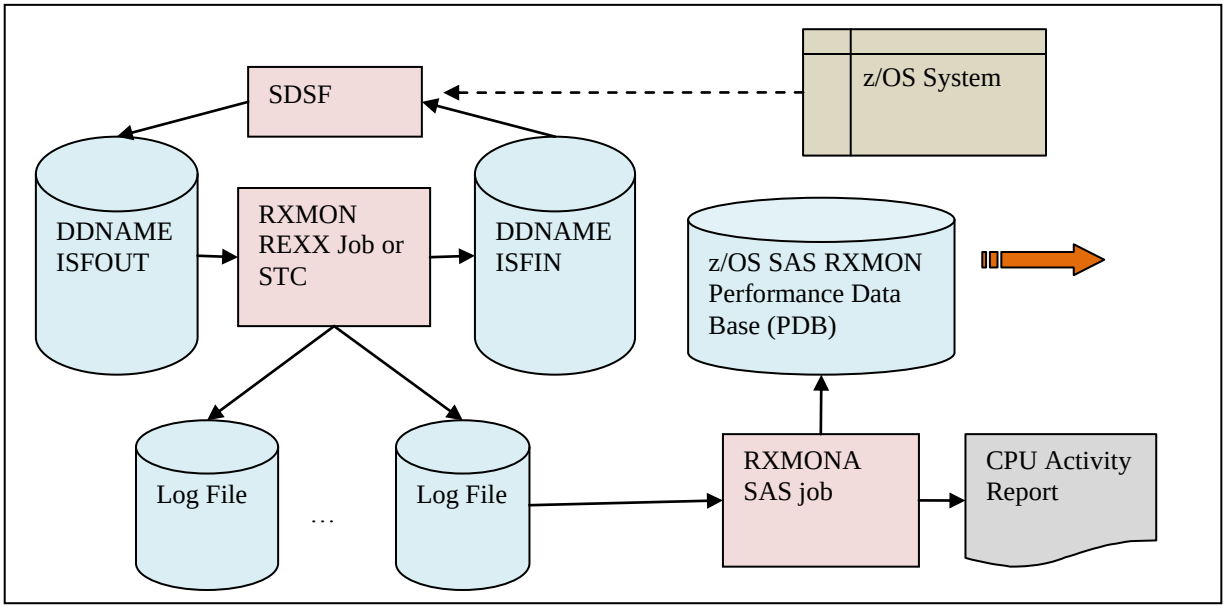



**Figure 6 RXMON – A REXX-based monitor with SAS based PDB creation and analysis**

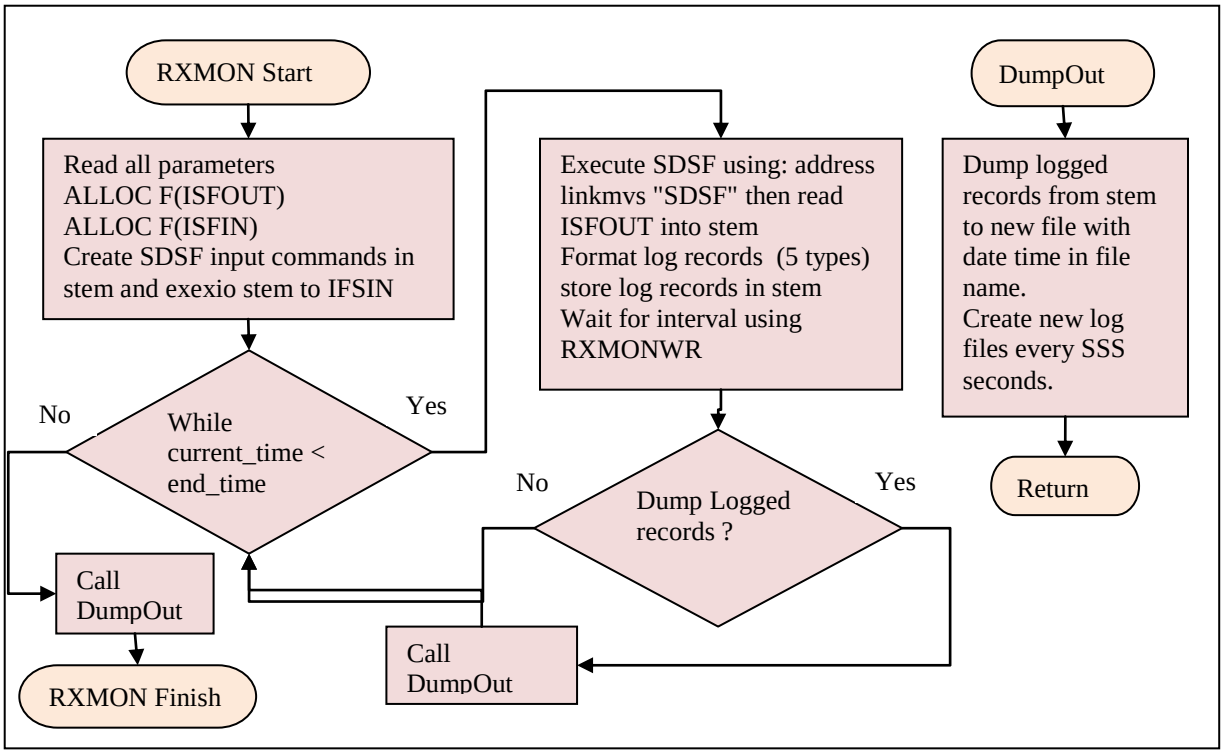



**Figure 7 RXMON program flow**

The RXMON program flow is shown in Figure 7 above, and you can see from this that the program is very simple. It leaves all the hard work of capturing the system information about the address spaces running on the system to the SDSF program supplied by IBM. We also assume that the SDSF program gets correct data from the system. The log files that are created are in a sequential data set with a name format such as: userid.RXMON.D0090221.T040610.LOG. You can see that the date and time are used in the log data set name to avoid collisions with existing data sets when the data set is created. The contents of the log data set are shown in Figure 8 below.

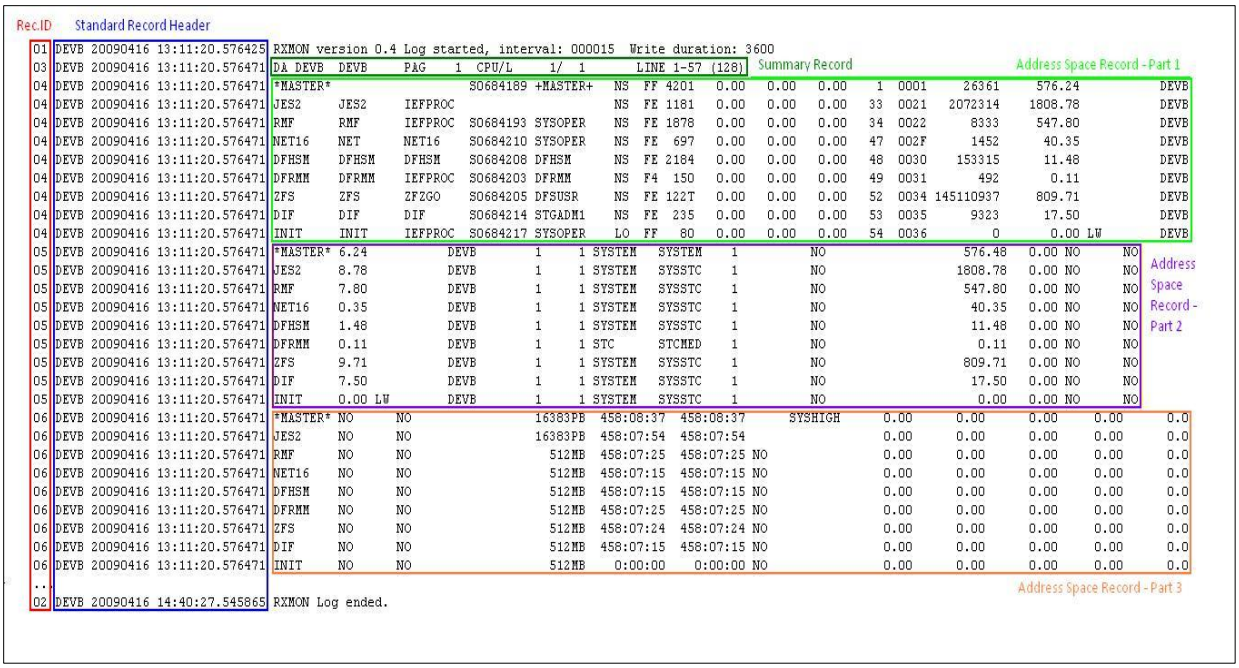



**Figure 8 Example of log records from RXMON**

The log records above shows only some of the fields and record types in the data. The first column highlighted in red is a record identifier, which is added to the SDSF data by the REXX program along with the system SMFID and a time stamp highlighted in blue. The log files for a test are concatenated together and processed into a performance database, and from there various reports are created. Following in Figure 9 is a CPU activity report for a test.

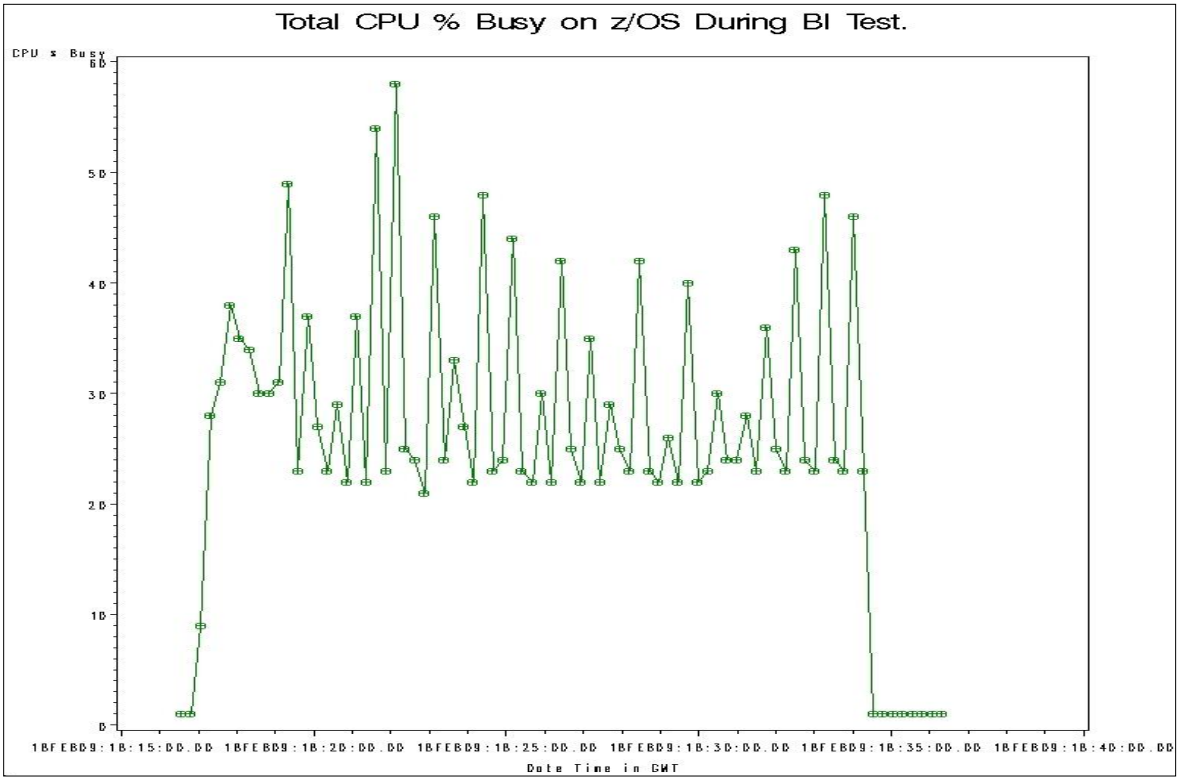



**Figure 9 RXMON data used for z/OS CPU activity report**

On Windows systems performance data was collected by the HP Performance Center tool and also by the NTSMF product. NTSMF simply logs Windows performance counters into comma-delimited text files. The contents of an NTSMF file are described in directory entries within the file. We wrote SAS code to read the file directory and generate additional SAS code in order to process the data into a performance database on a Windows system. NTSMF data can contain more information than data gathered by the HP Performance Center. In particular there is process-specific information that is useful when breaking apart the performance of components within the middle tier.

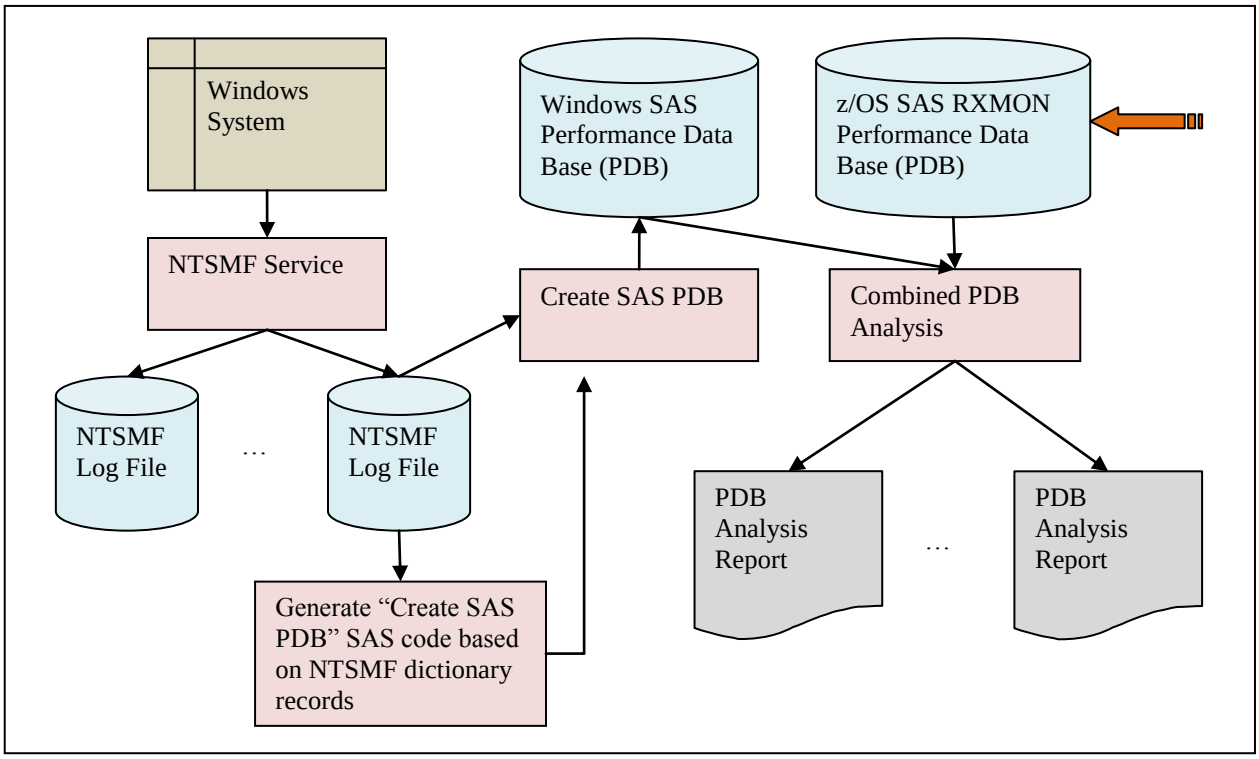



**Figure 10 NTSMF performance data gathering structure**

## DISCOVERIES AND SOLUTIONS

Our initial test structure (Figure 4) was created to determine if the new MDX query language extensions used by the OLAP server in SAS 9.2 would dramatically improve performance over the SAS 9.1.3 version. That testing uncovered two problems.

First of all, the way the OLAP server allocated work files to a native z/OS data set slowed down the OLAP transaction processing significantly. In particular it looked like only one of the eight CPUs was being used. After profiling the code it was determined that the dynamic allocation of the work file was being serialized, and because dynamic allocation uses a lot of time, all of the OLAP threads had to wait for each other to perform the work file allocation. This was happening to every OLAP transaction, which caused slow performance. The solution to this problem was to use zFS files instead of native z/OS data sets, as zFS files do not require allocation. To do this required only a simple modification of the WORK= parameter in the JCL procedure for the OLAP server, as seen in Figure 11 below.

Secondly, the tests appeared to be running short of memory, which showed up as an out of memory ABEND on the OLAP server. Even after the memory region for the OLAP server was increased to 1500 MB in the JCL procedure (see Figure 11), which is the maximum available, the OLAP server was able to handle only 64 users. After running a profile tool to examine the execution of the OLAP server we found that a huge amount of time was spent in the operating system memory allocation routines (GETMAIN). Memory allocation by the operating system is slow and CPU expensive, which is exacerbated by the high volume of small memory allocations. To fix this the memory management routines used by OLAP and other servers on z/OS had to be modified to keep and reuse memory allocations rather than free the memory back to the operating system after each use. These modifications were done

in a way that enabled us to turn them on and off via a SAS environment variable that was set in the configuration file for the OLAP server. See Figure 12 below for the format of the two options TKOPT_CPOOL3 and TKOPT_KEEPPOOL. As these options are keyword options they do not have parameters after the "=". We later applied these configuration options to the metadata server as well as the OLAP server; however, this does not always help the metadata server. Huge metadata repositories can cause problems. The use of these configuration options on the OLAP servers decreased both elapsed time and the amount of CPU time used.


**Figure 11 OLAP server STC procedure JCL**

```
*
* This file contains system environment variables for the
* OLAP server known by the started task name:
*      RDTCOLA1
*
reset
IMVS 'RDTC1.SAS92.TKMVSENV(TKMVSENV)'
IHFS '/u/rdtc1/ServerConfigC/Lev1/SASApp/tkmvsenv.cfg'
IHFS '/u/rdtc1/ServerConfigC/Lev1/SASApp/tkmvsenv_usermods.cfg'
set TKOPT_CPOOL3=
set TKOPT_KEEPPOOL=
* set TKOPT_OVFL_FIXED=2
* set TKOPT_OVFL_SMALL=2
* set TKOPT_OVFL_MEDIUM=2
* set TKOPT_OVFL_LARGE=2
set SASOTPCLOSE=
IHFS '/u/rdtc1/ServerConfigC/Lev1/SASApp/OLAPServer_LB/tkmvsenv_usermods.cfg'
set OLAPSERVER=Y
```

Insert memory performance optimization environment variables.

**Figure 12 OLAP server config file**

After improving the memory allocation speed, the next part of the memory problem to resolve was to increase the total amount of memory, so more users could execute OLAP transactions in parallel. To understand how this was achieved it is important to understand the structure of virtual memory on z/OS.

In Figure 13 several aspects of z/OS system storage are shown in a diagrammatic form.

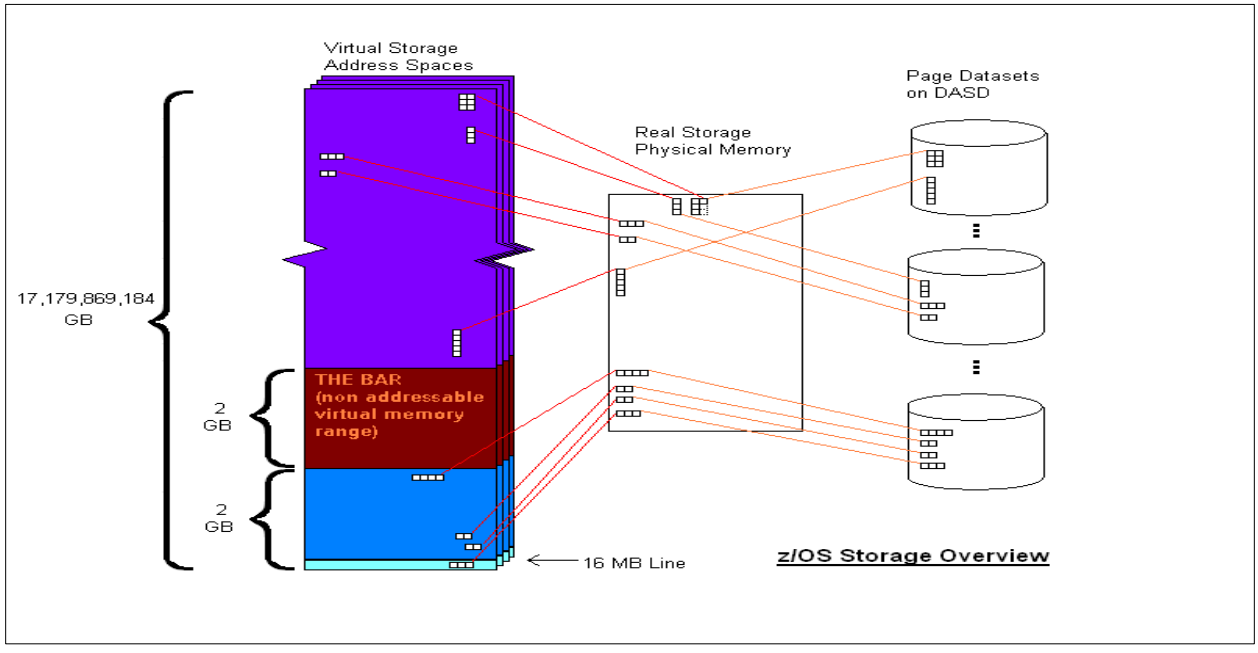



**Figure 13 z/OS system storage overview**

The colored part on the left represents address spaces, which are like processes on UNIX and Windows. An address space is a virtual storage representation of the memory tasks associated with that address space can access. Four thousand pages of virtual memory have real memory pages associated with them. As virtual memory use increases eventually more virtual memory is required than there is real memory available. At that time memory pages that are

not in current use are moved to a page data set. When a virtual memory page is referenced that is not in real storage it is moved back to real storage, possibly replacing another page that is not being used.
Programs running on z/OS can address 17,179,869,184 gigabytes of virtual memory if they are written to use 64-bit instructions. However, most z/OS programs do not use these instructions, and are limited to two gigabytes of virtual memory using 31-bit instructions.

Currently, the SAS 9.2 System on z/OS runs within a 31-bit address space. This has a significant performance impact on server applications that need memory for each user transaction that they service. On our system the servers were effectively limited to a region of 1,500 megabytes because the z/OS operating system reserves some space for its own operation.

The solution that would increase the amount of memory available to OLAP was to have more OLAP server address spaces working in parallel. This was achieved by configuring OLAP to use multiple load-balancing servers. Often this technique is used to have more machines clustered in order to increase the CPU cycles that OLAP can use, but here it was used to increase memory. After this change was made we were able to get five CPUs busy on z/OS compared to one, and the number of users was in the hundreds. We still had three CPUs idle, which was simply because our middle tier (WebSphere on Windows) was not able to supply enough work to keep the OLAP system on z/OS busy. To solve this we added more Windows hardware and configured a clustered middle tier. Once this was done we managed to get more than 1,000 users running, and in some tests as many as 1,400 users.

## RESULTING PERFORMANCE IMPROVEMENT ON Z/OS

The two graphs below summarize the elapsed time and CPU time decreases we achieved before moving to the load-balanced servers and the clustered middle tier.

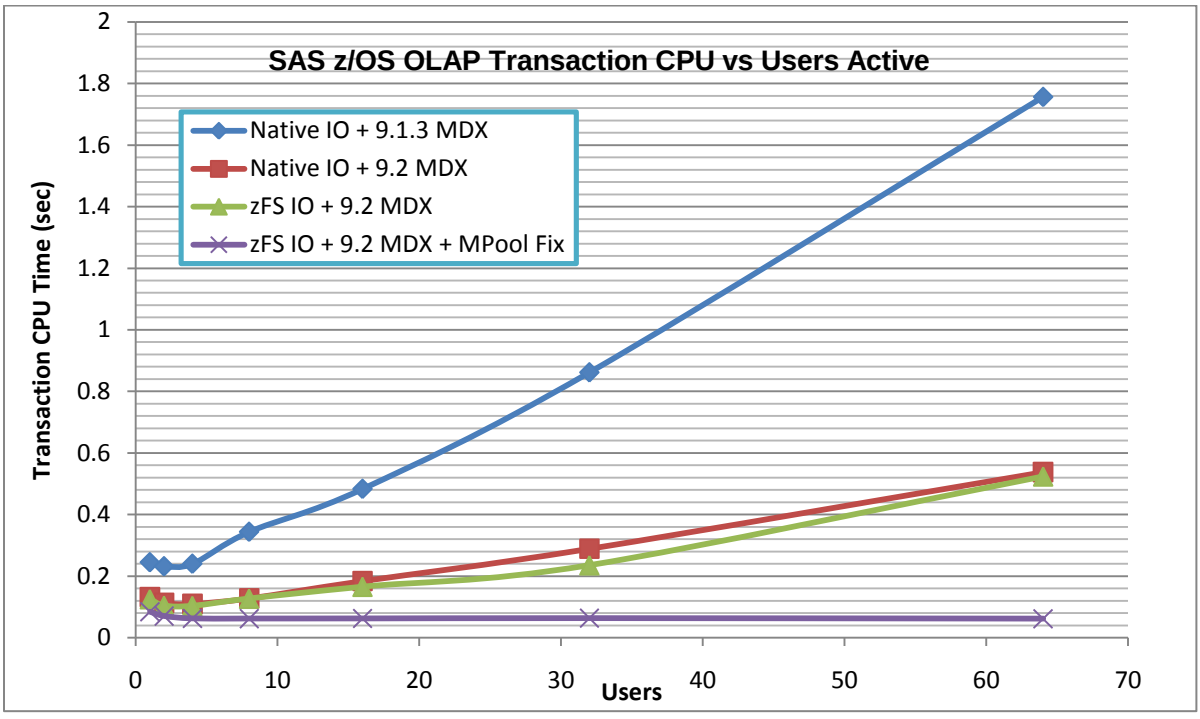



**Figure 14 Transaction CPU versus Number of Users Active**

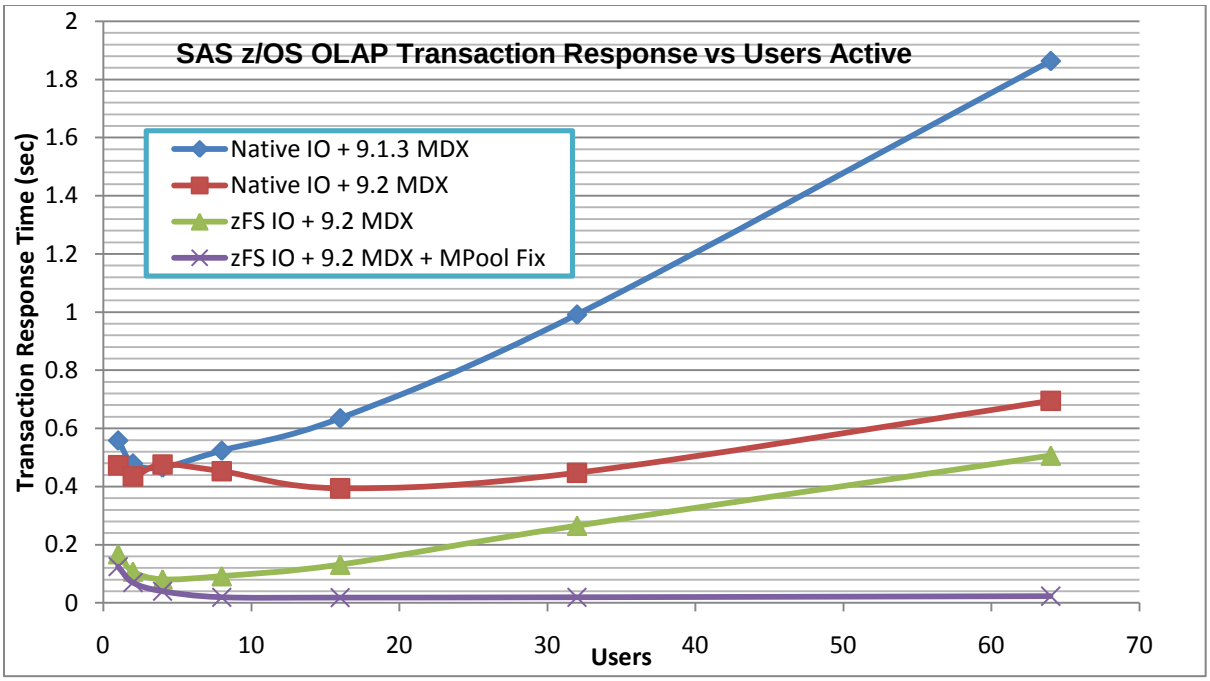



**Figure 6 Transaction Response Time versus Number of Users Active**

When we finally achieved between 1,000 and 1,400 users running it became much harder to find easy changes that we could make to reach the 1,500 active user level. With 1,400 users running, the system would not always remain stable and would suffer some ABENDs that were very hard to track. At this point we reported the very positive results achieved so far as summarized in the following table; however, we continued to search for the source of the instabilities.

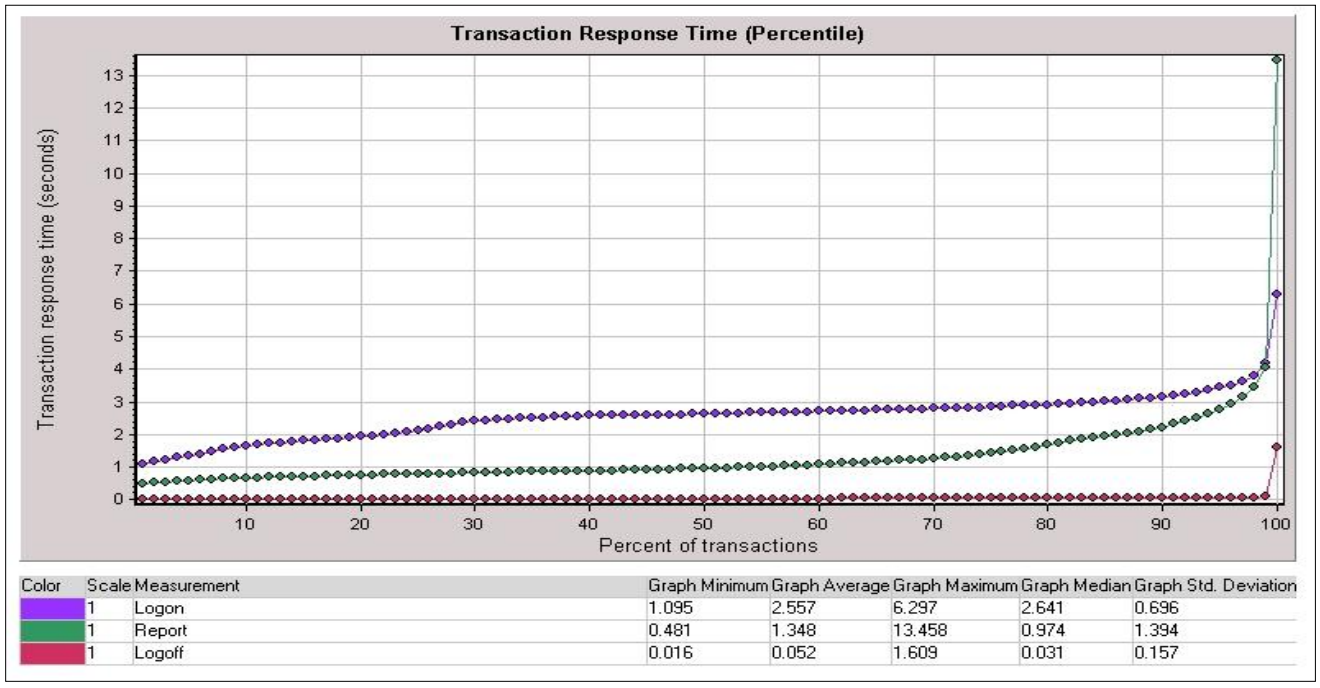
| Metric | Value |
|---|---|
| Commentary about findings | • Does not meet 1,500 active user criteria<br>• Meets 0.6 sec z9 CPU usage goal. |
| Total CPUs | 32 |
| Transactions | 45353 |
| Concurrent users | 1000 |
| OLAP Servers | 12 |
| User response time avg. | 1.203 sec. |
| User response time s.d. | 0.264 sec. |
| % Trans < 5 sec. | 99% < 5 sec. |
| OLAP CPU per transaction | 0.321 sec. |
| OMR CPU per transaction | 0.3589 sec. |
| Total OLAP CPU | 14597.37 sec. |
| Total OLAP Mem. | 13,891,321,856 bytes |
| Total OMR CPU | 16280.31 sec. |
| Total OMR Mem. | 1,150,976,000 bytes |

**Table 1: Comparison of results by metadata server**

Some graphs taken from HP Performance Center are shown in the following pages for the 1,000 user test. The detailed results are broken down into sub-sections: response time percentiles; response times during the test; CPU usage during the test; memory usage during the test. I/O and network activity were evaluated in earlier testing and were shown to have little impact on the overall performance. The one GB/s network connections were less than five percent busy. I/O was being satisfied from the cache very soon after the test started because only read activity was being used.

## RESPONSE TIME PERCENTILE

On the following graph the report line (green) is the most important as it reflects the response time for the reports the user ran via SAS Web Report Studio. The point where the five-second transaction response time intersects this line is used in the summary table above. This directly relates to a customer performance requirement that the general response time for 80% of the users must be less than five seconds.

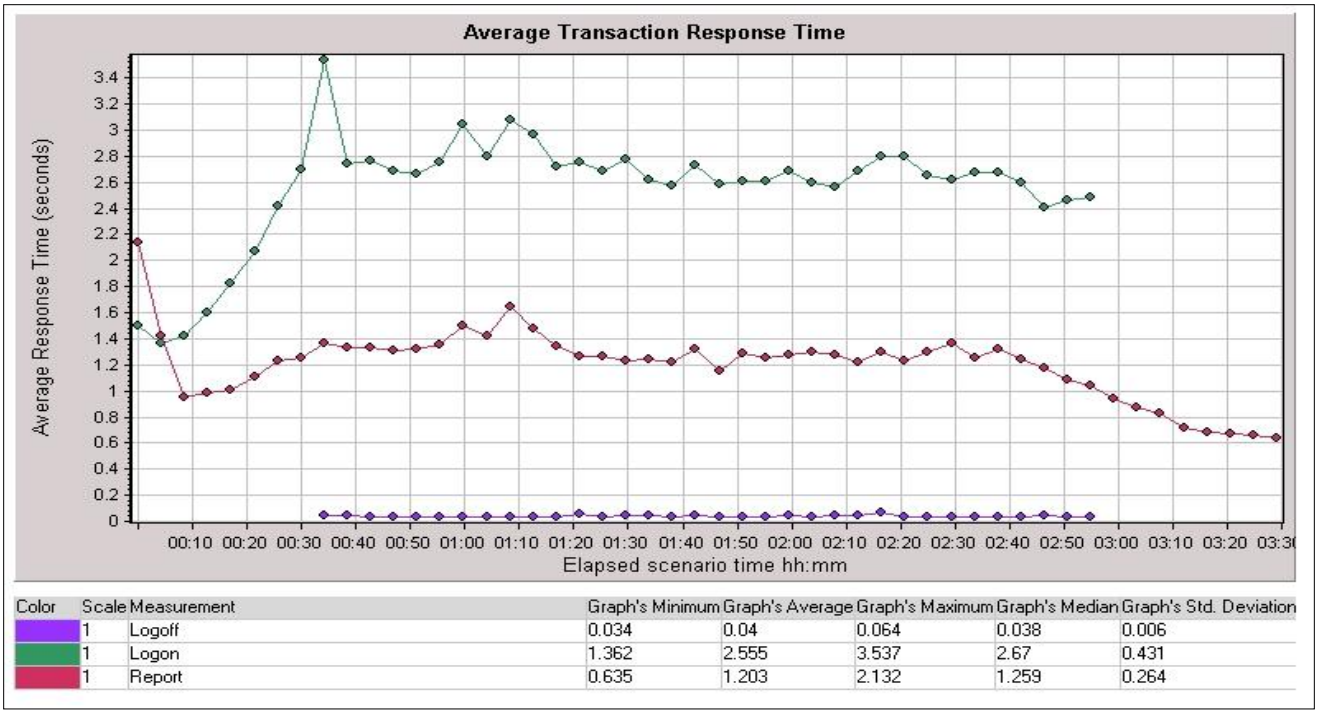



| Color | Scale | Measurement | Graph Minimum | Graph Average | Graph Maximum | Graph Median | Graph Std. Deviation |
|---|---|---|---|---|---|---|---|
| | 1 | Logon | 1.095 | 2.557 | 6.297 | 2.641 | 0.696 |
| | 1 | Report | 0.481 | 1.348 | 13.458 | 0.974 | 1.394 |
| | 1 | Logoff | 0.016 | 0.052 | 1.609 | 0.031 | 0.157 |

**Figure 76 Transaction response time percentiles for SAS 9.2 metadata server on z/OS configuration.**

## RESPONSE TIME

The following graph shows response times for the duration of the test. Again, the report line (red) is the most important as it reflects the response time for the report the user ran via SAS Web Report Studio.



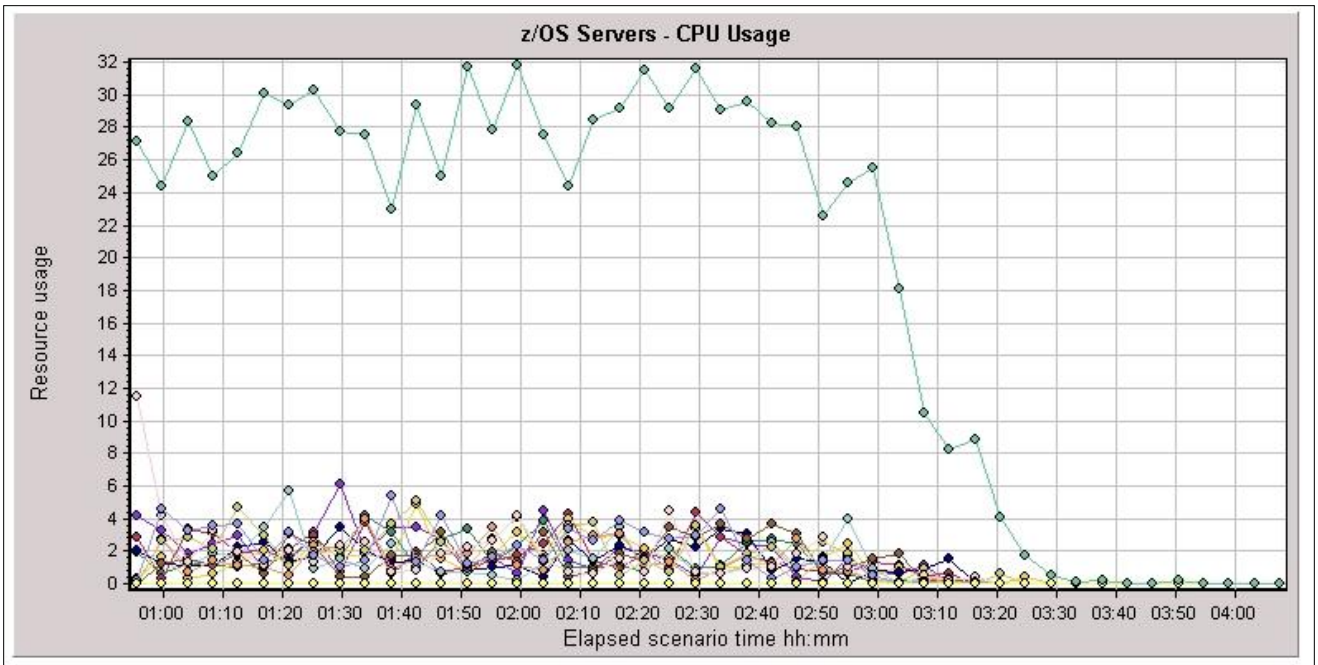

| Color | Scale | Measurement | Graph's Minimum | Graph's Average | Graph's Maximum | Graph's Median | Graph's Std. Deviation |
|---|---|---|---|---|---|---|---|
| | 1 | Logoff | 0.034 | 0.04 | 0.064 | 0.038 | 0.006 |
| | 1 | Logon | 1.362 | 2.555 | 3.537 | 2.67 | 0.431 |
| | 1 | Report | 0.635 | 1.203 | 2.132 | 1.259 | 0.264 |

**Figure 87 Transaction response time during test for SAS 9.2 metadata server on z/OS configuration.**

## CPU USAGE

The following graph shows the CPU usage for the OLAP clustered servers. Although the graphics look very busy, this is in fact an expected and good result. It indicates that load balancing among the 12 OLAP servers is being achieved. You cannot determine from the graphs which OLAP server is the master in the cluster. This is also very good, as it indicates the master is not a performance bottle neck, and the load-balancing overhead is probably low.

When examining the metadata server you will noticethat the metadata server CPU usage is ten times higher than that of the OLAP servers. This is not so disturbing if you realize that all 12 OLAP servers and the middle tier are using the metadata server.

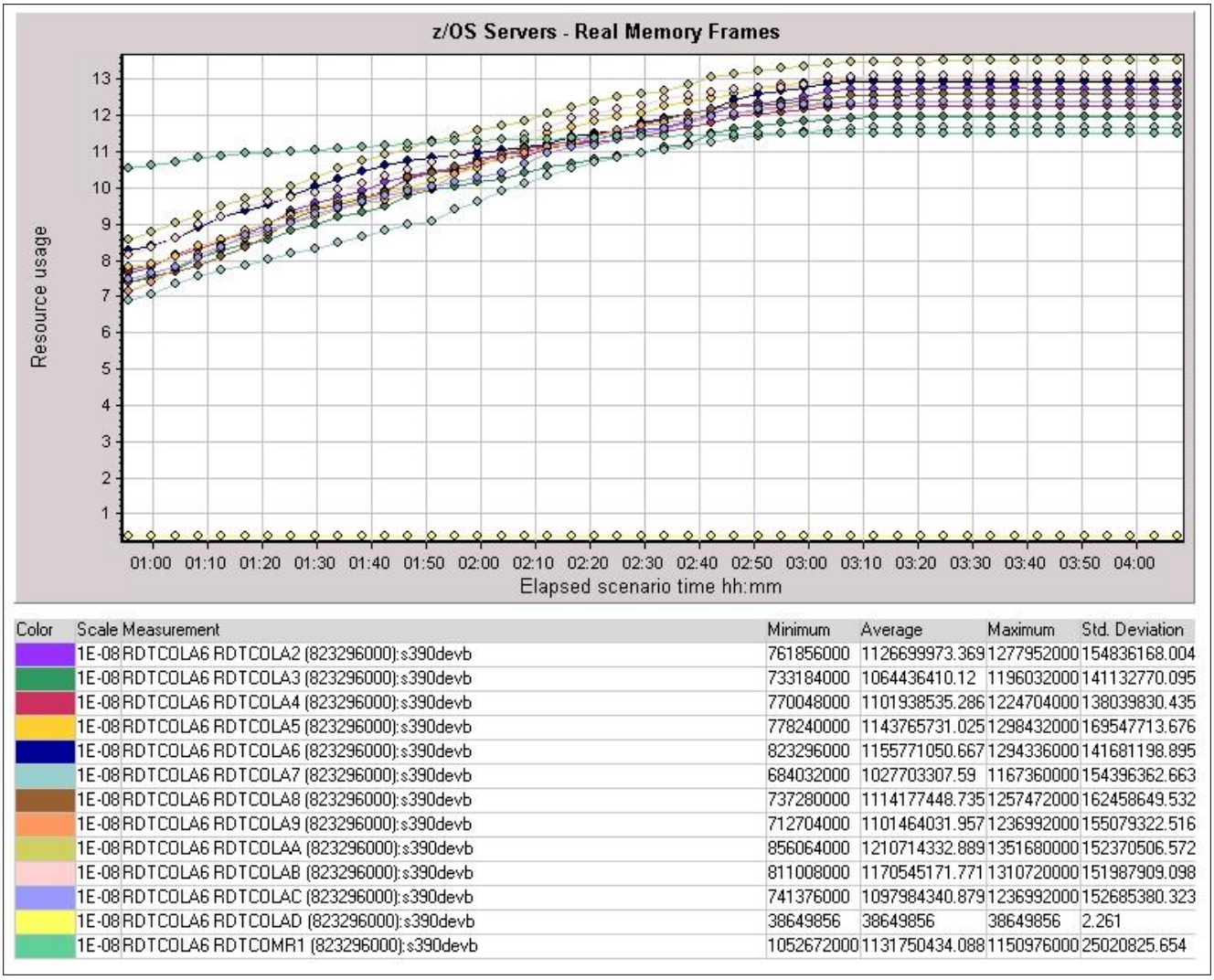



| Color | Scale | Measurement | Minimum | Average | Maximum | Std. Deviation |
|---|---|---|---|---|---|---|
| | 1 | RDTCOLA6 RDTCOLA2 (0.00):s390devb | 0 | 1.425 | 37.7 | 3.779 |
| | 1 | RDTCOLA6 RDTCOLA3 (0.00):s390devb | 0 | 1.152 | 39.15 | 3.996 |
| | 1 | RDTCOLA6 RDTCOLA4 (0.00):s390devb | 0 | 1.255 | 42.43 | 3.826 |
| | 1 | RDTCOLA6 RDTCOLA5 (0.00):s390devb | 0 | 1.236 | 32.47 | 3.977 |
| | 1 | RDTCOLA6 RDTCOLA6 (0.00):s390devb | 0 | 1.142 | 41.79 | 3.853 |
| | 1 | RDTCOLA6 RDTCOLA7 (0.00):s390devb | 0 | 1.108 | 49.19 | 4.115 |
| | 1 | RDTCOLA6 RDTCOLA8 (0.00):s390devb | 0 | 1.254 | 39.52 | 3.959 |
| | 1 | RDTCOLA6 RDTCOLA9 (0.00):s390devb | 0 | 1.077 | 44.1 | 3.339 |
| | 1 | RDTCOLA6 RDTCOLAA (0.00):s390devb | 0 | 1.533 | 47.41 | 4.667 |
| | 1 | RDTCOLA6 RDTCOLAB (0.00):s390devb | 0 | 1.245 | 54.27 | 4.195 |
| | 1 | RDTCOLA6 RDTCOLAC (0.00):s390devb | 0 | 1.589 | 52.12 | 4.951 |
| | 1 | RDTCOLA6 RDTCOLAD (0.00):s390devb | 0 | 0 | 0 | 0 |
| | 1 | RDTCOLA6 RDTCOMR1 (0.00):s390devb | 0 | 19.442 | 70.46 | 16.125 |

**Figure 18 CPU usage percentage during test for SAS 9.2 metadata server on z/OS configuration.**

## MEMORY USAGE

The memory usage graph shows an increase in the real memory used during the test. The servers are not shut down at the end of the test so memory assigned to the address spaces is retained until another address space steals it. As this test workload is the only workload on the system, the address spaces retain most of the memory allocated.




| Color | Scale | Measurement | Minimum | Average | Maximum | Std. Deviation |
|---|---|---|---|---|---|---|
| | 1E-08 | RDTCOLA6 RDTCOLA2 (823296000):s390devb | 761856000 | 1126699973.369 | 1277952000 | 154836168.004 |
| | 1E-08 | RDTCOLA6 RDTCOLA3 (823296000):s390devb | 733184000 | 1064436410.12 | 1196032000 | 141132770.095 |
| | 1E-08 | RDTCOLA6 RDTCOLA4 (823296000):s390devb | 770048000 | 1101938535.286 | 1224704000 | 138039830.435 |
| | 1E-08 | RDTCOLA6 RDTCOLA5 (823296000):s390devb | 778240000 | 1143765731.025 | 1298432000 | 169547713.676 |
| | 1E-08 | RDTCOLA6 RDTCOLA6 (823296000):s390devb | 823296000 | 1155771050.667 | 1294336000 | 141681198.895 |
| | 1E-08 | RDTCOLA6 RDTCOLA7 (823296000):s390devb | 684032000 | 1027703307.59 | 1167360000 | 154396362.663 |
| | 1E-08 | RDTCOLA6 RDTCOLA8 (823296000):s390devb | 737280000 | 1114177448.735 | 1257472000 | 162458649.532 |
| | 1E-08 | RDTCOLA6 RDTCOLA9 (823296000):s390devb | 712704000 | 1101464031.957 | 1236992000 | 155079322.516 |
| | 1E-08 | RDTCOLA6 RDTCOLAA (823296000):s390devb | 856064000 | 1210714332.889 | 1351680000 | 152370506.572 |
| | 1E-08 | RDTCOLA6 RDTCOLAB (823296000):s390devb | 811008000 | 1170545171.771 | 1310720000 | 151987909.098 |
| | 1E-08 | RDTCOLA6 RDTCOLAC (823296000):s390devb | 741376000 | 1097984340.879 | 1236992000 | 152685380.323 |
| | 1E-08 | RDTCOLA6 RDTCOLAD (823296000):s390devb | 38649856 | 38649856 | 38649856 | 2.261 |
| | 1E-08 | RDTCOMR1 (823296000):s390devb | 1052672000 | 1131750434.088 | 1150976000 | 25020825.654 |

**Figure 19 Real memory usage OLAP and metadata servers during test for SAS 9.2 metadata server on z/OS configuration.**

Sometime after reporting our performance improvements to the customer we uncovered two different causes of the stability problems. The first problem was caused by a failure in the memory manager's locking programming. This occurred only when the frequency of memory manager use by multiple threads was extremely high, which caused two CPUs to execute the same instruction sequence with only one instruction difference in their PSWs. The problem was fixed by changing the locking code to use the PLO instruction instead of the CDS instruction.

The second problem was caused by memory fragmentation. This is a situation caused when the entire virtual memory space becomes filled with an even spread of small memory allocations, and this causes a subsequent allocation of a large segment of memory to fail. Figure 20 below shows a diagrammatic view of memory fragmentation. In the case of the OLAP server it is suspected that the fragmentation arises after all of the virtual memory in the address space has been used intensively for a long time by two different memory usage patterns—specifically short-lived MDX query processing threads and the OLAP management thread, which is active as long as the OLAP server is active. More intensive study of the memory usage patterns will be needed to confirm that this is the exact cause. Nevertheless, a fix for the fragmentation has been made regardless of the exact cause. This involves retaining the large memory allocations for later reuse.

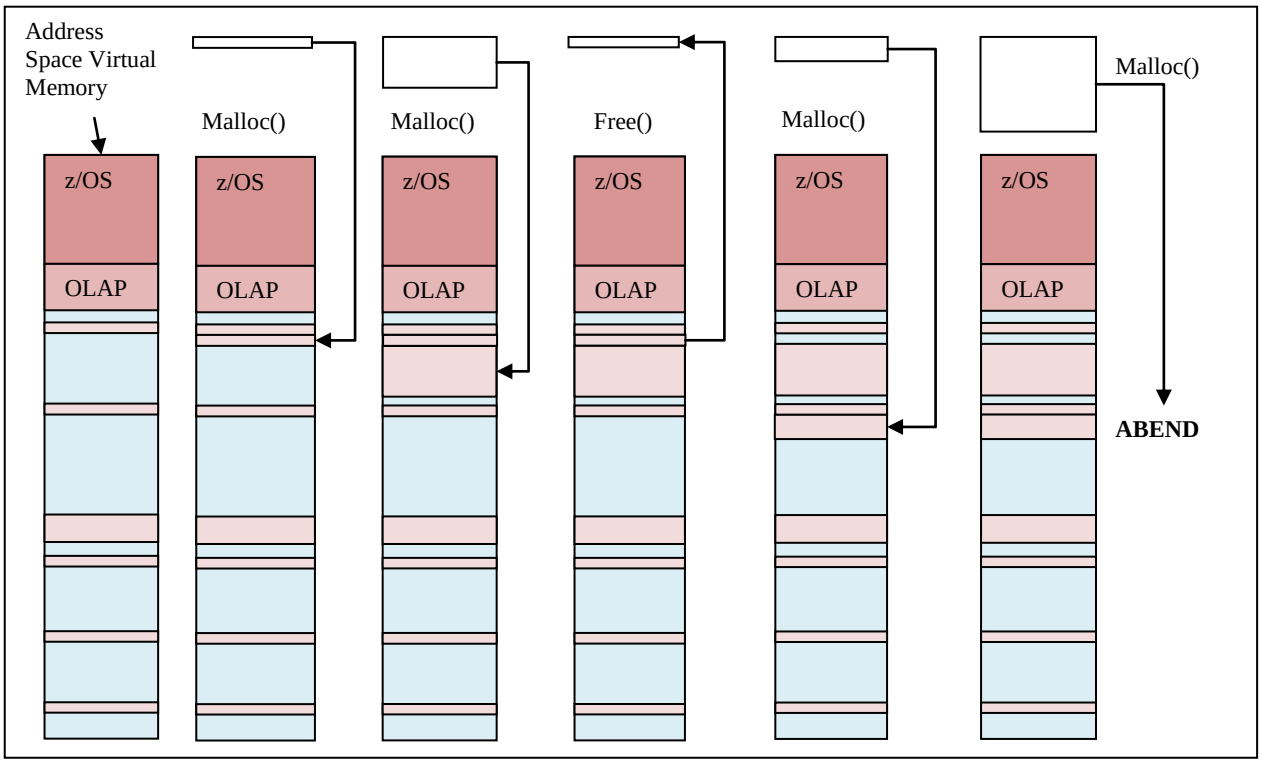



**Figure 20 Memory fragmentation**

## EXTENDING PERFORMANCE TESTING ON Z/OS

Members of the z/OS host department continue to test SAS BI systems on z/OS, and while we doing that we have uncovered areas where we need to extend and improve our product performance testing. Our RXMON tool is limited to capturing real memory use; however, we need to examine virtual memory use and SAS server memory use in more detail. To that end we have created a test monitoring tool that captures virtual memory use as well as other useful information. This tool is called MXPMON, which is based on the MXI tool by Rocket Software, written in the same way as RXMON using the REXX language. The overview of MXPMON operation is shown in the following diagram.

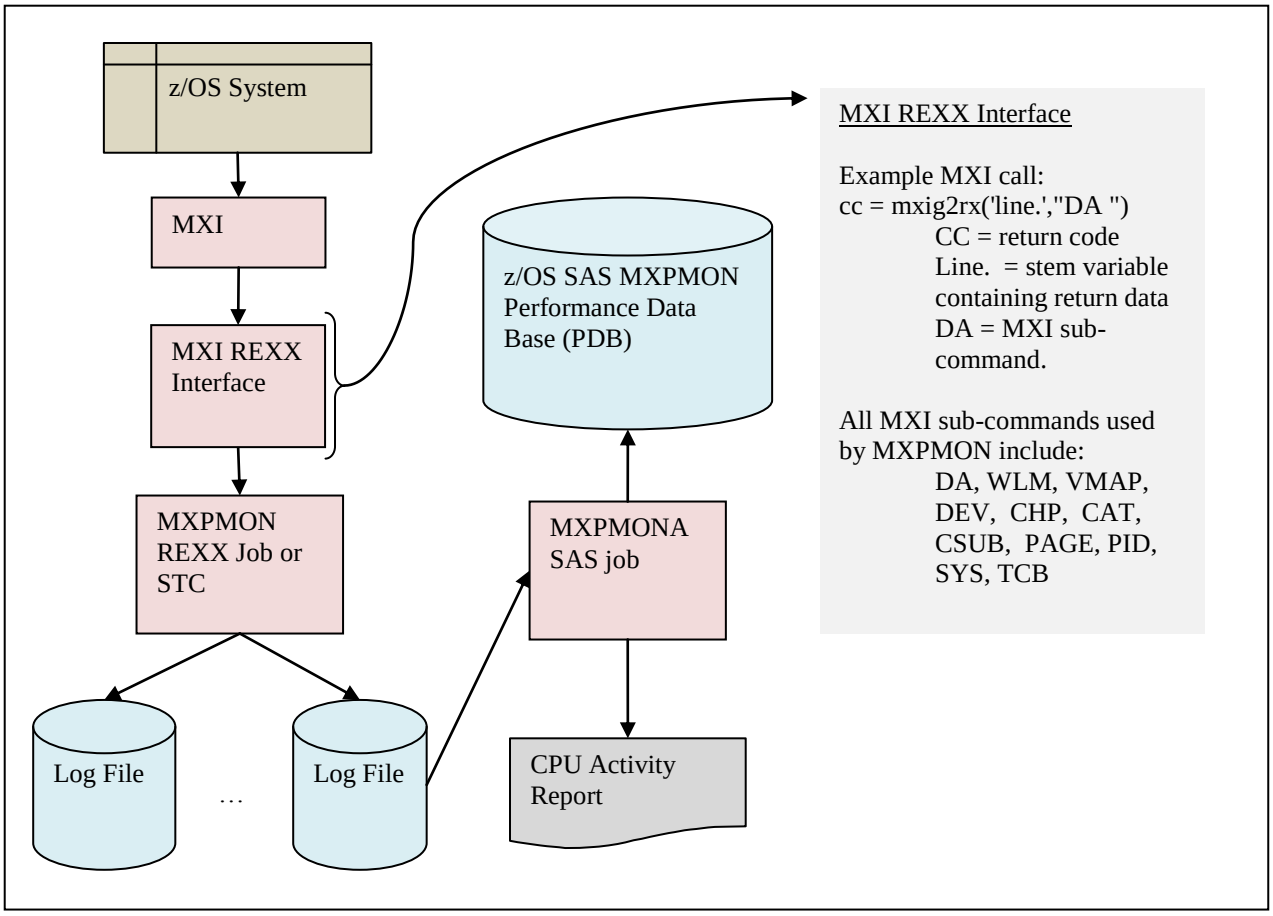



**Figure 21 MXPMON another REXX-based tool that uses the MXI tool.**

In future testing when we are using a middle Tier (WebSphere) on z/OS. We will be looking for tools to provide more data about our memory use within SAS servers and WebSphere servers as well as how that memory use relates to user transactions.

## EXTENDING PERFORMANCE OF SAS ON Z/OS FURTHER

Future areas where we will extend performance of the SAS System on z/OS even further are as follows:

- Improving memory manager algorithms to enable for faster memory reuse in long running applications.
- Supporting a z/OS metadata server in a 64-bit address space (as shown below in Figure 22).

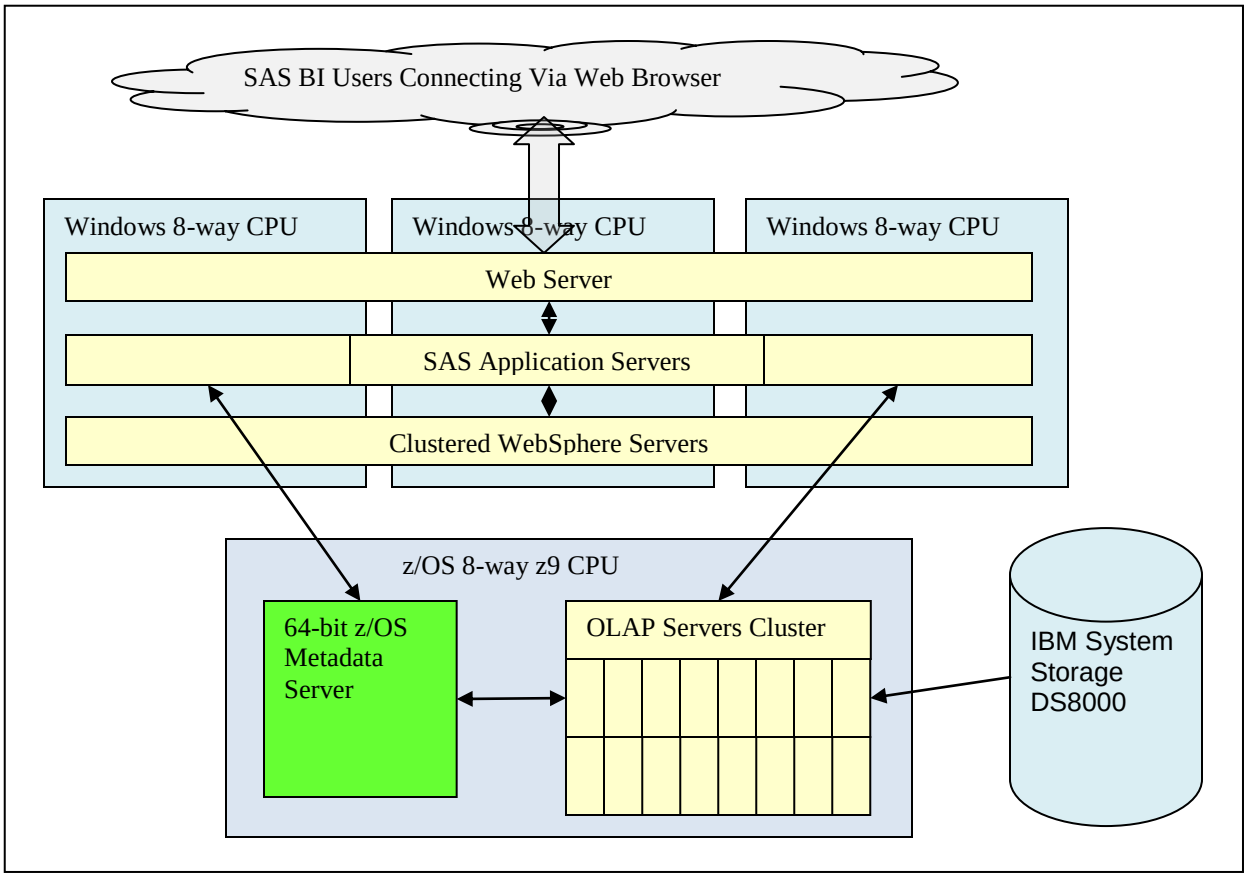



**Figure 22 Interconnection of servers with metadata server on 64-bit z/OS**

- Supporting the SAS Middle Tier on z/OS using WebSphere.

.

## CONCLUSION

We have been continuously testing performance on SAS BI for nearly three years with two or three people dedicated to the effort. This effort has been handsomely rewarded by the significant performance gains we have made, and by uncovering performance defects and fixing them before they caused problems for our customers. Some of the problems we detected could not have been found using any other technique.

While we improved SAS, we were simultaneously able to work with our valued customers and business partners, sharing the insight we have gained with them.

In many ways our testing just scraped the surface. We intend to continue testing various configurations of SAS BI in the future and will be able to directly apply our knowledge to testing 64-bit SAS BI on z/OS in the near future.

## REFERENCES


- IBM. 2006.. *z/OS TSO/E REXX Reference*, SA22-7790-07. 8th ed. IBM.
- IBM. 2008. *z/OS SDSF Operation and Customization*, SA22-7670-11. 10th ed. IBM.
- Hewlett-Packard. 2009. *HP Performance Center User Guide.* Hewlett-Packard.
- Rocket Software. 2009. *MXI Generation II: Users Guide, Version 5 Release 2*, MXI-0520-UG-07. Rocket Software Inc.

## ACKNOWLEDGMENTS

The data presented in this report was gathered analyzed and presented by a group of people—all of whom have contributed to our ongoing evaluation of SAS BI applications on z/OS

- David Fahey – Monitoring, analysis, and presentation.
- Dale Ingold – Testing, monitoring, and analysis.
- Ronnie Coggins – Configuration, setup, and testing.
- Dave Crow – Direction, coordination.
- Others have also assisted, including Craig Rubendall, Heino Rust, Mathias Ender, Saravana Chandran, Jae Cody, David Dement, Kevin Bestelmeyer, Don Poitras, Alan Beale, Susan Burton, Janice Hunnings, Dan Squillace, Keefe Hayes, Dan Taylor.
- Our business partner IBM

## RECOMMENDED READING

This Web site http://support.sas.com/rnd/emi describes techniques that measure the internal operation of the SAS BI system, and that of the WebSphere middle-tier environment.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Fahey
SAS Institute Inc.
1 Campus Drive
Cary, NC, 27513

ph: 919 677 8000
Fax: 919 677 8123
E-mail: david.fahey@sas.com
Web: www.sas.com