Abstract

PowerVM® provides the industrial-strength virtualization technologies for IBM Power Systems™ servers and blades. For this project several PowerVM features such as Active Memory™ Expansion (AME), Active Memory Sharing (AMS), and virtualization are explored in combination with Siebel CRM 8.1. Using simulations of thousands of users, we will demonstrate how a company using Siebel Customer Relationship Management (CRM) can capitalize on AIX® and PowerVM features. This paper describes the journey and findings.
# Table of contents

**Introduction** ............................................................................................................................... 1  
Prerequisites ................................................................................................................................. 1  

**PowerVM technologies** .............................................................................................................. 1  

**Oracle Siebel PSPP kit** .............................................................................................................. 2  
PSPP software components ........................................................................................................... 2  
POWER7 hardware components ....................................................................................................... 2  
Test environment infrastructure ....................................................................................................... 3  
Iterative process ............................................................................................................................ 4  
Dedicated and virtual environments ............................................................................................... 5  
Dedicated machine LPAR layout .................................................................................................... 6  
Virtualized machine LPAR layout .................................................................................................. 6  
Oracle Database with Easy Tier ..................................................................................................... 7  
PowerHA layout ............................................................................................................................. 9  

**The cost of virtualization** .......................................................................................................... 10  

**Active Memory Expansion** ....................................................................................................... 11  
Requirements ............................................................................................................................... 11  
Activating AME ............................................................................................................................ 11  
Setting AME expansion factor ....................................................................................................... 12  
  Determining the expansion factor ............................................................................................. 13  
Testing AME on the Siebel application tier .................................................................................. 14  
  Run20 – Collecting AME recommendations ............................................................................ 14  
  Run22 – testing 1.19 AME expansion factor ........................................................................... 16  
  Run29 – testing 1.45 AME expansion factor ........................................................................... 18  
  Analyzing AME performance ................................................................................................. 19  
Testing AME on the Siebel database tier ...................................................................................... 20  
  Run89 – Collecting AME recommendations ........................................................................... 20  
  Run92 – Testing AME 1.4 expansion factor ........................................................................... 21  
  Best AME practices ............................................................................................................... 23  

**Active Memory Sharing** .......................................................................................................... 24  
AMS requirements ......................................................................................................................... 24  
AMS Activation ............................................................................................................................. 24  
  AMS Paging devices .................................................................................................................. 25  
  AMS shared memory pool ......................................................................................................... 25  
  LPAR shared memory enabling .............................................................................................. 26  
  Siebel with AMS ..................................................................................................................... 27  
  Best AMS practices .............................................................................................................. 29  

**Simultaneous multithreading** .................................................................................................... 30  
Modifying the SMT value on-the-fly ............................................................................................ 30  
SMT best practices ....................................................................................................................... 32  

**Conclusion** .................................................................................................................................. 32  

**Resources** .................................................................................................................................. 33
Introduction

In the first quarter of 2012, IBM and Oracle began a joint project to test various features of PowerVM and AIX specifically as they might be used in a Siebel CRM application environment. In order to execute the required testing the team assembled a set of systems, consisting of IBM POWER7® processor-based Power® 740 Express and Power 750 Express servers running AIX 6.1 along with an IBM Storwize® v7000 SAN storage server. The Power 740 servers were set up with dedicated LPARs (logical partitions) while the Power 750 servers used virtualized LPARs. The purpose of the two environments is to compare the ability of Siebel CRM to capitalize on features of PowerVM. By comparing a known Siebel CRM 8.1 workload on the dedicated environment with the same workload running in a virtualized environment, the costs and technical differences between the two environments can be discerned.

This paper demonstrates the feasibility and viability of PowerVM technologies with the Siebel CRM product suite. PowerVM technologies with Siebel CRM is proven robust and effective for IBM and Oracle customers.

Prerequisites

It is assumed that the reader:

- Is familiar with Oracle Database and Siebel CRM software features and concepts
- Has an understanding of the IBM AIX 6.1 operating system
- Is aware of IBM POWER7 features and server offerings
- Is familiar with the Siebel Platform Sizing and Performance Program (PSPP) Kit

PowerVM technologies

IBM PowerVM virtualization technology is a combination of hardware and software that supports and manages the virtual environments on POWER5™, POWER5+™, POWER6®, and POWER7 processor-based systems.

Available on IBM Power Systems, and IBM BladeCenter® servers as optional Editions, and supported by the AIX, IBM i, and Linux® operating systems, this set of comprehensive systems technologies and services is designed to enable you to aggregate and manage resources using a consolidated, logical view. While many features are available with PowerVM, in this paper, the listed features are combined to deliver proof points with Siebel CRM 8.1 on AIX 6.1.

The PowerVM technologies include:

- Virtual Ethernet for all networking usage during the Siebel CRM load simulations.
- Shared Ethernet Adapter for Siebel Web server and Gateway server LPARs to reduce network resources.
- Disk storage virtualization for Siebel CRM enterprise LPARs. Virtual SCSI provides the database and other Siebel CRM tiers with storage. Each Siebel CRM LPAR has its own disk resources attached to a SAN. No local storage is available.
- IBM Micro-Partitioning® technology using uncapped resources for impulsive Siebel CRM demands.
- Multiple shared-processor pools between Siebel Application and Web tiers.
- IBM Active Memory Sharing between Siebel CRM LPARs.
Active Memory Expansion reduces memory usage on the Siebel Application Server.

Shared memory pools required for AMS where loaning of memory can save resources.

Additionally, the paper includes technologies like:

- SMT - Simultaneous Multi-threading of 1, 2 or 4 threads per CPU. We will identify best performance for Siebel Application Servers.
- Storwize V7000 includes IBM System Storage® Easy Tier™, a function that responds to the presence of solid-state drives (SSDs) in a storage pool that also contains hard disk drives (HDDs).
- PowerHA® for the Siebel Gateway Server and shared file system of Siebel CRM.

The main focus of this paper is not to measure and document performance of the features tested. However, in some cases performance metrics are provided when they were measured. The goal of these tests is to demonstrate combined technologies for the benefit of Oracle’s Siebel CRM and IBM customers.

**Oracle Siebel PSPP kit**

The Oracle Siebel Platform Sizing and Performance Program (PSPP) kit consists of Siebel CRM v8.1.1.4 Industry Applications and Oracle Database 11g Release 2 software. The workload simulates deployments of a large enterprise using Siebel CRM software. The kit comes with a well stocked 115GB database, custom scripts to drive a load and special business rules involving employees, partners, and customers using Call Center and communications components.

The mix of workload simulation and scripts used in this Optimize Solution testing are NOT comparable with other published Siebel CRM PSPP benchmarks. The kit framework was adjusted to simulate a vanilla Siebel CRM environment in order to emulate the look and feel of what a client might experience out-of-the-box. Any target number of users, for example 2000 users, are split among the five simulation scripts. The generated load used the same ratio of scripts for scalability and to ensure consistency for comparing the system loads across runs. The PSPP test kit publication discusses in detail the test scripts and contents of the test kit.

**PSPP software components**

The core software components used to complete the PSPP benchmark were as follows:

- Siebel CRM Release 8.1.1 (8.1.1.4+ FP) Industry Applications
- Oracle Database 11g Release 2
- IBM AIX 6.1

**POWER7 hardware components**

POWER7 processor technology includes many design features that contribute to the leadership performance of the Power 740 server and the Power 750 server. High processor to memory and I/O subsystem bandwidths provide faster movement of data throughout the system. The hardware topology used in our PSPP tests is shown in Figure 1.
Test environment infrastructure

Figure 1 shows the basic system hardware lay-out used for the PSPP kit. The design of the layout focused on providing step-by-step additions of functionality, starting with a set of Power 740 servers with dedicated LPARs and moving towards a set of fully virtualized Power 750 servers.

Oracle Application Test Suite (OATS) controllers drive the web client simulation. Two IBM System x3850 X5 workstations drove loads from 1000 to 4000 users. OATS controllers delivered HTML requests to a web server, which, depending on goals, was collocated with the database server or was a stand-alone web server. IBM HTTP Server version 6.1 accepted the HTTP request and generated requests to the Siebel Application Server. The web server and the application server could be housed on either the Power 740 or Power 750, depending on whether the dedicated or virtualized environment was desired. In turn, the application server requested data from the database server. The dedicated environment had its own database server and disks distinct from the virtualized environment.
Iterative process

Every run starts at the same controlled state, giving repeatability of results. Between runs, statistics are collected, database updates occur and log files are written. Each time a parameter is changed, another run is typically done to identify the effects of the parameter change. Recording and setup up of the load simulations works best as an iterative process.

The iterative process involved these steps:
1. Restoring the environment. This was done by restoring the database, rebooting all LPARs, removing all previous logs and synchronizing machine clocks.

2. Readying the systems for load. The Siebel tiers are then brought up, starting with the database, Siebel Gateway Server and Web server tiers. The Siebel Application Server tier software is brought online and validated. Finally, on the Oracle Application Test Suite (OATS) windows controller, an internet explorer connects and a single user login and logoff success indicates readiness.

3. Driving a simulated load and collecting statistics. The targeted number of Siebel CRM simulated users are slowly ramped up until all users have completed an iteration of their tasks. During the entire process statistics are being collected. While in the steady state environment, key tests and procedures are performed, for example collecting the AIX amepat command data for Active Memory Expansion.

4. System review and archiving. Once the simulation has completed, the data is reviewed and archived. Results are collated as required.

5. The process repeats itself for each new run. Measurements are taken on all LPARs and those statistics, results, and findings drive the next run.

### Dedicated and virtual environments

For our testing we established two available environments for testing functionality, a dedicated environment and a virtual environment. The term dedicated used throughout this paper refers to fixed CPU and memory resources. In a dedicated LPAR, the CPUs and memory cannot be shared or loaned to another LPAR. Dedicated resources are completely available to only one LPAR.

An LPAR can encompass dedicated or virtualized components within it. For example, in the dedicated environment, there are no virtualized disks, or Ethernet components. By contrast, within the virtualized environment the local drives and network cards are not used, virtualized devices, managed by the virtual I/O servers (VIOS) make up the disk and network resources. It is also possible to have a dedicated LPAR with virtualized components, where none of the virtualized components are shared.
The dedicated test environment, seen in Figure 2, has two defined LPARs, one on each Power 740 machine. The dedicated environment is well proven, with many data runs captured during a previous project. Next we discuss the virtualized environment. Having two sets of environments, a dedicated and a virtual, running side-by-side allows piecemeal isolation of any single LPAR and technology feature.

Virtualized machine LPAR layout

The virtualized test environment resides on two identical Power 750 servers. The virtualized environment is specifically configured for testing the PowerVM features and functionality. All LPARs within the virtual environment are fully virtualized, meaning that disks used virtual SCSI and networking used virtual Ethernet. In addition, all LPARs met the requirements for Live Partition Mobility. Note that testing Live Partition Mobility is beyond the scope of this exercise, but we hope to test it in a future project.

VIO servers are required for virtualization. Redundant VIO servers were considered, but the installed hardware lacked the necessary resources for creating additional VIO servers. A Power 750 server can fully support the complete Siebel CRM tier set residing upon a single server if desired.

A Siebel CRM enterprise deployment consists of four functional tiers. The tiers are the database server, Siebel Gateway Server, Web server, Application server. Siebel CRM best practices require the Siebel Gateway Server tier to be highly available and this is accomplished by using PowerHA. There are two Siebel Gateway Server LPARs, one within each Power 750 server, for PowerHA requirements. For simplicity, there is one web and one application LPAR. Customers may typically have multiple web and application LPARs for high availability and load balancing. The database tier is a dedicated virtualized
LPAR, such that virtualized resources were guaranteed to be available to the Oracle Database, and those same resources are not shared with other LPARs.

Figure 3 below shows the Siebel CRM enterprise tiers and the ability to migrate partitions to the second Power 750 server.

Each of the Siebel CRM enterprise tiers has their own unique characteristics as it applies the PSPP kit:

- The Siebel Gateway Server tier isn’t a very active LPAR, and is co-located with the NFS mounted Siebel shared file system. The Siebel shared file system manages attachments and user preference configuration files. The Siebel shared file system is NFS available as a requirement for multiple application servers. The Siebel Gateway Server is configured with PowerHA, in an active/active arrangement.
- The Oracle Database tier resides on its own LPAR. However, at times, the Siebel Gateway Server may be co-located and this is desired to compare the performance with the dedicated environment. The database tier has the lion’s share of file system attachments to the v7000 SAN storage.
- The Siebel Application Server typically has the largest CPU and memory footprints. The application server LPAR will be configured to support Active Memory Expansion and Active Memory Sharing.
- The Siebel Web server isn’t a CPU intensive tier for this workload. One web server is normally sufficient to support the simulation load. The Siebel Web server LPAR will share memory with the application server memory, using Active Memory Sharing.

**Oracle Database with Easy Tier**

The Oracle Database disks are housed on the Storwize V7000. The Storwize V7000 is a mid-range SAN disk array with some very high-end features. The V7000 includes a feature called IBM System Storage
Easy Tier. Easy Tier provides the ability to store hot disk pages on solid state drives. In order to understand how to enable Easy Tier, a few concepts need defining. Within the V7000, a storage pool is a collection of managed disks (MDisks) that jointly contain all of the data for a specified set of volumes. A managed disk is a logical unit of physical storage. Think of MDisks as a raid set, or mirrored set of disks, that act as a single disk of storage. By creating the storage pool (managed disk group) with both generic SSD MDisks and generic HDD MDisks, Easy Tier is automatically turned on for pools with both SSD MDisks and HDD MDisks. For the Oracle Database residing on the database tier, as soon as the SSD MDisk, mdisk0, was added to the storage pool, Easy Tier became active, as shown in Figure 4.

Once Easy Tier becomes active, hot pages begin migrating to the SSD Mdisk from the HD Mdisks, mdisk4 and mdisk7 respectively. The system automatically and non-disruptively moves frequently accessed data from HDD MDisks to SSD MDisks, thus placing such data in a faster tier of storage. The Oracle Database tier had exceptional I/O performance, and Easy Tier can provide you the unique ability to leverage Solid State Drives automatically.

Figure 4. Easy Tier is active
PowerHA layout

Figure 5 shows the topology of the PowerHA configuration used to provide High Availability (HA) for the Siebel Gateway Server and Siebel File System, during the Siebel PowerVM project. The environment consists of two virtual servers, or nodes, split across two Power 750 servers. Each server is running AIX 6.1 Technology Level 7 (TL07) and PowerHA 7.1.1 Service Pack 2 (SP02).

Since each node is virtual, only one Ethernet adapter is required per node, to satisfy HA redundancy requirements for the production network. A virtual Ethernet adapter is not real and is therefore not a single point of failure. In a PowerVM environment, the single point of failure is the hypervisor and network resilience is offloaded to the virtual I/O server (VIOS). The VIOS configuration in this environment is not shown in the topology for clarity since PowerVM configuration is outside the scope of this PowerHA discussion.

A total of three Storage Area Network (SAN) disks, or LUNs, were created on an IBM Storwize V7000 storage server for use in the PowerHA configuration. LUN hdisk1 is used as the Cluster Aware AIX (CAA) repository disk. In AIX 6.1, the AIX kernel is cluster aware and stores critical HA configuration data on a single CAA disk. PowerHA 7.1 is the first version of HA to use the CAA infrastructure. The CAA disk can
be physical or virtual. In this environment, the CAA disk and application software disks are virtual and are presented by PowerVM end-pool virtualization technology.

In previous PowerHA versions, a serial, non-IP heartbeat would be established through physical disks using target mode SSA or target mode SCSI technology. In PowerHA 7.1 and above, when the CAA disk is virtual, a non-IP heartbeat is achieved using SAN fabric-based communication technology. Serial-based communications are passed through VLAN 3358 to the VIOS, which performs SAN-based communications on behalf of the cluster node.

Figure 5 shows the home-node location of the Siebel software configured in the HA cluster. PowerHA1 is the home-node location for the Siebel Gateway Server which was installed and configured using an HA application resource group on LUN hdisk2. PowerHA2 is the home-node location for the Siebel File System which was created using a HA application resource group on LUN hdisk3. The Siebel File System is a NFS exported by PowerHA to the other servers which require access to the Siebel File System.

The Siebel HA cluster ran in an active-active configuration, where each application resource group could fail-over to the other node in the cluster. At any point in time, the Siebel Gateway Server and Siebel File System could be co-located on the same node, or switched entirely to the other node. The Siebel software functions normally regardless of the cluster node on which it resides. The rest of the Siebel Enterprise is unaware that the Siebel Gateway Server and Siebel File System are HA clustered. The Siebel Enterprise accesses the Siebel Gateway Server and Siebel File System using their service IP addresses, which are made highly available by the cluster.

More information about PowerHA for AIX, including a summary about how PowerHA works, the key technologies involved and how to approach building a simple 2-node cluster, can be found at the following IBM developerWorks website:

- The PowerHA for AIX (formerly HACMP) cheat sheet

Refer to “Appendix 2: PowerHA configuration and application monitoring scripts” for examples of the PowerHA commands used to create the HA cluster for the Siebel Gateway Server and Siebel File System and for examples of scripts used to monitor and maintain the HA applications.

Once the Siebel cluster is up and running, the Siebel Gateway Server and Siebel File System are accessed and used as per a standard Siebel CRM installation. Accessing software in a PowerHA cluster is no different to accessing software installed on a standard server. The only time that any differences may be noticed is when the software becomes inaccessible for a short period of time in the event that PowerHA has to relocate the software to another node in the cluster. The length of time taken to move an application resource group in a cluster is dependent on the software running in the resource group. The time required is directly proportional to how long it takes to shut down, clean up and restart the software being relocated.

The cost of virtualization

The availability of a dedicated set of machines and a separate virtual set of machines provided flexibility to mix and match dedicated with virtual LPARs. One question that was of interest is what is the cost of virtualization? From the recorded PSPP runs, it can be seen that the CPU reported on the virtualized environment increased by 3%. This experiment was done by having the database, and the Siebel web
server and Siebel Gateway Server tiers reside on dedicated Power 740 servers while the application tier resided on a Power 750 server. No other LPARs were running on the virtualization machines.

<table>
<thead>
<tr>
<th>What we learned:</th>
<th>CPU%</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated</td>
<td>69%</td>
<td>0</td>
</tr>
<tr>
<td>Dedicated on VIOS</td>
<td>72%</td>
<td>3%</td>
</tr>
<tr>
<td>Shared Pool on VIOS</td>
<td>72%</td>
<td>3%</td>
</tr>
</tbody>
</table>

*Table 1. Cost of virtualization test results*

**Active Memory Expansion**

Active Memory Expansion (AME) is the ability to define an area of memory for compressing infrequently used pages and reducing the memory requirements of the system as a whole. Two Siebel CRM enterprise tiers are likely candidates for AME, the database and the application tier.

Siebel Application Servers typically contain large memory footprints. It is not uncommon to see memory footprints above 64GB of memory. With AME, the Siebel Application Server can reduce the amount of required memory, with minimal CPU increase.

AME can also extend the usage of current memory, assuming available CPU for the additional users. The team did not test extending usage by increasing client loads, but it is assumed that if memory is saved, then that memory can be put to use for more clients.

**Requirements**

In order to use AME the system requires the following:

- AIX 6.1 TL04 SP2+
- AME activation
- Disabling 64KB pages

**Activating AME**

AME must be activated on the server prior to usage. To verify activation, using the HMC, go to Systems Management -> Servers within the left panel. In the right panel check the target server, Click on Task- >Properties. When the window opens up, click on the Capabilities tab. Find ‘Active Memory Expansion Capable’ row and validate the value is true. If AME is not enabled, purchasing the AME license and providing the key enables the AME feature. Figure 6. Enabling AME

Figure 6 below shows that AME is enabled on one of the Power 750 machines.
Once AME is available on the server, each LPAR has an AME option. By default AME is disabled within the LPAR. To enable AME within the LPAR, using the HMC, navigate to the LPAR profile. Click on the memory tab, and at the bottom, click the checkbox to enable AME. Once enabled, the AME expansion factor can be modified. In Figure 7 below, a Power 750 machine LPAR shows AME enabled and with an expansion factor of 1.4.

Be conservative in testing AME expansion factors. Aggressive expansion factor values can degrade performance of the system, causing slowness.
Determining the expansion factor

The AIX `amepat` command is a helpful tool in analyzing and determining the best possible AME expansion factor. The default `amepat` command output values for monitoring and recommending values is based on the physically defined memory within the LPAR. The physically defined memory is best used when extending the workload on the current system. The default command is:

```
amepat [-P recordfile ] [monitor_time_in_minutes]
```

For capturing recommendations for reducing memory usage within an LPAR, the AIX `amepat` command requires:

- Configuring memory so that there is little or no paging space usage
- Having AME disabled during the monitoring runs
Using the AIX `amepat` `-a` option

The `-a` option of the AIX `amepat` command bases memory estimations on the current in-use memory rather than the physically configured memory. By discovering the ‘in-use’ memory for the workload, fine-tuning the memory footprint is then viable.

**Testing AME on the Siebel application tier**

The Siebel Application Server is the most CPU and memory intensive tier within the Siebel CRM enterprise. The runs that followed are the results of discovering the best AME expansion factor.

**Run20 – Collecting AME recommendations**

Run20 started with 4000 PSPP clients driving a simulation. During steady state, the AIX `amepat` command was executed:

```
/usr/bin/amepat -e 1.10:3.00:0.3 -a 05
```

The AIX `amepat` command results recommended a 2.08 AME expansion factor. The amount of physically consumed memory is 53.5GB. See the “True Memory In-Use” value in Figure 8. From a previous run we knew the memory footprint for 4000 users was 60GB, and the results indicated 53.5GB would be required.
### Figure 8. Run20 amepat results

There are some important areas to note. The key information within the output is the:

<table>
<thead>
<tr>
<th>System Resource Statistics</th>
<th>Average</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Util (Phys. Processors)</td>
<td>3.43 [43%]</td>
<td>1.46 [18%]</td>
<td>5.49 [69%]</td>
</tr>
<tr>
<td>Virtual Memory Size (MB)</td>
<td>54189 [99%]</td>
<td>53839 [98%]</td>
<td>54363 [99%]</td>
</tr>
<tr>
<td>True Memory In-Use (MB)</td>
<td>54684 [100%]</td>
<td>54557 [100%]</td>
<td>54743 [100%]</td>
</tr>
<tr>
<td>Pinned Memory (MB)</td>
<td>2635 [5%]</td>
<td>2632 [5%]</td>
<td>2637 [5%]</td>
</tr>
<tr>
<td>File Cache Size (MB)</td>
<td>744 [1%]</td>
<td>710 [1%]</td>
<td>762 [1%]</td>
</tr>
<tr>
<td>Available Memory (MB)</td>
<td>68 [0%]</td>
<td>10 [0%]</td>
<td>186 [0%]</td>
</tr>
</tbody>
</table>

### Active Memory Expansion Modeled Statistics:

- Modeled Expanded Memory Size: 53.50 GB
- Average Compression Ratio: 2.37

### Expansion Modeled True Modeled CEU Usage

<table>
<thead>
<tr>
<th>Factor</th>
<th>Memory Size</th>
<th>Memory Gain</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.15</td>
<td>45.00 GB</td>
<td>8.50 GB [19%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>1.49</td>
<td>36.00 GB</td>
<td>17.50 GB [49%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>1.79</td>
<td>30.00 GB</td>
<td>23.50 GB [78%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>2.00</td>
<td>25.75 GB</td>
<td>27.75 GB [100%]</td>
<td>0.00 [0%]</td>
</tr>
</tbody>
</table>

### Active Memory Expansion Recommendation:

The recommended AME configuration for this workload is to configure the LEAR with a memory size of 25.75 GB and to configure a memory expansion factor of 2.08. This will result in a memory gain of 100%. With this configuration, the estimated CPU usage due to AME is approximately 0.00 physical processors, and the estimated overall peak CPU resource required for the LEAR is 5.49 physical processors.
- Virtual Memory Size (54189MB)
- True Memory In-Use (54684GB)
- Active Memory Expansion (Disabled)
- Modeled Expanded Memory Size (53.50GB)
- And finally the suggested table of expansion factors (4 lines are shown from 1.19 to 2.08 expansion factor)

**Run22 – testing 1.19 AME expansion factor**

Starting conservatively, the next successful run occurred with AME expansion factor of 1.19. We also did an expansion factor test of 2.08, but this failed during ramp-up and caused the system to behave sluggishly. Recall the purpose is to reduce memory footprint while achieving equal performance.

To enable AME with 1.19 expansion factor, the LPAR was changed as follows:

1. The LPAR was shut down prior to modifying the LPAR profile.
2. The memory size of the LPAR was changed from 53.5GB to 45 GB as specified by the 1.19 row within the recommended expansion factors.
3. The LPAR profile enabled Active Memory Expansion. The AME expansion factor was set to 1.19.
4. The LPAR was rebooted with the new changes to AME.

Validation of the changes is just a matter of running the AIX `amepat` command without arguments.

Run22 was successful without errors. We collected more information using the AIX `amepat` command. The salient section is shown below:
Figure 9. Run22 amepat results
As shown in Figure 9, the AME expansion factor, and the “Modeled Expanded Memory Size” is validated. Statistics include the actual amount of virtual memory consumed, along with the actual size of memory configured (True Memory In-Use). The important statistic is the “compressed memory” value, showing that AME has consumed 10749MB of memory, or nine-teen percent of the total 52 GB of memory. If the compression memory was zero, or zero per cent, then this workload would not be a candidate for AME.

Run29 – testing 1.45 AME expansion factor

As the header name suggests, this is the 29th run. Multiple runs occurred between run22 and run29, helping to hone in on the best AME expansion factor for Siebel CRM performance. The best AME expansion factor were values between 1.4 and 1.5. As the figure shows, the configuration (1.45 expansion factor) is estimating just 3% CPU to compress and uncompress memory from the AME memory pool. Figure 10 provides a means of completing the process of finding the best AME expansion factor for this workload.
Analyzing AME performance

Besides looking at the memory statistics among the PSPP simulated runs, it is useful to compare the response time, CPU percentage, memory, and compression ratios. The table below shows runs 20 through 30. Run21 and run26 had AME expansion factors too aggressive, causing the undesired effect of moving memory pages to swap space. Care must be taken, as aggressive swap can cripple a system.

![Table showing AME performance results](image)

The recommended AME configuration for this workload is to configure the LPAR with a memory size of 35.50 GB and to configure a memory expansion factor of 1.45. This will result in a memory gain of 46%. With this configuration, the estimated CPU usage due to AME is approximately 0.23 physical processors, and the estimated overall peak CPU resource required for the LPAR is 5.16 physical processors.

Figure 10. Run29 amepat results
<table>
<thead>
<tr>
<th>Run</th>
<th>Avg Response Time (secs)</th>
<th>AME Factor</th>
<th>CPU% Percent</th>
<th>Memory (GB)</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Virtual</td>
<td>Real</td>
</tr>
<tr>
<td>20</td>
<td>0.12</td>
<td>71%</td>
<td>54</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>2.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.14</td>
<td>1.19</td>
<td>78%</td>
<td>52</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>0.14</td>
<td>1.49</td>
<td>78%</td>
<td>52</td>
<td>37</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>1.79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0.41</td>
<td>1.67</td>
<td>90%</td>
<td>52</td>
<td>33</td>
</tr>
<tr>
<td>29</td>
<td>0.14</td>
<td>1.45</td>
<td>79%</td>
<td>52</td>
<td>36</td>
</tr>
<tr>
<td>30</td>
<td>0.14</td>
<td>1.4</td>
<td>78%</td>
<td>52</td>
<td>37</td>
</tr>
</tbody>
</table>

Table 2. AME performance comparison

Run20 is the baseline run where AME is disabled. Comparing run20 with run22 yields a cost of 7% CPU for AME compression at a 1.2 AME expansion factor. Increasing the AME expansion factor to 1.67 caused the CPU to climb to 90% and also caused a three-fold increase in average response time. A value of AME between 1.4 and 1.5 exhibits the best tradeoff between memory compression and performance. The AIX `amepat` command output suggests that a 1.45 expansion factor yields the best tradeoff between CPU usage and memory efficiency.

AME yielded 31% memory savings in the PSPP simulation on the Siebel Application Server for this workload. Each individual client’s savings may vary. Be prudent and conservative when testing AME expansion factor values.

**Testing AME on the Siebel database tier**

The iterative process of testing values of AME expansion factor on the database tier did not yield the equivalent advantage that we experienced with the application server tier. The database tier exercise is a good example of when Active Memory Expansion is not applicable. *Note: other database AME exercises have yielded AME memory savings; don’t take the results here as proof that the Siebel CRM database will not benefit from AME in your environment.*

**Run89 – Collecting AME recommendations**

Prior to collecting data for AME, we validated that Active Memory Expansion was disabled within the LPAR. That AME is disabled can be confirmed using the AIX `amepat` command output. Run89 started with 4000 clients driving the PSPP simulation. During steady state, the AIX `amepat` command was executed

```
/usr/bin/amepat -e 1.30:2.00:0.1 -a 05
```

The “-e” option provides a means of defining the window of estimated recommendations, while the ‘05’ value is the number of minutes to collect AME data.

By executing the AIX `amepat` command within a loop, multiple executions report how the `amepat` tool output varies over time. The largest memory usage reported over the captured executions became
the target and is shown in Figure 11 below. The actual memory used during the run89 simulation was 17791MB ("Virtual Memory Size"). The AIX amepat tool recommends testing values from 1.38 up to 1.98. The first AME expansion factor to test is near 1.4.

```
Command Invoked       : /usr/bin/amepat -a 1.30:2.00:0.1 -a 05
Date/Time of invocation : Sun Apr 22 10:46:19 PDT 2012
Total Monitored time   : 8 mins 27 secs

System Configuration:-----------------------
Partition Name            : p120n01
Processor Implementation Mode : POWER7
Number Of Logical CPUs     : 3
Processor Entitled Capacity : 2.00
True Memory               : 24.00 GB
SMT Threads               : 4
Active Memory Expansion   : Disabled

System Resource Statistics:-----------------------
                   Average     Min     Max
CPU Util (Phys. Processors)        0.16 [ 8%] 0.11 [ 6%] 0.24 [12%]
Virtual Memory Size (ME)             17791 [72%] 17778 [72%] 17818 [73%]
True Memory In-Use (MB)              24491 [100%] 24478 [100%] 24517 [100%]
Pinned Memory (MB)                  1883 [ 8%] 1883 [ 8%] 1883 [ 8%]
File Cache Size (MB)                 6594 [ 27%] 6594 [ 27%] 6594 [ 27%]
Available Memory (MB)                5958 [ 24%] 5932 [ 24%] 5972 [ 24%]

Active Memory Expansion Modeled Statistics:-----------------------
Modeled Expanded Memory Size : 18.25 GB
Average Compression Ratio : 3.10

<table>
<thead>
<tr>
<th>Expansion Factor</th>
<th>Modeled True Memory Size</th>
<th>Modeled Memory Gain</th>
<th>CPU Usage Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.38</td>
<td>13.25 GB</td>
<td>5.00 GB [33%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>1.59</td>
<td>11.50 GB</td>
<td>6.75 GB [59%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>1.79</td>
<td>10.25 GB</td>
<td>8.00 GB [78%]</td>
<td>0.00 [0%]</td>
</tr>
<tr>
<td>1.98</td>
<td>9.25 GB</td>
<td>9.00 GB [97%]</td>
<td>0.00 [0%]</td>
</tr>
</tbody>
</table>
```

Figure 11. Run89 amepat results

**Run92 – Testing AME 1.4 expansion factor**

Run92 used the same methodology as run22, with changes:

1. The database LPAR was shut down.
2. The LPAR profile memory size of the machine was reduced from 24 GB to 18 GB.
3. The LPAR profile enabled Active Memory Expansion.
4. The LPAR profile set the AME expansion factor to 1.4.
5. The database LPAR is booted with the new changes.
Figure 12 below shows the last AIX `amepat` output, having the highest memory usage. What is telling about the AIX `amepat` output for run92 is that memory compression affectively did not occur. The amount of memory compressed was 18MB, and zero percent of the memory. Follow up runs with higher AME expansion factor values did not show compression happening.

Figure 12. Run92 `amepat` results

Additionally, the AIX `amepat` output recommendation explains the possibility that different workloads may provide potential compression, but that this workload is too light to be a good candidate for AME.
Active Memory Expansion Recommendation:

The amount of currently compressed memory for this workload is small. Only 0.00% of the LPAR’s current expanded memory is compressed. This may be due to light load on the system. This tool relies on monitoring accesses to compressed memory to calculate AME CPU utilization estimates. Due to the small amount of compressed memory, this tool cannot currently make recommendations on other AME configurations. Please increase the load and run this tool again.

Figure 13. Additional Run92 amepat output

This is not surprising when analyzed. The Oracle SGA size was 13GB and that left 5 GB for client connections and memory.

Best AME practices

The technical experiences that emerged from the work with AME yielded the following insights:

- Siebel CRM has the ability to behave well with a large percentage of paging space consumed. Initially, the environment was tuned to take advantage of the paging space consumption. This showed through within the AIX amepat output, by having the virtual memory size much larger than the true memory size. And we were sizing using the true memory size. AME expansion factors recommendations became inconsistent. Therefore it is recommended to configure the system with ample memory (avoiding using paging space) during AME testing.

- If the target is to reduce memory, then use the ‘-a’ option with the AIX amepat command. The ‘-a’ option uses the actual memory used for estimating memory recommendations.

- The ‘-e’ option is useful for fine tuning and cutting down on the number of simulations. This option defines how the recommendation estimates are delivered.

- The actual execution of the AIX amepat command, typically takes much longer than the time specified as a command line argument (5 minutes). The figures show that up it took up to 23 minutes to execute the amepat command. Make sure your steady state environment can last the entire time the AIX amepat command needs to finish.

- Take your ‘collection’ measurements with AME disabled. The team found that the results yielded better resultant estimates for AME.

- Dynamic LPAR and AME did not work to our satisfaction. It would seem that DLPAR would be a perfect means of reducing the iterative process for finding the best AME value. The team tried using DLPAR to adjust the memory footprint while AME is running. DLPAR seemed to have no effect on AME.

- The simulations in this paper had very tight memory constraints. As a customer, leave yourself extra memory to handle volatile CPU peak or memory peaks. In our experience, an extra 10% additional memory can handle 25% user load volatility, and provided room for additional users.
AME requires re-tuning if the load increases dramatically. Changing the load from 2000 to 3000 users required additional memory; however the AME expansion factor of 1.45 held firm.

### Active Memory Sharing

Active Memory Sharing (AMS) intelligently flows memory from one logical partition (LPAR) to another for increased utilization and flexibility of memory. The concept is very similar to that of the shared processor pool and micro-partitions. It allows you to oversubscribe memory on a POWER6 system and let the system (the POWER Hypervisor™) allocate memory where it is needed. In situations where memory resources are tight and the environment provides the luxury of sharing memory resources, AMS is the perfect vehicle. One example might be a machine that is fully configured with lower capacity memory DIMMs and adding more memory would cause replacement of all DIMMs to larger sizes. Another example would be to create a temporary development LPAR by borrowing memory from other development non-critical LPARs that have spare memory.

AMS is best utilized with complementary LPARs where sharing of memory is not contentious. An example of two complementary LPARs would be an LPAR that works at night (Batch Loads) while another LPAR works during the day (Business Ledger LPAR). The Siebel CRM PSPP kit and its simulation does not provide an LPAR that is complementary to another LPAR within the Siebel CRM enterprise. However, for the sake of showing AMS within a live Siebel CRM environment, two Siebel CRM tiers within a shared memory pool were chosen.

The candidates for sharing memory within the PSPP are the database, Siebel Gateway Server, Application server and the Web server. The database is generally thought of as a mission critical LPAR and thusly, it is commonly configured with dedicated memory and processors. The Siebel Gateway Server tier is idle 99% of the time and is utilized during startup, shutdown, and during Siebel configuration changes. It has a memory footprint of 1 GB – too small to be effective. Through the process of elimination, the application server and the web server are selected.

### AMS requirements

The Active Memory Sharing documentation is very good in defining the AMS requirements and for any question please defer to the standard documentation. For brevity, in order to use AMS, the system requires the following:

- AIX 6.1 TL 3 or later
- Machine Firmware 710_43 or better
- HMC Management Console 7.7.2 SP1 or better
- Virtual I/O Server version 2.1.3.10-FP23 or better
- AMS Activation
- Virtualized Ethernet and virtualized disk I/O. (All I/O)
- AMS paging spaces defined on the VIOS servers.

### AMS Activation

To verify if AMS is activated, using the HMC, go to Systems Management ->Servers within the left panel. In the right panel check the target server, Click on Task->Properties. When the window opens up, click on
the Capabilities tab. Find ‘Active Memory Sharing Capable’ row and validate the value is true. If not, purchasing the AMS license and providing the key will be required to enable the feature.

**AMS Paging devices**

The hardest tasks during AMS setup is specifying, allocating and defining the required paging space devices. The number of required paging devices is based on the total number of LPARs that will be sharing the single shared memory pool. There is only one shared memory pool per server. If three LPARS will share the pool, then three paging devices are required. The size of all paging devices must be the same and at least as large as the largest LPAR’s memory footprint. For example, if there are three LPARS with memory sizes of 16GB, 24GB and 48GB, then 3 paging devices must be defined that are at least 48GB in size. It is prudent to think ahead and make paging devices as big as the potential growth of the largest LPAR’s memory footprint.

The lab technician created two paging devices per VIOS, each paging device of 64GB in size for the two LPARs participating in AMS.

The Active Memory Sharing documentation has ample description of the process to setup the shared memory pool via the HMC, once the paging spaces are available on the VIO servers. Without the paging devices, Active Memory Sharing will be disabled within the OS. From Figure 14, two paging devices can be seen, ams01 and ams02. Currently only one LPAR is booted, making the device status of ams02 as inactive.

**AMS shared memory pool**

Figure 15 below shows a shared memory pool configuration for one of the Power 750 machines. The maximum pool size defines how large the pool can grow using dynamic LPAR capability. The current pool size in the figure is 50GB. All LPARs using AMS have 50GB of memory to share.
LPAR shared memory enabling

Via the HMC, enabling AMS within an LPAR is simple. Within the LPAR profile, click on the memory tab, check "shared memory". The example below shows the values for the web LPAR within the Siebel CRM enterprise. The memory pool is listed as 48GB and the application server LPAR has 40GB of memory (Desired Memory) defined.
The team configured an AMS environment, highlighting the abilities of AMS with Siebel CRM. The environment included AME for the application server. The application server booted with a memory footprint of 36 GB. The web server was given a memory footprint of 8 GB and did not have AME enabled. The shared pool was not big enough to satisfy both LPARs’ request of 40 GB, if they fully utilized their memory. One or both of the LPARs will have to borrow memory from the other. Since the two LPARS do not have complementary loads, the expectation is to see loaning of memory occurring, and contention.

The AIX `vmstat` command provides the ability to view loaned memory. Specifying the –h option or the combination of –h with –v provides AMS statistics.

The AIX `vmstat` command provides the ability to view loaned memory. Specifying the –h option or the combination of –h with –v provides AMS statistics.

Table 3. AIX vmstat command output below was taken from the application server and shows the AIX `vmstat` –h command output (some fields were removed for simplicity sake). The AIX `vmstat`
pmem (physical memory) field shows the application server consuming 36GB of memory. The system was also using AME with a virtual memory size of 53GB, based on a 1.4 expansion factor. AME grabbed the memory early and the values of pmem stayed at 36GB for some time. Meanwhile, the loan field starts off at 1.16GB loaned away at the first interval, and ends with .04 GB loaned away at the end of the sixth interval. This meant that the web server had initially 1.16GB of memory, which the application server took back. Normally the pmem field would change with memory return, however, AME places different requirements upon the memory.

<table>
<thead>
<tr>
<th>r</th>
<th>b</th>
<th>us</th>
<th>sy</th>
<th>id</th>
<th>wa</th>
<th>pc</th>
<th>ec</th>
<th>hpi</th>
<th>hpit</th>
<th>pmem</th>
<th>loan</th>
<th>hr</th>
<th>mi</th>
<th>se</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>0</td>
<td>92</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>7.66</td>
<td>109.4</td>
<td>742</td>
<td>3</td>
<td>36</td>
<td>1.16</td>
<td>19:38:48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>0</td>
<td>90</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>7.52</td>
<td>107.4</td>
<td>684</td>
<td>4</td>
<td>36</td>
<td>0.88</td>
<td>19:39:03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>0</td>
<td>91</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>7.6</td>
<td>108.5</td>
<td>683</td>
<td>4</td>
<td>36</td>
<td>0.65</td>
<td>19:39:18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>0</td>
<td>91</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>7.46</td>
<td>106.5</td>
<td>584</td>
<td>5</td>
<td>36</td>
<td>0.46</td>
<td>19:39:33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>1</td>
<td>91</td>
<td>10</td>
<td>9</td>
<td>0</td>
<td>7.56</td>
<td>108.2</td>
<td>581</td>
<td>5</td>
<td>36</td>
<td>0.25</td>
<td>19:39:48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>1</td>
<td>90</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>7.58</td>
<td>108.2</td>
<td>498</td>
<td>5</td>
<td>36</td>
<td>0.04</td>
<td>19:40:03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. AIX vmstat command output

There is also loaned memory correlation that can be seen between the LPARs. Table 4 below shows the AIX vmstat command output, synchronized by timestamp, and the loaned memory values for the web server (on the left) and the application server. Note, in this example the shared memory pool size was 41GB.

Table 4. AIX vmstat output for the web server and application server

At steady-state during the PSPP simulation, the generated system load placed stress on the AMS shared memory. In Table 5, when the application server requested memory from the hypervisor, the hypervisor pulled memory from the web server and caused the application server to slow down. The system was still doing work as can be seen that the CPU didn’t drop, it increased to handle the load.
Later within the same run, see Table 6 below, the operating systems LRU page replacement algorithm kicked in, and pages had to be freed, (seen from the SR column). Processes were blocked waiting as the pages were freed and AMS pages were involved. Because of the artificial situation of demonstrating AMS with contentious Siebel CRM LPARs, this behavior is demonstrated. Artificially sharing memory with non-complementary LPARs should be avoided.

**Best AMS practices**

The PSPP simulation stressed the system further than a typical customer environment. The technical experiences working with AMS yielded the following insights:

- It's very important to know your workload for the LPARs using the shared memory pool of AMS. Production systems are not the place to experiment with AMS.
- AMS works best when LPARs are not competing for the same memory resources.
- There were times when AMS borrowed memory, even when memory was still available within the shared pool. Keep track of memory usage during non-critical periods.
- Understand the AIX `vmstat` command output for AMS. Know how HPI and HPIT influence memory loaning and keep HPI numbers small.
- When using Live Partition Mobility with AMS, the memory situation can improve. If the LPAR is migrating to another machine, the memory frees up on the source machine while resources...
on the target are taxed. Care must be planned such that a migrating LPAR with AMS doesn’t over-tax the size of the shared memory pool on the destination machine.

- AMS can co-exist with AME, as exemplified by the Siebel Application Server configuration. Be conservative when configuring AME with AMS. Keeping the shared memory pool bigger than the sum total LPAR memory requirements is best.

**Simultaneous multithreading**

Simultaneous multithreading (SMT) is the ability of a single physical processor to simultaneously dispatch instructions from more than one hardware thread context. Because there are four hardware threads per physical processor on the POWER7 processor, additional instructions can run at the same time.

The AIX `smtctl` command controls the enabling and disabling of processor simultaneous multithreading mode. By setting different values using the AIX `smtctl` command, the team configured the underlying system to test SMT4 (four threads), SMT2 (two threads) and SMT1 (one thread and disabled). Modifying SMT value is dynamic and changes can be seen in real time with monitoring tools like the AIX `nmon` utility.

**Modifying the SMT value on-the-fly**

In this PSPP run, the application server LPAR had an entitled capacity of four CPU cores, with the ability to borrow an additional four cores. The database, Siebel Web server and Siebel Gateway Server, were housed on the dedicated environment on different hardware. No other virtual LPARs were running while the test occurred and therefore all borrowed CPU resources were originally idle and available. The system started at the default value of SMT4. At steady state, we modified the SMT value using the command: `smtctl -t 2`. The change occurred within the chart at time 10:50. The system responded with fewer threads and the system statistics showed a dramatic step-up of CPU usage. At 11:05, the team modified SMT by disabling the ability to use simultaneous multi-threading, with the command: `smtctl -m off`. Thereafter, every 15 minutes we reversed our changes. A step-wise CPU utilization pattern results and is seen in Figure 17.

**NOTE:** Whenever lowering SMT values, it is wise to have ample CPU, in the event of consuming all CPUS.
CPUALL `nmon` worksheet records the percentage of CPU based on the entitled CPU value, which is four cores. The fact that it hit 100 percent should not be taken that the system was at 100 percent. The system actually used more than four cores worth of CPU, borrowing uncapped cores. For a more reliable picture of what was happening, the `nmon` LPAR worksheet is examined, specifically the ‘CPU% vs. VP’s’ graph (Figure 18).

**Figure 17. CPU utilization as SMT value is changed**

**Figure 18. CPU% vs VPs graph**

Here, the CPU percentage usage is above 50%, where 50% represents four cores and the 100% represents eight cores. Calculating averages during the 15 minute periods, the team generated the table with percentage difference between levels of SMT. The purpose of Table 7. SMTCTL performance impact is to show the reader the data points taken during the SMT exercise. Caution is in order to use this information for capacity planning; most likely customer results would be different. In our simulation, the default SMT value, (SMT4) yielded a 25% improvement.
### Table 7. SMTCTL performance impact

<table>
<thead>
<tr>
<th>SMTCTL VALUE</th>
<th>LPAR VP AVG</th>
<th>% Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:50:35</td>
<td>2</td>
<td>44.92</td>
</tr>
<tr>
<td>11:05:21</td>
<td>0</td>
<td>59.23</td>
</tr>
<tr>
<td>11:21:51</td>
<td>2</td>
<td>45.16</td>
</tr>
<tr>
<td>11:37:36</td>
<td>4</td>
<td>34.44</td>
</tr>
</tbody>
</table>

### SMT best practices

In most situations, it is best to leave the SMTCTL value to its default for performance. Testing the different values of SMT with Siebel CRM confirmed that the default works best with this workload. Here are some points to remember:

- Take caution, when dynamically modifying the SMTCTL value on a highly loaded system. In some testing, the system backed up and delayed client response times due to changing SMTCTL values.
- Wait for the SMT change to complete and stabilize prior to taking measurements. The time it takes to change from one SMT value to another is dependent on the load. The AIX `nmon` command is useful for providing real-time CPU de-allocation or allocation.
- It is safer and easier to go from SMT1 upwards to SMT4 since additional CPU resources become available as you go from SMT1 to SMT4.

### Conclusion

A Siebel CRM workload has been shown taking advantage of Active Memory Expansion and reducing memory usage by 30%. That same 30% of memory could support more users or be given or loaned to another LPAR. Virtualization provides the sharing of disk, and network resources, making the most of your valuable hardware resources at the cost of VIO servers and minimal 3% cost to the target LPAR. The Siebel CRM workload showed the 25% benefit of simultaneous multi-threading and why SMT4 is the default value.

The combination of PowerVM with Siebel CRM applications for Oracle and IBM customers provides a cornucopia of features for gaining the best utilization of Power hardware and resources. The features of Active Memory Expansion and Active Memory Sharing build on top of the strong virtualization foundation of PowerVM, and are providing Oracle and IBM customers with newly reclaimed CPU and memory resources. These test results confirm that Siebel CRM on IBM Power Systems clients can be confident when taking advantage of AIX, PowerVM and Power Systems technologies in their environments, and in obtaining cost of ownership gains through consolidation using these virtualization technologies.
Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM’s PSPP 21,000 user result and description of the PSPP Kit.

- Understanding Processor Utilization on POWER Systems - AIX
  http://www.ibm.com/developerworks/wikis/display/ WikiType/Understanding+Processor+Utilization+on+POWER+Systems+-+AIX

- IBM eServer pSeries [System p] Information Center
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp

- IBM Publications Center

- IBM Redbooks
  www.redbooks.ibm.com/

- Platform sizing and performance program white papers

- IBM PowerVM Live Partition Mobility
  http://www.redbooks.ibm.com/abstracts/sg247460.html?Open

- Active Memory Expansion Wiki
  https://www.ibm.com/developerworks/wikis/display/WikiType/IBM+Active+Memory+Expansion

- Active Memory Sharing
  http://www.redbooks.ibm.com/abstracts/redp4470.html

- NPIV and the IBM Virtual I/O Server (VIOS)
About the authors

Jubal Kohlmeier is an advisory software engineer in the IBM Systems & Technology Group (STG), Business Systems Division. Jubal has more than 30 years of industry experience in the computing industry and more than 15 years with Oracle’s Siebel CRM products. Jubal has worked with customers with Siebel CRM and Oracle Databases in achieving high volume mission critical solutions on IBM Power Systems and System x™ servers, and has delivered high watermark benchmark Siebel CRM proof points.

Mark Trbojevic is an advisory software engineer in the IBM Systems & Technology Group (STG), Business Systems Division. Mark has more than 25 years of experience in the computing industry and has been with IBM for 12 years. He has a background in developing and designing bespoke customer relationship management (CRM) software solutions for blue-chip companies. Mark has been supporting Oracle solutions on IBM Power Systems and System x™ hardware for more than ten years, specializing in Siebel CRM, Oracle Business Intelligence Enterprise Edition (BIEE) and Data Warehousing applications.
Appendix 1: Software versions and releases

The following software versions were used in the testing documented in this white paper.

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX Version</td>
<td>6.1 TL6</td>
</tr>
<tr>
<td>Oracle Database</td>
<td>11.2.0.2.3</td>
</tr>
<tr>
<td>Siebel CRM</td>
<td>8.1.1.4</td>
</tr>
<tr>
<td>Java™</td>
<td>1.5.0</td>
</tr>
<tr>
<td>OATS (Oracle)</td>
<td>9.21.0055</td>
</tr>
<tr>
<td>HMC</td>
<td>7.3</td>
</tr>
</tbody>
</table>

*Table 8. Software versions used*
Appendix 2: PowerHA configuration and application monitoring scripts

The following sections list a collection of shell scripts that were used to configure, run and monitor the AIX PowerHA cluster created for the Siebel CRM on PowerVM technology project.

**PowerHA cluster creation**

After installing and licensing AIX PowerHA software, a cluster can be created and maintained using menus within the AIX SMIT (System Management Interface Tool) interactive application. Alternatively, a cluster can be created and manipulated from the AIX command line using the PowerHA `clmgr` (cluster manager) tool.

Listing 1 shows the `clmgr` commands that were used to create and manage a 2-node PowerHA Siebel CRM cluster. Cluster node `ha1` was built to support a Siebel Gateway Server whilst node `ha2` was built to support a Siebel File System.

```bash
# Cleanup and remove any old cluster definitions
clmgr rm cl nodes=ALL
# If cleanup fails, force cleanup of CAA repository disk (see /var/adm/ras/syslog.caa)
rmcluster -r hdisk1
# Create the cluster/ net 02 and interfaces for en1
clmgr add cluster SiebelCluster REPOSITORY=hdisk1 NODES=ha1,ha2
# Add service address for Siebel Gateway
clmgr add service_ip siebel_ha_gateway NETWORK=net_ether_02
# Add service address for Siebel File System
clmgr add service_ip siebel_ha_filesystem NETWORK=net_ether_02
# Add application server/controller for Siebel Gateway
clmgr add application_controller SiebelGatewayAPP STARTSCRIPT="/powerha/SiebelGatewaySTART" STOPSCRIPT="/powerha/SiebelGatewaySTOP"
# Add application server/controller for Siebel File System
clmgr add application_controller SiebelFilesystemAPP STARTSCRIPT="/powerha/SiebelFilesystemSTART" STOPSCRIPT="/powerha/SiebelFilesystemSTOP"
# Add resource group for Siebel Gateway
clmgr add resource_group SiebelGatewayRG NODES=ha1,ha2 STARTUP=OHN FALLOVER=FNPN FALLBACK=NFB SERVICE_LABEL=siebel_ha_gateway APPLICATIONS=SiebelGatewayAPP VOLUME_GROUP=havg
# Add resource group for Siebel File System
clmgr add resource_group SiebelFilesystemRG NODES=ha2,ha1 STARTUP=OHN FALLOVER=FNPN FALLBACK=NFB SERVICE_LABEL=siebel_ha_filesystem APPLICATIONS=SiebelFilesystemAPP VOLUME_GROUP=havg2 VG_AUTO_IMPORT=false RECOVERY_METHOD=parallel FS_BEFORE_IPADDR=true EXPORT_FILESYSTEM="/files2/siebelfs"
# Add a monitor for Siebel Gateway
clmgr add application_monitor SiebelGatewayMON TYPE=Custom APPLICATIONS=SiebelGatewayAPP MODE=both STABILIZATION=60 RESTARTCOUNT=3 RESTARTINTERVAL=600 MONITORINTERVAL=60 HUNGSIGNAL=9 MONITORMETHOD="/powerha/SiebelGatewayAPPMON" CLEANUPMETHOD="/powerha/SiebelGatewayCLEANUP" RESTARTMETHOD="/powerha/SiebelGatewaySTART" NOTIFYMETHOD="/powerha/SiebelGatewayNOTIFY"
# Add a monitor for Siebel File System
clmgr add application_monitor SiebelFilesystemMON TYPE=Custom APPLICATIONS=SiebelFilesystemAPP MODE=both STABILIZATION=60 RESTARTCOUNT=3 RESTARTINTERVAL=600 MONITORINTERVAL=60 HUNGSIGNAL=9 MONITORMETHOD="/powerha/SiebelFilesystemAPPMON" CLEANUPMETHOD="/powerha/SiebelFilesystemCLEANUP" RESTARTMETHOD="/powerha/SiebelFilesystemSTART" NOTIFYMETHOD="/powerha/SiebelFilesystemNOTIFY"
```
Listing 1 – PowerHA `clmgr` commands used to create and manage the Siebel CRM cluster

Before the Siebel CRM cluster could be created and operated using the commands shown in Listing 1, some configuration ground work had to be completed. Explaining exactly how to prepare AIX servers for a PowerHA cluster is beyond the scope of this paper. Please refer to the latest available AIX PowerHA documentation for further details and full descriptions of the configuration tasks that need to be performed.

For completeness, the core configuration settings and AIX commands used to prepare the servers for the Siebel CRM cluster are reproduced in the following listings. Listing 2 shows the AIX `/etc/hosts` file from a node and displays the IP addresses and host names allocated for the Siebel CRM cluster. Listing 3 shows the AIX commands and configuration settings applied to the cluster nodes, prior to creating the Siebel CRM cluster with the `clmgr` commands shown in Listing 1.
# At minimum, this file must contain the name and address for each
# device defined for TCP in your /etc/net file. It may also contain
# entries for well-known (reserved) names such as timeserver
# and printserver as well as any other host name and address.
#
# The format of this file is:
# Internet Address Hostname # Comments
# Internet Address can be either IPv4 or IPv6 address.
# Items are separated by any number of blanks and/or tabs. A ' #
# indicates the beginning of a comment; characters up to the end of the
# line are not interpreted by routines which search this file. Blank
# lines are allowed.
#
# Internet Address   Hostname    # Comments
# 192.9.200.1       net0sample  # ethernet name/address
# 128.100.0.1       token0sample # token ring name/address
# 10.2.0.2          x25sample   # x.25 name/address
# 2000:1:1:209:6bff:feee:2b7f ipv6sample # ipv6 name/address
# 127.0.0.1         loopback    localhost # loopback (lo0) name/address
# 10.10.10.1        siebel_ha_gateway
# 10.10.10.2        siebel_ha_filesystem
# 10.10.10.173      ha1
# 10.10.10.180      ha2
# 10.10.0.173       ha1_boot0
# 10.10.0.180       ha2_boot0
# 10.10.1.173       ha1_boot1
# 10.10.1.180       ha2_boot1

Listing 2 - AIX /etc/hosts file showing Siebel CRM cluster IP address and host name definitions

# Setup Network Interfaces on HA1 with Non-Routable IP Addresses
# mktpip -i en1 -h ha1 -a 10.10.0.173 -m 255.255.255.0 -g 129.40.71.254 -A no -t N/A
# mktpip -i en2 -h ha1 -a 10.10.1.173 -m 255.255.255.0 -g 129.40.71.254 -A no -t N/A
# chdev -l en1 -a alias4=10.10.10.173,255.255.255.0
# chdev -l en2 -a alias4=10.10.11.173,255.255.255.0

# Setup Network Interfaces on HA2 with Non-Routable IP Addresses
# mktpip -i en1 -h ha2 -a 10.10.0.180 -m 255.255.255.0 -g 129.40.71.254 -A no -t N/A
# mktpip -i en2 -h ha2 -a 10.10.1.180 -m 255.255.255.0 -g 129.40.71.254 -A no -t N/A
# chdev -l en1 -a alias4=10.10.10.180,255.255.255.0
# chdev -l en2 -a alias4=10.10.11.180,255.255.255.0

# Create volume group on cluster application shared disk
mkvg -y havg -f -n -C hdisk2
# Activate the volume group on this node
varyonvg havg

# Create the application filesystem on the volume group
crfs -v jfs2 -g havg -a size=5G -m /files -a logname=INLINE

# Deactivate the volume group so it can be imported on the other node
varyoffvg havg

# Use SSH to import the volume group on the other node
ssh ha2 importvg -n -y havg hdisk2

# Use the following commands on other node to clean up the AIX ODM if the 'importvg'
# command fails with the following error...
# 0516-1939 : PV identifier not found in VGDA.
# 0516-780 importvg: Unable to import volume group from hdisk2.
# ssh ha2 rmdev -dl hdisk2
# ssh ha2 cfgmgr

# Setup disk for Siebel File System

# Create volume group on cluster application shared disk. Specify the MAJOR NUMBER
mkvg -V 40 -y havg2 -f -n -C hdisk3

# Activate the volume group on this node
varyonvg havg2

# Create the application filesystem on the volume group
crfs -v jfs2 -g havg2 -a size=5G -m /files2 -a logname=INLINE

# Deactivate the volume group so it can be imported on the other node
varyoffvg havg2

# Use SSH to import the volume group on the other node. Specify the MAJOR NUMBER
ssh ha2 importvg -V 40 -n -y havg2 hdisk3

# Use the following commands on other node to clean up the AIX ODM if the 'importvg'
# command fails with the following error...
# 0516-1939 : PV identifier not found in VGDA.
# 0516-780 importvg: Unable to import volume group from hdisk3.
# ssh ha2 rmdev -dl hdisk3
# ssh ha2 cfgmgr

# Update cluster rhosts file
echo ha1 > /etc/cluster/rhosts
echo ha1_boot0 >> /etc/cluster/rhosts
echo ha1_boot1 >> /etc/cluster/rhosts
echo ha2 > /etc/cluster/rhosts
echo ha2_boot0 >> /etc/cluster/rhosts
echo ha2_boot1 >> /etc/cluster/rhosts

# Restart cluster clcmd service
stopsrc -s clcmd
sleep 2
startsrc -s clcmd

# To enable risc6000clsmuxpd for clstat, edit '/etc/snmpdv3.conf' file. Add following to end of file
VACM_VIEW defaultView 1.3.6.1.4.1.2.3.1.2.1.5 - included -

# Restart snmpd daemon
stopsrc -s snmpd
sleep 2
PowerHA cluster application scripts

The purpose of an AIX PowerHA cluster is to provide high availability to applications installed within and managed by the cluster. Cluster application are typically stopped, started, monitored and maintained within a cluster through the use of shell scripts.

The following listings reproduce the shell scripts that were created to control and maintain the Siebel CRM PowerHA cluster for the Siebel CRM PowerVM technologies project. These scripts and functions serve only as examples of a possible solution for cluster application management scripts and are in no way definitive. Application installations and configurations vary according to business requirements and cluster application management scripts need to be designed to meet specific business needs.

Listing 4 shows a shell script containing a collection of common shell functions used by most of the cluster application management scripts.

```bash
#!/bin/ksh
################################################################################
### Name: HAScriptFunctions
### Description: This module contains KSH functions used by PowerHA Application Scripts
###
################################################################################

## General Functions

function notify_message
{
    # Unload parameters into more meaningful variables
    _resource=$1
    case ${_resource} in
        gateway) _message="Siebel Gateway Is Experiencing Problems!" ;;
        web) _message="IBM IMS Web Server Is Experiencing Problems!" ;;
        filesystem) _message="Siebel Filesystem Is Experiencing Problems!" ;;
        server) _message="Siebel App Server Is Experiencing Problems!" ;;
        *) printf '
Unknown Resource Name In Notify Function
' ;;
    esac
    # WALL the message to all users
    if [ ! -z ${_message} ]; then
        wall "PowerHA Notify: ${_message}"
    fi
    # Always return TRUE
    return 0
}

function check_siebel_gateway
{
    # Unload parameters into more meaningful variables
    _install_root=$1
    _service_name=$2
    #...
```
_admin_id=$3
_admin_id_password=$4

if [ -f /tmp/suspend_ha_monit          
    printf \n    return 0
elif [ ! -f ${_install_root}/siebenv.sh ]; then
    printf \n    return 0
else
    printf \n    # We'll use a temporary command file
    _command_file=/tmp/ha_script.$$  
    if [ -f ${_command_file} ]; then
        rm ${_command_file}
    fi

    # Put the code we need to check the Gateway into a command file. Everything
    # between the 'cat' and the 'EOF' will be written to the file. Commands and
    # variables which aren't escaped with ';' will be interpreted before being
    # written to the file
    cat <<EOF >>${_command_file}
    #!/bin/ksh
    # Set the Siebel Environment Variables
    .${_install_root}/siebenv.sh
    # Test the state and report the result
    if srvredit -q -u ${_admin_id} -p ${_admin_id_password} -g ${_service_name} -e none -z -c
        \$Gateway.VersionString' > /dev/null; then
            printf \n            _status=0
        else
            printf \n            _status=1
    fi
    # Return the Gateway Status to the calling program
    exit \$_status
    EOF
    # Make the command file executable
    chmod 755 ${_command_file}

    # Execute the command file as the Siebel Admin User
    su - ${_admin_id} -c ${_command_file}
    _status=$?
    # Delete the command file
    rm ${command_file}
    # Return the result of the command file
    return $_status
} 

function start_siebel_gateway 
{
    # Unload parameters into more meaningful variables
    _install_root=$1
    _admin_id=$2

    if [ ! -f ${_install_root}/siebenv.sh ]; then
        printf \n        return 1
    else
        printf \n        # We'll use a temporary command file
        _command_file=tmp/ha_script.$$  
        if [ -f ${_command_file} ]; then
            rm ${_command_file}
        fi

        # Test the state and report the result
        if srvredit -q -u ${_admin_id} -p ${_admin_id_password} -g ${_service_name} -e none -z -c
            \$Gateway.VersionString' > /dev/null; then
                printf \n                _status=0
            else
                printf \n                _status=1
        fi
        # Return the Gateway Status to the calling program
        exit \$_status
        EOF
        # Make the command file executable
        chmod 755 ${_command_file}

        # Execute the command file as the Siebel Admin User
        su - ${_admin_id} -c ${_command_file}
        _status=$?
        # Delete the command file
        rm ${command_file}
        # Return the result of the command file
        return $_status
    }
# Put the code we need to start the Gateway into a command file. Everything
# between the 'cat' and the 'EOF' will be written to the file. Commands and
# variables which aren't escaped with '"' will be interpreted before being
# written to the file
cat <<EOF >>${_command_file}
#!/bin/ksh
#
# Set the Siebel Environment Variables
. ${_install_root}/siebenv.sh
#
# Change to the Siebel Gateway log folder (helps contain crash files!)
cd \${SIEBEL_ROOT}/log
#
# Start the Siebel Gateway
start_ns
#
# Return the Gateway Status to the calling program
exit \$?
EOF

# Make the command file executable
chmod 755 ${_command_file}
# Execute the command file as the Siebel Admin User
su - ${_admin_id} -c ${_command_file}
_status=$?
# Delete the command file
rm ${_command_file}
# Return the result of the command file
return ${_status}
fi

###
function stop_siebel_gateway
{

# Unload parameters into more meaningful variables
_install_root=$1
_admin_id=$2

if [ ! -f ${_install_root}/siebenv.sh ]; then
    printf 'Siebel Gateway Not Available!

    return 0
else
    printf 'Stopping Siebel Gateway

    # We'll use a temporary command file
    _command_file=/tmp/ha_script.$$ 
    if [ ! -f ${_command_file} ]; then
        rm ${_command_file}
    fi

    # Put the code we need to stop the Gateway into a command file. Everything
    # between the 'cat' and the 'EOF' will be written to the file. Commands and
    # variables which aren't escaped with '"' will be interpreted before being
    # written to the file
    cat <<EOF >>${_command_file}
    #!/bin/ksh
    #
    # Set the Siebel Environment Variables
    . ${_install_root}/siebenv.sh
    #
    # Change to the Siebel Gateway log folder (helps contain crash files!)
    cd \${SIEBEL_ROOT}/log
    #
    # Stop the Siebel Gateway
    stop_ns
    #
    # Return the Gateway Status to the calling program
    exit \$?
    EOF

    # Make the command file executable
    chmod 755 ${_command_file}
    # Execute the command file as the Siebel Admin User
    su - ${_admin_id} -c ${_command_file}
    _status=$?
    # Delete the command file
    rm ${_command_file}
    # Return the result of the command file
    return ${_status}
fi
###
chmod 755 ${_command_file}

# Execute the command file as the Siebel Admin User
su - ${_admin_id} -c ${_command_file}
_status=$?

# Delete the command file
rm ${_command_file}

# Return the result of the command file
return ${_status}

fi

#####################################################################
# Siebel File System Functions
#####################################################################
function check_siebel_filesystem
{
    # Unload parameters into more meaningful variables
    _siebelfs=$1
    _admin_id=$2

    printf '
Checking Siebel Filesystem\n'

    # We'll use a temporary command file
    _command_file=/tmp/ha_script.$$;
    if [ -f $_command_file ]; then
        rm $_command_file
    fi

    # Put the code we need to check the Filesystem into a command file. Everything
    # between the 'cat' and the 'EOF' will be written to the file. Commands and
    # variables which aren't escaped with '\' will be interpreted before being
    # written to the file
    cat <<EOF >>${_command_file}
    #!/bin/ksh
    _filename="${_siebelfs}/att/filesystem_ok"
    # Try to create a file in the Siebel File System
    touch \${_filename}
    # Check if we succeeded
    if [ -f \${_filename} ]; then
        printf '\nSiebel Filesystem Available\n'
        _status=0
    else
        printf '\nSiebel Filesystem Unavailable\n'
        _status=1
    fi

    # Remove the file we created
    rm \${_filename} 2>/dev/null

    # Return the App Server Status to the calling program
    exit \${_status}
EOF

    # Make the command file executable
    chmod 755 ${_command_file}

    # Execute the command file as the Siebel Admin User
    su - ${_admin_id} -c ${_command_file}
    _status=$?

    # Delete the command file
    rm $_command_file

    # Return the result of the command file
    return ${_status}
}

Listing 4 - Core shell script function used by the cluster scripts

Listing 5 shows the cluster script used to start up the Siebel Gateway Server.

#!/bin/ksh
#Backend monitoring script for Siebel Gateway
#Number: SiebelGatewaySTART
#Description: This script is called to start the Siebel Gateway
#Arguments: none
#Returns: 0 Gateway Started
#           anything else Gateway Not Started

function start_siebel_gateway
{
    # Unload parameters into more meaningful variables
    _siebelfs=$1
    _admin_id=$2

    printf '
Starting Siebel Gateway\n'

    # We'll use a temporary command file
    _command_file=/tmp/ha_script.$$;
    if [ -f $_command_file ]; then
        rm $_command_file
    fi

    # Put the code we need to check the Filesystem into a command file. Everything
    # between the 'cat' and the 'EOF' will be written to the file. Commands and
    # variables which aren't escaped with '\' will be interpreted before being
    # written to the file
    cat <<EOF >>${_command_file}
    #!/bin/ksh
    _filename="${_siebelfs}/att/gateway_started"
    # Try to create a file in the Siebel Gateway
    touch \${_filename}
    # Check if we succeeded
    if [ -f \${_filename} ]; then
        printf '\nSiebel Gateway Started\n'
        _status=0
    else
        printf '\nSiebel Gateway Not Started\n'
        _status=1
    fi

    # Remove the file we created
    rm \${_filename} 2>/dev/null

    # Return the result of the command file
    return ${_status}
EOF

    # Make the command file executable
    chmod 755 ${_command_file}

    # Execute the command file as the Siebel Admin User
    su - ${_admin_id} -c ${_command_file}
    _status=$?

    # Delete the command file
    rm $_command_file

    # Return the result of the command file
    return ${_status}
}

Listing 5 - Cluster script used to start up the Siebel Gateway Server.
# Load HA script functions
./powerha/HAScriptFunctions

cleanup_siebel_gateway /files/siebel/gtwysrvr sadmin
start_siebel_gateway /files/siebel/gtwysrvr sadmin

# Return exit code returned from function
exit $?;

Listing 5 - Cluster script to **start** a Siebel Gateway Server

Listing 6 shows the cluster script used to shut down the Siebel Gateway Server.

#!/bin/ksh

# Load HA script functions
./powerha/HAScriptFunctions

cleanup_siebel_gateway /files/siebel/gtwysrvr sadmin
stop_siebel_gateway /files/siebel/gtwysrvr sadmin

# Return exit code returned from function
exit $?;

Listing 6 - Cluster script to **stop** a Siebel Gateway Server

Listing 7 shows the cluster script used to monitor if the Siebel Gateway Server is running and available.

#!/bin/ksh

# Load HA script functions
./powerha/HAScriptFunctions

check_siebel_gateway /files/siebel/gtwysrvr siebel_ha_gateway sadmin sadmin

# Return exit code returned from function
exit $?;

Listing 7 - Cluster script to **monitor** a Siebel Gateway Server

Listing 8 shows the cluster script used to clean up a failed instance of the Siebel Gateway Server, in preparation for it to be restarted.

#!/bin/ksh

# Load HA script functions
./powerha/HAScriptFunctions

cleanup_siebel_gateway /files/siebel/gtwysrvr sadmin
start_siebel_gateway /files/siebel/gtwysrvr sadmin

# Return exit code returned from function
exit $?;

Listing 8 - Cluster script to **clean up** a Siebel Gateway Server
## Name: SiebelGatewayCLEANUP
## Description: This script is called to clean up a failed Siebel Gateway
## Arguments: none
## Returns: 0 Gateway Cleaned Up anything else Gateway Not Cleaned Up

`# Load HA script functions
./powerha/HAScriptFunctions

stop_siebel_gateway /files/siebel/gtwysrvr sadmin
cleanup_siebel_gateway /files/siebel/gtwysrvr sadmin

# Return exit code returned from function
exit $?`

Listing 8 - Cluster script to clean up a Siebel Gateway Server

Listing 9 shows the cluster script used to send out an appropriate notification message in the event of the Siebel Gateway Server experiencing an issue.

```bash
#!/bin/ksh

# Load HA script functions
./powerha/HAScriptFunctions

notify_message gateway

# Always return TRUE
exit 0
```

Listing 9 - Cluster script to send out a notification of a Siebel Gateway Server event

Listing 10 shows the cluster script used as a place holder for any possible Siebel File System startup commands. In this installation, there are no specific actions that need to be performed to start up a Siebel File System.

```bash
#!/bin/ksh

# Nothing to start here. Return TRUE
```

Listing 10 - Cluster script to start up a Siebel File System
Listing 10 - Cluster script to **start** a Siebel File System

Listing 11 shows the cluster script used as a place holder for any possible Siebel File System shutdown commands. In this installation, there are no specific actions that need to be performed to shut down a Siebel File System.

```bash
#!/bin/ksh
#####################################################################
## Name: SiebelFilesystemSTOP
## Description: This script is called to stop the Siebel File System
## Arguments: none
## Returns: 0 Filesystem Stopped
## anything else Filesystem Not Stopped
#####################################################################
# Nothing to stop here. Return TRUE
exit 0
```

Listing 11 - Cluster script to **stop** a Siebel File System

Listing 12 shows the cluster script used to monitor if the Siebel File System is available.

```bash
#!/bin/ksh
#####################################################################
## Name: SiebelFilesystemAPPMON
## Description: This script is called to check if the Siebel Filesystem is alive and well
## Arguments: none
## Returns: 0 Filesystem Available
## anything else Filesystem Not Available
#####################################################################
./powerha/HAScriptFunctions
check_siebel_filesystem /files2/siebelfs sadmin
# Return exit code returned from function
exit $?
```

Listing 12 - Cluster script to **monitor** a Siebel File System

Listing 13 shows the cluster script used as a place holder for any possible Siebel File System cleanup commands. In this installation, there are no specific actions that need to be performed to clean up a failed Siebel File System.

```bash
#!/bin/ksh
#####################################################################
## Name: SiebelFilesystemCLEANUP
## Description: This script is called to cleanup a failed Siebel Filesystem
## Arguments: none
## Returns: 0 Filesystem Cleaned Up
## anything else Filesystem Not Cleaned Up
#####################################################################
```
Listing 13 - Cluster script to clean up a Siebel File System

Listing 14 shows the cluster script used to send out an appropriate notification message in the event of the Siebel File System experiencing an issue.

Listing 14 - Cluster script to send out a notification of a Siebel File System event
Appendix 3: Siebel CRM Environments

siebelenv.sh file

```bash
#siebelenv.sh
if [ ! -d "$HOME" ]; then
  echo "ERROR: HOME must be set prior to starting the Siebel server!"
  exit 1
fi
SIEBEL_ROOT=/files/siebel/siebsrvr ; export SIEBEL_ROOT
SIEBEL_HOME=/files/siebel/siebsrvr ; export SIEBEL_HOME
SIEBEL_LIB_DIR=/files/siebel/siebsrvr/lib ; export SIEBEL_LIB_DIR
ORACLE_HOME=/files/oracle/product/11.2.0/client ; export ORACLE_HOME
MWHOME=${SIEBEL_ROOT}/mw ; export MWHOME
SQLANY=${SIEBEL_ROOT}/SYSBsa90 ; export SQLANY
SIEBEL_LANGUAGE=enu ; export SIEBEL_LANGUAGE
LANGUAGE=enu ; export LANGUAGE
PATH=${PATH}:${SIEBEL_ROOT}/bin:${MWHOME}/bin ; export PATH
PATH=${PATH}:${SQLANY}/bin
if [[ a${LIBPATH} = ${LIBPATH}a ]]
  then LIBPATH=${SIEBEL_ROOT}/lib:${MWHOME}/lib:${SQLANY}/lib:/usr/lib
else LIBPATH=${SIEBEL_ROOT}/lib:${MWHOME}/lib:${SQLANY}/lib:/usr/lib:${LIBPATH}
fi
export LIBPATH
AIXTHREAD_SCOPE=S ; export AIXTHREAD_SCOPE
AIXTHREAD_MUTEX_DEBUG=OFF ; export AIXTHREAD_MUTEX_DEBUG
AIXTHREAD_RWLOCK_DEBUG=OFF ; export AIXTHREAD_RWLOCK_DEBUG
AIXTHREAD_COND_DEBUG=OFF ; export AIXTHREAD_COND_DEBUG
#LDR_CNTRL=IGNOREUNLOAD ; export LDR_CNTRL
MWREGISTRY=${MWHOME}/system/registry.bin ; export MWREGISTRY
MWUSER_DIRECTORY=${MWHOME}/.mw/core_data ; export MWUSER_DIRECTORY
ODBCINI=${MWHOME}/.mw/system/registry.bin ; export ODBCINI
SIEBEL_GATEWAY=pspp_gateway:2320; export SIEBEL_GATEWAY
SIEBEL_UNIXUNICODE_DB=ORACLE; export SIEBEL_UNIXUNICODE_DB
export LIBPATH=${LIBPATH}:${ORACLE_HOME}/lib:${ORACLE_HOME}/network/lib
NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252; export NLS_LANG
LANG=EN_US.UTF-8; export LANG
MWOS=aix4; export MWOS
export EXTHRM=ON
export PATH=${PATH}:${ORACLE_HOME}/bin
export TNS_ADMIN=${ORACLE_HOME}/network/admin
export SIEBEL_OSD_MAXLIMITS=1
export SIEBEL_ASSERT_MODE=0
export SIEBEL_USE_BINDING=TRUE
export LDR_CNTRL=PREREAD_SHLIB@LOADPUBLIC@MAXDATA=0x50000000
export AIXTHREAD_MUTEX_FAST=ON
#
# Function that calculates PSPP OSD USERS based on Siebel Component Guidelines
#
function calculate_osd_users
{
  _targeted_users=$1
  _users_per_om=$2
  _tasks_per_om=$3
  _number_of_oms=$(print "scale=4;${_targeted_users}/${_users_per_om}\+0.9999" | bc)
  _number_of_oms=$(print "scale=0;\${_number_of_oms}\+1" | bc)
  _number_of_tasks=$(print "scale=0;\${_number_of_oms}\*\${_tasks_per_om}" | bc)
  print \"${_number_of_tasks}"}
#
# Calculate OSD USERS for each PSPP COMPONENT
#
# Parameters passed to function are: TARGETTED_USERS USERS_PER_OM TASKS_PER_OM
#
_cc123_users=$(calculate_osd_users 1750 150 170)
_sc3_users=$(calculate_osd_users 750 40 60)
```

Applying PowerVM to Oracle’s Siebel Customer Relationship Management (CRM)
http://www.ibm.com/support/techdocs
© Copyright 2012, IBM Corporation
_loyalty_users=$(calculate_osd_users 0 200 220)

**Modified AIX parameters**

```bash
_loyalty_engine_realtime_users=$(calculate_osd_users 0 200 200)
_loyalty_engine_interactive_users=$(calculate_osd_users 0 200 200)
_eai_users=$(calculate_osd_users 0 200 200)

export NUMBER_OF_USERS=$(print "scale=0;${_cc123_users}+${_sc3_users}+${_loyalty_users}+${_loyalty_engine_realtime_users}+
${_loyalty_engine_interactive_users}+${_eai_users}" | bc)
export SIEBEL_OSD_NLATCH=$(print "((7 * ${NUMBER_OF_USERS}) + 1000)" | bc)
export SIEBEL_OSD_LATCH=$(print "((1.2 * ${NUMBER_OF_USERS})/1" | bc)

info:
Description = "tunsave -F /tmp/tunsave.txt"
AIX_level = "6.1.6.15"
Kernel_type = "MP64"
Last_validation = "2012-07-17 16:54:48 PDT (current, reboot)"

sched:

vmo:
kernel_heap_psize = "65536"
nokilluid = "1"
kernel_psize = "65536" # RESTRICTED not at default value
mbuf_heap_psize = "65536" # RESTRICTED not at default value
vmm_mpsize_support = "2" # RESTRICTED not at default value

iio:
umclust = "128"
umfsbufs = "256"

raso:

no:
ipqmaxlen = "512"
rfc1323 = "1"
tcp_init_window = "3"
tcp_mssdflt = "1440"
tcp_nagle_limit = "1"
tcp_nodelayack = "1"
tcp_pmtu_discover = "0"
tcp_recvspace = "262144"
tcp_sendspace = "262144"
udp_pmtu_discover = "0"
udp_recvspace = "65536"
udp_sendspace = "32768"
net_malloc_police = "32768" # RESTRICTED not at default value

nfso:

Siebel configuration parameters

```
set server siebel_app1
change param DSMaxCachedCursors=0 for named subsystem ServerDataSrc
change param DSMaxCachedDataSets=0 for named subsystem ServerDataSrc
change param MaxTasks=500 for comp SRBroker
change param MinMTServers=1 for comp SRBroker
change param MaxMTServers=1 for comp SRBroker
change param MaxTasks=2 for comp SCBroker
change param DfltTasks=2 for comp SCBroker
change param ConnForwardAlgorithm=RR for comp SCBroker
change EVTLogLvl %=1 for comp SCBroker
enable comp group Fins
enable comp group Communications
disable comp group EAI
disable comp group LoyaltyEngine
disable comp group Loyalty
manual start comp SvtrTbCleanUp
manual start comp SvtrTaskPersist
manual start comp AdminNotify
manual start comp INSeServiceObjMgr_enu
manual start comp FINSConsoleObjMgr_enu
manual start comp FINSeBankingObjMgr_enu
```
Applying PowerVM to Oracle's Siebel Customer Relationship Management (CRM)
http://www.ibm.com/support/techdocs
© Copyright 2012, IBM Corporation

51

```plaintext
# Base Tuning Values...
# _component=FINSObjMgr_enu
# _targeted_users=4200
# _users_per_om=150
# _tasks_per_om=170
# _users_per_db_conn=10
# _sisnapi_conns_per_om=5
change param MaxTasks=4760 for comp FINSObjMgr_enu
change param MaxMTServers=28 for comp FINSObjMgr_enu
change param MinMTServers=28 for comp FINSObjMgr_enu
change param MaxSharedDbConns=420 for comp FINSObjMgr_enu
change param MinSharedDbConns=420 for comp FINSObjMgr_enu
change param MinTrxBDBCConns=420 for comp FINSObjMgr_enu
change param Number0fListRows=7 for comp FINSObjMgr_enu
change param EnableCDA=FALSE for comp FINSObjMgr_enu
change param SystemSWFName=CCHtmlType.swf for comp FINSObjMgr_enu
change EvtLogLvl %=0 for comp FINSObjMgr_enu
change EvtLogLvl EventContext=1 for comp FINSObjMgr_enu
change EvtLogLvl ObjMgrSqlLog=1 for comp FINSObjMgr_enu
change EvtLogLvl SQLParseAndExecute=1 for comp FINSObjMgr_enu

# Base Tuning Values...
# _component=eCommunicationsObjMgr_enu
# _targeted_users=1800
# _users_per_om=50
# _tasks_per_om=70
# _users_per_db_conn=10
# _sisnapi_conns_per_om=2
change param MaxTasks=2520 for comp eCommunicationsObjMgr_enu
change param MaxMTServers=36 for comp eCommunicationsObjMgr_enu
change param MinMTServers=36 for comp eCommunicationsObjMgr_enu
change param MaxSharedDbConns=180 for comp eCommunicationsObjMgr_enu
change param MinSharedDbConns=180 for comp eCommunicationsObjMgr_enu
change param SessPersSisnConn=35 for compdef eCommunicationsObjMgr_enu
change param eProdCfgSnapshotFlg=TRUE for comp eCommunicationsObjMgr_enu
change param eProdCfgRemote=FALSE for comp eCommunicationsObjMgr_enu
change param eProdCfgNumOfCachedObjects=10000 for comp eCommunicationsObjMgr_enu
change param eProdCfgNumOfCachedFactories=10 for comp eCommunicationsObjMgr_enu
change param eProdCfgNumOfCachedWorkers=15 for comp eCommunicationsObjMgr_enu
change param CompressHistoryCache=FALSE for comp eCommunicationsObjMgr_enu
change param Number0fListRows=7 for comp eCommunicationsObjMgr_enu
change param EnableCDA=FALSE for comp eCommunicationsObjMgr_enu
change param CommEnable=FALSE for comp eCommunicationsObjMgr_enu
change param EnableSIFocusTracking=TRUE for comp eCommunicationsObjMgr_enu
change param SystemSWFName=CCHtmlType.swf for comp eCommunicationsObjMgr_enu
change EvtLogLvl %=0 for comp eCommunicationsObjMgr_enu
change EvtLogLvl EventContext=1 for comp eCommunicationsObjMgr_enu
change EvtLogLvl ObjMgrSqlLog=1 for comp eCommunicationsObjMgr_enu
change EvtLogLvl SQLParseAndExecute=1 for comp eCommunicationsObjMgr_enu
```

Applying PowerVM to Oracle’s Siebel Customer Relationship Management (CRM)
http://www.ibm.com/support/techdocs
© Copyright 2012, IBM Corporation