Sun Fire™ 15K Open System Controller (OpenSC) White Paper

Jacob Lowman and Dan Anderson
Enterprise System Products

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 U.S.A.
650-960-1300

Part No. 816-3266-10
November 2001, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

CHAPTER 1

OpenSC

This paper describes how to set up and verify third-party software on system controller (SC) boards, creating an open system controller (OpenSC). The SC board hosts System Management Services (SMS) software that monitors and controls a Sun Fire™ 15K system. Applications that run on the OpenSC, other than the SMS software and the Solaris™ operating environment, are considered third-party software. Third-party applications are expected to be lightweight, such as monitoring and backup agents, and not to demand intensive system resources. Described here are the SMS resource requirements, the maximum permitted resource consumption for third-party software, and techniques to help ensure that SMS receives the resources it needs to function properly.

It is the responsibility of the customer (system administrator) to ensure that adequate OpenSC resources are available to support SMS and all other software packages that are being installed on the OpenSC board.

This paper is intended for system administrators, system engineers, and others who are installing and configuring a Sun Fire 15K system controller board with third-party applications.

SMS Functionality

SMS software runs on a system controller board (running the Solaris operating environment) dedicated to controlling and monitoring a Sun Fire 15K system. To function correctly, SMS must be able to respond in a timely manner to incoming events from the system. This includes responding to environmental conditions, such as over-temperature boards, software problems (panics), and hardware problems (arbstops). If SMS is delayed in responding, these events can be dropped or handled too late. Information about failures could be lost with no alerts or log trail. In a worst-case scenario, hardware can also be damaged due to a lack of timely action by SMS.

Impact of Third-Party Applications on the OpenSC

Traditionally, SMS software has been the only application allowed to run on an SC. This was required to prevent non-SMS software from interfering with the SMS critical mission of monitoring and controlling a Sun Fire 15K system. This is especially important for SMS software running on an older SC board, which may be slower or have less memory.

However, there has been increasing demand to run third-party agents on the SC. These agents gather information about the Sun Fire 15K system and SMS, and report back to a central server, allowing centralized control of multiple systems on the network. Backup servers are similarly structured, allowing centralized backup of multiple networked systems.

When running third-party software on an OpenSC, the primary goal must be non-interference with SMS software. SMS software normally requires few hardware resources, but when it needs these resources, they must be immediately available.

In order for non-SMS agents or other software to run on the OpenSC, you must take reasonable steps to ensure non-interference with the mission of the SMS software, which is monitoring and controlling the Sun Fire 15K system. These steps include:

- Applying recommended patches for the Solaris operating environment and SMS software on a regular basis
- Measuring the resource requirements of third-party software and considering *as a whole* the impact of all applications on the OpenSC.

  Software resource requirements usually describe system impact for the standalone use of an application. These requirements typically provide only generic recommendations such as "Software XYZ requires a system with a minimum of 128 Mbytes memory." Many recommendations do not mention the cumulative impact of multiple software packages running on a single system. In other words, you need to look at the whole forest, not just one tree.
- Planning for sufficient hardware resources for the SC board to run all SMS-resident software without slowing system response.

These steps are described in the following sections.

Recommended Patches

Sun produces two sets of patches for the Sun Fire 15K software:*Recommended and Security Patch Clusters for Solaris* and SMS patches.

The *Recommended and Security Patch Clusters for Solaris* contain Solaris software updates of universal interest for each version of the Solaris operating environment. These selected patches are important and highly recommended because they provide fixes for critical system and user- or security-related bugs. Some of these patches also fix performance problems. They are generally safe to apply, as opposed to higher-risk patches, or patches with new features, new drivers, or low-priority fixes, which are not included in these patch clusters. A prudent system administrator keeps systems current with the latest recommended patch level to protect against system problems.

An OpenSC should be regularly updated with all the SMS patches available for the particular release of SMS software used, except for special-case patches noted in the patch README file. These patches should be applied only on a case-by-case basis.

Both the *Recommended and Security Patch Clusters for Solaris* and the SMS patches are available at the SunSolve^{SM} web site, located at:
http://SunSolve.Sun.COM/.

Performance Improvements

SMS software has been made more robust so that third-party applications can be run on the OpenSC. The performance improvements include the use of real-time processes, which enable SMS to run with fewer resources and to function continuously, even when CPU and memory utilization on the OpenSC is high. These performance enhancements are available beginning with SMS 1.1.

Performance Improvements to the Solaris Operating Environment

The Solaris 8 software improvement most relevant to the OpenSC involves better thread handling. When a real-time SMS thread is blocked by a lower-priority thread, the kernel temporarily assigns a higher priority to the blocking thread in order to quickly complete and release the blocked resource. This results in faster SMS response time to Sun Fire 15K server events.

For details, see *What's New in the Solaris 8 Operating Environment* at `http://www.Sun.COM/software/solaris/`.

OpenSC Resource Requirements

The following table (continued on page 5) lists the SC minimum resource requirements, obtained from the *SMS 1.1 Installation Guide and Release Notes*, and the OpenSC resource requirements. OpenSC resource requirements are higher than the SC minimum requirements because of the extra load expected on the OpenSC.

TABLE 1-1 Required OpenSC Resources

Resource	SC Minimum Requirement	OpenSC Minimum Requirement
Solaris operating environment	Version 8 HW 10/01	Version 8 HW 10/01
System Management Services (SMS)	Version 1.1	Version 1.1
System Controller board	CP1500	CP1500
Disk space	3.0 Gbytes for the Solaris operating environment and SMS software	3.5 Gbytes for the Solaris operating environment and SMS software Additional 512 Mbytes or more disk space for third-party applications
Processor speed	440 MHz	440 MHz

TABLE 1-1 Required OpenSC Resources *(Continued)*

Resource	SC Minimum Requirement	OpenSC Minimum Requirement
CPU utilization	None	35% idle
Real memory	256 Mbytes	256 Mbytes
Swap space	2 Gbytes	2 Gbytes Memory requirements may exceed these minimum requirements, depending on the software used on the OpenSC. The amount of real memory required is usually 256 Mbytes. The swap space is usually 2 Gbytes. For details, see "Sizing OpenSC Memory."

Sizing OpenSC Memory

To measure the memory requirements of the OpenSC board, you must consider the cumulative requirements of all applications *as a whole*, not just the impact of an individual application. To do this, first determine the type of run-time environment involved by answering the following questions:

- What is the maximum number of domains running on the Sun Fire 15K system?
- Is Sun™ Management Center installed on the OpenSC?
- Are third-party applications running on the OpenSC?
  - If so, what are their virtual and real memory requirements?
  - What is the CPU overhead?

Calculating Memory Usage by Third-Party Applications

If the OpenSC is running third-party applications, determine how much virtual and real memory is used by these applications. This memory amount can usually be found in the installation or administrator guide for the application. If this information is not available, you can easily calculate it using the memory usage output from the `pmap` command. Use this command when the system is not thrashing (paging at a high rate) and the application is in an active running state, so that the command output shows how much resident memory the application

requires when it is active, but not thrashing. For information on determining whether a system is thrashing, see “Verifying That You Have Sufficient Real Memory” on page 10.

TABLE 1-2 explains the typographic conventions used inthis document.

TABLE 1-2 Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files.
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Replace command-line variables with real names or values.	To delete a file, type `rm` *filename*.

The following example shows how to size an application called CST (Configuration and Service Tracker), which has one process, `cstd`. (What this application does is not relevant here, as it serves only as an example of measuring memory usage.)

To obtain the memory information needed to calculate the memory usage for the application, type `pmap -x`, followed by the process ID of the application, as shown in the following example:

```
# pgrep cstd
406
# /usr/proc/bin/pmap -x 406
406:  /opt/SUNWcstv/bin/cstd -b
Address  Kbytes Resident Shared Private Permissions   Mapped File
. . .
total Kb  2848  2496  1400  1096
```

The last line in this example shows that 1096 Kbytes of resident private memory is being used. To obtain the virtual memory amount, subtract the shared memory from the total memory, then round up the resulting value. For instance, the virtual memory is 1448 Kbytes, which is derived by subtracting the shared memory (1400 kbytes) from the total memory (2848 Kbytes). Round up this value to 2 Mbytes. In this example, CST requires 1 Mbyte of resident memory and 2 Mbytes of virtual memory.

Some third-party applications have their own application-specific shared libraries. For these applications, you must add the real and virtual memory sizes of these libraries. The virtual memory used for these shared libraries is approximately the same as the shared library (*.so) file size. The resident memory used by shared libraries is also shown in the `pmap -x` command output.

Using the OpenSC Memory Worksheet

To calculate the virtual and real memory requirements for an OpenSC board, use the following OpenSC memory worksheet. TABLE 1-3 is an example of a completed worksheet, which contains sample entries in bold font. In all cases, supported OpenSC configurations must have at least the minimum amount of memory and other resources specified in TABLE 1-1,“Required OpenSC Resources” on page 4.

TABLE 1-3 OpenSC Memory Worksheet Example

Line	Item	Number	Real Memory (MB)	Virtual Memory (MB)	Real Memory Subtotal (MB)	Virtual Memory Subtotal (MB)
1	System	1	60	236	60	236
2	Base SMS	1	57	110	57	110
3	Domains (1-8)	**4**	3x no. of domains	7x no. of domains	**12**	**28**
4	Sun Management Center (0 or 1)	**0**	0 or 40	0 or 48	0	0
5	Third-party applications				0	0
6	Subtotal (lines 1 to 6)				**129**	**374**
7	Kernel buffer memory (Mbytes)	**256 MB RAM**	15% of RAM		38	
8	Recommended real memory (lines 7 and 8)				167	

TABLE 1-3 OpenSC Memory Worksheet Example *(Continued)*

Line	Item	Number	Real Memory (MB)	Virtual Memory (MB)	Real Memory Subtotal (MB)	Virtual Memory Subtotal (MB)
9	Reserved for /tmp/ in swapfs			1600		1600
10	Subtract amount of real memory					**-256**
11	Recommended swap space size			(Virtual memory subtotal – real memory total) + /tmp/ reserved		**1718**

The following procedure describes the steps to complete the worksheet for your OpenSC configuration. Each step also explains the sample entries, shown in bold font, made in the example worksheet. For information on how the predetermined values in the worksheet were derived, see "Calculating Memory Values" on page 23.

▼ To Complete the OpenSC Memory Worksheet

1. **In line 3:**

   a. **In the Number column, enter the highest number of domains (1 to 8) that you expect to have for your Sun Fire 15K system.**

   b. **Multiply the number of domains by 3 Mbytes and enter the result in the Real Memory Subtotal column.**

   c. **Multiply the number of domains by 7 Mbytes and enter the result in the Virtual Memory Subtotal column.**

   In line 3 of the example worksheet, 4 domains are specified, which results in 12 Mbytes for the Real Memory Subtotal and 28 Mbytes for the Virtual Memory Subtotal.

2. **If Sun Management Center is installed and running, enter the following on line 4:**

   a. **In the Number column, enter 1.**

   b. **In the Real Memory Subtotal column, enter 40 Mbytes.**

   c. **In the Virtual Memory Subtotal column, enter 48 Mbytes.**

   In line 4 of the example worksheet, Sun™ Management Center is not used, so 0 is entered in the Number, Real Memory Subtotal, and Virtual Memory Subtotal columns.

3. **In line 5, enter the real and virtual memory amounts required for any third-party applications that will be running on the OpenSC board. For details on estimating these memory requirements, see** "Calculating Memory Usage by Third-Party Applications" on page 5."

   In line 5 of the example worksheet, 0 is entered in the Real Memory Subtotal, and Virtual Memory Subtotal columns because no third-party applications are being used.

4. **In line 6, subtotal the values in the Real Memory Subtotal column and the Virtual Memory Subtotal column.**

   In line 6 of the example worksheet, the subtotal for the Real Memory Subtotal values is 129, and the subtotal for the Virtual Memory Subtotal values is 374.

5. **In line 7:**

   a. **In the Number column, enter the RAM that you will need. You must round this value up to the next 32 Mbyte increment. This number must be greater than 115 percent of the subtotal for the Real Memory entered in line 7. The Solaris operating environment uses 15 percent of the RAM for kernel buffer memory.**

   b. **In the Real Memory Subtotal column, enter 15 percent of the RAM specified in the Real Memory Subtotal column. This is the amount of buffer memory used by the kernel.**

   In line 7 of the example worksheet, 256 Mbytes of RAM is specified in the Number column, which is greater than the 129 Mbytes of Real Memory Subtotal entered in line 6. Also, 15 percent of 256 Mbytes of RAM yields 38 Mbytes of kernel buffer memory.

6. **In line 8, add the values from lines 6 and 7 and enter the resulting value in the Real Memory Subtotal column. If the value is less than 256 Mbytes, use the minimum amount of 256 Mbytes.**

   In line 8 of the example worksheet, adding 129 Mbytes and 38 Mbytes results in the minimum memory requirement of 167 Mbytes.

7. **Line 9 needs no entry.**

8. **In line 10, the Virtual Memory Subtotal column, enter the negative value of the RAM supplied in line 8.**

   In line 7 of the example worksheet, the RAM value is 256 Mbytes, so -256 is specified in the Virtual Memory Subtotal column in line 10.

9. **In line 11:**

   a. **Add the subtotals from lines 6 and 9 in the Virtual Memory Subtotal column, then subtract the virtual memory total (negative RAM) in line 10 from that amount.**

b. Enter the resulting value in the Virtual Memory Subtotal column. This number is the minimum swap space size needed by the OpenSC board.

In line 11 of the example worksheet, 374 Mbytes of virtual memory is added to 1.6 Gbytes (for swap), which results in 1974 Mbytes. The real memory value, 256 Mbytes, is subtracted from 1974 Mbytes, which yields a minimum swapfile size of 1718 Mbytes.

Note – The size limit for a swap partition is 2 Gbytes. However, you can add multiple swap partitions if needed.

Bottom line: The minimum amount of memory needed for a OpenSC board is 256 Mbytes. If you do not want to calculate the exact amount required, 256 Mbytes of memory is more than sufficient if you are using other monitoring software. Two Gbytes of swap space is more than sufficient for virtual memory and `swapfs` (`/tmp/`) space.

Verifying That You Have Sufficient Real Memory

Virtual memory consists of real memory (RAM) and page file (swap) space on disk. Unlike some other systems, real memory for the Solaris operating environment is not mirrored in a swap file. It is no longer necessary to duplicate a page of swap for each page of real memory, so the old rule that “swap space size should be twice real memory size” no longer applies. The only swap space required is the amount of virtual memory that exceeds the real memory for your system.

The amount of virtual memory required depends on the *working set model* for a process. The working set is the set of pages a process needs to work effectively. A working set needs to be in real memory or the program may thrash. Thrashing occurs when there is insufficient real memory for all the working sets of a process. As a result, the system spends an excessive amount of time paging the process working sets in and out of swap space.

The working set for a program is defined as $W(t, \omega)$, which is the set of pages referenced from time $(t - \omega)$ to time t. Typically, a working set for a program does not change much over time, although it can change drastically on occasion. Increasing the time period, ω, does not have much effect on the working set. Pages currently in use are likely to be used in the near future. Memory outside the working set is rarely, if ever, used. Therefore, a program that uses only its working set in real memory and the remainder in swap space will perform almost as effectively as if all

of its pages were in memory. This is true even though disk access time is about 100,000 times slower than RAM access time (about 10,000,000 nanoseconds versus 100 nanoseconds).

However, if there is insufficient real memory to keep the working set for a process in memory, the process can easily thrash and run more slowly. Running fewer processes or adding more real memory keeps the process working set in memory and stops the process from thrashing. Thrashing can affect the SMS ability to handle events in a timely manner, due to timeouts and lost interrupts. Prevent thrashing by properly sizing the system for all applications that it runs.

How do you know if a system is thrashing? The easiest way is to check the paging scan rate (sr). The kernel for the Solaris operating environment uses a page scanner, which scans a circular list of pages in memory in order to reclaim memory and swap it out to disk. Pages not referenced since the last cycle are paged out of memory. The scanner runs faster when demand for memory increases. If the demand is too high, memory in a working set for a process can be removed from real memory, which slows those processes. This can prevent SMS processes from reacting quickly to real-time events. Also, as its scan rate increases, the page scanner uses more CPU time.

If you suspect the system is thrashing, use the `vmstat` command to sample and display virtual memory statistics. This command adds little overhead and can safely run for long periods of time, if required. To use this command, type `vmstat` followed by the number of times you want it to sample, optionally followed by the frequency to sample, in seconds.

For example,

```
% vmstat 5
```

prints results every five seconds, while

```
% vmstat 3 100
```

prints results every three seconds for 100 times.

Review the `sr` column in the output displayed by `vmstat`. Ignore the entries in the first row, as the values are cumulative based on when the system was booted. If subsequent values in the `sr` column are non-zero, the system is thrashing. Ignore the `po` (page-out) column, as those values includes `swapfs` (swap file system or `/tmp/`) activity.

The following example shows the `vmstat` output for a system that is thrashing:

```
% vmstat 3
procs    memory              page                 disk         faults       cpu
r  b w swap    free re   mf  pi  po  fr  de    sr  dd f0 s0 --  in  sy   cs  us sy id
0  0 0 669856 7336  41 233   5   5   5  136   0   4  0  0  0 234 3107 1459 5  8  87
3  4 0 597232 2136 131 717 354 317 533 128 302  48  0  0  0 399 4889 2252 65 33  2
7  1 0 597120 3408 175 745 197 133 218   0  137  61  0  0  0 430 4757 2130 67 33  0
11 0 0 595832 2456 145 757 184 221 376 424 272  26  0  0  0 378 5235 2380 65 35  0
```

In the output above, note the `sr` column but ignore the first entry. The `sr` values are 302, 137, and 272, which indicate the system is thrashing heavily.

The next example shows a system that is not thrashing. The values in the `sr` column are zero, indicating that there is no excessive page scanning by the Solaris kernel to free pages:

```
% vmstat 3
procs   memory            page            disk          faults      cpu
r b w  swap free  re  mf pi po fr de  sr dd f0 s0 --  in  sy   cs  us sy id
0 0 0 672728 8376 41 236  4  5  5  0   0  4  0  0  0 243 3358 1585  6  9 85
0 0 0 672472 6960 29  46  0  0  0  0   0  0  0  0  0 239 3858 1924  8  4 88
0 0 0 672488 6992 59 374  0  0  0  0   0  0  0  0  0 237 4215 1933  5 11 83
0 0 0 666968 7248 87 811  2  0  0  0   0 13  0  0  0 266 4971 1938 24 29 47
0 0 0 672520 7200 47 176  0  0  0  0   0  0  0  0  0 292 4043 2043  8  6 86
0 0 0 672520 7200  0   0  0  0  0  0   0  0  0  0  0 240 3516 1861  4  0 96
0 0 0 672520 7200 31  74  0  0  0  0   0  0  0  0  0 235 3726 1876 14  4 82
```

CPU Utilization

When some people notice a system with low CPU utilization, for example 25 percent, they might ask, “Why is this high-performance system not doing anything? Should we use a lower-end system or find more work for it to do?” (Forget for the moment that the OpenSC is required to perform critical real-time tasks to manage the Sun Fire 15K platform)

Surprisingly, the answer to the latter question may be “no.” For a batch-processing system, where response times are not as critical, high utilization is usually preferred. However, for an interactive or a real-time controlling system such as the OpenSC, response time is more critical than high CPU utilization. High utilization leads to slower response time, as noted in queuing theory. As utilization approaches 100 percent, the wait time increases exponentially.

Queuing theory uses models to predict utilization and wait time for a client/server system. The operations in a retail bank, hospital emergency room, or computer server are examples of a client/server system. One of the basic assumptions in queuing theory is that the arrival time between two customers is an exponential distribution. In other words, long periods between customer arrivals are more unlikely than short periods.

Queuing theory is best illustrated by an example. Assume there is a small town bank, the Bank of Ethel, which has one teller (Ethel), and several customers. Ethel's utilization (how busy she is) can be determined by using the following formula for a single-server model (one teller):

$$\rho = \frac{\lambda}{\mu} < 1$$

where ρ (rho) is the proportion of time the servers (tellers) are busy (on a scale of 0 to 1.0, where 0 is no customers at all and 1 indicates the server is completely busy), λ (lambda) is the mean arrival rate, and μ (mu) is the mean service rate per server.

In this example, if two customers come to the Bank of Ethel every hour and Ethel serves an average of six customers an hour, $\lambda = 2$, $\mu = 6$, and Ethel's utilization is: $\rho = 2 / 6 = 1/3$, or about 0.33. Multiply ρ by 100 to convert ρ to percent, for example, 33 percent.

The number of customers expected in the bank at any one time is:

$$L = \frac{\lambda^2}{\mu(\mu - \lambda)}$$

In the above example, the expected number of customers is $L = 2^2 / 6(6 - 2) = 1/6$ or about 0.17. That is, on average there will be about 0.17 customers at the Bank of Ethel, which is not very busy.

What if the number of customer arrivals increases from two to five per hour? Then $\lambda = 5$, and the utilization will be $\rho = 5/6$ or about 0.83. This means that Ethel will be serving a customer 83 percent of the time. However, this has a drastic effect on L, the expected number of customers in the bank. In this example, $L = 5^2/6(6 - 5) = 25/6$ or about 4.17 customers in the bank, on the average. The number of customers waiting for service is (L - 1), so in this example, there are about 3.17 customers waiting for service. This example shows why high utilization and immediate service are not possible at the same time.

For an OpenSC board, assume the CPU is the server. Utilization, ρ, is the percentage that the CPU is in use. The CPU “customers” are processes that are either being serviced by the CPU or waiting in the queue. The *load average* shown by various commands in the Solaris operating environment is (L - 1), which represents the number of waiting customers.

FIGURE 1-1 illustrates how increased utilization drastically increases customer wait time.

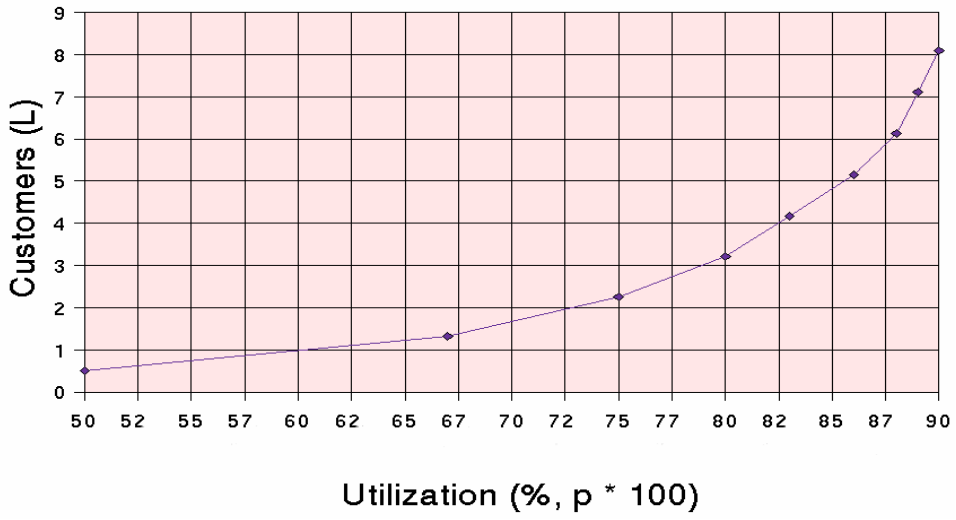

FIGURE 1-1 Utilization and Customers Expected

The X-axis is the utilization of a single server, ranging from 50 percent to 90 percent. The Y-axis shows L, the expected number of customers at any one time for a given level of utilization, ρ. If L is greater than or equal to 2, at least one customer is always waiting. At 60 percent utilization or less, almost no one is waiting for service. When utilization exceeds 72 percent or so, a customer is almost always waiting. When utilization exceeds 80 percent, multiple customers are usually waiting. In conclusion, to have quicker service, you must sacrifice high utilization. The highest utilization you can have without having customers wait for service is usually about 65 percent.

SMS CPU Requirements

Average CPU utilization should be under 65 percent, as explained in the previous section. This amount allows sufficient CPU resources for SMS software to immediately handle error conditions on a Sun Fire 15K system. The following table shows the approximate utilization currently used by SMS software at its busiest state (bringup of domains):

TABLE 1-4 CPU Utilization for SMS Software

Domains	Sun CP1500 Average CPU Usage
1 to 8	50.00%

If you are using the Sun Management Center, add the CPU overhead values in TABLE 1-5:

TABLE 1-5 Additional CPU Utilization for Sun Management Center

Domains	Sun CP1500 Average CPU Usage
1 to 8	15.00%

Due to the requirement of a high level of reliability and serviceability of SMS software running on the OpenSC, we currently do not recommend the use of third-party applications with nine or more domains. Consult your Sun service representative for alternatives available outside of this white paper.

CHAPTER 2

OpenSC Board Hardware

The OpenSC board must have enough hardware resources to operate the maximum number of domains planned on the Sun Fire 15K system it controls, as well as what is required to run third-party applications.

The proper hardware technology depends on the usage profile of the third-party software. Less intensive software requires less CPU and memory than resource-hungry applications. You must determine the CPU and real memory resources needed for these applications before determining the appropriate hardware to use.

Modification or upgrade of the OpenSC board by a customer is strongly discouraged. Consult a Sun service representative for OpenSC upgrade procedures and the hardware options currently available.

Using a Faster Processor

If CPU utilization is too high for your OpenSC configuration (over 65 percent), using a faster processor will reduce CPU utilization dramatically. A faster processor is also appropriate if you add monitoring or third-party software, such as Sun Management Center, which uses a lot of CPU.

When comparing CPUs, remember to consider the CPU family and L2 cache size, in addition to CPU speed. For example the UltraSPARC™ II processor is faster than the UltraSPARC™ IIi processor. The UltraSPARC II processor comes with more L2 cache (1 to 4 Mbytes) than the UltraSPARC IIi processor (256 Kbytes to 2 Mbytes).

Modification or upgrade of the OpenSC board by a customer is strongly discouraged. Consult a Sun service representative for OpenSC upgrade procedures and the hardware options currently available.

Adding More Memory

If a system does not have sufficient memory, it will thrash—that is, swap pages excessively in and out of real memory. Having sufficient memory modules will prevent thrashing and enable software to run with fewer interruptions on the OpenSC board.

Adding More Swap Space

Adding more swap space if needed will improve system reliability, even though it will not improve system performance. If a system runs out of memory, processes cannot allocate more memory and they will begin to fail. Swap space is required to save inactive processes and memory regions, and to handle overflow in `swapfs` (`/tmp/`). Additionally, automatic SMS failover may need to occasionally propagate large files. Therefore, it is important to have an adequately-sized swap file to hold these files.

Adding More Disk Space

The SMS software and Solaris operating environment require 3.0 Gbytes of unused disk space. If all disk resources are consumed, the OpenSC can exhibit strange behavior, such as freezing, respawning processes, or login failures, and event information might not be saved in logs. A third-party application can require additional disk space for the software as well as any additional data it could log to the disk. Remember to allocate this additional space when configuring the partitions. The minimum additional disk space recommended for a third-party application is 512 Mbytes. A third-party application can require more disk space than the minimum requirements. It is important evaluate the requirements of the third-party application and size the partition appropriately.

CHAPTER 3

Conclusion

The OpenSC board has the critical job of monitoring the Sun Fire 15K system so that it stays up and running. Installing third-party software requires extra consideration to help ensure that these applications do not affect the proper operation of the OpenSC.

Installing third-party software on the OpenSC requires you to perform the following steps:

- Determine that your OpenSC meets the minimum hardware and software requirements for OpenSC. These requirements are generally greater than the SC minimum requirements.
- Estimate the amount of real memory and swap space required by your SMS and third-party software. This amount can be greater than the OpenSC minimum requirements.
- Verify that your CPU utilization is not too high. If it is, consider using a faster CPU or a more powerful system controller board.

Modification or upgrade of the OpenSC board by a customer is strongly discouraged. Consult a Sun service representative for OpenSC upgrade procedures and the hardware options currently available.

APPENDIX **A**

Memory

This section contains a blank OpenSC memory worksheet and describes how to to determine the memory requirements for your SMS configuration.

Blank OpenSC Memory Worksheet

TABLE A-1 is provided on the following page, for your convenience, to use in calculating OpenSC memory.

TABLE A-1 Blank OpenSC Memory Worksheet

Line	Item	Number	Real Memory (MB)	Virtual Memory (MB)	Real Memory Subtotal (MB)	Virtual Memory Subtotal (MB)
1	System	1	60	236	60	236
2	Base SMS	1	57	110	57	110
3	Domains (1-8)		3x no. of domains	7x no. of domains		
4	Sun Management Center (0 or 1)		0 or 40	0 or 48		
5	Third-party applications					
6	Subtotal (lines 1 to 6)					
7	Kernel buffer memory (Mbytes)		15% of RAM			
8	Recommended real memory (lines 7 and 8)					
9	Reserved for /tmp/ in swapfs			1600		1600
10	Subtract amount of real memory					
11	Recommended swap space size			(Virtual memory subtotal – real memory total) + /tmp/ reserved		

Calculating Memory Values

The Blank OpenSC Memory Worksheet (TABLE A-1) is used to size virtual and real memory for an OpenSC board, based on the number of Sun Fire 15K domains controlled by the OpenSC and the number of SMS, Sun Management Center, and third-party applications involved.

This section provides details on how the predetermined memory values in the worksheet are derived. The `pmap` command provides output that is used to calculate the following memory values in the worksheet.

- Process private resident memory
- Process private virtual memory
- System memory (line 1 of the worksheet)
- Base SMS memory (line 2 of the worksheet)
- Each domain SMS memory (line 3 of the worksheet)
- Sun Management Center memory (line 4 of the worksheet)
- Kernel buffer memory (line 7 of the worksheet)
- `swapfs` (line 9 of the worksheet)

The following sections describe how each of these values is calculated.

Process Private Resident Memory

Use the `pmap -x` *processID* command to obtain the amount of private resident memory used by a process, where *processID* is the number of the process ID (for example, 406). In the output displayed, note the `Private` column in the last line. This value is the total private memory, in kilobytes, for a process that is resident in real memory.

Process Private Virtual Memory

To find the approximate amount of private virtual memory that a process uses, run the `pmap -x` *processID* command. In the output displayed, locate the total resident shared memory (shown on the last line under the `Shared` column) and subtract it from the total process memory (shown on the last line under the `Kbytes` column).

The resulting value is the approximate total private virtual memory for a process. It is slightly higher than the actual amount needed, as it includes nonresident shared memory (paged-out memory that is also used by other processes). The resulting value is a reasonable measurement of the private virtual memory required, which does not underestimate the required memory.

For further information on memory sizing, see Richard McDougall's paper, "*The Solaris Memory System: Sizing, Tools and Architecture.*"

System Memory (Line 1)

System memory usage consists of memory used by system processes, system shared libraries, CDE libraries, and CDE processes. TABLE A-2 shows the values for these items and the resulting totals for system memory: 60 Mbytes of real memory and 236 Mbytes of virtual memory. (The kernel buffer memory used is discussed in a subsequent section.)

TABLE A-2 System Memory Values

Memory	Resident Memory (Mbytes)	Virtual Memory (Mbytes)	Description
System processes	21	56	Use pmap -x to find the private resident and virtual memory of root processes, that is /usr/lib/*, /usr/sbin/*, /usr/sadm/*, rpc.*, and mibiisa.
System shared libraries	15	25	The total resident and virtual memory for system shared libraries, which are under /usr/lib/, is approximately 15 and 25 Mbytes (see McDougall, pp. 34-36).
CDE libraries	5	10	Use pmap -x to find the highest shared memory among /usr/dt/bin/* and /usr/openwin/bin/* processes.
CDE processes	19	145	Use pmap -x to find the private resident and virtual memory of Xsun, speckeysd, fbconsole, sdt_shell, ttsession, and dt* processes.
System subtotal	60	236	

Base SMS Memory (Line 2)

Base SMS memory consists of memory used by the OpemSC shared libraries and SMS platform-wide processes. TABLE A-3 shows the subtotals for base SMS memory: 57 Mbytes of real memory and 110 Mbytes of virtual memory.

TABLE A-3 Base SMS Memory Values

Memory	Resident Memory (Mbytes)	Virtual Memory (Mbytes)	Description
SMS shared libraries	8	10	Add the file sizes of the *.so files under /opt/SUNWSMS/lib/ This gives the approximate virtual memory used by the OpenSC libraries.
SMS platform software	49	100	Use `pmap -x` to find the private resident and shared memory of SMS platform-wide processes. That is, `hwad`, `dsmd`, `esmd`, `fomd`, `mld`, `mand` `frad`, `tmd`, `pcd`, `kmd`, `osd`, `efe`, `codd` and `ssd`.
SMS platform subtotal	57	110	

Each Domain SMS Memory (Line 3)

Each domain on the Sun Fire 15K system requires two processes on the OpenSC:

- `dxs`
- `dca`

Output from the `pmap -x` command shows that these two processes take 3 Mbytes of private resident memory and 7 Mbytes of private virtual memory.

Sun Management Center Memory (Line 4)

On the OpenSC, Sun Management Center requires three agents to send information back to the Sun Management Center server. One agent monitors the OpenSC board itself, while the other two agents monitor the Sun Fire 15K platform. Sun Management Center appears in the `ps` command as three processes, all named `esmd`.

TABLE A-4 presents the output from the `pmap -x` command for the Sun Management Center processes and the resulting subtotals for resident and virtual memory:

TABLE A-4 Sun Management Center Memory Values

Memory	Resident Memory (Mbytes)	Virtual Memory (Mbytes)
Sun Management Center host agent	11	15
Sun Management Center platform agent	25	27
Sun Management Center config reader	4	6
Sun Management Center subtotal	40	48

Due to the large footprint of the Sun Management Center server or console, running either on the OpenSC is not recommended.

Kernel Buffer Memory (Line 7)

The following quote from page 37 of Richard McDougall's paper, *The Solaris Memory System: Sizing, Tools and Architecture*, provides a general rule of thumb for determining kernel RAM:

> The amount of memory that the kernel uses varies significantly, based on the size of the tunable parameters. A lot of the tunable parameters are set at boot in proportion with the amount of physical RAM in the system. As a general rule of thumb, if all of the parameters are standard, you can allow about 15 percent of physical RAM for the kernel.

`swapfs` (Line 11)

The `swapfs` file system implements a file system in virtual memory, using both RAM and, if necessary, swap space, to store files. However, because this memory is not preserved across reboots, `swapfs` is used only for temporary files, usually `/tmp/`. SMS1.1 automatic failover uses `/tmp/` to propagate modified files to the other OpenSC in a dual configuration, mainly log files and recently changed configuration files, as part of the process of failing over to the other OpenSC. It is important to allocate enough space for `swapfs`, as log files can become quite large. Otherwise, the OPenSC might behave strangely or even fail. Allocating 2 Gbytes of

swap space for `/tmp/` should be sufficient for SMS 1.1 automatic failover. This amount is in addition to the amount of required virtual memory that exceeds available RAM.

If the system is experiencing or has experienced an unusually high level of activity, it is important to monitor the file system. This is especially important for `/var` on which the post and log files are located. It is important for the health of the system to maintain a tidy disk. Although SMS regulates some of its own files, third-party applications will not normally be supervised by SMS software. In rare instances, a file could grow to many megabytes in size. It is important to trim or remove these file so that they will not interfere with normal SMS operation.

Swap space is also used as reserve memory to help protect against "memory leaks." Memory leaks occur from software allocating memory but not freeing it after it is no longer needed. This is usually most noticeable during unusual and long-lasting error conditions.

References

Sun reference manuals are available at `http://docs.sun.com/`.

1. “Managing Quotas (Tasks),” *System Administration Guide, Volume II*, part number 805-7229-10 (2000).

   Provides information on managing user quotas on file systems.

2. Sun performance information Web page at `http://www.Sun.COM/sun-on-net/performance.html`

   Provides links to papers and books on Sun performance.

3. *Sun Performance and Tuning: Java and the Internet* by Adrian Cockcroft & Richard Pettit, 2nd ed. (Prentice Hall, 1998; ISBN 0-13-95249-4).

   Practical book by two Sun engineers on how to measure Solaris operating environment and application performance, and how to improve it.

4. “The Solaris Memory System: Sizing, Tools and Architecture,” by Richard McDougall (rev. D, May 1998), at `http://www.Sun.COM/sun-on-net/performance/vmsizing.pdf`

   Offers practical information on measuring system and application memory usage.

5. *Configuration and Capacity Planning for Solaris Servers* by Brian Wong (Prentice Hall, 1997; ISBN 0-13-349952-9).

Chapter 9 has practical information on virtual memory, file systems, and scheduling. Other chapters cover real-life examples on various classes of servers.

6. *Solaris Internals: Core Kernel Components* by Richard McDougall & Jim Mauro (Prentice Hall, 2000; ISBN 0-13-022496-0).

   Chapter 5, "Solaris Memory Architecture," includes information on Virtual Memory and the page-scanning algorithm. See `http://www.Sun.COM/books/`

7. Solaris Resource Manager (SRM) Reserves or limits CPU and other system resource usage for a set of processes.
   - Solaris Resource Manager (SRM) Web page `http://www.Sun.COM/solaris/ds/ds-resourcemgr/`
   - *Solaris Resource Manager 1.2 System Administration Guide*, part number 806-4053-10 (2000).
   - *Solaris Resource Manager 1.2 Reference Manual*, part number 806-4777-10 (2000).
   - *Solaris Resource Manager 1.2 Release Notes*, part number 806-4778-10 (2000).
   - *Solaris Resource Manager 1.2 Installation Guide*, part number 806-4776-10 (2000).

8. System Management Services (SMS) 1.1
   - *System Management Services (SMS) 1.1 Installation Guide and Release Notes*, part number 816-0901-10.
   - *System Management Services (SMS) 1.1 Administrator Guide*, part number 816-0899-10.
   - *System Management Services (SMS) 1.1 Reference Manual*, part number 816-0900-10.
   - *Sun Management Center 3.0 Release Notes for Sun Fire 15K Systems*, part number 806-6825-05 (2001).
   - *Sun Management Center 3.0 Release Notes for Sun Fire 15K Systems*, part number 806-6825-05 (2001).

9. *OpenSSP White Paper*, by Dan Anderson, part number 806-7352-10 (2000).

   Describes OpenSSP for the Sun Enterprise 10000 system.

10. *Securing the Sun Fire 15K System Controller*, by Alex Noordergraaf and Dina Kurktchi, part number 816-2722-10 (2001).

    Describes security for the Sun Fire 15K system controller.