

WEB CONSOLIDATION ON THE SUN FIRE™



© 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, Sun Fire, Solaris, CoolThreads and SunDocs are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Table of Contents

Web Consolidation on the Sun Fire™ T1000 Server using Solaris™ Containers	1
Technology Under Evaluation	1
Sun Fire T1000 Server	1
Solaris Containers	2
Web Serving Environment	2
Web Consolidation using Solaris Containers	3
Configuring the Resource Pools	4
Configuring the Zones	7
Monitor and Manage Zones and Resource Pools	11
Building and Configuring the Apache Web Server	13
Evaluation of Consolidated Web Server Environment	14
Testing Scenario	15
Results	15
Conclusions	15
About the Author	16
Acknowledgements	16
References	16
Ordering Sun Documents	16
Accessing Sun Documentation Online	16

Web Consolidation on the Sun Fire™ T1000 Server using Solaris™ Containers

Reducing the costs of IT infrastructure and improving the manageability and efficiency of web services pose significant challenges for many organizations in today's economic climate. Recent studies from IDC[5] describe the challenges IT managers face administering the proliferation of x86-based servers used to run web services applications. The reports from those studies reveal that using large number of x86-based systems can increase space and power consumption, as well as cost and asset management overhead. In addition, many of these x86-based systems run a mixture of operating system and application software leading to increased management complexity and potential security concerns.

Faced with these challenges, many organizations are attracted by the idea of consolidating web and application services from multiple x86-based servers to a smaller number of high-performance servers. This approach strives to help simplify management, improve performance, and increase the efficiency of delivering web services. The combined capabilities of the Sun Fire™ T1000 server and Solaris™ Containers technology in particular offer significant promise as a web-tier consolidation platform. The Sun Fire T1000 server offers high aggregate throughput performance in a small, power-efficient footprint. Solaris containers provide a complete, isolated, and secure runtime environment for applications, enabling multiple web servers to safely and efficiently run on the same platform.

This paper explores the configuration and testing of the Sun Fire T1000 server as a web-tier consolidation platform. It discusses methodologies used to consolidate multiple web servers onto a single Sun Fire T1000 server, and explains the steps used to configure the Solaris Containers. In addition, to determine the effectiveness of this approach, testing was performed to evaluate the consolidated Sun Fire T1000 system against a baseline configuration of current Xeon servers, a popular choice as web server platform.

Technology Under Evaluation

Effective web-tier consolidation requires a range of technologies working together. This article evaluates the use of the Sun Fire T1000 server based on the UltraSPARC® T1 processor with CoolThreads™ technology, Solaris Containers to safely partition web server instances, and the open-source Apache web serving environment.

Sun Fire T1000 Server

The Sun Fire T1000 server is a high density compute server platform based on Sun's UltraSPARC T1 processor. The major benefits of this platform are high aggregate throughput performance with very efficient power, cooling and space consumption.

Sun's innovative UltraSPARC T1 processor leverages chip multithreading (CMT) technology to both drastically improve computational density and greatly reduce power density. The UltraSPARC T1 processor implements CMT technology by combining chip multi-processing and hardware multi-threading with an efficient instruction pipeline. The result is a processor that provides a thread-rich environment that can effectively mask memory access latencies and provide improved scalability for many applications.

This new processor hardware architecture has the following features in a single chip package:

- Each physical chip contains eight cores (individual execution pipelines).
- Four threads per core provide a total of 32 active thread contexts.
- Each core has separate Level 1 Instruction and Data caches shared by the four threads.
- All eight cores share a unified Level 2 cache on chip.
- Memory is unified to provide low latency to all cores.
- The processor is fully SPARC V7, V8, and V9 binary compatible.

In addition to the UltraSPARC T1 processor, the Sun Fire T1000 platform supports up to 16 Gbytes of DDR2 main memory, four 1000 baseT onboard network interfaces, and an 80 Gbyte SATA disk drive. With a 1U rack server footprint and 300 Watt power supply, the Sun Fire T1000 server provides a high performance low-power alternative to many other systems.

Solaris Containers

Solaris Containers consist of a group of technologies that work together to efficiently manage system resources, virtualize the environment, and provide a complete, isolated, and secure runtime environment for applications. Solaris containers include two important technologies: Solaris Zones partitioning technology and resource management tools. Solaris Zones enable an administrator to create separate environments for applications on a single system, while the resource management framework allows for the allocation, management, and accounting of system resources such as CPU and memory.

- *Solaris Zones*

New to the Solaris 10 Operating System is a unique partitioning technology called Solaris Zones that can be used to create an isolated and secure environment for running applications. A zone is a virtualized operating system environment created within a single instance of the Solaris Operating System. Zones can be used to isolate applications and processes from the rest of the system. This isolation helps enhance security and reliability since processes in one zone are prevented from interfering with processes running in another zone.

- *Resource Management (Processor Sets and Dynamic Resource Pools)*

Resource management tools provided with the Solaris Operating System help enable system resources such as CPU resources to be dedicated to specific applications. CPUs in a multiprocessor system can be logically partitioned into processor sets and bound to a resource pool, which in turn can be assigned to a Solaris zone. Resource pools provide the capability to separate workloads so that consumption of CPU resources do not overlap, and also provide a persistent configuration mechanism for processor sets and scheduling class assignment. In addition, the dynamic features of resource pools enable administrators to adjust system resources in response to changing workload demands.

Web Serving Environment

Web serving application software and deployments can vary greatly depending on the requirements for the specific environment. A variety of commercial and open-source web servers are used in many organizations. Generally available research shows the Apache 1.3.33 web server having the largest market share today of all web servers[1].

Although both Apache 1.3.33 and 2.0 are bundled with the Solaris Operating system, this consolidation project used the Apache 1.3.33 web server source kit downloaded directly from the Apache Software Foundation.¹ This open-source kit was selected so that the same code could be used on both the Sun Fire T1000 server and on x86-based systems running the Linux operating system.

Web Consolidation using Solaris Containers

Solaris Containers provide a logical mechanism that can be used to consolidate multiple web servers onto a single platform. By hosting web servers in zones, they can be isolated to help ensure that web transactions in one zone are secure from web transactions executed by another web server. In addition, CPU resources can be managed using dynamic resource pools to prevent CPU-intensive web operations on one web server from impacting others.

This paper describes a particular methodology using zones and resource pools to consolidate multiple web servers on a single Sun Fire T1000 server. For this project, three unique zones were created and each zone was configured to execute a separate instance of the Apache web server (see Figure 1). In addition, each instance of the Apache web server was configured to use a separate network interface on the Sun Fire T1000 server. For example, the web server instance running in the first zone utilized the bge1 interface, while the web server instance running in the third zone utilized bge3.

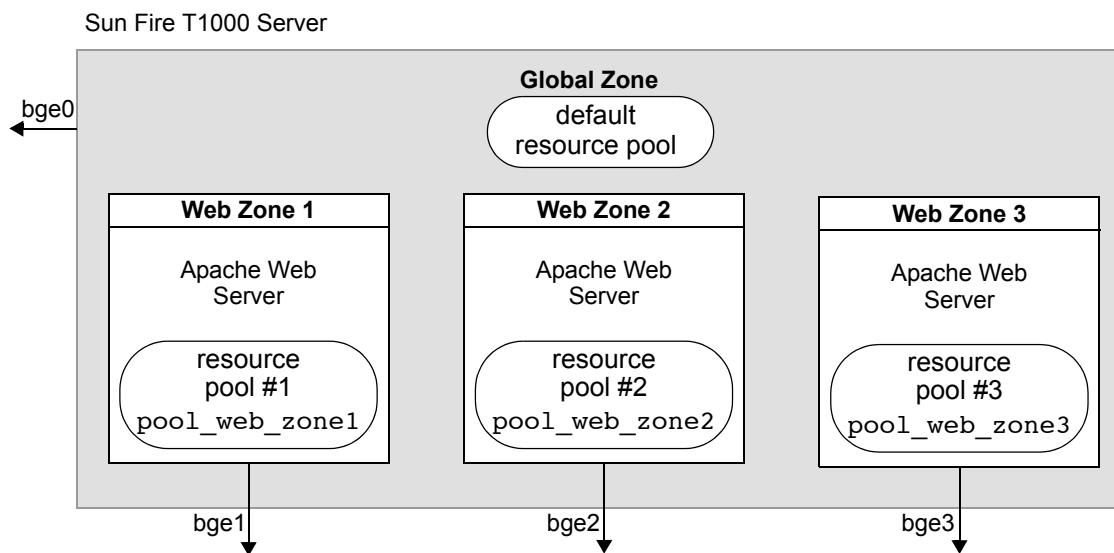


Figure 1. Web Consolidation Topology on the Sun Fire T1000 Server.

While Solaris zones provide a virtual environment to shield each web server instance, by default all zones have access to the full set of logical CPUs enabled in the system. To help ensure that each web server instance does not utilize excessive CPU cycles, the system was configured with three resource pools each bound to a separate zone. With 32 total logical CPUs available with the Sun UltraSPARC T1 processor,

1. The Apache 1.3.33 web server source kit can be downloaded directly from the Apache Software Foundation at <http://www.apache.org>.

each resource pool was configured with a minimum of 4 and a maximum of 10 logical CPUs. The remaining two logical CPUs were reserved for system administration use.

The following sections include the commands used to configure the Apache web server on the Sun Fire T1000 server using Solaris zones and resource pools.

Note – Local zone setup and configuration information can be found in generally available system administration documents [2][3]. Please refer to these documents for detailed information on the steps needed to install and configure local zones and to setup resource pools.

Configuring the Resource Pools

The steps in this section establish and configure the resource pools that will eventually be associated with the three zones.

1. Enable the resource pools facility and create a default configuration using the following two `pooladm` commands. The first command enables the resource pools facility, and the second saves the active configuration to the default file, `/etc/pooladm.conf`:

```
global# pooladm -e
global# pooladm -s
```

2. Create and bind a separate resource pool and processor set for each zone using a minimum of 4 CPUs and a maximum of 10 CPUs.
 - a. The following three `poolcfg` commands create and bind the resource pool and processor set for the first zone. The first command creates a processor set named `pset_web_zone1`; the second command creates a resource pool named `pool_web_zone1`; and the third command binds the newly created processor set and resource pool:

```
global# poolcfg -c 'create pset pset_web_zone1 (uint pset.min = 4; uint pset.max = 10) '
global# poolcfg -c 'create pool pool_web_zone1 '
global# poolcfg -c 'associate pool pool_web_zone1 (pset pset_web_zone1) '
```

- b. Similarly, use the `poolcfg` utility to create the processor sets (`pset_web_zone2` and `pset_web_zone3`) and resource pools (`pool_web_zone2` and `pool_web_zone3`) for the second and third zones using the same minimum and maximum CPU resource parameters.

3. Use the `pooladm -c` command to save the resource pool configuration created above, and then verify the configuration using the `pooladm` command with no arguments to report the processor sets and pool bindings:

```
global# pooladm -c
global# pooladm

System global
string system.comment
int system.version 1
boolean system.bind-default true
int system.poolid.pid 1933

pool pool_web_zone1
int pool.sys_id 1
boolean pool.active true
boolean pool.default false
int pool.importance 1
string pool.comment
pset pset_web_zone1

pool pool_default
int pool.sys_id 0
boolean pool.active true
boolean pool.default true
        int pool.importance 1
        string pool.comment
        pset pset_default

pool pool_web_zone3
        int pool.sys_id 3
        boolean pool.active true
        boolean pool.default false
        int pool.importance 1
        string pool.comment
        pset pset_web_zone3

pool pool_web_zone2
        int pool.sys_id 2
        boolean pool.active true
        boolean pool.default false
        int pool.importance 1
        string pool.comment
        pset pset_web_zone2

pset pset_web_zone1
        int pset.sys_id 1
        boolean pset.default false
        uint pset.min 4
        uint pset.max 10
        string pset.units population
        uint pset.load 4352
        uint pset.size 9
        string pset.comment
```

```
        cpu
            int    cpu.sys_id 23
            string  cpu.comment
            string  cpu.status on-line

...
output truncated for brevity; details on CPUs 19, 28, 25, 24, 27, 26, 13, and 12 included in original listing
...

pset pset_web_zone3
    int    pset.sys_id 3
    boolean pset.default false
    uint   pset.min 4
    uint   pset.max 10
    string pset.units population
    uint   pset.load 3746
    uint   pset.size 9
    string pset.comment

        cpu
            int    cpu.sys_id 18
            string  cpu.comment
            string  cpu.status on-line

...
output truncated for brevity; details on CPUs 5, 4, 1, 0, 3, 2, 15, and 14 included in original listing
...

pset pset_web_zone2
    int    pset.sys_id 2
    boolean pset.default false
    uint   pset.min 4
    uint   pset.max 10
    string pset.units population
    uint   pset.load 4552
    uint   pset.size 9
    string pset.comment

        cpu
            int    cpu.sys_id 20
            string  cpu.comment
            string  cpu.status on-line

...
output truncated for brevity; details on CPUs 17, 16, 7, 6, 9, 8, 11, and 10 included in original listing
...
```

```

pset pset_default
    int    pset.sys_id -1
    boolean pset.default true
    uint   pset.min 1
    uint   pset.max 65536
    string pset.units population
    uint   pset.load 49
    uint   pset.size 5
    string pset.comment

    cpu
        int    cpu.sys_id 21
        string cpu.comment
        string cpu.status on-line

...
output truncated for brevity; details on CPUs 22, 29, 31, and 30 included in original
listing
...

```

Configuring the Zones

The following steps illustrate how to configure the local zones for the web server instances in this scenario. For more detailed information on configuring zones, please see the *System Administration Guide: Solaris Containers—Resource Management and Solaris Zones*.

1. Create the zones for each web server instance using the `zonecfg` utility. In this example, the zone named `web_zone1` is created:

```

global# zonecfg -z web_zone1
web_zone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:web_zone1> create

```

- a. Specify the zone path, set `autoboot` value to be true, and associate the resource pool named `pool_web_zone1` with this zone:

```

zonecfg:web_zone1> set zonepath=/export/home/web_zone1
zonecfg:web_zone1> set autoboot=true
zonecfg:web_zone1> set pool=pool_web_zone1

```

The `zonepath` parameter specifies the location where the zone root file system is created. This file system consumes approximately 100 Mbytes of disk space.

- b. Add a file system, specifying the mount point and the file system type:

```

zonecfg:web_zone1> add fs
zonecfg:web_zone1:fs> set dir=/export/home/docs/public
zonecfg:web_zone1:fs> set special=/export/home/docs/public
zonecfg:web_zone1:fs> set type=lofs
zonecfg:web_zone1:fs> end

```

- c. Add a virtual network interface, specifying the IP address and the physical device for this network interface:

```
zonecfg:web_zone1 add net
zonecfg:web_zone1:net set address=135.135.10.211
zonecfg:web_zone1:net set physical=bge1
zonecfg:web_zone1:net end
```

The Sun Fire T1000 server contains four onboard controllers named bge0 – bge3. In this scenario, interface bge0 is configured only in the global zone and is used for system administration. Interfaces bge1 – bge3 are used for the local zones created in this scenario, web_zone1 – web_zone3.

Note that the physical device (bge1) specified for this net resource in the new local zone was already configured in the global zone with a separate IP address. When the zone is booted, the `ifconfig` utility reports this the new interface as bge1:1. In the global zone, both interfaces bge1 and bge1:1 are reported, however only bge1:1 is visible in the local zone.

- d. Add a comment by using the `attr` resource type:

```
zonecfg:web_zone1 add attr
zonecfg:web_zone1:attr set name=comment
zonecfg:web_zone1:attr set type=string
zonecfg:web_zone1:attr set value="Web Server Zone 1"
zonecfg:web_zone1:attr end
```

- e. Verify and commit the zone configuration, and exit the `zonecfg` utility:

```
zonecfg:web_zone1 verify
zonecfg:web_zone1 commit
zonecfg:web_zone1 exit
```

2. Install and boot the zone using the `zoneadm` utility:

```
global# zoneadm -z web_zone1 install
Preparing to install zone <web_zone1>.
Creating list of files to copy from the global zone.
...

global# zoneadm -z web_zone1 ready

global# zoneadm -z web_zone1 boot
```

3. Log into the new zone using the console option and configure the name and password information for this zone. Note that the hostname will not accept underscores; during the initial boot, the host name was set to `web-zone1`:

□

4. Repeat Steps 1 through 3 above to create the Web Server zones `web_zone2` and `web_zone3`, using the values in Table 1 for the network interface and the resource pool parameters. Each local zone is bound to a separate resource pool by specifying the `pn12262 488.81998` `m 998 278.38449` `502.31998` `m` (

□

□

6. Log into any of the newly created local zones. Verify that each has a dedicated network interface and the public static files are mounted under `/export/home/docs/public`:

```

global# zlogin web_zone1

[Connected to zone 'web_zone1' pts/2]
Last login: Sun Nov  6 09:23:20 on pts/3
Sun Microsystems Inc.   SunOS 5.10           Generic January 2005

# zonename
web_zone1

# ifconfig -a
lo0:3: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index
1
        inet 127.0.0.1 netmask ff000000
bge1:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 135.135.10.211 netmask ffffffff0 broadcast 135.135.10.255

# df -kl
Filesystem            kbytes    used    avail capacity  Mounted on
/                     73681625 22481417 50463392    31%    /
/dev                  73681625 22481417 50463392    31%    /dev
/export/home/docs/public
                     73681625 22481417 50463392    31%    /export/home/docs/public
/lib                  4994830 3611400 1333482     74%    /lib
/platform             4994830 3611400 1333482     74%    /platform
/sbin                 4994830 3611400 1333482     74%    /sbin
/usr                  4994830 3611400 1333482     74%    /usr
proc                   0         0         0         0%    /proc
ctfs                   0         0         0         0%    /system/contract
swap                 13288128    280 13287848     1%    /etc/svc/volatile
mnttab                 0         0         0         0%    /etc/mnttab
fd                     0         0         0         0%    /dev/fd
swap                 13287904     0 13287904     0%    /tmp
swap                 13287936     32 13287904     1%    /var/run

```

Monitor and Manage Zones and Resource Pools

The Solaris Operating System contains a number of utilities that enable administrators to efficiently monitor and manage CPU resources within Solaris zones.

- The `poolstat` utility provides current load and configuration information for all resource pools. This information is useful when monitoring and dynamically adjusting CPU resources among multiple zones:

```
global# poolstat -r all
id pool                type rid rset                min  max size used load
 1 pool_web_zone1      pset  1 pset_web_zone1            4   10   10 0.00 40.6
 0 pool_default        pset -1 pset_default              1 66K    3 0.00 0.06
 3 pool_web_zone3      pset  3 pset_web_zone3            4   10    9 0.00 41.0
 2 pool_web_zone2      pset  2 pset_web_zone2            4   10   10 0.00 41.4
```

- To display the currently defined processor sets, use the `psrset` command:

```
global# psrset
user processor set 1: processors 24 25 26 27 28 23 12 13 19 21
user processor set 2: processors 6 7 8 9 10 11 16 17 20 22
user processor set 3: processors 0 1 2 3 4 5 14 15 18
```

- To report the CPU utilization for both processes and all zones, use the `prstat -z` command:

```
global# prstat -z
Total: 445 processes, 866 lwps, load averages: 101.90, 101.17, 93.52
  PID USERNAME  SIZE  RSS STATE  PRI NICE    TIME  CPU PROCESS/NLWP
 27964 web        8080K 7016K run    33  0    0:00:00 0.1% web-cgi.p/1
 28055 web        5008K 3944K run    32  0    0:00:00 0.0% web-cgi.p/1
 28208 web        4688K 3624K run    24  0    0:00:00 0.0% web-cgi.p/1
 28173 web        3232K 2344K sleep  42  0    0:00:07 0.0% httpd/1
 18952 web        3232K 2344K sleep  43  0    0:00:37 0.0% httpd/1
 18314 web        3232K 2344K sleep  39  0    0:00:56 0.0% httpd/1
 18969 web        3232K 2344K sleep  37  0    0:00:38 0.0% httpd/1
 18785 web        3232K 2344K sleep  54  0    0:00:37 0.0% httpd/1
 18887 web        3232K 2344K sleep  46  0    0:00:37 0.0% httpd/1
 18488 web        3232K 2344K sleep  49  0    0:00:57 0.0% httpd/1
 18183 web        3232K 2344K sleep  44  0    0:00:56 0.0% httpd/1
 16976 web        3232K 2344K sleep  48  0    0:01:22 0.0% httpd/1
 17266 web        3232K 2344K sleep  45  0    0:01:19 0.0% httpd/1
 21187 web        3232K 2344K sleep  49  0    0:01:40 0.0% httpd/1
 21227 web        3232K 2344K sleep  36  0    0:01:42 0.0% httpd/1
ZONEID  NPROC  SIZE  RSS MEMORY    TIME  CPU ZONE
   3     122  442M  288M  1.6%    0:36:06 1.8% web_zone2
   1     121  438M  279M  1.5%    0:36:01 1.7% web_zone1
   2     118  434M  281M  1.5%    0:36:54 1.6% web_zone3
   0      71  343M  127M  0.7%    0:30:49 0.0% global
Total: 462 processes, 881 lwps, load averages: 101.54, 101.18, 94.01
```

- The `mpstat` utility can also be used with the `-P` option to display CPU utilization for a specified processor set. In this example, information on processor set 1 is displayed:

```
global # mpstat -P 1

CPU minf mjf xcal  intr  ithr  csw  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
12  261  0  50  201  25  361  9  105  231  0  1630  6  7  0  87
13  251  1  31  159  0  339  9  95  225  0  1555  5  7  0  88
19  108  0  799  183  117  126  6  37  109  0  606  5  9  0  86
21  4  0  8  77  0  152  0  14  41  0  64  0  5  0  94
23  258  1  31  153  0  332  10  96  157  0  1627  6  7  0  87
24  245  1  29  135  0  290  8  74  232  0  1385  5  7  0  88
25  224  0  25  121  0  265  7  65  224  0  1281  5  6  0  89
26  209  0  24  113  0  251  7  61  213  0  1211  5  6  0  89
27  194  0  121  966  402  239  7  57  196  0  1141  5  6  0  90
28  259  1  30  141  0  322  10  84  165  0  1635  6  7  0  88
```

- The `poolcfg` utility can be used to transfer CPUs between processor sets.

If the `mpstat` report shows that one of the resource pools is more heavily utilized than the others, then the `poolcfg` facility can be used to dynamically transfer CPU resources from the default processor set or from one zone to another. For example, the first command below transfers one CPU from the default processor set to the processor set associated with `web_zone2`. Similarly, the second command transfers one CPU from `web_zone1` to `web_zone3`:

```
global# poolcfg -dc 'transfer 1 from pset pset_default to pset_web_zone2'

global# poolcfg -dc 'transfer 1 from pset pset_web_zone1 to pset_web_zone3'
```

Alternately, the administrator can dynamically adjust the minimum and maximum values assigned to the resource pools to ensure sufficient CPU resources. For example, if `web_zone1` now needs a minimum of 8 and maximum of 12 CPUs to provide certain service levels rather than the original values set initially, the system administrator can use the `poolcfg` facility to modify the configuration:

```
global# poolcfg -dc 'modify pset pset_web_zone1 (uint pset.min = 8; uint pset.max = 12)
```

Build

Note – Apache, like most commercial web servers, supports the CGI interface for dynamic web operations. While the testing performed for this project used static web page requests, the Apache web server for each zone was configured to execute dynamic requests. The default directory for storing CGI utilities is in `/usr/local/apache/cgi-bin`. However a strategy similar to that used for static web pages could be used for CGI programs.

After building and deploying the Apache binaries in the global zone, the configuration file `/usr/local/apache/conf/httpd.conf` file was modified with the location of the document directory, `/export/home/docs/public`. The location of the Process ID file `pidfile` and log directory was set to the `/export/home/apache` directory which is unique in each local zone. In addition, the `MaxClients` parameter was set to 2048, to increase the number of child processes available to handle HTTP requests.

The web server instances are started in each local zone using the same command syntax as that used in the global zone:

```
# /usr/local/apache/bin/apachectl start
```

Within each zone, the Apache web server listens on TCP port 80 for HTTP requests received on the network interface configured only for that zone. Thus HTTP requests received in `web_zone1` are processed only by the web server running on the processor set configured for that zone.

Evaluation of Consolidated Web Server Environment

To validate the web serving environment and demonstrate the benefits of consolidating multiple web servers using Solaris Containers, a Sun Fire T1000 server and a typical modern x86-based system were evaluated.

- *Sun Fire T1000 server*

The Sun Fire T1000 server used in this evaluation featured an eight-core UltraSPARC T1 processor, 16 Gbytes of main memory, four 1000 baseT onboard network interfaces, and an 80 Gbyte SATA disk drive. With a single 300 W power supply, the server enclosure has a 1U rack server footprint. This system was installed with the Solaris 10 OS and configured with three Solaris zones, each running a separate instance of the Apache 1.3.33 web server.

- *Baseline x86-based system*

The baseline system used for this evaluation is a general purpose rack server targeted for the web server market with two 3.6 GHz Xeon processors, 4 Gbyte of main memory, two onboard Gbit network controllers, and two 73 GB internal SCSI disks with onboard RAID controller. With a single 550W power supply, the server enclosure has a 1 RU footprint. This system was installed with Red Hat Enterprise Linux 4 Update 1 and the two internal drives were configured as a RAID 1 mirror using the system BIOS utilities.

Testing Scenario

On both the Sun Fire T1000 server and the x86-based system, the testing was performed with only the Apache modifications listed earlier (see “Building and Configuring the Apache Web Server” on page 13) and with access logging disabled on both systems. There was no special tuning of operating system or web server options performed on either system, except that the file system holding the static HTML files was mounted using the `noatime` option to reduce disk activity.

A simple web serving workload was executed on both the Sun Fire T1000 server and the baseline x86-based server for evaluation purposes. The web workload consisted of 12 client drivers generating concurrent HTTP requests with a mix of static HTML files ranging from 200 bytes to 1M bytes, with an average requested web page of 16 Kbytes. On the Sun Fire T1000 server, requests were sent equally to all three web server instances. The workload driver measured the number of concurrent users as well as average HTTP operations per second.

Studies based on IT end-user surveys[6] have shown that most servers average 45 - 50% CPU utilization. The performance statistics collection tools provided by each operating system were used to achieve similar levels of CPU utilization on both systems. During this time, measurements were also collected on network throughput and power consumption.

Results

Table 2 details the results of the web server workload testing evaluating the Sun Fire T1000 server configured with three web servers each running in a local zone and a baseline x86-based system:

Table 2. Web server performance comparison.

System	Users	HTTP op/sec	Power Consumption	CPU Utilization
Sun Fire T1000 server	4100	12,597	175 Watts	46 %
Baseline x86 System	2000	6,595	260 Watts	47 %

As shown above, the Sun Fire T1000 server utilizing Solaris Containers technology to consolidate multiple web servers was able to handle more than two times the number of users and close to twice the HTTP throughput all while consuming approximately two-thirds the power consumption with similar CPU utilization on this simple workload.

Conclusions

This paper demonstrates that the Sun Fire T1000 server using Solaris Containers technology can be used as an effective Web-tier consolidation platform. The Solaris Containers technology enables multiple web server applications to each run in their own isolated application execution environment on a single server. Resource management tools included with the Solaris Operating System enable resources such as CPUs and memory to be easily allocated to each zone, monitored, and dynamically adjusted to meet changing performance requirements. Using a single operating system can help reduce complexity and grant easier system administration. Furthermore, consolidating the Web-tier onto a single energy-efficient platform like

the Sun Fire T1000 server can provide energy usage and space saving advantages over running web services on multiple individual servers.

About the Author

Kevin Kelly is an engineer in the Strategic Applications Engineering (SAE) group at Sun Microsystems, Inc.

Acknowledgements

The author would like to recognize the Strategic Applications Engineering (SAE) group and Performance and Availability Engineering (PAE) group for their contributions to this article.

References

- [1] Web Server Survey Reports
<http://news.netcraft.com/>
http://www.securityspace.com/s_survey/data/200510/index.html
- [2] *System Administration Guide: Solaris Containers—Resource Management and Solaris Zones*, Part Number 817-1592-11.
To access this document, go to <http://docs.sun.com/app/docs/doc/817-1592>
- [3] Lageman, Menno. "Solaris Containers – What They Are and How to Use Them," *Sun BluePrints OnLine*, May 2005.
To access this document, go to <http://www.sun.com/blueprints/0505/819-2679.pdf>
- [4] *Apache Performance Notes*.
<http://httpd.apache.org/docs/1.3/misc/perf-tuning.html>
- [5] *Server and Storage Consolidation 2004: Executive Interview Summary Report*.
<http://www.idc.com>
- [6] *Server and Storage Consolidation 2004: End-User Summary Report*.
<http://www.idc.com>

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com/>

