Deployment Guide for an Open Source Stack on the Solaris 10™ OS

Viet Pham

November 2006 (updated December 2007)

Table of Contents

1. Introduction

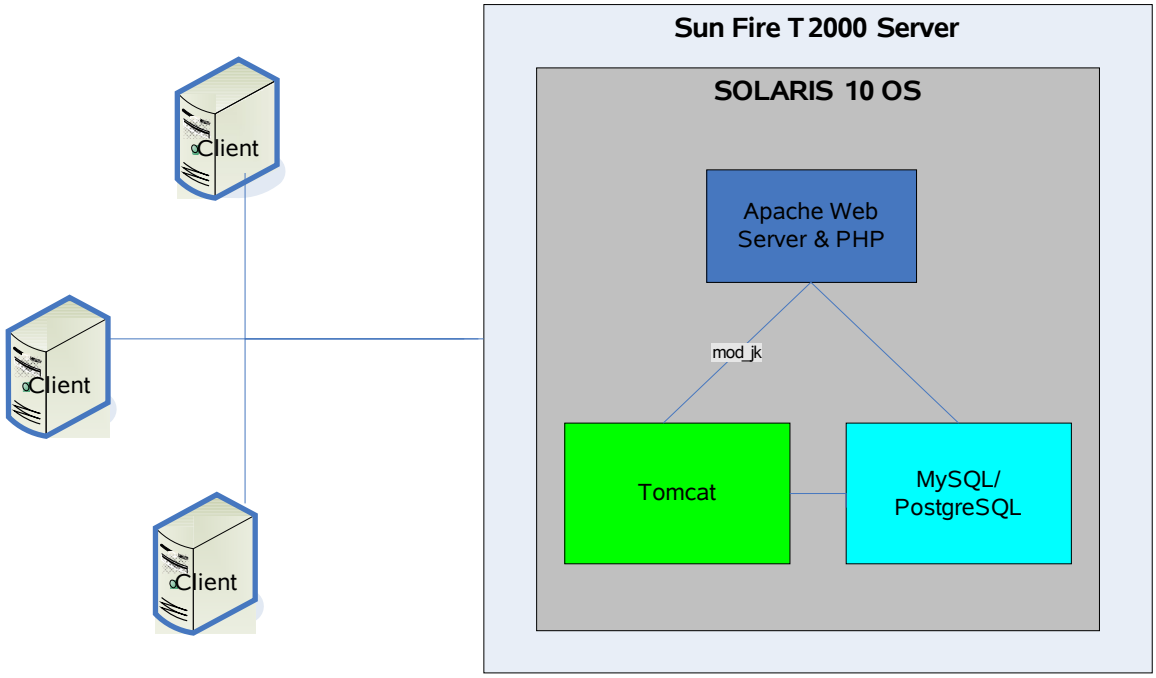
1.1 Scope

This guide serves as a starting point for users to install, configure, and perform basic tuning of the open-source stack SAMP (Solaris 10 OS, Apache 2.0.52, MySQL 5.0 or PostgreSQL, and PHP 5). There are similar papers for SAMP on the Internet. However, this paper will also include PostgreSQL and Tomcat. The Solaris Zones feature (part of Solaris Containers technology in the Solaris 10 OS) is also discussed to help users with scalability in mind. For tasks where existing documentation is adequate, only links are provided.

2. Basic SAMP Deployment Architecture

2.1. Overall Architecture

Deployment architecture is shown in the following diagram.



2.2. Installation and Configuration Instructions

2.2.1. Apache

Installation Instructions

Use Apache (1.x or 2.x) bundled with the Solaris 10 OS:

- Apache 1.x is installed at `/usr/apache`. Binary utilities such as `apachectl`, `httpd`, and `ab` are installed at `/usr/apache/bin`. Configuration files such as `httpd.conf` and `*.conf` are installed at `/etc/apache`.
- Apache 2.x is installed at `/usr/apache2`. Binary utilities such as `apachectl`, `httpd`, and `ab` are installed at `/usr/apache2/bin`. Configurations such as `httpd.conf` and `*.conf` are installed at `/etc/apache2`.

Binary Installation

When you need Apache binary versions that are later than the ones bundled with the Solaris 10 OS, you could go to Covalent Technologies (covalent.net), a Sun partner that offers professional support for Apache, to download newer binary versions. Installation instructions are also included in the download.

Note: If your platform is UltraSPARC®, use the optimized binaries for the CoolThreads Optimized Open Source Software Stack (Cool Stack), located at http://cooltools.sunsource.net/coolstack/index.html.

Source Code Compilation

To get the latest Apache version, use this installation method to compile the source code.

Note: If your platform is SPARC®, it is recommended that you use GCC for SPARC Systems located at http://cooltools.sunsource.net/gcc/index.html. GCC for SPARC Systems extends GCC to be able to use the optimizing Sun Code Generator for SPARC Systems.

Otherwise, use the following steps to compile Apache from source code:

- Download source code at http://httpd.apache.org/download.cgi
- `cd` to the folder you downloaded the source code in
- Run `gunzip *.gz`
- Run `tar xf *.tar`
- `cd` to the folder the `tar` command just created
- Run `CFLAGS='-DHARD_SERVER_LIMIT=2048 -DUSE_SO_LINGER -DSINGLE_LISTEN_UNSERIALIZED_ACCEPT' ./configure –prefix= absolute_path_to_where_you_want_to_install_apache –with-mpm=worker –enable-so`
- Run `make`
- Run `make install`

Note: Binary utilities and configuration files are located at `absolute_path_to_where_you_installed_apache/bin` and `absolute_path_to_where_you_installed_apache/conf`, respectively. The rest of this document will refer to `absolute_path_to_where_you_installed_apache` as `APACHE_HOME`.

Configuration

If you choose to use Apache bundled with the Solaris 10 OS or binary installation, the default configuration is good enough. If you choose to use the source code compilation installation method and are looking for advanced configuration for high performance in Apache, skip to the advanced deployment section. However, if you are just looking for a basic configuration to get Apache up and running, open `APACHE_HOME/conf/httpd.conf` and make sure that the following basic configurations are set:

- `User nobody`
- `Group nobody`

Start and Shut Down Apache Server:

- `cd to APACHE_HOME/bin`
- To start the server, run `./apachectl start`
- Point your browser to `http://www.server_name`. You should see a welcome page that starts with something like this: "If you can see this, it means that the installation of the Apache web server software on this system was successful. You may now add content to this directory and replace this page..." (see http://www.apache.org/foundation/preFAQ.html for more information).
- To shut down the server, run `./apachectl stop`

2.2.2. MySQL

Installation Instructions

Binary Installation:

- Add `mysql group`:
  - `groupadd mysql`
- Create `mysql user` and bind it to `group mysql`:
  - `useradd -g mysql mysql`
- `cd /usr`
- `mkdir local`
- Download MySQL 32-bit binary under the section labeled "Misc/special builds/packages downloads" to `/usr/local`
- `gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xf -`
- `ln -s full-path-to-mysql-VERSION-OS mysql`
- `cd mysql`
- Run `scripts/mysql_install_db –user=mysql`
- Run `chown -R root .`

- Run `chown -R mysql data`
- `chgrp -R mysql .`
- Run `bin/mysqld_safe –user=mysql`

ODBC Driver Installation:

- Download ODBC driver (32 -bit version) from http://dev.mysql.com/downloads/connector/odbc/3.51.html
- `gunzip *.gz`
- `pkgadd -d *.pkg`

Configuration

Please see the SDN article "MySQL InnoDB Performance Tuning for the Solaris 10 OS" at http://developers.sun.com/solaris/articles/mysql_perf_tune.html for configuration and tuning tips.

Note: If your platform is UltraSPARC, use the optimized binaries for the CoolThreads Optimized Open Source Software Stack (Cool Stack), located at http://cooltools.sunsource.net/coolstack/index.html.

2.2.3. PostgreSQL

Installation Instructions

Binary Installation:

If you have the Solaris 10 06/06 OS, PostgreSQL is already bundled. Please follow documentation located at http://www.postgresql.org/docs/. For other Solaris 10 OS versions, download the binaries located at http://pgfoundry.org/projects/solarispackages/.

Source Code Compilation:

- Download source code from http://www.postgresql.org/ftp/source/
- `gunzip *.gz`
- `tar xf *.tar`
- `cd postgres-version`
- `./configure –without-readline`
- `gmake`
- `su`
- `gmake install`
- `useadd -d /export/home/postgres -s /usr/bin/bash postgres`
- `mkdir /export/home/postgres`
- `chown postgres:10 /export/home/postgres`
- `mkdir /usr/localpgsql/data`
- `chown postgres /usr/local/pgsql/data`

- `su – postgres`
- `/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data`
- `/usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &`
- `/usr/local/pgsql/bin/createdb test`
- `/usr/local/pgsql/bin/psql test`

ODBC driver installation:

- Download source code from http://www.postgresql.org/ftp/odbc/versions/src/
- `gunzip *.gz`
- `tar xf *. tar`
- `cd odbc-version`
- `./configure`
- `make`
- `make install`

Start Postgres:

- `psql database_name` (from shell prompt)
- `\c database_name2` (from within Postgres interactive console to change database)

2.2.4. PHP

Installation Instructions

Binary Installation:

- For SPARC systems, get the binary and installation instructions at http://sunfreeware.com/programlistsparc10.html#php.
- For x86 systems, get the binary and installation instructions at http://sunfreeware.com/programlistintel10.html#php.

Source Code Compilation:

- Download source code for PHP (version 5.1.4) at http://www.php.net/get/php-5.1.4.tar.gz/from/a/mirror
- Untar the source code using these steps:
  - Change directory to where you downloaded the source code
  - `gunzip *.gz`
  - `tar xf *.gz`
  - Run `./configure --prefix=absolute_path_to_where_you_want_to_install_php --with-apxs2=absolute_path_to_apache_root_directory/bin/apxs`

- Run `make`
- Run `make install`

Configuration:

- After the command `make install` exits, the line `LoadModule php5_module modules/libphp5.so` should have been automatically added to `httpd.conf`. If not, add it to `httpd.conf`.
- Add these lines to `httpd.conf`:

```
AddType application/x-httpd-php .php .phtml

AddType application/x-httpd-php-source .phps
```

2.2.5. Tomcat

Installation Instructions:

- Download Tomcat from http://tomcat.apache.org/
- `cd` to the folder in which you downloaded Tomcat
- `gunzip *.gz`
- `tar xf *.tar`
- Set `JAVA_HOME=path_to_java`
- Set `TOMCAT_HOME=path_to_root_directory_you_installed_tomcat`
- `cd TOMCAT_HOME/bin/`
- Run `./startup.sh`
- Point your browser to `http://server_name:8080`. If you see the welcome page, Tomcat works.

Configuration

There is no configuration if Tomcat is a stand-alone server. You can start putting your application under the `TOMCAT_HOME/webapps/` folder and expect it to work. If Tomcat needs to work with Apache, see the next section.

2.2.6. Connecting Components Together

In order for PHP web applications to access the MySQL database, you need to compile PHP using the same steps mentioned previously except change the `configure` command to the following:

```
./configure --prefix=absolute_path_to_where_you_want_to_install_php    --with-
apxs2=absolute_path_to_apache_root_directory/bin/apxs –with-mysql=
absolute_path_to_where_you_installed_mysql
```

In order for PHP web applications to access a PostgreSQL database, you need to compile PHP using the aforementioned steps except change the `configure` command to the following:

```
./configure --prefix=absolute_path_to_where_you_want_to_install_php    --with-
apxs2=absolute_path_to_apache_root_directory/bin/apxs –with-postgres=
absolute_path_to_where_you_installed_postgres
```

Note: Do not compile PHP with both MySQL and Postgres at the same time using the following `configure` command because it does not work. You can compile only one database support at a time.

```
./configure --prefix=absolute_path_to_where_you_want_to_install_php    --with-
apxs2=absolute_path_to_apache_root_directory/bin/apxs –with-postgres=
absolute_path_to_where_you_installed_postgres –with-mysql=
absolute_path_to_where_you_installed_mysql
```

To connect Apache with Tomcat, you need to install `mod_jk` using the following steps:

- Download the `mod_jk` module source code from http://tomcat.apache.org/connectors-doc/
- `gunzip *.gz`
- `tar xf *.tar`
- `jakarta*/jk/native`
- `./configure –with-apxs=/APACHE_HOME/bin/apxs`
- `make`
- `make install`
- Put the following lines in `httpd.conf`:

```
LoadModule jk_module modules/mod_jk.so

<IfModule mod_jk.c>

      JkWorkersFile APACHE_HOME/conf/workers.properties
      JkLogFile     APACHE_HOME/logs/mod_jk.log
      JkLogLevel    info
      JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
      JkRequestLogFormat     "%w %V %T"
      JkAutoAlias "TOMCAT_HOME/webapps"
      <Directory "TOMCAT_HOME/webapps/*/*">

            Options FollowSymLinks

            AllowOverride None

            Order allow,deny

            Allow from all

      </Directory>
      JkMount /*.jsp ajp13
      JkMount /*/*.jsp ajp13

 <IfModule mod_jk.c>
```

- Copy `jakarta*/jk/conf/workers.properties to APACHE_HOME/conf`
- Edit the `workers.properties` content to have these lines:
  - `workers.tomcat_home=TOMCAT_HOME`
  - `workers.java_home=/usr`
  - `ps=/`
  - `worker.list=ajp13`

- worker.ajp13.port=8009
- worker.ajp13.host=localhost
- worker.ajp13.type=ajp13
- worker.ajp13.cachesize=10
- worker.ajp13.cache_timeout=400
- worker.ajp13.socket_keepalive=1
- worker.ajp13.recycle_timeout=200

3. Performance

3.1. Apache Tuning

- First, make sure the worker module was loaded by running the command `APACHE_HOME/bin/httpd -l`. If `worker.c` shows up, Apache is using multithreaded mode, which can handle more requests with less memory footprint than the prefork mode (`prefork.c`). The configuration for worker mode is encapsulated in the following directive:

```
<IfModule worker.c>
       StartServers         2
       MaxClients           150
       MinSpareThreads      25
       MaxSpareThreads      75
       MaxRequestsPerChild  0
</IfModule>
```

Full documentation of how to tune these parameters to suit your web server's load is at http://httpd.apache.org/docs/2.2/mod/quickreference.html. However, the following is a quick summary that an administrator can use to figure out how to adjust the numbers:

In worker mode, the server is started with a parent process, which then forks child processes. Each child process can fork threads. Each thread is capable of serving one request. (`Parent process -> Child processes -> Threads`). Child processes and threads creation are dynamically adjusted to meet the server's load. However, an administrator can configure the limits using the following logics:

- `StartServers` is the number of child processes to be started when the web server starts. The parent process automatically forks more of these child processes to meet the load demand or kills unused child processes to free up memory. However, the maximum is capped at `ServerLimit` (see documentation for how to use this directive), which is supposed to be the maximum of 2048 because that is how many Apache compilations (`CFLAGS='-DHARD_SERVER_LIMIT=2048`) are hard coded.
- `ThreadsPerChild` is the number of threads a child process can create. The default value is 25. This number must be less than or equal to `TheadLimit`. (See documentation for how to use `ThreadsPerChild` and `ThreadLimit` directives.)
- `MaxClients` is the total number of threads all child processes can create. It is also the maximum number of concurrent requests Apache can serve at any given time. `MaxClients/TheadsPerChild` must be less than or equal to `ServerLimit`.
- `MinSpareThreads` is the minimum number of threads child processes must make available to server requests at all times.
- `MaxSpareThreads` is the maximum number of threads child processes can make available.

- `MaxRequestsPerChild` is the total number of requests a child process can serve before it is killed and recreated. Setting it to zero is telling the server never to kill a child process. The advantage of setting to zero is you avoid the overhead of killing and recreating child processes. The disadvantage is you run the risk of gradually exhausting memory if your web application has memory leaks. The administrator needs to know the nature of your web application in order to weigh the value to set.
- Other tuning settings include (see documentation for more details on these parameters):
  - Set `MaxKeepAliveRequests as 0` in `httpd.conf`.
  - Set `ServerTokens` to `Minimal` in `httpd.conf`.
  - Make sure `KeepAlive` is on in `httpd.conf`.
  - Make sure `HostNameLookups` is off in `httpd.conf`.
  - Make sure `LogLevel` is `warn` in `httpd.conf` and move log directory to storage.
  - Make sure `AllowOverride` is `None` in `httpd.conf` to avoid the checking of `.htaccess`.
  - Make sure only necessary modules are loaded.

3.2. Solaris Tuning

If your platform is UltraSPARC (Sun Fire™ T2000 Server), use the following recommended tuning:

TCP Tuning on Sun Fire T2000 Server:

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
ndd -set /dev/tcp tcp_conn_req_max_q 81920
ndd -set /dev/tcp tcp_conn_req_max_q0 81920
ndd -set /dev/tcp tcp_max_buf 4194304
ndd -set /dev/tcp tcp_cwnd_max 2097152
ndd -set /dev/tcp tcp_recv_hiwat 20480
ndd -set /dev/tcp tcp_xmit_hiwat 400000
ndd -set /dev/tcp tcp_local_dack_interval 500
```

Network Port Tuning on Sun Fire T2000 Server:

```
#!/bin/bash
  INSTS="0 1 2 3"
  NIC=ipge
  for port in $INSTS ; do
       ndd -set /dev/$NIC instance $port
       ndd -set /dev/$NIC rx_intr_pkts 600
       ndd -set /dev/$NIC rx_intr_time 600
  done
```

`Interrupt Coalescing` Tuning on Sun Fire T2000 Server:

Run the following command in a shell to make sure only virtual CPUs 29, 30, 31 serve the interrupts. That forces CPUs from 0 to 28 to be dedicated to serving requests only.

```
psradm -i 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

Tuning in `/etc/system` on Sun Fire T2000 Server:

```
set px:px_fabric_die = 0
set px:px_fabric_die_rc_ue = 0
set px:px_fabric_die_rc_ue_gos = 0
set segmap_percent=60
set smallfile=1572864
set maxpgio=128
set rlim_fd_max=260000
set shmsys:shminfo_shmax=0xffffffff
set tune_t_fsflushr = 1
set autoup = 60
set sq_max_size = 10000
```

4. Advanced Deployments

4.1. Apache on Solaris Zones

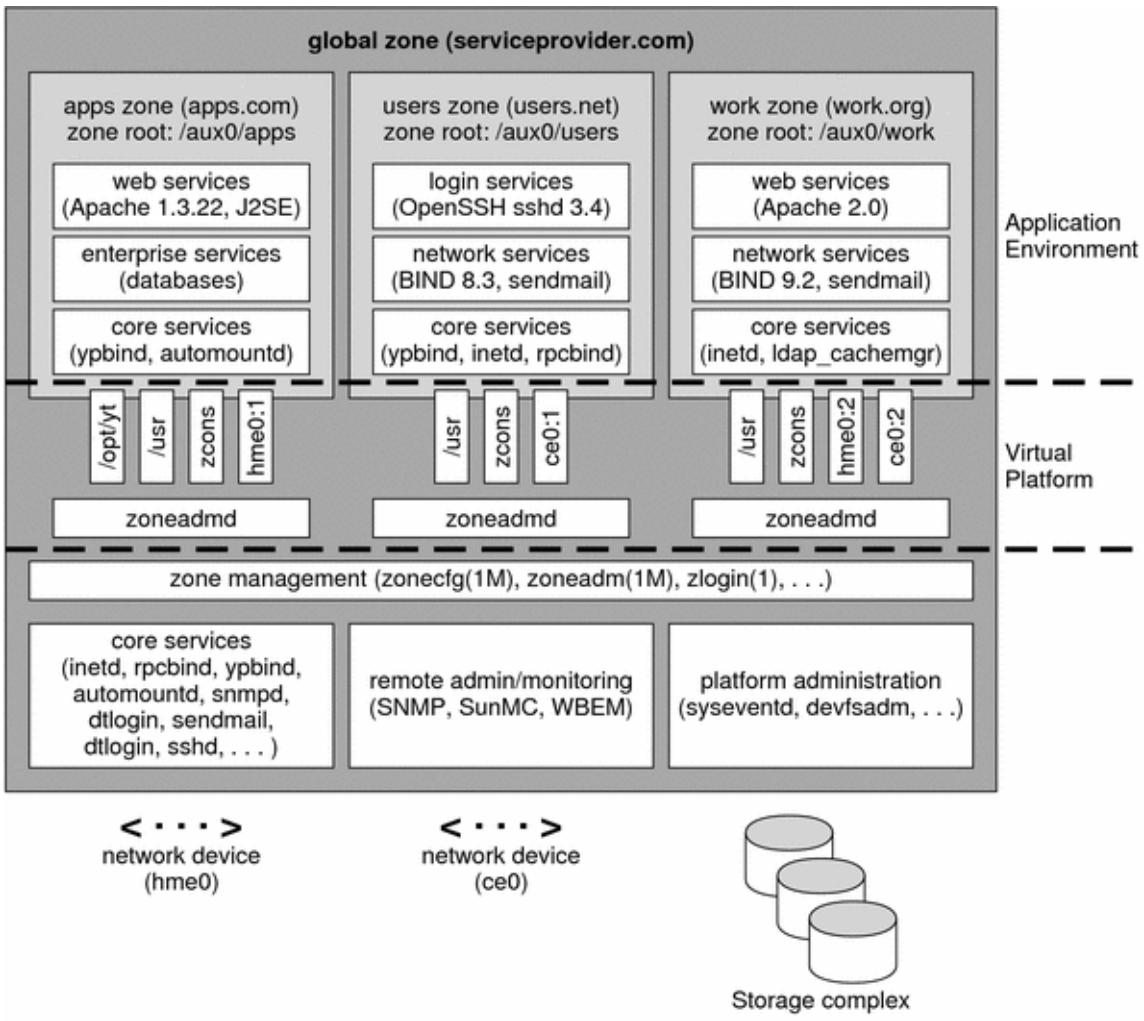
4.1.1. Introduction to Solaris Zones

The Sun BluePrints article *Web Consolidation on the Sun Fire T1000 using Solaris Containers* explains:

"*The Solaris Zones partitioning technology is used to virtualize operating system services and provide an isolated and secure environment for running applications. A zone is a virtualized operating system environment created within a single instance of the Solaris OS. When you create a zone, you produce an application execution environment in which processes are isolated from the rest of the system. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with super user credentials cannot view or affect activity in other zones.*

There are two types of non-global zone root file system models: sparse and whole root. The sparse root zone model optimizes the sharing of objects. The whole root zone model has a copy of necessary files for configuration."

The following diagram illustrates a typical zone environment that has three zones. They are called apps zone, users zone, and work zone. Both the apps zone and the work zone contain a different version of Apache. Multiple zones with multiple Apache instances can be used to scale up throughput. This work has been done and is documented in *Web Consolidation on the Sun Fire T1000 using Solaris Containers:* http://www.sun.com/blueprints/1205/819-5149.html.



4.2 How to Install and Set Up Zones

If you prefer a GUI that walks you through the process of installing and configuring zones, download and install the packages located at [http://cooltools.sunsource.net/consolidation/index.html.](http://cooltools.sunsource.net/consolidation/index.html)

However, if you prefer using the command line, follow these steps to install and configure zones.

4.2.1 Using a Sparse Zone

Modify the following script using the provided comments and run it to set up a sparse zone:

```
#!/bin/bash
# Use zonecfg command to configure or create a new zone
echo "CREATE ZONE"
zonecfg -z sparse
create

#Set the path of the zone (sparse is the name of the zone you just created)
echo "SET PATH"
set zonepath=/export/home/sparse

#Set autoboot=true to make sparse zone boot when global zone is booted
echo "SET AUTOBOOT"
set autoboot=true
```

```
#Set the resource pool only when resource pool service is on. Otherwise, skip the next step.
#set pool=pool_default

#Add mount point system
echo "ADD MOUNT POINT"
add fs

#Set mount point for the file system to /usr/local
echo "SET MOUNT POINT"
set dir=/usr/local

#Specify that /opt/local in the global zone is to be mounted as /usr/local
echo "SPECIFY GLOBAL ZONE TO BE MOUNTED"
set special=/opt/local

#Set the file system type to lofs
echo "SET FILE SYSTEM"
set type=lofs

#End the file system specification
end

#Perform the next 3 steps only when you want to create a sparse zone:
echo "INHERIT FILES"
add inherit-pkg-dir
set dir=/usr/sfw
end

#Add network interfaces
echo "SET UP NETWORK"
add net

#Set IP address, you must have static IP for zone to work. Replace
set address=x.x.x.x

#Set the physical device type for the network interface. Replace ce0 with your own device
type
set physical=ce0

#End network specification
end

#Add a zone-wide resource control
echo "SET UP RESOURCE CONTROL"
add rctl

#Set the name of the resource control to zone.cpu-shares
set name=zone.cpu-shares

#Add values for the privilege, the share limit, and the action to be taken when that
threshold is reached.
add value (priv=privileged,limit=20,action=none)

#End resource control specification
end

#Add a comment by using the attr resource type.
echo "COMMENTS"
add attr

#Set name to "comment"
set name=comment

#Set the type to string.
```

```
set type=string

#Set the value to a comment that describes the zone.
set value="Testing Apache Web Server on Zone"

#End the attr resource type specification.
end

#Verify the zone configuration for the zone.
echo "VERIFY"
verify

#Commit the setting
echo "COMMIT"
commit


#Exit the zoncfg
exit

#Create folder /export/home/sparse(sparse is the name of the zone you created at the
beginning)
echo "CREATE sparse FOLDER"
mkdir /export/home/sparse

#Change permission to 700
echo "CHANGE PERMISSION"
chmod 700 /export/home/sparse

#Verify the zone configuration
echo "VERIFY ZONE CONFIGURATION"
zoneadm -z sparse verify

#Install the zone
echo "INTALL ZONE"
zoneadm -z sparse install

#Boot the zone
echo "BOOT ZONE"
zoneadm -z sparse boot

#Verify the boot
echo "VERIFY BOOT"
zoneadm list -v

#Login zone
echo "LOGIN ZONE"
zlogin -C sparse
```

4.2.2 Using a Whole Zone

Modify the following script using the comments provided and run it to set up a whole zone:

```
#!/bin/bash

# Use zonecfg command to configure or create a new zone
echo "CREATE ZONE"
zone -z apa
create

#Set the path of the zone (apa is the name of the zone you just created)
echo "SET PATH"
set zonepath=/export/home/apa

#Set autoboot=true to make apa zone boot when global zone is booted
```

```
echo "SET AUTOBOOT"
set autoboot=true

#Set the resource pool only when resource pool service is on. Otherwise, skip the next step.
#set pool=pool_default

#Add mount point system
echo "ADD MOUNT POINT"
add fs

#Set mount point for the file system to /usr/local
echo "SET MOUNT POINT"
set dir=/usr/local

#Specify that /opt/local in the global zone is to be mounted as /usr/local
echo "SPECIFY GLOBAL ZONE TO BE MOUNTED"
set special=/opt/local

#Set the file system type to lofs
echo "SET FILE SYSTEM"
set type=lofs

#End the file system specification
end

#Add network interfaces
echo "SET UP NETWORK"
add net

#Set IP address. You must have a static IP for zone to work.
set address=x.x.x.x

#Set the physical device type for the network interface. Replace ce0 with your device type.
set physical=ce0

#End network specification
end

#Add a zone-wide resource control
echo "SET UP RESOURCE CONTROL"
add rctl

#Set the name of the resource control to zone.cpu-shares
set name=zone.cpu-shares

#Add values for the privilege, the share limit, and the action to be taken when that
threshold is reached.
add value (priv=privileged,limit=20,action=none)

#End resource control specification
end

#Add a comment by using the attr resource type.
echo "COMMENTS"
add attr

#Set name to "comment"
set name=comment

#Set the type to string.
set type=string

#Set the value to a comment that describes the zone.
set value="Testing Apache Web Server on Zone"

#End the attr resource type specification.
```

```
end

#Verify the zone configuration for the zone.
echo "VERIFY"
verify

#Commit the setting
echo "COMMIT"
commit

#Exit the zoncfg
exit

#Create folder /export/home/apa (apa is the name of the zone you created at the beginning)
echo "CREATE apa FOLDER"
mkdir /export/home/apa

#Change permission to 700
echo "CHANGE PERMISSION"
chmod 700 /export/home/apa

#Verify the zone configuration
echo "VERIFY ZONE CONFIGURATION"
zoneadm -z apa verify

#Install the zone
echo "INTALL ZONE"
zoneadm -z apa install

#Boot the zone
echo "BOOT ZONE"
zoneadm -z apa boot

#Verify the boot
echo "VERIFY BOOT"
zoneadm list -v

#Login zone
echo "LOGIN ZONE"
zlogin -C apa
```

Note: After zone installation and configuration, use the previous instructions to install, configure, and tune Apache, PHP, MySQL/PostgreSQL, and Tomcat.

5. About the Author

Viet Pham (viet.pham@sun.com) works in Market Development Engineering at Sun Microsystems, in the open source segment. His focus is to make sure the Solaris 10 OS is a low-cost platform on which to deploy open source applications.

6. References

System Administration Guide: Solaris Containers - Resource Management and Solaris Zones, Chapter 16, Introduction to Solaris Zones: http://docs.sun.com/app/docs/doc/817-1592/6mhahuonv?a=view

"MySQL InnoDB Performance Tuning for the Solaris 10 OS":
http://developers.sun.com/solaris/articles/mysql_perf_tune.html

Web Consolidation on the Sun Fire T1000 using Solaris Containers:
http://www.sun.com/blueprints/1205/819-5149.html