EVALUATION OF PERFORMANCE OF SOLARIS TRUSTED EXTENSIONS USING CONTAINERS TECHNOLOGY

GENTI DACI

Faculty of Information Technology
Polytechnic University of Tirana, Albania

UDC: 004.45

Key words: Solaris Containers.

Abstract: Server and system administrators have been concerned about the techniques on how to better utilize their computing resources. Today, there are developed many technologies for this purpose, which consists of running multiple applications and also multiple operating systems on the same hardware, like VMWARE, Linux-VServer, VirtualBox, Xen, etc. These systems try to solve the problem of resource allocation from two main aspects: running multiple operating system instances and virtualizing the operating system environment. Our study presents an evaluation of scalability and performance of an operating system virtualization technology known as Solaris Containers, with the main objective on measuring the influence of a security technology known as Solaris Trusted Extensions. Solaris. We will study its advantages and disadvantages and also the overhead that it introduces to the scalability of the system's main advantages.

ISSN: 1804-0527 (online) 1804-0519 (print)

Vol.8 (2), PP. 63-69

Introduction

During the latest years, a lot of projects have been looking on virtualizing operating system environments, such as FreeBSD Jail, Linux-VServer, Virtuozzo etc. This virtualization technique is based in using only one underlying operating system kernel. Using this paradigm the user has the possibility to run multiple applications in isolation from each other. The basic idea can be normally described as running groups of processes that cannot be interrupted by others in different virtual environments.

Solaris Containers are built on the same paradigm, offering virtualization on the operating system-level. Solaris approach on Virtualization, extends its virtual operating system environment to include many more features of a separate machine, such as a per-Container console, dedicated system log, packaging database, run level, identity (YP, NIS), and IPC facilities. They make it possible to run more than one instance of the operating system into the same kernel. Many of the resources of our Server architecture are usually not properly and efficiently used, like CPU cycles, RAM, storage etc. With this technology we can better utilize these resources and make them useful for the entire.

These Containers act as completely isolated virtual machines within a computer aiming to reduce costs in both hardware and system administration. Furthermore, the Solaris Containers mechanism can provide protection through compartmentalization for separate virtual machines on a single physical machine. It is cheaper to install and to configure, because only a single copy of OS is involved, compared to several OS instances in the case of Xen. Furthermore, it is not limited for high-end systems compared to logical partitioning. Moreover, the granularity of resource allocation is fine-grained than the logical partitioning. In comparison to virtual machine monitors, Solaris Zones reduces performance overheads and reduces the cost of administration because there are no multiple operating system instances in a system.

Operating systems

Solaris/OpenSolaris are Operating Systems performing as the main building blocks of computer systems; they provide the interface between user applications and computer hardware. An operating system is a program that acts as an intermediary between the user of a computer and the hardware. The purpose of it is to provide an environment in which a user can execute programs in a convenient and efficient manner. As stated by Silberschatz (2002), Solaris is a multiuser, multitasking, multithreading operating environment, developed by Sun Microsystems. Solaris is a Unix-based operating system introduced by Sun Microsystems in 1992 as the successor to SunOS. Solaris is known for its scalability, especially on SPARC systems, and for its features such as Zones, DTrace and ZFS. The majority of its codebase is now open source software via the OpenSolaris project. Watters (2005) explained that OpenSolaris is an open source operating system based on Sun Microsystems' Solaris. It is derived from the Unix System V Release 4 codebase, with significant modifications made by Sun since it bought the rights to that code in 1994. It is the only open source System V derivative available.

Containment strategies

A Solaris Container is a runtime environment for applications. Parts of the container are Solaris 10 Resource Manager and Solaris Zones. Zones isolate application components from one another even though they share the same instance of the Solaris OS. The container establishes limits for resource consumption, such as CPU, memory. Sun's BrandZ technology is used to run Linux applications on the Solaris Operating System. Linux applications run unmodified in the secure non-global zone environment. This enables the developing, testing, and deploying of Linux applications on Solaris.

Related Containment Technologies on other operating systems are: chroot (Unix OS), FreeBSD jails, Systrace, AppArmor, Xen and VMWare.

AIM (Advanced Integration Matrix) The AIM Multiuser Benchmark, also called the AIM Benchmark Suite, is a job throughput benchmark widely used by UNIX computer system vendors. AIM is a program written in C that forks many processes called tasks, concurrently running random set of subtests called jobs. There are 53 kinds of jobs, with different aspect of the operating system, such as disk-file operations, process creation, user virtual memory operations, pipe I/O, and compute-bound arithmetic loops. Each subrun reports a metric of Jobs completed per minute, with the final report for the overall benchmark being a table of that throughput metric versus number of tasks. A given system will have a peak number of tasks N at which the jobs per minute is maximized. Either N or the value of the jobs per minute at N is the metric of interest. The peak performance is the highest jobs/minute the system achieved. The sustained performance is the square root of the total area under your performance curve up to the point of crossover. The point of crossover is that point at which the Jobs

per Minute/User Load = 1.0. The JTI (job timing index) rating is the worst case JTI.

Solaris zones

Solaris Zones are a component of the Solaris Container environment. A zone is a virtualized operating system environment created within a single instance of the Solaris Operating System. When creating a zone, we produce an execution environment where 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 with superuser permissions a process cannot view or affect activity in other zones. A zone also provides an abstract layer that separates applications from the physical attributes of the machine, like physical device paths. We can create a maximum number of 8192 zones on a system. 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 provides the maximum configurability.

FIGURE 1. DYNAMICS OF STRUCTURE OF INDICATOR (SAMPLE)

The commands to install, boot, halt, uninstall and clone non-global zones are zoneadm, zonecfg and zlogin. The cycle of zones is: Configured → Incomplete → Installed → Ready → Running.

Advantages and disadvantages

Zones are used for systems that integrate a number of applications on a single server. Its advantages are the lower cost and complexity of managing numerous machines. They enable more efficient resource utilization of the system. Other

advantages are the isolation that this technology introduces to the services and also the security. We have a higher level of security using Zones because in case of an attacker breaking into the zone, he cannot break into the other non-global zones or into the global zone. So a zone is completely isolated and it is transparent from the rest of the system. It communicates with the rest of the system through networking API. A zone does not need a dedicated CPU, a physical device, or a piece of physical memory. These resources can be multiplexed across some zones, or allocated using the resource management feature. Some disadvantages are that

the zones need some extra resources, like file system usage, processor cycles and memory usage.

Isolation and security

Solaris Zones partitioning technology, provides a means of virtualizing operating system services to create an isolated environment for running applications. This isolation prevents processes that are running in one zone from monitoring or affecting processes running in other zones. Basic communication between zones is accomplished by giving each zone IP network connectivity. An application running in one zone cannot observe the network traffic of another zone. This isolation is maintained even though the respective streams of packets travel through the same physical interface. The applications are also prevented from monitoring or intercepting each other's network traffic, file system data, or process activity. Actions taken by a zone administrator in a non-global zone do not affect the rest of the system.

Design and implementation

The test and experiments are based on the below hardware system:

Toshiba Satellite A110, PSAB0E-00F00MGE, Genuine Intel® CPU T2250 @ 1.73 GHz, Physical Memory 1527 MB (Kingston DDR2 PC2-5300 1024MB, Samsung DDR2 PC2-4300 512MB), RealTek RTL8139/810x Family Fast Ethernet NIC, Western Digital WDC WD2500BEVS-00UST0 ATA 250 GB (8MB buffer size), 51GB ZFS partition.

And the operating system and software characteristics are like below:

SunOS 5.11 snv_101b i86pc i386 i86pc Solaris, which is OpenSolaris 2008.11 but this are also applicable to Solaris 10, AIM (advanced integration matrix) benchmark v7 and v9 Caldera International, Inc.

We will make our experiments based on the creation, installing and booting of zones. The goal of the experiments is to minimize the overhead introduced by the zones regarding to memory, storage and processing power. The first part consists in the configuration and the installation of a zone. We estimate the zone creation with the time and disk usage to measure the disk usage. We create the zone in the following way:

```
zonecfg -z zone1 "create; set
zonepath=/zones/zone1; set autoboot=true; add net;
set address=192.168.0.1/24; set
physical=rge0; end; commit; exit"
```

Next we install the zone and estimate it with:

```
time zoneadm -z zone1 install
```

And then we gain the results of the package information and the disk usage with *pkginfo* and *du -hs*.

When installing a zone by default the whole directory tree is copied to the non-global zone thus increasing the installation time and also the disk usage. In order to decrease this we use specified directories by the global zone directory tree by inheriting them.

```
zonecfg -z zone1 "create; set
zonepath=/zones/zone1; set autoboot=true; add
inherit-pkg-dir; set dir=/usr; end; add inherit-
pkg-dir; set dir=/sbin; end; add inherit-pkg-dir;
set dir=/bin; end; add inherit-pkg-dir; set
dir=/platform; end; add inherit-pkg-dir; set
dir=/lib; end; add net; set address=192.168.0.1/24;
set physical=rge0;
end; commit; exit"
```

Again with the same method we will show that in this way the overhead introduced by the creation and installation of zones is lower than the first one. The next experiments that we will make are based on AIM benchmark and show the performance of the system as a whole and the overhead introduced by zones. The benchmarks will evaluate the performance of a multiuser system (aim7) for example a server, and of a single user system (aim9) based on a number of tests, like arithmetic tests, disk/fs I/O tests, IPC, function calls, algorithmic tests etc. Firstly we will execute the benchmark only on the global zone without any non-global zone installed. And then the benchmark will be executed on the global zone with the running non-global zones.

Results

Below are presented the results of the tests and experiments:

1) The installation of a zone (the whole root zone)

```
#time zoneadm -z zone1 install
```

*A ZFS file system has been created for this zone
Authority: Using <http://pkg.opensolaris.org/release/>.*

Image: Preparing at /zones/zone1/root ... done.

Cache: Using /var/pkg/download.

Installing: (output follows)

DOWNLOAD PKGS FILES XFER (MB)

Completed 52/52 7862/7862 72.41/72.41

PHASE ACTIONS

Install Phase 12939/12939

PHASE ITEMS

Reading Existing Index 9/9

Indexing Packages 52/52

Done: Installation completed in 235.814 seconds.

Next Steps: Boot the zone, then log into the zone console (zlogin -C) to complete the configuration process

real 3m56.131s

user 1m4.098s

sys 0m11.722s

pkginfo lwc -l 908

pkginfo -d /zones/zone2/root/var/sadm/pkg/ lwc -l 908

du -hs /zones/zone2 894M /zones/zone2

2)The installation of a zone with inherited packages (the sparse zone)

#time zoneadm -z zone1 install

A ZFS file system has been created for this zone.

Authority: Using <http://pkg.opensolaris.org/release/>.

Image: Preparing at /zones/zone1/root ... done.

Cache: Using /var/pkg/download.

Installing: (output follows)

DOWNLOAD PKGS FILES XFER (MB)

Completed 52/52 7862/7862 72.41/72.41

PHASE ACTIONS

Install Phase 12939/12939

PHASE ITEMS

Reading Existing Index 9/9

Indexing Packages 52/52

Done: Installation completed in 229.814 seconds.

Next Steps: Boot the zone, then log into the zone console (zlogin -C) to complete the configuration

process

real 3m54.225s

user 1m4.009s

sys 0m10.512s

pkginfo lwc -l 908

pkginfo -d /zones/zone2/root/var/sadm/pkg/ lwc -l 118

du -hs /zones/zone2 358M /zones/zone2

The paper presents (see Appendix) the results of the aim benchmark in the two case without and with non global-zone running, from left to right.

Conclusion

Based on the results of the tests and experiments we can say that the sparse zone installation gives a lower overhead to the system in terms of disk usage, memory and processing power. The number of packages that the zone requires to be installed, the time needed for the zone to be installed also the storage that the zone consumes are lower. Approximately the rates of time, package and disk usage between sparse zone and whole root zone are 18/23, 1/9 and 1/3. These rates vary on the configuration of the sparse zone. The performance and the overhead that a zone introduce to the system based on the benchmark is at this rates: Peak = 976.5/993.5, Sustained = 344.4/365.6, Minimum JTI

= 91/92. Also from the singleuser benchmark (aim9) results we can say that the values of the tests differ very little from each other and we can conclude that in this way we have achieved a better utilization of hardware and operating system without affecting the overall performance of our system. We have shown that with this virtualization technology we can make use of all of our resources and thus achieving a higher processing power.

References

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 2002. „Operating System Concepts, 6th Edition“, John Wiley and Sons Inc, page.

Dr. Paul A. Watters, 2005, “Solaris 10, The Complete Reference“, McGraw Hill,

Sun Microsystems 805–3727–10, 2005, Solaris System Administration Guide, Volume 1, page

Sun Microsystems 805–3727–10, 2005, System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Caldera International AIM Multiuser Benchmark - Suite VII, Suite IX, Version 1.1., Inc.

Daniel Price, Andrew Tucker, 2004, Solaris Zones: Operating System Support for Consolidating Commercial Workloads, Sun Microsystems, Inc.

Slicing and Dicing servers, A guide to virtualization and containment technologies, Sun BluePrints™ OnLine, October 2005

Chandan B.N, June 2005, “New Security Features in Solaris 10 and Dtrace“, Sun Microsystems Inc.

Paul Lovvik, Joseph Balenzano, May 2005, “Bringing Your Application Into the Zone“

Jeff Victor, 2006, “Solaris Containers technology architecture guide“, Sun Microsystems,

Appendix

TABLE 1. BENCHMARK WITHOUT SOLARIS TRUSTED EXTENSIONS

Arithmetic				InterProcess Communication	
Thousand Ops per second				Messages per Second	
	Add	Div	Mult	TCP/IP	197145
Short	3480000	108000	607050	UDP/IP	175000
Int	3570000	85970	608040	FIFOs	8600
Long	3360000	86400	607560	PIPEs	447950
Float	1116000	357000	690000	Shared RAM	272400
Double	1026000	346500	690000	Stream Pipe	225450
				DataGram Pipe	205050
Disk/Filesystem I/O				Library System	
(Kbytes per Second and Ops per Second)				Various Rates	
	Cashed	Sync		Setjmp/Longjmp	16300500
Rand Read	37647	89		Ram Copy	-2147483648.0
Seq Write	70620	86		Numeric Rates	1267050
Rand Write	49708	88		Trig Rtas	680000
Seatches		7387		String Rtns	3000
Copy		50945		Mem Rtns 1	2145000
Seq Read		159683		Mem Rtns 2	764350
Create/Close		31034		Sort/Searc	1365
Link/Unlin		10017		Signals	129000
Directory		1685000		Misc. Rtns	5245
Memory & Process Management				Shell Rtns 1	52
Ops per Second				Shell Rtns 2	54
Brk()	524378	Paging	82450	Shell Rtns 2	54
Exec ()	165	Fork ()	935	Algorithm Tests	
Function Call				Operations per Second	
Ops per Second				Newton/Raphson	-1.0
0 Args	116736000	1 Arg	266752000	3D projection	2403750
2 Args	240128000	15 Arg	56064000	Linear System	815
				Taylor Series	9478750
				Integer Sieve	66

TABLE 2. BENCHMARK WITH SOLARIS TRUSTED EXTENSIONS

Arithmetic				InterProcess Communication	
Thousand Ops per second				Messages per Second	
	Add	Div	Mult	TCP/IP	194085
Short	3480000	108000	607950	UDP/IP	173600
Int	3552238	85950	607320	FIFOs	8600
Long	3432835	86400	607680	PIPEs	447350
Float	1116000	355500	684000	Shared RAM	268000
Double	1011940	334500	690000	Stream Pipe	174150
				DataGram Pipe	199000
Disk/Filesystem I/O				Library System	
(Kbytes per Second and Ops per Second)				Various Rates	
	Cashed	Sync		Setjmp/Longjmp	15948500
Rand Read	33971	80		Ram Copy	-2147483648.0
Seq Write	67764	87		Numeric Rates	1267650
Rand Write	14490	84		Trig Rtas	680000
Searches		7350		String Rtns	2985
Copy		45850		Mem Rtns 1	2130000
Seq Read		35840		Mem Rtns 2	744000
Create/Close		31840		Sort/Searc	1360
Link/Unlin		9135		Signals	128500
Directory		1665000		Misc. Rtns	5570
Memory & Process Management				Shell Rtns 1	52
Ops per Second				Shell Rtns 2	55
Brk()	532835	Paging	83300	Shell Rtns 2	54
Exec ()	164	Fork ()	952	Algorithm Tests	
Function Call				Operations per Second	
Ops per Second				Newton/Raphson	-1.0
0 Args	115968000	1 Arg	265984000	3D projection	2413250
2 Args	236288000	15 Arg	52480000	Linear System	815
				Taylor Series	9485750
				Integer Sieve	67

FIGURE 1.A and FIGURE 1.B

FIGURE 2.A and FIGURE 2.B

FIGURE 3.A and FIGURE 3.B

FIGURE 4.A and FIGURE 4.B

FIGURE 5.A and FIGURE 5.B