Solaris Zones: Operating System Support for Server Consolidation

Andrew Tucker
David Comay
Sun Microsystems, Inc.

A growing number of users are interested in improving the utilization of their computing resources through consolidation and aggregation. Consolidation is already common in mainframe environments, where technology to support running multiple applications and even operating systems on the same hardware has been in development since the late 1960's. Such technology is now becoming an important differentiator in other markets (such as Unix/Linux servers), both at the low end (virtual web hosting) and high end (traditional data center server consolidation).

Zones are a new operating system abstraction for partitioning systems, allowing multiple applications to run in isolation from each other on the same physical hardware. This isolation prevents processes running within a zone from monitoring or affecting processes running in other zones, seeing each other's data, or manipulating the underlying hardware. Zones also provide an abstraction layer that separates applications from physical attributes of the machine on which they are deployed, such as physical device paths and network interface names.

Much of the previous work in this area has involved running multiple operating system instances on a single system, either through the use of hardware partitioning [1, 4] or virtual machine monitors [2, 3, 8]. Hardware partitioning, while providing a very high degree of application isolation, is costly to implement and is generally limited to high-end systems. In addition, the granularity of resource allocation is often poor, particularly in the area of CPU assignment. Virtual machine implementations can be much more granular in how resources are allocated (even time-sharing multiple VM's on a single CPU), but suffer significant performance overheads. With either approach, the cost of administering multiple operating system instances can be substantial.

More recently, a number of projects have explored the idea of virtualizing the operating system environment, rather than the physical hardware [5, 6, 7]. These efforts differ from virtual machine implementations in that there is only one underlying operating system kernel, which is enhanced to provide increased isolation between groups of processes. The result is the ability to run multiple applications in isolation from each other within a single operating system instance.

Zones build on this concept by extending this virtual operating system environment to include many of the features of a separate machine, such as a per-zone console, system log, packaging database, run level, identity (including name services), and interprocess communication facility. For example, each zone has a virtualized view of the process table (as reflected in the `/proc` file system) that reflects only the processes running in that zone, as well as a virtualized `/etc/mnttab` file that shows only file system mounts within the zone.

A set of administrative tools have been developed to manage zones, allowing them to be configured, installed, patched, upgraded, booted, rebooted, and halted. As a result, zones can be administered in a manner very similar to separate machines. In fact, some types of administration are significantly easier; for example, an administrator can apply a patch to every zone on a system with a single command.

A zone can either be bound to a dedicated pool of resources (such as a number of CPUs or a quantity of physical memory), or can share resources with other zones according to defined proportions. This allows the use of zones both on large systems (where dedicated resources may be most appropriate) and smaller ones (where a greater degree of sharing is necessary). It also allows administrators to make appropriate tradeoffs depending on the relative importance of resource isolation versus utilization.

Zones provide for the delegation of many of the expected administrative controls for the virtual oper-

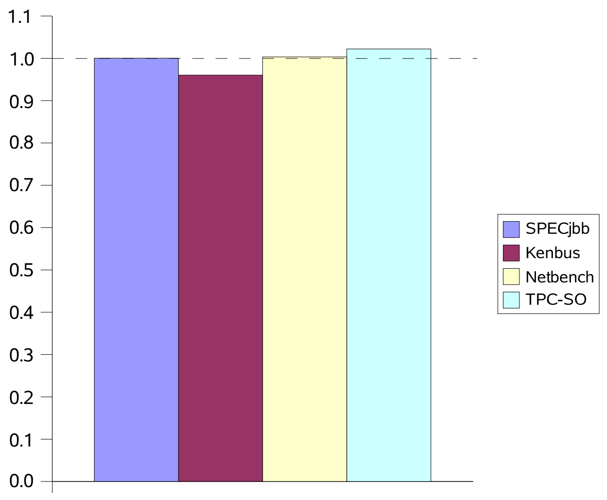

Figure 1: Normalized Performance within a Zone

ating system environment. Since each zone has its own name service identity, it also has its own notion of a password file and its own `root` user. The proportion of CPU resources that a zone can consume can be defined by an administator, and then that share can be further divided among workloads running in the zone by the (potentially different) zone administrator. In addition, the privileges available within a zone (even to the root user) are restricted to those that can only affect the zone itself. As a result, even if a zone is compromised by an intruder, the compromise will not affect other zones in the system or the system as a whole.

Zones also allow sharing of file system data, particularly read-only data such as executables and libraries. Portions of the file system can be shared between all zones in the system through use of the read-only loopback file system (or `lofs`), which allows a directory and its contents to be spliced into another part of the file system. This not only substantially reduces the amount of disk space used by each zone, but reduces the time to install zones and apply patches, and allows for greater sharing of text pages in the virtual memory system.

Figure 1 shows the performance of a variety of workloads (Java application server, time-sharing, networking, and database) running in a zone, as compared to the same workloads running without zones. As can be seen from the graph, the performance impact from using zones is small to nonexistent.

Zones are being developed as part of the *N1 Grid Containers* feature in Solaris 10, planned for general release in late 2004. An early version of Solaris 10 (which includes an initial implementation of zones) is available for download from `http://wwws.sun.com/software/solaris/10/`.

References

[1] Alan Charlesworth et al. The Starfire SMP interconnect. In *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, 1997.

[2] Paul Barham et al. Xen and the art of virtualization. In *Proceedings of the 19th Symposium on Operating Systems Principles*, 2003.

[3] P. H. Gum. System/370 Extended Architecture: Facilities for virtual machines. *IBM Journal of Research and Development*, 27(6), 1983.

[4] IBM Corp. *Partitioning for the IBM eServer p-Series 690 system.*

[5] Poul-Henning Kamp and Robert Watson. Jails: Confining the omnipotent root. In *2nd International System Administration and Networking Conference (SANE 2000)*, May 2000.

[6] `http://www.linux-vserver.org`.

[7] `http://www.virtuozzo.com`.

[8] Carl Waldspurger. Memory resource management in VMware ESX server. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.