SLICING AND DICING SERVERS
A GUIDE TO VIRTUALIZATION AND CONTAINMENT TECHNOLOGIES

Harry J. Foxwell, U.S. Client Solutions
Isaac Rozenfeld, U.S. Client Solutions

Sun BluePrints™ OnLine — October 2005

Part No 819-3734-10
Revision 1.0, 10/20/05
Edition: October 2005

Please
Recycle

Table of Contents

Slicing and Dicing Servers — A Guide to Virtualization and Containment Technologies

Part of an emerging family of containment technologies, *server virtualization* is designed to help reduce *server sprawl* — the proliferation of individual hardware servers and accompanying management and resource allocation problems. Today, IT managers and executives are starting to consider a variety of virtualization and containment technologies available on Microsoft Windows, Linux, the Solaris™ Operating System (Solaris OS) and other environments. There is also renewed interest among industry and academic researchers in this area [1], [2], as virtualization is a key technology in the deployment of both computational and business service grid architectures [3]. However, significant confusion remains regarding the terminology and techniques involved, as well as the trade-offs among the range of current solutions.

This Sun BluePrints™ article focuses on the motivation behind server-oriented containment and virtualization — *secure, efficient, and cost-effective workload management* — and discusses the concepts, vocabulary, and techniques currently available to help achieve it. Other forms of virtualization, such as those used for storage and networks, are not discussed. Directed at IT managers, CIOs, and CTOs responsible for computer resource allocation decisions, this article assumes general familiarity with IT infrastructure and management issues, and provides an overview of various solutions. Detailed technical knowledge of the techniques presented is not required. The first section reviews the requirements and challenges of workload management. Subsequent sections discuss the origins of virtualization and containment, currently available solutions and trade-offs, and a brief discussion of future technologies.

Workload Management — Requirements and Challenges

For every complex problem, there is an answer that is clear, simple, and wrong. — H. L. Menken

Deploying applications, selecting appropriate server resources to support them, and managing the resulting environment is a complex problem. Many IT managers take a simple approach — assign each application its own server. Why? They do not want applications to interfere with each other in any way, and perceive this can only be accomplished through dedicated, application-specific hardware servers. This belief may be motivated by mistrust of the application, mistrust of other users or applications that could potentially share the same server, not wanting to put too many eggs in one basket, or other technical and organizational reasons.

Figure 1 depicts a typical non-virtualized server. A single operating system per server directly initializes and controls all hardware. The result of this simplistic approach is often *server sprawl* — a large number of servers that are typically under-utilized, are difficult to manage effectively, and increase requirements for data center space, cooling, and power. Server utilization problems are common when expected application demand and peak loads placed on the server are variable and uncertain. As a result, organizations tend to over compensate. Most servers are considerably oversized — and therefore significantly under-utilized, with utilization rates as low as five to 15 percent — for much of their deployment life. Conversely, if a server is overcommitted, spare resources from other systems cannot be easily transferred to ease the problem.

Complicating this situation are additional requirements for data backup, high availability, and keeping both applications and operating systems up to date.

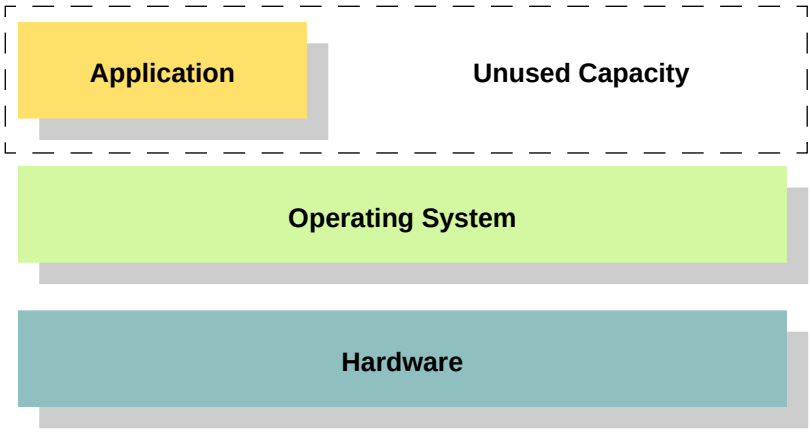


Figure 1. A non-virtualized environment

A variety of hardware and software technologies has evolved to help address these problems. Nearly all of these solutions involve some form of containment. *Containment* is a concept. An online search for a definition of the term results in a variety of definitions with a common theme — the prevention of spreading material or effects beyond a barrier or boundary. A *container* is a specific implementation of the containment concept. For example, a bottle is a container for liquid; it prevents liquid from spreading into places in which it is not wanted, and at the same time protects the liquid from contaminants. When applied to computing environments, a server can be thought of as one type of container for an application environment. The boundaries of the server prevent the application from affecting other systems, and protect the applications running on it from most external effects. These boundaries may be physical (hardware-based) or implemented in software. Examples of containment technologies include physical server partitioning, such as IBM's physical partitions (PPARs) and Sun's Dynamic System Domains, and software-based solutions such as VMware (an EMC company), BSD Jails, Microsoft Virtual Server 2005, and Solaris™ Containers. These and other forms of containment are discussed in more detail in the following sections.

Containing Services

In computing environments, it may be important to contain applications, processes, groups of users, and possibly complete operating systems. Each of these categories can be thought of as a *service*, a long-lived set of software objects with well-defined states, error boundaries, start and stop mechanisms, and dependency relationships to other services. A service must be viewed and managed — that is, *contained* — as a single entity. A container is therefore a *bounded environment* for a service; such environments can be implemented and managed using a wide variety of hardware and software technologies.

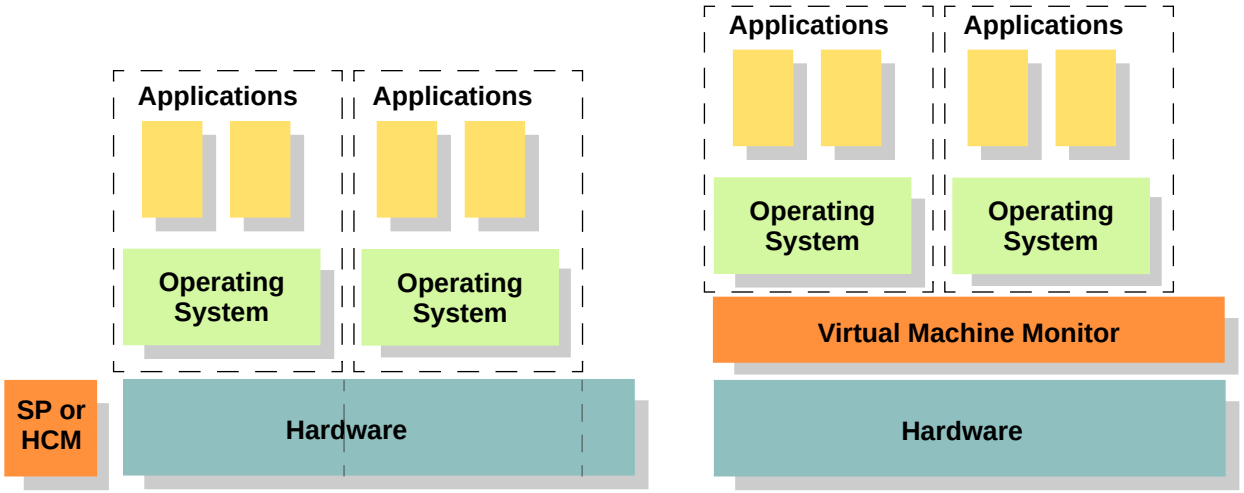
Ideally, container solutions should provide:

- *Resource containment*, the ability to allocate and manage resources assigned to the container, such as CPUs, memory, network and I/O bandwidth
- *Security containment*, the bounding of user, namespace, and process visibility, hiding activity in each container from other containers, limiting unwanted process interaction, and limiting access to other containers
- *Fault containment*, hardware errors (failed components) and software errors (such as memory leaks) in one container should not affect other containers
- *Scalability*, the ability to exploit enterprise class systems by creating and managing a potentially large number of containers without significant performance overhead
- *Flexible resource allocation*, the ability to share resources from a common pool or to dedicate resources to specific containers
- *Workload visibility*, the ability to view system activity both from within the container and from a global system perspective
- *Management framework*, tools and procedures to create, start, stop, re-create, restart, reboot, move, and monitor containers, as well as provisioning and versioning
- *Hardware independence*, where possible, containment technologies should not require special hardware
- *Native operating system support*, solutions should not require a custom or ported kernel, as this has an impact on ISV supportability

Approaches to Containment

The first general purpose mainframe computers were very large, very expensive, and relatively few in number. As a result, utilization and efficiency were critical. These systems were designed to run many tasks simultaneously using more than one operating system at very high (over 90 percent) utilization rates. To provide multiple independent and contained execution environments, mainframes used hardware configuration managers (HCM) or virtual machine monitors (VMM), also called hypervisors. These programs interact directly with the system hardware, and sometimes run on a dedicated component known as a service processor (SP).

The HCM and VMM software enable system hardware to be partitioned into multiple containers. The term *virtual* is used to indicate that access to the physical hardware is abstracted to hide implementation details. For example, a VMM-controlled container accesses system resources, such as disks and network cards, through interfaces presented by the VMM rather than direct communication with the device. When running on specialized hardware, these containers can be fully isolated and independent environments capable of being separately powered, configured, booted, and administered. Figures 2a and 2b illustrate the different approaches taken by HCM- and VMM-based techniques.



Figures 2a and 2b. Hardware configuration managers and virtual machine monitors take different approaches to controlling hardware resources

Many vendors utilize a similar approach, including the IBM VM/360 introduced in 1967 and its descendants, the IBM VM/370 and current IBM z/VM systems [4], as well as Sun's Dynamic System Domains [5]. Containers constructed in this manner are called *hardware domains* or *partitions*, and may support different operating systems, or different releases of the same operating system, in each partition. Creating these partitions typically requires specialized hardware, and the number and size of partitions supported on a given server may be limited. For example, the Sun Fire™ E25K Server [6] supports a maximum of 18 domains; the smallest domain must use at least a two or four processor system board, and each domain requires its own boot device and network connection.

The purpose of this and other containment techniques is to enable safe and efficient workload management on a single, physically shared resource. Safe—because any solution should, by definition, have the means of being secure. Efficient —because sufficient tools and technologies that aid in the effort already exist. Technologies like Sun's Dynamic System Domains and IBM's Dynamic Logical Partitions (LPARs) are designed to work in conjunction with resource management and partition reconfiguration services. These services permit the reallocation of CPU and other resources from one partition to another in order to balance loads and improve utilization. Reallocation is dynamic, meaning the reassignment of a resource from one partition to another partition typically does not require shutting down the operating system or applications running in the partitions. Operating systems and applications are fully contained within their respective partitions, and effectively run on separate hardware servers, albeit within the same physical system enclosure. What happens within one partition — resource consumption, application misbehavior, security issues, and hardware faults — generally has no effect on other partitions. When deploying multiple containers on a shared resource, any single point of failure must be recognized and addressed with redundant hardware and software components, such as clustering or other high availability solutions.

The popular blade server architecture can be viewed as a kind of hardware partitioning for large numbers of similar or identical applications, such as Web servers. Each blade is an individual hardware server running a complete operating system and application set. While blade servers can address a variety of scalability and redundancy issues, individual blades are constrained by their size, typically one or two processors with limited memory. These constraints can reduce the flexibility needed to allocate sufficient CPU and memory resources for demanding applications.

Software-Based Containment

Hardware containment methods originated the 1960s and 1970s on early mainframe systems and continue today on modern, enterprise-class servers. But they almost always require specialized systems that are capable of hardware partitioning, like the Sun Fire E25K server. In recent years, however, several commercial and open source software-based containment solutions have emerged. These solutions generally do not require specialized hardware, and can run on a wide range of systems, from laptops and desktop workstations, to mid-range and enterprise-class servers. Some of these software-based containment solutions, such as VMware ESX Server [7], operate as shown in Figure 2b but do not require special hardware to run the VMM. For example, VMware ESX Server runs on the Sun Fire V40z server with AMD Opteron processors running the Solaris OS, Linux, or Microsoft Windows as guest operating environments [8].

Figure 3 describes the general architecture of software-based containment solutions that use a *hosted* VMM. In these solutions, a primary operating system runs directly on the system hardware, and a VMM runs as an application under the host operating system. VMware Workstation [9] and Microsoft Windows Virtual Server 2005 [10] are modern examples of this type of hosted VMM environment. The hosted VMM permits multiple guest operating systems, such as Linux, Microsoft Windows, or the Solaris OS, along with their applications, to run simultaneously in a contained manner on the host system. Administrative tools are provided to allocate and change resources among the guest operating systems. Additionally, applications can be run directly on the primary operating system, ignoring the VMM entirely.

Not all server containment technologies require a VMM. In fact, VMMs can consume significant CPU resources as they rewrite or redirect guest operating system code, especially when they need to intercept and redirect privileged guest operating system instructions.

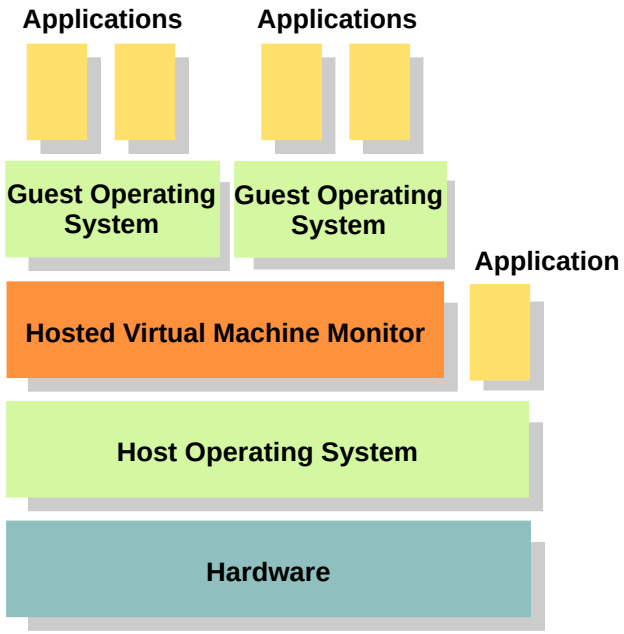


Figure 3. Software partitioning using a hosted VMM

Solaris 10 Containers

Remember that an operating system's primary task is the efficient management of processes. The operating system allocates shares of system resources, such as CPUs, memory, and I/O, and sets minimum guaranteed boundaries for the execution of the processes that use them. If a collection of processes and resources can be defined and bounded to match the requirements of a contained server environment, server virtualization can be accomplished efficiently without the use of a separate VMM. This type of containment has been described as operating system virtualization [11], and is the approach taken with Solaris™ Containers [12]. In the Solaris OS, virtual server environments are implemented using a type of container called a Solaris™ Zone. Other types of containers exist in the Solaris OS, such as projects and limit nodes. Much of the current discussion and literature about Solaris Zones treats a Zone and Container as if they were equivalent. To be clear, a Zone is one type of container, one that encapsulates a server environment, limits the effects of that environment on other system activities (including other active zones), and protects the environment from outside influence. Since a container is defined as a bounded environment for a service, and a service is a group of processes managed as a whole, then a zone is a container for the service or group of processes that implements a virtual server.

Figure 4 illustrates the concept of Solaris 10 Containers. The Solaris Operating System runs directly on the hardware, manages the boot process, and initializes interfaces to the CPUs, memory, host bus adapters, network interface cards (NICs), storage, and device drivers in the system. Only *one* instance of the Solaris OS runs on the hardware, and it is referred to as the *global zone*. The administrator defines one or more non-global zones that contain virtual server environments. A non-global zone appears to all users — end users, applications, developers, and the zone administrator — as a fully realized server with its own host name, IP address, process and name space, root and user names and passwords, and network devices

and file systems. The OpenSolaris effort illustrates Sun's intention to demystify the notion of vendor lock-in by making such technology available to the development community through an open source project [27].

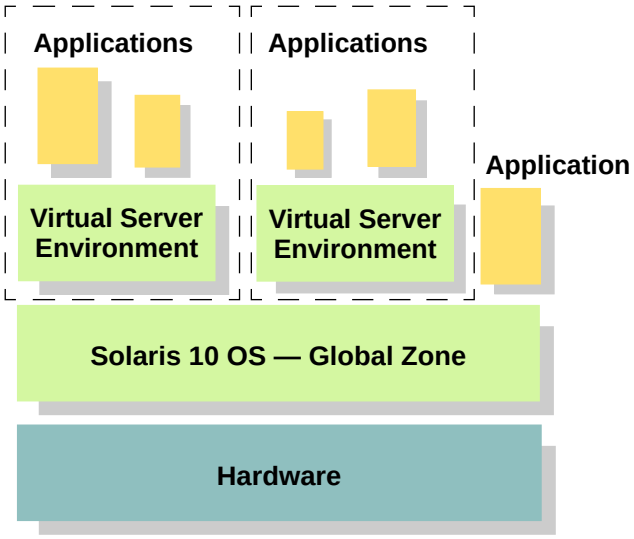


Figure 4. Solaris 10 Containers and the Global Zone

Note that Solaris applications are not required to be contained in zones. Similar to a traditional approach, applications can run directly in the global process and name space of a single operating system instance. However, the benefits of being able to contain the application in a non-global zone cannot be realized.

As multiple non-global zones are defined and implemented on a system running the Solaris OS, workload management of the non-global zones must also be considered in order to achieve the goal of secure and efficient resource allocation among multiple active zones. When defining a non-global zone, it is important to determine how resources should be allocated. Resources may be simply shared with the global zone, or dedicated to a specific zone. For example, a server with multiple network interfaces can be configured so that one non-global zone is assigned exclusive access to one interface, while all other zones share access to the remaining interfaces. The Solaris Resource Manager software, included with the Solaris OS, permits these resource assignments and Quality of Service (QoS) parameters to be defined for CPU, memory and network usage so that application demands in one zone do not affect the performance of other zones [13]. For example, the Solaris Resource Manager may be used to guarantee that database queries are assigned at least 50 percent of available system resources, or to limit compiler tasks to no more than 10 percent of available resources.

The Solaris 10 OS also addresses one of the requirements for a systemically secure IT architecture [14] by providing secure execution containers for applications. Solaris Containers run with reduced privileges within the global zone. Non-global zone processes cannot modify these privileges, load kernel modules, or alter shared read-only file systems provided by the global zone. Processes in non-global zones are fully observable and can be audited from the global zone. As a result, Solaris Containers technology provides an important building block for creating a secure IT infrastructure [15]. This can be accomplished through

the use of standardized operating environment configurations, which can help promote more consistent security and predictability while aiding organization compliance efforts.

Other Containment Technologies

The community of open source developers has contributed several containment and virtualization solutions, including FreeBSD Jails [16], User Mode Linux [17], and Xen [18]. These approaches vary widely in their security models. Some, like Xen, allow system hardware and raw disk devices to be accessed directly, potentially compromising the integrity of their containment model. Tools for monitoring and allocating container resources for these solutions are often missing or incomplete. Additionally, some solutions, such as Xen, require guest operating system kernels be modified to implement their container model. These solutions operate as previously illustrated in Figure 2b.

It is worth mentioning at this point one other common form of containment and virtualization — the Java™ 2 Enterprise Edition (J2EE) environment. Used extensively in Web-based solutions, J2EE implements the same general concepts using much the same terminology just discussed. Java™ technology provides a hardware-independent virtual machine, called the Java™ Virtual Machine (JVM), which executes Java programs in a contained environment. When implemented in a server environment such as the Sun Java™ Enterprise System Application Server [19] or IBM's WebSphere Application Server [20], Java applications run in a J2EE container [21] which provides security, namespace, and resource containment, as well as application management services.

As organizations continue to focus on greater operational efficiency through server consolidation and footprint and cost reduction, containment and virtualization technologies are attracting attention. In addition, many enterprises are looking to reduce heat output and electricity consumption, as well as the overall equipment population in their datacenters. As grid solutions evolve, virtualization of the operating system and application environments will become more widespread and include ad hoc container creation for arbitrary executables [22]. Finally, as processor architectures continue their trend toward multiple CPU cores per chip and multiple execution threads per CPU [23], it may one day be possible to divide a single chip into multiple containers capable of running different operating systems simultaneously. Indeed, one of Sun’s partners, Advanced Micro Devices, has made significant advancements in chip-based virtualization technology [24].

Containment and Virtualization Trade-Offs — When You Need It, When You Don’t

Now that you have a general idea of what containment and virtualization are all about, how do you decide which technologies to use? It is important to contain application environments in some way to ensure security and stability. Virtualization technologies can help:

- Manage the task of resource allocation among these environments
- Enable consolidation of server workloads
- Allow for hosting of untrusted or hostile applications,
- Provide flexible development, testing, and production environments.

Additionally, some virtualization technologies provide a repeatable way to troubleshoot a contained application service from a global view of the operating system, without affecting other application services. For example, the Solaris™ Dynamic Tracing (DTrace) [23] facility available in the Solaris 10 OS provides built-in instrumentation and observability of the kernel and applications, reducing the time it takes to identify and correct performance and stability problems.

The following questions may prove helpful when considering virtualization solutions:

- Are existing servers over-provisioned and under-utilized?
- How long does it take the organization to install, configure, and deploy a new server or application environment?
- Does the number and variety of hardware servers in use make it difficult to manage them all effectively?
- Is there a need to run a variety of operating systems, such as the Solaris OS, Linux, and Microsoft Windows?
- Is there a need to run different versions or patch levels of one operating system family?
- Is there a need run multiple instances of the same operating system?
- Is there a need to quickly create and reconfigure server environments for testing, development, and production?
- Is the expertise and staff available to support multiple, virtualized instances of several operating systems?
- Are in-house and commercial applications supported in a virtualized environment? What are the licensing costs and pitfalls?
- Do the virtualization solutions include tools for monitoring and managing contained applications and operating systems?
- What is the cost of the virtualization solution?
- What is the performance impact or overhead of the solution?
- How will virtualized systems be backed up?
- How can you create redundant or highly available configurations using virtualized systems?
- How can you update and maintain multiple virtual operating systems and applications?

Virtualization technologies are maturing, enabling them to provide contained environments for quickly creating and testing applications and operating systems, guarantee application quality of service (QoS), and increase overall system utilization and return on investment (ROI). At the same time, some virtualization technologies can add complexity to the overall IT infrastructure, increase licensing and administration costs, potentially add system overhead, and make diagnosis of system problems more difficult. For example, consider an environment in which Oracle software is run under VMware on the Linux operating system on a Sun V40z server. If a problem is experienced, how do you identify its cause? Which vendor is ultimately responsible for helping you find and fix the issue?

Solaris 10 Containers address many, but not all, of these questions and issues. The Solaris OS includes Containers at no additional cost, as well as resource allocation and management tools.With the Solaris OS, there is only one operating system image to update and maintain. All application processes in non-global zones can be observed, controlled, and audited, and secure workload management capabilities are built in to the operating system. Moreover, exploiting Solaris Containers on Sun hardware that supports

Dynamic System Domains can provide even greater flexibility in deploying fully contained and manageable application environments while getting maximum utilization from hardware investments. Additional tools, such as the Solaris Container Manager 1.1 software [25] are now available to help system managers deploy Solaris Containers.

The references listed at the end of this document provide additional detail on the wide range of current and evolving containment and Sun virtualization technologies. Readers interested in these technologies should pay particular attention to the Sun BluePrints articles by Lageman [11], and Collier-Brown [12].

Selecting Container Technologies

The biggest factor impacting the selection of one containment model over another is the operating environment for which the application to be contained has been written. Facts that affect infrastructure decisions are often taken into account late in the process — after the application design and development phases have been initiated. Operational business drivers that are applicable in the search for the right containment model, such as footprint and cost reduction, may appear to have been overlooked. By overlooking these types of business requirements, decision makers end up being limited in terms of choice, with the resulting performance impact that choice inherently implies. The above holds true for cases where an application is written for one platform environment, and the cost of porting or migrating it to another cannot be justified or is otherwise not an option.

Once the operating system — the core software foundation for most application services, excluding specialty appliances — has been agreed upon, options for containment models are limited and tend to be vendor-specific. At times, the selection of an application development platform is driven by existing hardware investments. In this case, the decision-making flowchart can be shortened due to the elimination of non-applicable containment models.

These are not the only important factors. Security and effective resource management are also essential. The benefits of consolidation are best seen in high end servers, where compute resources can be dynamically re-allocated on an as-needed basis between various versions of operating systems. Furthermore, resource usage can be reported and fed into a chargeback system. This in turn enables business groups to validate budgets and plan hardware acquisitions.

On the other hand, large scale search engine architectures[26] place a heavy emphasis on home-grown software layers in an effort to provide redundancy and price/performance of low-cost commodity hardware. The notion of containment, therefore, may not apply in the traditional sense. Instead, containment is achieved through an architecture that relies on a collective design of many load-balanced low-cost servers clustered together. As a result, a problem with a particular node is self-contained. This type of problem can be easily remediated by a new set of hardware that runs the same software layers as thousands of other nodes.

So what can you do? Ask the following question: What are my application's behavioral trends and needs? This is an important question because the answer does not imply favoritism of one type of containment or another. It is, however, an answer that is usually quite difficult to obtain, and may require months of performance data analysis, or weeks of interviews with application developers and systems administrators.

If appropriate, projected business growth should also be taken into account. Getting the correct answer may be challenging, but will prove helpful when navigating the maze of containment choices. While it is important to consider pure CPU resource requirements, network and storage demands must also be considered. Several network ports may need to be present on the underlying hardware architecture in order to have a redundant configuration. Multiple alternate paths to data storage may be required. These two facts alone may steer the requirement toward a larger server that is capable of hosting multiple operating systems concurrently. Consider the following requirements:

- *Support multiple operating systems and applications*

  If there is a need to contain various operating systems types and applications by allowing more efficient use of underlying x86 hardware, one option may be to rely on VMware as the VMM, letting other operating systems be configured as guest images. This enables versions of the Solaris OS, and Linux and Microsoft Windows operating environments to co-exist on the same hardware platform. Note that the management of such a system, and the ability to troubleshoot performance problems when all three operating environments are active, may prove challenging.

- *Reduce costs*

  Cost reduction will always be a transcending requirement in any technical solution. Therefore, if the requirement circles back to a single hardware platform that can be physically partitioned using HCMs, then IBM's Logical Partitions or Sun's Dynamic System Domains can be utilized for UNIX applications requiring high availability.

Table 1 outlines several virtualization guidelines that can aid the decision making process.

Table 1. Virtualization guidelines

If You Have...	Then...
Multiple Solaris OS or Open Source Applications	Consolidate using Solaris Containers
Mixture of Linux and Solaris OS Applications	Consider migrating Linux applications to Solaris OS Consolidate Solaris OS applications using Solaris Containers on one server, and Linux applications using VMware ESX virtual machines on another server
Microsoft Windows and/or Linux Applications Running on Different Versions of Operating Systems	Consolidate onto a single server using VMware ESX virtual machines
Applications on Different Operating Systems and Architectures from IBM	Consider migrating to Solaris Containers or consolidate using IBM LPARs

The Pros and Cons of Virtualization Technologies

Looking at the pros and cons of each approach is also important when selecting a virtualization technology and approach.

- *Platform availability*

  Different platforms provide different levels of availability. VMware only supports 32-bit applications on the x86 and x64 processor set. Solaris Containers operate on x86, x64 and UltraSPARC® processors. IBM's LPARs can only operate on IBM Power-based servers and mainframes.

- *Performance*

  If overhead is a concern, VMware or LPARs may not be an option. The layers of software and different operating system kernels present in these software architectures may add additional overhead and slow overall system performance.

- *Manageability*

  If data center manageability is a crucial factor, then Solaris Containers are a solid choice. The operating system standardization they ensure for a large percentage of applications helps ease the management burden, and footprint reduction is possible with high end servers. In contrast, VMware and LPARs primarily address footprint reduction while fostering a diversity in the data center that can significantly increase the management effort.

- *Isolating software problems*

  If being able to isolate software problems is important — and it commonly is — then observability is essential. If an application in a VMware guest operating system is not behaving as expected, it may be difficult to gain an understanding of what is happening in the operating system with respect to the entire platform and the foundation environment. With Solaris Containers, the global zone allows complete visibility into all containerized application services and the underlying hardware.

- *Containing Microsoft Windows legacy applications*

  Containing legacy Microsoft Windows applications while staying on the Microsoft Windows platform, Microsoft's Virtual Server 2005 [10] enables the creation of virtual machines on top of the Windows 2003 Server operating system. This approach may make sense if the management and performance tools available on Windows 2003 Server are in line with delivering ease-of-management and hardware sharing between various instances of Microsoft Windows environments being contained. Some examples of this approach may include applications that can not be ported from legacy Windows NT 4.0 environments. As with VMware and Solaris Containers, Microsoft's virtual Server enables the creation of virtual network and virtual disk devices, providing containerized I/O access to applications. Microsoft also offers a Virtual Server Migration Toolkit (VSMT) that can be used to aid the migration of applications from legacy Microsoft Windows environments. While Microsoft provides the technical capability to run different Windows operating systems and applications on the same physical computer, IT managers should consider the trade-offs of cost, management and performance of running legacy Microsoft Windows operating system versions concurrently on the same server.

The Cost of Container Technologies

One of the elements affecting the selection of a technical solution is the cost associated with acquiring and maintaining operation of the technology. Some operating systems provide technology at no additional cost, while other vendors make such technology their primary business. Notwithstanding hardware and support costs, application licensing costs can often be a significant factor and also need to be considered. Some database vendors offer pricing options that consider the quantity of processors on which the application would execute as a means of deriving the cost of the license. This approach favors optimized, container-

based environments — an Oracle database can be setup to execute on only a subset of all processors available in the entire server.

In the case of an Oracle database, both soft and hard partitioning schemes are available. *Soft partitioning* segments the operating system using resource managers, and gives database administrators the ability to set the number of processors to the exact number being licensed. The virtualization of application execution environments is not required. In contrast, *hard partitioning* takes software partitioning one level further by considering the underlying operating system's virtualization capabilities, including the ability to assign unique user accounts, IP addresses, file systems and network interfaces to various container environments present on the server. With the Solaris 10 OS, a container can utilize all the resources available to the global zone. This default behavior does not take into account CPU pools, enabling all available CPUs to be used more effectively by processes running in the operating system (Figure 5).

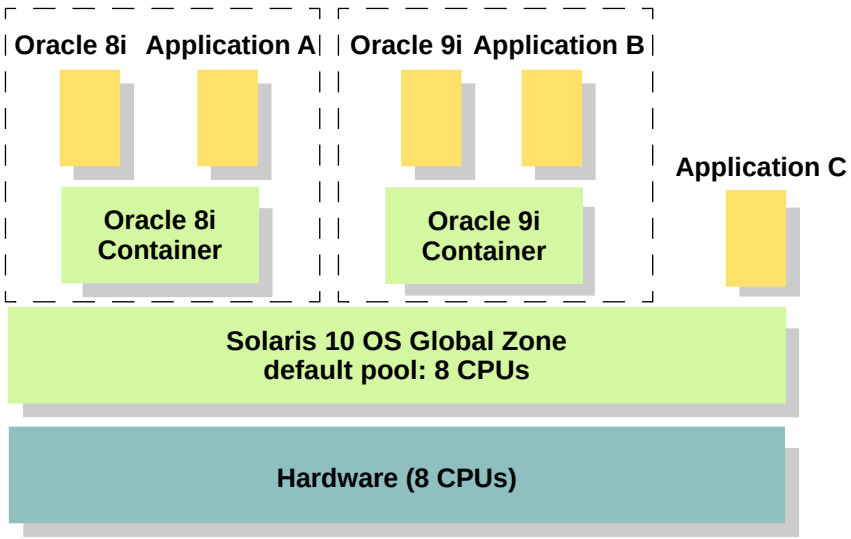


Figure 5. Databases in Solaris Containers

Furthermore, a container can be bound to a resource pool comprised of a subset of CPUs. Figure 6 illustrates such a scenario. A resource pool is defined to have a minimum of one, and a maximum of three CPUs, and an Oracle8i container executing within its boundaries. The remaining 5 CPUs are used for executing the Oracle9i software, as well as applications A, B and C, or just application C alone. As a result, the Oracle8i software need only be licensed for three CPUs, and the Oracle9i software for five CPUs — even though there are eight CPUs in the system. Solaris Containers allow applications to be assigned a dedicated set of CPU resources, and ensure those applications never receive access to more resources than they are entitled. Security is built-in by leveraging the Solaris OS privilege and process rights management model, which is a key differentiator compared to other container security models [15].

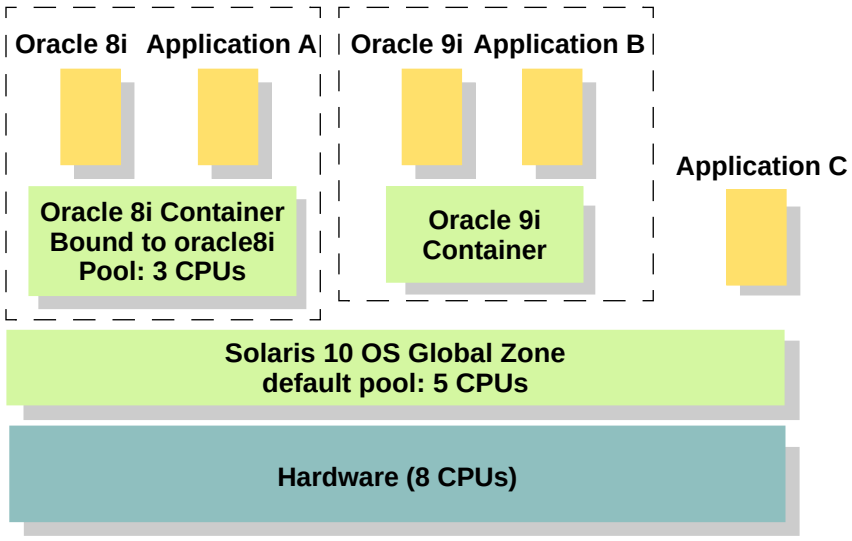


Figure 6. Resource Pools and Solaris Containers

Summary

Containment and virtualization technologies are beginning to play a critical role in IT infrastructure design decisions. It is important for IT managers, CTOs, and CIOs to understand these evolving concepts and solutions, and to know what questions to ask about their potential benefits and pitfalls. Both hardware-and software-based solutions should be part of an overall strategy for reducing server sprawl and increasing resource utilization in a systemically secure manner. This Sun BluePrints article should serve as an introduction to this complex topic. Other Sun BluePrints articles, listed in the references that follow, provide more technical detail on Sun's approach to containment.

About the Authors

Harry Foxwell is a Senior Technical Specialist for the Government division of Sun Microsystems in the Washington, D.C. area, where he is responsible for solutions consulting and customer education on the Solaris OS, Linux operating system, and grid technologies. Prior to joining Sun in 1995, Harry worked as a UNIX and Internet specialist for Digital Equipment Corporation, and has worked with UNIX systems since 1979. He also maintains one of Sun's internal web sites devoted to Linux technical information, and has been influential in developing and promoting Sun's Linux operating system and x86 hardware strategy. Harry received his doctorate in Information Technology from George Mason University in May, 2003, and has since taught graduate level courses at George Mason University in operating systems and electronic commerce. He may be contacted about this Sun BluePrints article at *harry.foxwell@sun.com*.

Isaac Rozenfeld is an IT Architect in Sun’s Advanced Datacenter Practice group, focusing on the adoption of the Solaris 10 OS. Based in the New York area, Isaac participates in advisory, architecture and delivery of technical solutions at several financial services organizations. Since joining Sun in 1998, Isaac has focused on platform and resource management in the Solaris OS. Isaac received a Bachelor of Science

degree in computer science from Queens College, City University of New York. Isaac can be reached at *isaac@sun.com*. He also maintains a blog at `http://blogs.sun.com/unixman`.

Acknowledgments

The authors thank the following for their advice and contributions to this article:

- Joost Pronk Van Hoogeveen, Solaris Product Architect
- Menno Lageman, Technical Specialist

Additional valuable comments and suggestions were provided by Glenn Brunette, Mark de Groot, Holger Leister, John Meyer, Jeff Victor, and Ray Voight.

References

Containment

[1] IEEE Computer, May 2005. Resource Virtualization Renaissance special issue:
`http://www.computer.org/computer/homepage/0505/GEI/`

[2] ACM/Usenix Conference on Virtual Execution Environments, June 2005:
`http://www.veeconference.org`

[3] Strong, Paul, Enterprise Grid Computing. ACM Queue, July/August 2005:
`http://www.acmqueue.com`

[4] VM History and Heritage:
`http://www.vm.ibm.com/history`

[7] VMware ESX Server 2.5:
`http://www.vmware.com/products/server/esx_features.html`

[8] Consolidation through Virtualization with Sun's x64 Servers:
`http://www.sun.com/amd/briefs/consolidation-sol-bf.pdf`

[9] VMware Workstation 5:
`http://www.vmware.com/products/desktop/ws_features.html`

[10] Microsoft Virtual Server 2005:
`http://www.microsoft.com/windowsserversystem/virtualserver`

[11] Operating System Virtualization, Ideas International, Inc., (July 13, 2005).
`www.ideasinternational.com/TTM_Sept_05.pdf`

[12] Solaris Containers – What They Are and How to Use Them:
`http://www.sun.com/blueprints/0505/819-2679.pdf`

[17] User Mode Linux:
http://user-mode-linux.sourceforge.net/

[18] The Xen Virtual Machine Monitor:
http://www.cl.cam.ac.uk/Research/SRG/netos/xen/

[19] Sun Java System Application Server:
http://www.sun.com/software/products/appsrvr/index.xml

[20] WebSphere Application Server:
http://www.ibm.com/software/webservers/appserv/was/

[21] J2EE Containers:
http://docs.sun.com/source/819-0215/containers.html

[22] Computing As We Know It:
http://blogs.sun.com/roller/page/Gregp/20050224

[23] Throughput Computing:
http://www.sun.com/processors/throughput/

[27] OpenSolaris Community: Zones
http://opensolaris.org/os/community/zones/

Workload Management

[5] Dynamic Reconfiguration & Dynamic System Domains:
http://www.sun.com/servers/highend/dr_sunfire

[6] SunFire E6900 Server:
http://www.sun.com/servers/midrange/sunfire_e6900

[13] Creating Self-Balancing Solutions with Solaris Containers:
http://www.sun.com/blueprints/0605/819-2888.pdf

[24] DTrace:
http://www.sun.com/bigadmin/content/dtrace/

[25] Sun Management Center — Solaris Container Manager:
http://www.sun.com/software/products/container_mgr/

[26] Luiz Andre Barroso, Jeffery Dean, Urs Holzle, *Web Search for a Planet: The Google Cluster Architecture*, IEEE Computer Society, IEEE Micro, March-April, 2003.

Security

[14] Toward Systemically Secure IT Architectures
www.sun.com/software/security/docs/systemic-security-wp-1.pdf

[15] BigAdmin Feature Article: Practical Security Using Solaris
www.sun.com/bigadmin/features/articles/container_security.html

[17] Jails: Confining the Omnipotent Root:
http://docs.freebsd.org/44doc/papers/jail/jail.html

[23] Secure Virtual Machine Architecture Reference Manual:
http://enterprise.amd.com/downloadables/Pacifica_Spec.pdf

Related References

Sun and Oracle Consolidation
http://www.sun.com/third-party/global/oracle/consolidation/solaris10.html

Sun Mainframe Reshosting
http://www.sun.com/datacenter/mainframe/index.html

Ordering Sun Documents

The SunDocs℠ 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/

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:
http://www.sun.com/blueprints/online.html

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com