30 Server Consolidation: Virtualisation and Other Partitioning Techniques

If you have not read the book check the introduction to “More Tasters” on the web site http://www.b.king.dsl.pipex.com/ to understand the scope and objectives of tasters.

Hardware partitioning of a single server into multiple environments has been widely used over the last 6-7 years, albeit more typically on the larger (enterprise level) servers. Software partitioning has a longer pedigree; the ability to run multiple operating system instances on a single physical hardware server was originally provided by IBM for mainframe systems circa 25 years ago but it has generally been a forgotten subject until fairly recently. Interest has been rekindled on x86 systems mainly due to the cost implications of deploying dedicated servers for each component of a service, particularly Internet services. This taster discusses both approaches to partitioning, although it concentrates more on the software solutions.

30.1 Starting Terminology

Dynamic Domain is Sun’s name for a hardware partition. ***LPAR*** (IBM) and ***nPartitions*** (HP)

Hypervisor is an alternative name for a VMM (see below)

LPAR is IBM’s name for a hardware partition

NPartitions is HP’s name for a hardware partition

OSS (Open Source Software)

Virtual (arg!!). When reading literature on server virtualisation the term “virtual” appears in almost every sentence. It can become very difficult to understand precisely which “virtual” the author is talking about; is it from the perspective of the operating system instance (i.e. the guest operating system) or from the perspective of the VMM?!

VM (Virtual Machine). In this context it is the name that is given to an operating system instance that runs under the control of a VMM.

VMM (Virtual Machine Monitor) is the name that is typically given to the controlling software that hosts multiple operating system instances (VMs).

30.2 Technology Outline

30.2.1 Hardware Partitioning

Hardware partitioning allows the division of a physical server into multiple systems where each has its own operating system instance and can operate independently. As shown in Figure 30-1, a standard approach by the major server vendors is to allow medium to high-end systems to be partitioned at the board (sometimes called the cell) level; the diagram shows two boards, each with 4 CPUS (C) and memory modules (M). These systems are engineered such that each board, or multiple thereof, is a self-contained entity that contains a number of CPUs, memory, buses and IO connectivity. They may also have resilience features, e.g.

separate power supplies. The main advantages of this method are that it will be the best performing due to the physical separation and potentially the most resilient partitioning option. The disadvantage is the relatively coarse granularity (e.g. a minimum of (say) 4 CPUs and 16GB of memory) and hence the cost implications. It is arguably best used for DB servers, large file servers and possibly application servers.

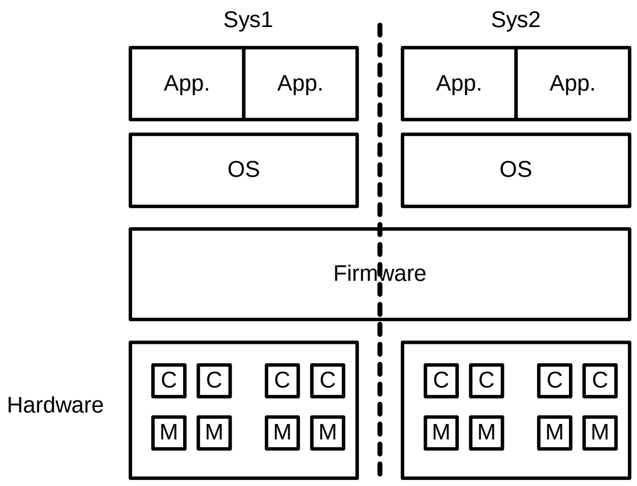

Figure 30-1 Board or Cell Level Partitioning

A refinement that is available from some vendors is to allow a finer grain by partitioning at the resource level, e.g. at the individual CPU level. Figure 30-2 shows a simple splitting of one board into two partitions, each with two CPUs and 50% of the board memory.

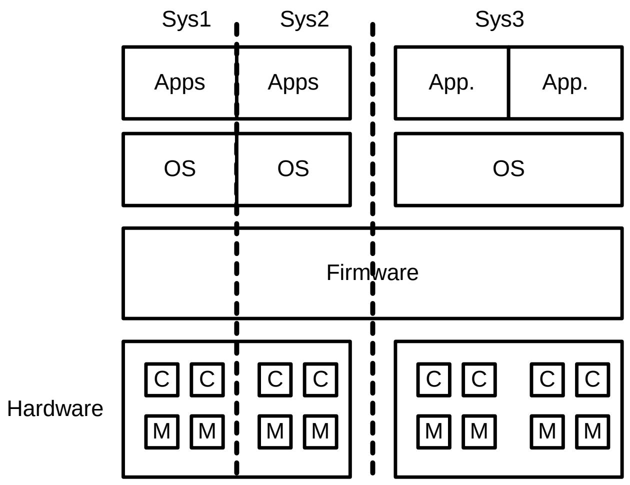

Figure 30-2 CPU Level Partitioning

30.2.2 Partitioning Within an OS

As shown in Figure 30-3 on the following page a more recent technique is to compartmentalise a single operating system instance into multiple containers (or zones). The Jail subsystem in FreeBSD is an early example of this approach. A container will typically consist of the application plus access to selected system services. In addition, it will share common parts of the kernel. The main advantage is that each container is isolated, e.g. a process in container 1 cannot see or access a process in container 2, and a problem in one of them should not affect the others. The disadvantage is that a problem in the common parts of kernel can affect all containers. An interesting example using this method is the modification of an OSS kernel to support multiple instances of a network stack.

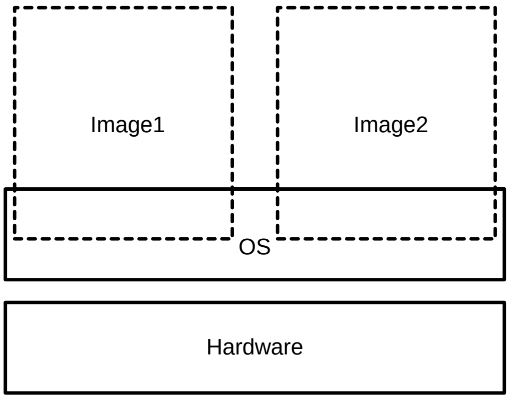

Figure 30-3 OS Containers

30.2.3 Server Virtualisation

Back in the 1960s IBM mainframes were essentially batch systems but the need was recognised to support timesharing systems. Although the OS was eventually improved to support multiple workloads types, including TSO (Time Sharing Option) running under MVS, one strand of IBM worked on a Virtual Machine Monitor (VMM) that could support a timesharing system. The VMM runs on the hardware and provides support for multiple guest operating systems (or VMs) that sit on top of it (see Figure 30-4). Each guest OS has an exact copy of the hardware, albeit virtualised. CMS (Conversational Monitor System) was the guest OS which supported a single user running timesharing services. A system call by an application is handled by the operating system that it is running on, i.e. the guest OS. If the call invokes (say) a disk IO, the attempted access by the guest OS would be trapped by the hypervisor (an alternative name for a VMM).

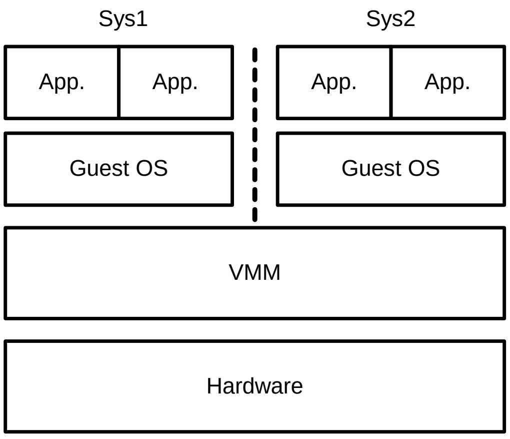

Figure 30-4 Server Virtualisation

Customers perceived another use for VM/370 as it came to be known. Mainframes were expensive machines and the majority of the client base had a single machine to support all production and test environments. The obvious dangers were the possibility of test environments, particularly when testing changes to the operating software itself, bringing the whole machine down. Therefore, the ability to support multiple environments in self-contained guests, e.g. MVS Production and MVS Test, was well received. Other uses for VM/370 included support for multiple production and test instances for DOS/VSE, IBM's smaller operating system.

Across the industry, VM/370 is now categorised as an example of full system virtualisation, which is defined as a system where an exact copy of the real hardware can be presented to the guest operating systems. The advantage of this type of system is simply that software and guest OSes can run unmodified. The potential disadvantage is the extent to which the VMM has to intervene in the running of the guest OS, particularly in the execution of privileged instructions and in virtual memory access. In VM/370 performance assists were developed, e.g. in the area of special instructions that would otherwise have to be simulated by the VMM. Virtual memory access can cause significant performance problems. In essence, there is an additional layer: when the CPU encounters a virtual address within the guest environment it has to map it to the VMM's memory space using a second page table lookup (The guest has already done a memory mapping from its own perspective). VM/370 got around this doubling-up issue by using a shadow table to map a virtual address within a guest to real memory, and by allowing the guest OS to access the shadow table directly, thus minimising the need for VMM intervention. As IBM was providing all the OSes there was minimal risk of problems caused by misbehaving guests. z/VM, as it is now known, supports Linux.

Moving forward to more recent times, the tendency, particularly on x86 platforms, has been to deploy dedicated servers for each component of an overall service. Products are more prevalent than bespoke software and the vendors tend to mandate that their software should run on dedicated hardware, as they fear the possible effects of running alongside other products. Reliability is one issue, x86 platforms do not have a particularly wonderful record, and not helped by the fact that many developers assume that they have the machine to themselves. Performance is another reason. The result of these fears is that much hardware can be deployed, particularly for network services, web servers and application servers, hardware that may be under-utilised overall. Hence the perceived market for products that could reduce the overall requirement for hardware by allowing multiple items of software to co-reside on a single physical server. However, IA-32 was not developed with virtualisation in mind. Examples, over and above the standard VMM issues that were discussed above, include:

- There is a small set of important instructions that do not have to be executed in "privileged" mode, as they would on other platforms, which can affect the stability of the system

- The ability of the platform to support a large range of devices makes virtualisation more complex.

First appearing in 1999, VMware is an example of a commercial product that provided full system virtualisation on IA-32. They now provide several variants, ESX being the most performant version.

Various researchers have investigated the modification of the guest OS, typically commodity operating systems such as Linux or Windows, to allow satisfactory performance. This approach has been given the soubriquet “Paravirtualisation”. Examples of this approach include Xen and Denali. Paravirtualisation techniques vary but they include:

- **Guest OS idle loops**. Rather than sitting on a busy wait, as many OSes do, an instruction is executed that causes a context switch to the VMM so that it can schedule another guest OS
- **System calls registered with a fast handler** which allows the guest OS to access the processor directly, avoiding going via the VMM
- **Guest OS** runs at a lower privilege level than the VMM
- **minimise virtual memory issues** by only allowing one address space per guest
- **improve memory performance** by allowing the guest OS to have read access to hardware page tables. The VMM handles page updates which can be batched up by the guest OS to reduce the overheads
- **Interrupt handling.** The additional overheads of passing an interrupt from the VMM to the responsible guest can make the approach of queueing interrupts attractive, rather than attempting to handle it immediately
- **Provide generic IO devices** to circumvent the problem of having to support every device under the sun. An alternative approach is to run the VMM as part of an OS, e.g. as a Windows service, thus making use of the standard OS to support devices. This is sometimes called a hosted operating system.

30.3 Performance

To summarise some of the points that have been made in the previous section:

- Hardware partitioning is the best but most expensive performer. The most likely use is for DB, file and application servers
- In server virtualisation, a plain vanilla VMM is likely to impose a significant performance overhead, necessitating thoughtful design changes to make performance acceptable
- The performance of OS partitioning is likely to depend very much on the degree to which use of common parts of the kernel causes locks.

Resource Management is an important topic, i.e. the ability to share the physical resources fairly or via specified rules among the guest OSes. For example, it should be possible to share the available CPU(s) on a fixed or proportional basis. In the latter case, the usage will depend on how many guests are active at the time. Each guest may be allocated a number of shares. If a guest has two shares out of a total of eight shares it will receive a minimum of 25% of the available CPU.

It is important to understand if multiprocessor support is likely to be required. At the time of writing some VMMs only support a single physical CPU. Some VMMs will support multiprocessors but any guest OS is limited to using a single CPU.

30.4 General Observations

Server virtualisation is undoubtedly useful in development and test environments where low usage is the norm, with the possible exception of stress testing.

As for production environments, it is necessary to ensure that a single server will support the throughput requirements of multiple guests. Marketing literature makes out that you only use 5% of CPU, according to studies! Check this out. Understand the timing and scope of the peaks of each workload to see precisely what the machine will have to cope with. Also, check that the required service levels can be achieved and maintained; I have come across application software that provides satisfactory response times at low CPU utilisations, say 20-30%, but not at higher levels. This can be due to a mixture of hardware queueing and software design issues.

There are few published benchmarks in the server virtualisation area. I understand that the licensing agreements of some commercial vendors forbid the publication of any benchmarks involving their product. What there is tends to be limited to running a single instance of a workload. While this type of testing is essential to understand the basic overheads of virtualisation and its effects on performance, more representative testing that involves multiple guests running multiple workloads is required. This means that it is the responsibility of the customer to perform benchmarks in any areas where doubts exist on performance.

High Availability is not easily supported in a virtualised environment. Within the confines of a single physical server software failures can be handled for simple N+1 clusters where no state information is maintained. More complex clustering arrangements where services are spread across multiple physical servers are problematic at this time and will require careful investigation.

30.5 Further Reading

Rose, R., *Survey of System Virtualisation Techniques*, available on the web

Barham, P., Dragovic, B., Fraser, K., Hand, S. et al, *Xen and The Art of Virtualisation*,

Clark, B., Deshane, T., Dow, E., Evanchik, S. et al, *Xen and The Art of Repeated Research*,

Olsen, K., Russell, S., Sallah, G., Seetharaman, C., Watts, D., *Server Consolidation with the IBM eserver xSeries 440 and VMware ESX Server*, ibm.com/redbooks

Whitaker, A., Shaw, M., Gribble, S., *Scale and Performance in the Denali Isolation Kernel*, University of Washington, available on the citeseer web site

Kamp, P., Watson, R., *Jail: Confining the Omnipotent Root*, FreeBSD Project, available on the web at docs.freebsd.org

Zec, M., *Implementing a Clonable Network Stack in the FreeBSD Kernel*, USENIX 2003 Technical Conference

Microsoft Virtual Server 2005 white paper, Microsoft web site

HP Partitioning Continuum, A Technical Positioning Paper, June 2002 - available on the HP web site

System Administration Guide: N1 Grid Containers, Resource Management and Solaris Zones, available on the Sun Microsystems website.