

Xen and the Art of Virtualization

P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R.
Neugebauer, I. Pratt, A. Warfield

Charalampos S. Nikolaou
charnik@di.uoa.gr

Department of Informatics and Telecommunications

January 27, 2008

- 1 Motivation
- 2 Approaches
- 3 Xen Architecture
- 4 Xen Evaluation

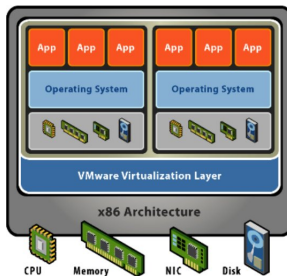
- Power Savings
- Reduce Costs
- Hardware isolation
- Legacy operating systems
- Testing
- Security and performance isolation
- Maintenance - Management

The idea

Partition a machine to support concurrent execution of multiple operating systems in one machine.

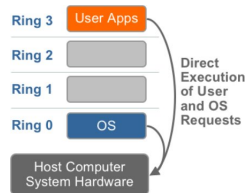
The approach behind this idea is called **Virtualization**.

In general, virtualization is implemented by a software layer between the OS and the hardware providing an abstraction more or less identical to the underlying machine.



The most privileged entity is the OS itself.
OS provides Users with a hardware abstraction
functionally identical to the physical hardware.

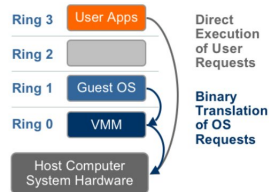
- x86 architecture offers four levels of privileges known as Ring 0, 1, 2 and 3. The level of Ring determines the degree of privilege that the current executing instruction has.
- User applications run in Ring 3, while the OS runs in Ring 0, because of the need to execute privileged instructions for accessing/managing the hardware.
- User level code is directly executed on the CPU.



Traditional

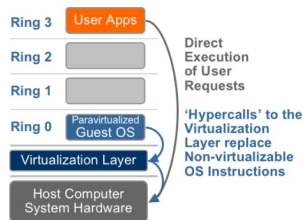
VMM (Virtual Machine Monitor) exposes its hardware as being functionally identical to the physical hardware (VMware's approach).

- VMM provides each **Guest OS** with a virtual BIOS, virtual devices and virtual memory management.
- The Guest OS is not aware it is being virtualized.
- VMM translates **binary** kernel code (on the fly) into new sequences of instructions that have the intended effect. Code is cached for future use.
- User level code is directly executed on the CPU for high performance virtualization.
- Sophisticated and difficult approach.



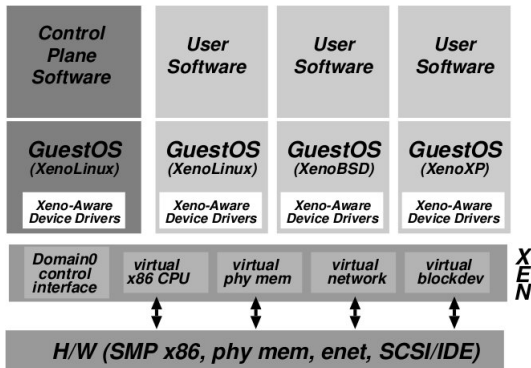
VMM exposes a hardware abstraction that is similar but not identical to the underlying hardware (Xen's approach).

- The **hypervisor** (or VMM) provides an interface to OS kernel through **hypercalls**.
- Paravirtualization involves **modifying** the OS kernel to replace certain kernel calls with hypercalls that communicate directly with the VMM.
- The Guest OS have to be modified. There might arise licence problems.
- User level code is directly executed on the CPU for high performance virtualization.
- Relatively easy approach.



Challenges to be faced

- Virtual machines must be isolated from one another:
it is not acceptable for the execution of one running Guest OS (or **domain**) to adversely affect the performance of another.
- Support of a variety of different operating systems.
- The performance overhead introduced by virtualization should be small.



The figure depicts the structure of a machine running the Xen hypervisor, hosting a number of different guest operating systems, including Domain0 running control software in a XenoLinux environment.

Execution, Scheduling, Timers

Privileged instructions are paravirtualized by requiring them to be validated and executed within Xen.

Scheduling approach:

- Uses Borrowed Virtual Time algorithm to schedule domains.
- Important to mitigate the problem of one domain executing code that can adversely affect another domain.

Provision of different types of timers:

- Real Time (time that always advances regardless of the executing domain),
- Virtual Time (time that only advances within the context of the domain),
- Wall Clock Time (time that takes into account local offsets for time zone and DST).

Control Transfer, Event Handling

Control Transfer:

- Synchronous interaction from Guest OS to Xen using **hypercalls** (e.g. I/O request).
- Asynchronous interaction from Xen to Guest OS using event-callback handlers specified by the Guest OS (e.g. I/O completion).

Exceptions and Eventing:

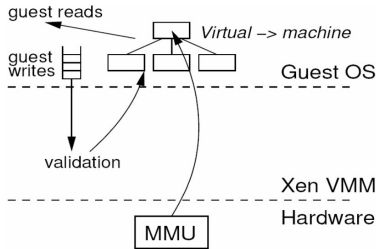
- These include memory faults and software traps.
- Guest OS registers a descriptor table for exception handlers within Xen.
- Guest OS may install a “fast” handler for system calls, avoiding indirecting through Xen on every call.

MMU / Virtual Memory

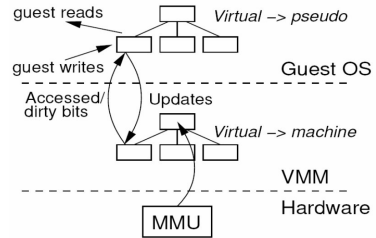
When the Guest OS requires a new page table (e.g. new process creation):

- ① it allocates it from its own memory,
- ② it is registered with Xen,
- ③ it then relinquishes all direct write privileges to the memory.
- ④ All subsequent updates must be validated by Xen.
- ⑤ The Guest OS, generally, batch these update requests to decrease the cost of calling the hypervisor.

Memory Management Paravirtualization/Full Virtualization



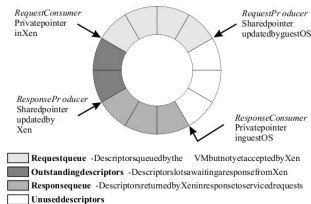
(a) MMU Paravirtualization



(b) MMU Full Virtualization

Figure: MMU Virtualization approaches

Data I/O Management



- Data I/O are transferred to and from domains via Xen through the use of a buffer descriptor ring.
- This is a system that is based around a pair of producer consumer pointers, one set used within the guest OS, the other within the hypervisor.
- This allows for the decoupling of when data arrive/are accessed and the event notification.

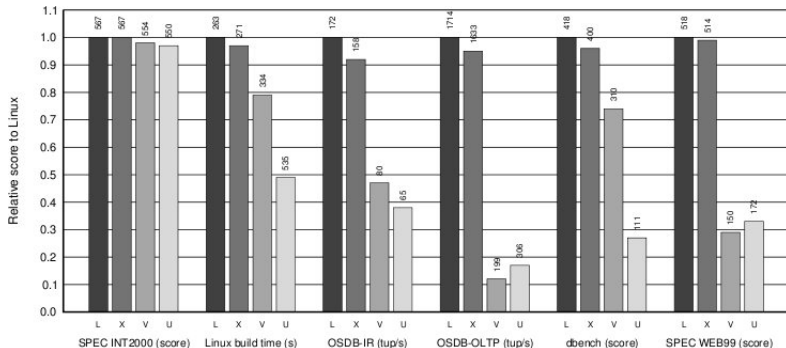


Figure 3: Relative performance of native Linux (L), XenLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

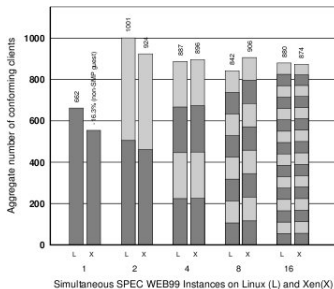


Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers; higher values are better.

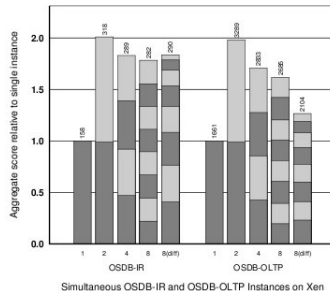


Figure 5: Performance of multiple instances of PostgreSQL running OSDB in separate Xen domains. 8(diff) bars show performance variation with different scheduler weights.

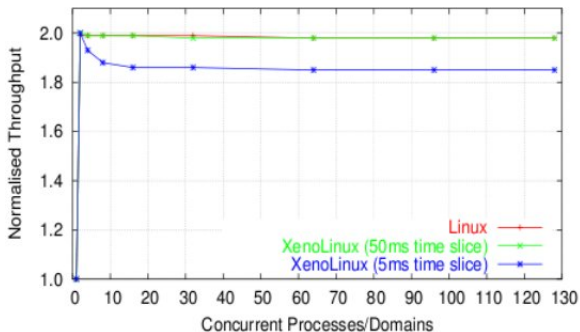


Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains




Conclusions

Xen's paravirtualization approach:

- has great performance (comparable to a native single OS system),
- is highly scalable, **BUT**
- requires OS's kernel source modification which due to licences might be impossible!

VMware in cooperation with XenSource have proposed an interface which every OS that would like to be virtualized must adhere to. Paravirtualization seems to be the winner!

References

-  P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield,
Xen and the Art of Virtualization, Proc. ACM-SOSP 2003.
-  VMWare White Paper,
Understanding Full Virtualization, Paravirtualization, and Hardware Assist.
-  The LaTeX Beamer Class Homepage.
<http://latex-beamer.sourceforge.net/>.

The End

Thank you!