Paper Review: *Xen and the Art of Virtualization*

Paper Summary

In this work, the authors designed and implemented Xen, an x86 virtual machine monitor (VMM). Xen can be used to create and manage virtual machines on single physical machine. Each virtual machine can run its own Operating system. The authors aimed with their design to host up to 100 virtual machine instances simultaneously on a one modern machine. Another goal is to support wide variety of operating systems in Xen. In addition to that, Xen as virtualization technique brought a number of advantages with and reduced the limitations existing in systems that do not use any virtualization technique.

Using virtualization concept Xen can address the issues associated with the use of systems that do benefit from virtualization. Xen gives the ability to run different operating systems on the same physical machine that will be able to run applications implemented for different operating systems. Also, Virtual machines can be stored on disk and launched at different physical machine. Additionally, Xen can be used to replace a number of physical servers with only one physical server. This concept is called server consolidation and has many advantages. In sever consolidation the number of used machines will be reduced. Consequently, the energy consumption will be reduced and costs associated with the used hardware and consumed energy will be also reduced. In addition, the hardware resources will be better utilized so that for less hardware resources more services can be provided. Also, virtual machine infrastructure can provide high availability, reliability, and failure recovery.

The authors used Para-Virtualization, one of several virtualizations techniques, in designing Xen to provide high performance. Xen was designed to run on X86 CPU architecture which has four privilege levels starting from zero, the highest level, to 3, the lowest level. The privilege zero gives the ability to execute sensitive instructions that can be directly executed on hardware such as the CPU and memory. By default the user applications run in level 3 and the operating systems use the privilege level zero to have direct interaction with the hardware. However, in proposed approach level zero is assigned to Xen to have full control on the instructions that must be executed on hardware. An operation system running in virtual machine has to run in lower privilege level than Xen. In this case, the operating systems cannot execute level zero instructions. Therefore, the operating systems must be modified in order to port them to Xen. The required modifications are replacing all level zero instructions in the operating system with calls to equivalent instructions in Xen. When the operating system need to execute any sensitive instruction, the instruction will be forwarded to Xen by calling an equivalent subroutine in Xen which will validate the instruction and execute it on behave of the calling operating system.

Critical Review

Comparing to full virtualization technique Xen provides higher performance using para-virtualization. In full virtualization CPU emulation is used to handle and execute operations issued by operating system kernel. However, in Xen, which uses para-virtualization, only privileged instructions are forwarded to Xen to be executed on hardware. Using para-virtualization Xen gained higher performance in the cost of some modifications required in kernel of the operating systems in order to be ported in the virtual environment. Although the required modification is relative small comparing to the kernel size, a sufficient knowledge about both systems, the operating system and Xen itself, is needed by the party who is charge to do the modifications. Also, the operating system providers must agree to perform the required modifications for non-open source operating systems.

One possible solution to this problem is to use hardware assistant virtualization technique. Using this technique the operating systems do not need to be modified to be run in virtual system. The operating systems can execute the highest privileged operations as usual without any restrictions. In this virtualization, the CPU is designed to trap any privileged operation to the virtual machine monitor which will validate and execute them. The use of hardware assistant virtualization avoids doing any modification to the operating systems in order to port them in virtual environment.

Another problem in the proposed approach is the limitation on scalability. The authors aimed with their design to support maximum number of operating systems. Hosting a large number of functional-complete operating systems will have impact on the performance. The proposed approach can run only a small number of operating systems simultaneously without significant effect on the performance. The proposed solution will not work properly in projects where the scalability plays an important role for their success.

Porting only lightweight operating systems in the VMM could improve the scalability and the performance. This solution is a tradeoff between scalability the variety of the supported applications. In other words, this solution is a good choice to improve the scalability if you are only targeting a small range of different applications. If you want to maximize the number of concurrent running operating systems, these operating systems have to be lightweight and do not consume relative large memory and CPU cycles. In the solution the operating systems are not allowed to run in their full capabilities by scarifying some operating systems features that are not required by the supported applications

Applying some of techniques used in Denali could provide high scalability. Denali is a virtual machine monitor and aims to support concurrent running of large number of untrusted internet services on single physical machine. Denali can run 100s to 1000s of virtual machines simultaneously on one physical machine without significant degradation of the performance. One of the used techniques in Denali to improve the scalability is queening and batching of hardware interrupts. The VMM in Denali queues the hardware interrupts for each virtual machine and passes each queue of interrupts to the appropriate virtual machine only when it becomes active. Denali does not context switch the virtual machines to deliver the interrupts. Therefore, the overhead associated with the context switch will be significantly decreased. Decreasing the overhead of context switch leads to increase the scalability.

Performance could be seen as another issue in the proposed approach. In Xen the hardware resources are completely virtualized on software level. Managing the physical resources including memory and of I/O devices between virtual machines is completely implemented in Xen software. As consequence, extra avoidable overhead is added to the system.

Hardware assistant virtualization could be exploited to increase the performance. In the time when the approach was proposed no virtualization on hardware level was supported yet. However, end of 2005 CPUs companies started to implement many features of virtualizations in the hardware. Making use of the implemented virtualization features in hardware will increase the performance.

Expensive sharing data between domains could be considered as an issue in Xen. Isolation between domains is one of the key concepts of the virtualization to provide security. The most virtualization approaches aim to provide high isolation and prevent sharing data between domains on the same physical machine. However, there are some situations in which domains running on the same machines need to exchange data. Because of the isolation enforced by the VMM, domains can share data only over the network. This is very expensive considering that these domains run on the same physical machine.

Despite the isolation sharing data between domains can be done under the control of Xen. The same concept of I/O device data transfer can be used to share data between domains. The domain that wants to send data to another domain can save these data in its own memory space and stores the memory reference of the data in the ring buffer. Then, it informs Xen by invoking an equivalent hypercall. Xen moves the data from the sender memory space into the receiver memory space. Finally, Xen delivers notification to the receiver domain that will read the data from its own memory space. This solution provides the ability to exchange data between domains using the same concept that is used to send and receive data to and from I/O devices.