CMPSCI 677 Distributed and Operating Systems **Spring 2018**

Lecture 5: February 25

Lecturer: Prashant Shenoy *Scribe:* ***Ao Liu***

5.1 Case studies

5.1.1 Volunteer Computing

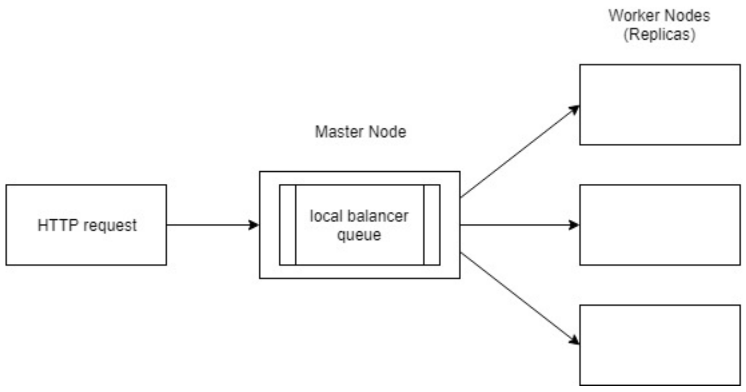
Volunteer computing is based on the idea that volunteers donate CPU cycles/storage when it's not in use. A coordinator partitions a large job into small chunks and sends them to the volunteer nodes. Seti@home, BOINC and P2P backups are examples for this paradigm.

5.1.2 Condor

Condor makes use of idle cycles on workstations in a LAN. It can be used to run large batch jobs and long simulations. It has a central job management system, which idle machines contact. They are then assigned jobs. It supports process migration and flexible job scheduling policies.

5.1.3 Replicated Web Server

Here, one of the nodes acts as a load balancing switch and others as replicas. Requests arrive at a queue which acts a load balancer, which employs policies like least loaded or round robin.



5.2 Virtualization

Virtualization is the process of providing an interface to mimic the behavior of another system or components. It can be broadly classified into the following categories:

5.2.1 Type of Interfaces

- Assembly instructions
- System calls
- APIs

Depending on what is replaced or mimiced, we obtain different forms of virtualization.

5.2.2 Type of Virtualization

5.2.2.1 Emulation

In emulation, a software simulation of a particular type of hardware is done using another. It can use binary translation to convert instructions on the fly and simulate registers in software. It suffers from the extra overhead of a full implementation.

Examples: Bochs, VirtualPC for Mac, QEMU.

5.2.2.2 Full/Native Virtualization

In native virtualization, one or more OSs and the applications they contain are run on top of virtual hardware. Here, the underlying hardware and virtual hardware are of the same type, but the VM simulates enough hardware to allow the OSs to be run in isolation.

Examples: IBM VM family, VMWare Workstation, Parallels, VirtualBox.

5.2.2.3 Para-virtualization

Here, the VM does not simulate hardware. The kernel of the guest OS is modified so that it uses special APIs to call the hypervisor instead of directly accessing the hardware.

Examples: Xen, VMWare ESX Server.

5.2.2.4 OS-level Virtualization

Here the OS allows multiple secure virtual environments to be run. Each application sees an isolated OS. It also serves the use case of backward compatibility.

Examples: Solaris Containers, BSD Jails, Linux Vserver, Linux containers, Docker.

5.2.2.5 Application level virtualization

Here, the application is given its own copy of components that are not shared like global objects, registry files etc.

Example: JVM, Rosetta on Mac (also emulation), WINE.

5.2.3 Type of Hypervisors

Hypervisor/VMM is the virtualization layer, which makes resource management, isolation and scheduling. There are 2 types of hypervisors that act like real hardware. Type 1 hypervisor runs on ”bare metal”, whereas type 2 hypervisor runs on a host OS and the guest OS runs inside the hypervisor.

5.2.4 How Virtualization Works?

CPU supports kernel (ring 0) and user (ring 3) modes. There is a set of instructions (such as I/O, change MMU settings etc.) that can only be executed in kernel mode. Such instructions are called *sensitive instructions*. Privileged instructions cause a trap when executed in kernel mode. As a result, type 1 virtualization is feasible if sensitive instruction is subset of privileged instructions. Recent Intel/AMD CPUs have hardware support, named Intel VT and AMD SVM. The idea is to create containers where a VM and guest can run and that hypervisor uses hardware bitmap to specify which instruction should trap, so that sensitive instruction in guest traps to hypervisor.

5.2.5 Type 1 Hypervisor

In this type, unmodified OS is running in user mode (or ring 1), but it thinks it's running in kernel mode (*virtual kernel mode*). The guest OS traps on privileged instructions and uses VT to trap sensitive instructions. Under this mode, hypervisor is the ”real kernel”. Upon trap, it executes privileged operations or emulates what the hardware would do.

5.2.6 Type 2 Hypervisor

Type 2 works without VT support and sensitive instructions are replaced by procedures that emulate them. For example, VMWare, upon loading program, scans code for basic blocks and replaces sensitive instructions by VMWare procedure using binary translation.

5.2.7 Para-virtualization

Both type 1 and 2 hypervisors work on unmodified OS. In contrast, para-virtualization modifies OS kernel to replace all sensitive instructions with hypercalls. Thus, OS behaves like a user program making system calls and hypervisor executes the privileged operation invoked by hypercall.