L8 Virtualization and Cloud Platforms

Concurrency

May be parallel, or not

Progress of multiple elements of the set overlap in time

A single thread of execution can time slice a set of tasks to make partial progress over time

- Time 0: Work on first 25% of Task 0
- Time 1: Work on first 25% of Task 1
- Time 2: Work on first 25% of Task 2
- Time 3: Work on first 25% of Task 3
- Time 4: Work on second 25% of Task 0
- Time 5: Work on second 25% of Task 1

Parallelism

Progress on elements of the set occur at the same time

Multiple execution units enable progress to be made simultaneously

Processor 1

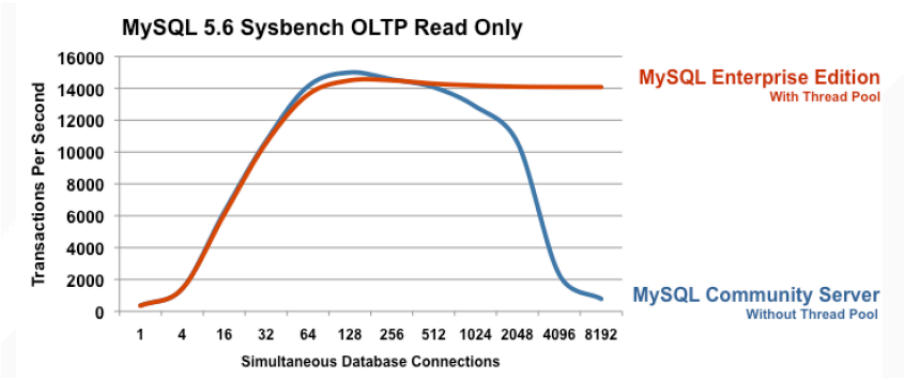
- Time 0: 1st 25% of Task1
- Time 1: 2nd 25% of Task1
- Time 2: 3rd 25% of Task1
- Time 3: 4th 25% of Task1
- Time 4: 1st 25% of Task3

Processor 2

- Time 0: 1st 25% of Task2
- Time 1: 2nd 25% of Task2
- Time 2: 3rd 25% of Task2
- Time 3: 4th 25% of Task2
- Time 4: 1st 25% of Task4

**

Too much parallelism causes thrashing, excessive switching, lower performance



Distribued Systems -- Chap3 Process

Using threads at the client side

1. Multi-threaded web client

   Web browser scans an incoming HTML page, finds out more file needs to be fetched

   Each file is fetched by a individual thread => blocking HTTP request

   High network latency

2. Multiple RPC calls

   A client can do many RPC calls at the same time, each with different threads

   It waits until all results have beem returned

   If calls are to different servers, we can do linear speed-up

Thread-level parallelism: TLP

Let c_i denote the fraction of time that exactly i threads are being executed simultaneously.

$$TLP = \frac{\sum_{i=1}^{N} i \cdot c_i}{1 - c_0}$$

with N the maximum number of threads that (can) execute at the same time.

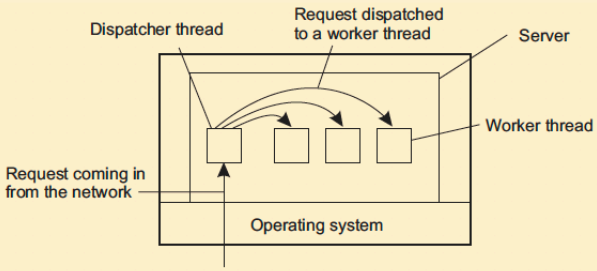
A typical Web browser has a TLP between 1.5 and 2.5

Threads are primarily used for logically organizing browser.

Using threads at the server side

1. Improve performance
   - A new thread cost is much lower than a new process
   - Single thread server means that multi-processer is meaningless
   - hide network latency by reacting to next request while previous one is being replied
2. Better Structure
   - Blocking calls simplifies the overall structure.
   - Smaller and easier to understand due to simplified flow of control

Dispatcher/worker model



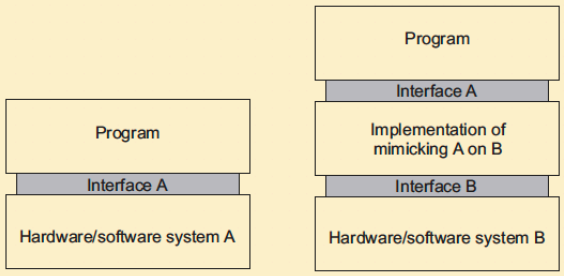
Overview

Model	Characteristics
Multithreading	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls

Virtualization

- Ease of portability and code migration
- Isolation of failing or attacked components

Principle: mimicking interfaces



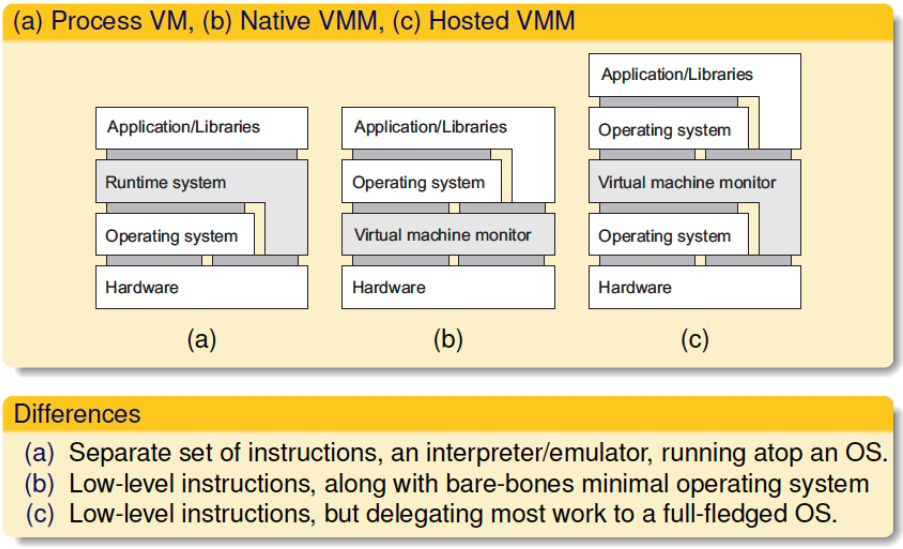
Three levels of interfaces

- Instruction Set Architecture (Different Machine instructions)

  Priviledged instructions and general instructions

- Sysgten calls
- Library calls (API)

Three different Virtualization



Privileged instruction

If and only if executed in user mode, it causes a trap to the operating system

Non-previledged instruction

The rest instruction

Control-sensitive instruction

may affect configuration of a machine (e.g. one affecting relocation register or interrupt table)

Behavior-sensitive instruction

effect is partially determined by context (e.g., POPF sets an interrupt-enabled flag, but only in system mode)

Sensitive instructions

Those that the *hypervisor* or *virtual machine monitor* (VMM) wants to trap and emulate to give an unmodified OS the illusion it owns its hardware resources

Privileged instructions

Refers to the set of instructions that your ISA defines as privileged

Condition for Virtualization

If the set of sensitive instructions for that computer is a subset of the set of privileged instructions ==>VMM

Problem

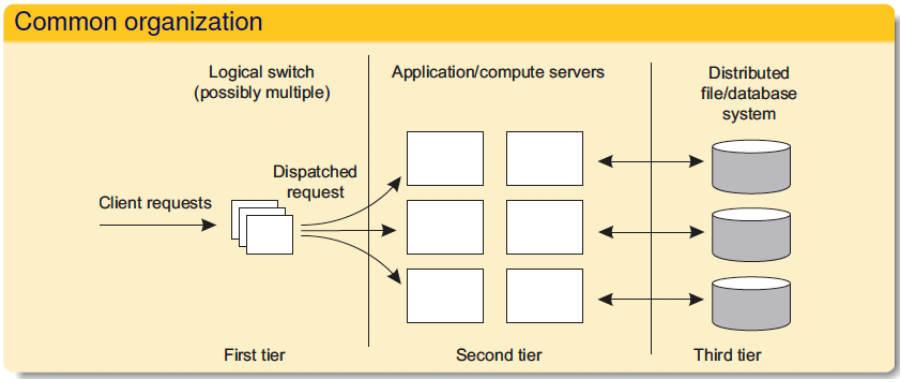
There may be sensitive instructions that are executed in user mode without causing a trap to the operating system

Solution

- Emulate all instructions
- Wrap nonprivileged sensitive instructions to divert control to VMM
- Paravirtualization

  Modify guest OS, either by preventing nonprivileged sensitive instructions, or making them nonsensitive (i.e., changing the context)

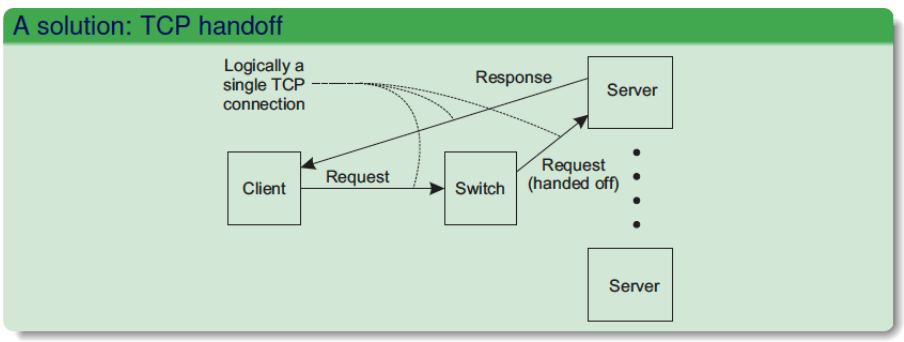
Three different tiers



First Tier

- Passing requests to an appropriate server
- request dispatching

But let the first tier to handle all communication from/to the cluster ==>bottleneck



Migrating a VM

1. Pushing memory pages to the new machine and resending the ones that are later modified during the migration process.
2. Stopping the current virtual machine; migrate memory, and start the new virtual machine
3. Letting the new virtual machine pull in new pages as needed: processes start on the new virtual machine immediately and copy memory pages on demand.

Problem

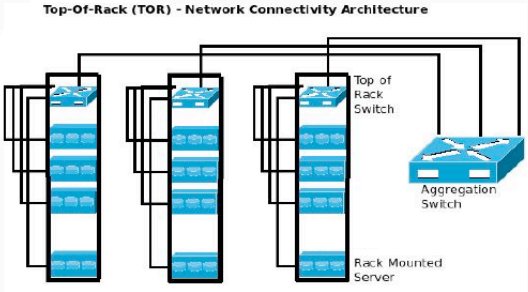
Migrating a VM may takes tens of seconds

DataCenter

Each rack has a "top of rack" network switch that connects it to the rest of the datacenter network

Connecting racks together

"Aggregation" and "Core" network switches provide connectivity between racks



DataCenter Performance

- Ideal: Homogeneous performance
  - Uniform bandwidth/latency between all servers
- Reality (typical): Heterogeneous performance
  - Two servers in the same rack
    - Very high bandwidth/very low latency
  - Two servers in same row (not same rack)
    - Medium bandwidth / medium latency
  - Two servers in different rows
    - Low bandwidth / high lantecy

Modularity

- Containers filled with a 2 or 4 rows of servers

- Many containers

VMM Switches