Note Takers: Chau, Bao Kham (cs162-bb) and Quang Tran (cs162-bc)

Topic: Real time system

A system in which clock time matters - Real time system. Normally, in a computer system, time is virtual.

For example: computer in auto assembly line is real time. The assembly line needs to move the car before the robot wields the car frame.

- anti-ballistic missle defense also an example

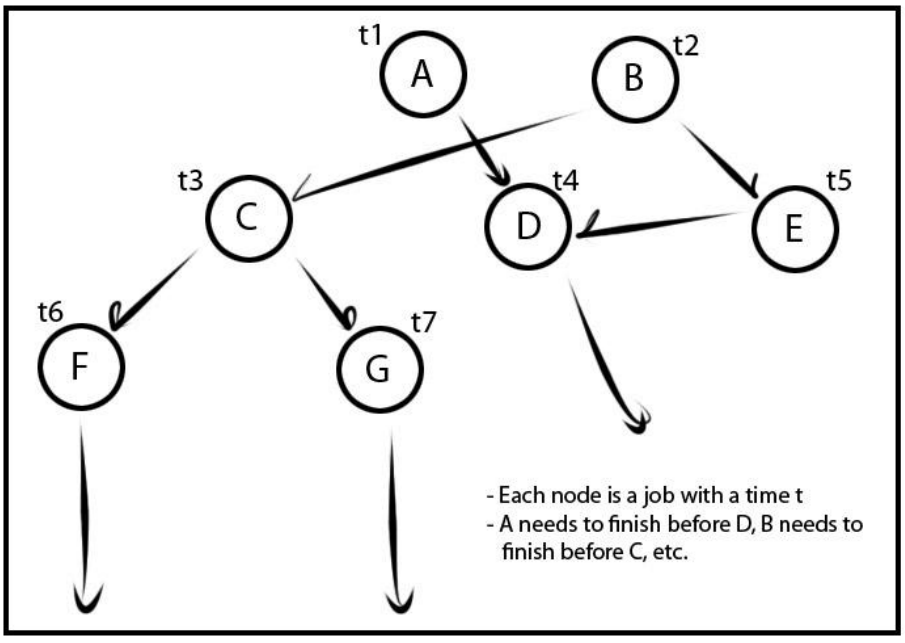
Requirements: meet your deadline. Meet them 100% of the time (in theory). In practice, you can sometimes settle for 99%

-> implies 2 things: 1. You must meet deadline.

2. You must handle the aggregate load. If there are 1000 events a minute, even if none of them have a specific deadline, you still have to be able to handle 1000 events a minute.

Real Time System: mostly workload fixed (incoming jobs are not random or unknown). You know how long something takes to run. Very often sequence of even is known.

1) deadline scheduler. Jobs arrive in random time, unknown processing time (may know distribution) before. This time we know how long jobs going to take, now we have to work out a schedule for the jobs. Usually deals with physical events.



Things happen in milli seconds and shorter

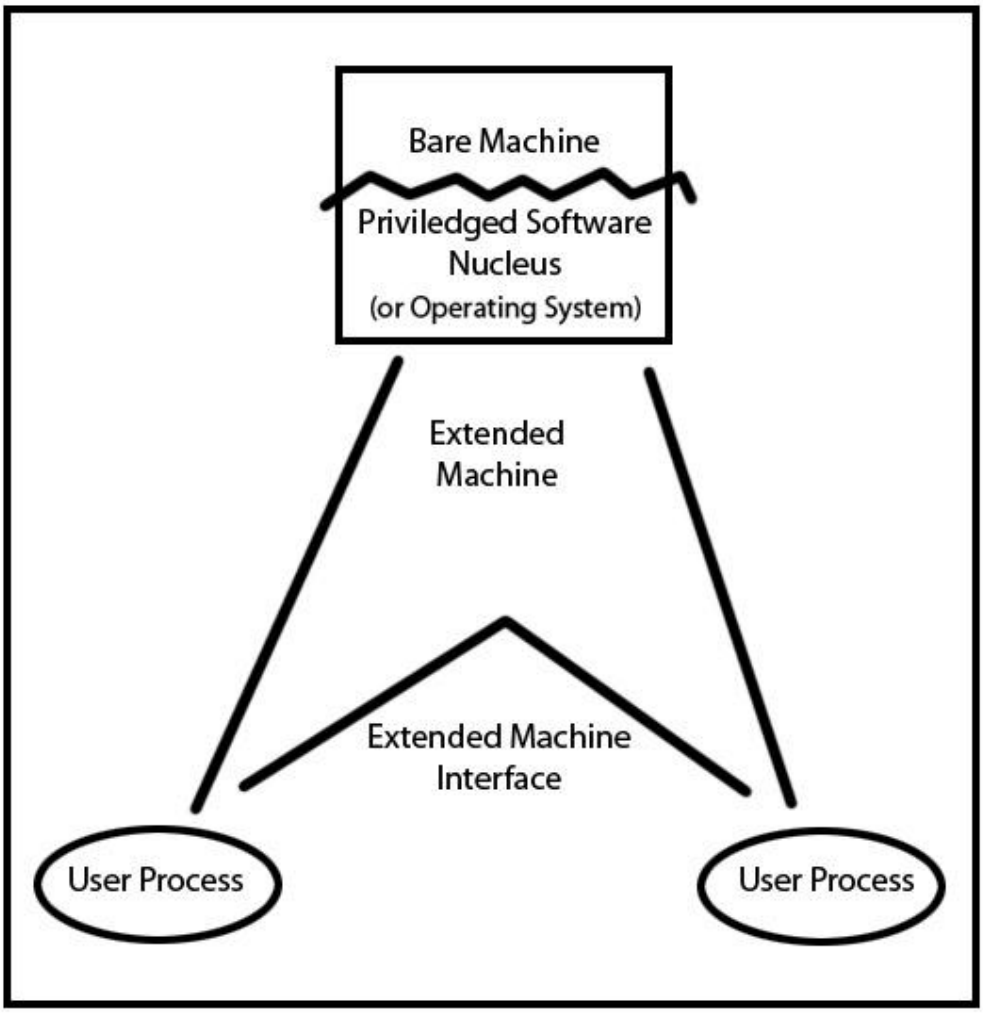
Don't want page faults or unexpected I/O because these take millisecond that might throw the scheduler off schedule. So if there are I/O, put it in buffer

- People says they don't like to use cache memory or virtual memory (nothing get paged, all codes on hand) because it is unpredictable. But cache miss only takes nanoseconds
- These days it's impossible to buy stuffs without cache memory

Topic: Virtual Machine
read Robert Goldberg :Survey of Virtual Machine Research"

Until about 5 years ago, no one seemed to be interested in Virtual machines. Now companies are doing stuff like VMware and Virtual Machines have become a hot topic.

Virtual machine is a software supported copy of the basic (hardware) machine.



Provides interface that allows you to run programs from different OS.

- Virtual Machine monitor and Virtual machine replicates the same underlying machine hardware.

- Emulators are used to provide dissimilar bare machine interface. Even though the architecture is different, program should run
- Both cases, they provide copies of machine.

Why would you want Virtual Machine?

- You can run several OS at the same time.
- Debug the OS so that if bad things happen it only destroys the virtual machine.
- Use a single machine to develop network software. Run multiple OS releases to see if it's stable enough to use.

Advantage: high reliability and high security because you can give one user one machine. It looks like a copy of the machine, anything it does to the other machine will look like communication. In OS, Virtual memory and stuffs are shared, in Virtual Machine there is a thicker wall to these sharing.

Some history:

- Historically, the first ones ran on IBM mainframes. V/370, M44/44X
- Hitachi had a mainframe that ran on an 8400 computer

Reasons why VM is becoming hot: people are building massive servers. One easy way is to run virtual machine on this stuff. It's easy to manage with high security. And you can port one process on one machine to another.

- instead of having an OS, have virtual machines handle everything.
- Interprocess communication, easy to trade processes.

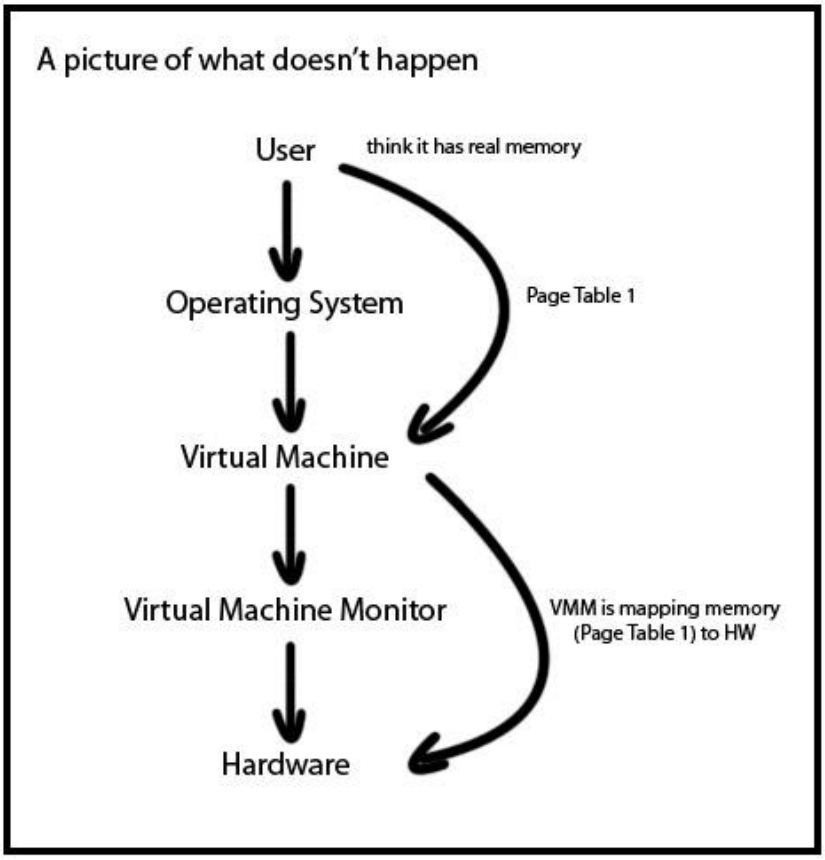
Implementation:

- Virtual Machine is only plausible when it is running on the real hardware almost all of the time. You can't simulate virtual machine.
- You're running virtual machine almost all the time your code is running on real hardware.
- Exception: You have to trap and simulate sensitive information. Anything that makes the virtual OS go into supervisor state, that will make the virtual OS trap because the virtual OS is a user process.
- The trap is to the Virtual machine monitor and the monitor will figure out what's going on and work it out. This isn't implementable this on any machine that doesn't trap sensitive information.
    * X86 doesn't quite virtualize. Virtual Machine monitor has a bit to keep track whether the virtual OS is in "user" or "supervisor" state

* When teacher says OS, he means the thing running on the virtual Machine.

How do you deal with memory mapping and page fault?

Problem: we have two page tables and hardware doesn't know how to handle 2 page tables.



Solution: To map memory of VM to real machine, we use a composition of Page table 1 (which translate VM virtual memory to VM hardware) and Page table 2 (which translate Virtual memory to real hardware)

* In most machines - TLB is entirely a hardware construct. Bottom line has to be: the user is running at the full speed
* On Page fault, we must figure out who handles it, the Virtual machine monitor or the actual hardware
* Page fault is a trap, always traps to the Hardware Monitor. Aha, we have a trap in PageTable 1.5. From there, we see where the miss occurred. PT1 or PT2.
* it looks to the hardware as if it were just one page table
* PT1.5 monitored by the Virtual Machine Monitor
* OS knows nothing about either PT1 or PT2. you can compose as many levels as you want

Topic: I/O

Can also have a virtual I/O system, which is usually a subset of the real I/O.

Each VMM monitor maintain a bit for each VM which specifies whether the VM is in user or supervisor state (mentioned above)

There's a discussion of hardware virtualizer in goldberg article - prof says not to worry about it

Prof says don't worry about Hardware virtualizer

- problem with IBM's mainframe system is that it's not user friendly.

- clunk and difficult to use. Plus parts of it are pretty slow.
  at some point a substantial amount of IBM mainframes are running VM
- we're down to 1-maybe 2 mainframes. Now everyone except IBM is pretty much
  gone.

Topic: Virtual Machine Performance

Most of the time VM runs on real hardware (so most of the time runs at real speed). Sometimes slow down:

* support privileged instructions
* maintain status of virtual processor
* support paging
* console functions
* acceptance and reflection of interrupts to individual VMs
* translation of channel programs
* maintenance of clocks (virtualize the clock - 2 clocks virtual clock and real clock)

Enhance performances:

* OV = R, page table becomes an identify mapping

- dedicate some resources
  * another implementation is to sometimes take the OP its memory is infinitely large
  * VMM are responsible for
  * Each OS responsible VM for that VM
  * EX. OS think they have vastly more physical memory than that the OS provides.
- give certain critical VMs priority to run
- run virtual = real
- let VM do paging instead of OS
  * IBM sold 360 for lots of money.
  * lots of companies were clones.
  * IBM started moving things to the microprocessor.
  * making it hard to replicate.

- modify OS to avoid slow features
- extend VMM to provide performance enhancements

Virtualizing the x86 (which doesn't really virtualize right)

1) a process running on VM can determine that it is not running in privileged state because there are two bits that indicates it.

2) some privileged instructions don't trap if run in user state.

Solution: have software to scan these situations and generate appropriate codes.

Remote Procedure Call:

want to make a call to another machine (remote procedure call)

read paper: "implementing remote procedure calls"

Issues:

- How are arguments and results passed when there is no shared address space? Could fake call by reference. But usually pass by values
- How is the callee located/identified?
- you would like the remote procedure call to not be different from a local procedure call.
- there's overhead
- Data integrity and security concerns when using open communication channels

Mechanism

5 components:

* Caller (user)
* User stub - interface b/w caller and communication

mechanism

* Communications Package (RPC RUntime)
* Server stub - interface b/w communications package and

callee

* Server (callee)