Today

- VM issues
- OS Retrospective
- Future of OS

Review: Memory Management

- What are some algorithms for memory page replacement?
- What is prefetching?
  - Tradeoffs?

ALLOCATING FRAMES

Allocation of Frames

How many frames should each process get?

Allocation of Frames

How many frames should each process get?

- Each process needs a minimum number of frames
- Maximum is total frames in the system
- Allocation schemes vary

Fixed Allocation

- Equal allocation
  - Example: if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
  - Alternatively: Keep some as free frame buffer pool
- Proportional allocation
  - Allocate according to the size of process
  - Dynamic as degree of multiprogramming, process sizes change

Priority Allocation

- Use a proportional allocation scheme using *priorities* rather than size
- If process P_i generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number

Global vs. Local Allocation

- Global replacement ← More Common
  - Process selects a replacement frame from the set of all frames
    - one process can take a frame from another
  - But then process execution time can vary greatly
  - But greater throughput so more common
- Local replacement
  - Each process selects from only its own set of allocated frames
  - More consistent per-process performance
  - But possibly underutilized memory

Non-Uniform Memory Access

- Assumption: all memory accessed equally
- Many systems are NUMA
  - Speed of access to memory varies
  - Consider system boards containing CPUs and memory, interconnected over a system bus
- Optimal performance comes from allocating memory close to the CPU on which the thread is scheduled
  - Modify scheduler to schedule the thread on the same system board when possible

Thrashing

- If a process does not have "enough" pages, page-fault rate is very high
  - Page fault to get page
  - Replace existing frame
  - But quickly need replaced frame back
  - This leads to:
    - Low CPU utilization
    - Operating system thinking that it needs to increase the degree of multiprogramming
    - Another process added to the system
- ***Thrashing*** ≡ a process is busy swapping pages in and out

Demand Paging and Thrashing

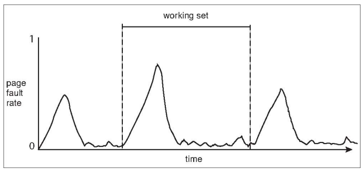
- Why does demand paging work?
  Locality model
  - Set of pages that are actively used together
  - Process migrates from one locality to another
  - Localities may overlap
- Why does thrashing occur?
  Σ size of locality > total memory size
  - Limit effects by using local or priority page replacement

How can we prevent thrashing?

- Working set strategy
  - Conceptual model of working set
  - Collection of pages a process is actively using
    - Must be memory-resident to prevent this process from thrashing
- Divide processes into two groups: active and inactive
  - When a process is active, its entire working set must always be in memory
    - never execute a thread whose working set is not resident.
  - When a process becomes inactive, its working set can migrate to disk.

Working Sets and Page Fault Rates

- Direct relationship between working set of a process and its page-fault rate
- Working set changes over time
- Peaks and valleys over time



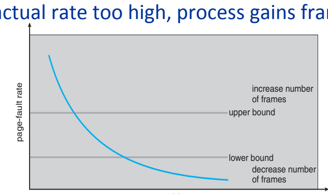
Keeping Track of the Working Set

- Approximate working set during some period Δ with interval timer + a reference bit
- Example: Δ = 10,000
  - Timer interrupts after every 5,000 references
  - Keep in memory 2 bits for each page
  - Whenever a timer interrupts, copies reference bits and sets the values of all reference bits to 0
  - If one of the bits in memory = 1 ⇒ page in working set

Page	0-5000	5000-1000
1	0	0
2	1	0
3	1	1

Page-Fault Frequency

- More direct approach than WSS
- Establish "acceptable" page-fault frequency (PFF) rate and use local replacement policy
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame



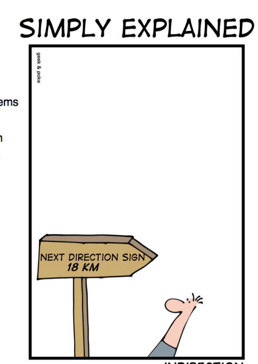
Thrashing in Practice

- Today's OSs don't worry much about thrashing
- With personal computers, users can notice thrashing and handle it themselves:
  - Typically, just buy more memory
  - Or, manage active applications
- Thrashing was a bigger issue for timesharing machines with dozens or hundreds of users:
  - Why should I stop my processes just so you can make progress?
  - System had to handle thrashing automatically.
- Technology changes make it unreasonable to operate machines in a range where memory is even slightly overcommitted
  - better to just buy more memory

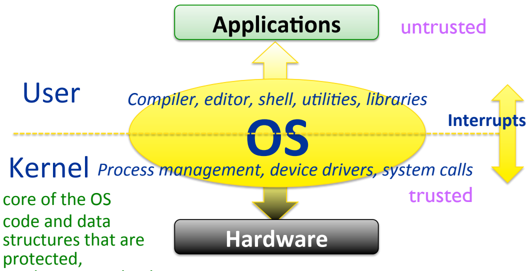
OS RETROSPECTIVE

Indirection

A famous aphorism of David Wheeler goes: "All problems in computer science can be solved by another level of indirection";[1] this is often deliberately mis-quoted with "abstraction layer" substituted for "level of indirection". Kevlin Henney's corollary to this is, "...except for the problem of too many layers of indirection."



Operating System



- core of the OS
- code and data structures that are protected, can be accessed only in the kernel mode

Mode stored in a register

Why Study OS?

- Understanding the OS helps you write better code
- Understand a wide range of system designs and tradeoffs of those designs

Course Objectives

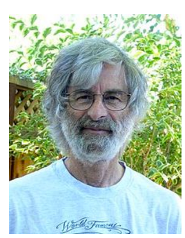
- to demystify the interactions between the software you have written in other courses and hardware,
- to familiarize you with the issues involved in the design and implementation of modern operating systems,
- and to explain the more general systems principles that are used in the design of **all** computer systems

Learning Objectives

- Explain how operating systems manage concurrent processes including the complete life-cycle of user processes, threads, process synchronization, and deadlock avoidance.
- Evaluate algorithms used for process scheduling, memory allocation, and disk access.
- Understand how operating systems manage physical and virtual memory including segmentation and paging.
- Develop programs that emulate or interact with operating system code.

WHERE DO WE GO FROM HERE?

What is a distributed system?



Leslie Lamport
2013 Turing Award Winner

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." -- Leslie Lamport

Motivation

- Key goal for distributed systems is resource sharing

Coordination

- How to assign data and functions among nodes?
  - E.g., to spread the work across an elastic cluster?
- How to know which node is “in charge” of a given function or data object?
- Goals: safe, robust, even, balanced, dynamic, etc.
- Coordination is the key to building scalable distributed systems.

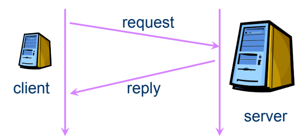
Distributed File Systems: Goal & Challenges

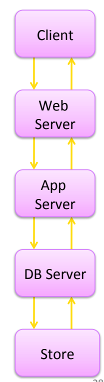
Goal: Transparent access to remote files

- Enable programs to store and access remote files as though they were local
- Access at any time from any computer
- Comparable performance and reliability to local disks

Why would you want this?
What are some of the hard issues?
Know any examples?

A service

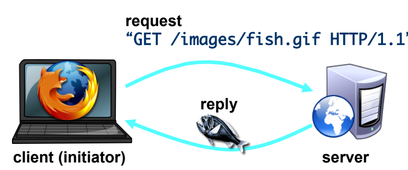




Server performance

- How many clients can the server handle?
- What happens to performance as we increase the number of clients?
- What do we do when there are too many clients?

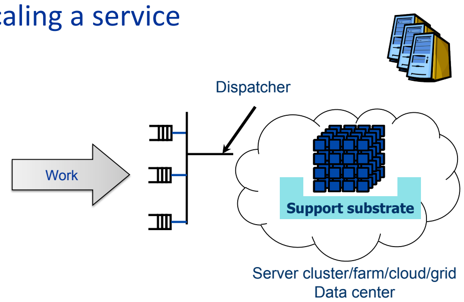
A simple, familiar example




```
sd = socket(…);
connect(sd, name);
write(sd, request…);
read(sd, reply…);
close(sd);
```

```
s = socket(…);
bind(s, name);
sd = accept(s);
read(sd, request…);
write(sd, reply…);
close(sd);
```

Scaling a service

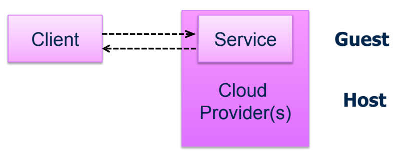


Add servers for scale and robustness.
Issues: state storage, server selection, request routing, etc.

Cloud > server-based computing

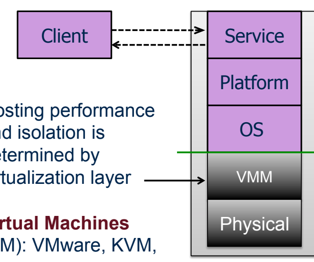
- Client/server model (1980s -)
- Now called Software-as-a-Service (SaaS)

Host/guest model



- Service is hosted by a third party
  - flexible programming model
  - cloud APIs for service to allocate/link resources
  - on-demand: pay as you grow

IaaS: Infrastructure as a Service



EC2 is a public IaaS cloud (fee-for-service).

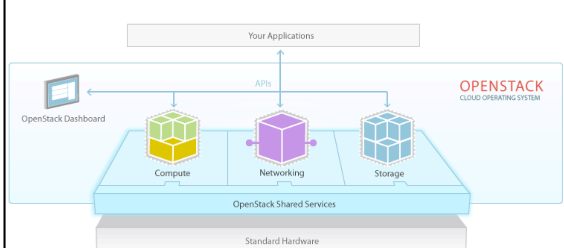
Deployment of private clouds is growing rapidly w/ open IaaS cloud software.

Hosting performance and isolation is determined by virtualization layer

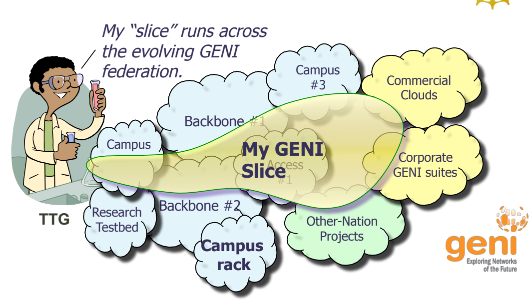
Virtual Machines (VM): VMware, KVM, etc.

XCAT openstack

OpenStack, a Cloud Operating System



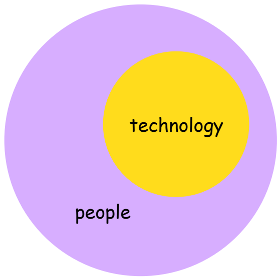
GENI: slices and federation



Slice: an end-to-end virtual network context spanning multiple sites, with configurable topology and properties, e.g., containment and isolation.

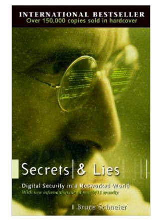
Security

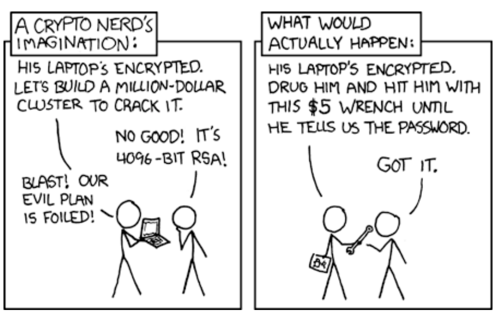
- Where are the boundaries of the “system” that you would like to secure?
- Where is the weakest link?
- What happens when the weakest link fails?



The First Axiom of Security

- “Security is at least as much a social problem as it is a technical problem.”
  - Translation: humans are the weak link.
- Never lose sight of the social dimension.
  - Keys left in lock
  - Phishing
  - Executable attachments
  - Trojan software
  - Post-it passwords
  - Bribes, torture, etc.
  - Etc.





Looking Ahead

- Final Exam
  - Take Home Question – typed, PDF
    - 20% of final exam
  - In-class portion
- Evaluations – due Sunday
  - Add EC points to project grade, worth 45% of course grade
- Project due today
- Office hours
  - Monday and Tuesday afternoon