Goals for Today

- Finish Page Replacement
- Working Set/Thrashing
- Introduction to networking

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne. Many slides generated from my lecture notes by Kubiatowicz, Vern Paxson, and Randy Katz.

Allocation of Page Frames (Memory Pages)

- How do we allocate memory among different processes?
  - Does every process get the same fraction of memory? Different fractions?
  - Should we completely swap some processes out of memory?
- Each process needs minimum number of pages
  - Want to make sure that all processes that are loaded into memory can make forward progress
  - Example: IBM 370 – 6 pages to handle SS MOVE instruction:
    » instruction is 6 bytes, might span 2 pages
    » 2 pages to handle from
    » 2 pages to handle to
- Possible Replacement Scopes:
  - Global replacement – process selects replacement frame from set of all frames; one process can take a frame from another
  - Local replacement – each process selects from only its own set of allocated frames

Fixed/Priority Allocation

- Equal allocation (Fixed Scheme):
  - Every process gets same amount of memory
  - Example: 100 frames, 5 processes⇒process gets 20 frames
- Proportional allocation (Fixed Scheme)
  - Allocate according to the size of process
  - Computation proceeds as follows:
    \[ a_i = \frac{s_i}{S} \times m \]
    where:
    - \( s_i \) = size of process \( p_i \) and \( S = \sum s_i \)
    - \( m \) = total number of frames
  - Priority Allocation:
    - Possible behavior: If process \( p_i \) generates a page fault, select for replacement a frame from a process with lower priority number
  - Other schemes?
    - Change adaptively during process lifetime
Page-Fault Frequency Allocation

- Can we reduce Capacity misses by dynamically changing the number of pages/application?
- Establish "acceptable" page-fault rate
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame
- Question: What if we just don't have enough memory?

Thrashing

- If a process does not have "enough" pages, the page-fault rate is very high. This leads to:
  - low CPU utilization
  - operating system spends most of its time swapping to disk
- Thrashing = a process is busy swapping pages in and out
- Questions:
  - How do we detect Thrashing?
  - What is best response to Thrashing?

Locality In A Memory-Reference Pattern

- Program Memory Access Patterns have temporal and spatial locality
  - Group of Pages accessed along a given time slice called the "Working Set"
  - Working Set defines minimum number of pages needed for process to behave "well"
- Not enough memory for Working Set → Thrashing
  - Better to swap out process?

Working-Set Model

- \( \Delta = \text{working-set window} = \text{fixed number of page references} \)
  - Example: 10,000 instructions
- \( \text{WS}_i \) (working set of Process \( P_i \)) = total set of pages referenced in the most recent \( \Delta \) (varies in time)
  - If \( \Delta \) too small will not encompass entire locality
  - If \( \Delta \) too large will encompass several localities
  - If \( \Delta = \infty \) ⇒ will encompass entire program
- \( D = \sum \text{WS}_i \) = total demand frames
  - if \( D > m \) ⇒ Thrashing
  - Policy: if \( D > m \), then suspend/swap out processes
  - This can improve overall system behavior by a lot!
What about Compulsory Misses?

• Recall that compulsory misses are misses that occur the first time that a page is seen
  – Pages that are touched for the first time
  – Pages that are touched after process is swapped out/swapped back in
• Clustering:
  – On a page-fault, bring in multiple pages “around” the faulting page
  – Since efficiency of disk reads increases with sequential reads, makes sense to read several sequential pages
• Working Set Tracking:
  – Track working set of application
  – When swapping process back in, swap in working set

Demand Paging Summary

• Replacement policies
  – FIFO: Place pages on queue, replace page at end
  – MIN: Replace page that will be used farthest in future
  – LRU: Replace page used farthest in past
• Clock Algorithm: Approximation to LRU
  – Arrange all pages in circular list
  – Sweep through them, marking as not “in use”
  – If page not “in use” for one pass, than can replace
• Nth-chance clock algorithm: Another approx LRU
  – Give pages multiple passes of clock hand before replacing
• Second-Chance List algorithm: Yet another approx LRU
  – Divide pages into two groups, one of which is truly LRU and managed on page faults.
• Working Set:
  – Set of pages touched by a process recently
• Thrashing: a process is busy swapping pages in and out
  – Process will thrash if working set doesn’t fit in memory
  – Need to swap out a process

Administrivia

• Project 2 will be out today
• Midterm next week:
  – Wednesday, March 9th
  – Closed book, one page of hand-written notes (both sides)
• Midterm Topics: Everything up to this Wednesday, March 2nd

5min Break
What do this two have in Common?

- First printing press
- Key idea: splitting up text in individual components
  - E.g., lower, upper case letters
- Both lower the cost of distributing information

The Internet

Johann Gutenberg (1398-1468)

The ARPANet

- Paul Baran
  - RAND Corp, early 1960s
  - Communications networks that would survive a major enemy attack
- ARPANet: Research vehicle for “Resource Sharing Computer Networks”
  - 2 September 1969: UCLA first node on the ARPANet
  - December 1969: 4 nodes connected by phone lines

ARPANet Evolves into Internet

<table>
<thead>
<tr>
<th>ARPANet</th>
<th>TCP/IP</th>
<th>NSFNet</th>
<th>Deregulation &amp; Commercialization</th>
<th>WW</th>
<th>SaS</th>
<th>AIP</th>
</tr>
</thead>
</table>

SATNet: Satellite network
PRNet: Radio Network

Web Hosting
Multiple ISPs
Internet2 Backbone
Internet Exchanges

Application Hosting
ASP: Application Service Provider
SaS: Software as a Service
Provider (e-commerce toolkit, etc.)

ARPANET GEOGRAPHIC MAP, OCTOBER 1980
ISP: Internet Service Provider
NAP: Network Access Point

Networking: How Hard Can It Be?

- You just string a wire (or other signaling path) between two computers …
- … first one pushes bits down the link …
- … and the second one gets them up. Right?
- Where does it get tricky? What are the challenges?
Why Networking Is Challenging

• Fundamental challenge: the speed of light

• Question: how long does it take light to travel from Berkeley to New York?

• Answer:
  – Distance Berkeley → New York: 4,125 km
  – Traveling 300,000 km/s: 13.75 msec

Fundamental Challenge: Speed of Light

• Question: how long does it take an Internet "packet" to travel from Berkeley to New York?

• Answer:
  – For sure ≥ 13.75 msec
  – Depends on:
    » The route the packet takes (could be circuitous!)
    » The propagation speed of the links the packet traverses
      • E.g., in optical fiber light propagates at about 2/3 \( C \)
      » The transmission rate (bandwidth) of the links (bits/sec)
        • and thus the size of the packet
        » Number of hops traversed (store-and-forward delay)
        » The “competition” for bandwidth the packet encounters (congestion). It may have to sit & wait in router queues.
  – In practice this boils down to: ≥ 40 msec

Fundamental Challenge: Speed of Light

• Question: how many cycles does your PC execute before it can possibly get a reply to a message it sent to a New York web server?

• Answer:
  – Round trip takes ≥ 80 msec
  – PC runs at (say) 3 GHz
  – 3,000,000,000 cycles/sec * 0.08 sec = 240,000,000 cycles
  – Thus,
    » Communication feedback is always dated
    » Communication fundamentally asynchronous

Fundamental Challenge: Speed of Light

• Question: what about between machines directly connected (via a local area network or LAN)?

• Answer:
  % ping www.icir.org
  PING www.icir.org (192.150.187.11): 56 data bytes
  64 bytes from 192.150.187.11: icmp_seq=0 ttl=64 time=0.214 ms
  64 bytes from 192.150.187.11: icmp_seq=1 ttl=64 time=0.226 ms
  64 bytes from 192.150.187.11: icmp_seq=2 ttl=64 time=0.209 ms
  64 bytes from 192.150.187.11: icmp_seq=3 ttl=64 time=0.212 ms
  200 µsec = 600,000 cycles
  – Still a loooong time …
  – … and asynchronous
Why Networking Is Challenging (con’t)

- Fundamental challenge: components fail
  - Network communication involves a chain of interfaces, links, routers and switches …
  - … all of which must function correctly.

- Question: suppose a communication involves 50 components which work correctly (independently) 99% of the time. What’s the likelihood the communication fails at a given point of time?
  - Answer: success requires that they all function, so failure probability = 1 - 0.99^{50} = 39.5%.

- So we have a lot of components, which tend to fail …
  - … and we may not find out for a looong time

---

Examples of Network Components

- Links
  - Fibers
  - Coaxial Cable

- Interfaces
  - Ethernet card
  - Wireless card

- Switches/Repeaters
  - Large router
  - Telephone switch

---

Why Networking Is Challenging (con’t)

- Challenge: enormous dynamic range
  - Round-trip times (latency) vary 10 µsec’s to sec’s (10^6)
  - Data rates (bandwidth) vary from kbps to 10 Gbps (10^7)
  - Queuing delays inside the network vary from 0 to sec’s
  - Packet loss varies from 0 to 90+%!
  - End system (host) capabilities vary from cell phones to supercomputer clusters
  - Application needs vary enormously: size of transfers, bidirectionality, need for reliability, tolerance of jitter

- Related challenge: very often, there is no such thing as “typical”. Beware of your “mental models”!
  - Must think in terms of design ranges, not points
  - Mechanisms need to be adaptive
Why Networking Is Challenging (con’t)

- Challenge: **different parties must work together**
  - Multiple parties with **different agendas** must agree how to divide the task between them

- Working together requires:
  - **Protocols** (defining **who does what**)
    - These generally need to be **standardized**
  - Agreements regarding how different types of activity are treated (**policy**)

- Different parties very well might try to "**game**" the network’s mechanisms to their advantage

Why Networking Is Challenging (con’t)

- Challenge: **incessant rapid growth**
  - Utility of the network scales with its size
  - Fuels **exponential growth** (for more than 2 decades!)

- Adds another dimension of **dynamic range** …
  - … and quite a number of **ad hoc** artifacts

Why Networking Is Challenging (con’t)

- Challenge: **there are Bad Guys out there**

- As the network population grows in size, so does the number of
  - Vandals
  - Crazies

- What really matters, though: as network population grows, it becomes more and more attractive to
  - **Crooks**
    - (and also **spies and militaries**)
Why Crooks Matter for Networking

• They (and other attackers) seek ways to misuse the network towards their gain
  – Carefully crafted “bogus” traffic to manipulate the network’s operation
  – Torrents of traffic to overwhelm a service (denial-of-service) for purposes of extortion / competition
  – Passively recording network traffic in transit (sniffing)
  – Exploit flaws in clients and servers using the network to trick into executing the attacker’s code (compromise)

• They do all this energetically because there is significant $$$ to be made
Summary

- Networking is about design in the presence of challenges/constraints:
  - Not akin to e.g. programming languages / compilers
    » Which have well-developed theories to draw upon
  - Much more akin to operating systems
    » Abstractions
    » Tradeoffs
    » Design principles / "taste"