CSE 536 Adv OS

Unix: Time-sharing system. (Big advancement over batch processing sys.)

- computer center
- punched card

process:
- schedule
  - SJF
  - FCFS
  - Round robin
  - priority
  - RTOS
  - Interaction

Fundamental reason?
- computer very expensive
dev.
- programming model.

- edit code
- run
- debug

- Shell programs?
  - interpreted
  - compiled

Int. process comm. & sync.

memory management
- paging/segmentation
demand paging
virtual memory
- buddy allocation scheme

7 big data numerical
- scientific programming
Next step in evolution of OS

new HW → new apps → new OS → new apps

Why

Claim - Von-Neumann model is still popular?

Stored program model

1. Simple
2. Moore's Law
3. Amdahl's Law
$T_1$ on 1 proc. sys.

$T_n$ is time of a program on $n$ processor system

$T_1 > T_2 > T_3 \ldots > T_n$

Speedup $S_n$ : speed up on $n$ proc.
syst.

\[
S_n = \frac{T_1}{T_n}
\]

\[
T_n = T_1 \frac{f_s}{n} + T_1 \frac{1-f_s}{n}
\]

\[
S_n \to \infty \leq \frac{1}{f_s}
\]
1. Every program has substantial overhead
   parallel program perf. constrained by Amdahl's law

2. Parallelization is expensive/difficult/make programs complex
   debug/develop

   producer -> consumer
   |
   v
   wait unit
   buffer
   buffer not empty
   has to wait until
   buffer is at least
   one empty slot
   on buffer full
   - multiple producers & multiple consumers
     - locks/semaphores

3. By the time parallel program is fully developed the reg. program's performance may have caught up due to Moore's Law!
Moore Law: transistor density increases (doubles) every 18 months.

Electricity → dissipate heat as heat

Heat sink → heat sink

Air cooling.

Pipeline arch.

\[ F | I D | E X | R B \]

I₁, I₂, I₃, I₄

\[ t_1, t_2, t_3, t_4 \]

\[ t = t_1 + t_2 + t_3 + t_4 \]

\[ t_1 = t_2 = t_3 = t_4 = 2 \]

Ex time non-pipelined ≤ 4

Ex time on 4-way pipelined proc.

Ideally, processor is able to execute one instr. per cycle.
\[
N_e \rightarrow N_e \\
\frac{N_e}{4 + (N-1)\frac{E}{4}} \propto \frac{N_e}{4 + (N-1)\frac{E}{4}} \propto N_e
\]

**Pollack's Rule.**

\[\text{Perf} \propto \sqrt{A}\]

\[\text{Power} \propto A\]

- single core
- 2-core

\[
\text{Perf} = \sqrt{2A} \\
\text{Power} = 2A
\]

- 2-core
- \[
\text{Perf} = 2\sqrt{A} \\
\text{Power} = 2A
\]

**Motivator behavior**

- multitasking
- ideal provided better perf-power ratio.

**Virtualization**

- Xen