Deadlock Avoidance

3 Chopsticks, 3 philosophers ≤ max 2 requests

Safe

Unsafe

A (2)  B (2)  C (2)

A (1)  B (2)  C (2)

A (1)  B (2)  C (1)

A ×  B  C

4/22/24

Deadlock Wrapup & Memory Management

request 1 chopstick at a time

Avail = 3

Avail = 2

Avail = 1

Avail = 0

Starts out

to be avail = 2

2 - 1 + 2 = 3

(A request 1 more)

(A will finish & return both)

Can finish

1 - 1 + 2 = 2

2 - 0 + 2 = 2

2 - 1 + 2 = 3

Can finish

no one can finish their future request
Deadlock Detection

→ rare event, let it happen, detect & recover when it happens

→ Resource Allocation Graph

- if there's a cycle:
  - single instance resource ⇒ deadlock
  - multi-instance resource ⇒ potential deadlock

→ recover from deadlock

→ abort / terminate a process in the cycle
Physical Memory Management
- volatile, byte addressable, order of GBs
- ~200 cycles access latency

Problem: many processes, limited physical memory

Attempt #1: let one process use all of physical memory
- no need for translation
- context switch

→ write A’s VAS to disk
→ load B’s VAS into memory

- how do we enforce kernel/user memory separation
→ Attempt #2: Divide up physical memory among processes

→ low context switch overhead
→ virtual to physical address translation
→ base & bound registers (hw support)

\[ PA = VA + \text{base} \quad (VA < \text{bound}) \]

base register \( 0 \) for process A
bound register \( 100 \)

→ how do we support memory growth?
→ variable sized allocation leads to external fragmentation

\[ \text{D doesn't fit, can't allocate} \]
Compaction

→ Can we share library code or support can fork with this approach?
  → translation done on the entire VAS, can't share part of it
Attempt #3: finer grained translation & permission

- divide the VAS into fixed size pages
- sharing page A (library code)

- process 1's VAS
  - A
  - B
  - C
  - D
  - E

- process 2's VAS
  - A
  - E

- physical memory

What should be translated

- virtual address
  - 64 bits
  - \( \frac{52}{12} \) offsets
  - page 4

- offset within a page
  - only need to translate page #