CSE 531: Distributed and Multiprocessor Operating Systems

Class: 15 Date: 10/5

Fall 2017

Note Title

Scheduling

- CPU scheduling —
- Memory → demand paging....
- disk head scheduling + I/O scheduling

CPU (uniprocessor)

- FIFO
- SJF
- Round Robin
  - ↳ multilevel feedback Qs
    - ↳ priority

Linux → nice scheduler

Epoch "

CFS ...

Multicore scheduling

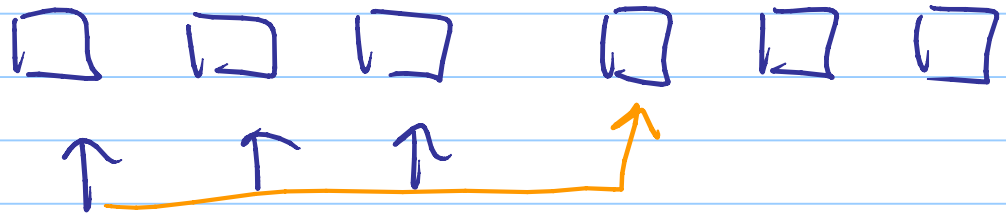
small → N processors

medium → M – tasks or processes or appl

large → T threads (total)

Simple, round robin

→ thread 8 → ('random') order



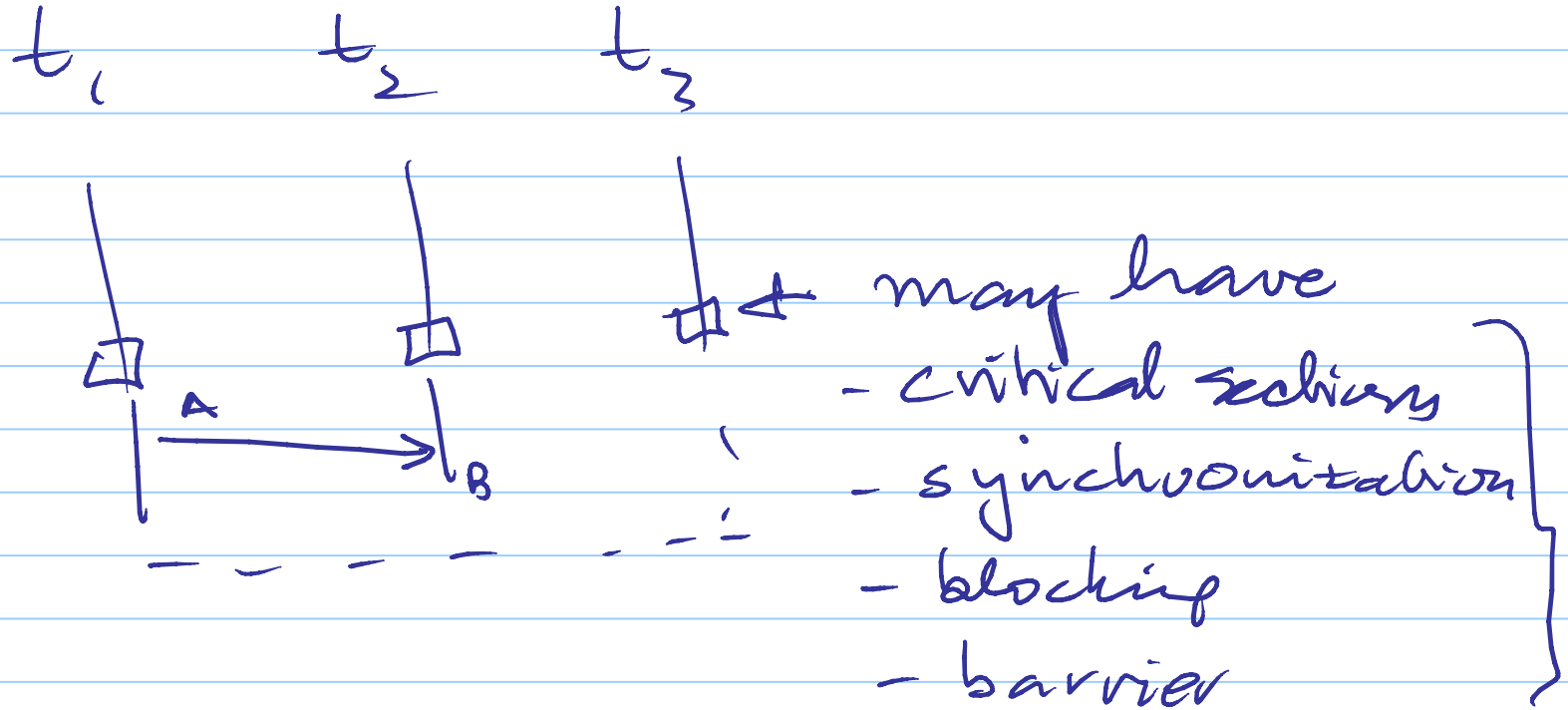
running

self schedule each processor

"Better" → Coscheduling

↳ when running a thread from a $Task_i$, schedule all the other threads of $Task_i$ at the same time

Task

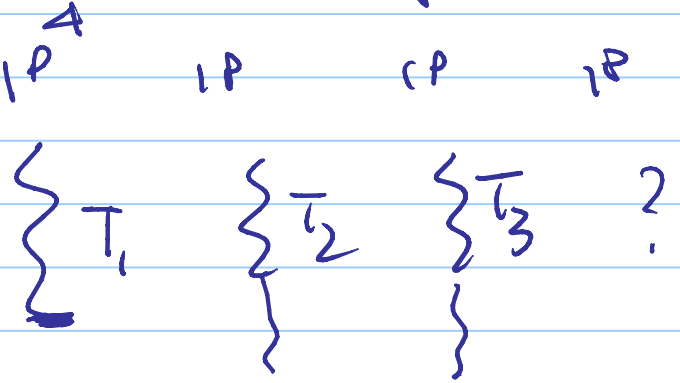

t_1
t_2
t_3
A
B
may have
- critical sections
- synchronization
- blocking
- barrier

Coscheduling

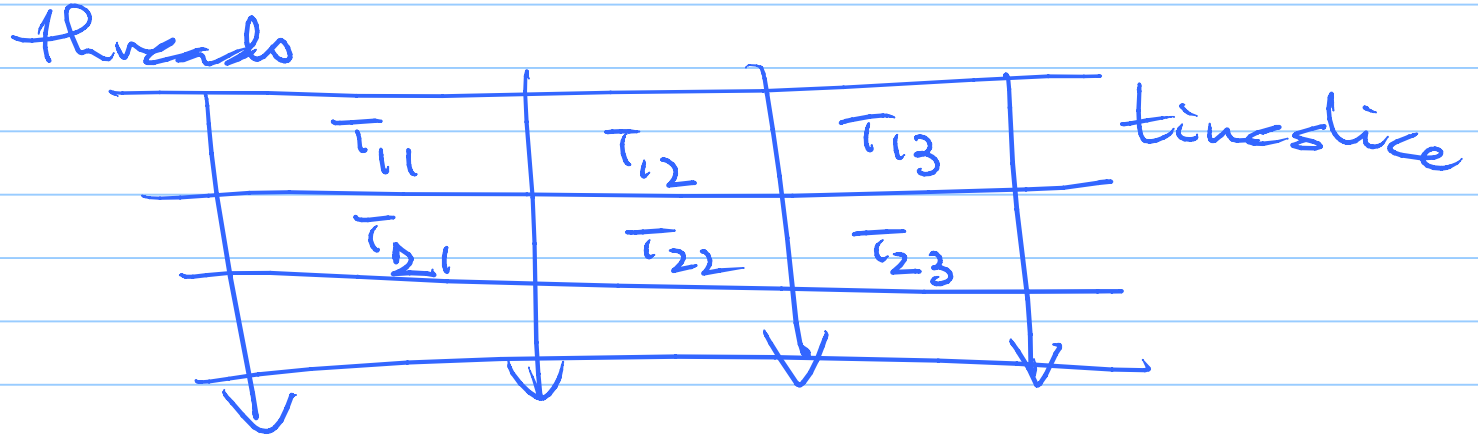
- schedule all together
  - if a thread blocks, keep it loaded on processor → it will get unblocked soon (assumption)
    - ↳ if does not happen then assume the block is for a long time

Strict coscheduling – Gang scheduling

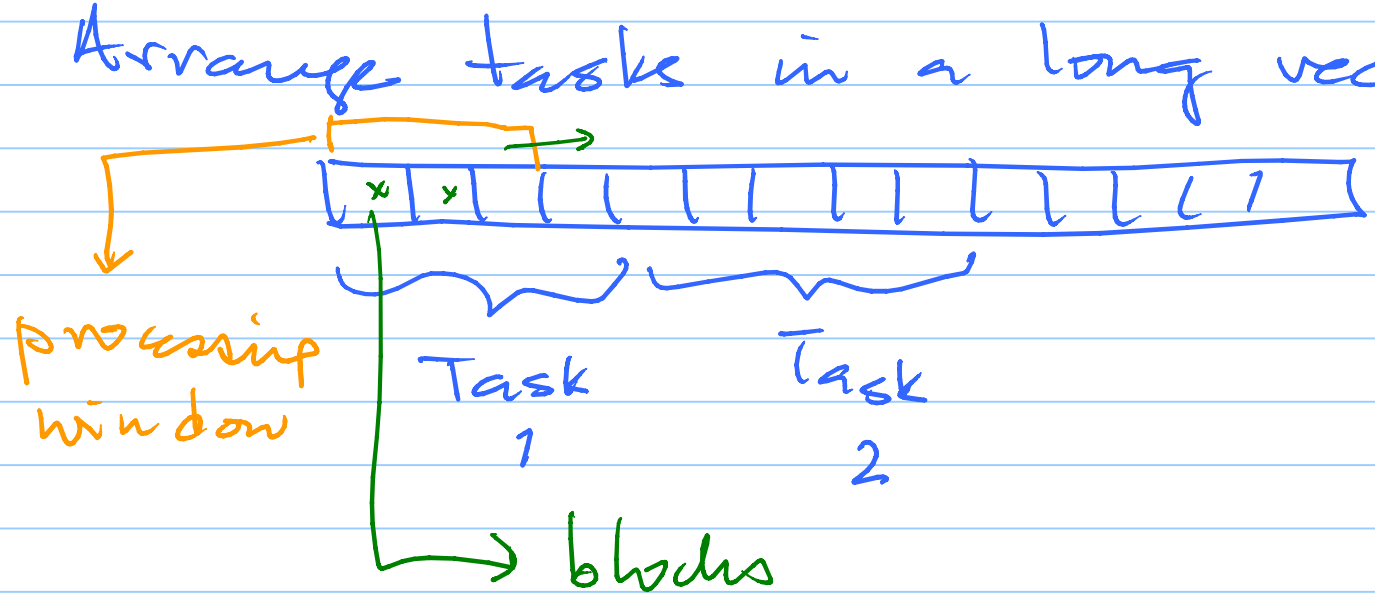
– All threads of a task must be scheduled together



MATRIX coscheduling (same as Gang)

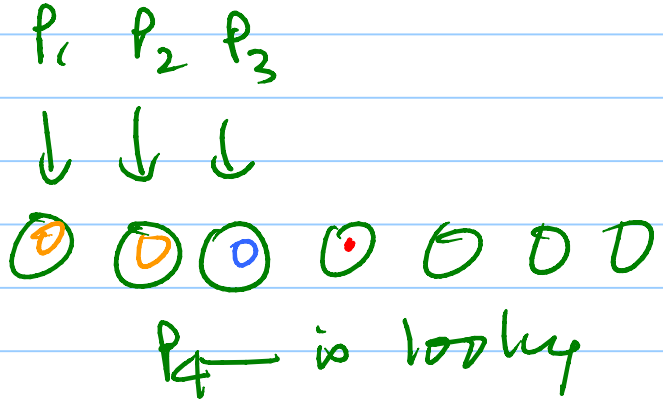


Continuous coscheduling

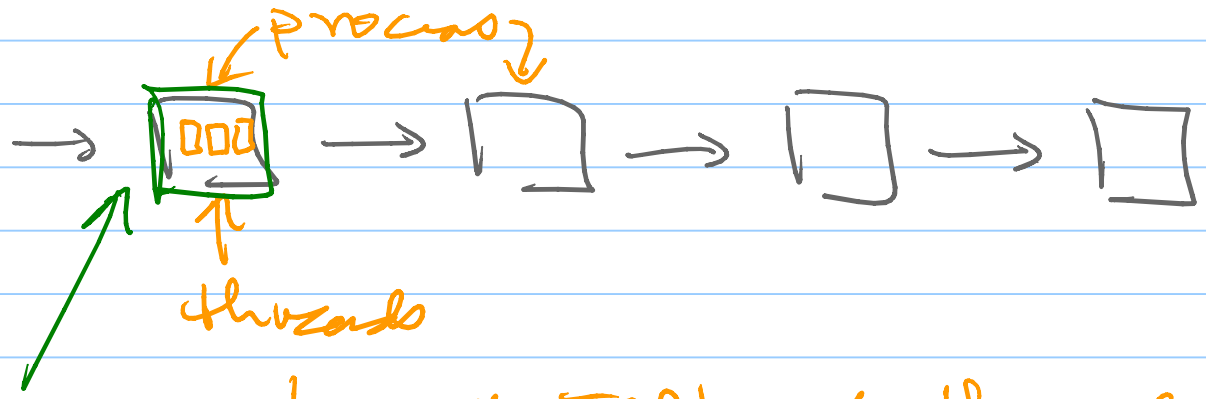


Dynamic coscheduling

- individual processors make scheduling decisions (independently)
- But cooperate → pick a better candidate based on other assignments



Process Aware Round Robin



↳ all TCB's of threads of a process is lumped together

- P processes & threads → allocate P.S time for ~~quanta~~ the process
  → round robin the threads (small time quanta)

Dynamic Partitioning

- Tasks poll to get ideal # of processors (each task is allocated an equal fraction of total processors)
- Task Processes can shave allocation (partitioning)

Hand off scheduling

- threads give hints to scheduler
  - discouragement hint
    - ↳ weak, mild, strong
      - do not run me
  - hand off hint
    - ↳ run a particular thread

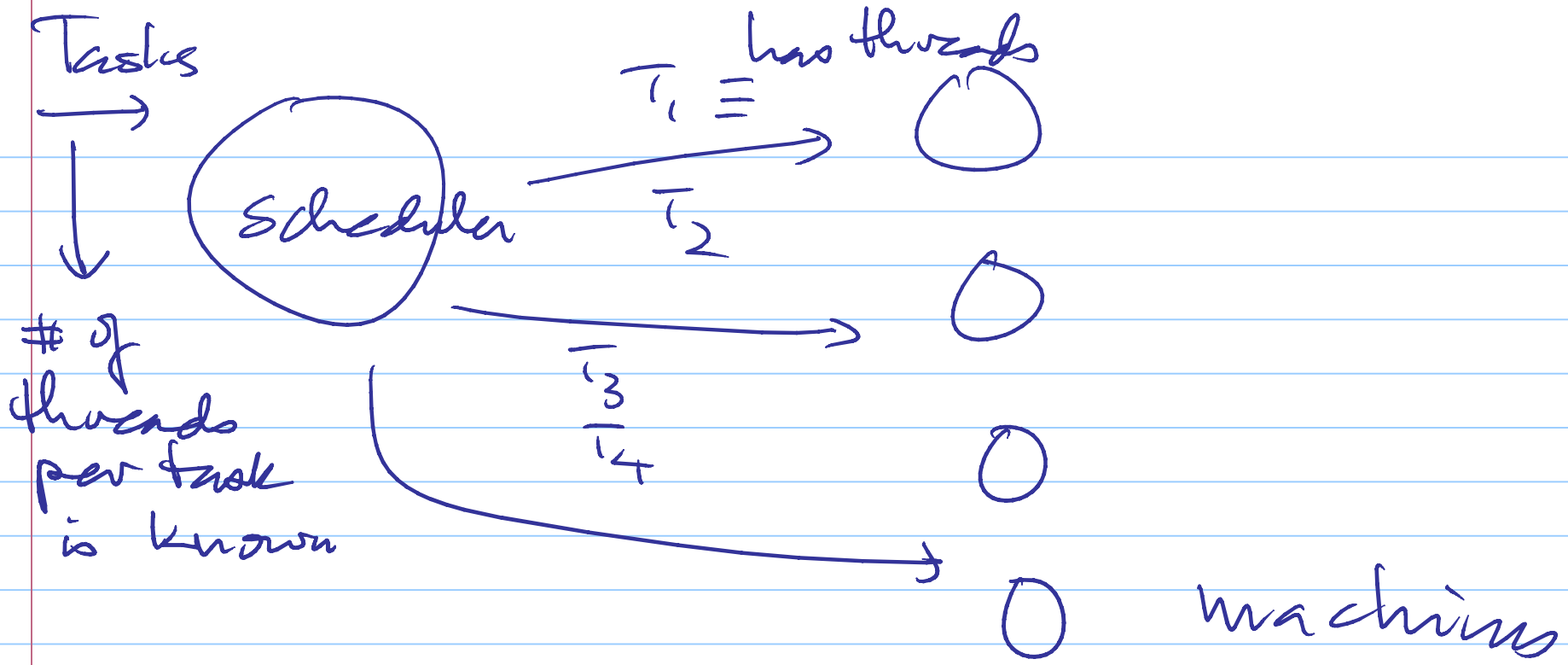
Affinity scheduling

- if thread x gets preempted from processor y then thread x should be scheduled on processor y the next time it runs

Clusters scheduling

↳ multiple machines on a fast network

- each machine has multiple cores (shared mem)
- process migration possible (hard)


Tasks
has threads
T_1 ≡
Scheduler
T_2
T_3
T_4
# of threads per task is known
machines

- Scheduler is aware of thread behaviour (creation, termination, etc)
  - scheduler can make migration decision