https://hao-ai-lab.github.io/dsc204a-w24/

DSC 204A: Scalable Data Systems
Winter 2024

Machine Learning Systems

Big Data

Cloud

Foundations of Data Systems

Logistics

- PA1: Release this Friday
  - Topic: Setup Ray and write some distributed programs using Ray
  - 2 weeks to finish
- TAs to use their office hours to do recitations for PA1
  - Will send out a form for you to vote for preferred length and time slot
- Reading summary clarification
  - You only need to **submit 2 pages in total for all readings** per week

Logistics

- Need one more class today to wrap up OS
- This week reading:
  - Understanding the cloud computing stack: SaaS, PaaS, and IaaS
  - Above the Clouds: A Berkeley View of Cloud Computing

Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia
(Comments should be addressed to abovetheclouds@cs.berkeley.edu)

- Two invited speaker confirmed (Dates TBD):
  - Stephanie Wang (Ray core main contributor, Incoming AP@UW)
  - Ion Stoica

Recap Practice

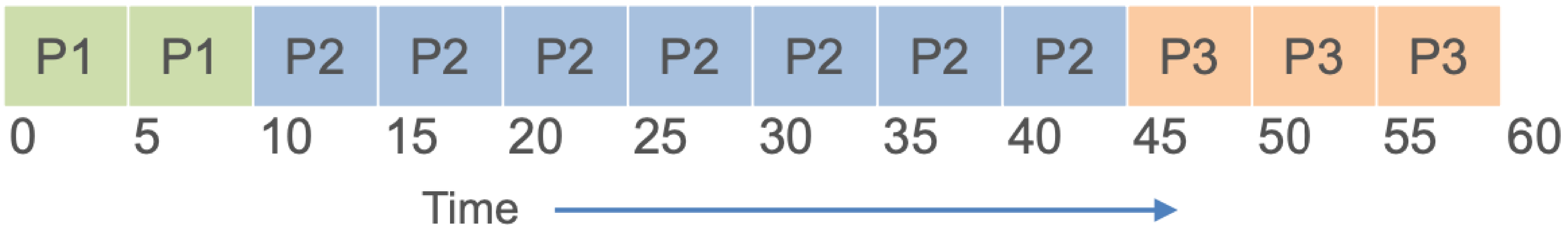
Here is a Gantt Chart for 3 processes of the given lengths that arrive at the different times given.

A) What is the rough *average response time*?

B) What is the rough *average turnaround time*?

C) Which scheduling policy/policies discussed in class (FIFO, SJF, SCTF, RR) may produce this given schedule? Explain clearly.

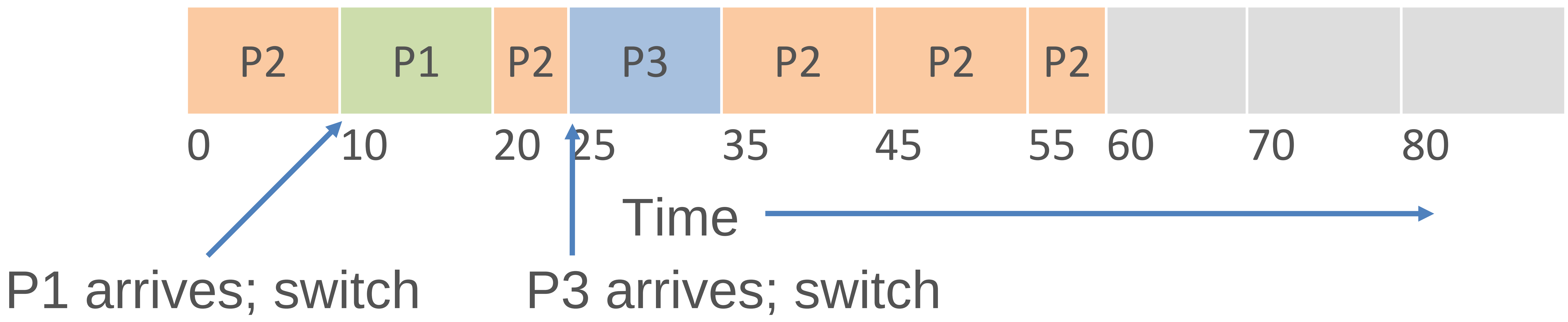
P1, P2, and P3 are of lengths 10, 35, and 15 units, resp. and arrive at times 0, 10, and 20, resp.



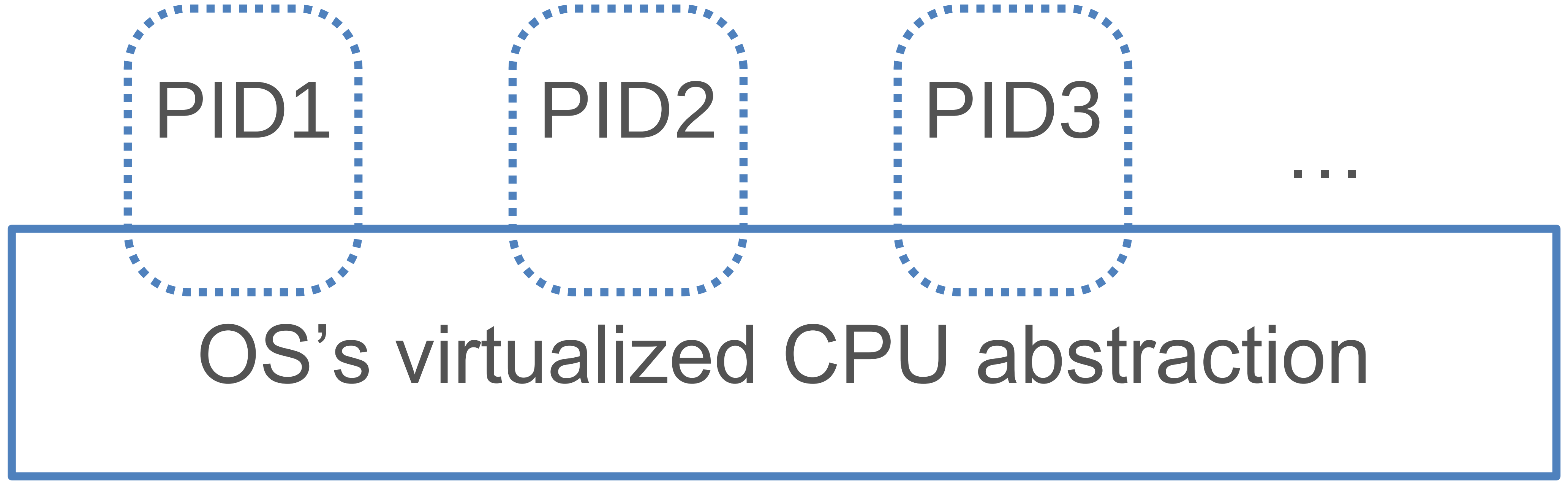
Review Preemption case: SCTF

- Shortest Completion Time First
- Jobs might not all arrive at same time; preemption possible

Example: P1, P2, P3 of lengths 10,40,10 units arrive at different times



Let's Implement It!



GAP2: How to virtualize CPU resources temporally and **spatially**?

Physical Processor

"Placement" naturally emerges:

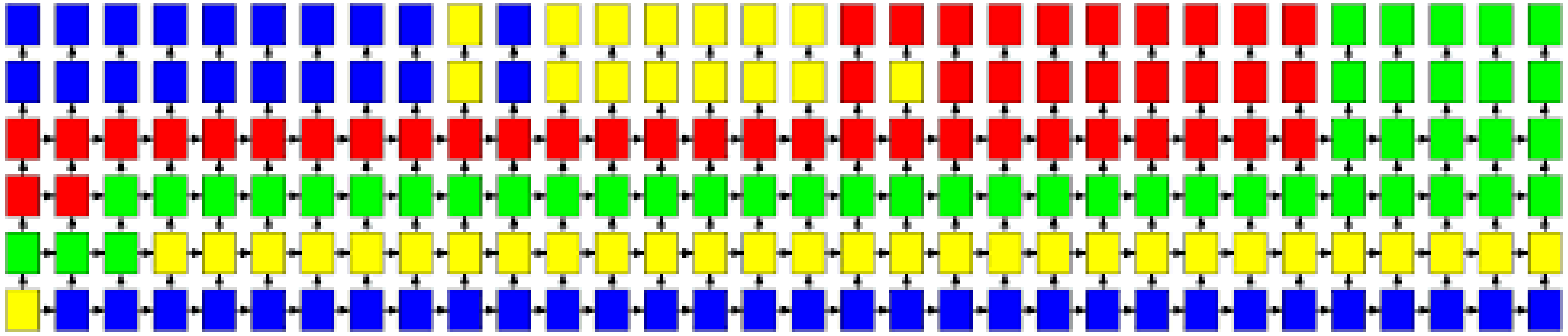
Q: how to place processes on each processor so **the objective** is optimal?

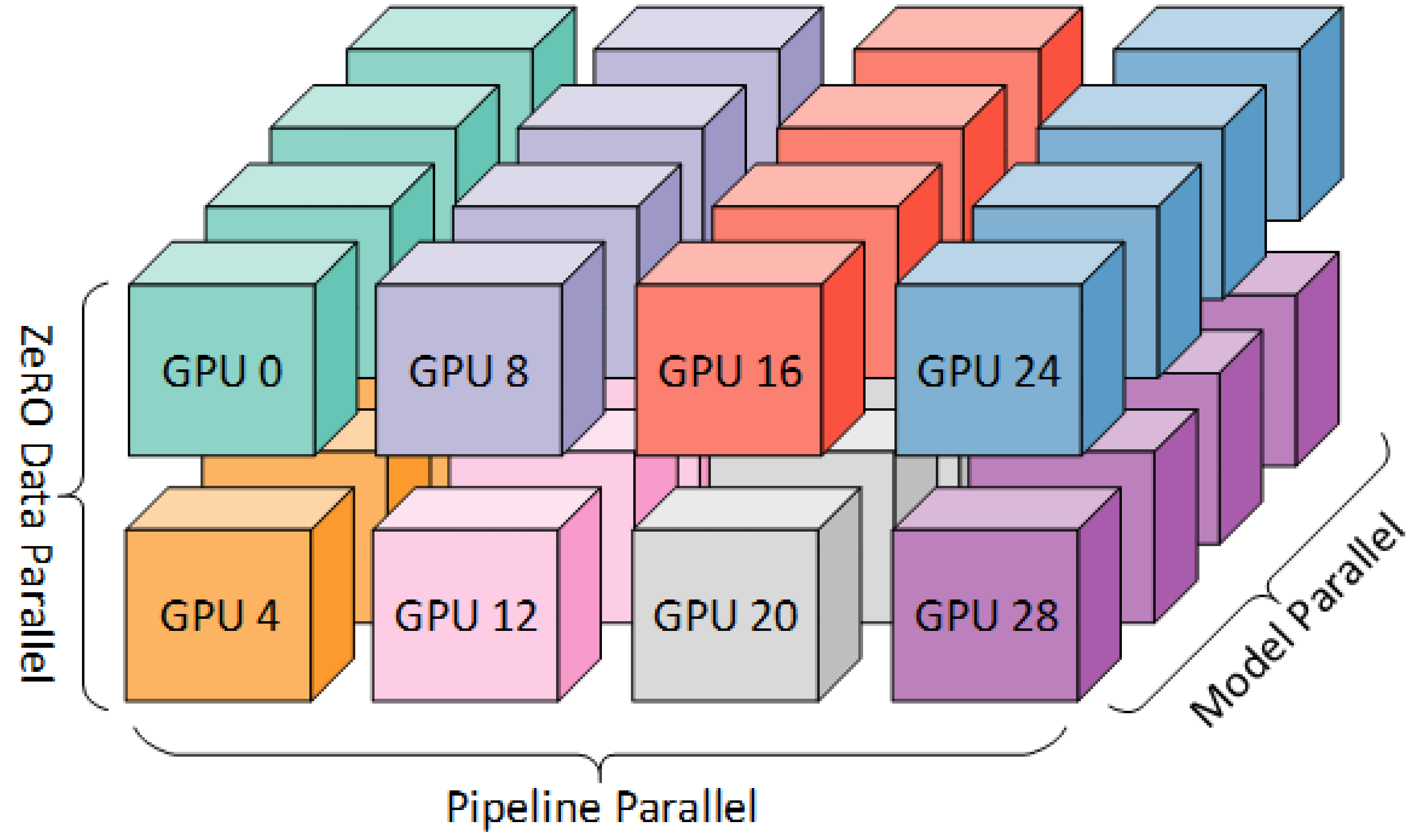
Concurrency

- Scheduling for multiprocessing/multicore is more complex
- **Load Balancing:** Ensuring different cores/proc. are kept roughly equally busy, i.e., reduce **idle times**
- Multi-queue multiprocessor scheduling (MQMS) is common
  - Each proc./core has its own job queue
  - OS moves jobs across queues based on load
  - Example Gantt chart for MQMS:

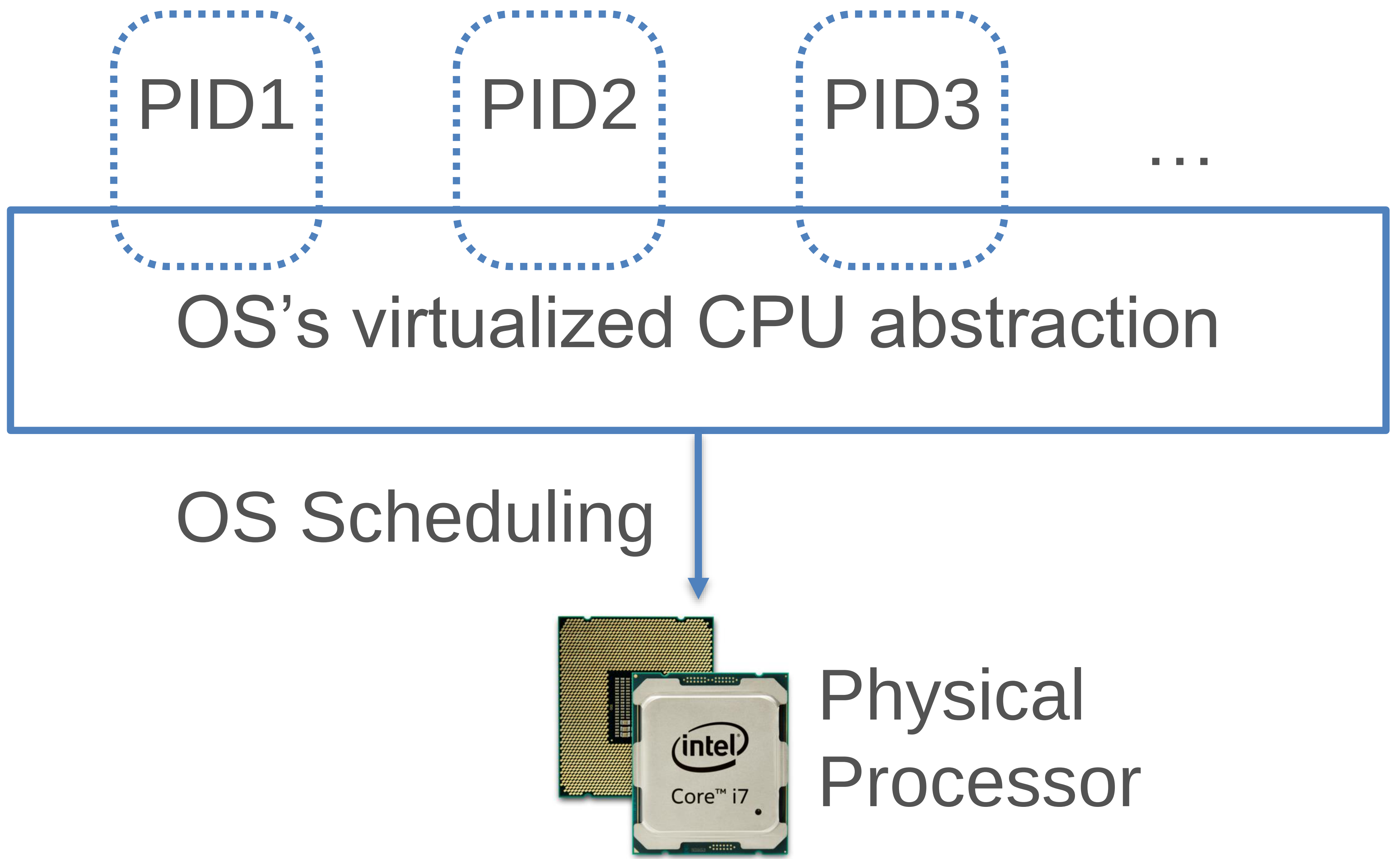
	0	10	20	30	40	50	60	70	80
CPU 1:	P1	P1	P3	P3	P3	P3	P1	P1	P1
CPU 2:	P2	P2	P2	P1	P1	P2	P2	P3	P3

Placement in Deep Learning





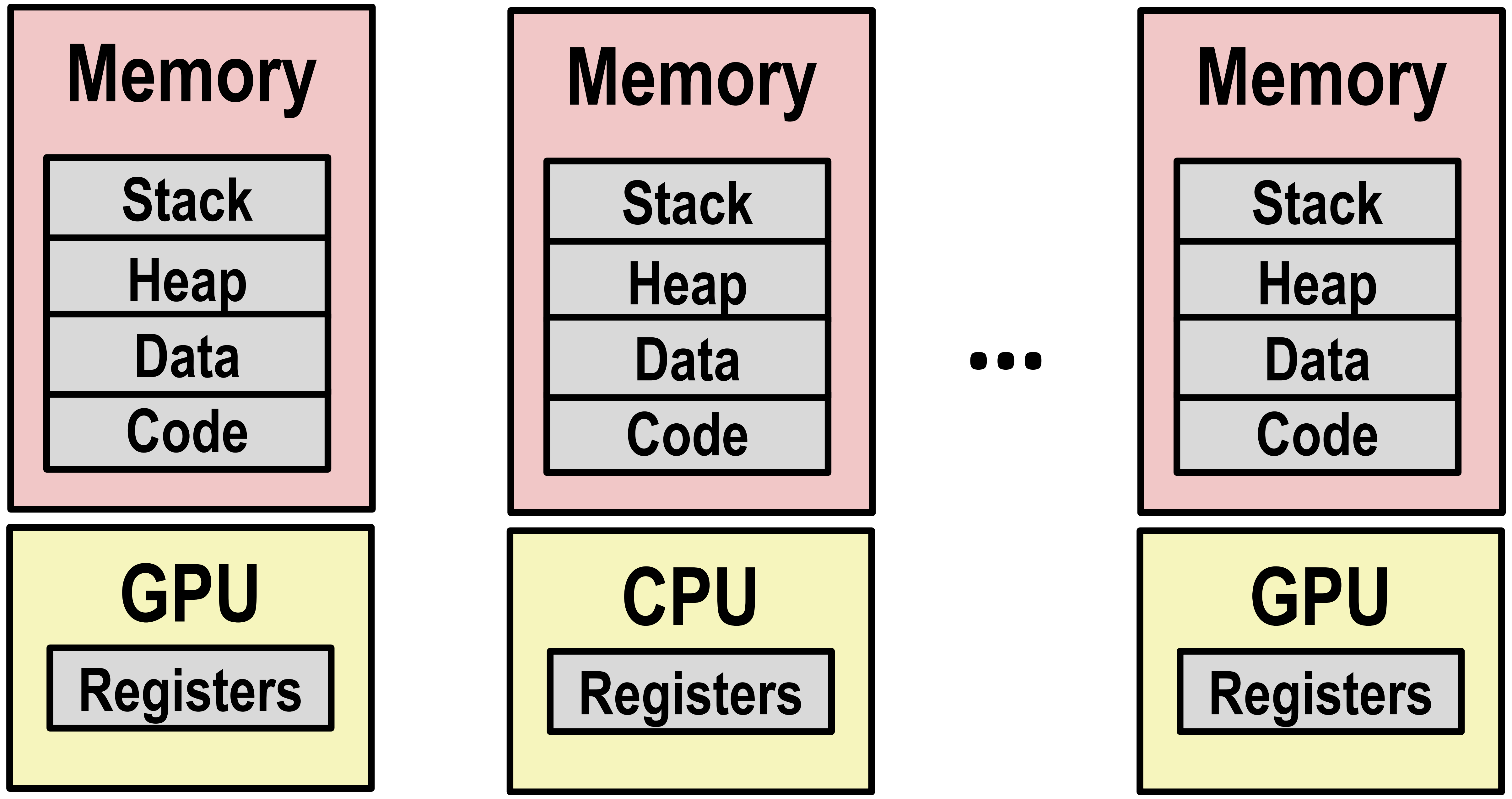
Last Issue in PM: Inter-process communication (IPC)



- Inter-process communication is provided in System Calls API

Recap

- ~~Strawman solution~~ **-> spatial-temporal sharing of CPUs with scheduling**
- Assign 1/3 of the memory to each APP



G1. Convenient?
G3: protection?
G2. Efficient?

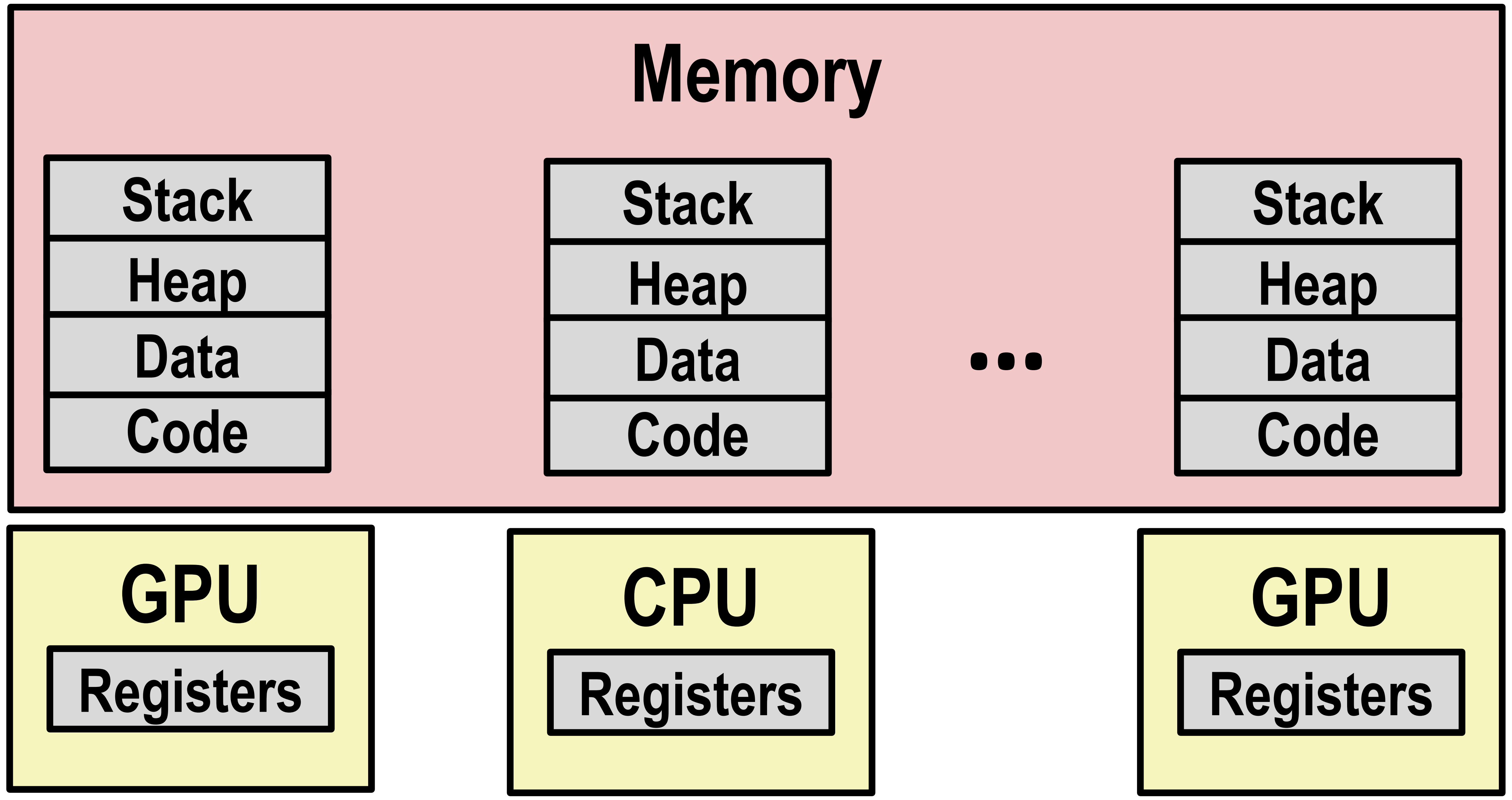
- G2.1 can I run N processes but not N times slower?
- **G2.2 can I run N apps with total mem > physical memory cap**

Foundation of Data Systems: where we are

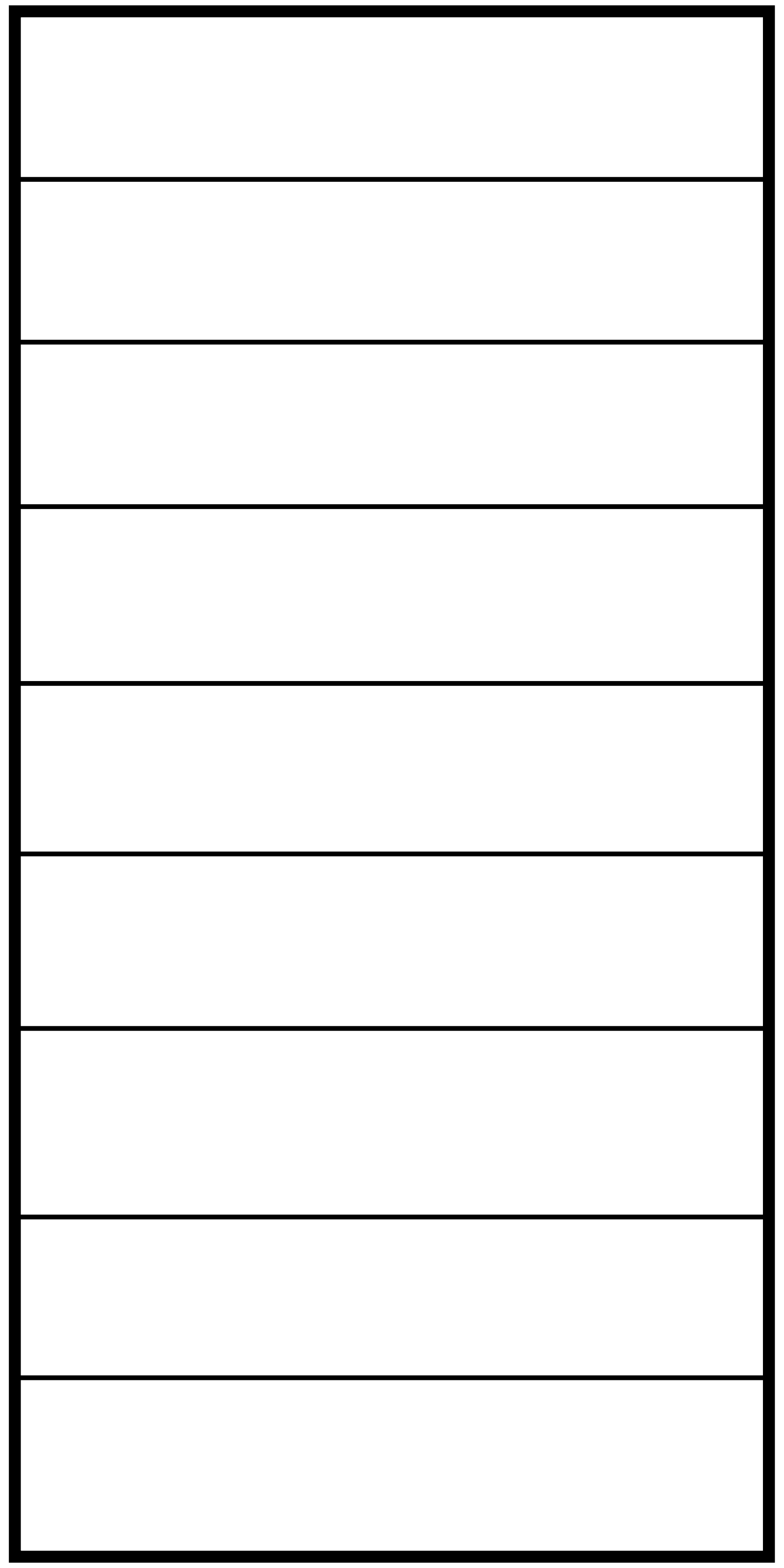
- Computer Organization
  - Representation of Data
  - Processors, memory, storages
- Operating System Basics
  - Process, scheduling, concurrency
  - **Memory management**
  - File systems

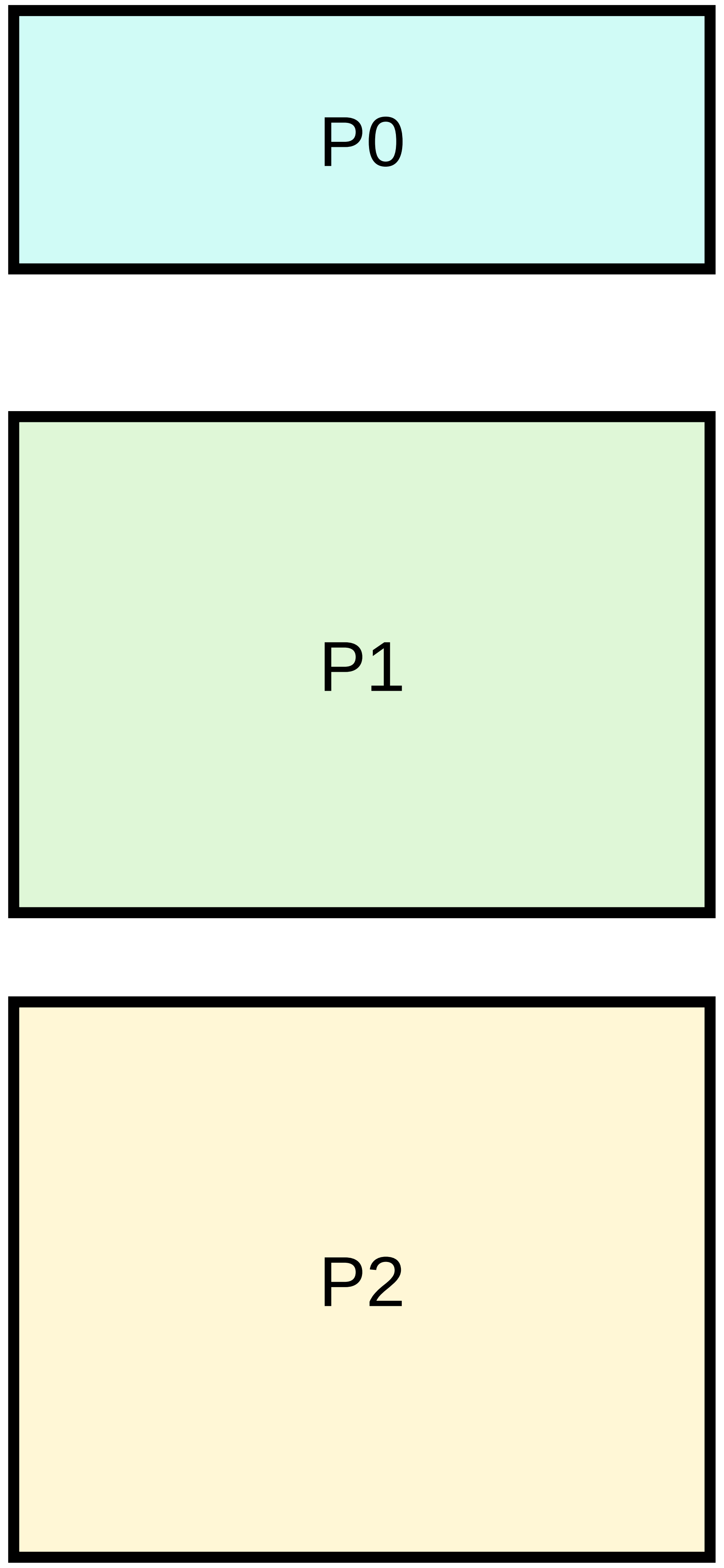
In Reality

- We have a pool of memory shared across many processes

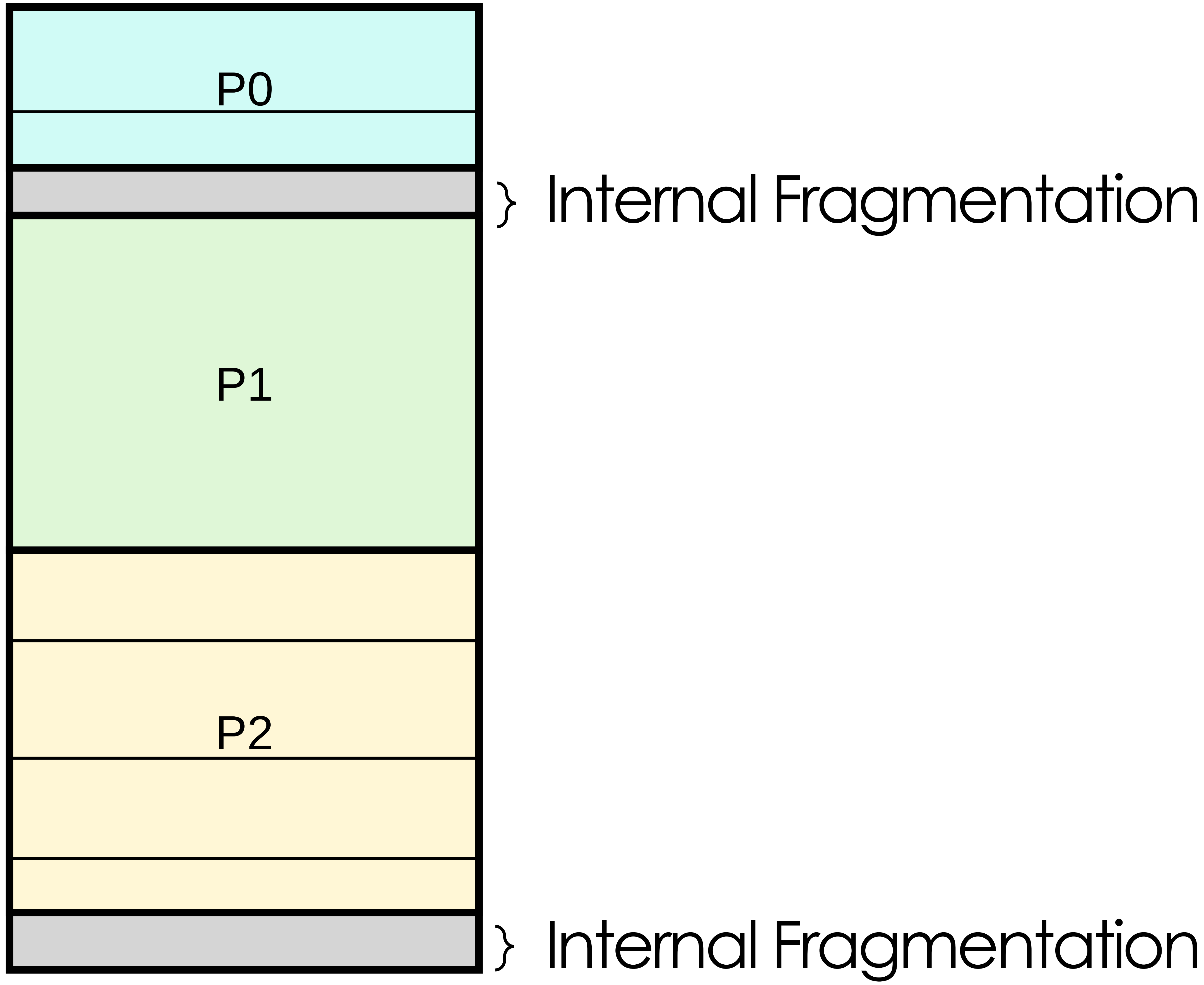


Memory management v0

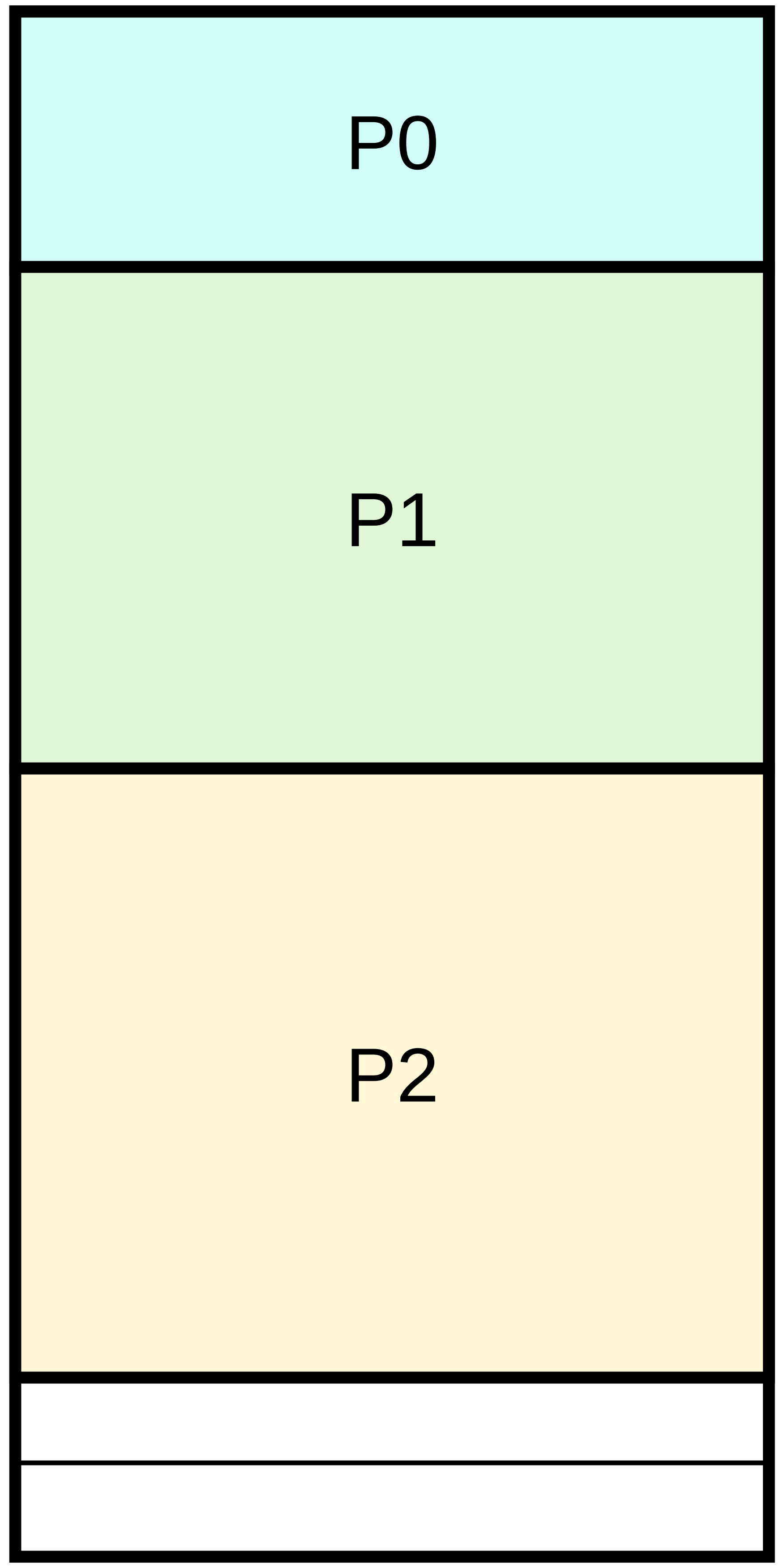




Memory management v0: Internal fragmentations

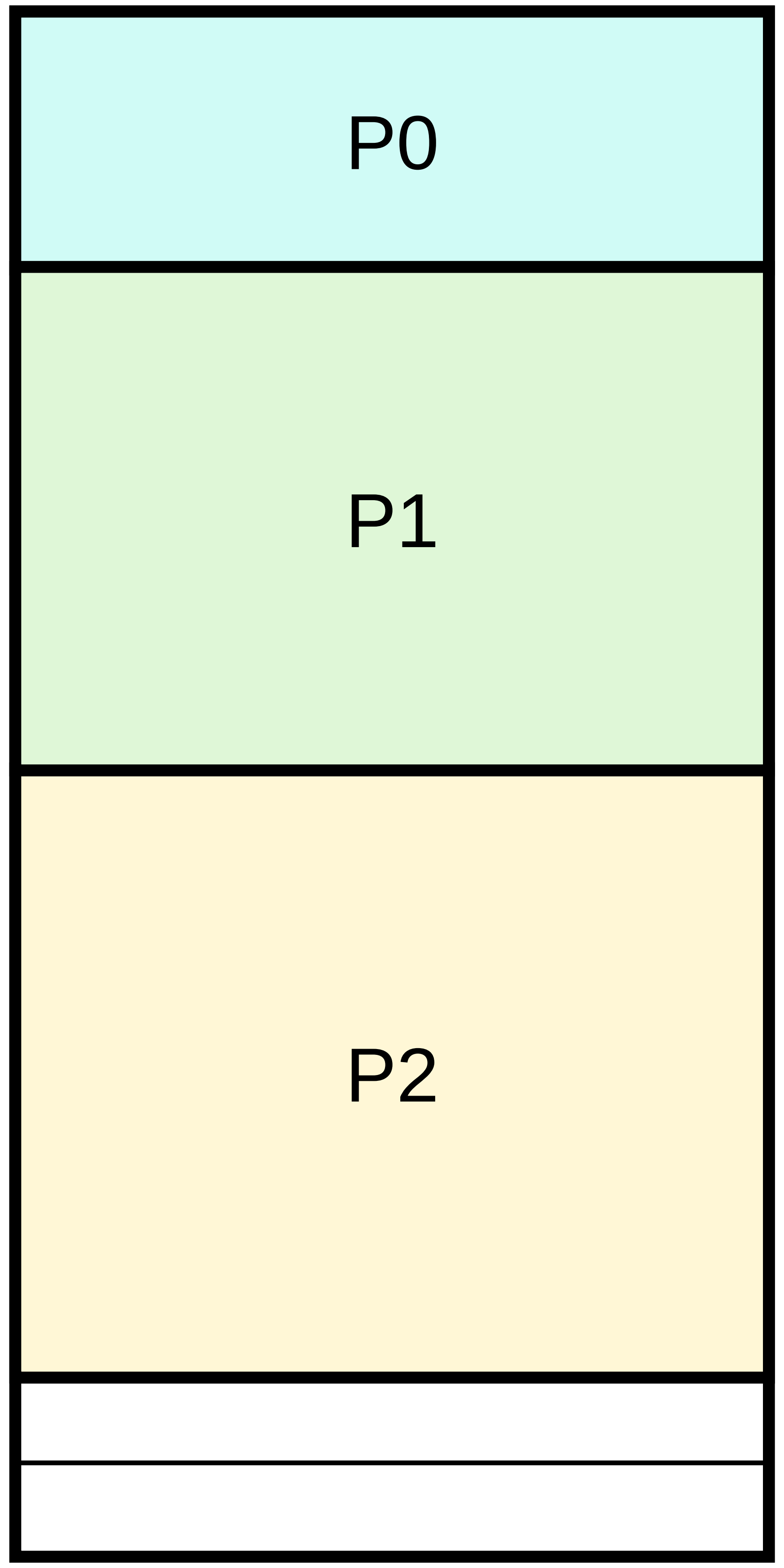


Memory management v1: use a smaller chunk

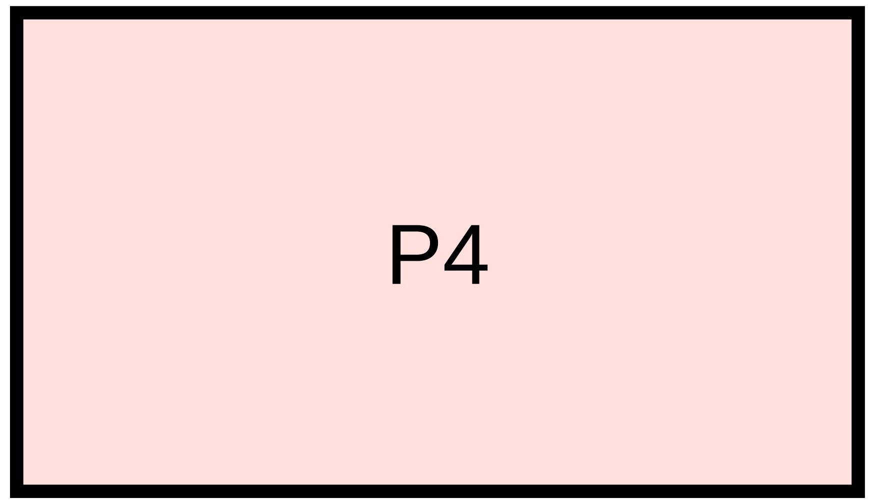


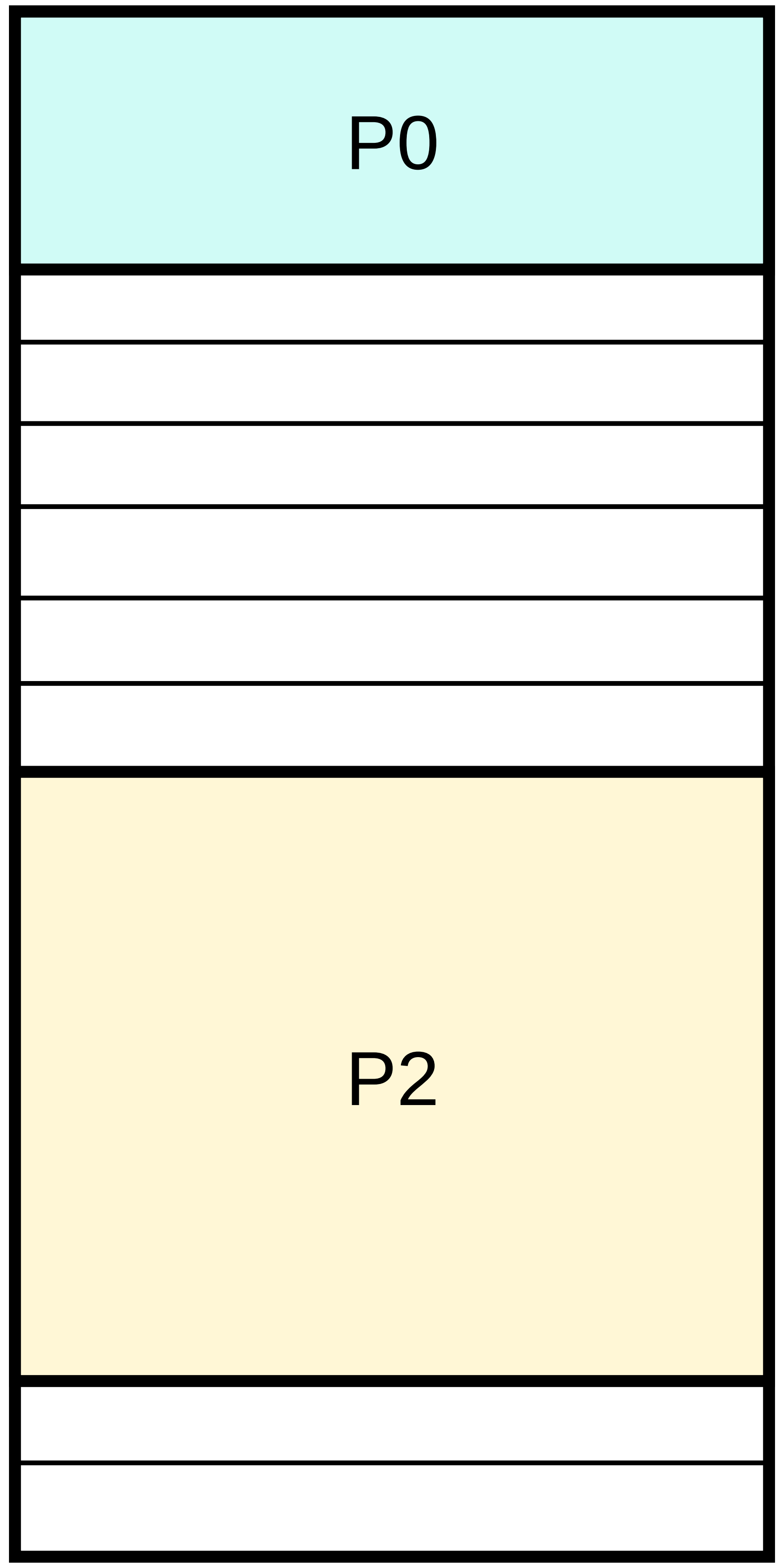
Q: What is the maximum possible amount of internal fragmentation per process?

Memory management v1

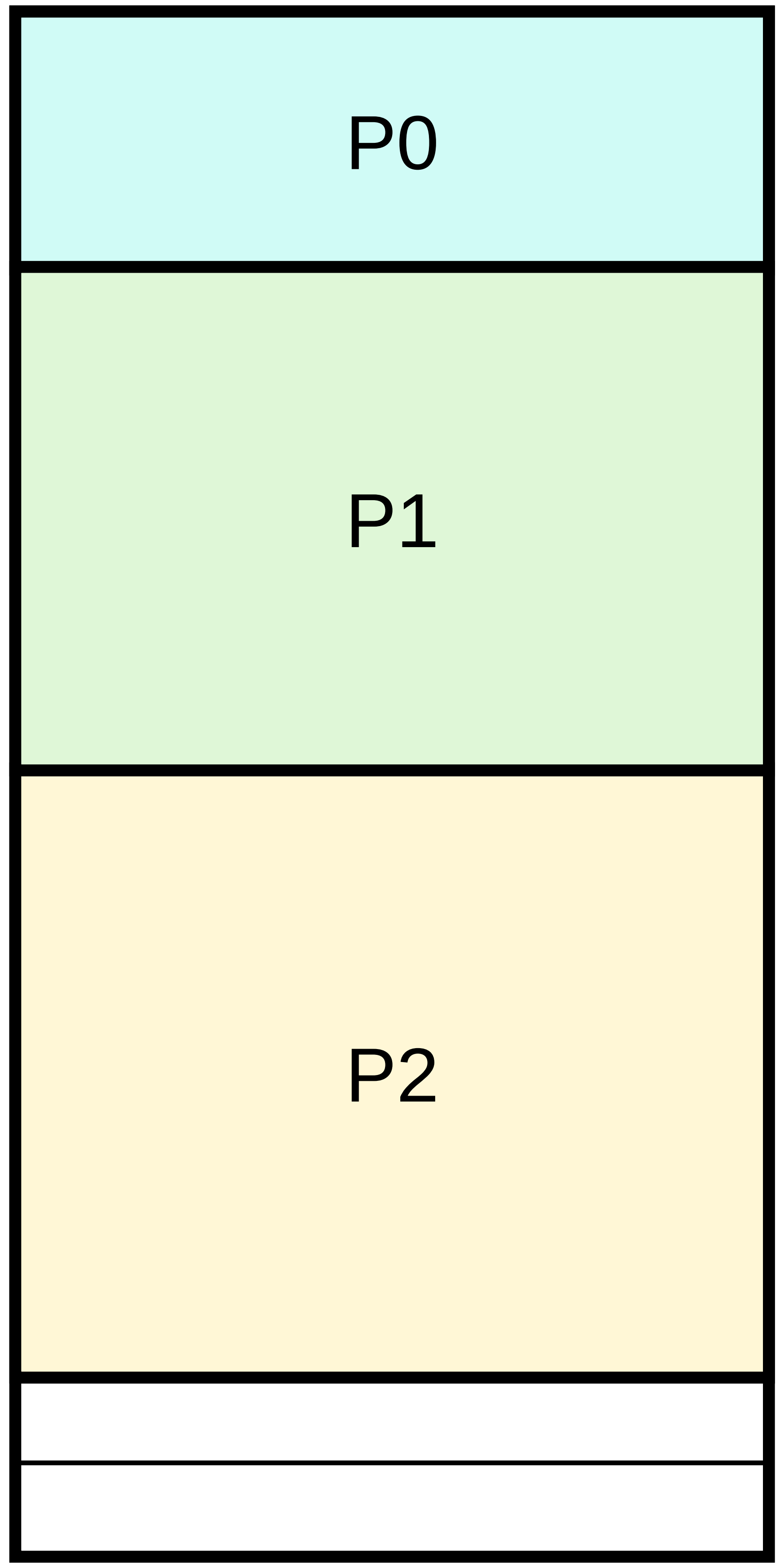


P1 finishes, P4 arrives

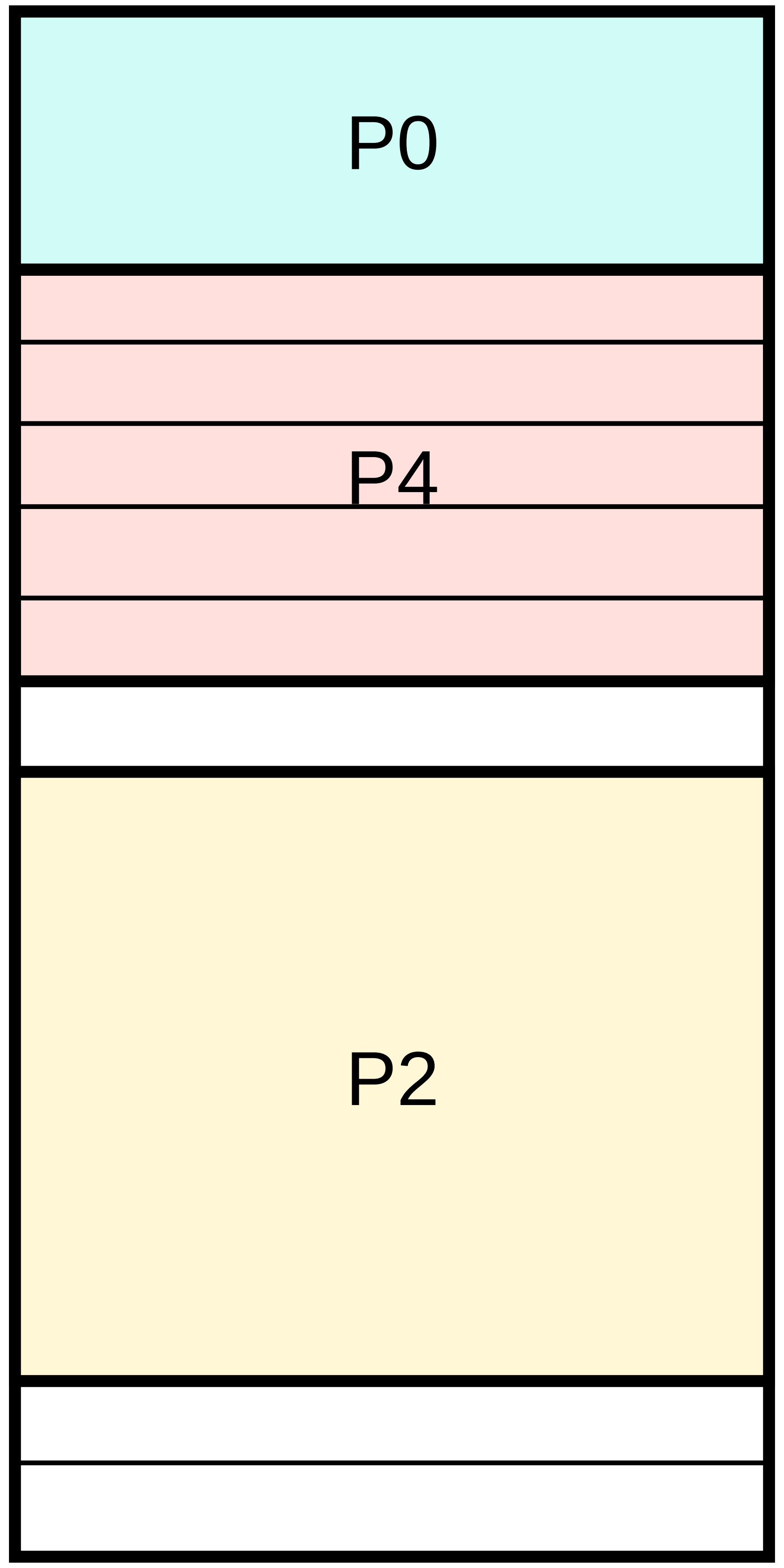




Memory: v2

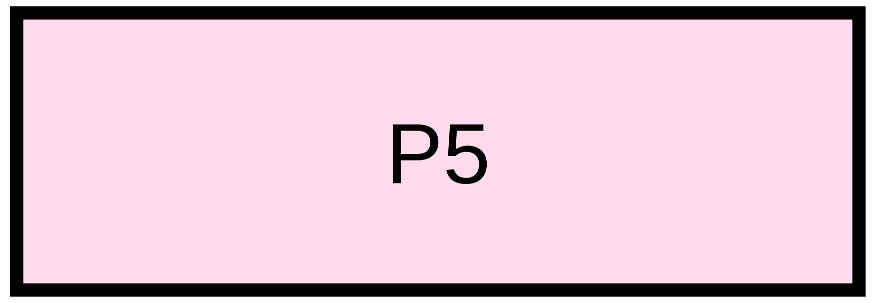


P4 scheduled



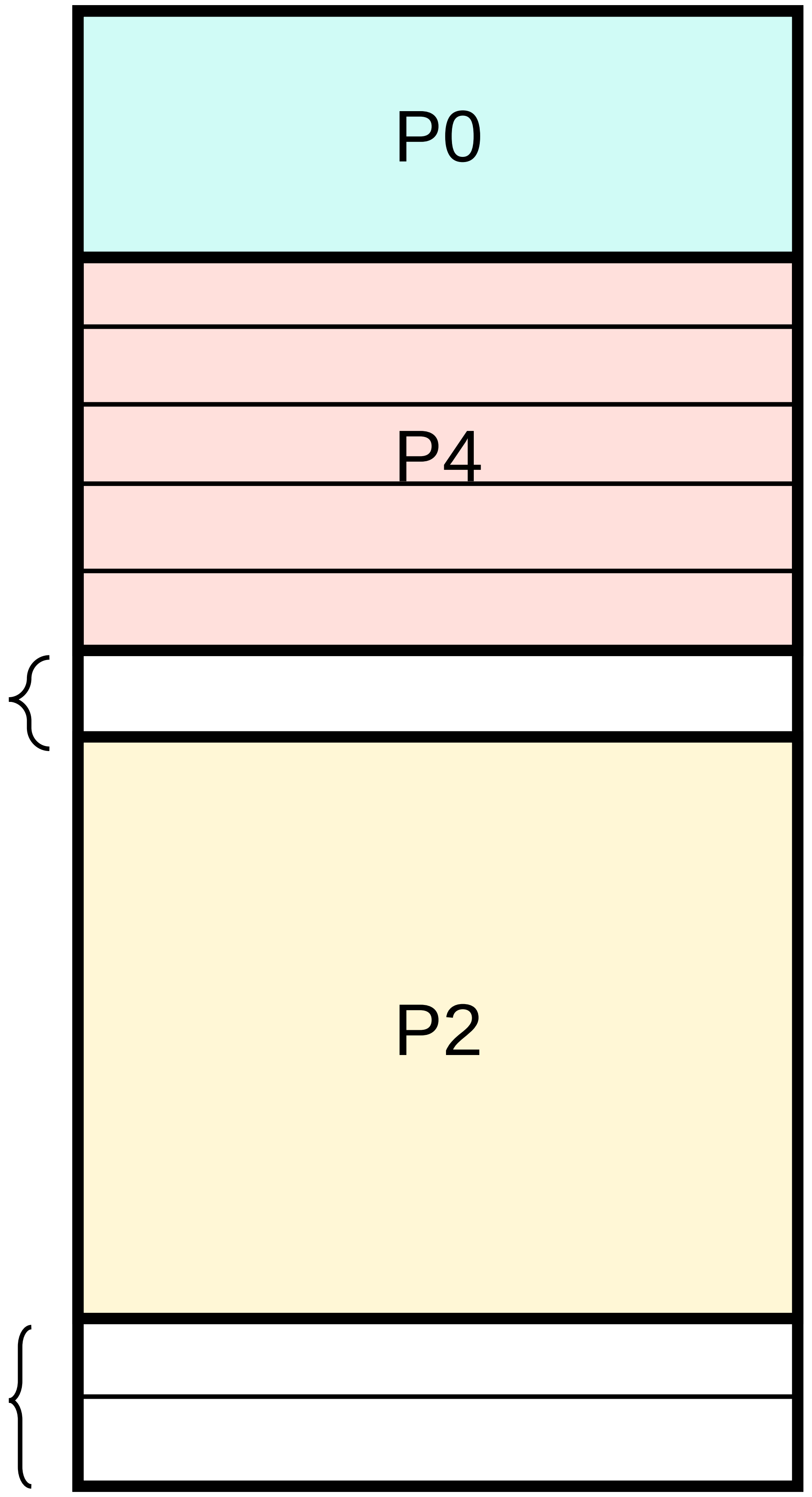
Memory: v2

P5 arrived



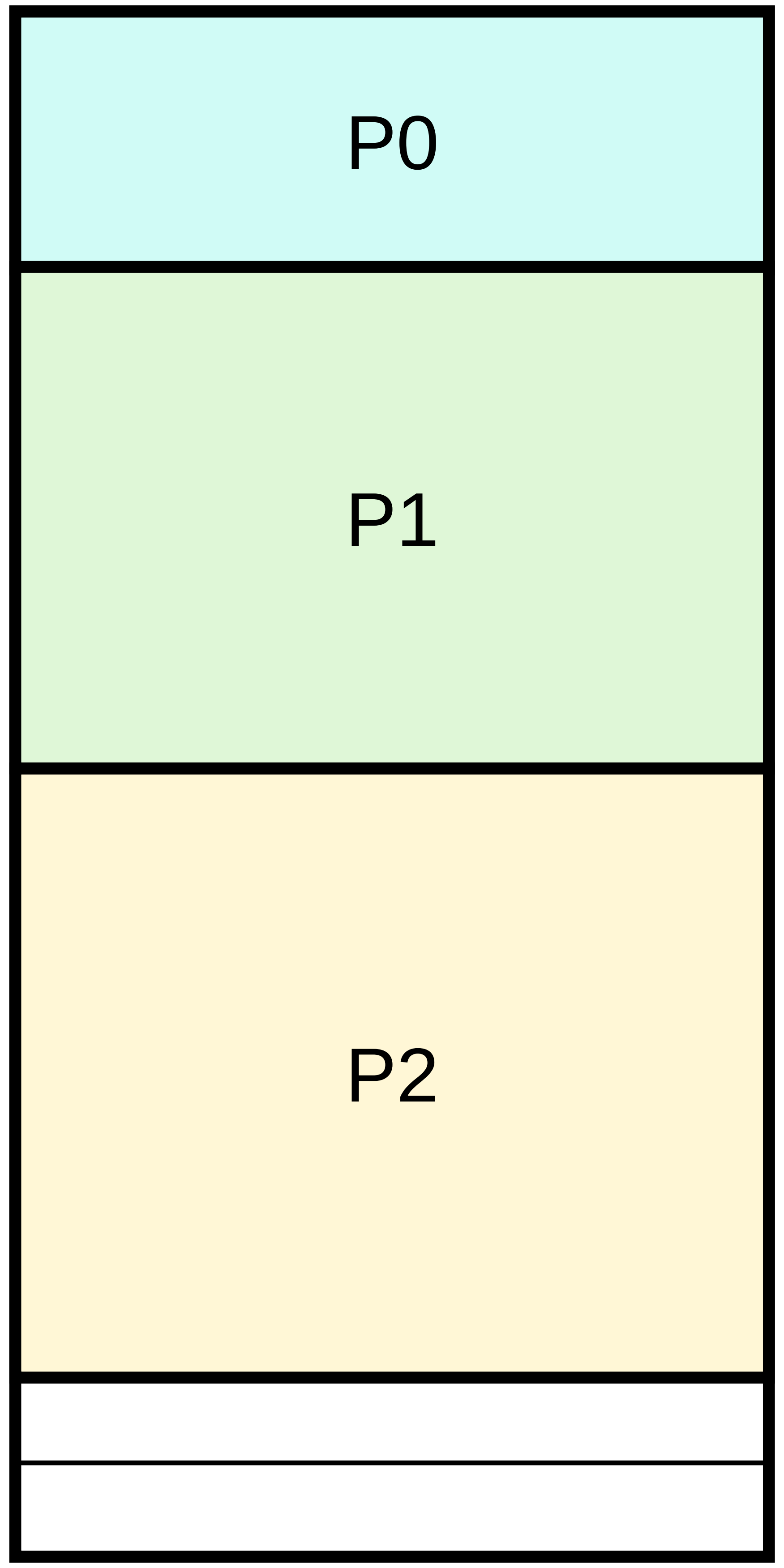
Problem:
There is enough memory for P5, but it cannot be scheduled.

Q: How to address external fragmentation?



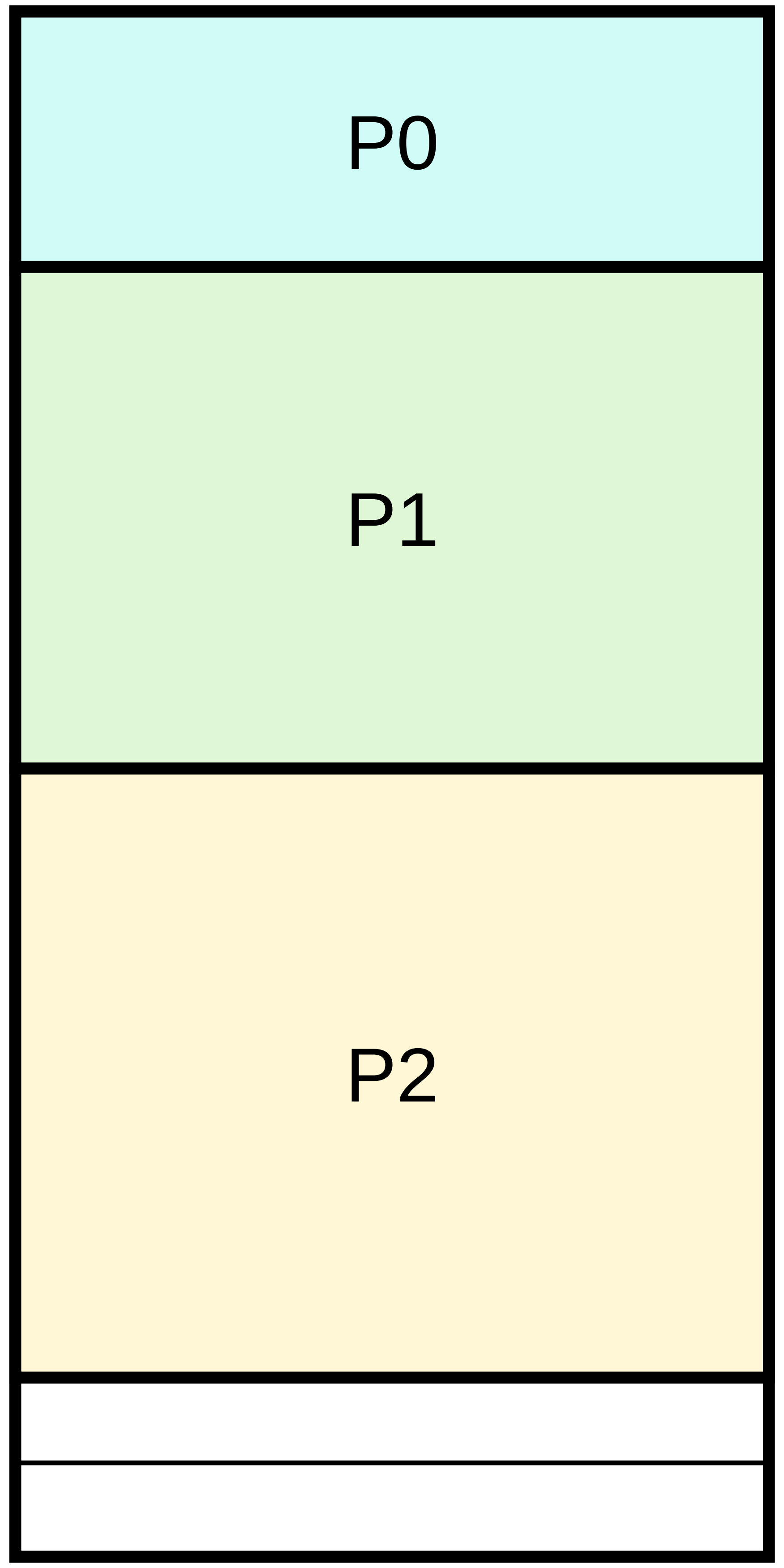
external fragmentation

Other Problems?



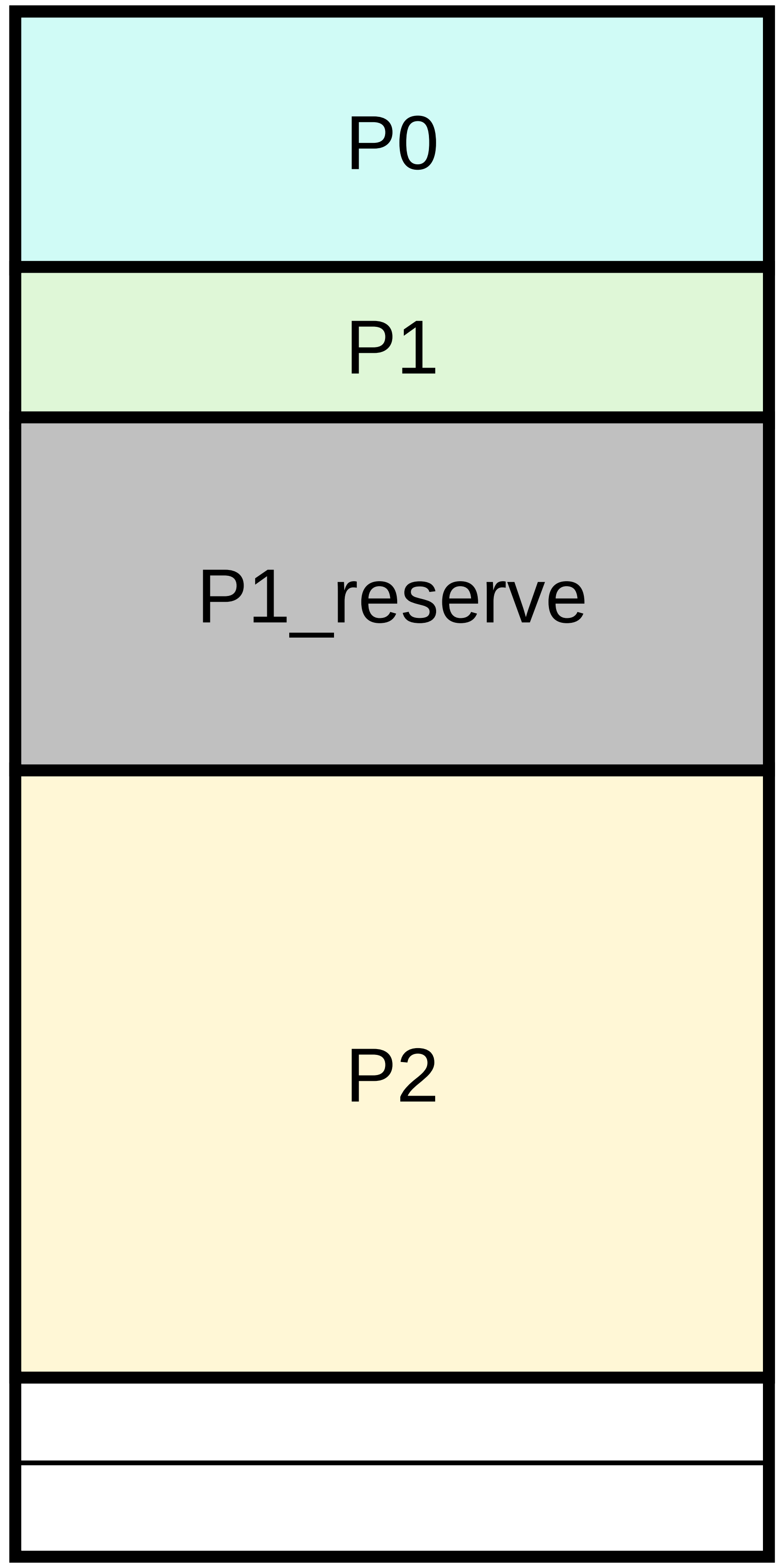
Problem: We can never schedule processes with their memory consumption greater than memory cap

Other Problems?



Problem:
What if we are unsure about how much memory P0/P1/P2 will eventually use?

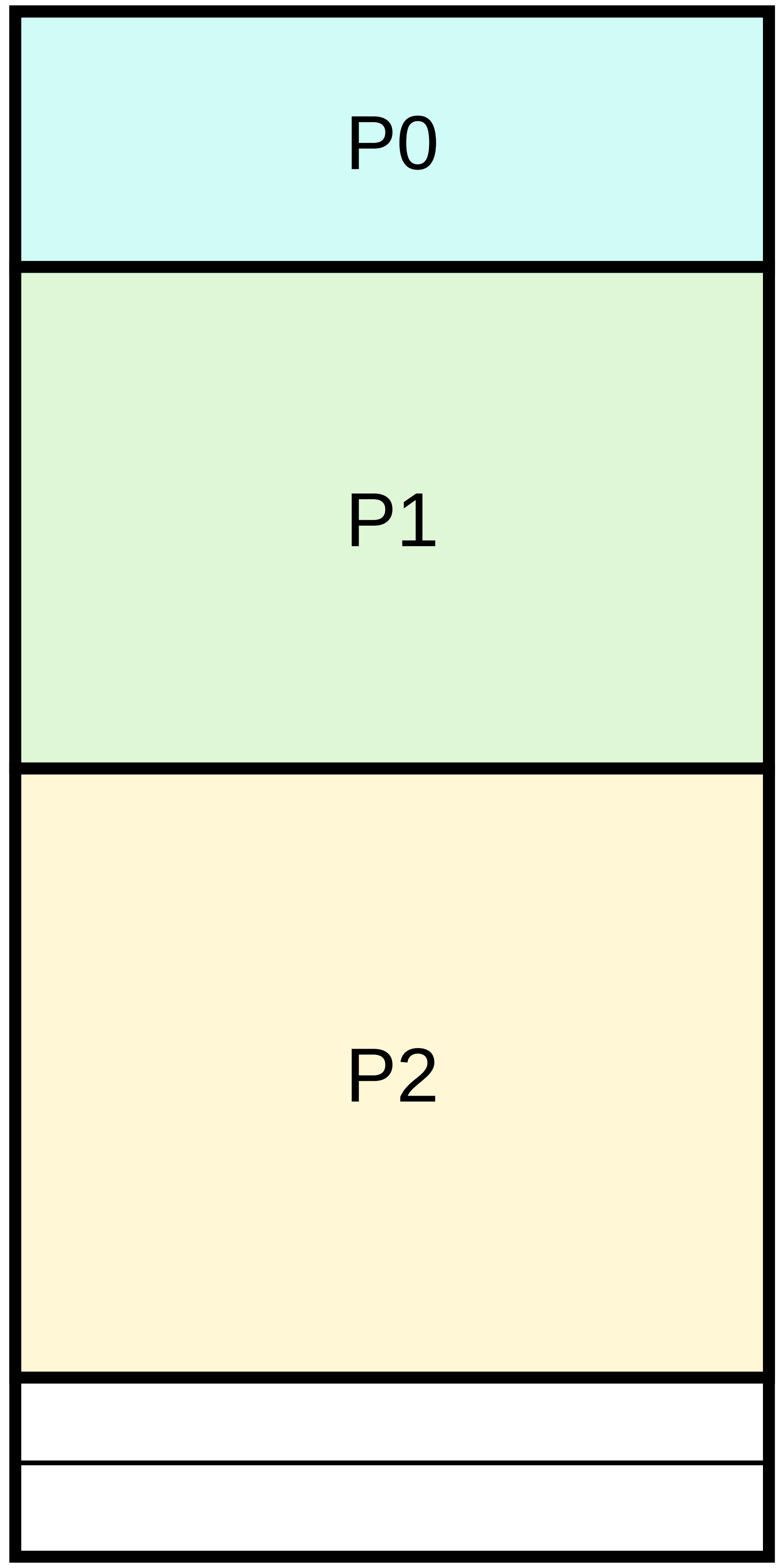
Other Problems?



Problem:
What if we are unsure about how much memory P0/P1/P2 will eventually use?

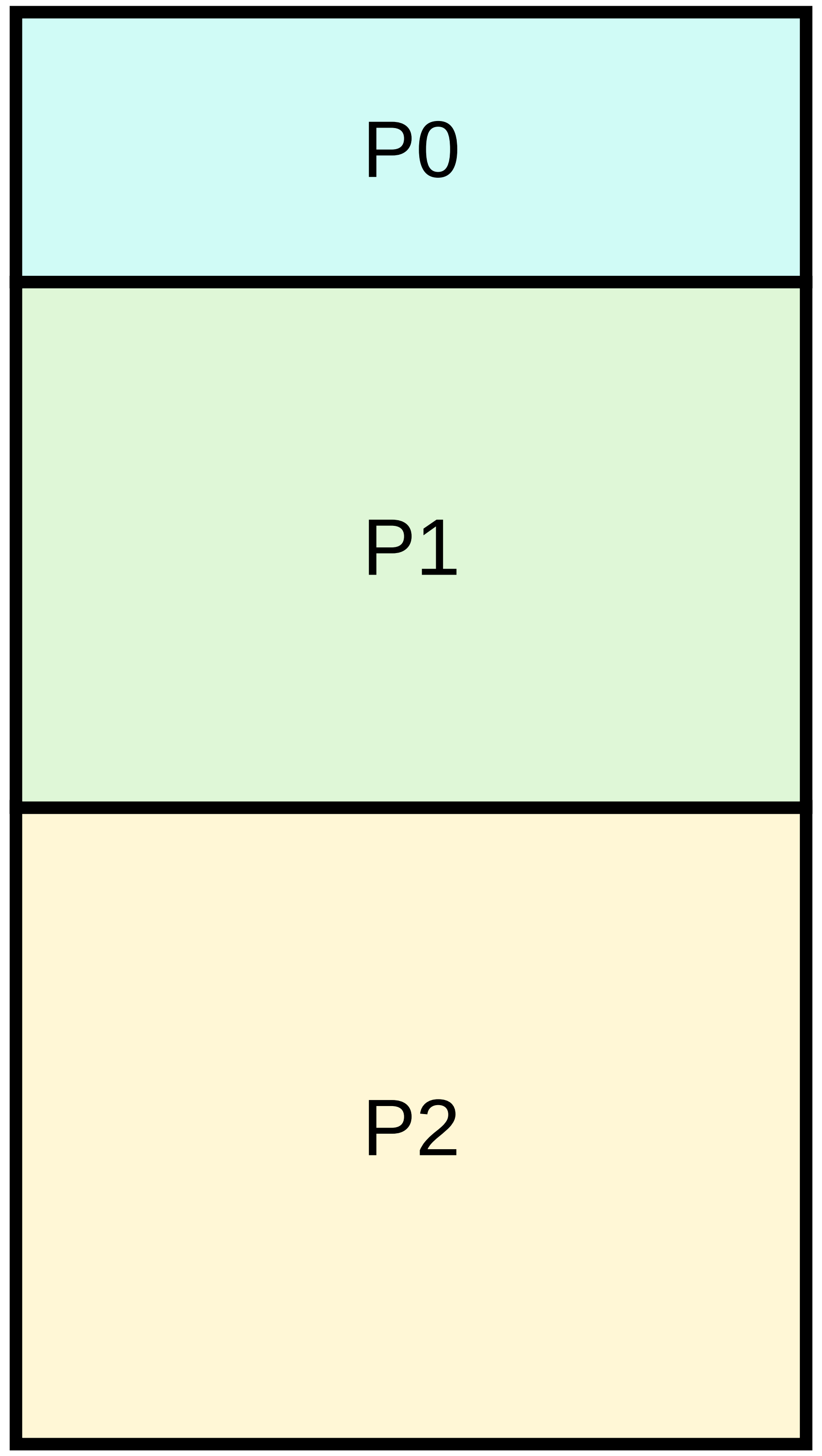
P1_reserve is the reservation overhead

Other Problems?

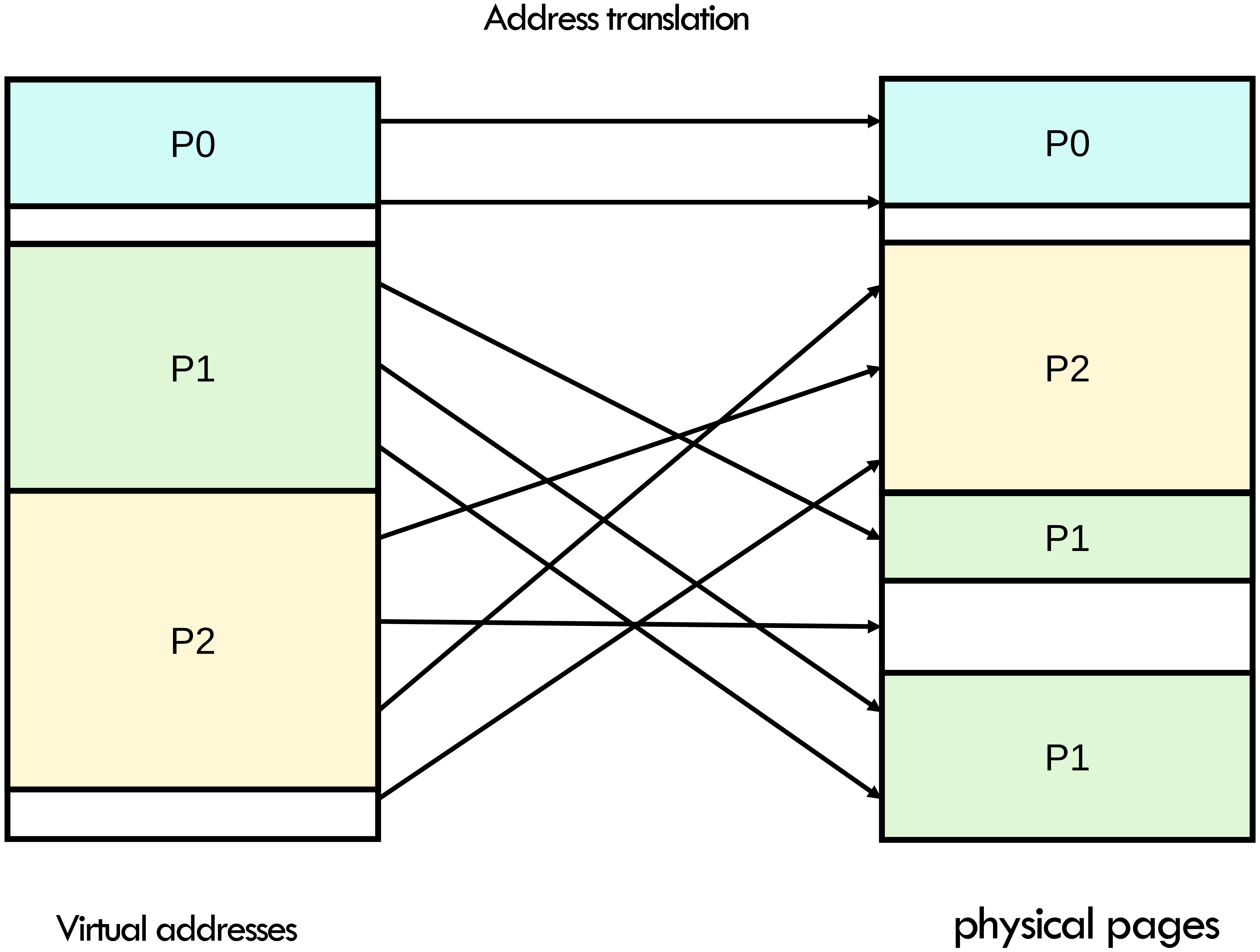


What if we **know exactly** how much memory P0/P1/P2 will **eventually** use, any problem?

Virtual Address Table



Processes is **given the impression** that it is working with large, near-infinite, contiguous memory



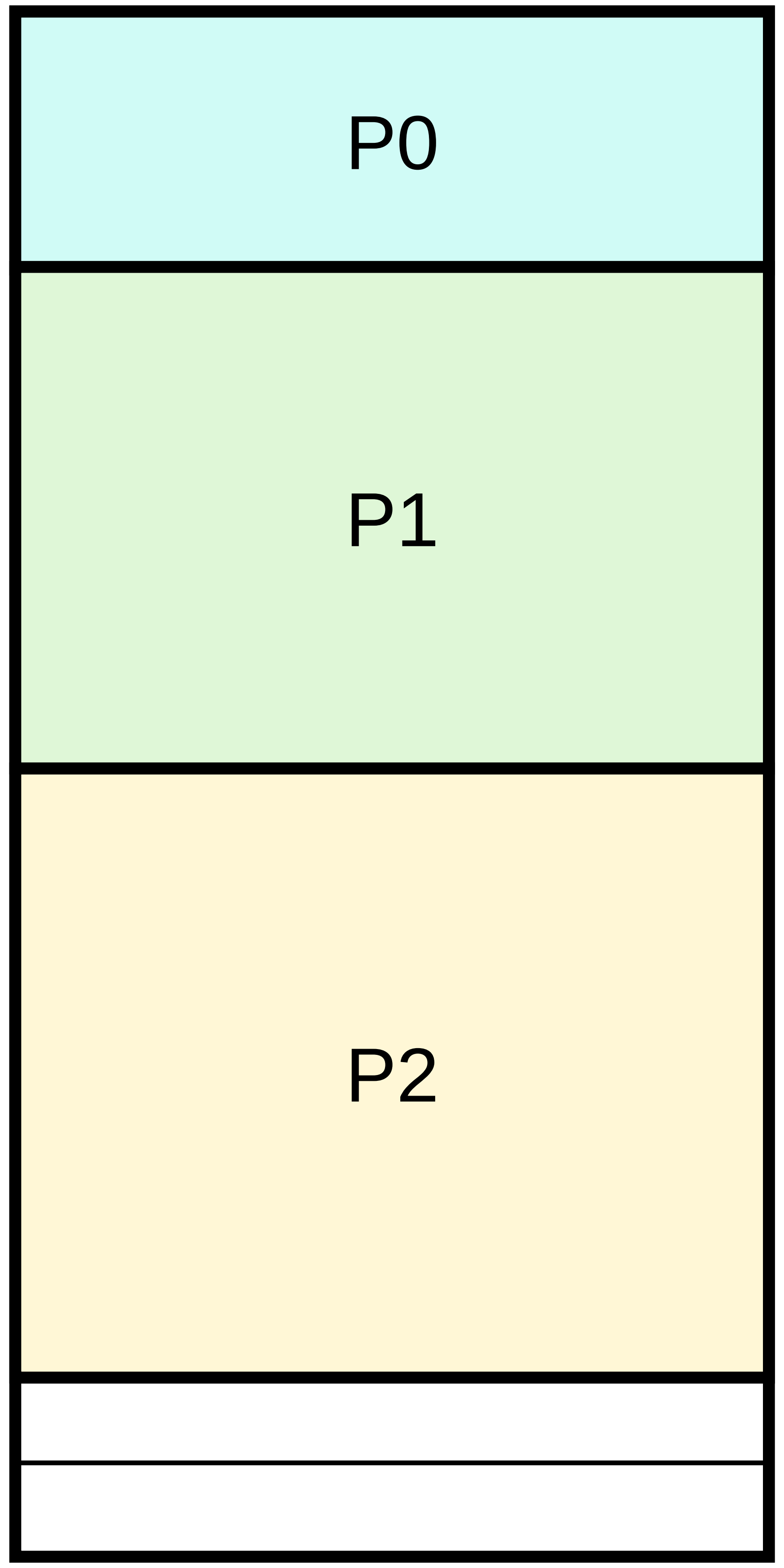
Pages and virtual memory

- **Page**: An abstraction of *fixed* size chunks of memory/storage
- **Page Frame**: Virtual slot in DRAM to hold a page's content
- Page size is usually an OS config
  - e.g., 4KB to 16KB
- OS **Memory Management** can
  - Identify pages uniquely
  - Read/write page from/to disk when requested by a process

Virtual Memory

- **Virtual** Address vs **Physical** Address:
  - Physical is tricky and not flexible for programs
  - Virtual gives "isolation" illusion when using DRAM
  - OS and hardware work together to quickly perform **address translation**
  - OS maintains **free space list** to tell which chunks of DRAM are available for new processes, avoid conflicts, etc.

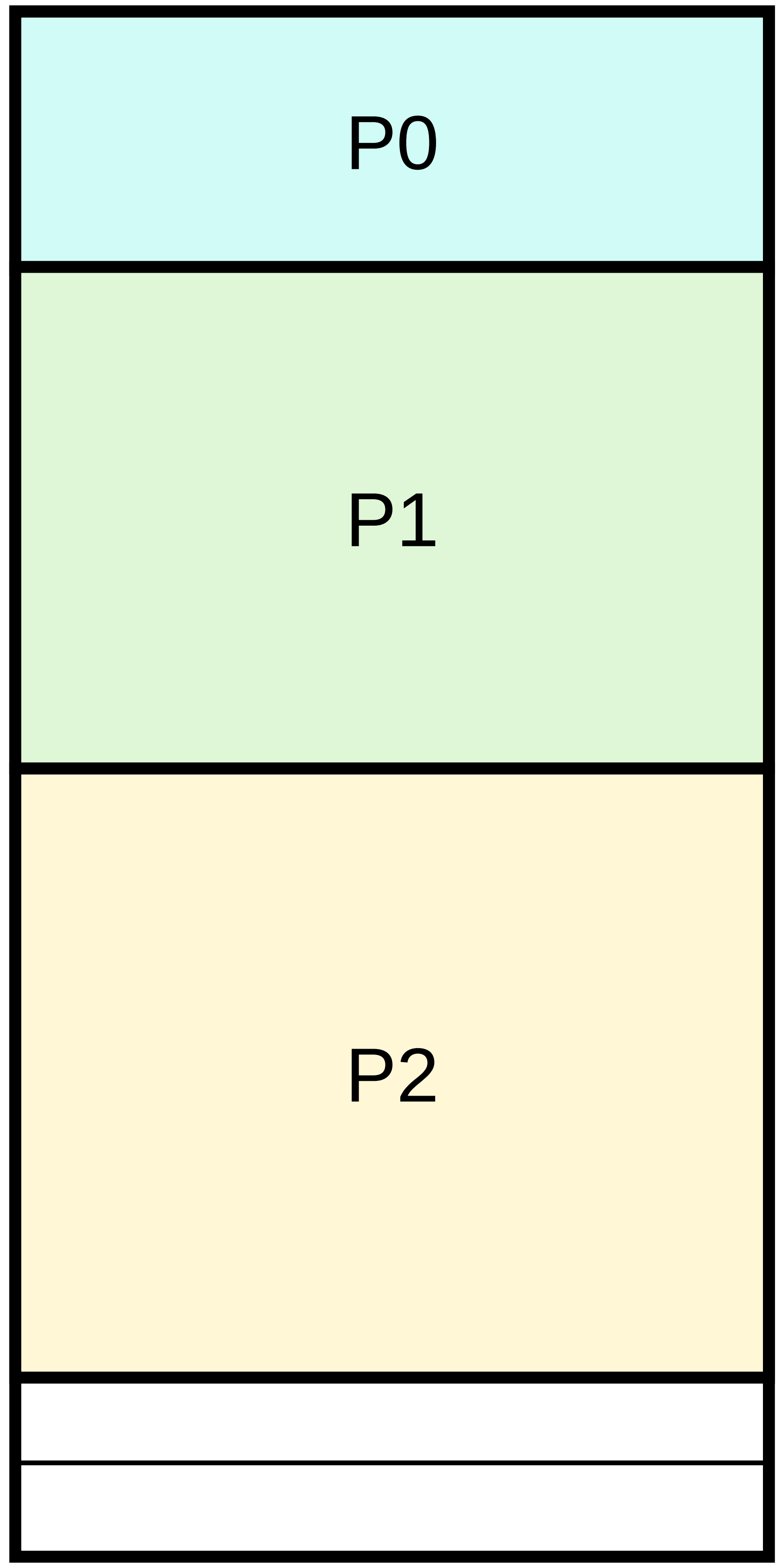
Problem addressed?



Problem: We can never schedule processes with their memory consumption greater than memory cap

Solution: create more virtual addresses than physical memory cap. Map additional ones to disk.

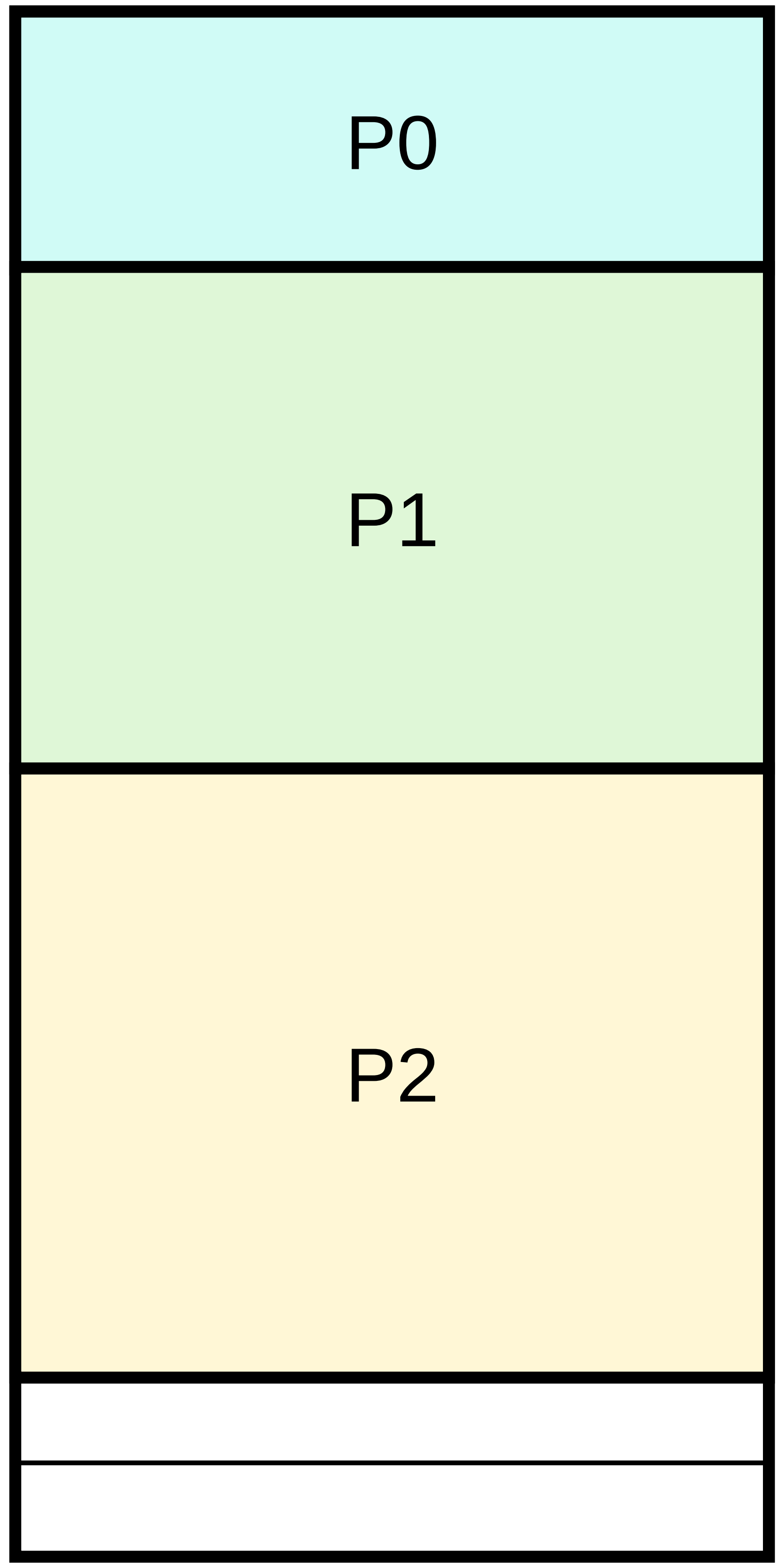
Problem addressed?



Problem:
What if we are unsure about how much memory P0/P1/P2 will eventually use?

Reserve on virtual tables, resolve the mapping between virtual and physical pages on-the-fly

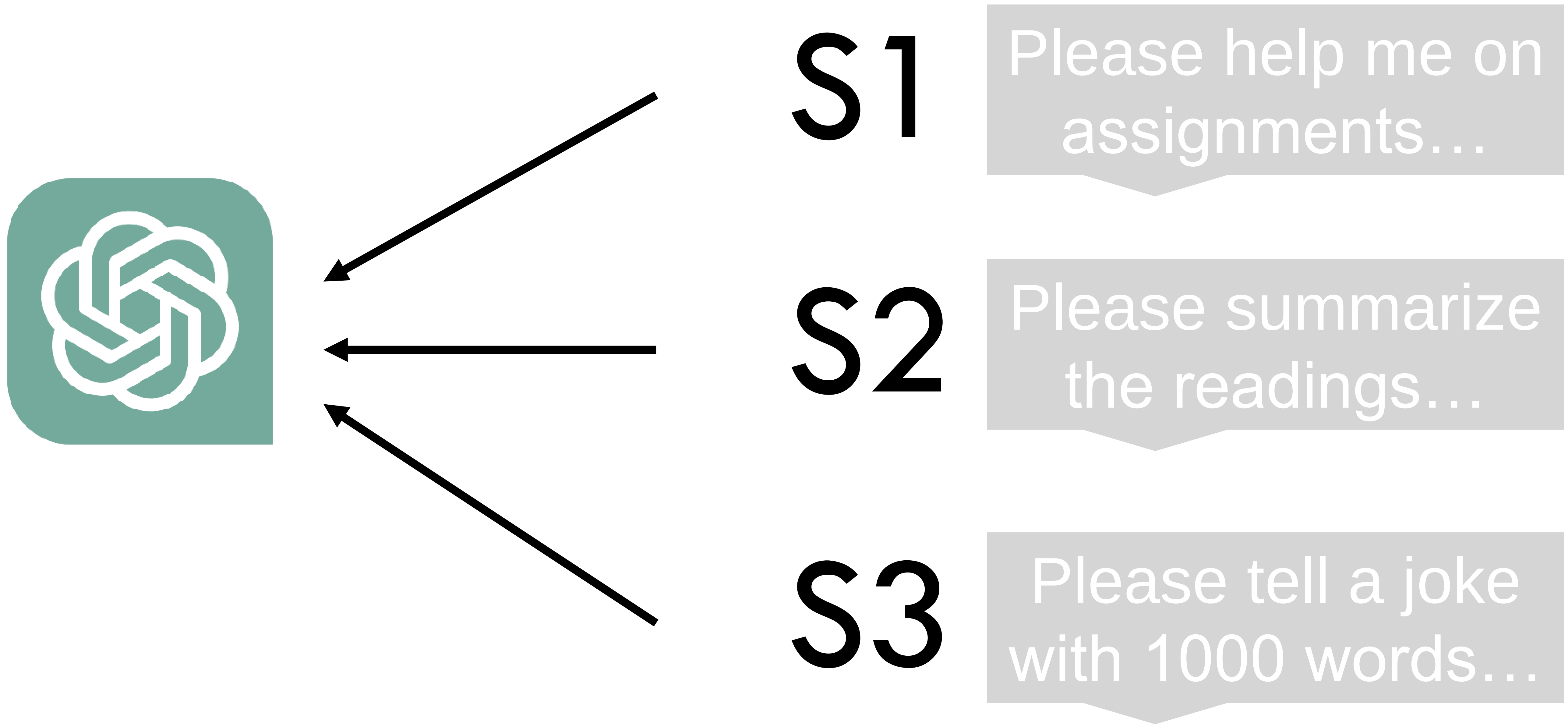
Problem addressed?



What if we **know exactly** how much memory P0/P1/P2 will **eventually** use, any problem?

Because we do everything on the fly – we minimize opportunity cost

Scheduling in ChatGPT



- How to allocate memory for LLM query?
  - Hint: think each LLM query as a process
- Q: Why could this make per LLM request cheaper?

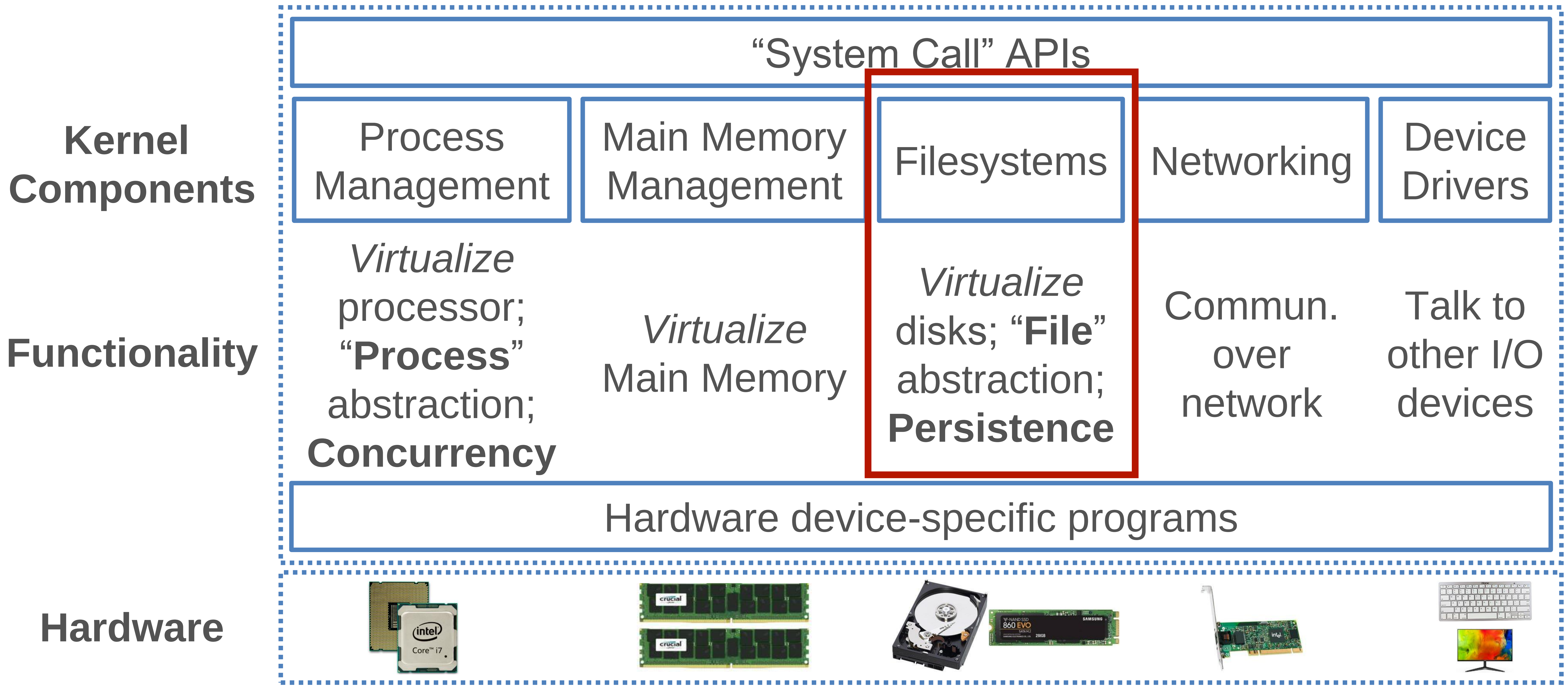
Efficient memory management for large language model serving with pagedattention
W Kwon, Z Li, S Zhuang, Y Sheng, L Zheng, CH Yu, J Gonzalez, H Zhang, ...
Proceedings of the 29th Symposium on Operating Systems Principles, 611-626

Foundation of Data Systems: where we are

- Computer Organization
  - Representation of Data
  - Processors, memory, storages
- Operating System Basics
  - Process, scheduling, concurrency
  - Memory management
  - **File systems**

Modules

- **System call:** The core of an OS with modules to abstract the hardware and APIs for programs to use



Q: *What is a file?*


MORTGAGE
INVESTMENTS
SAVINGS

Abstractions: File and Directory

- File: A persistent sequence of bytes that stores a logically coherent digital object for an application
  - File Format: An application-specific standard that dictates how to interpret and process a file's bytes
  - 100s of file formats exist (e.g., TXT, DOC, GIF, MPEG); varying data models/types, domain-specific, etc.
  - Metadata: Summary or organizing info. about file content (aka *payload*) stored with file itself; format-dependent
- Directory: A cataloging structure with a list of references to files and/or (recursively) other directories
  - Typically treated as a special kind of file
  - Sub dir., Parent dir., Root dir.

Filesystem

- Filesystem: The part of OS that helps programs create, manage, and delete files on disk (sec. storage)
- Roughly split into *logical level* and *physical level*
  - Logical level exposes file and dir. abstractions and offers System Call APIs for file handling
  - Physical level works with disk firmware and moves bytes to/from disk to DRAM

Filesystem

- Dozens of filesystems exist, e.g., ext2, ext3, NTFS, etc.
  - Differ on how they layer file and dir. abstractions as bytes, what metadata is stored, etc.
  - Differ on how data integrity/reliability is assured, support for editing/resizing, compression/encryption, etc.
  - Some can work with (“mounted” by) multiple OSs

Virtualization of File on Disk

- OS abstracts a file on disk as a virtual object for processes
- File Descriptor: An OS-assigned +ve integer identifier/reference for a file's virtual object that a process can use
  - 0/1/2 reserved for STDIN/STDOUT/STDERR
  - File Handle: A PL's abstraction on top of a file descr. (fd)