Parallel Systems
Welcome to cs380p

Chris Rossbach + Calvin Lin

CS380p

Outline for Today

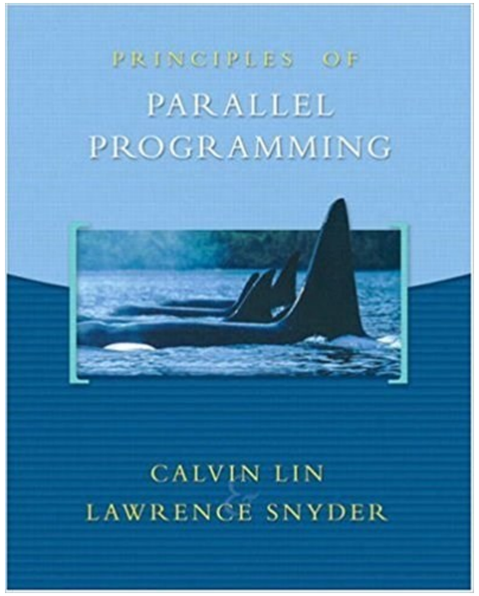
- Course Overview
- Course Details and Logistics
- Concurrency & Parallelism Basics
    - Motivation
    - Problem Decomposition

Acknowledgments: some materials in this lecture borrowed from or built on materials from:

- *Emmett Witchel, who borrowed them from: Kathryn McKinley, Ron Rockhold, Tom Anderson, John Carter, Mike Dahlin, Jim Kurose, Hank Levy, Harrick Vin, Thomas Narten, and Emery Berger*
- *Mark Silberstein, who borrowed them from: Blaise Barney, Kunle Olukoton, Gupta*

Course Details

Course Name:	**CS380P – Parallel Systems**
Lectures:	Online
Class Web Page:	http://www.cs.utexas.edu/users/rossbach/cs380p
Instructors:	Chris Rossbach + Calvin Lin
Text:	Principles of Parallel Programming (ISBN-10: 0321487907)



Please read the syllabus!

Why you should take this course

Why you should take this course

- Parallelism is super-cool and super-important

Why you should take this course

- Parallelism is super-cool and super-important
- You'll learn important concepts and background

Why you should take this course

- Parallelism is super-cool and super-important
- You'll learn important concepts and background
- Have *fun* programming cool systems
  - GPUs! Multi-core!
  - Modern infrastructure and programming languages
  - Interesting synchronization primitives (not just about locks!)

Why you should take this course

- Parallelism is super-cool and super-important
- You'll learn important concepts and background
- Have *fun* programming cool systems
  - GPUs! Multi-core!
  - Modern infrastructure and programming languages
  - Interesting synchronization primitives (not just about locks!)

Two perspectives:
- The "just eat your kale and quinoa" argument
- The "it's going to be fun" argument

My first computer

My first computer

My first computer

CPU

My first computer

CPU

Storage

My first computer

CPU

Storage

Tape drive!
(also good for playing heavy metal music)

My first computer

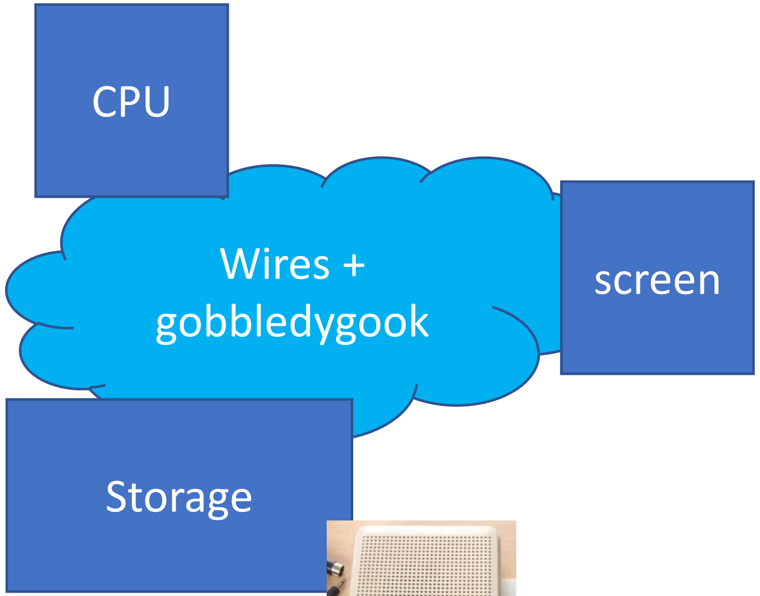
CPU

screen

Storage

Tape drive!
(also good for playing heavy metal music)

My first computer



Tape drive!
(also good for playing heavy metal music)

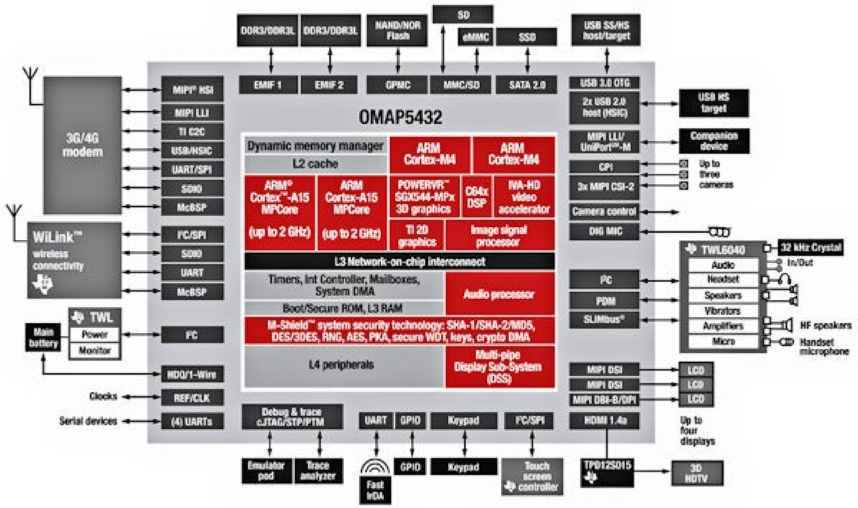
My current computer

My current computer

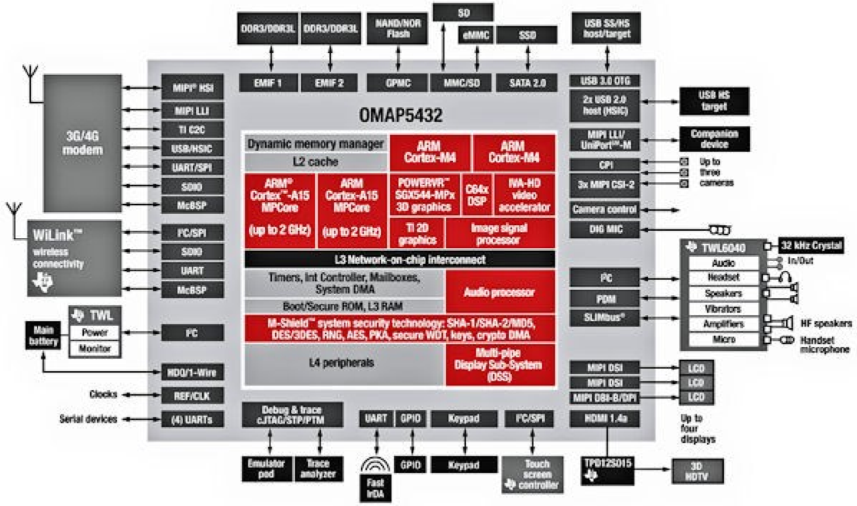
Too boring…

Another of my current computers

Another of my current computers

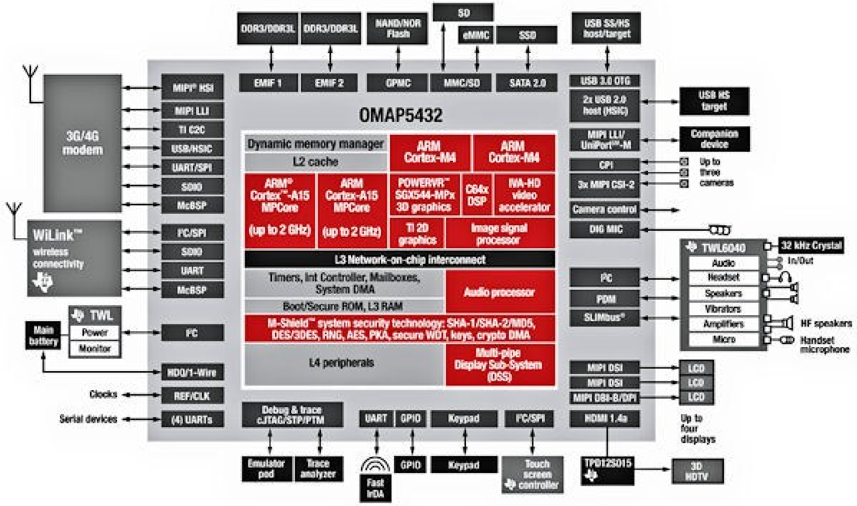


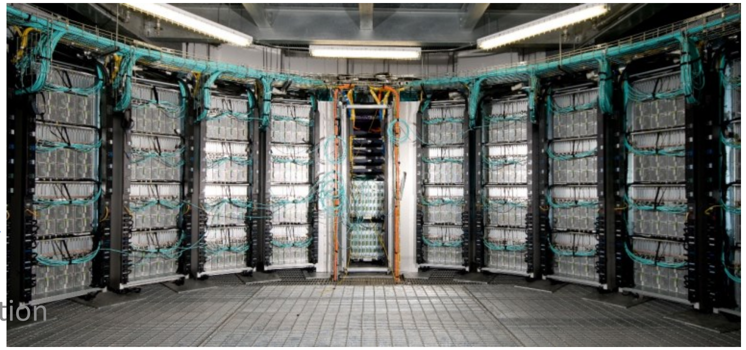
Another of my current computers



- CPU
- CPU
- GPU
- Image DSP
- Crypto
- …

Another of my current computers





CPU

CPU

GPU

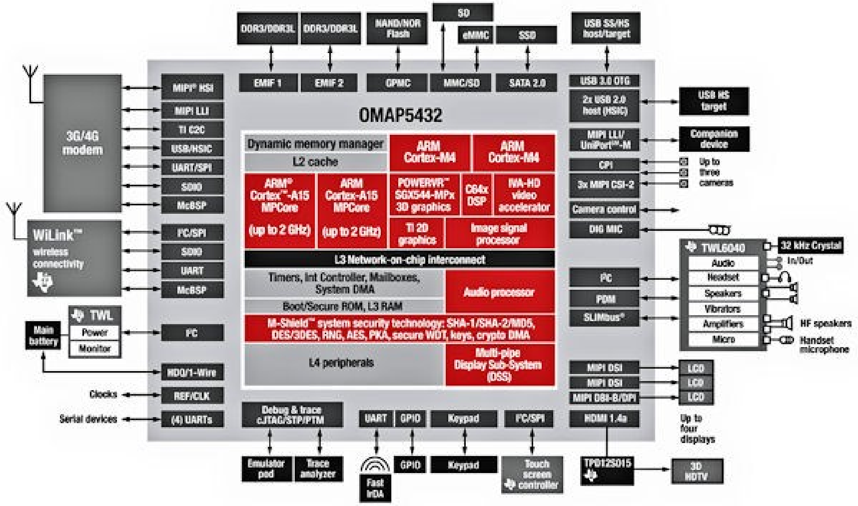
Image DSP

Crypto

…

Another of my current computers

A lot has changed but…
the common theme is…??



CPU

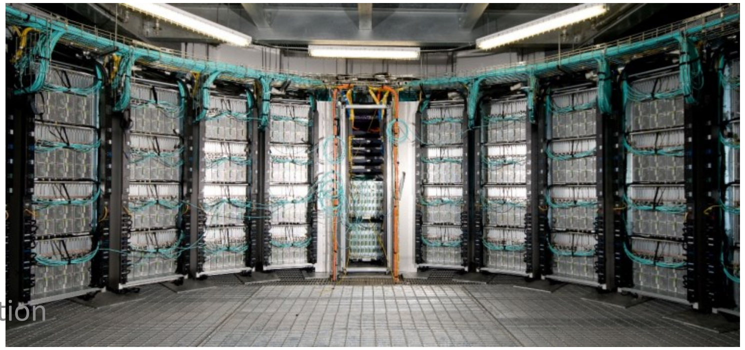
CPU

GPU

Image DSP

Crypto

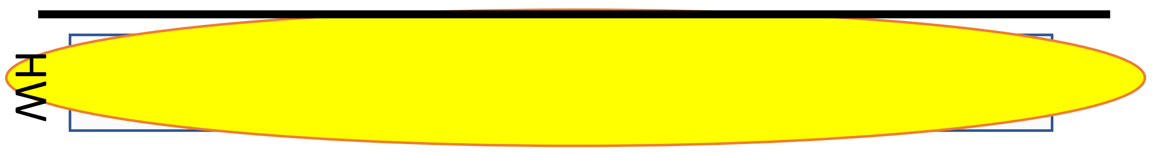
…



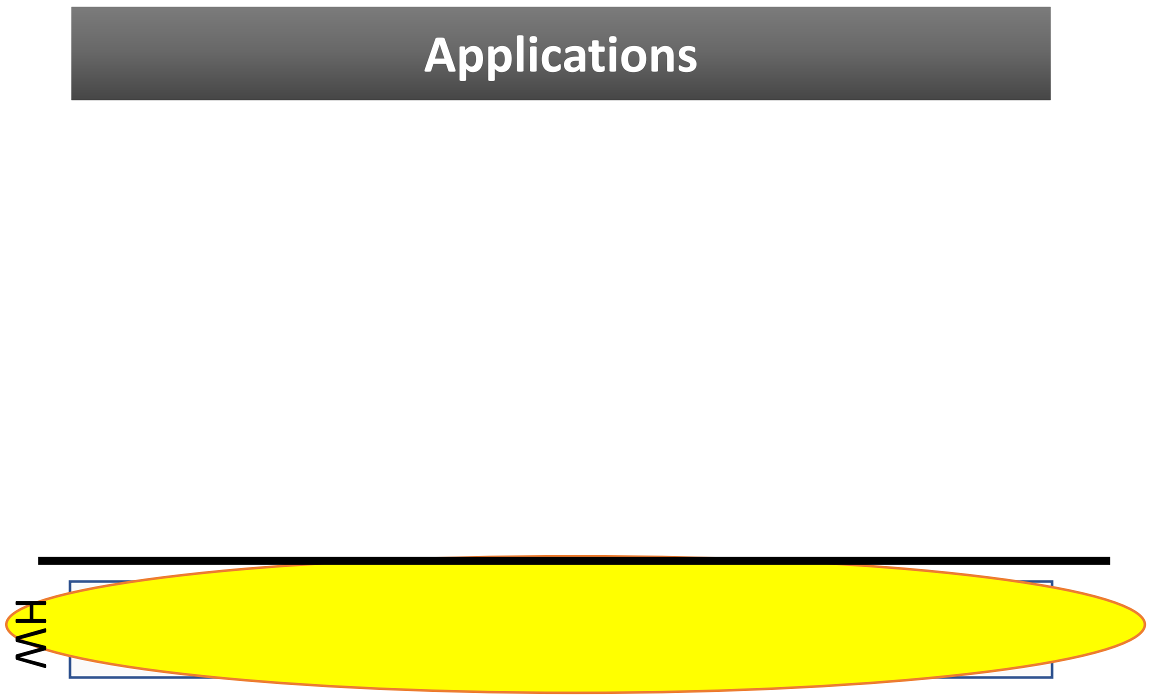
Modern Technology Stack

Modern Technology Stack

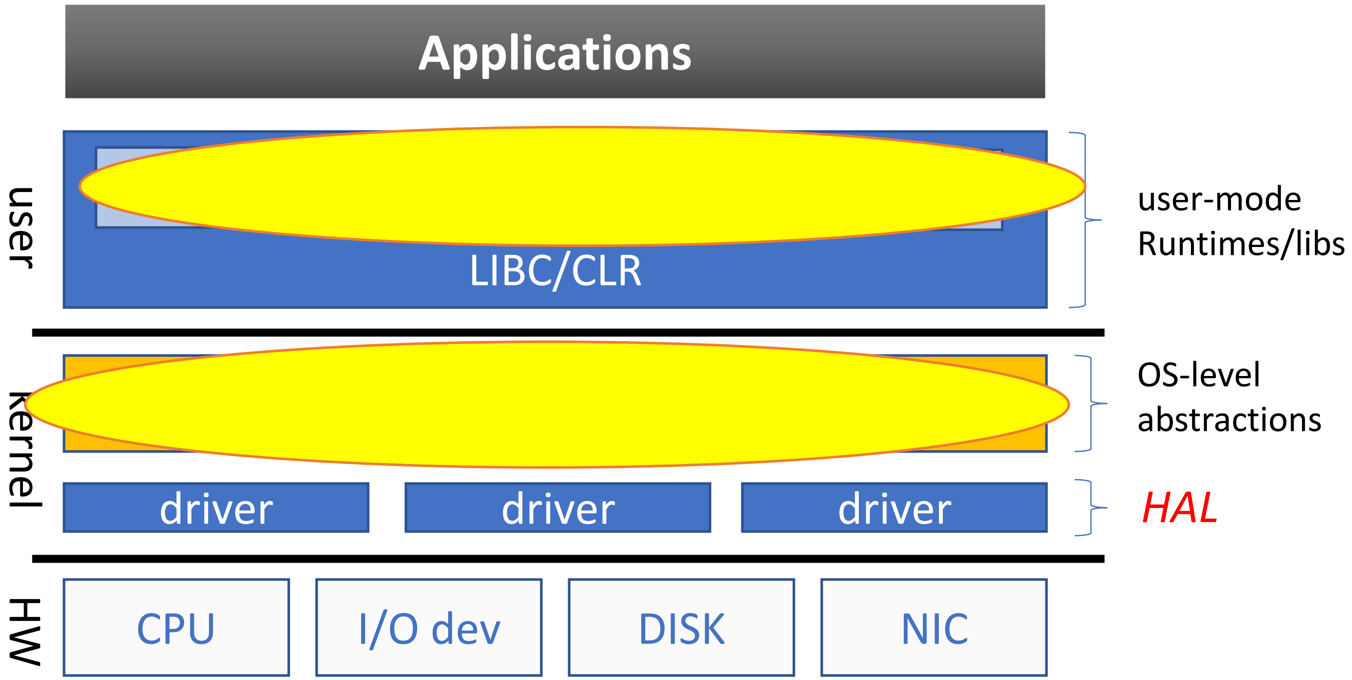
Modern Technology Stack



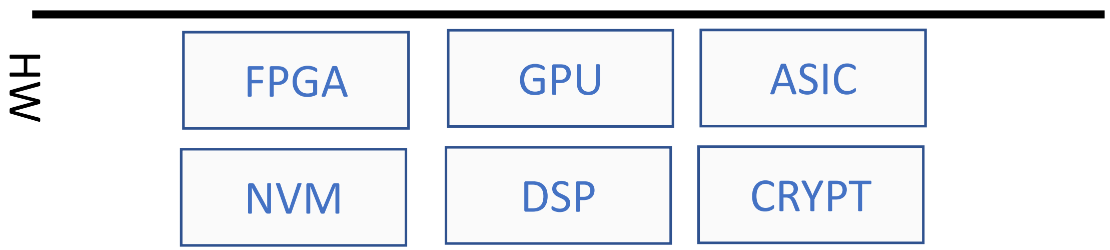
Modern Technology Stack



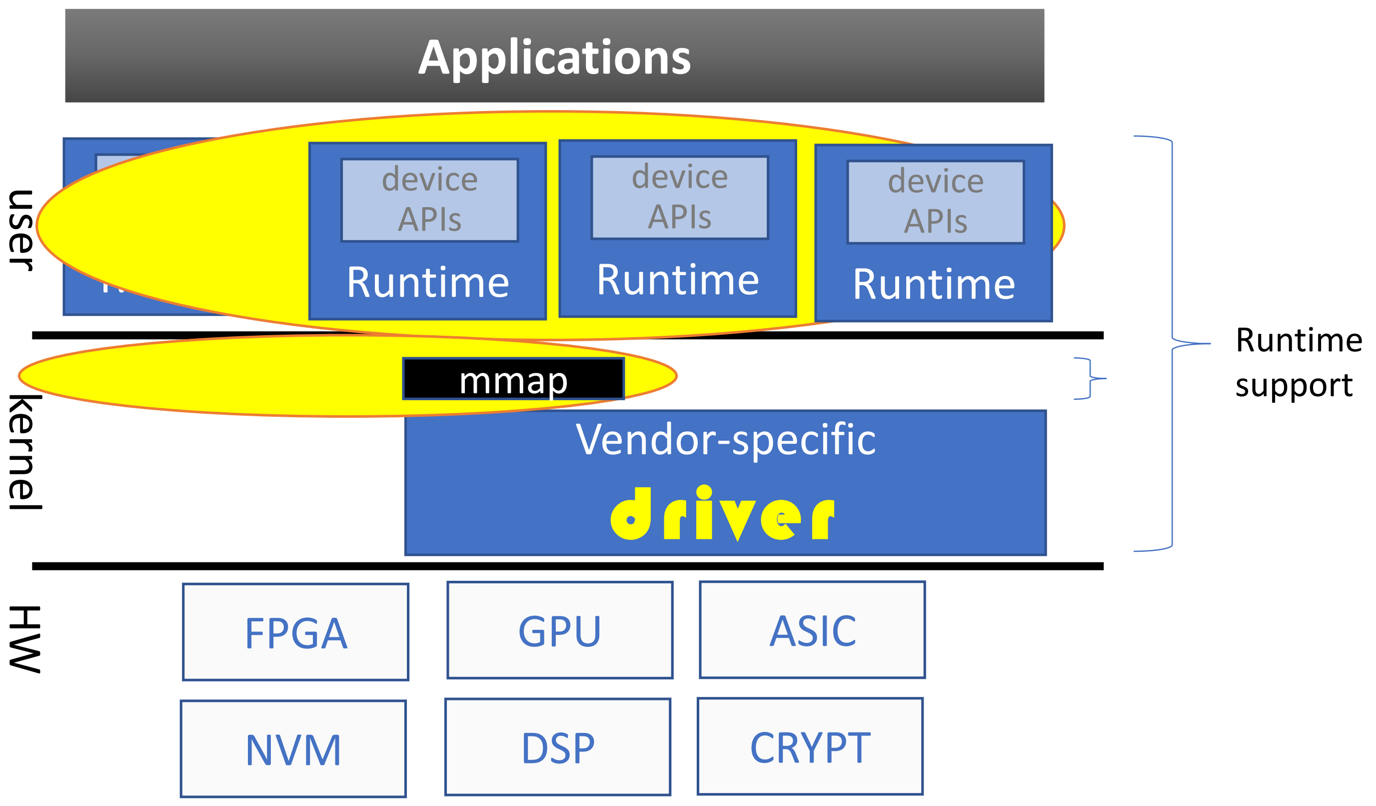
Modern Technology Stack



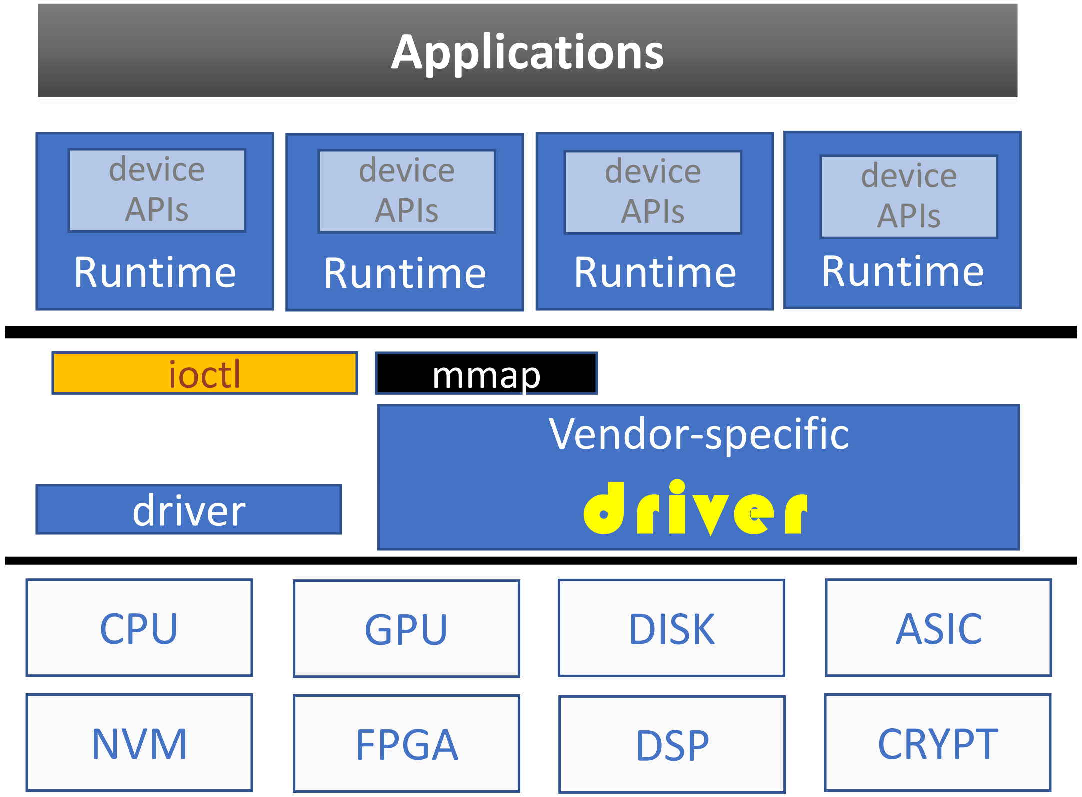
Modern Technology Stack



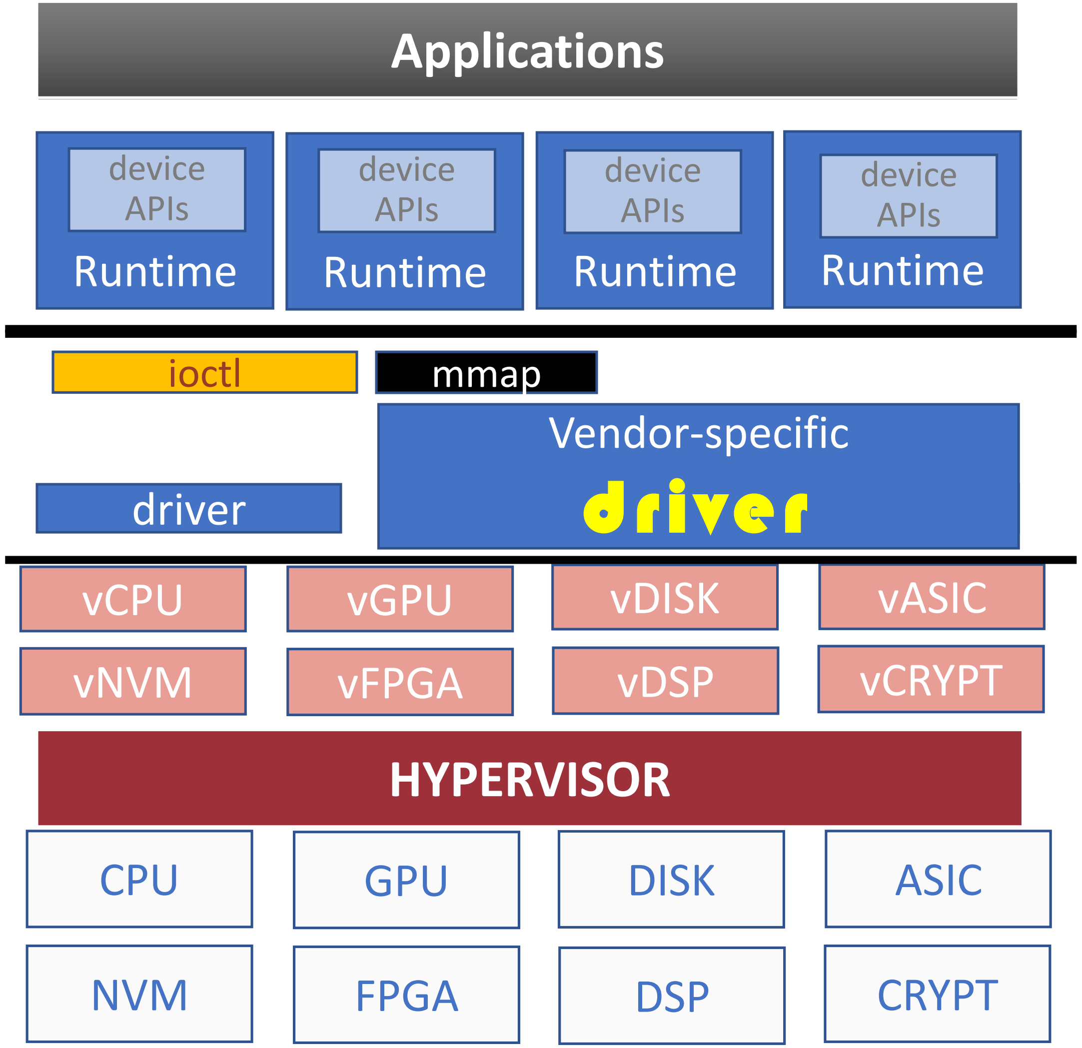
Modern Technology Stack



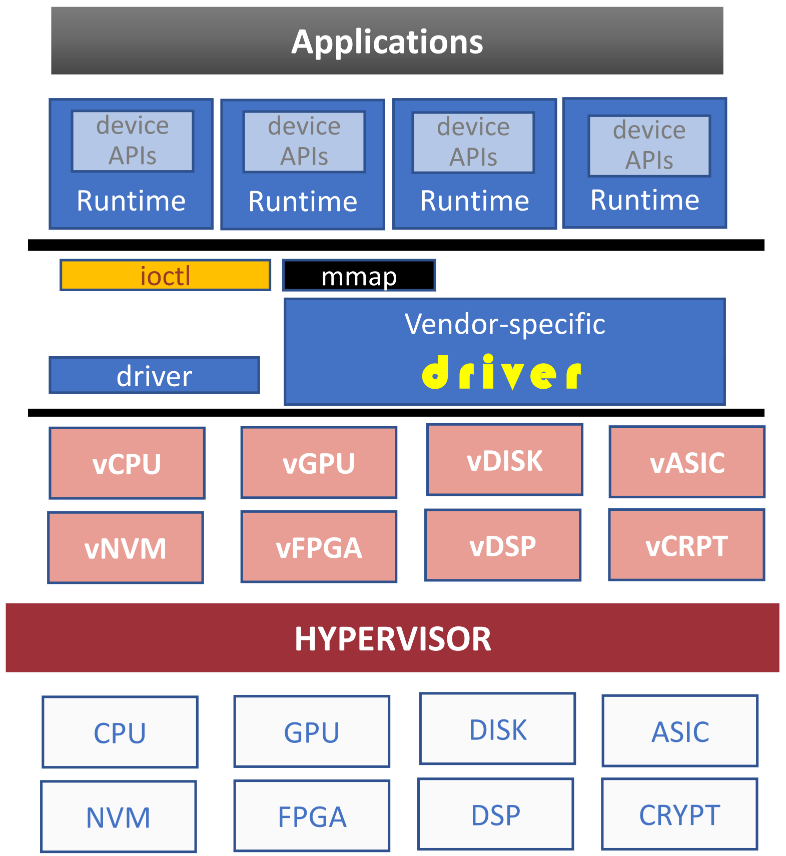
Concurrency and Parallelism are Everywhere



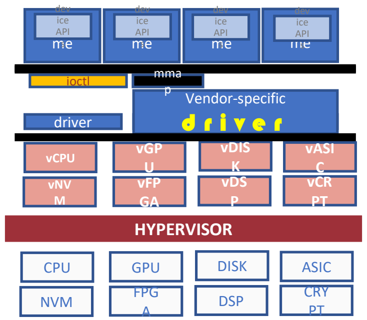
Concurrency and Parallelism are Everywhere



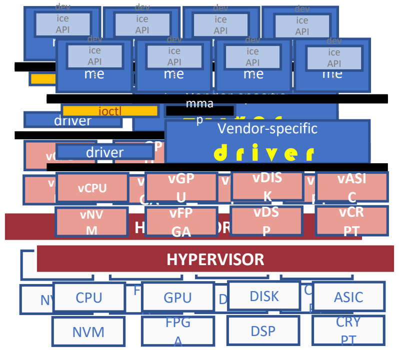
Concurrency and Parallelism are Everywhere



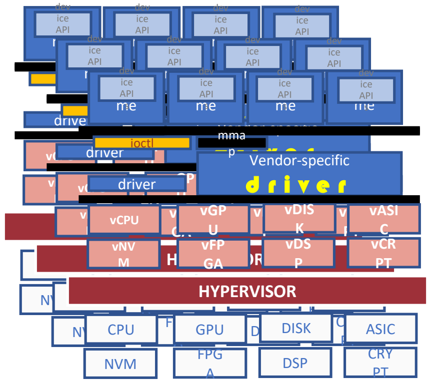
Concurrency and Parallelism are Everywhere



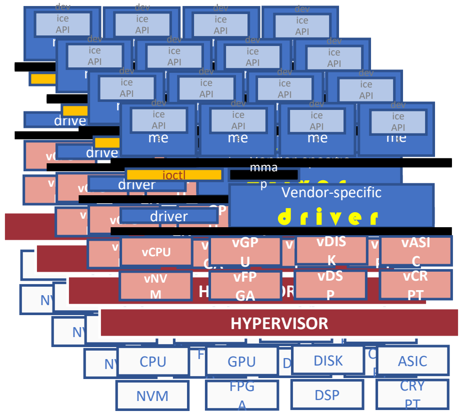
Concurrency and Parallelism are Everywhere



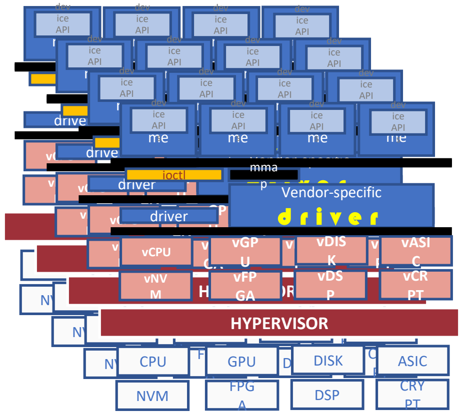
Concurrency and Parallelism are Everywhere

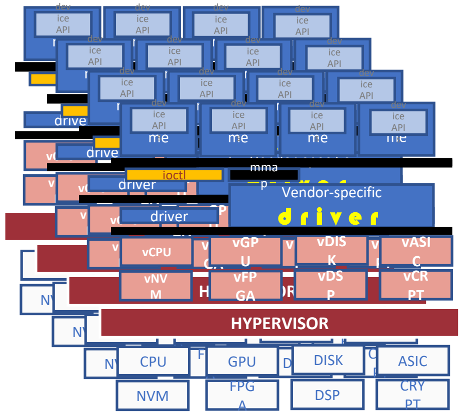


Concurrency and Parallelism are Everywhere

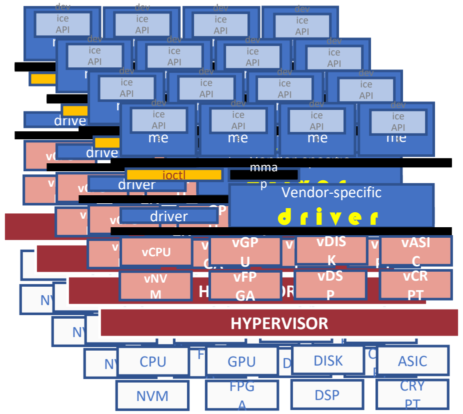


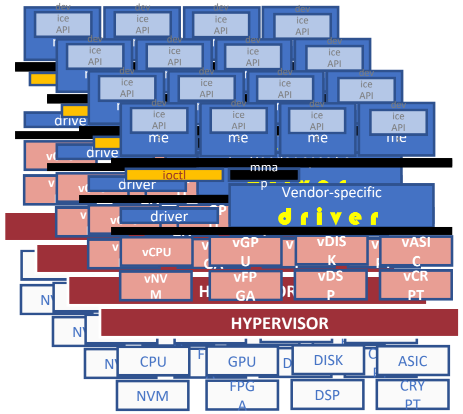
Concurrency and Parallelism are Everywhere

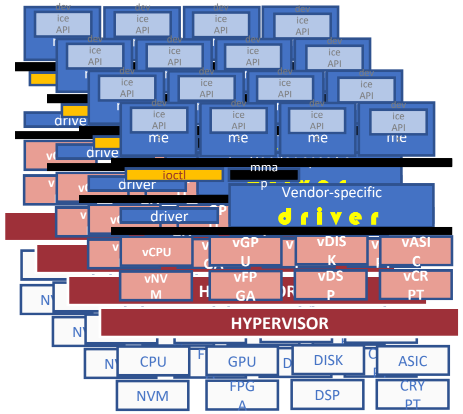




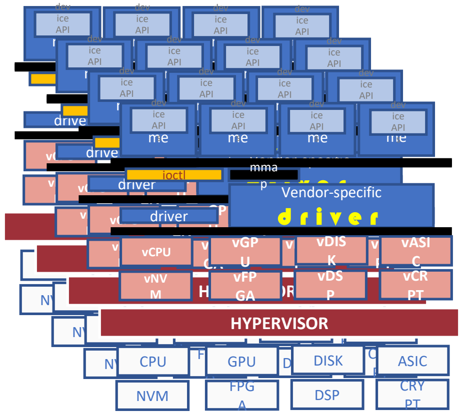
Concurrency and Parallelism are Everywhere

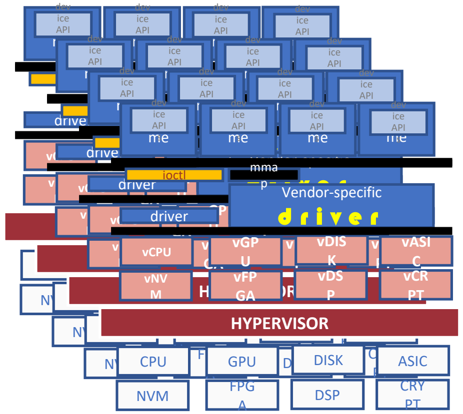


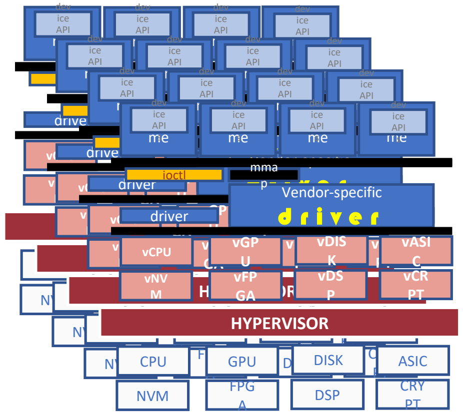


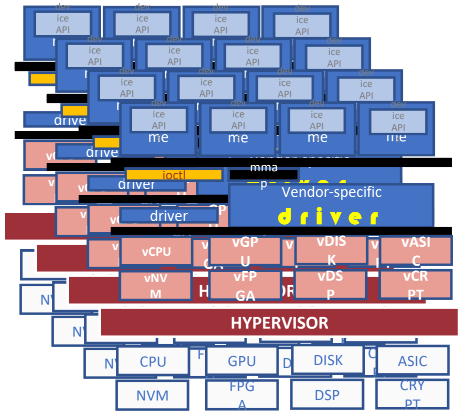


Concurrency and Parallelism are Everywhere

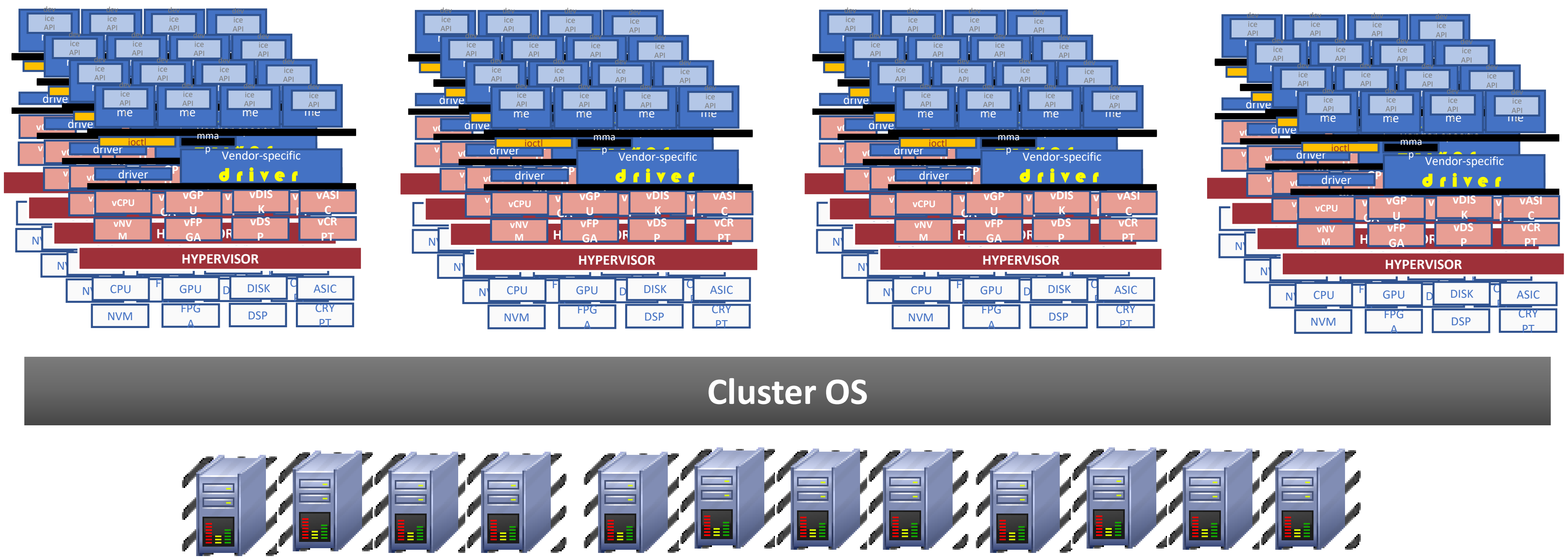




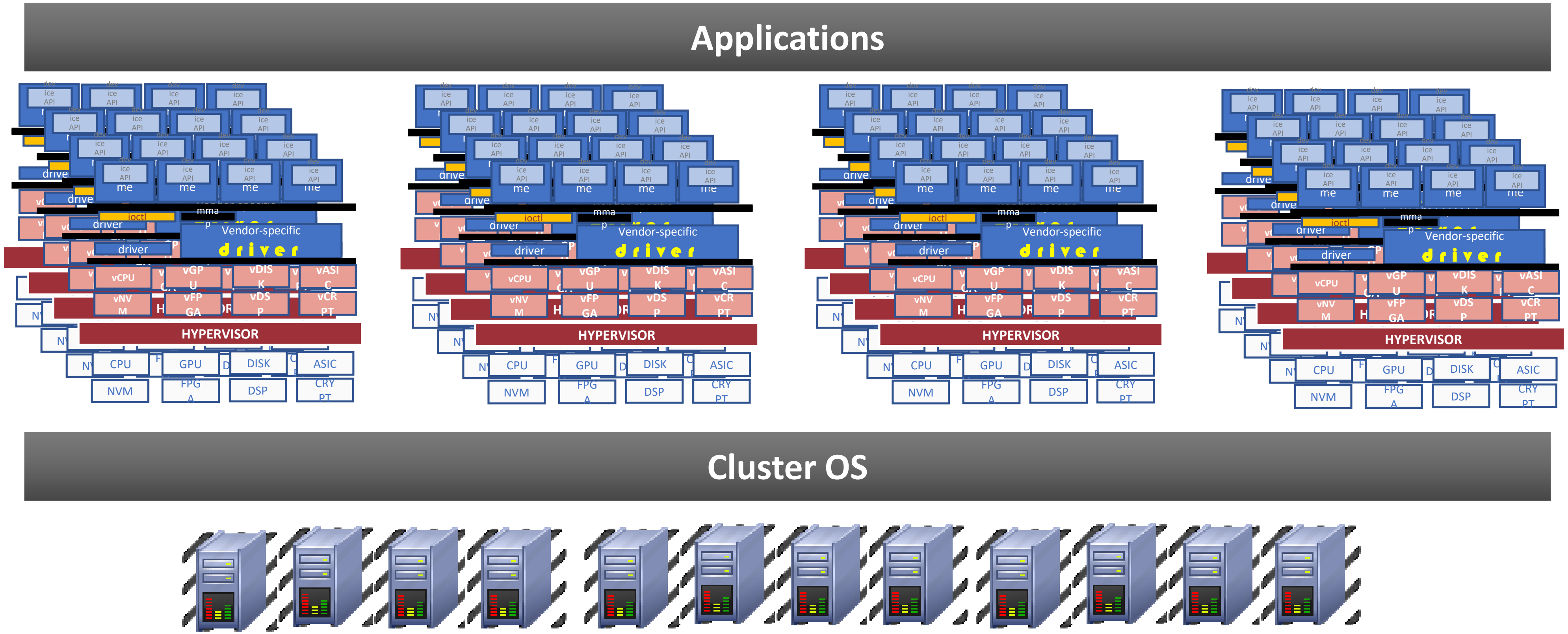




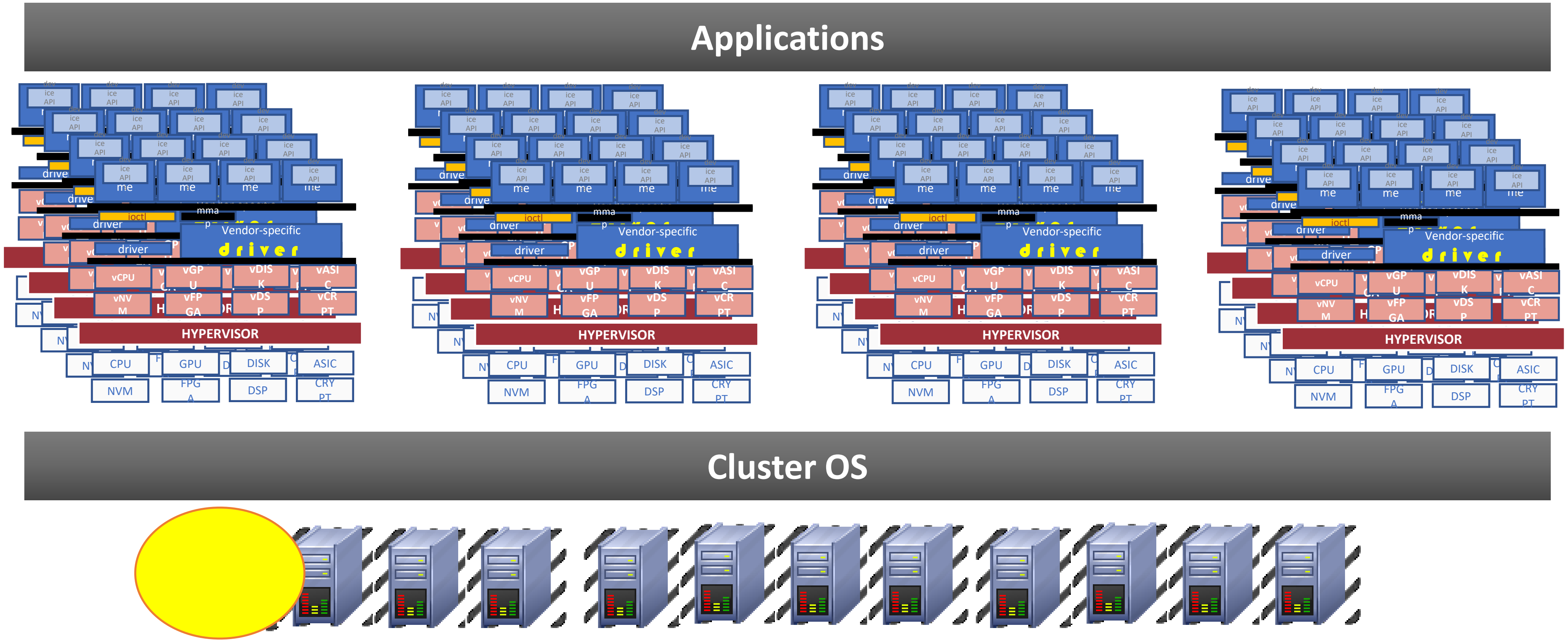
Concurrency and Parallelism are Everywhere



Concurrency and Parallelism are Everywhere

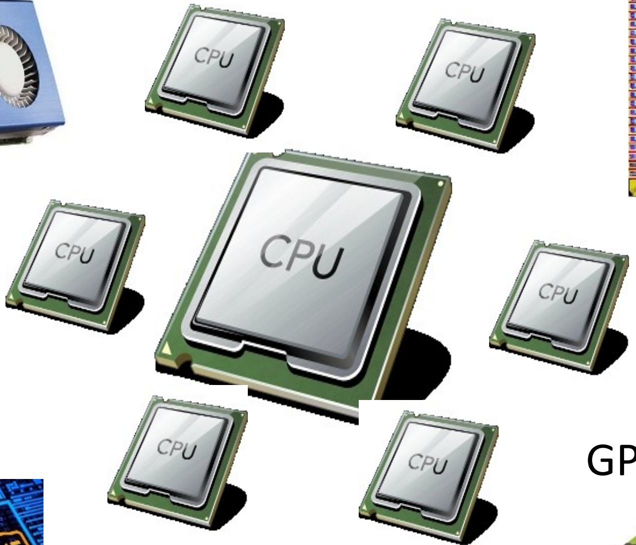


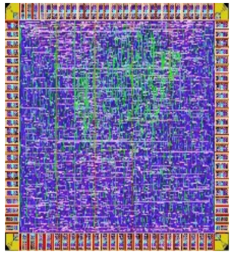
Concurrency and Parallelism are Everywhere



Concurrency and Parallelism are everywhere

DSP

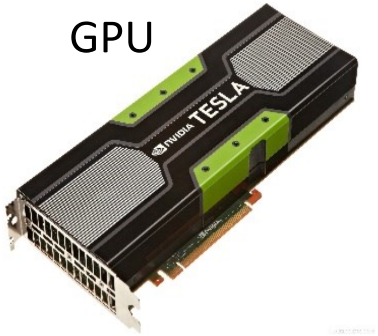


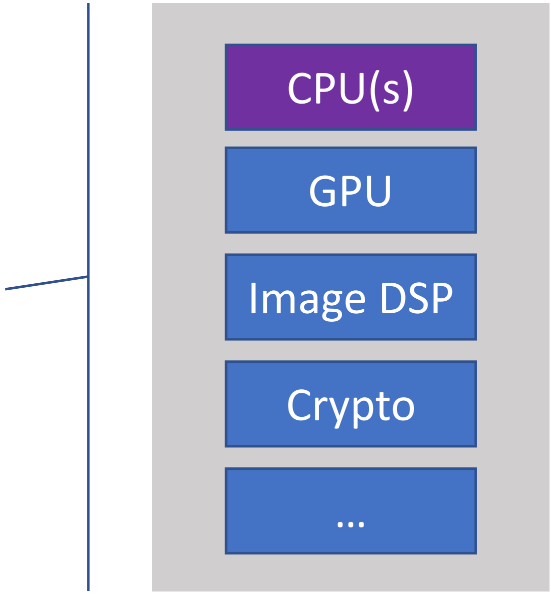


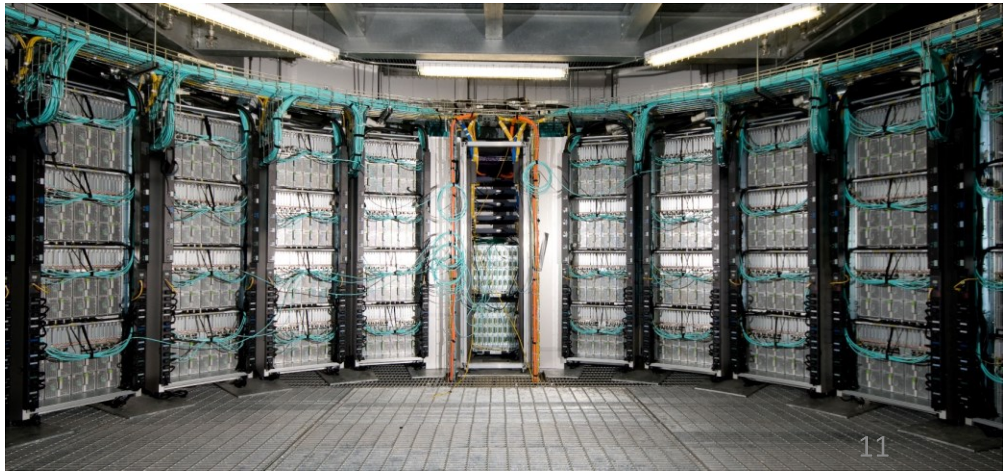
FPGA

Crypto

GPU

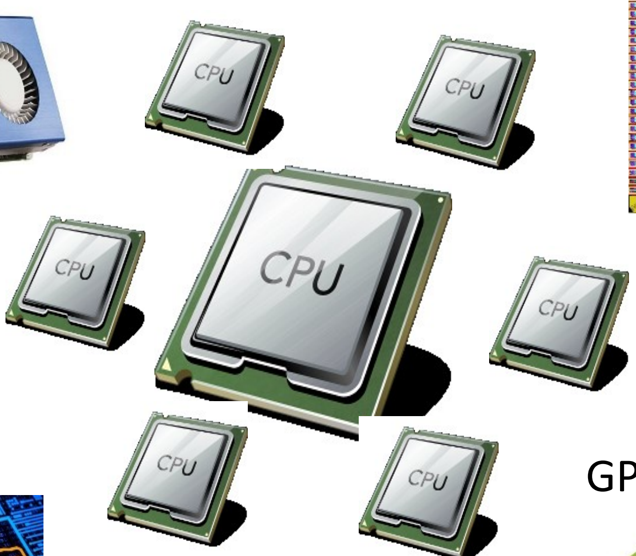


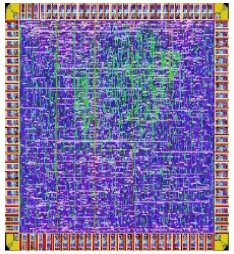




Concurrency and Parallelism are everywhere

DSP

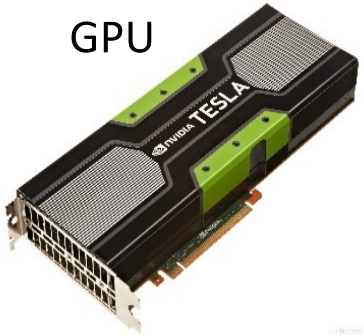




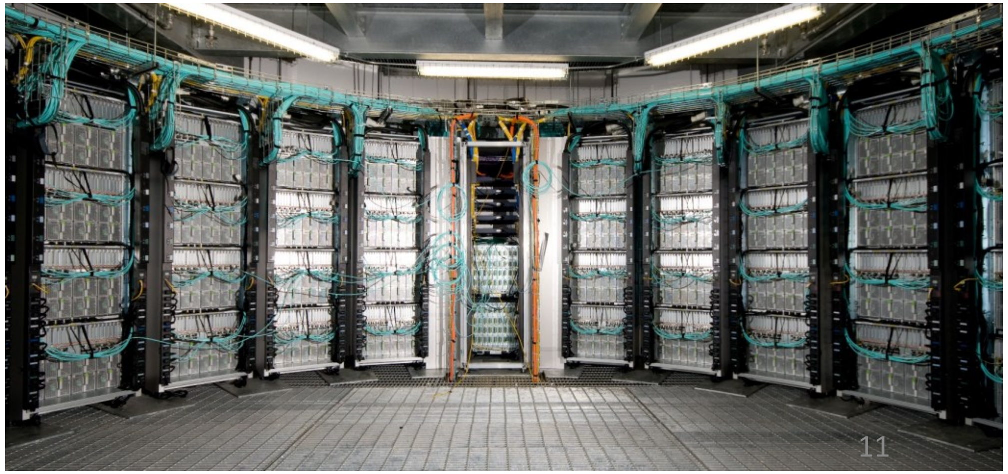
FPGA

Crypto

GPU

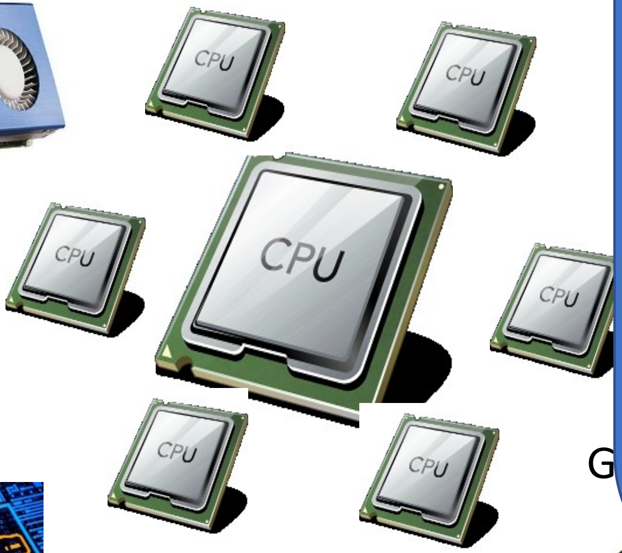


- CPU(s)
- GPU
- Image DSP
- Crypto
- …



Concurrency and Parallelism are everywhere

DSP



Crypto

G

Key concerns:

CPU(s)

GPU

Image DSP

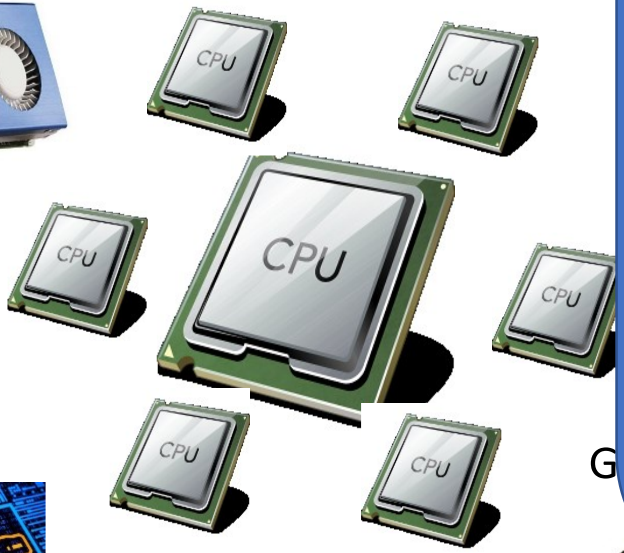
Crypto

...

Concurrency and Parallelism are everywhere

DSP

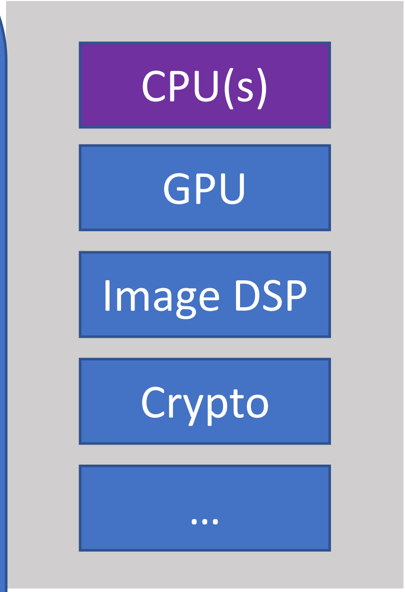
Crypto



Key concerns:

- Concurrency/parallelism can't be avoided anymore (want a job?)
- A program or two playing with locks and threads isn't enough

Course goal is to expose you to lots of ways of programming systems like these

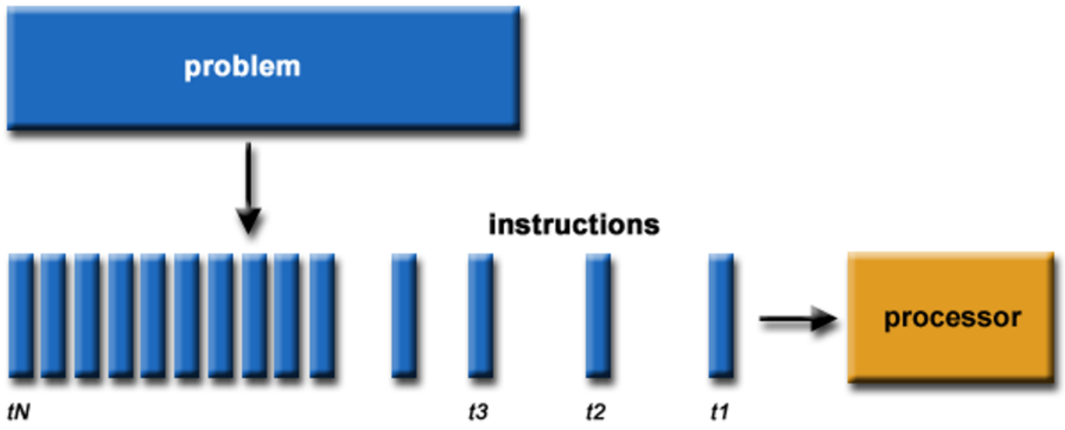


Goal: Make Parallelism Your Close Friend
Method: Use Many Different Approaches

Abstract	Concrete
Locks and Shared Memory Synchronization	Basic Locking
	Prefix sum – pthreads
Language Support	Go lab: condition variables, channels, go routines Rust lab: type-safety, 2PC
Parallel Architectures	GPU Programming Lab
HPC	MPI: Barnes-Hut lab
Modern/Advanced Topics	• Specialized Runtimes / Programming Models • Auto-parallelization • Race Detection

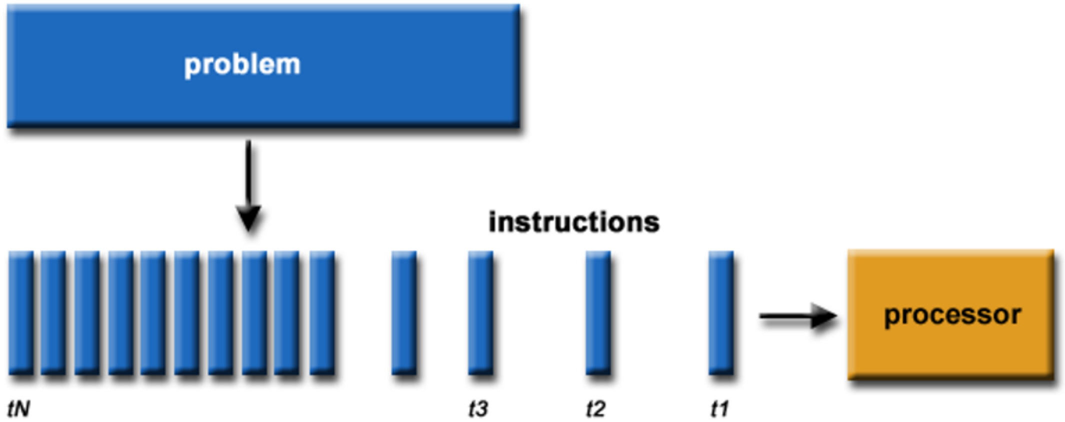
Serial vs. Parallel Program

Serial vs. Parallel Program

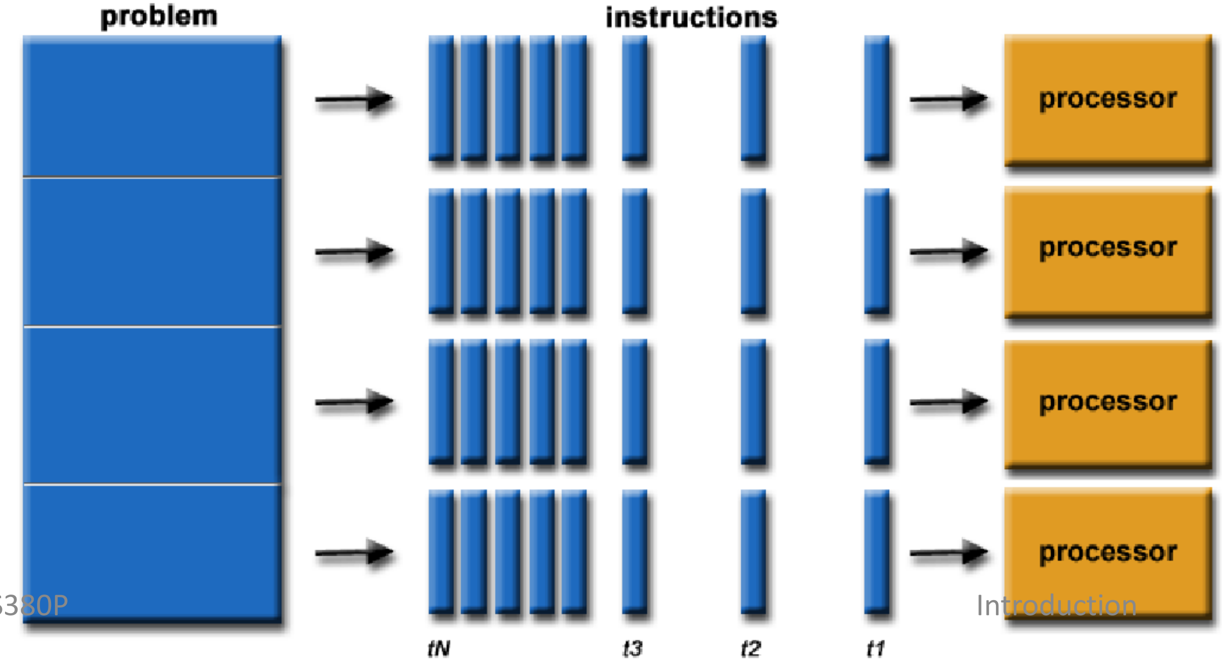


One instruction at a time (apparently)

Serial vs. Parallel Program

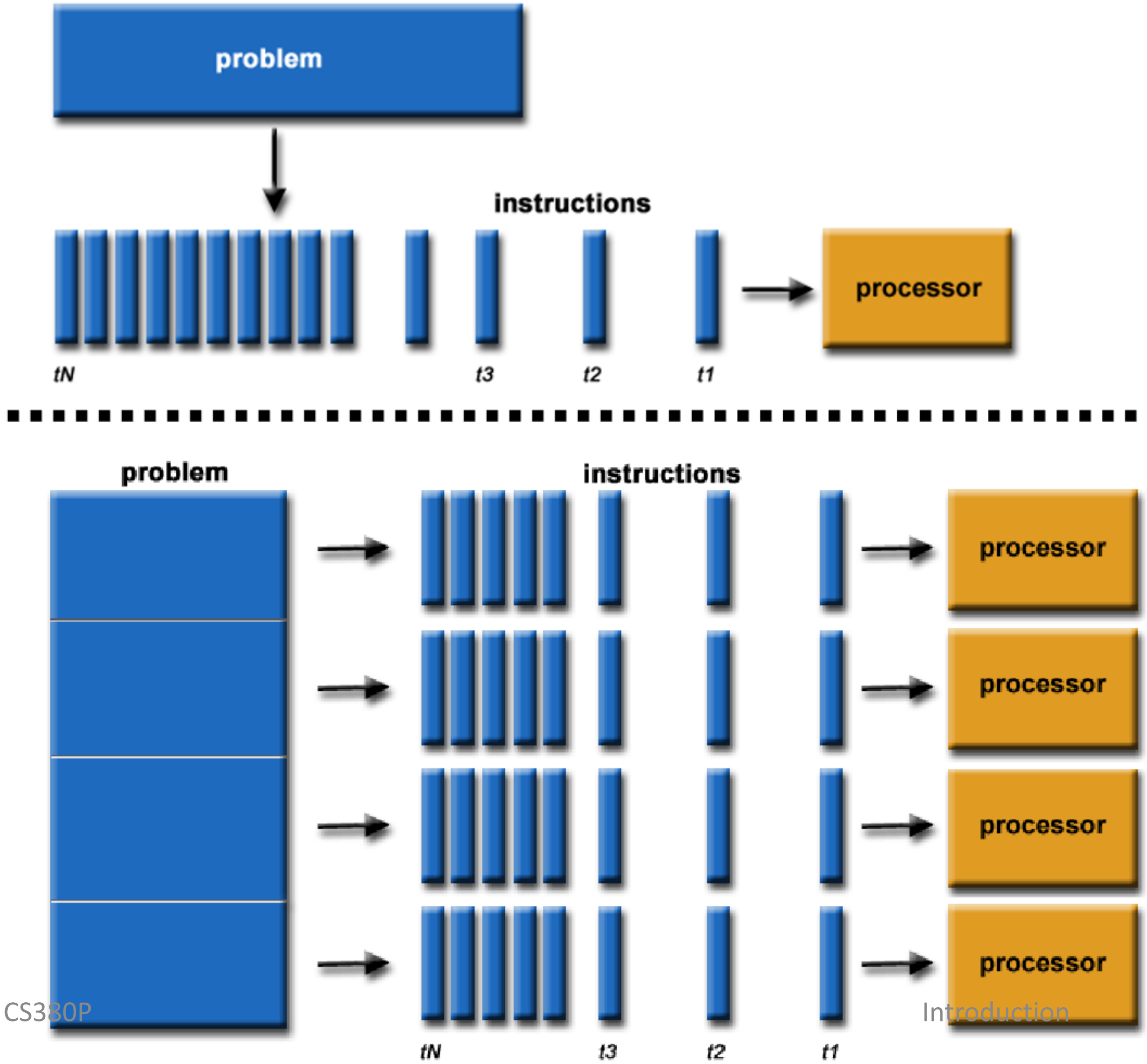


One instruction at a time (apparently)



Multiple instructions in parallel

Serial vs. Parallel Program

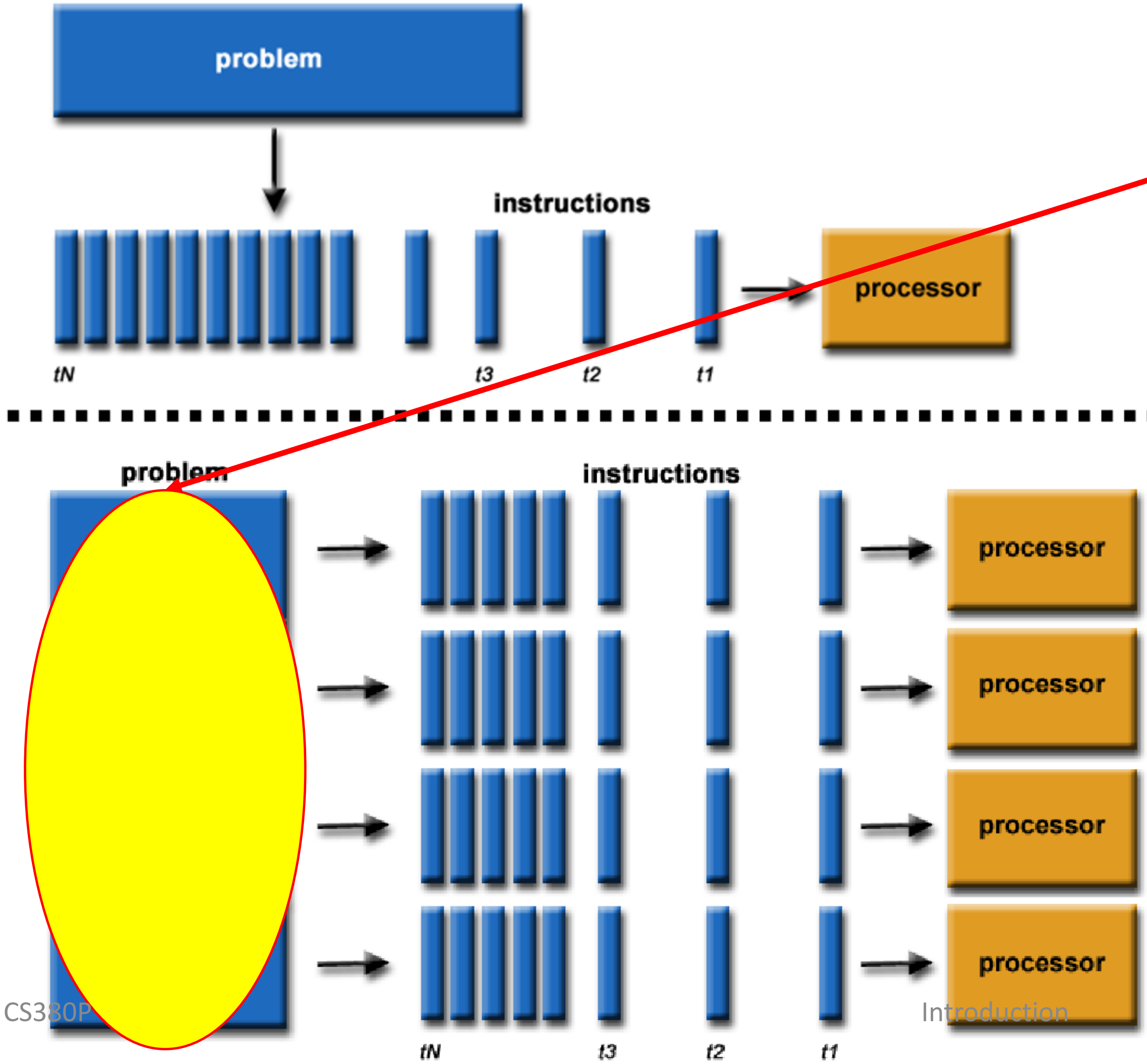


Key concerns:

O

Multiple instructions in parallel

Serial vs. Parallel Program

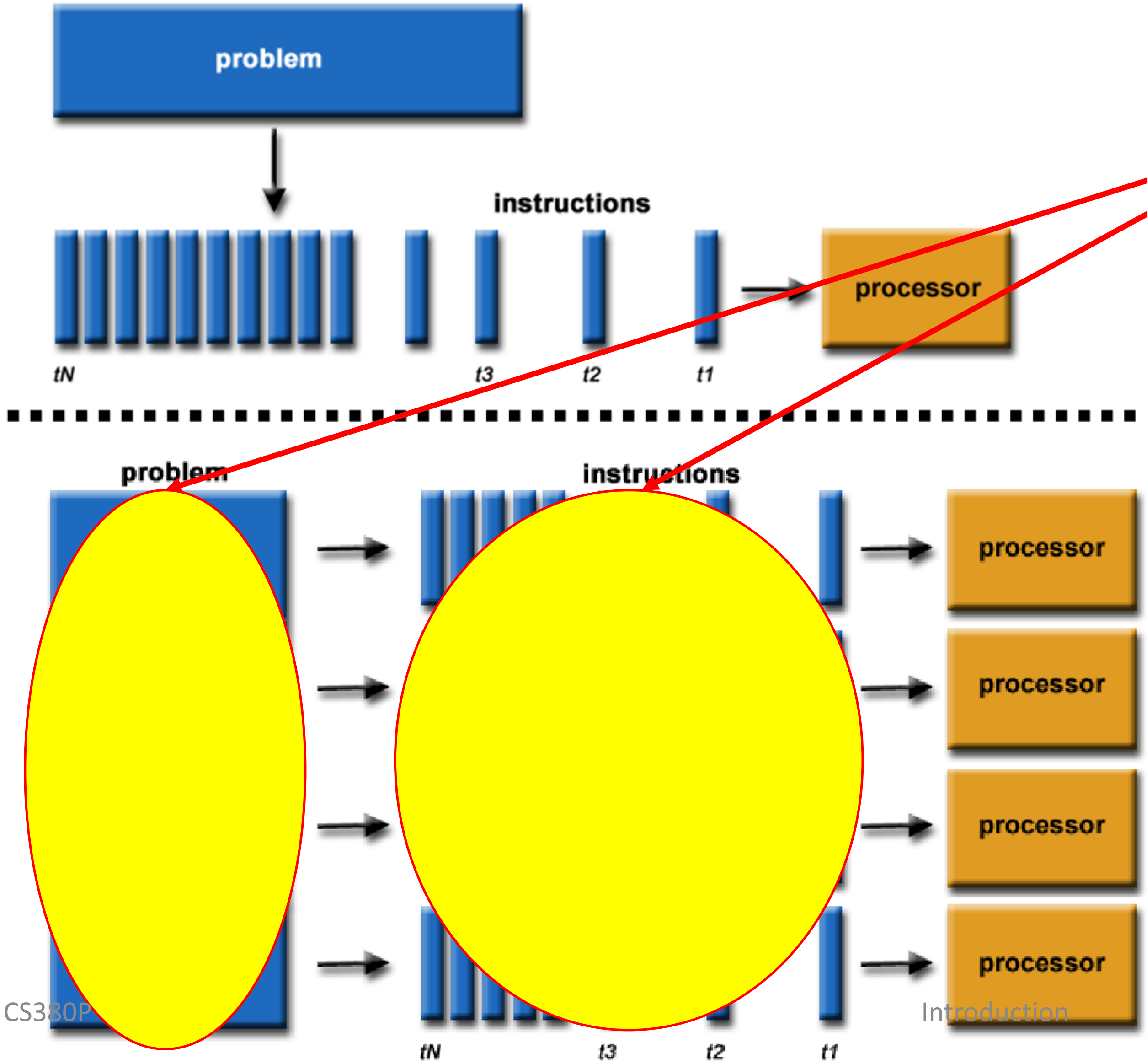


Key concerns:

- Programming model

Multiple instructions in parallel

Serial vs. Parallel Program

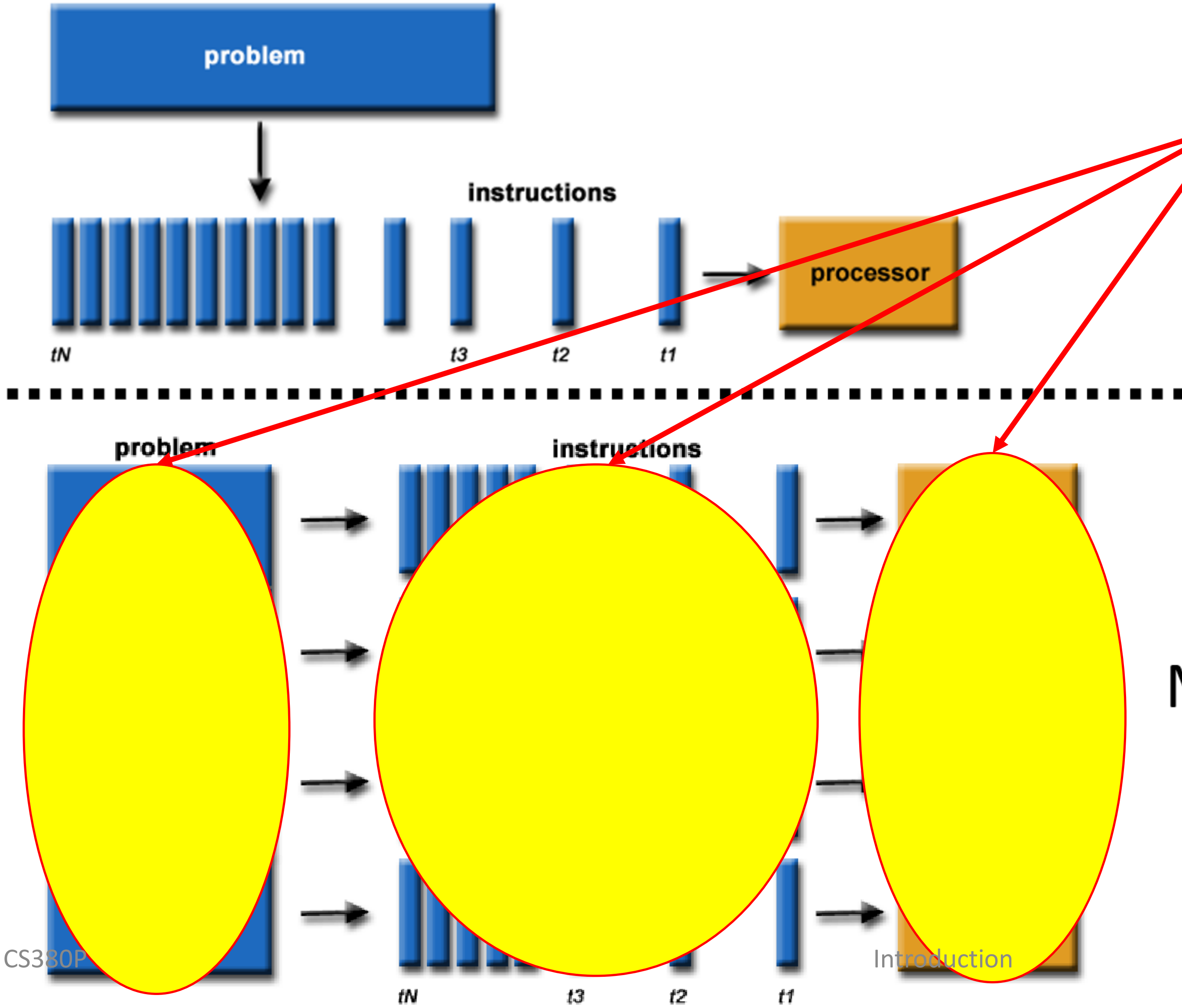


Key concerns:

- Programming model
- Execution Model

Multiple instructions in parallel

Serial vs. Parallel Program

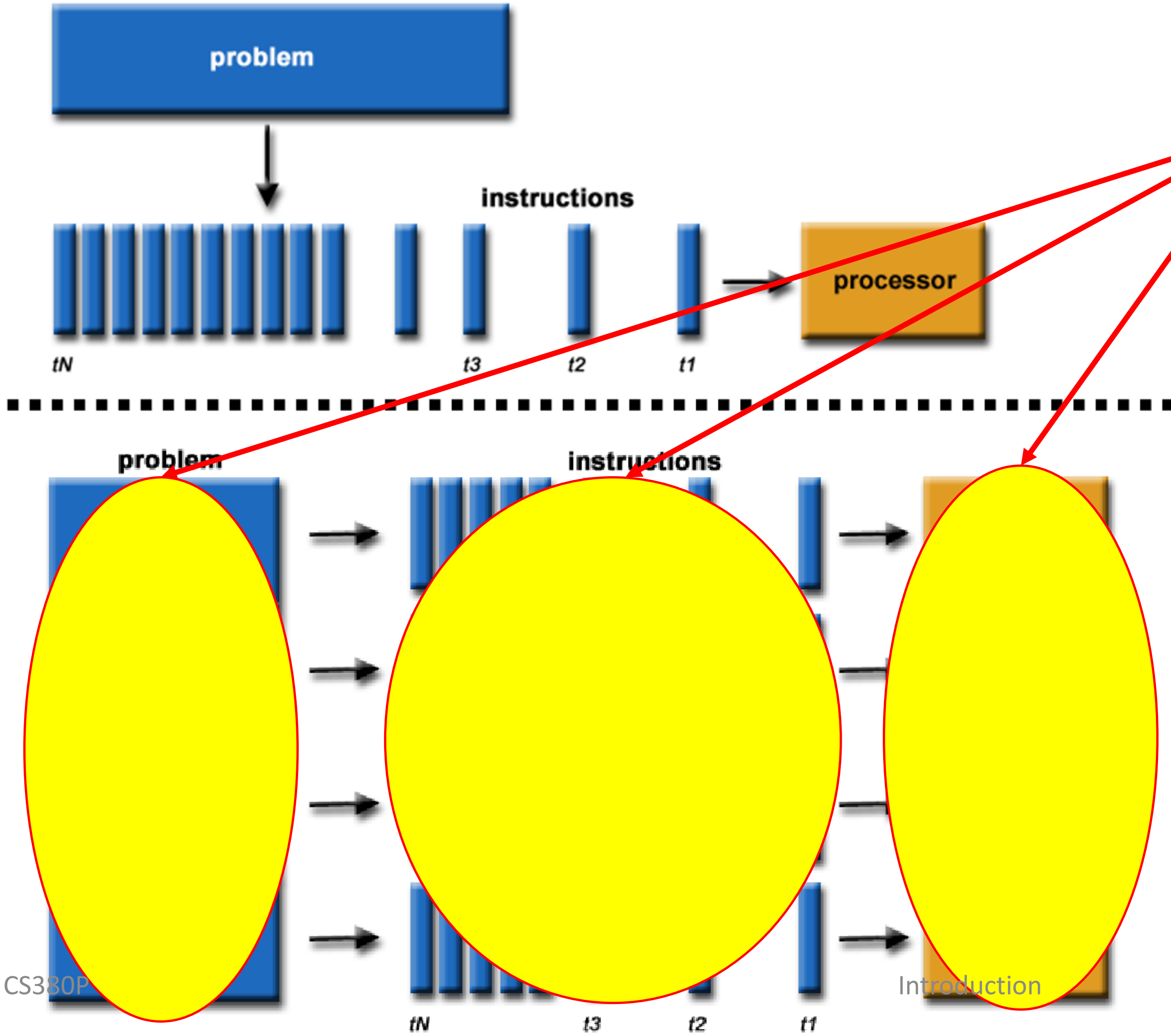


Key concerns:

- Programming model
- Execution Model
- Performance/Efficiency

Multiple instructions in parallel

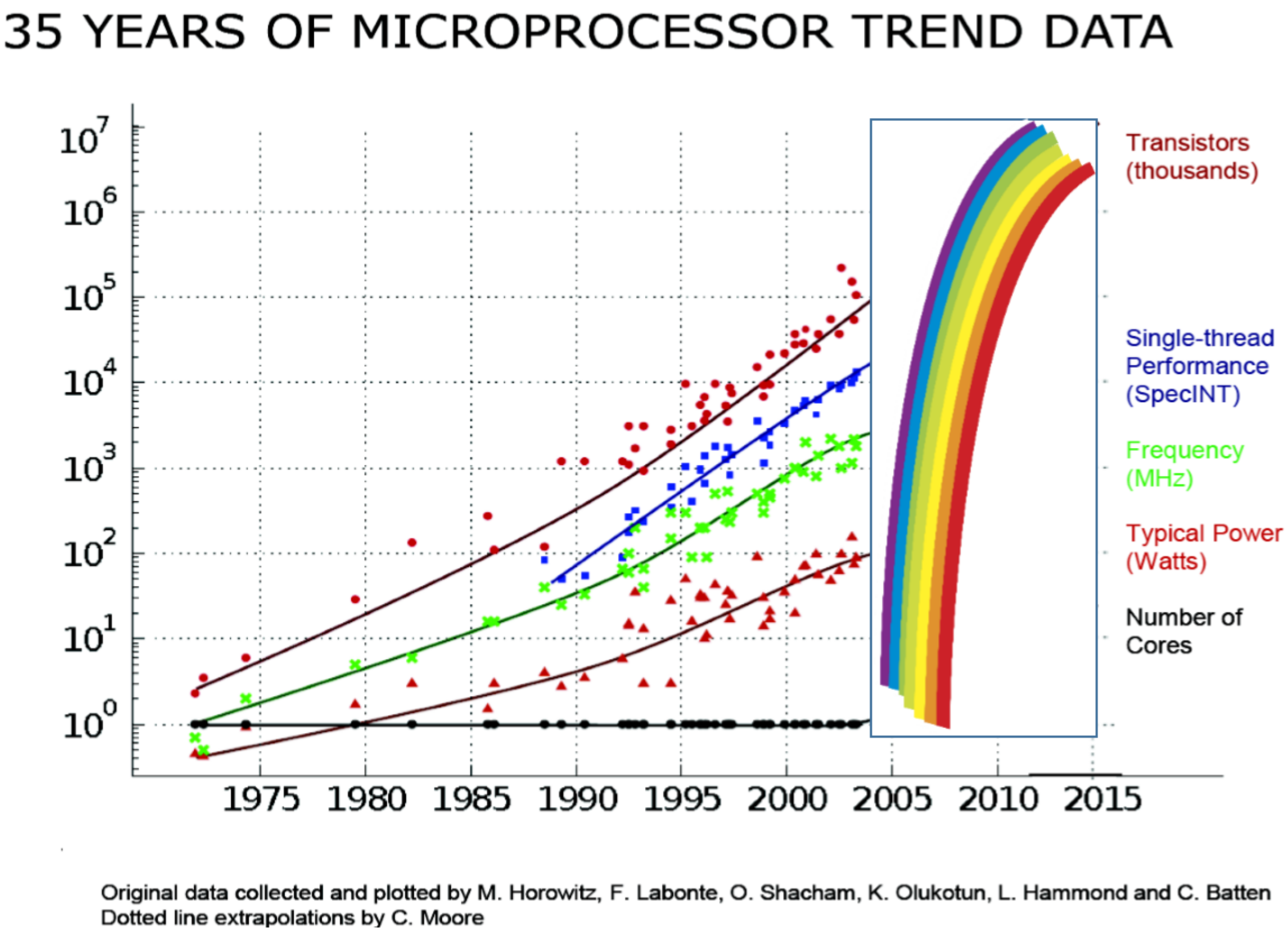
Serial vs. Parallel Program



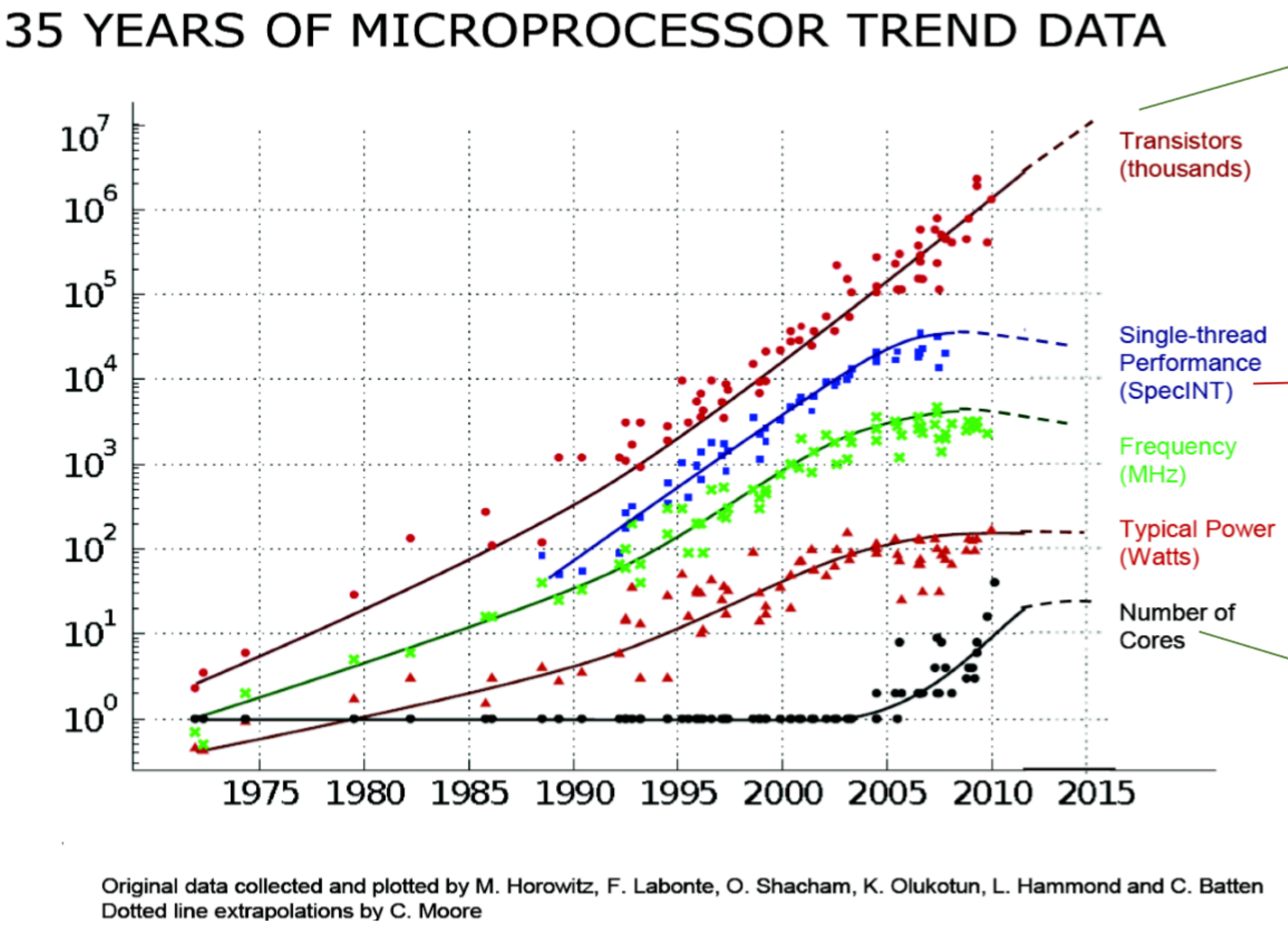
Key concerns:
- Programming model
- Execution Model
- Performance/Efficiency
- Exposing parallelism

Multiple instructions in parallel

Technology Trends



Free lunch – is over ☹



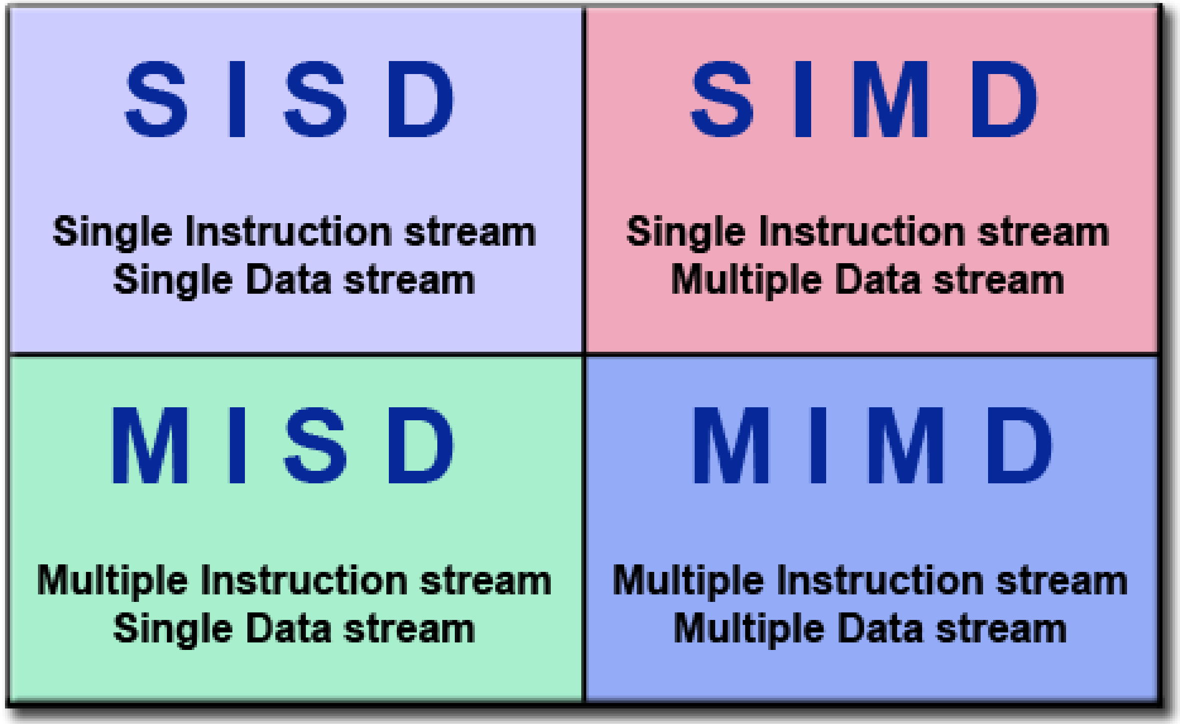
Transistor number grows (Moore's law)

Sequential performance no longer improves

Cores number grows

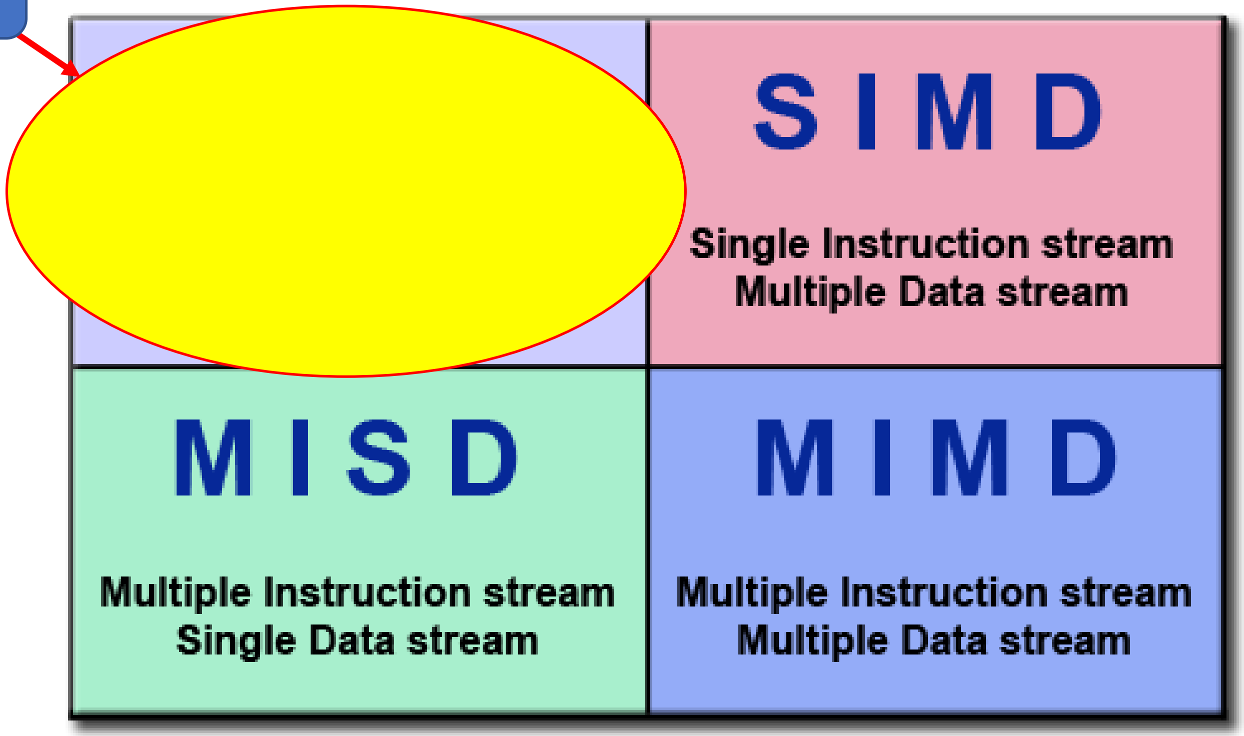
Execution Models: Flynn’s Taxonomy

Execution Models: Flynn's Taxonomy

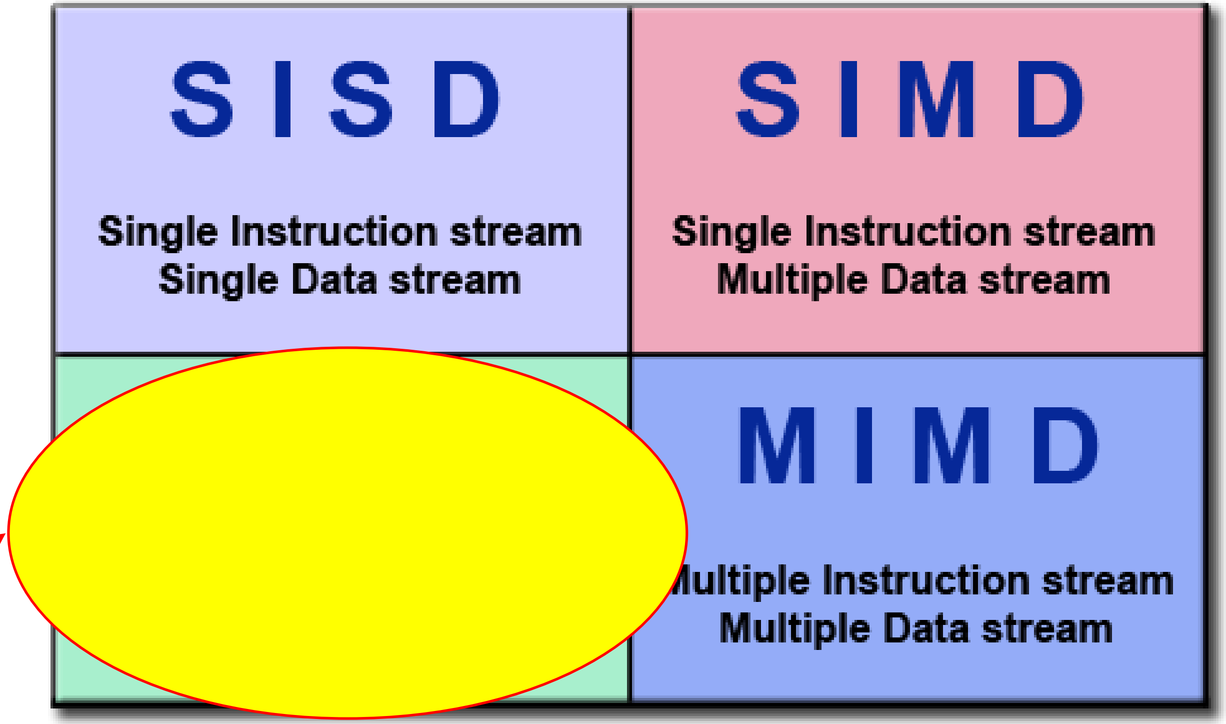


Execution Models: Flynn's Taxonomy

Normal Serial program



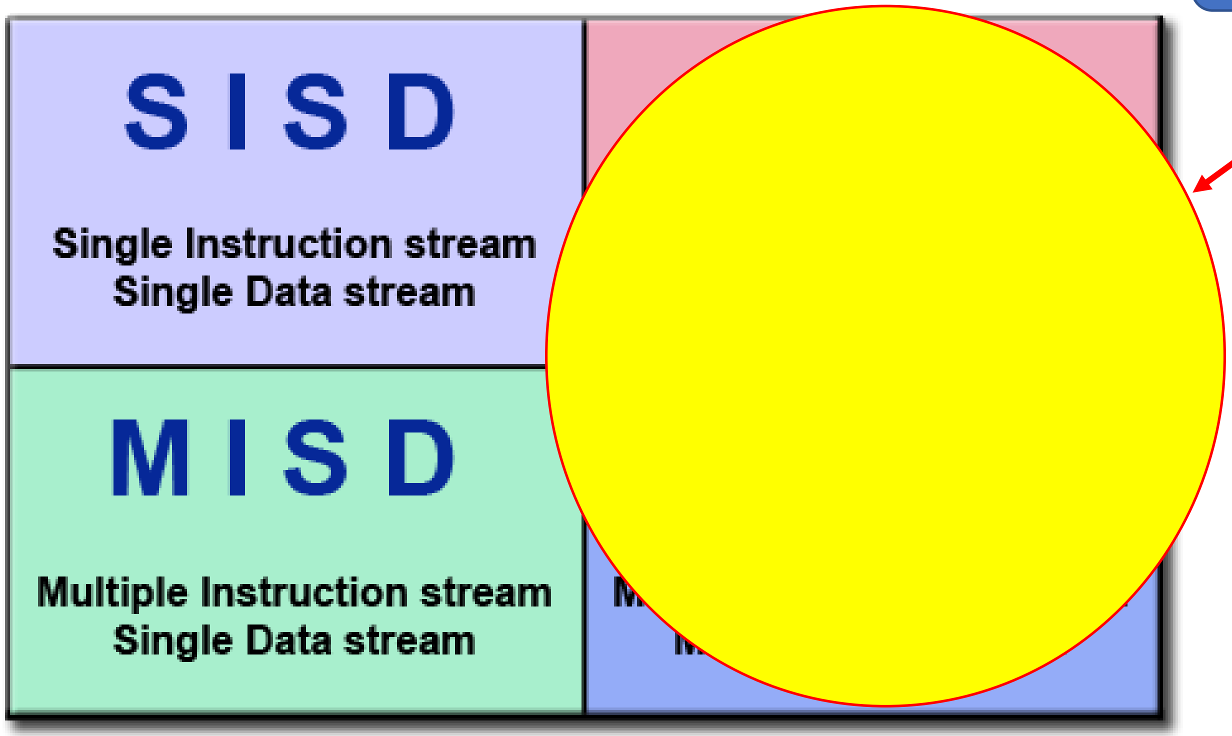
Execution Models: Flynn's Taxonomy



Uncommon architecture:
Fault – tolerance
Pipeline parallelism

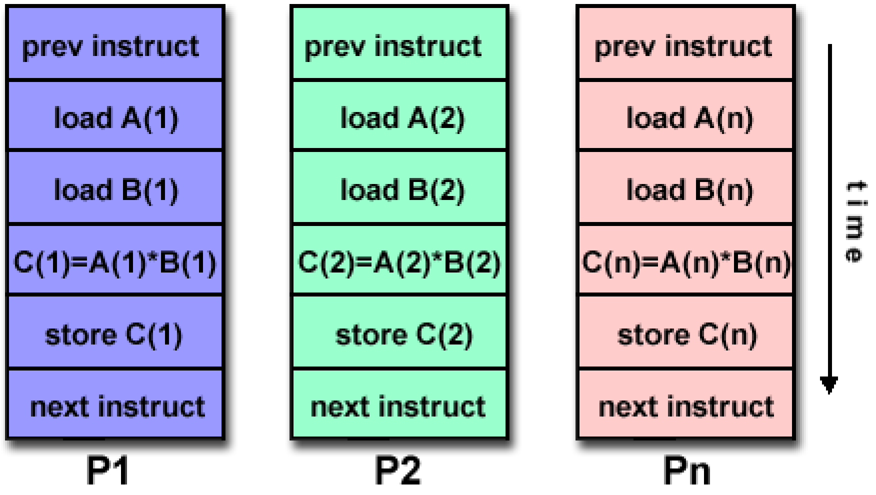
Execution Models: Flynn's Taxonomy

Our main focus



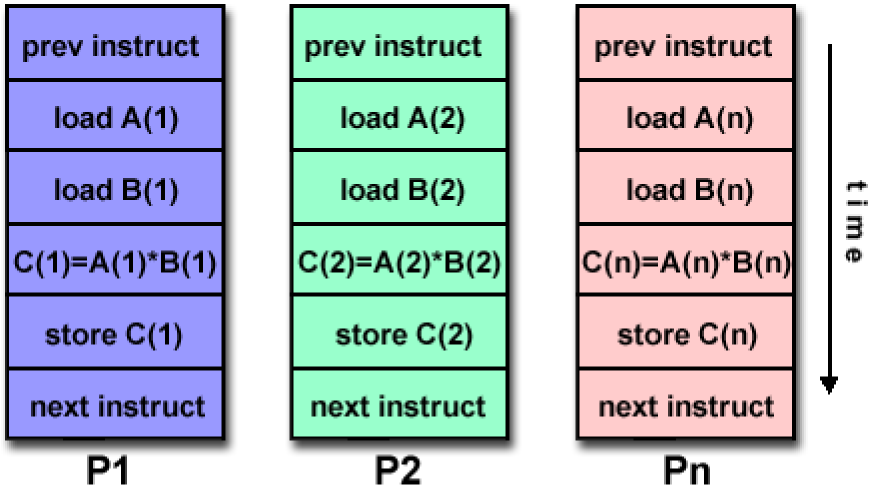
SIMD

SIMD



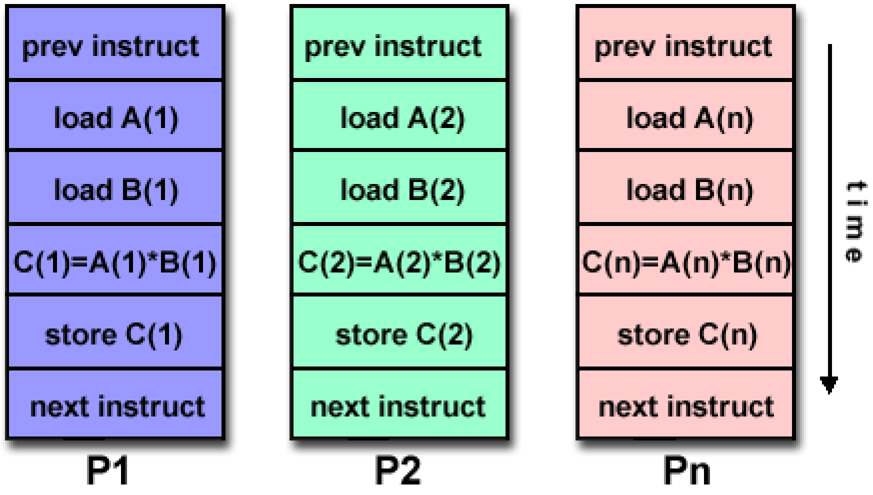
prev instruct
load A(1)
load B(1)
C(1)=A(1)*B(1)
store C(1)
next instruct
P1
prev instruct
load A(2)
load B(2)
C(2)=A(2)*B(2)
store C(2)
next instruct
P2
prev instruct
load A(n)
load B(n)
C(n)=A(n)*B(n)
store C(n)
next instruct
Pn
time

SIMD

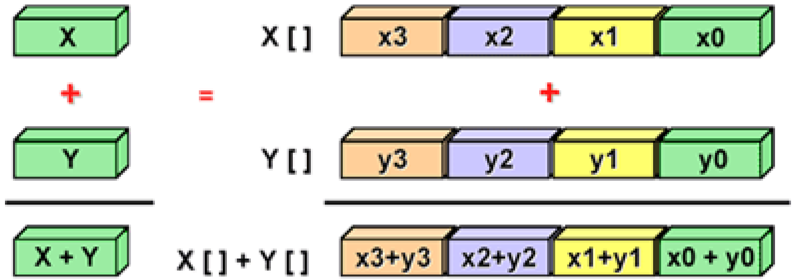


- Example: vector operations (e.g., Intel SSE/AVX, GPU)

SIMD



- Example: vector operations (e.g., Intel SSE/AVX, GPU)



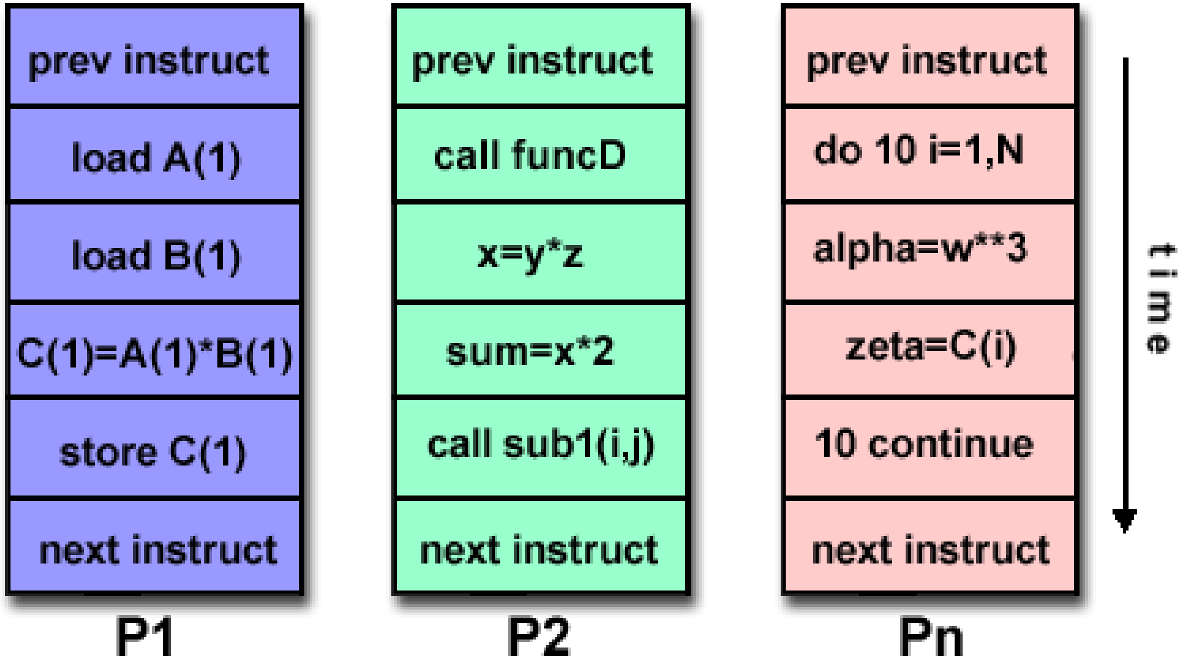
MIMD

MIMD

- Example: multi-core CPU

MIMD

- Example: multi-core CPU



Problem Partitioning

Problem Partitioning

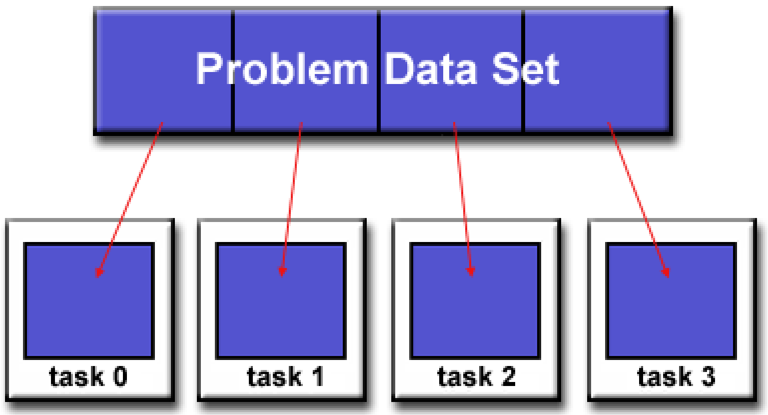
- Decomposition: Domain v. Functional

Problem Partitioning

- Decomposition: Domain v. Functional
- Domain Decomposition (Data Parallel)
  - SPMD
  - Input domain
  - Output Domain
  - Both

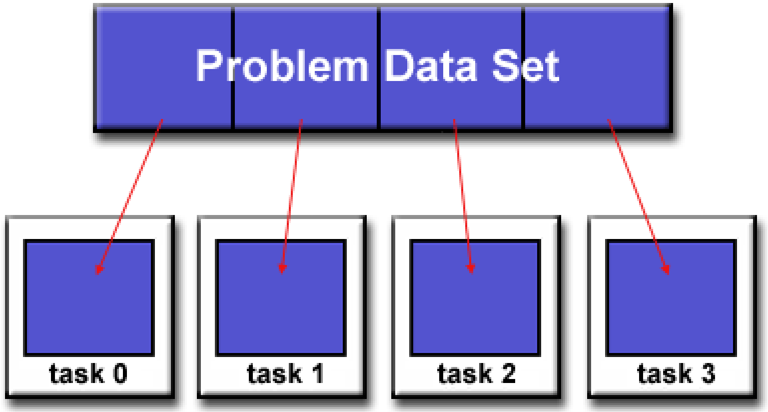
Problem Partitioning

- Decomposition: Domain v. Functional
- Domain Decomposition (Data Parallel)
  - SPMD
  - Input domain
  - Output Domain
  - Both



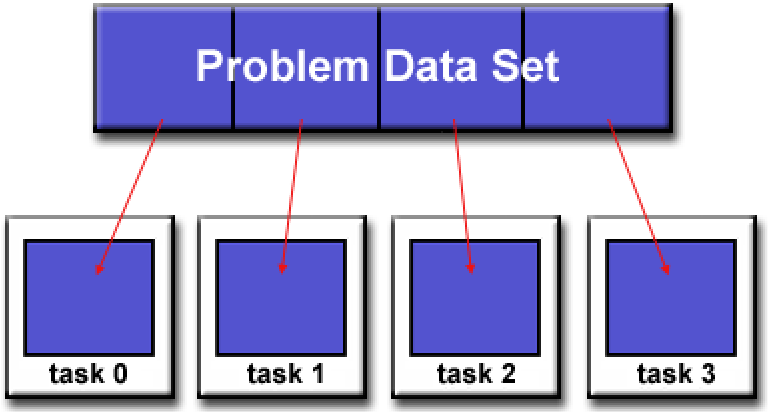
Problem Partitioning

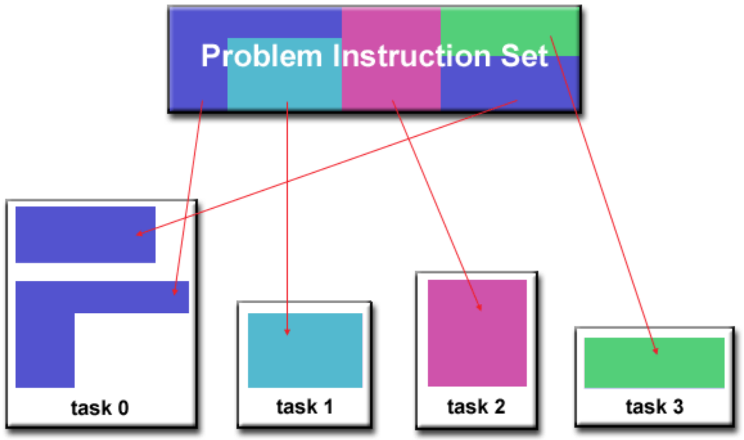
- Decomposition: Domain v. Functional
- Domain Decomposition (Data Parallel)
  - SPMD
  - Input domain
  - Output Domain
  - Both
- Functional Decomposition (Task Parallel)
  - MPMD
  - Independent Tasks
  - Pipelining



Problem Partitioning

- Decomposition: Domain v. Functional
- Domain Decomposition (Data Parallel)
  - SPMD
  - Input domain
  - Output Domain
  - Both
- Functional Decomposition (Task Parallel)
  - MPMD
  - Independent Tasks
  - Pipelining

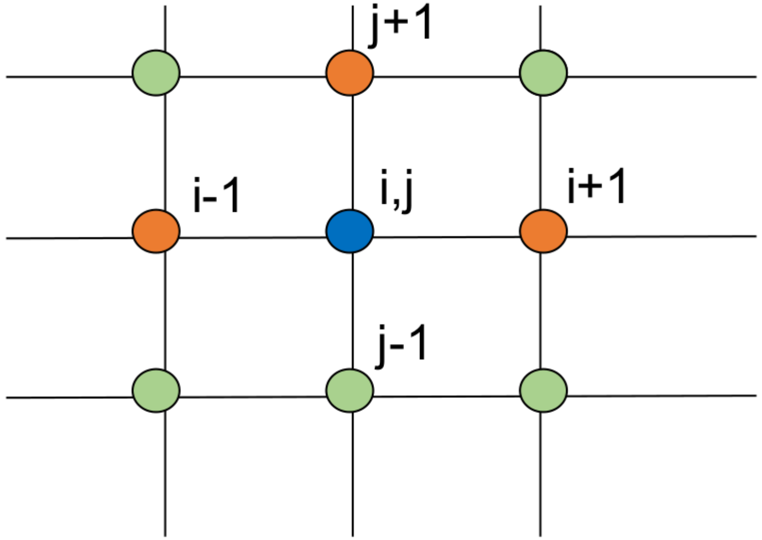




Game of Life

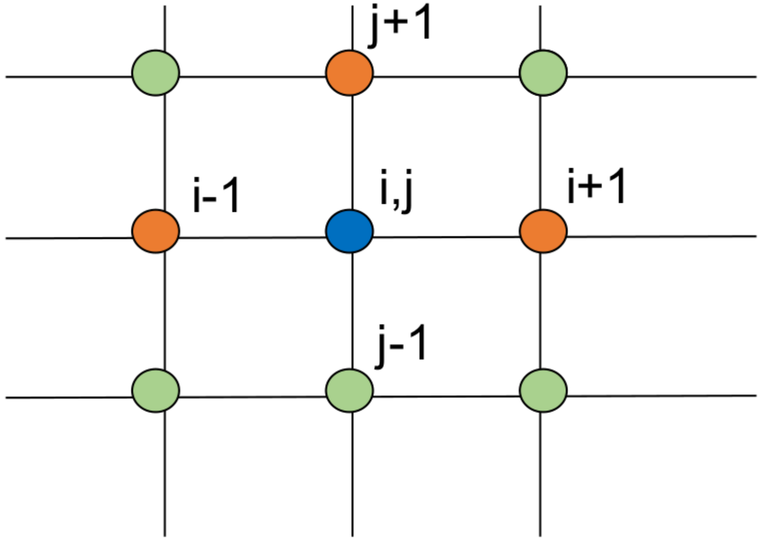
Game of Life

- Given a 2D Grid:
- $v_t(i,j) = F\left(v_{t-1}(of\ all\ its\ neighbors)\right)$



Game of Life

- Given a 2D Grid:
- $v_t(i,j) = F\left(v_{t-1}(of\ all\ its\ neighbors)\right)$



What decomposition fits “best”?

- Domain (data parallel)
- Functional (task parallel)

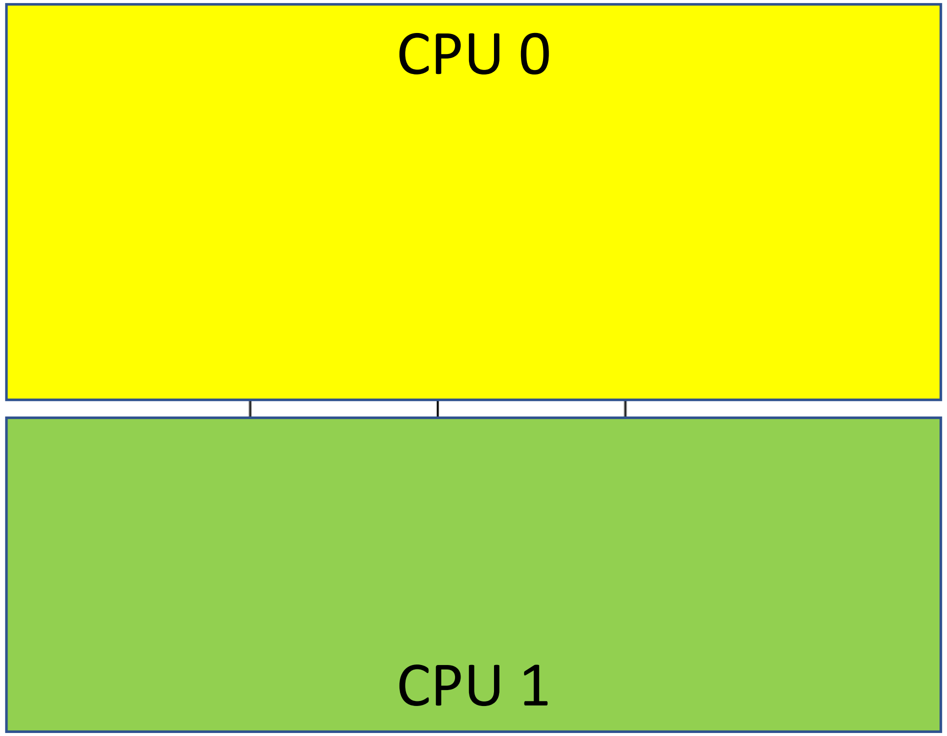
Domain decomposition

Domain decomposition

Each CPU gets part of the input

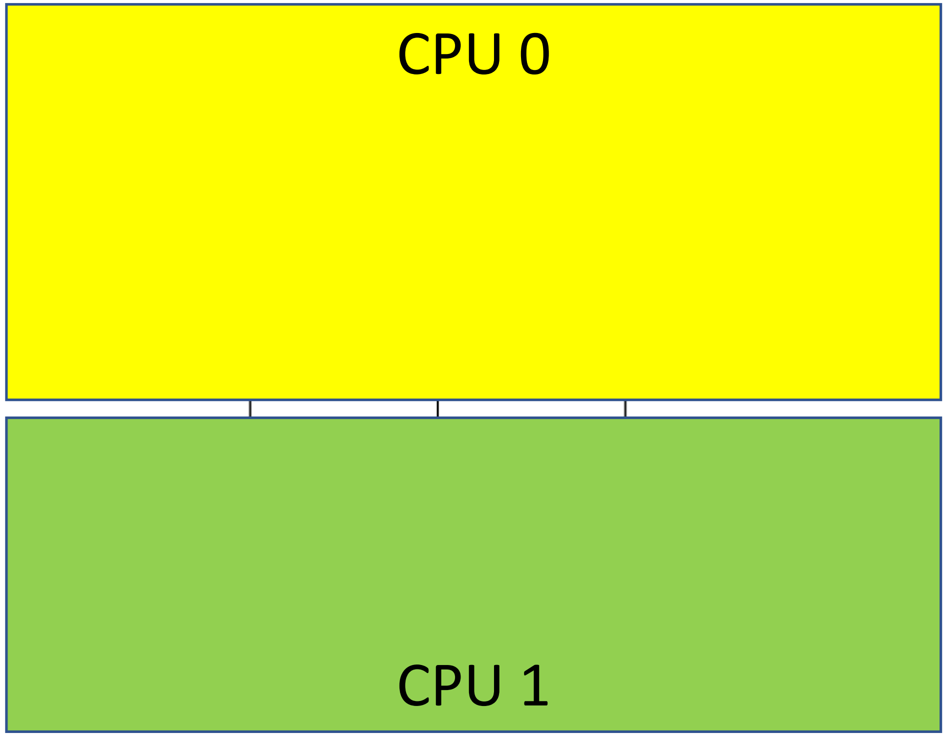
Domain decomposition

Each CPU gets part of the input



Domain decomposition

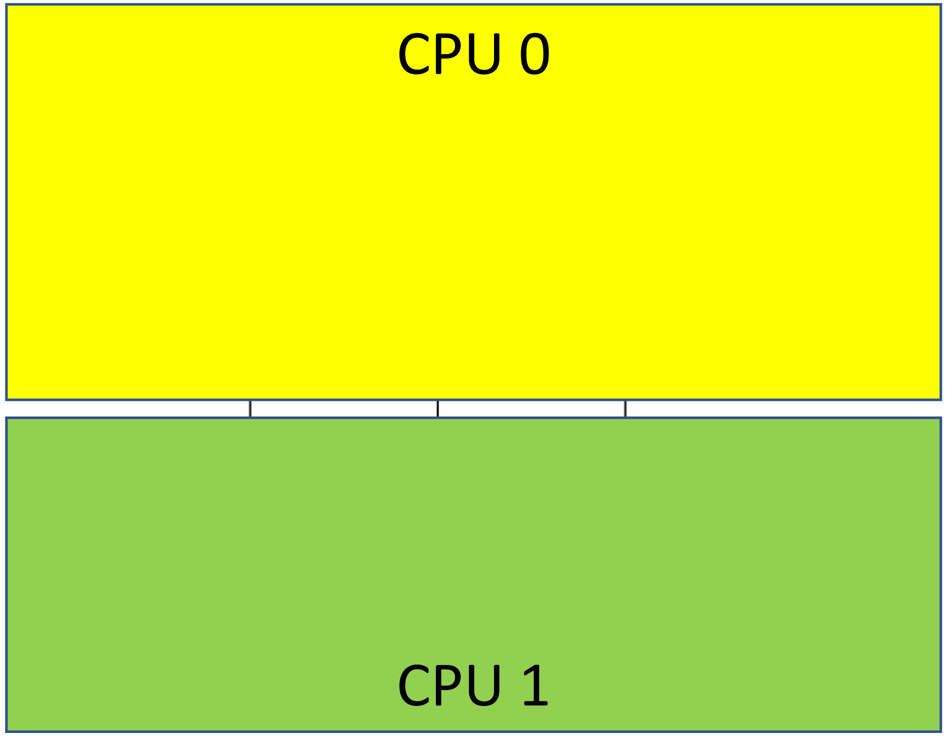
Each CPU gets part of the input



For next time:

Domain decomposition

Each CPU gets part of the input

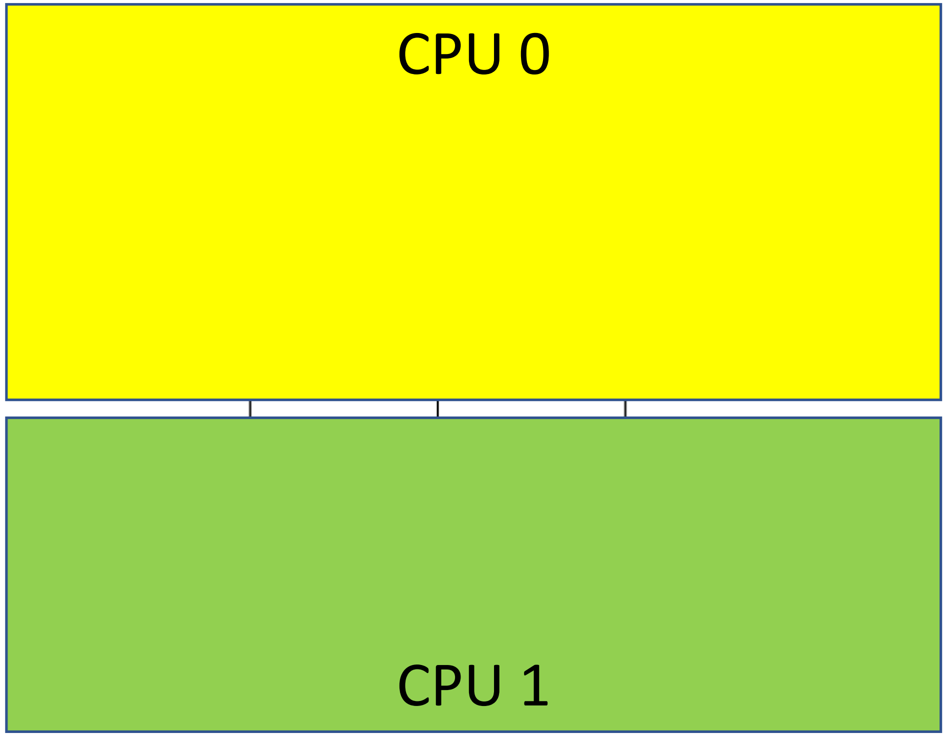


For next time:

- What issues/challenges might arise with this solution?

Domain decomposition

Each CPU gets part of the input



For next time:

- What issues/challenges might arise with this solution?
- How could we do a functional decomposition?