EASTERN WASHINGTON UNIVERSITY
start something big

Threads and Flynn's Taxonomy

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Outline Today

- Concept of Threads
- Data Parallel and Task Parallel
- Flynn's Taxonomy
- Concept of Parallelizing a Problem
- Some Parallel Patterns

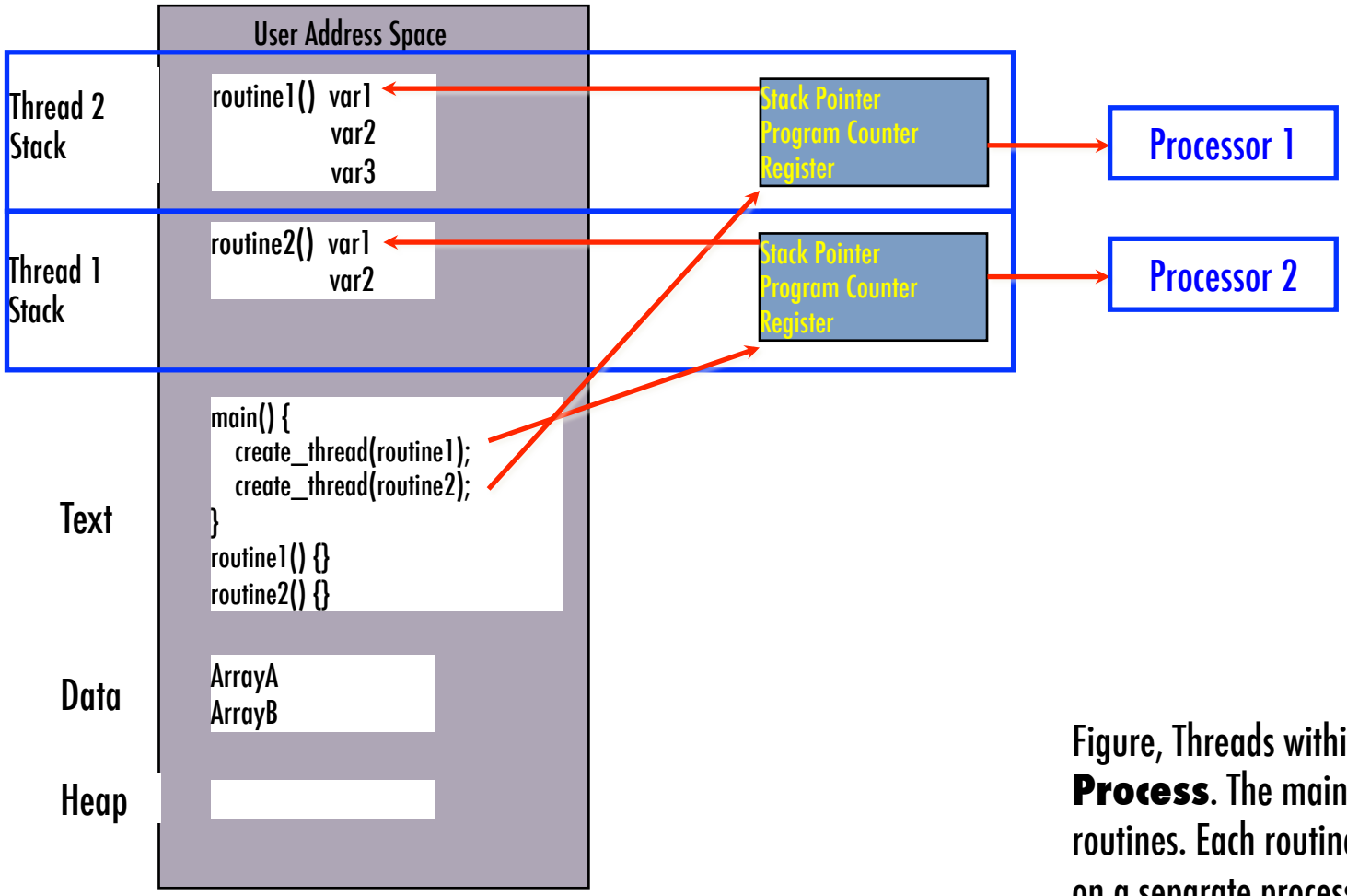
Concept of Thread

- GPU programming also uses the concept of Thread.
- A thread, or thread of execution, is a unit of parallelism.
- A thread has everything needed to execute a **stream of instructions**,
  - a private program text, a call stack, a program counter.
- All threads belonging to the same process (program) can see the **global** memory belonging to the program.
  - Multiple threads can cooperate to compute on global data.

Concept of Thread

- From the **programmer's point of view**, a thread means a procedure (routine or function) that can be scheduled to run on a processor.
  - Inside a single thread, instructions are executed sequentially.

Concept of A Thread



Figure, Threads within a **Unix Process**. The main program has two routines. Each routine might be running on a separate processor simultaneously.

Concept of Thread

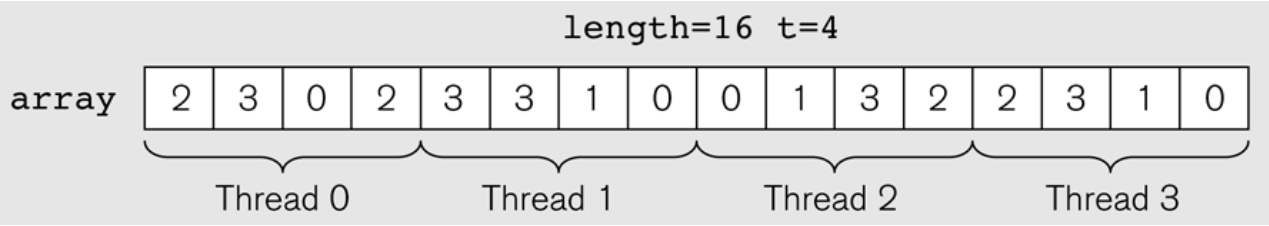
- Multiple threads of execution can run concurrently.
  - On a single processor/core, the processor switch rapidly between threads.
  - On multiple processors/cores, threads may execute truly simultaneously (in parallel).
- A **Linux/Unix** program begins with a single thread, the main thread.
  - We can create threads using functions from the pthreads library.
  - An arbitrary number of threads can be created.
  - When a thread finishes, it typically join the main thread. The main thread must be waiting for the join to occur.

Data Parallel and Task Parallel

- Data Parallel Program
  - Each process or thread does the same thing on its part of the data.
    - Different thread works on different part of data.
  - E.g. Matrix Multiplication, JPEG compression
- Task Parallel Program
  - Different processes or thread carry out different tasks.
  - E.g, Producer-Consumer Problem, I/O prefetching, Pipelines.

Data Parallel and Task Parallel

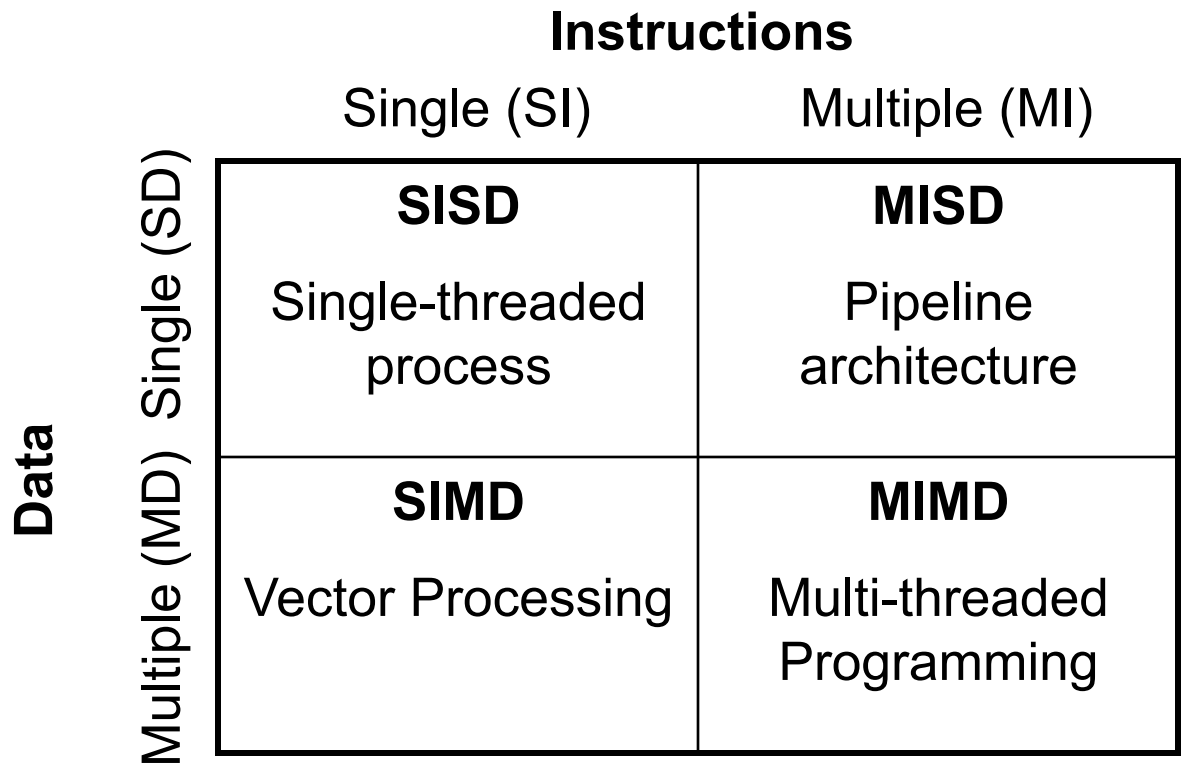
- Data Parallel Program Example **Counting 3s**
  - Counting how many integer 3 exists in an array.
  - To implement a parallel version of this code, we need partition the array,
    - so that each thread is only responsible for counting the number of 3s in 1/***t*** of the array, where ***t*** is the number of threads.



Schematic diagram of data allocation to threads. Allocations are consecutive indices.

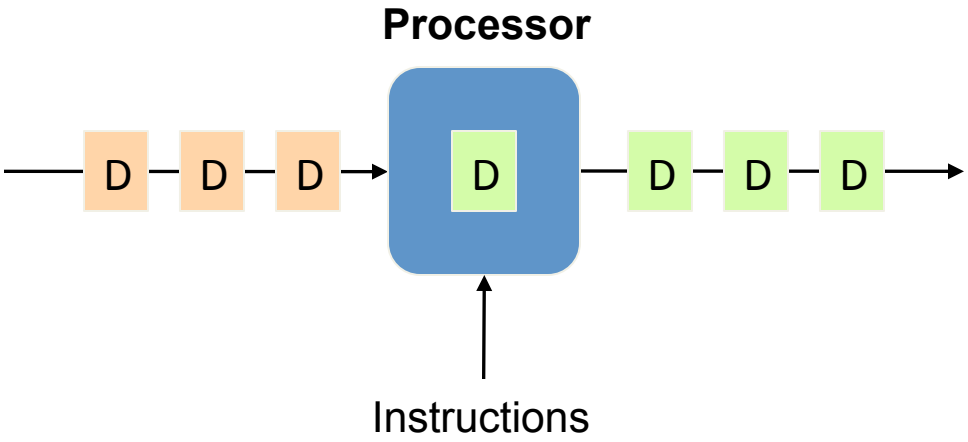
Flynn's Taxonomy

- Categories of Parallel Computer Architectures



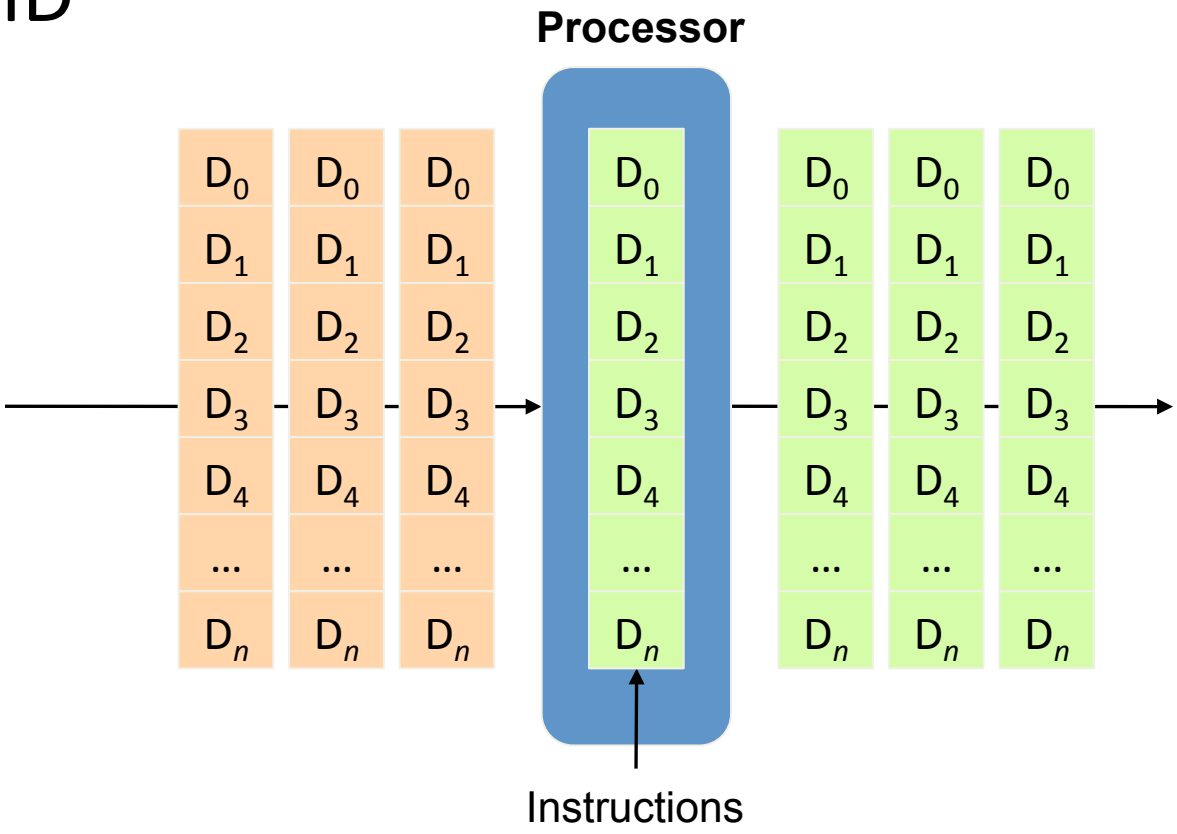
Flynn's Taxonomy

- SISD
  - The usual and typical sequential computer.
  - A single instruction is performed on a single data element at a time.



Flynn's Taxonomy

- SIMD



Flynn's Taxonomy

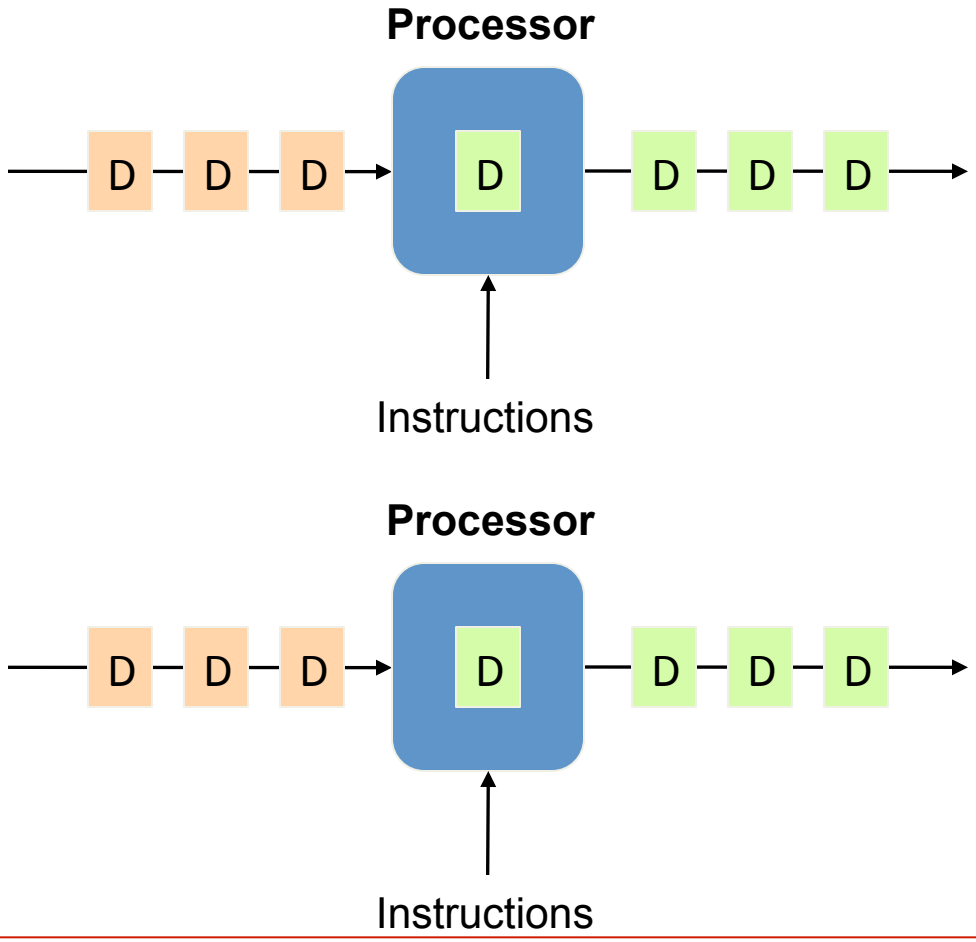
- SIMD
  - Many processors or functional units do the same thing at the **same time** to multiple data elements.
  - E.g. a vector processor has **n** functional units, when doing "vmul v1 v2", one operation applied to multiple dataset elements in vector multiplication.
  - All elements perform the same calculation.
- Great for homogenous computation.

Flynn's Taxonomy

- SIMD Limitations
  - Processing elements (PE) cannot execute different instructions at a same time.
  - What about an if statement? or a while loop?
  - Vector processors do the same thing to all values, so cannot support conditional directly.
  - In strict SIMD, loops must execute the same number of times on all processors.
  - What about linked lists? or trees?
    - They must all be the same length or shape.

Flynn's Taxonomy

- MIMD



Flynn's Taxonomy

- MIMD
  - Threaded programming on multicore.
  - Different processors can do different things to different data.
- MISD
  - E.g. Redundancy in real time control systems such as Apollo II guidance system or Jet fighter control system
    - Use redundant calculation for reliabilities.

Flynn's Taxonomy

- SPMD – Single program, Multiple Data
  - All threads in a program execute the same thread function.
    - Processing Units(PU) execute the same program on different part of data.
    - Traditional pthreads program or MPI can do this.
    - Cuda programming also falls into this category.
  - Differs from SIMD, in that Processing Units in SPMD system do NOT need to be executing the same instruction at the same time.

GPU's Terminology

- SIMT(Single Instruction Multiple Threads)
  - Nvidia's term to describe CUDA
  - Specific to GPU programming,
  - Unlike classic SIMD, we do NOT need to know how many Processing Elements (PE) there are.
  - Execution is not strictly synchronized.
    - Threads may execute nearby instructions simultaneously.
    - PE1 can execute instruction **i**, when PE2 is executing instruction **i + 10** inside a same function.
  - We will have more info. when talking about execution model on GPUs.

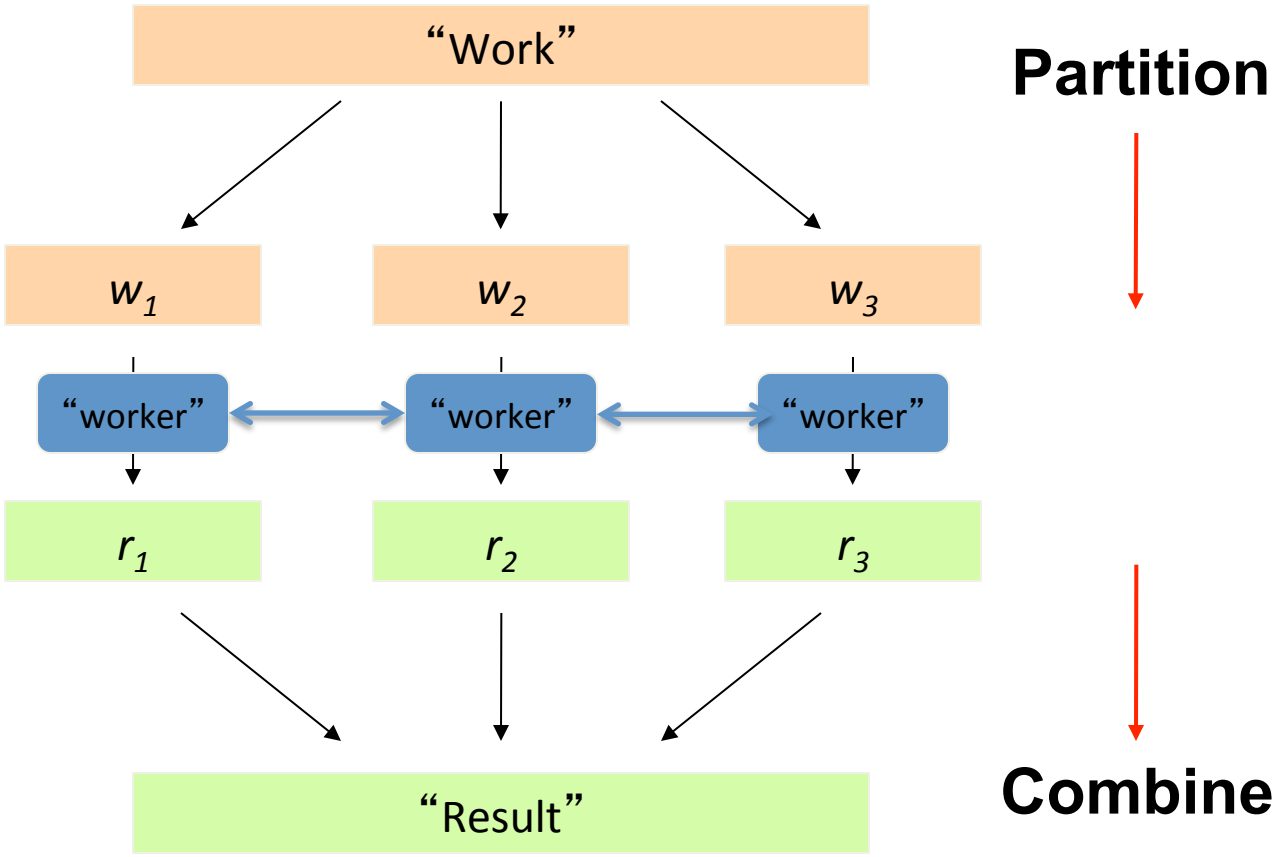
Parallelizing a Problem

- It all boils down to
  - Divide-and-conquer
  - Throwing more hardware at the problem
- Some thoughts
  - Identify the part is parallelizable.
    - loops, recursion
    - Data dependency or independent
    - Access shared resources
    - Consider patterns, like reduction, producer-consumer, client-server, etc.

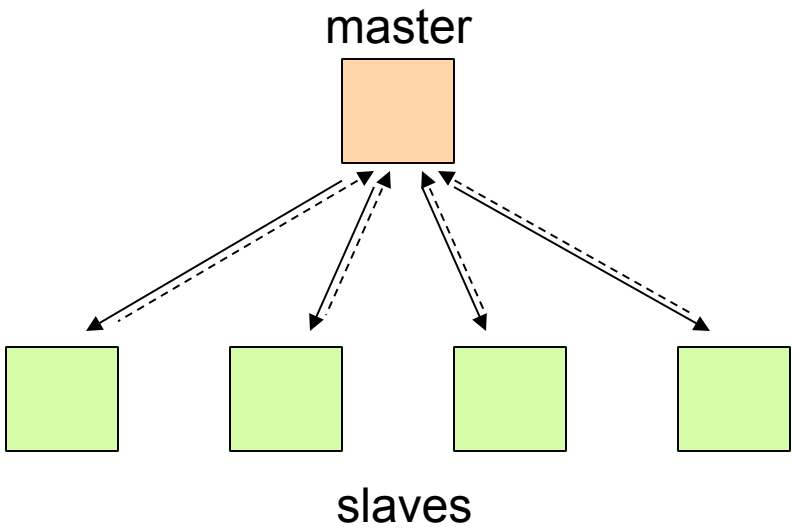
Parallelizing a Problem

- Parallelization problems arise from:
  - Communication between workers
  - Access to shared resources (e.g., data)
- Thus, we need a synchronization system!

Parallelize Problem

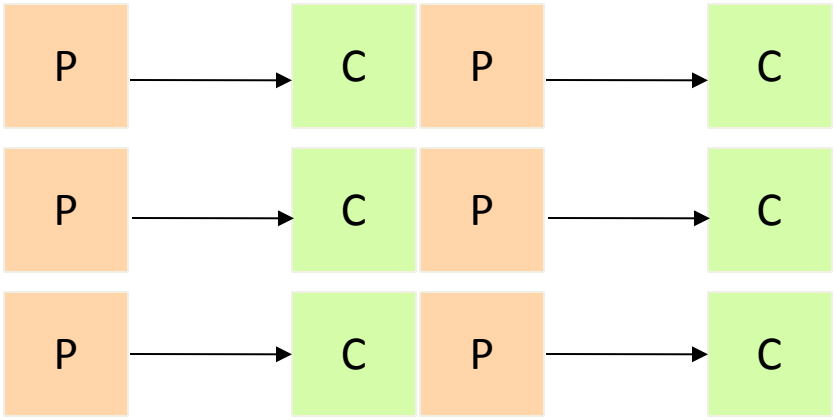


Parallel Paradigms



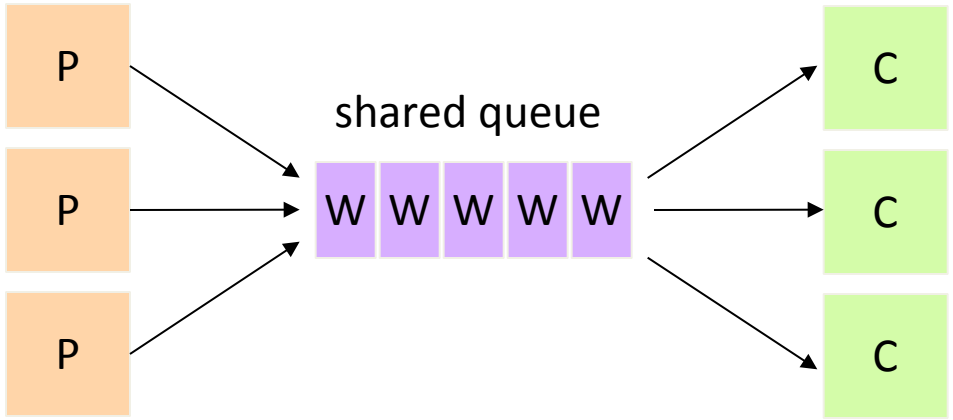
Parallel Paradigms

– Producer-Consumer Flow(pipeline)



Parallel Paradigms

- Work Queue



Summary

- Concept of Threads
- Data Parallel and Task Parallel
- Flynn's Taxonomy
- Concept of Parallelizing a Problem
- Some Parallel Patterns

Next Class

- Demo of Data Parallel with pthreads.
- Warm up Homework one