“My hypothesis is that we can solve [the software crisis in parallel computing], but only if we work from the algorithm down to the hardware — not the traditional hardware first mentality”

Tim Mattson, Principal Engineer at Intel, 1993

Lecture A.4: Application Parallelism

CS205: Computing Foundations for Computational Science
Dr. David Sondak
Spring Term 2021

HARVARD
School of Engineering and Applied Sciences

IACS
INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

Lectures developed by Ignacio M. Llorente

Before We Start

Where We Are

Computing Foundations for Computational and Data Science

How to use modern computing platforms in solving scientific problems

CS205: Contents

APPLICATION SOFTWARE

APPLICATION PARALLELISM

PARALLEL PROGRAM DESIGN

PROGRAMMING MODEL

Optimization

OpenACC

OpenMP

MPI

Spark

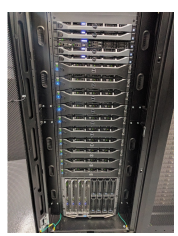
Map-Reduce

B. BIG COMPUTE

PLATFORM

C. BIG DATA

amazon web services



Open Nebula

FAS RC

FAS RC CANNON HARVARD'S LARGEST CLUSTER

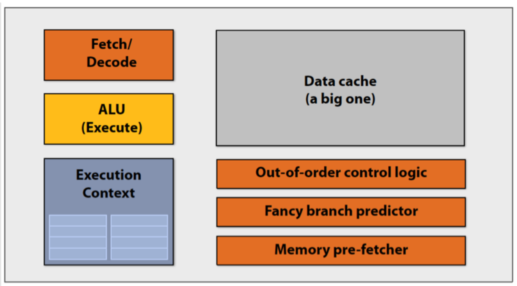
CLOUD COMPUTING

PARALLEL ARCHITECTURES

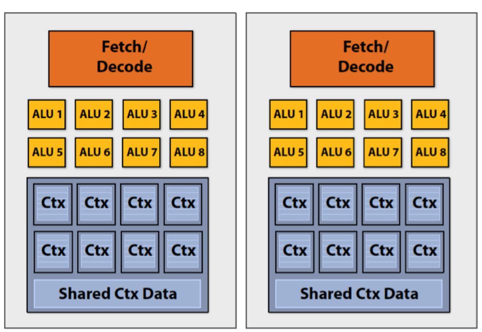
Context

Application Parallelism

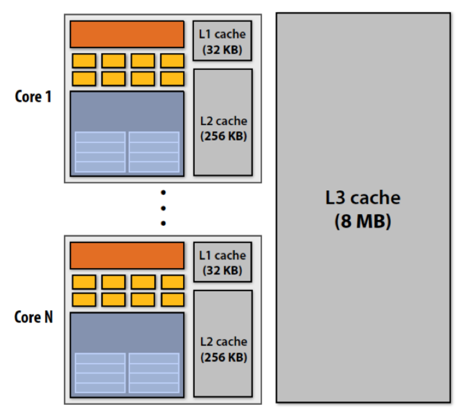
How to make effective use of existing parallelism at different levels?



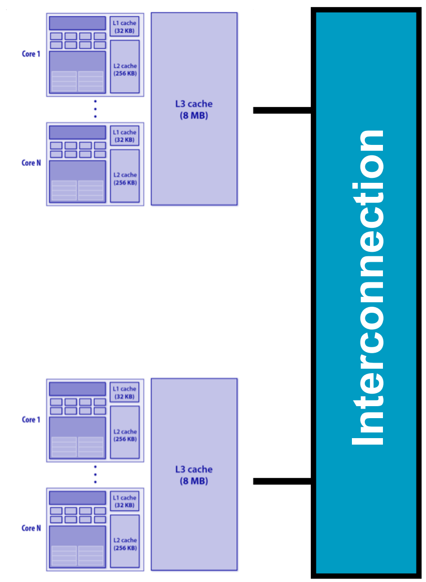
ILP/Data



Many-core



Multi-core



Multi-node

Roadmap

Application Parallelism

Types of Applications

Levels of Parallelism

Types of Parallelism

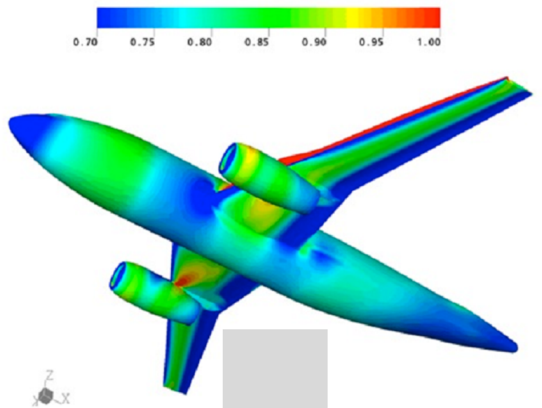
Parallel Execution Models

Types of Applications

Types of Applications

Big Compute vs Big Data

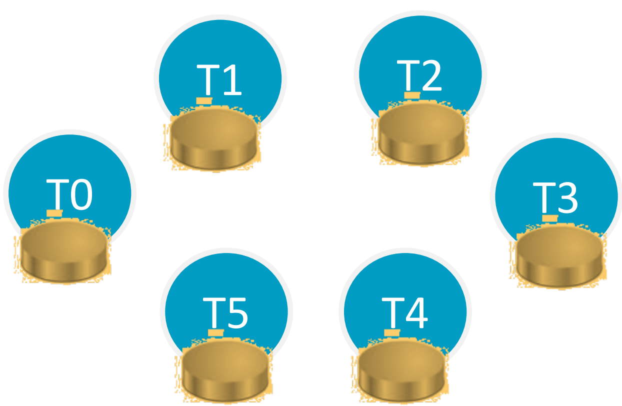
"Big" Compute



Big Data

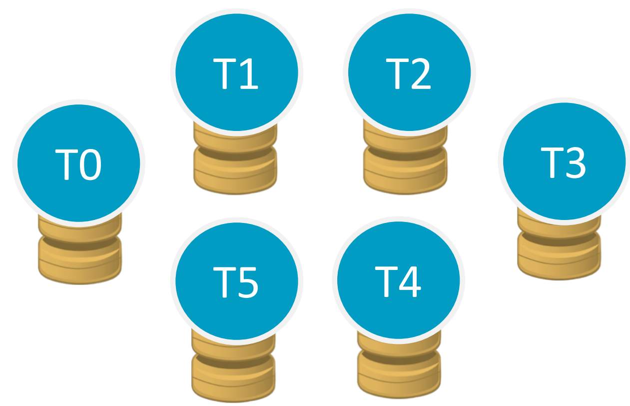
Compute-intensive

Bringing data to compute



Data-intensive

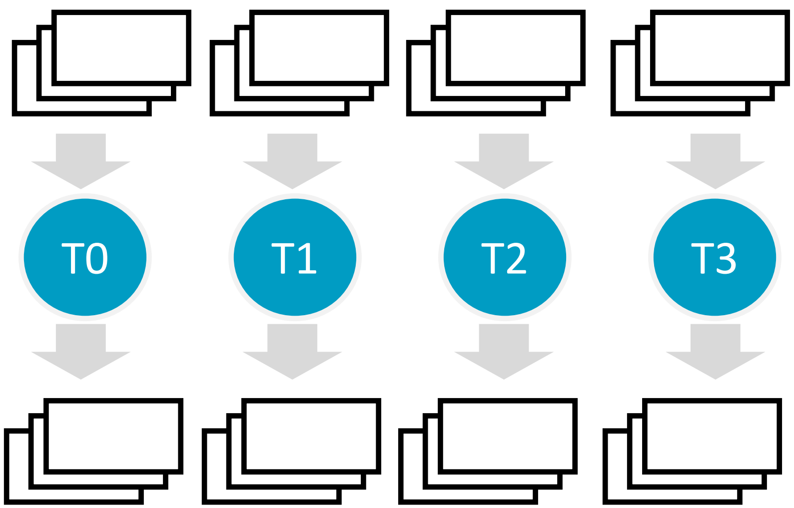
Bringing compute to data



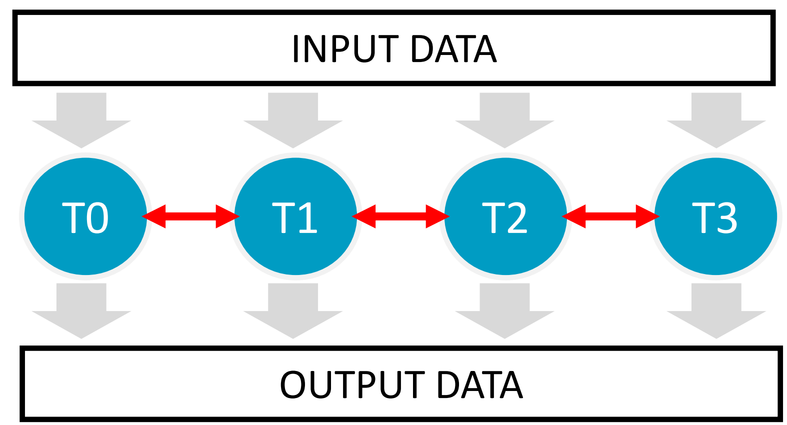
Types of Applications

Compute-Intensive Applications

Paradigm	Independent parallel tasks that are performed simultaneously to address a particular part of the problem
Challenge	Decompose the application into tasks and define their communication and synchronization
Bottleneck	CPU/Network
Input data	Gigabyte-scale to describe initial conditions
Programming	Optimization, OpenACC, OpenMP and MPI



HTC (Capacity)



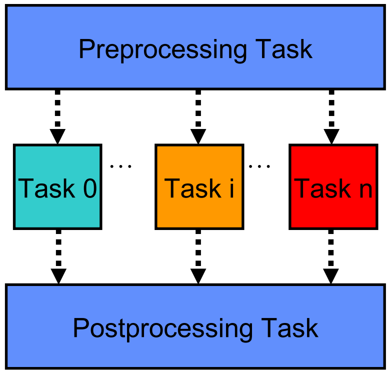
HPC (Capability)

Types of Applications

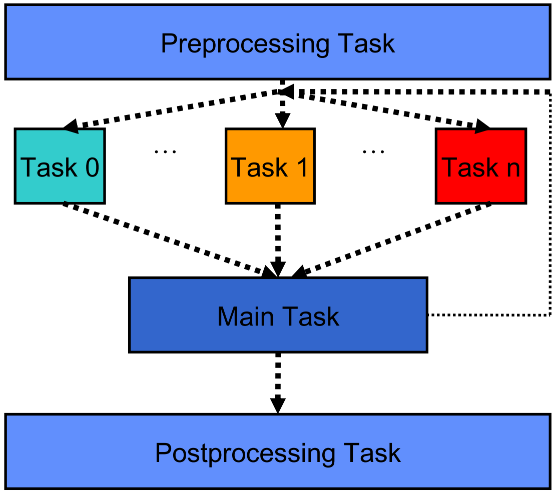
Compute-Intensive Applications

High Throughput Computing (Capacity)

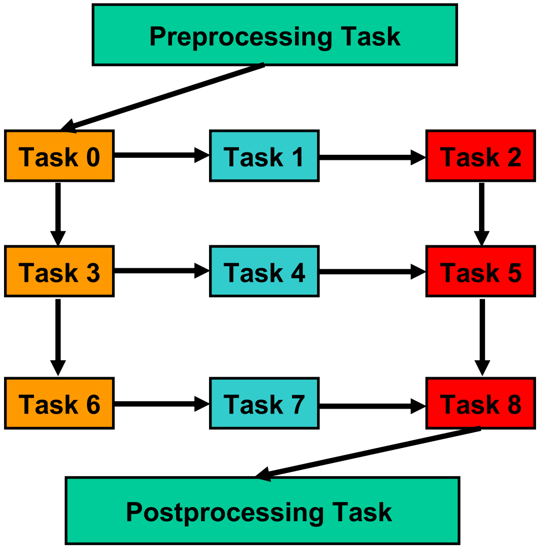
- Application consists of a very large number of independent tasks to explore design space
- Tasks could have dependencies, as in the case of workflows
- Optimize for throughput: number of tasks executed per time unit



Embarrassingly Parallel



Main-worker



Types of Applications

Compute-Intensive Applications

HTC: Simplest Approach to Distributed Processing

- Visualization: Image rendering
- Bioinformatics: DNA sequence analysis
- Marketing: Analyze the purchase pattern of BestBuy customers in MA
- High Energy Physics: Generate 10^6 CMS experiment events
- IT: Database queries
- Finance: Monte Carlo to value and analyze instruments, portfolios and investments
- Automotive: Mechanical forensics using Monte Carlo for Crash Analysis
- …

Types of Applications

Compute-Intensive Applications

Examples?

Number of Instances?

Size of each instance?

MTC: Many-task Computing

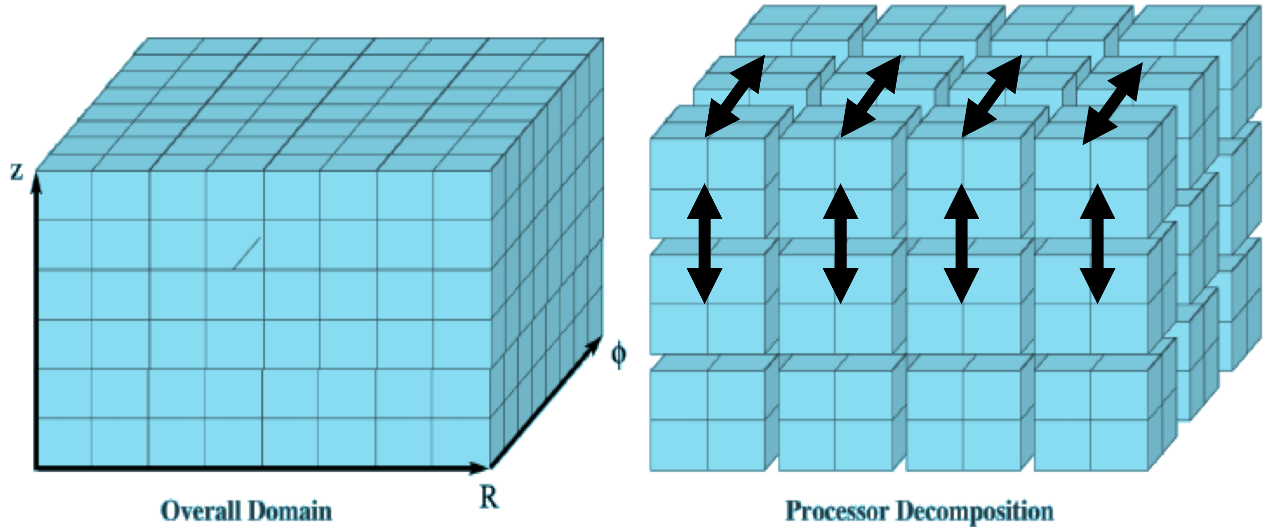
- MTC is reminiscent of HTC, but it differs in the emphasis of using many computing resources over short periods of time to accomplish many computational tasks

Types of Applications

Compute-Intensive Applications

High Performance Computing (Capability)

- High-fidelity long-running large-scale computations
- Application consists of multiple parallel tasks that are performed simultaneously to address a particular part of the problem
- Tasks communicate and/or synchronize with each other
- Optimize for speed: number of floating point operations per time unit



Types of Applications

Compute-Intensive Applications

HPC: Parallel Programming

- Aeronautics: CFD simulation
- Earth Science: Simulation of weather and science
- …

Types of Applications

Compute-Intensive Applications

Examples?

Level of coupling?

Tightly-coupled vs. Loosely-coupled

Types of Applications

Compute-Intensive Applications

HTC	HPC
Independent execution of multiple instances of the same application	Parallel execution of a single large application instance
Throughput (jobs per second)	Speed-up (execution time)
Straightforward implementation, it does not require parallel programming	Difficult and time-consuming implementation
Main challenge is to tune the job scheduler (LRMS) of the management platform	Main challenge is to tune the parallel implementation of the code
Requires fast storage access	Requires low latency, high bandwidth network (Tightly-coupled vs. Loosely-coupled)

Main part of the course

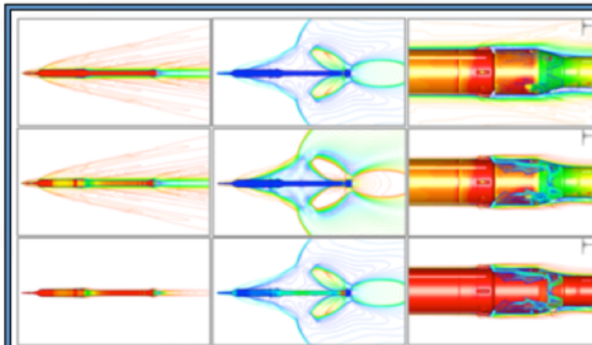
HA (maintain readiness): Time-sensitive mission-critical applications that require HPC resources on-demand (NASA)

Types of Applications

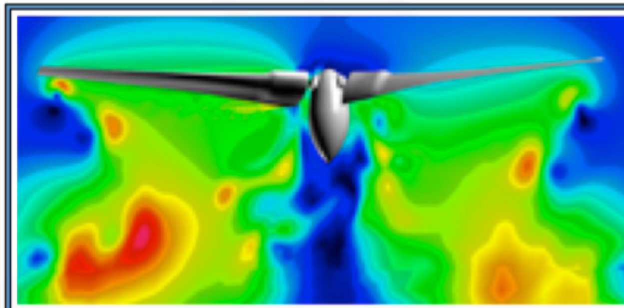
Compute-Intensive Applications

NASA's Diverse HPC Requirements

1) Engineering requires HPC resources that can process large ensembles of moderate-scale computations to efficiently explore design space **(high throughput / capacity)**
2) Research requires HPC resources that can handle high-fidelity long-running large-scale computations to advance theoretical understanding **(leadership / capability)**

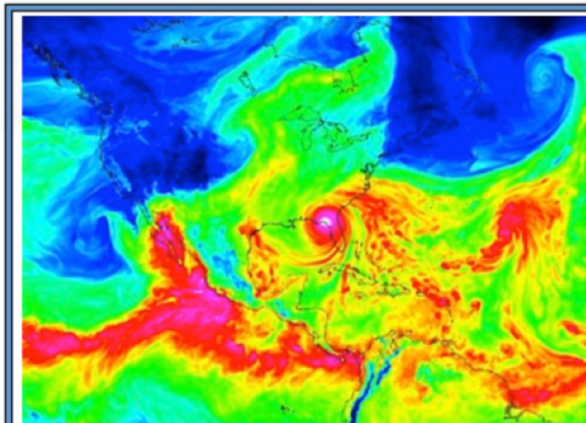


Aerodynamic database generation



Rotary wing unsteady aerodynamics

3) Time-sensitive mission-critical applications require HPC resources on demand **(high availability / maintain readiness)**



Real-time hurricane prediction

National Aeronautics and Space Administration

6

http://hpcuserforum.com/presentations/korea2013/Biswas-NASA%20HPC-UF-Seoul.pdf

Types of Applications

Compute-Intensive Applications

What is more difficult to code, HPC or HTC?

What is the critical part for achieving performance in both profiles?

What type of architecture is more cost-effective in both cases?

Types of Applications

Different Approaches

Centralized
Coupled

- **Network Links**
- **Administration**
- **Homogeneity**

Decentralized
Decoupled

MPP (Massive Parallel Processors)

Dedicated Clusters

Network Systems Intranet/Internet

High Performance Computing

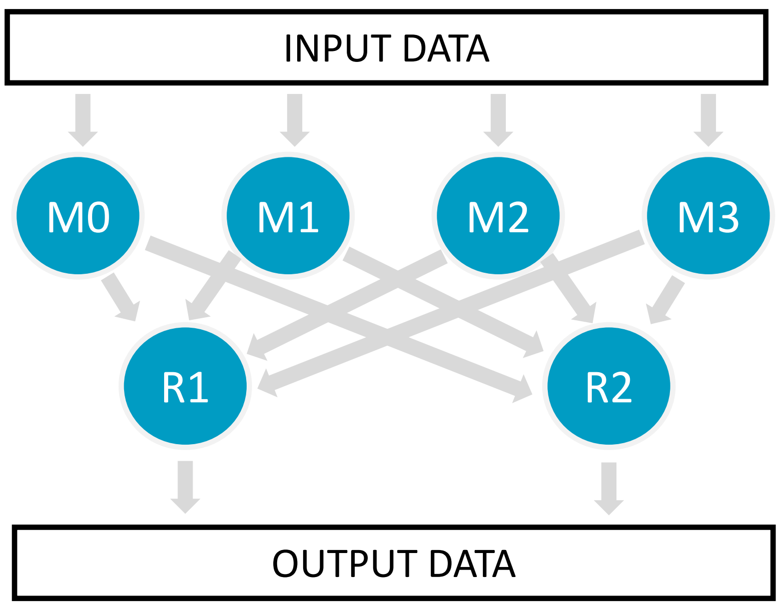
High Throughput Computing

Where to invest my budget?

Types of Applications

Data-intensive Applications

Paradigm	Same task is applied to large volumes of data
Challenge	Partition the data into multiple segments and the subsequent combination of the intermediate results in multiple stages
Bottleneck	Storage
Input data	Far beyond gigabyte-scale: datasets are commonly on the order of tens, hundreds, or thousands of terabytes
Programming	MapReduce, Spark



Types of Applications

Data-intensive Applications

Science is Facing a Scalability to 'Big Data' Challenge.

- Scalability of applications
  - Data set is growing exponentially
    - Weather and Climate Modelling
    - Life sciences data driven bio genomics: Molecular data for species doubling every 18 months
    - Computational Neuroscience: 2 PetaBytes for $1mm^3$ in human connectome
    - Computational Cosmology
    - Computational Ecology
  - Modelling complexity is increasing
    - Ecology combined with data driven biology
- New data models: distributed arrays, trees, hash tables, dictionaries (key-value pairs)
- Resilience and failure of components is the norm with billions of threads on millions of processors

Types of Applications

Data-Intensive Applications

Examples?

Big data?

Data Science vs. Big Data

Levels of Parallelism

Levels of Parallelism

From Bit to Application Parallelism

EXTERNAL

Coarse-grained

Application Level
- Independent applications within a system

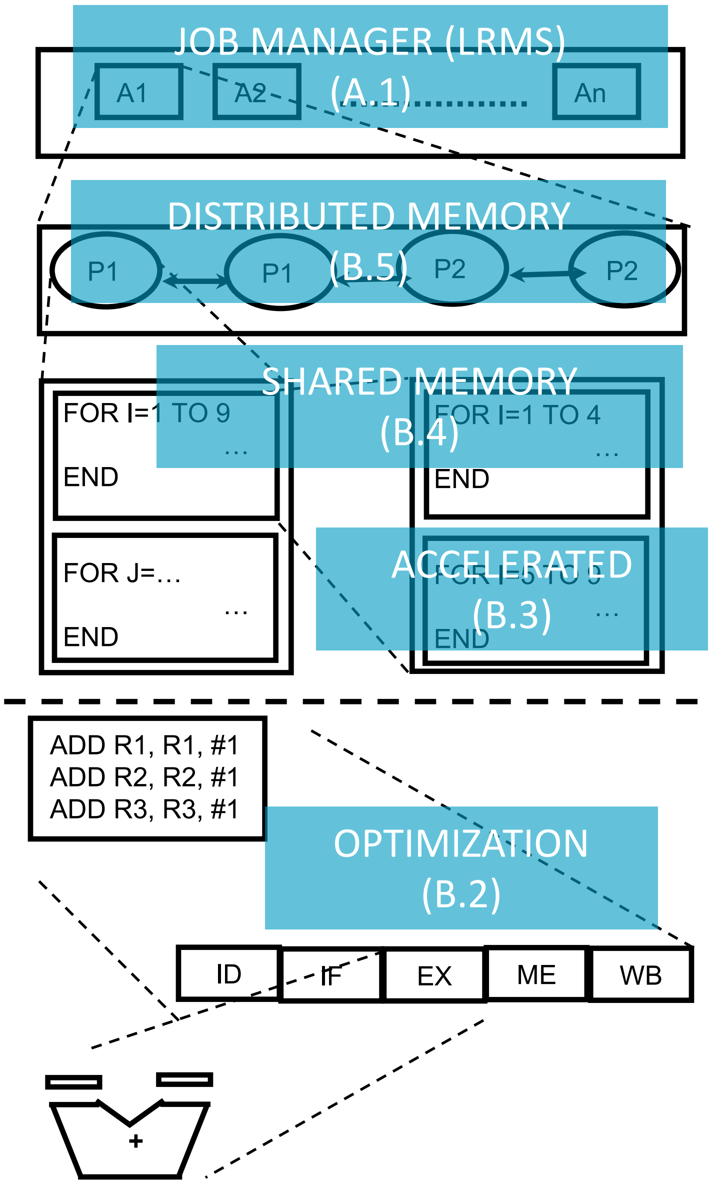
Task Level
- Interrelated tasks within an application

Fine-grained

Procedure Level
- Regions of code within a task

Loop Level
- Iterations within a loop

THREADS



INTERNAL

Instruction Level => SUPERSCALAR
- Instructions with an executable

Instruction Phase Level => PIPELINING
- Overlap execution of multiple instructions

Bit Level => ARITHMETIC UNIT
- Processor word size

Levels of Parallelism

Granularity

Application Level	**Very-coarse Grained** > Tens of thousands of instructions Exploited by job managers (SLURM)
Task Level	**Coarse Grained** > Thousands of instructions Exploited by the programmer (MPI)
Procedure Level	**Medium Grained** > Thousand instructions Exploited by the programmer (OpenMP)
Loop Level	**Fine Grained** < 500 instructions Exploited by the programmer (OpenMP/OpenACC)
Instruction Level	**Very-fine Grained** < 20 instructions Exploited by the compiler and the CPU (-O3)

Levels of Parallelism

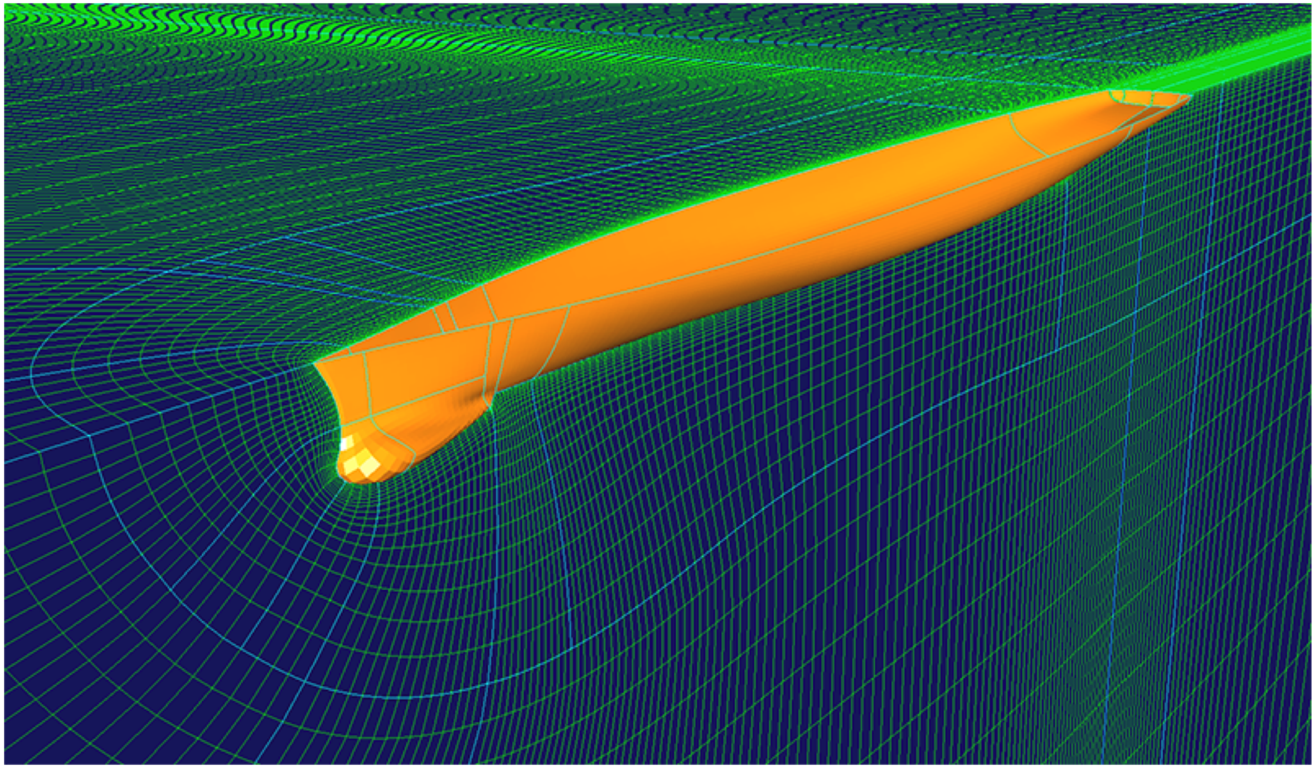
Towards a Hybrid Approach

Coarse-Grained	Fine-Grained
Higher level parallelism: Domain decomposition in numerical simulation	Lower level parallelism: Loop decomposition
Does require knowledge about the application	Does not require knowledge about the algorithm or method implemented by the code
Development of a new parallel version of the application	Parallelization of an existing "sequential" code version
Higher scalability and, in principle, efficiency	Faster development
Distributed memory, multi-node, MPI-like	Shared memory, multi-core, OpenMP-like

Hybrid Parallelism

Levels of Parallelism

Towards a Hybrid Approach



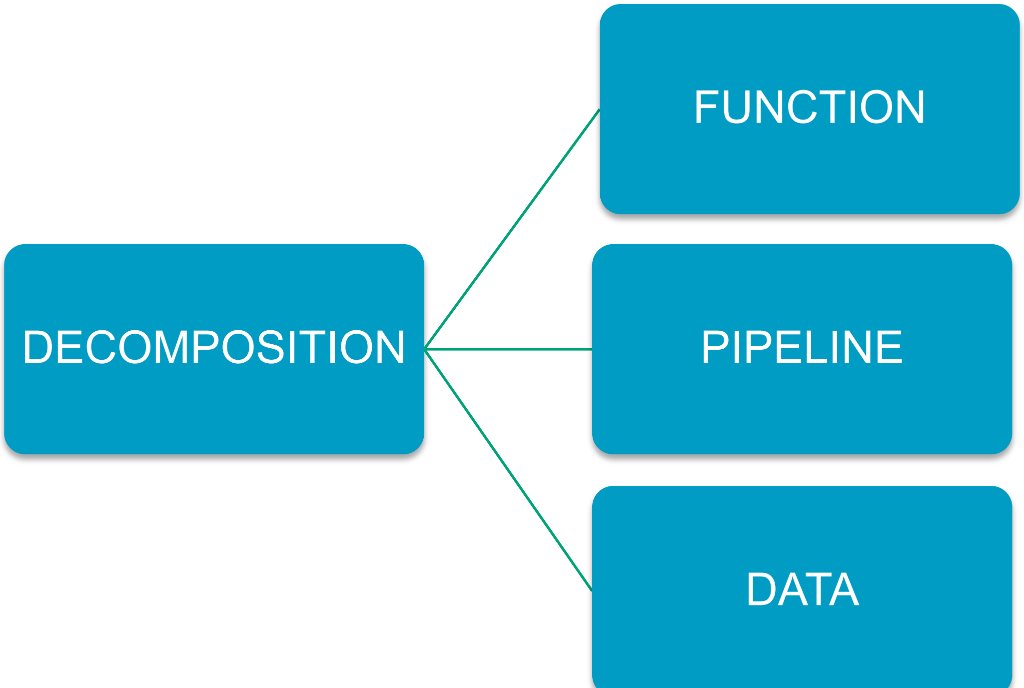
Evaluate the Application vs. Evaluate the Code

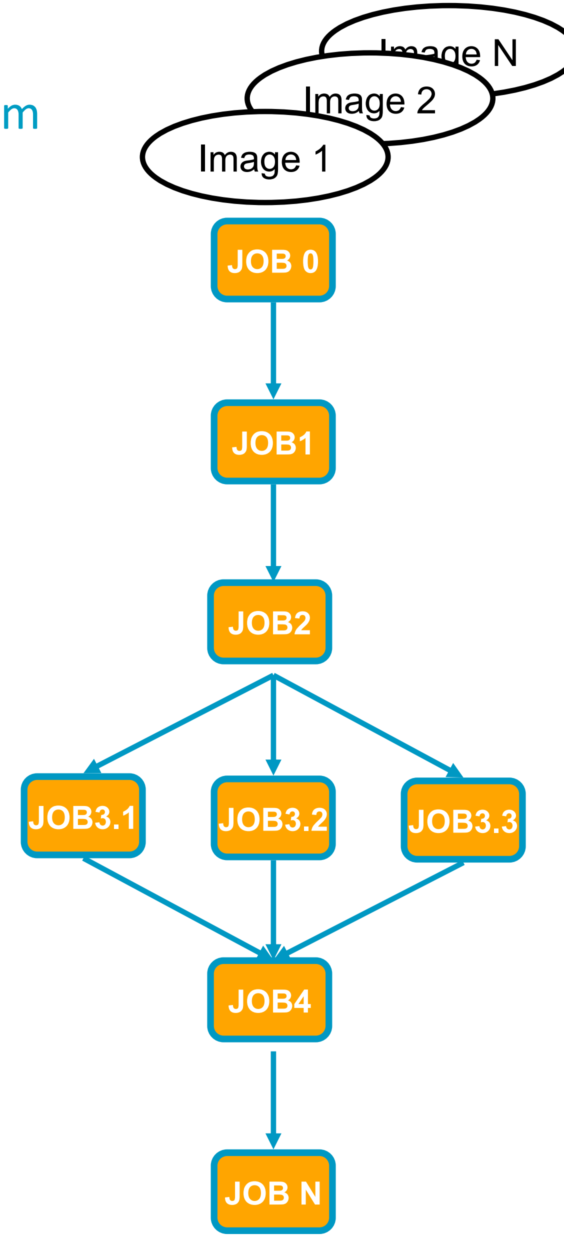
Develop a Parallel App vs. Parallelize a Code

Types of Parallelism

Types of Parallelism

Data, Function and Pipeline Parallelism



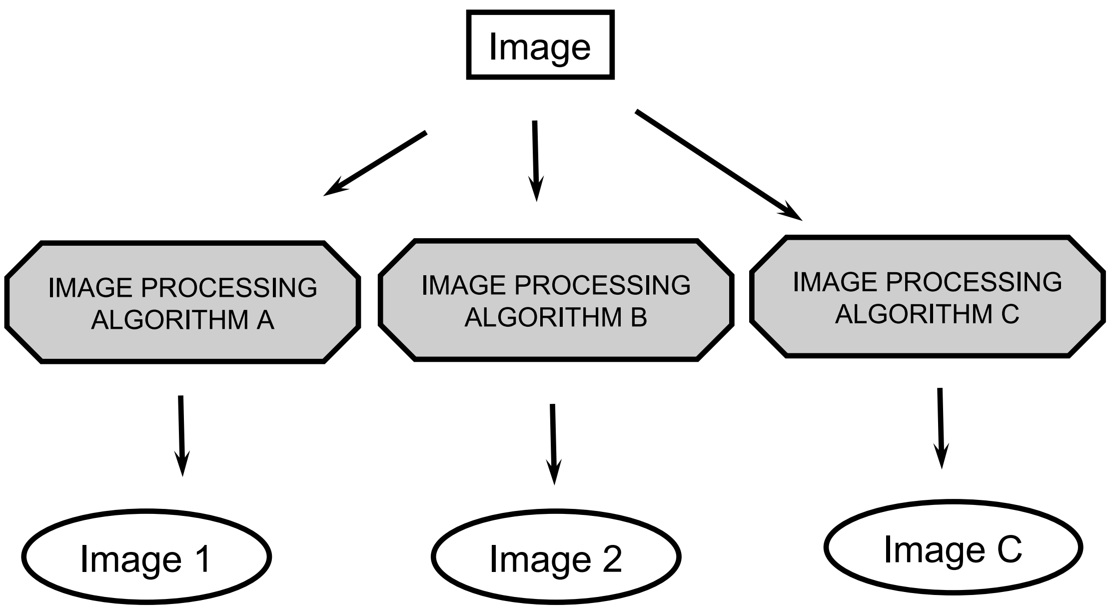


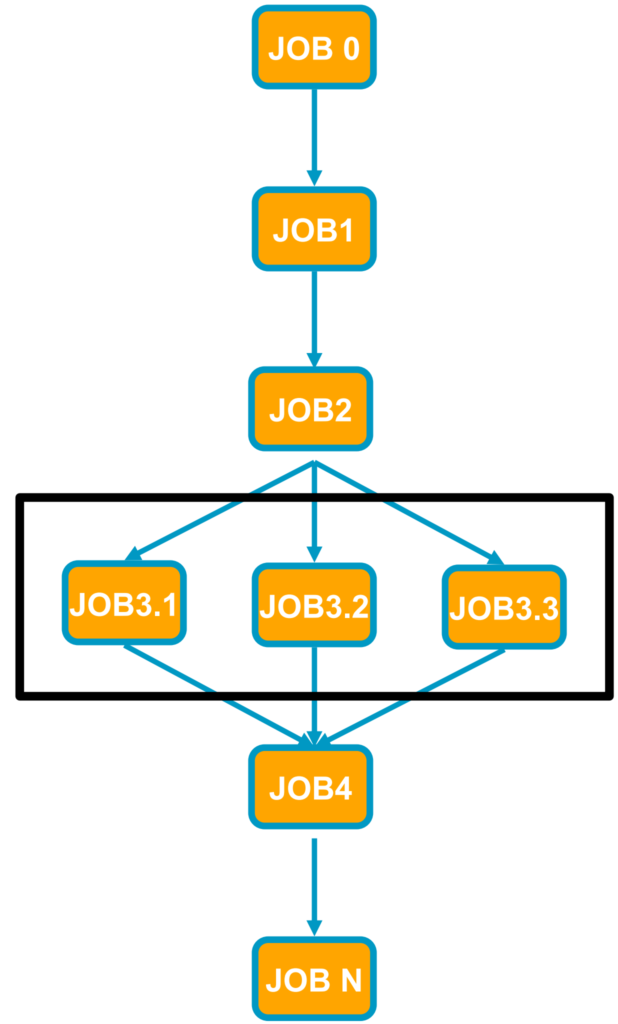
Types of Parallelism

Function (Control) Parallelism

Different Jobs Running on the Same Data

- Several functions on the same data
- No dependencies between the tasks, so all can run in parallel



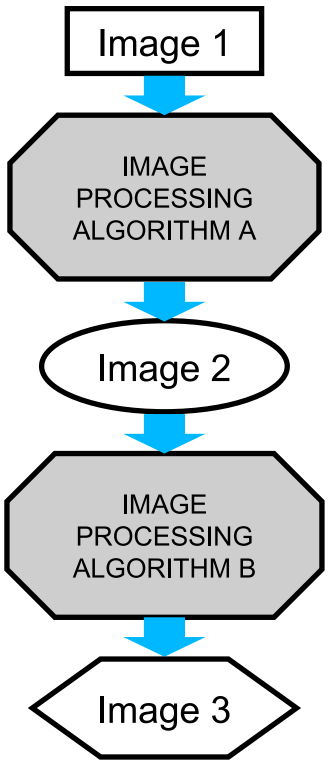


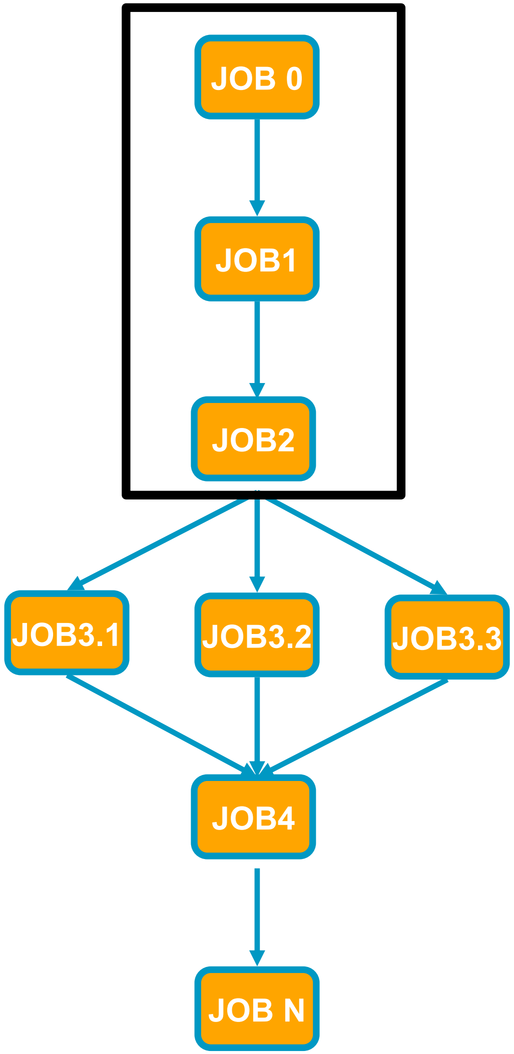
Types of Parallelism

Pipeline Parallelism

Output of One Job is the Input to the Next

- Each task can run in parallel
- Throughput impacted by the longest-latency element in the pipeline



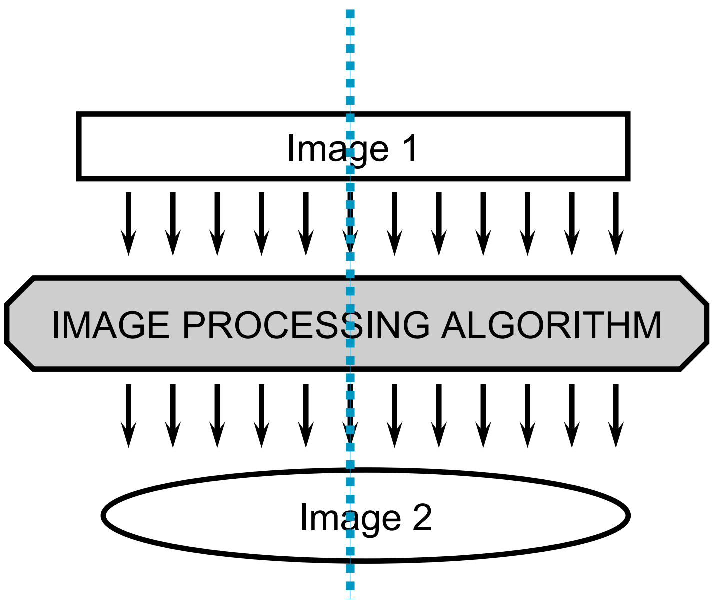


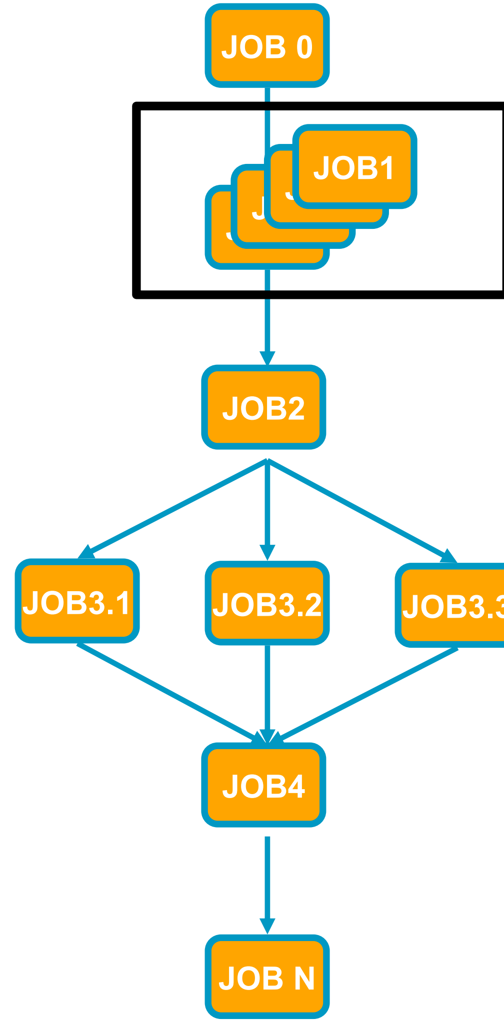
Types of Parallelism

Data Parallelism

Same Job Run on Different Data in Parallel

- Data partitioning across nodes
- Could require communication between task instances





Types of Parallelism

Tasks Decomposition for Data Parallelism

Dividing the work into multiple tasks

- Often, there are many valid decompositions for a given computation

Static vs. Dynamic

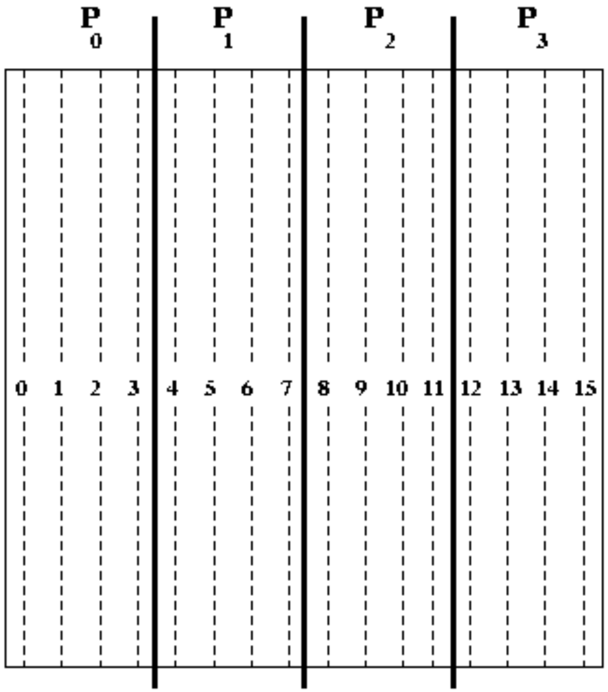
- Static: Decide the decomposition at the beginning of the program computation
- Dynamic: Decide the decomposition dynamically, based on the input characteristics

Types of Parallelism

Data Distribution May not Mean Load Balance

Balanced

- Matrix addition

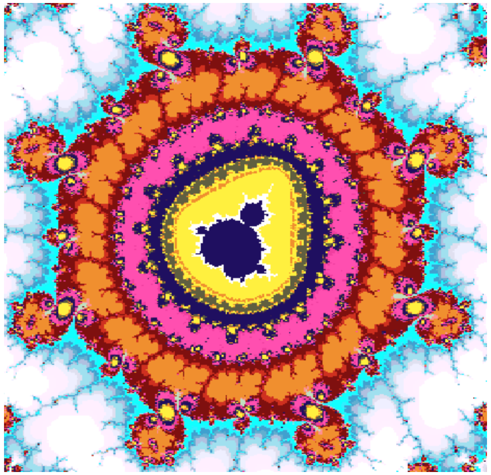


(a) Columnwise block stripping

Unbalanced

- Mandelbrot

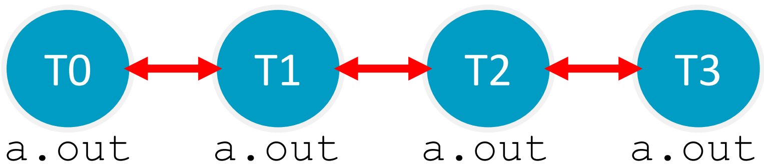
```
for m = 0 … 1023 do
  for n = 0 … 1023 do
    c = x+yj
    a = 0+0j
    while ((k<1000) and (abs(a)<2)) do
      a = a^2+c
      k = k+1
    end
    data(n,m) = k
  end
end
```



Parallel Execution Models

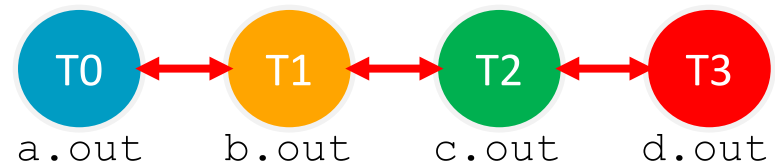
Parallel Execution Models

SPMD vs MPMD



Single Program – Multiple Data

- A single program executes on all tasks simultaneously, but at a single point in time, they may be executing the same or different instructions .
- Most common form of parallelism.
- Think of multiple processors running the same program at different points.



Multiple Program - Multiple Data

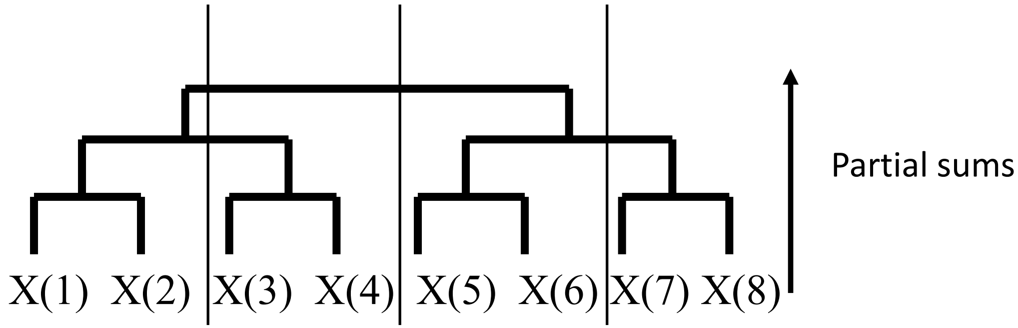
- Each task may be executing the same or different programs than other tasks

Parallel Execution Models

Parallel Algorithms

What Is it?

- A parallel algorithm is an algorithm that can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result.



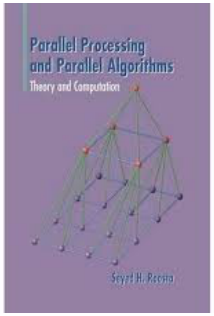
- Most of today's algorithms are sequential, that is, they specify a sequence of steps in which each step consists of a single operation
- Parallel algorithms usually exhibit a higher level parallelism at the cost of a higher number of operations

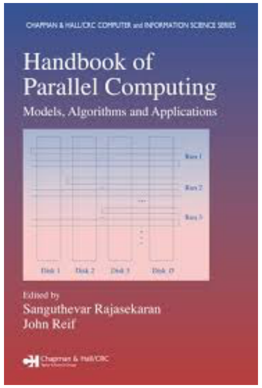
Parallel Execution Models

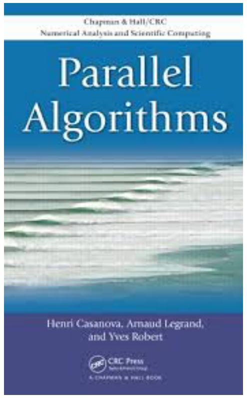
Parallel Algorithms

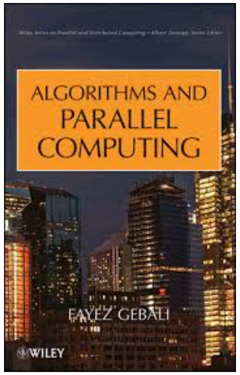
Examples of sequential algorithms?

Examples of parallel algorithms?









Next Steps

- Lab session this week:
  - Help with HWA
  - I.7 (Part 1) Installation of MPI in a single node
  - Linpack compilation (Performance Competition!)

- Get ready for next lecture:
  - A.5. Designing Parallel Programs

Questions

Application Parallelism