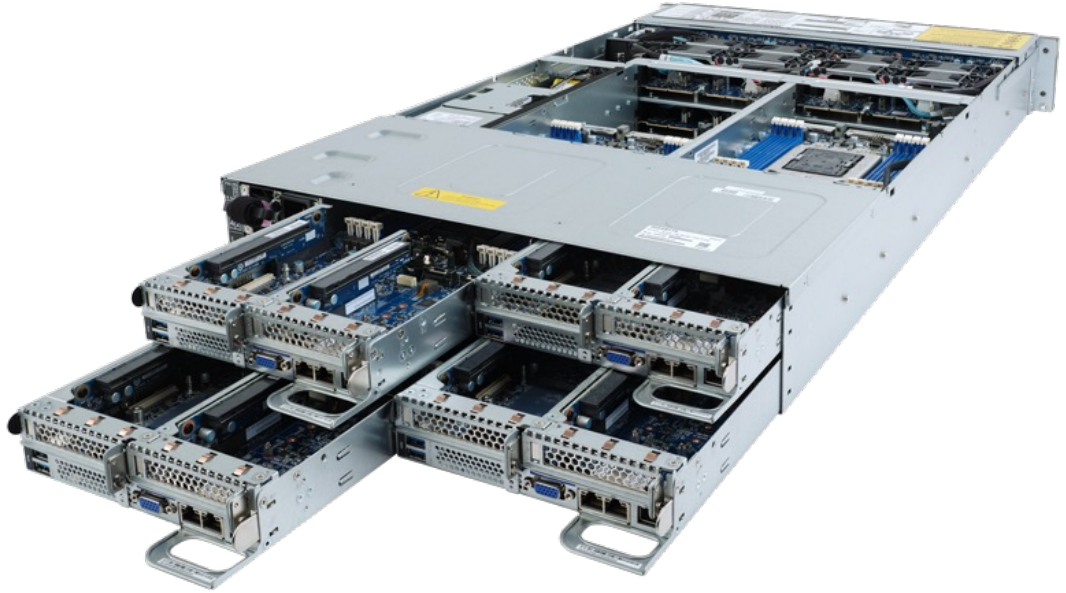
Intro to Parallel Computing on HPC

2022

Why Do We Care?

MacBook Pro 2021
Apple M1 Pro Chip
8-core CPU
16GB unified memory
512GB SSD
$1999

Penguin Altus XE2242
4in1 chassis. Each compute node has:
96 cores dual socket AMD EPYC 7642
512GB DDR4 3200MHz ECC memory
2TB SSD NVMe
$8000



Why Do We Care?

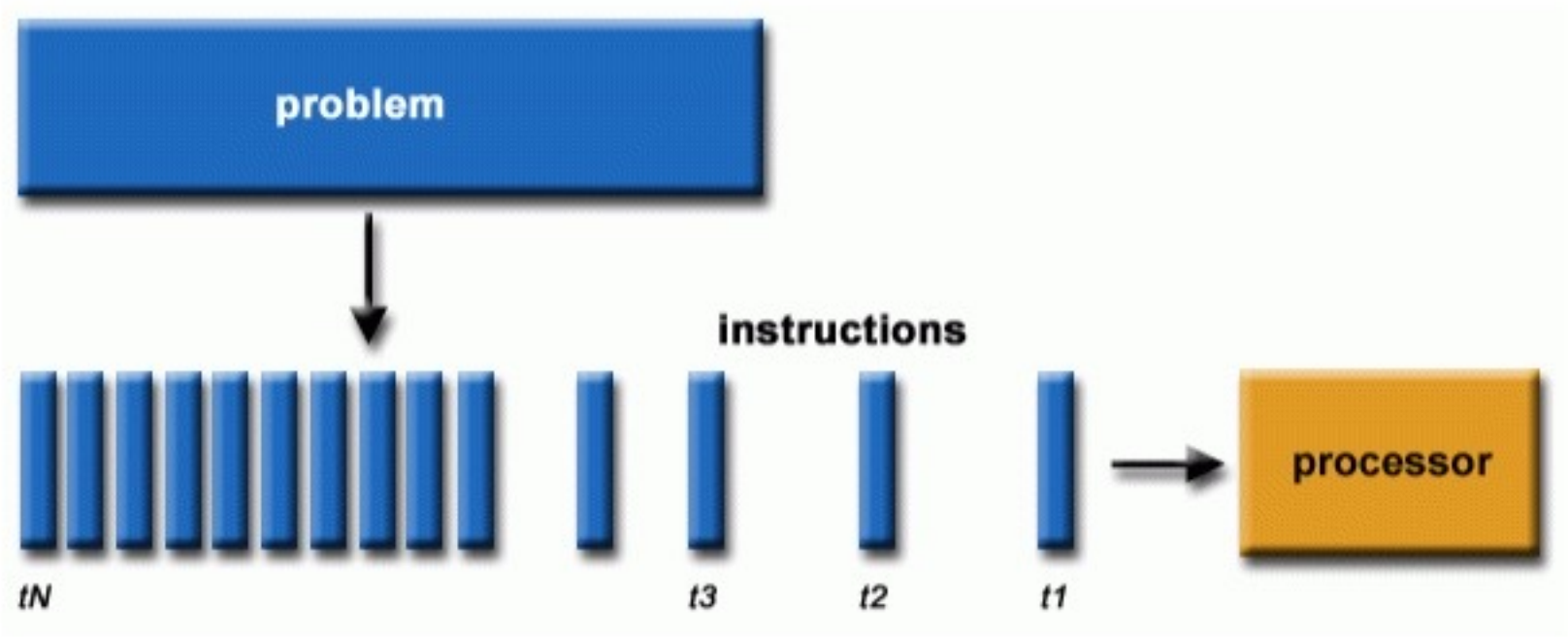
MacBook Pro 2021
Apple M1 Pro Chip
8-core CPU
16GB unified memory
512GB SSD
$1999

Puma Cluster
269 compute nodes
25,824 cores dual socket AMD EPYC 7642
137,728GB DDR4 3200MHz ECC memory
2PB SSD NVMe shared storage
538TB local storage
$2.7M

What is Parallel Computing

Serial Computing

A problem is broken into a discrete series of instructions
Instructions are executed sequentially one after another
Executed on a single processor
Only one instruction may execute at any moment in time



Serial computing generic example

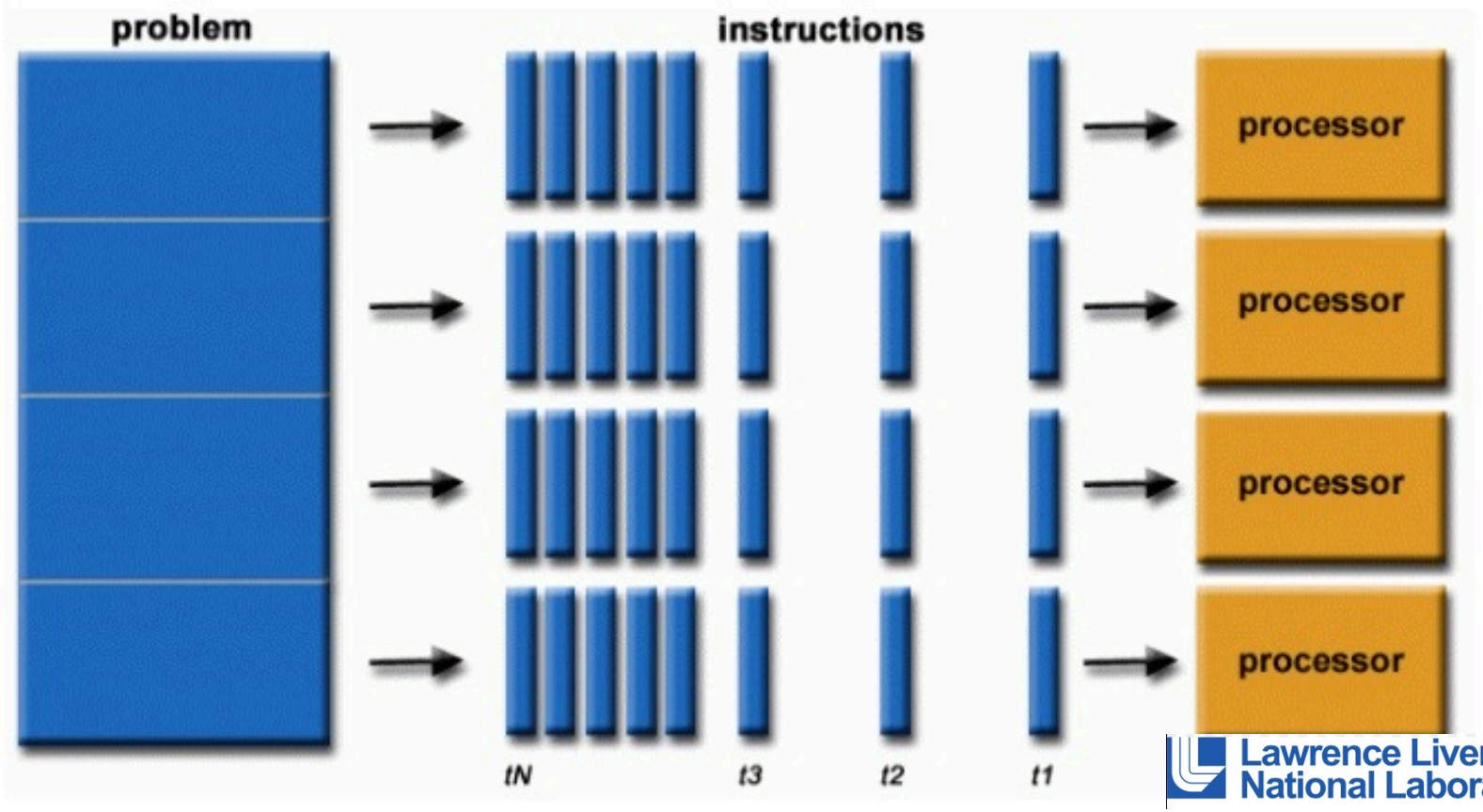
Lawrence Livermore National Laboratory

The University of Arizona

What is Parallel Computing

Parallel Computing

A problem is broken into discrete parts that can be solved concurrently
Each part is further broken down to a series of instructions
Instructions from each part execute simultaneously on different processors
An overall control/coordination mechanism is employed



Lawrence Livermore National Laboratory

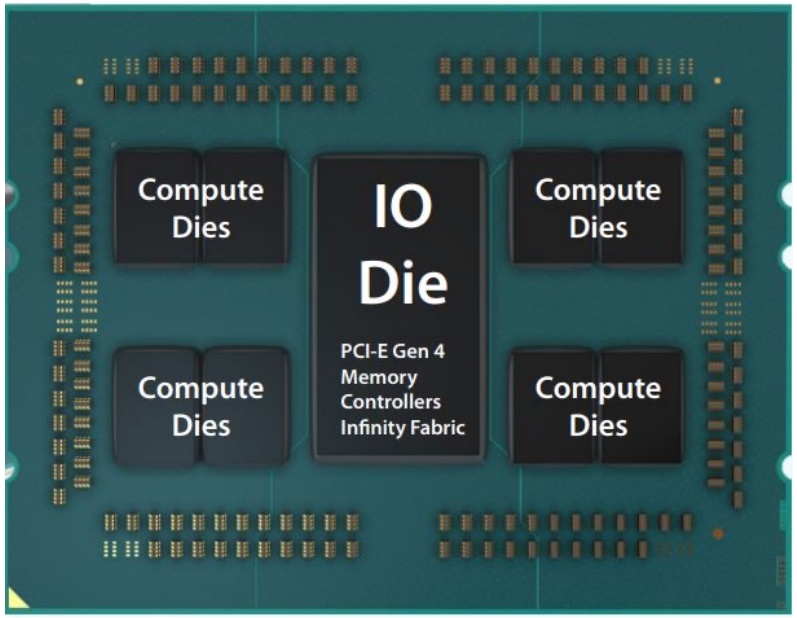
THE UNIVERSITY OF ARIZONA

What is Parallel Computing

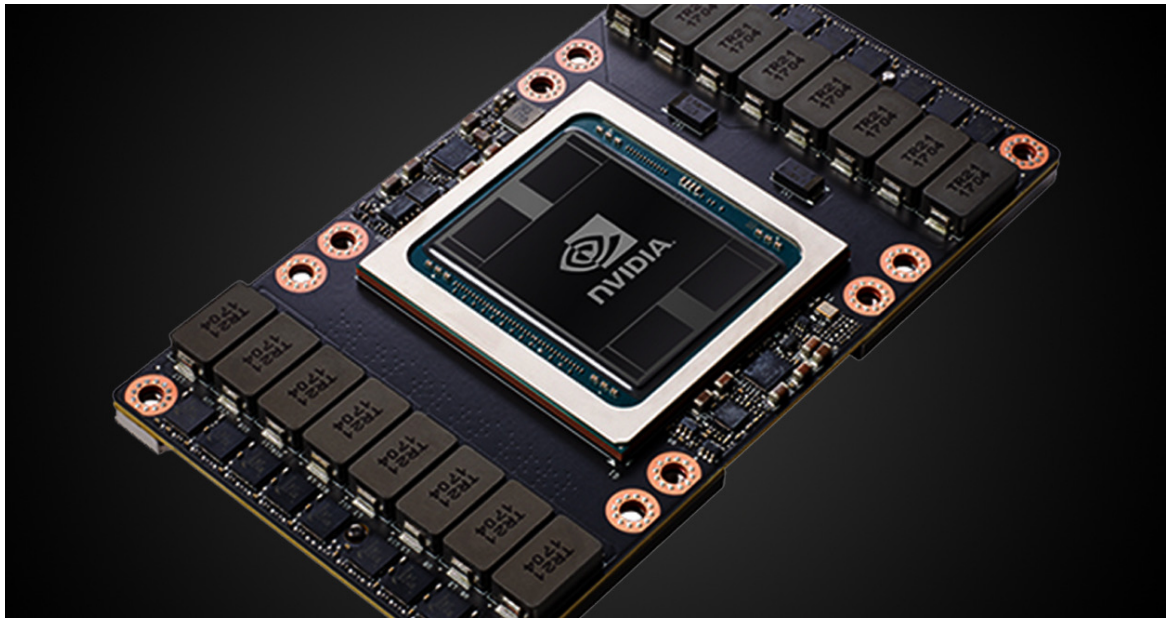
Parallel Computers

Virtually all stand-alone computers today are parallel from a hardware perspective:

- Multiple functional units (L1 cache, L2 cache, branch, prefetch, decode, floating-point, graphics processing (GPU), integer, etc.)
- Multiple execution units/cores
- Multiple hardware threads



AMD EPYC Rome CPU



Nvidia V100 GPU

What is Parallel Computing

Parallel Computers

Networks connect multiple stand-alone computers (nodes) to make larger parallel computer clusters.

Each compute node is a multi-processor parallel computer in itself

Multiple compute nodes are networked together with a high-speed network

Special purpose nodes, also multi-processor, are used for other purposes, with GPU's or extra memory

RACK 1 (STANDARD)		RACK 2 (STANDARD)		RACK 3 (STANDARD)		RACK 4 (STANDARD)		RACK 5 (GPU/MEM)	RACK 6 (NET/EXPANSION)	
4x460P9 PDU, 68.8kW MAX		4x460P9 PDU, 68.8kW MAX		4x460P9 PDU, 68.8kW MAX		4x460P9 PDU, 68.8kW MAX		4x460P9 PDU, 68.8kW MAX	4x460P9 PDU, 68.8kW MAX	
r1u42n1	r1u42n2	r2u42n1	r2u42n2	r3u42n1	r3u42n2	r4u42n1	r4u42n2	r5u42n1	cable management space	
r1u41n1	r1u41n2	r2u41n1	r2u41n2	r3u41n1	r3u41n2	r4u41n1	r4u41n2	r5u41n1	spine1	
r1u40n1	r1u40n2	r2u40n1	r2u40n2	r3u40n1	r3u40n2	r4u40n1	r4u40n2	r5u40n1	cable management space	
r1u39n1	r1u39n2	r2u39n1	r2u39n2	r3u39n1	r3u39n2	r4u39n1	r4u39n2	r5u39n1	spine2	
r1u38n1	r1u38n2	r2u38n1	r2u38n2	r3u38n1	r3u38n2	r4u38n1	r4u38n2		cable management space	
r1u37n1	r1u37n2	r2u37n1	r2u37n2	r3u37n1	r3u37n2	r4u37n1	r4u37n2	r5u37n1	spine3	
r1u36n1	r1u36n2	r2u36n1	r2u36n2	r3u36n1	r3u36n2	r4u36n1	r4u36n2		cable management space	
r1u35n1	r1u35n2	r2u35n1	r2u35n2	r3u35n1	r3u35n2	r4u35n1	r4u35n2	r5u35n1	spine4	
r1u34n1	r1u34n2	r2u34n1	r2u34n2	r3u34n1	r3u34n2	r4u34n1	r4u34n2		cable management space	
r1u33n1	r1u33n2	r2u33n1	r2u33n2	r3u33n1	r3u33n2	r4u33n1	r4u33n2	r5u33n1	edge1	
r1u32n1	r1u32n2	r2u32n1	r2u32n2	r3u32n1	r3u32n2	r4u32n1	r4u32n2		cable management space	
r1u31n1	r1u31n2	r2u31n1	r2u31n2	r3u31n1	r3u31n2	r4u31n1	r4u31n2	r5u31n1	mgmt	
r1u30n1	r1u30n2	r2u30n1	r2u30n2	r3u30n1	r3u30n2	r4u30n1	r4u30n2		switch mgmt gigabit switch	
r1u29n1	r1u29n2	r2u29n1	r2u29n2	r3u29n1	r3u29n2	r4u29n1	r4u29n2	r5u29n1	cable management space	
r1u28n1	r1u28n2	r2u28n1	r2u28n2	r3u28n1	r3u28n2	r4u28n1	r4u28n2		r6u28n1	r6u28n2
r1u27n1	r1u27n2	r2u27n1	r2u27n2	r3u27n1	r3u27n2	r4u27n1	r4u27n2	r5u27n1	r6u27n1	r6u27n2
r1u26n1	r1u26n2	r2u26n1	r2u26n2	r3u26n1	r3u26n2	r4u26n1	r4u26n2		r6u26n1	r6u26n2
r1u25n1	r1u25n2	r2u25n1	r2u25n2	r3u25n1	r3u25n2	r4u25n1	r4u25n2	r5u25n1	r6u25n1	r6u25n2
r1m2		r2m2		r3m2		r4m2		r5u24n1	r6u24n1	r6u24n2
cable management space		cable management space		cable management space		cable management space		cable management space	r6u23n1	r6u23n2
r1l2		r2l2		r3l2		r4l2		r5l1	r6u22n1	r6u22n2
r1l1		r2l1		r3l1		r4l1		r5m1	r6u21n1	r6u21n2
cable management space		cable management space		cable management space		cable management space		cable management space	r6u20n1	r6u20n2
r1m1		r2m1		r3m1		r4m1		r5u19n1	r6u19n1	r6u19n2
r1u18n1	r1u18n2	r2u18n1	r2u18n2	r3u18n1	r3u18n2	r4u18n1	r4u18n2		r6u18n1	r6u18n2
r1u17n1	r1u17n2	r2u17n1	r2u17n2	r3u17n1	r3u17n2	r4u17n1	r4u17n2	r5u17n1	cable management space	
r1u16n1	r1u16n2	r2u16n1	r2u16n2	r3u16n1	r3u16n2	r4u16n1	r4u16n2		r6l1	
r1u15n1	r1u15n2	r2u15n1	r2u15n2	r3u15n1	r3u15n2	r4u15n1	r4u15n2	r5u15n1	r6m1	
r1u14n1	r1u14n2	r2u14n1	r2u14n2	r3u14n1	r3u14n2	r4u14n1	r4u14n2		cable management space	
r1u13n1	r1u13n2	r2u13n1	r2u13n2	r3u13n1	r3u13n2	r4u13n1	r4u13n2	r5u13n1	r6u13n1	r6u13n2
r1u12n1	r1u12n2	r2u12n1	r2u12n2	r3u12n1	r3u12n2	r4u12n1	r4u12n2		r6u12n1	r6u12n2
r1u11n1	r1u11n2	r2u11n1	r2u11n2	r3u11n1	r3u11n2	r4u11n1	r4u11n2	r5u11n1	r6u11n1	r6u11n2
r1u10n1	r1u10n2	r2u10n1	r2u10n2	r3u10n1	r3u10n2	r4u10n1	r4u10n2		r6u10n1	r6u10n2
r1u09n1	r1u09n2	r2u09n1	r2u09n2	r3u09n1	r3u09n2	r4u09n1	r4u09n2	r5u09n1	r6u09n1	r6u09n2
r1u08n1	r1u08n2	r2u08n1	r2u08n2	r3u08n1	r3u08n2	r4u08n1	r4u08n2	r5u08n1	r6u08n1	r6u08n2
r1u07n1	r1u07n2	r2u07n1	r2u07n2	r3u07n1	r3u07n2	r4u07n1	r4u07n2	r5u07n1	r6u07n1	r6u07n2
r1u06n1	r1u06n2	r2u06n1	r2u06n2	r3u06n1	r3u06n2	r4u06n1	r4u06n2	r5u06n1	r6u06n1	r6u06n2
r1u05n1	r1u05n2	r2u05n1	r2u05n2	r3u05n1	r3u05n2	r4u05n1	r4u05n2	r5u05n1	r6u05n1	r6u05n2
r1u04n1	r1u04n2	r2u04n1	r2u04n2	r3u04n1	r3u04n2	r4u04n1	r4u04n2	r5u04n1	r6u04n1	r6u04n2
r1u03n1	r1u03n2	r2u03n1	r2u03n2	r3u03n1	r3u03n2	r4u03n1	r4u03n2	r5u03n1	r6u03n1	r6u03n2
								r5u02n1	r6u02n1	
64	nodes	64	nodes	64	nodes	64	nodes	16	40	
6144	cores	6144	cores	6144	cores	6144	cores	1536	3840	

Puma Rack layout

Lawrence Livermore National Laboratory

THE UNIVERSITY OF ARIZONA

Why Use Parallel Computing

Parallel Computers ..

Reduce Complexity

In the natural world, many complex, interrelated events are happening at the same time, yet within a temporal sequence.

Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real-world phenomena.

Example: Natural Language Processing models have billions of parameters

Galaxy Formation

Planetary Movments

Climate Change

Real world phenomena can be simulated with parallel computing

Rush Hour Traffic

Plate Tectonics

Weather

Lawrence Livermore National Laboratory

THE UNIVERSITY OF ARIZONA

Why Use Parallel Computing

Parallel Computers ..

SAVE TIME
In theory, throwing more resources at a task will shorten its time to completion, with shorter time to results.
Parallel computers can be built from cheap(ish), commodity components.

Working in parallel shortens completion time

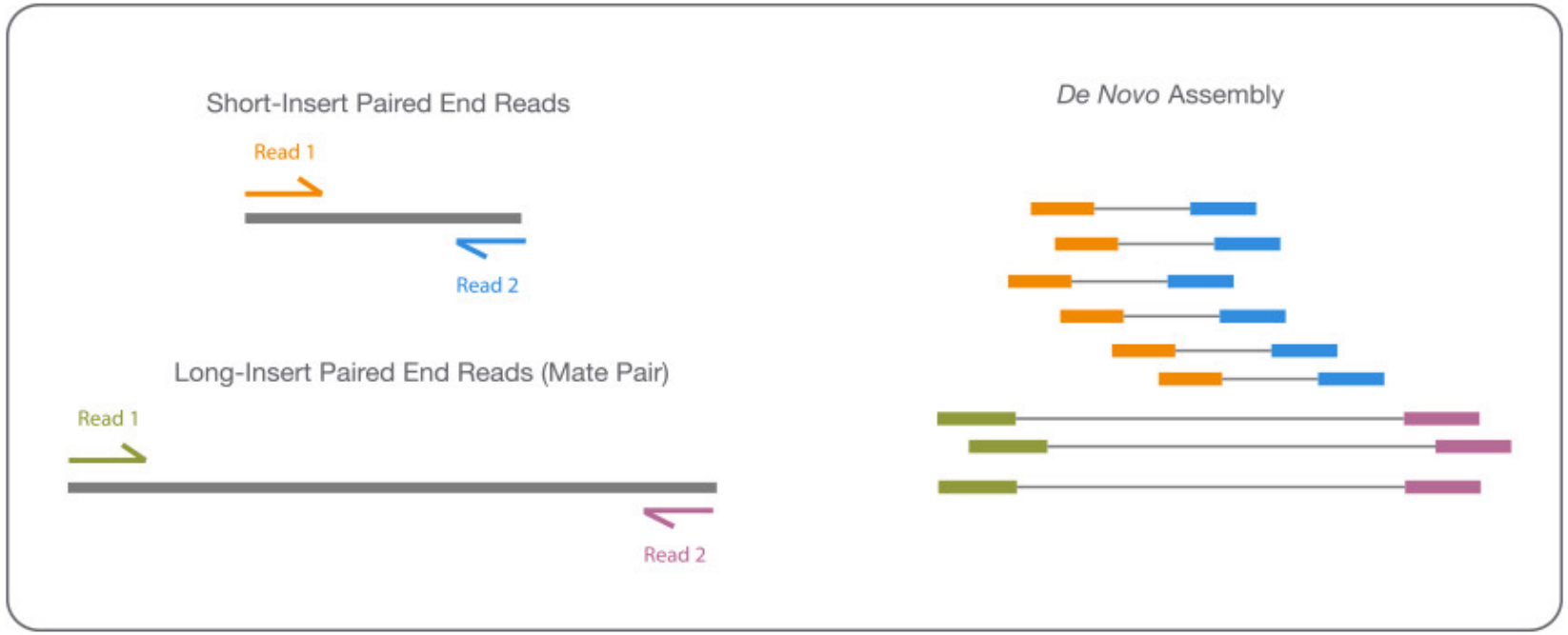
Why Use Parallel Computing

Parallel Computers ..

PROVIDE CONCURRENCY

A single compute resource can only do one thing at a time. Multiple compute resources can do many things simultaneously.

Example: The so-called shotgun sequencing technique generates the sequences of many thousands of small DNA fragments.



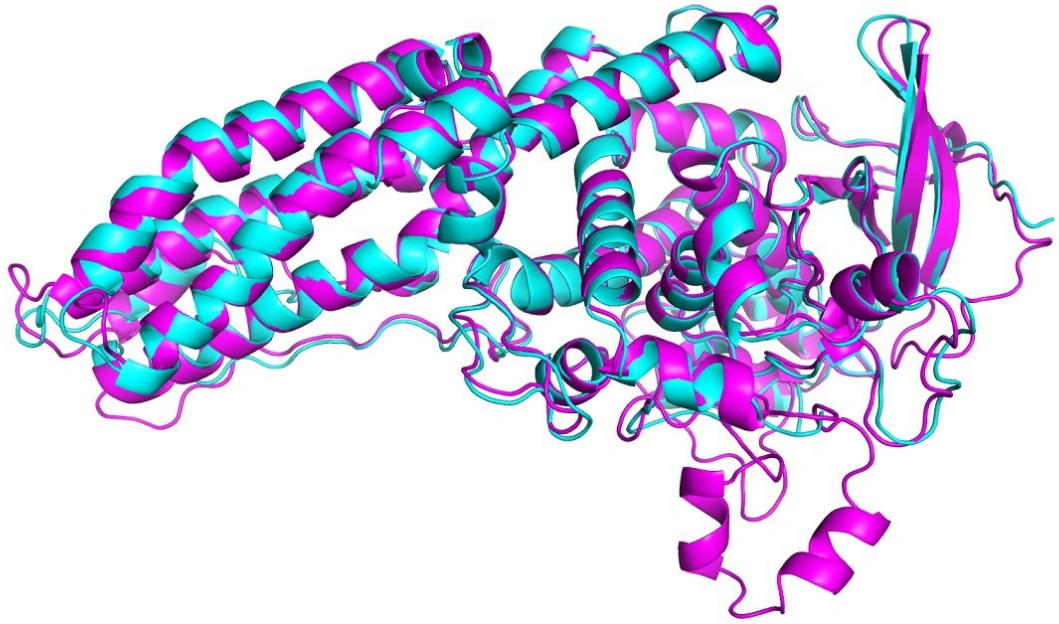
Why Use Parallel Computing

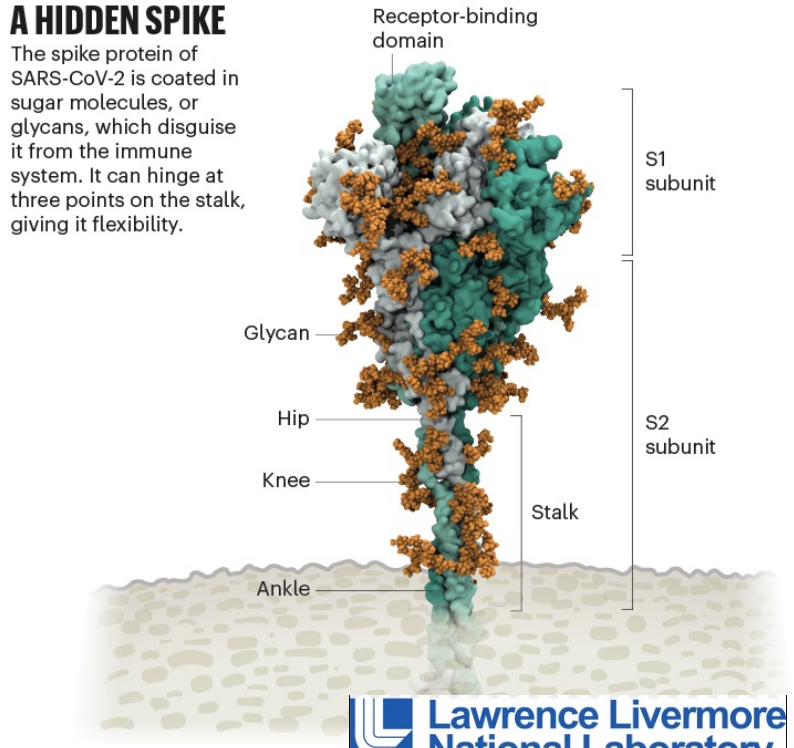
Parallel Computers ..

TAKE ADVANTAGE OF NON-LOCAL RESOURCES

Using compute resources on a wide area network, or even the Internet when local compute resources are scarce or insufficient.

Example: Folding@Home is a distributed computing project to simulate protein dynamics. Supercomputers at the three public Arizona universities have contributed the 46th most credits to the project at 55 billion





Lawrence Livermore National Laboratory

THE UNIVERSITY OF ARIZONA

Parallel Computing Terminology

Node – an individual computer. Many of them together comprise a supercomputer

CPU – aka socket or processor. A physical device mounted on the motherboard. Puma nodes have two CPU's

Core – capable of conducting independent work.
Puma CPU's have 48 cores for a total of 96
However, in Slurm, cpu = core

Process – instance of a program, with access to its own memory, state and file descriptors

Task – a logically discrete section of computational work.
By default, Slurm allocates one cpu / task

Shared Memory – all cores have direct access to common physical memory. Run on a single node.

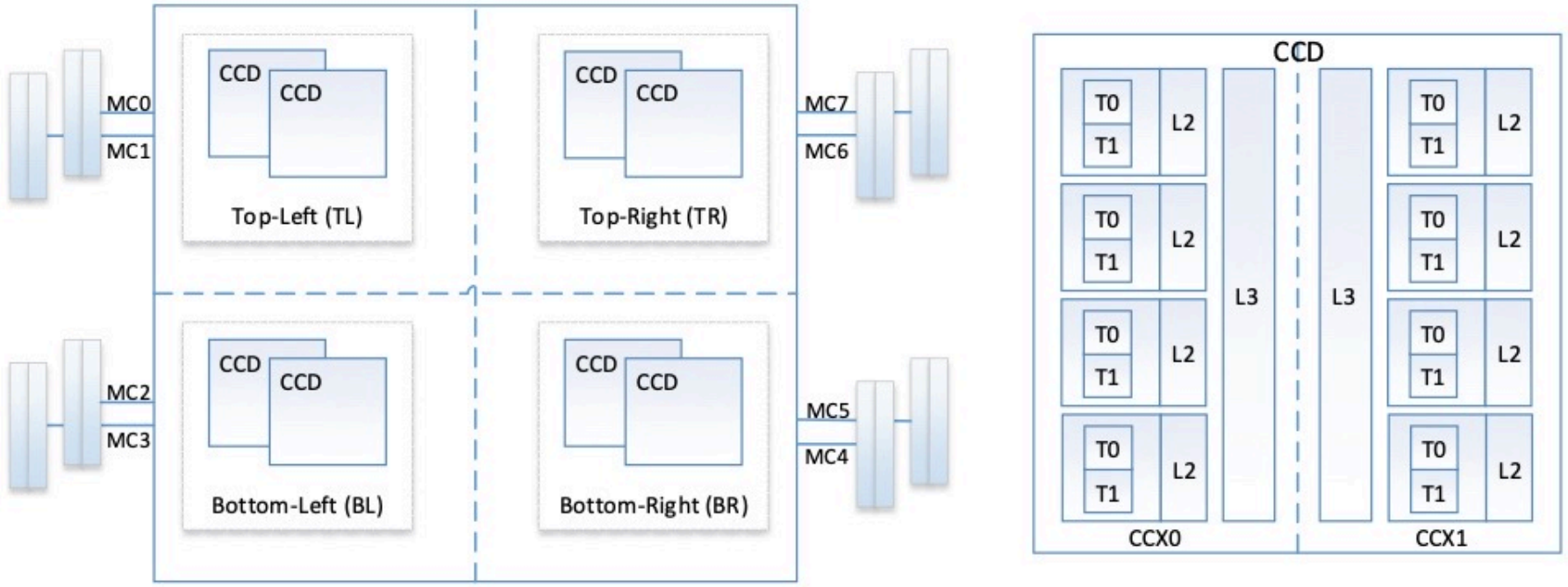
Threads – highest level of code executed by a processor. Each process has at least a single thread
Not the same as hyperthreading which is turned off on all compute nodes.

Parallel Computing Terminology

NUMA – Non-Uniform Memory Access. Global address space shared by all cores. Memory is local to each processor or remote, which is slower. Our AMD CPUs have 4 quadrants and we set NPS1 meaning one NUMA node per socket.

Cache – memory that is much faster but smaller and expensive. L1 is on the core, L2 is next to each core and L3 is shared between 4 cores.
Cache Coherence – a write to the cache of one processor is not seen by the others

AMD Rome Core Complex



Parallel Computing Terminology

Distributed Memory – connecting compute nodes with a high-speed network to support large memory or large core workloads (Message passing between nodes)

High Throughput Computing – compute that runs on one node, and in many cases, on one core

Data Parallel Model – also referred to as the Partitioned Global Address Space (PGAS) model

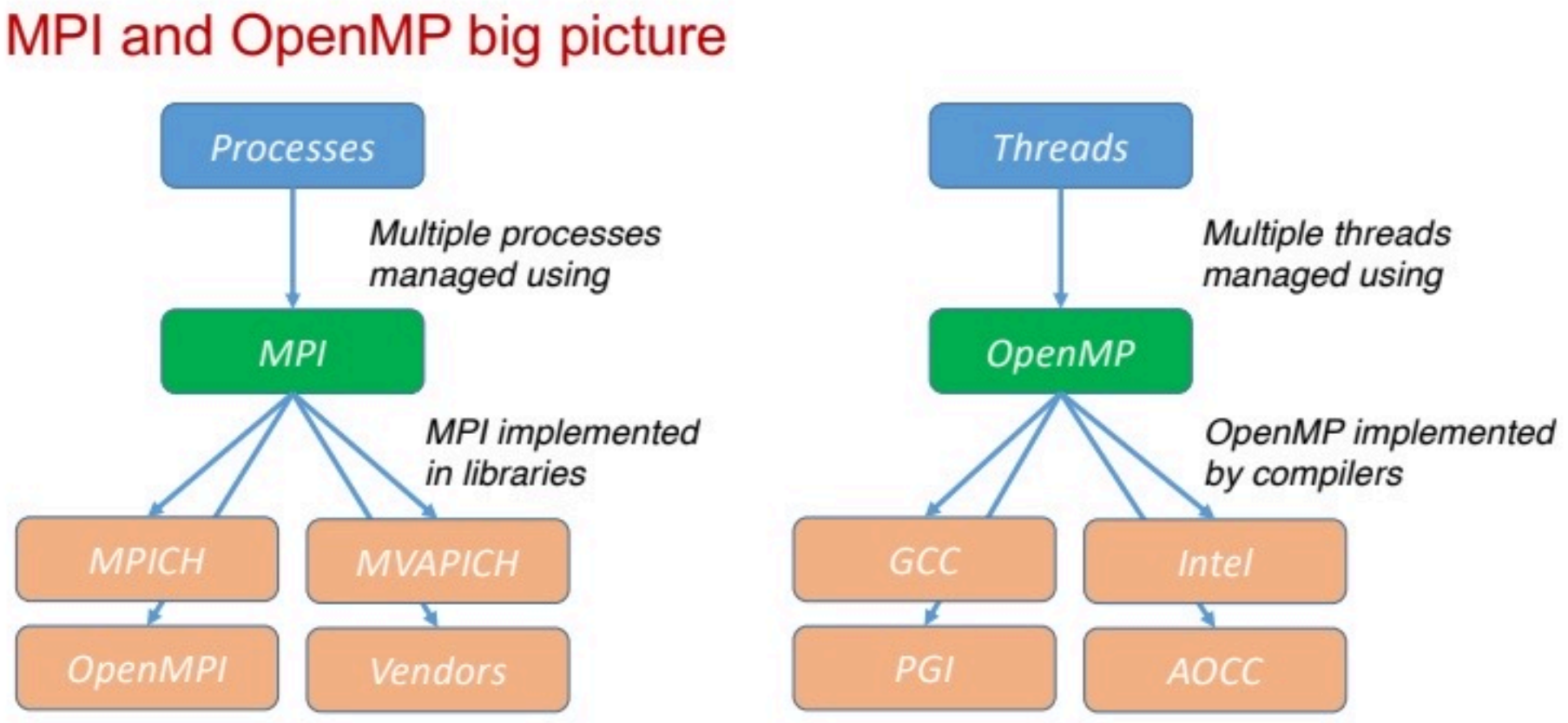
Massively Parallel – workloads that use many hundreds or thousands of cores

Embarrassingly Parallel – solving many similar but independent tasks with little or no task coordination.

MPI – Message Passing Interface standard which defines multi-node communication. Implementations include OpenMPI *, Intel MPI *, MPICH, MVAPICH

* We encourage these on Puma and Ocelote

Parallel Computing Terminology



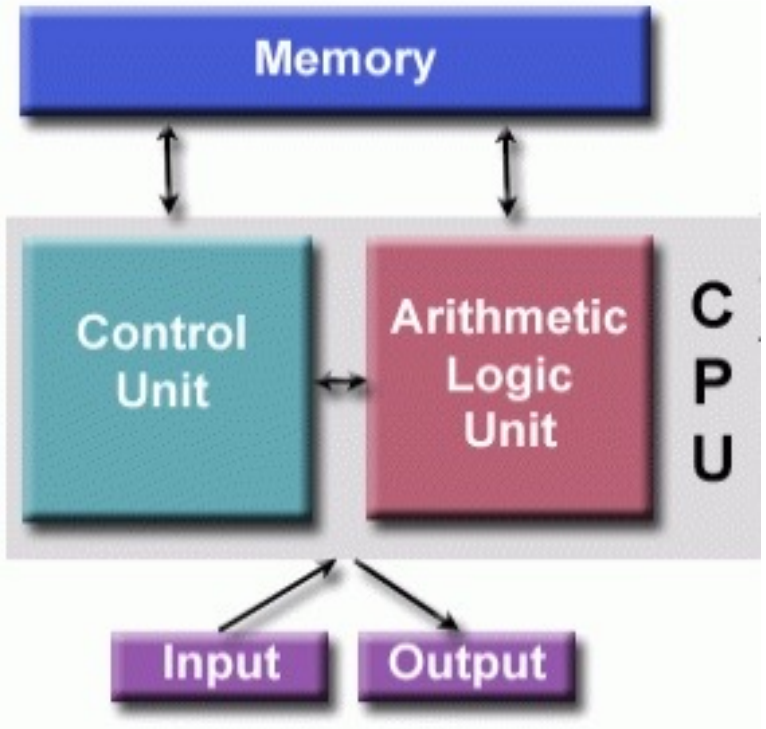
MPI is a standard for parallelizing C, C++ and Fortran code to run on distributed memory (multiple compute node) systems.

OpenMP is an application programming interface (API) for shared-memory (within a node) parallel programming in C, C++ and Fortran

Parallel Computing Theory

von Neumann Computer Architecture

Named after the Hungarian mathematician John von Neumann who first authored the general requirements for an electronic computer in his 1945 papers.
Also known as "stored-program computer" - both program instructions and data are kept in electronic memory. Differs from earlier computers which were programmed through "hard wiring".
Since then, virtually all computers have followed this basic design:



John von Neumann circa 1940s
(Source: LANL archives)

Lawrence Livermore National Laboratory

The University of Arizona

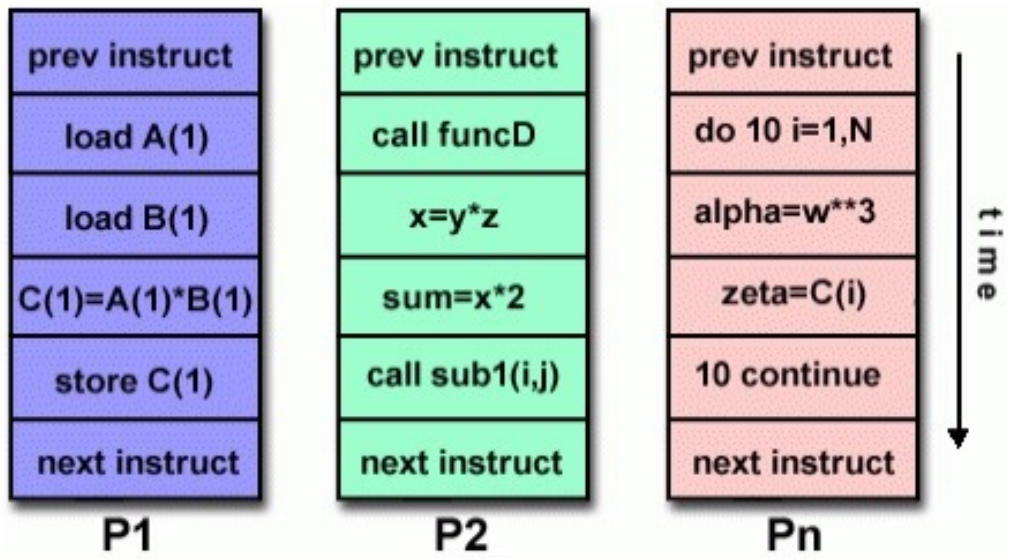
Parallel Computing Theory

Flynn’s Classical Taxonomy

There are a number of different ways to classify parallel computers.
One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.
Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of ***Instruction Stream*** and ***Data Stream***. Each of these dimensions can have only one of two possible states: ***Single*** or ***Multiple***.
The matrix below defines the 4 possible classifications according to Flynn:

S I S D Single Instruction stream Single Data stream	**S I M D** Single Instruction stream Multiple Data stream
M I S D Multiple Instruction stream Single Data stream	**M I M D** Multiple Instruction stream Multiple Data stream

MIMD : Puma



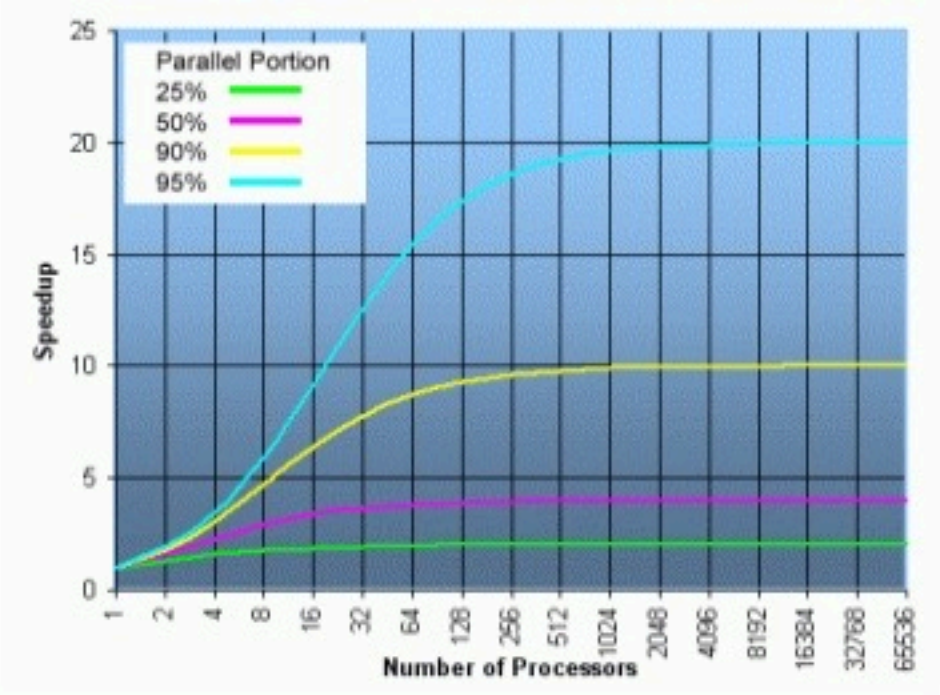
Lawrence Livermore National Laboratory

THE UNIVERSITY OF ARIZONA

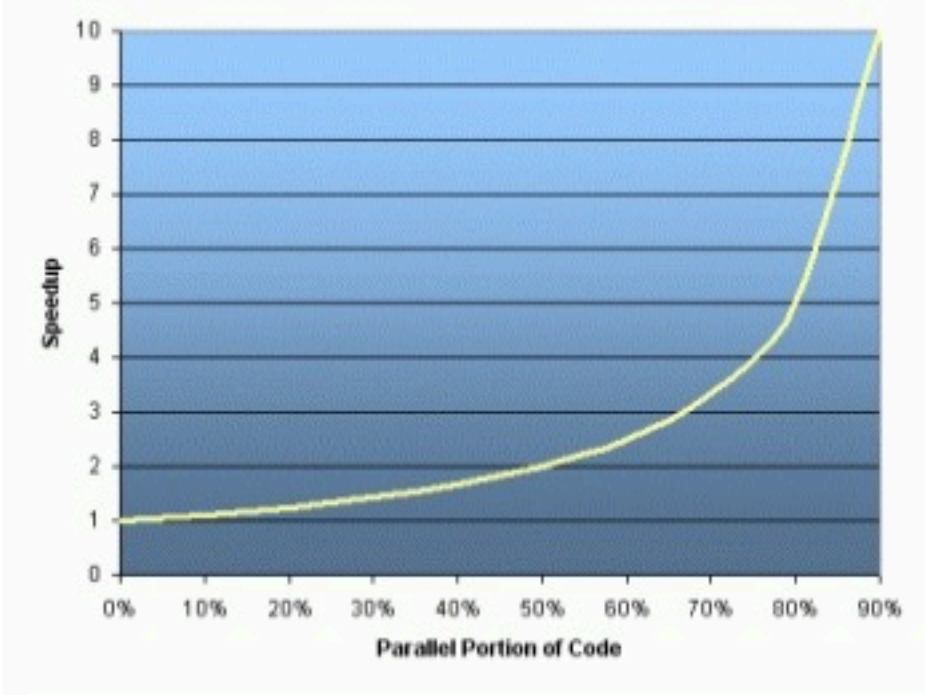
Parallel Computing Theory

Amdahl's Law

Amdahl's Law states that potential program speedup is defined by the fraction of code (P) that can be parallelized.



Speedup when introducing more processors



Amdahl's law

Lawrence Livermore National Laboratory

THE UNIVERSITY OF ARIZONA

Parallel Computing Theory

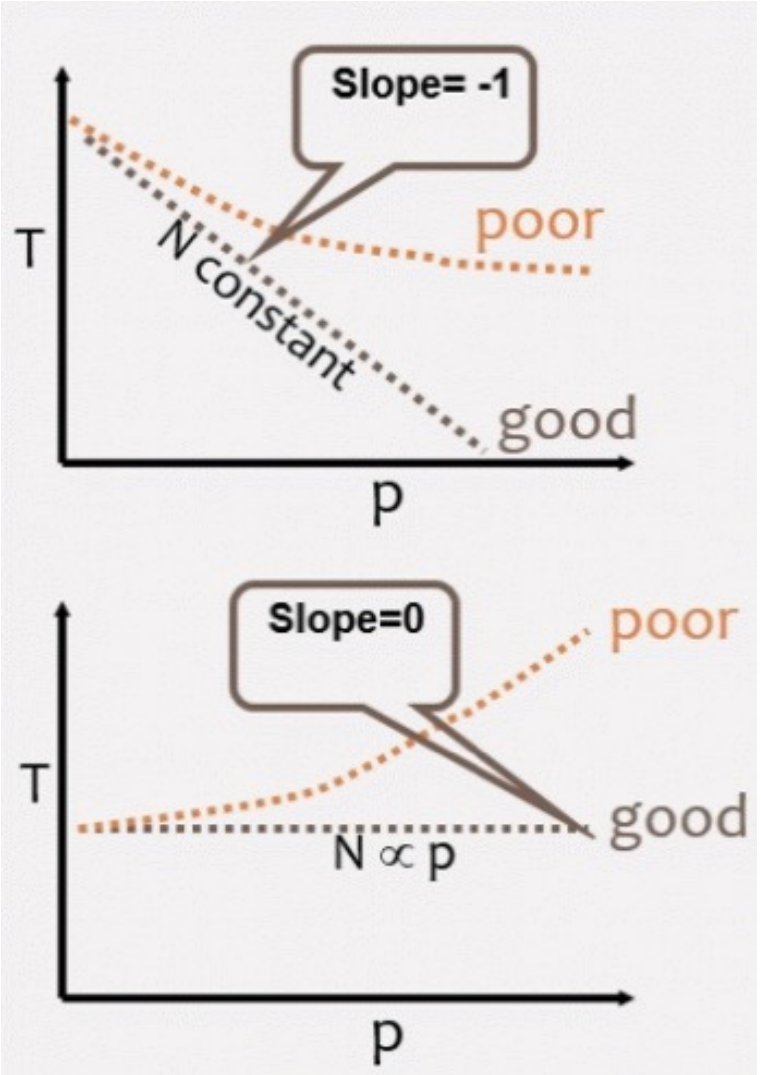
Scalability

Strong scaling (Amdahl):

The total problem size stays fixed as more processors are added.
Goal is to run the same problem size faster
Perfect scaling means problem is solved in 1/P time (compared to serial)

Weak scaling (Gustafson):

The problem size *per processor* stays fixed as more processors are added. The total problem size is proportional to the number of processors used.
Goal is to run larger problem in same amount of time
Perfect scaling means problem Px runs in same time as single processor run

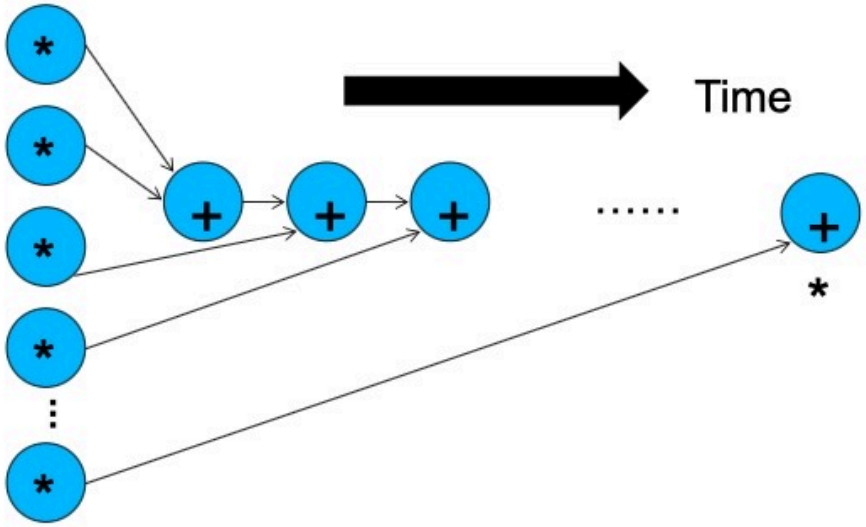


Parallel Computing Theory

Serialized code

A naïve inner product algorithm of two vectors of one million elements each

- All multiplications can be done in one time unit (parallel)
- Additions to a single accumulator in one million time units (serial)



Amdahl's Law

- If fraction X of a computation is serialized, the speedup cannot be more than 1/X

In this example, X = 50%

- Half the calculations are serialized
- No more than 2X speedup is possible no matter how many cores are used

Parallel Computing Theory

Load Balance

The total amount of time to complete a parallel job is limited by the thread that takes the longest to finish

Assume that a job takes 100 units of time for one person to finish

- If we break up the job into 10 parts of 10 units each and have 10 people to do it in parallel, we can get a 10X speedup
- If we break up the job into 50, 10, 5, 5, 5, 5, 5, 5, 5, 5 units, the same 10 people will take 50 units to finish, with 9 of them idling for most of the time. We will get no more than 2X speedup.

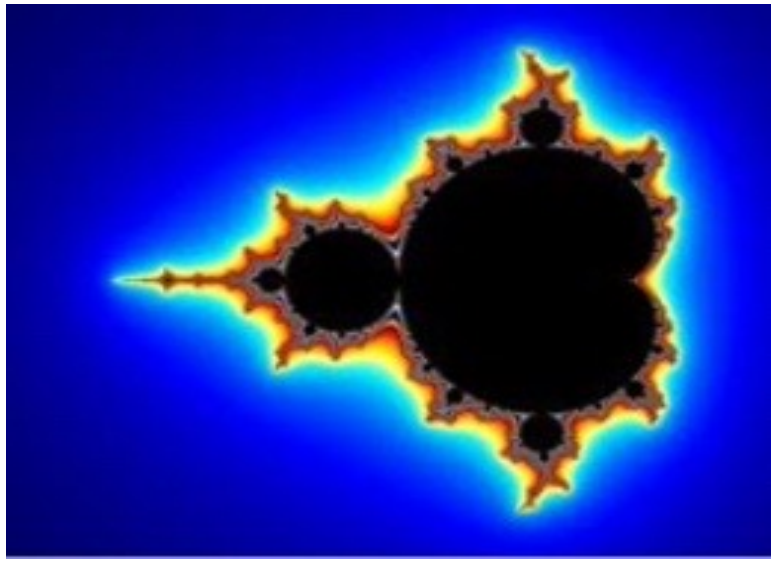
Parallel Computing Theory

Load Imbalance

Caused by non-uniform data distributions
- sparse datasets
- highly concentrated spatial data areas

Occurs in astronomy, medical imaging, rendering

If each thread processes the input data of a given spatial volume unit, some will do a lot more work than others



Parallel Computing Terminology

MPI Implementations

Julia There is a Julia language wrapper for MPI

MATLAB has its own parallel extension library implemented using MPI and PVM

Python Implementations of MPI include pyMPI, mpi4py, pypar, and MYMPI
The Boost C++ Libraries acquired Boost:MPI which include the MPI Python Bindings.

R Bindings of MPI include Rmpi and pbdMPI.

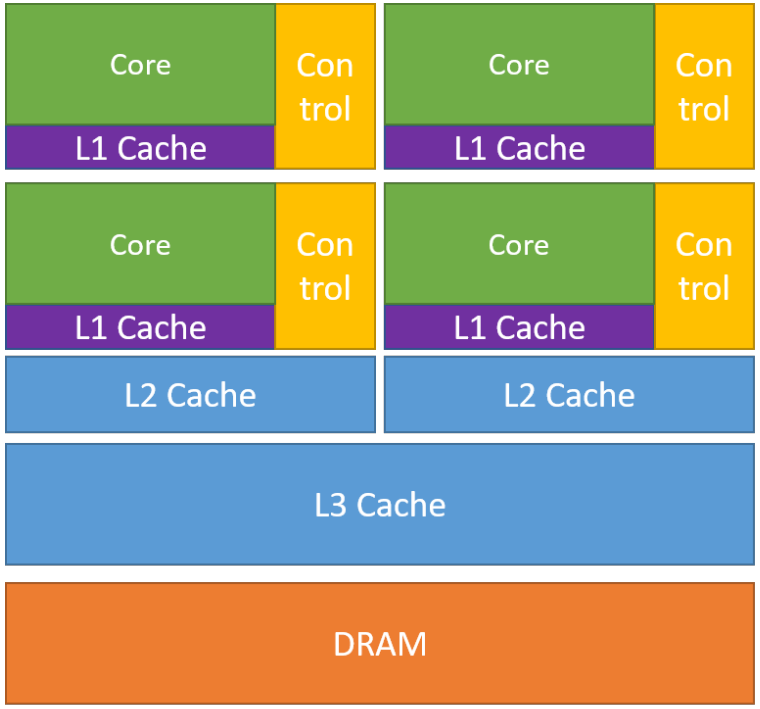
On HPC we support OpenMPI and Intel MPI. OpenMPI is a default module that is loaded with GCC 8.3
Intel MPI is provided when you unload OpenMPI and GCC, and then load the Intel compiler.
By default, modules on HPC are compiled with OpenMPI

Parallel Computing CPU vs GPU

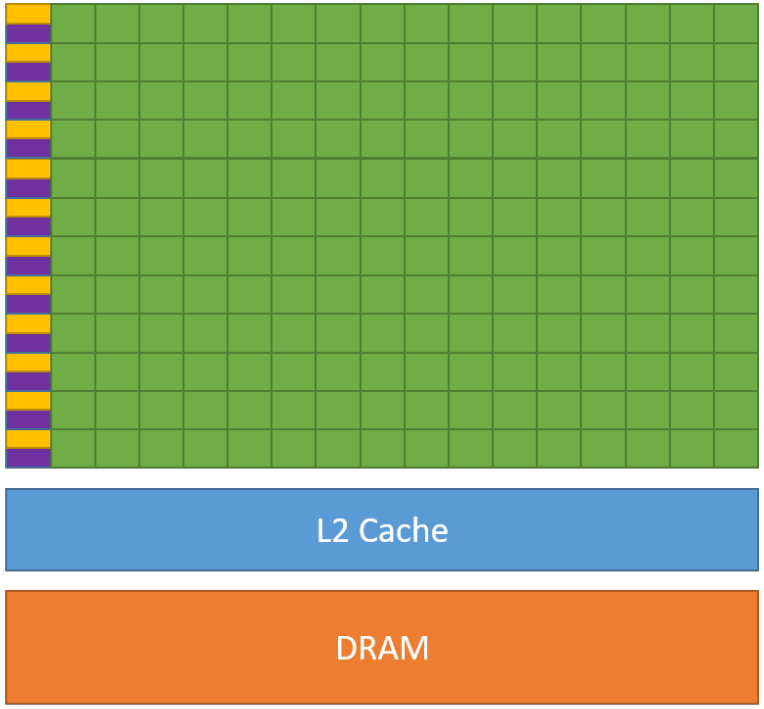
GPUs were made for parallel computing with many more, less powerful cores

- Large caches
- Sophisticated control
- Powerful ALU

- Small cache
- Simple control
- Many Energy efficient ALUs



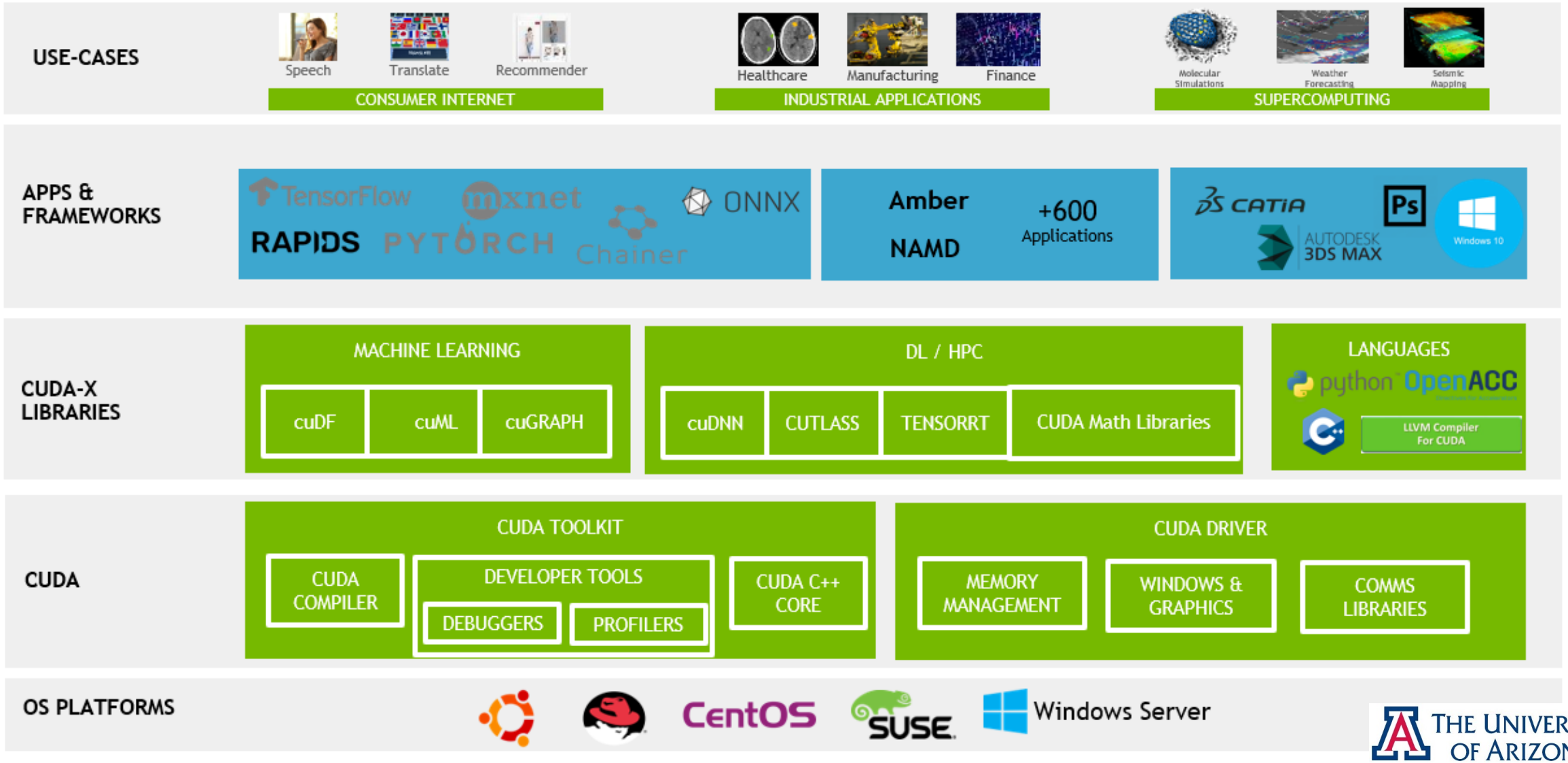
CPU



GPU

Parallel Computing GPU

Nvidia has an elaborate and growing ecosystem based on CUDA which provides the parallel support

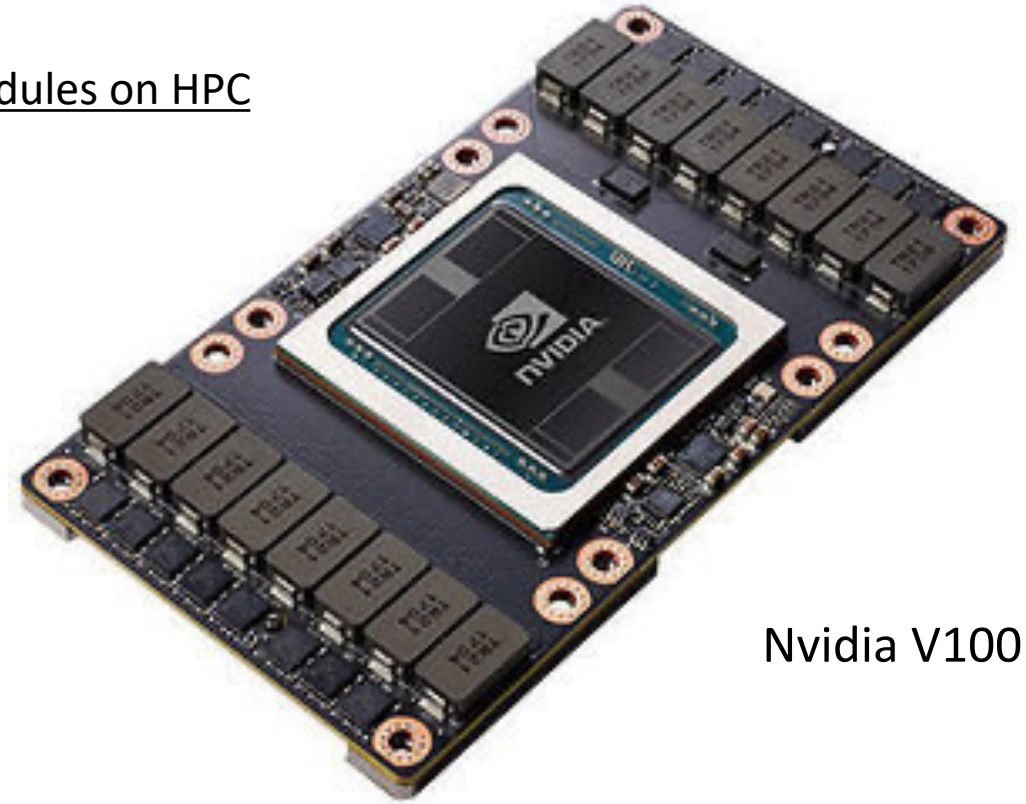


Parallel Computing GPU

- CPUs for **sequential** code where latency matters
- GPU's can be 20X or more times faster for **parallel** code

<u>Most of these applications are installed as modules on HPC</u>

Tensorflow
PyTorch
Matlab
NAMD
LAMMPS
Quantum ESPRESSO
Gromacs
Relion
Nvidia RAPIDS
Julia
Folding@home
Caffe2
Schrodinger



Nvidia V100

Parallel Programming

This parallel algorithm to calculate part of the Fibonacci series ..

$$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^{-n}}{2\varphi - 1}.$$

where

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1.61803\,39887\ldots$$

.. is beyond the scope of this workshop.
A more extensive overview of parallel programming can be found at:
https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

Performance Analysis and Tuning

Installed as a module

HPCToolkit/
hpctoolkit

HPCToolkit performance tools: measurement and analysis components

17 Contributors | 109 Issues | 2 Discussions | 292 Stars | 51 Forks

An integrated suite of tools for measurement and analysis of program performance

Installed in operating system

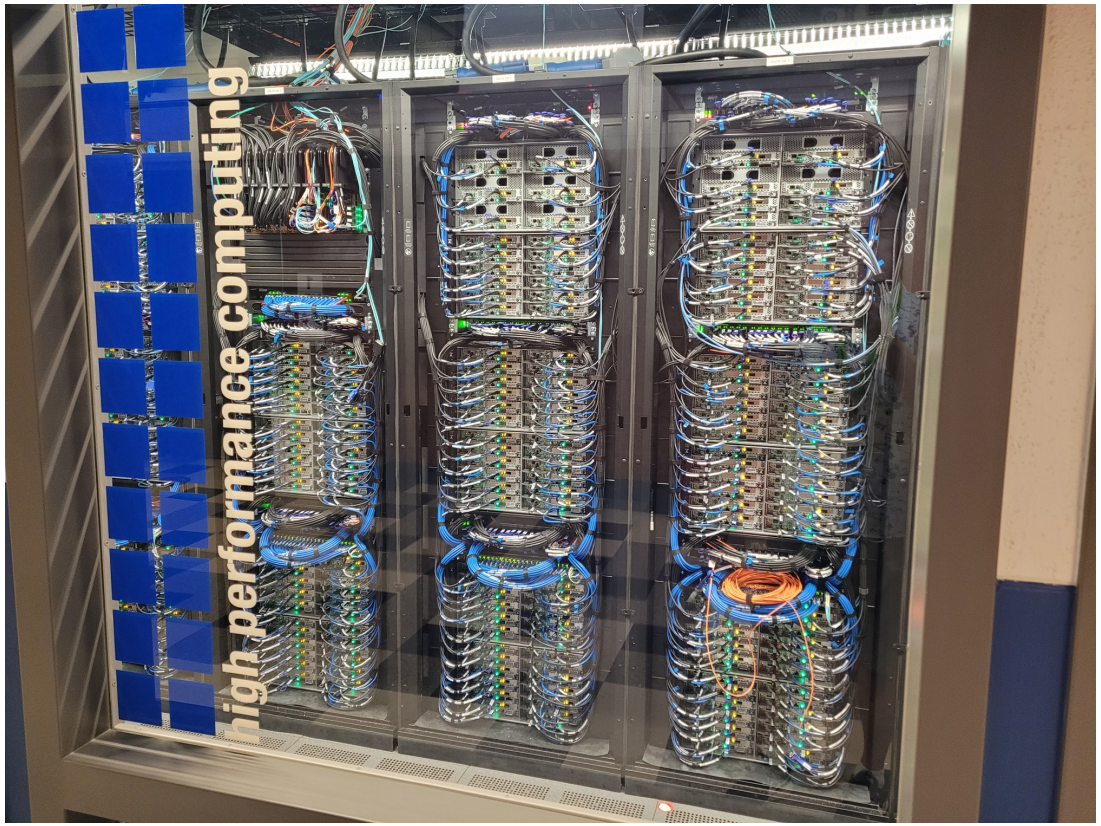
Tools that can automatically detect many memory management and threading bugs, and profile your programs in detail.

Parallel Computing on HPC

This example Slurm script runs the Hello World executable on 10 cores on each of 3 nodes

```
#!/bin/bash
#SBATCH --job-name=Multi-Node-MPI-Job
#SBATCH --ntasks=30
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=10
#SBATCH --time=00:01:00
#SBATCH --partition=standard
#SBATCH --account=YOUR_GROUP

module load gnu8 openmpi3
mpicc -o hello_world hello_world.c
mpirun -np $SLURM_NTASKS ./hello_world
```



https://ua-researchcomputing-hpc.github.io/MPI-Examples/Multi-Node-MPI-Job/

Parallel Computing on HPC

Using an Array to Submit Multiple jobs

Instead of this:
for i **in** $(seq 1 10); **do** sbatch script.slurm <submission options> ; **done**

Do this:

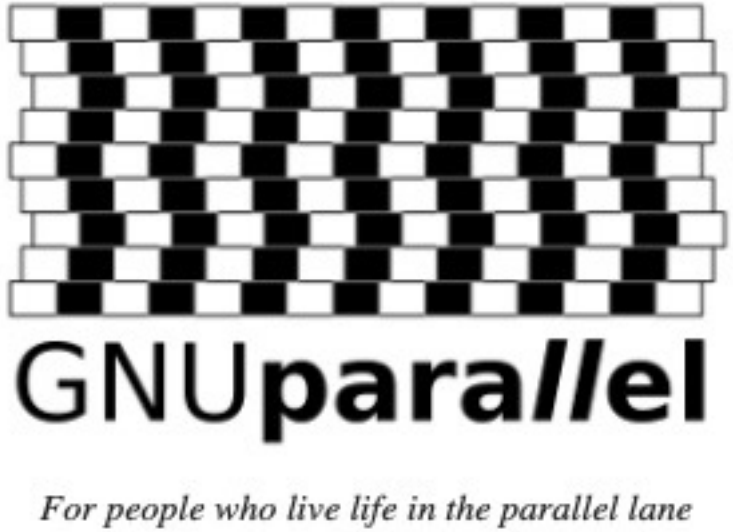
```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --time=00:01:00
#SBATCH --partition=standard
#SBATCH --account=YOUR_GROUP
#SBATCH --array 1-5

echo "./sample_command input_file_${SLURM_ARRAY_TASK_ID}.in"
```

https://ua-researchcomputing-hpc.github.io/Array-and-Parallel/Basic-Array-Job/

Parallel Computing on HPC

Using gnu parallel to parallelize multiple tasks within one job




```
#!/bin/bash
#SBATCH --ntasks=28
#SBATCH --nodes=1
#SBATCH --time=00:01:00
#SBATCH --partition=standard
#SBATCH --account=YOUR_GROUP

module load parallel
seq 1 100 | parallel 'DATE=$( date +"%T" ) && sleep 0.{} && echo "Host: $HOSTNAME ; Date: $DATE; {}"’’
```

Output

```
(ocelote) [netid@junonia ~]$ head slurm-74027.out
Host: i10n18 ; Date: 16:45:55; 1
Host: i10n18 ; Date: 16:45:55; 10
Host: i10n18 ; Date: 16:45:55; 11
```

https://ua-researchcomputing-hpc.github.io/Array-and-Parallel/Basic-Parallel-Job/

Parallel Computing on HPC - R

Quick Intro to Parallel Computing in R
https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html

Using an Array with an R script
You can create an R script that generates 1000 randomized 1s and 0s,
store them as a dataframe, then save the dataframe to an output file.
Then run this R script as an array job.
https://ua-researchcomputing-hpc.github.io/R-Examples/R-Array-Jobs/

Check out the tidyverse – an opinionated collection of R packages designed for data science
install.packages("tidyverse")

For an excellent hands-on Parallel Analysis in R tutorial:
https://github.com/ljdursi/beyond-single-core-R
It covers these packages:
parallel, foreach, bigmemory, Rdsm, pbdR

Parallel Computing on HPC - Python

The Python **multiprocessing** package is a popular way to distribute a workflow over multiple CPUs on a single node. In this example, we'll be spreading 1000 processes over multiple CPUs using a **pool** of workers.

```
#!/usr/bin/env python3

import os, time
from multiprocessing import Pool

num_cpus=int(os.environ["SLURM_CPUS_ON_NODE"])
print("Running a pool with %s workers"%num_cpus)

def f(x):
        # The sleep will allow us to get a sense of the speedup
        time.sleep(1)
        return x*x

if __name__=="__main__":
        with Pool(num_cpus) as p:
                data = range(1,1000)
                # p.map takes an interable, maps it onto a function, and executes the processes
                # distributed over the allocated CPUs using the pool of workers
                output = p.map(f,data)
```

Parallel Computing on HPC - Python

.. continued. Submit the script with different CPU allocations to see the speedup.
Set –cpus-per-task=N

```
#!/bin/bash
#SBATCH --job-name=Sample_Multiprocessing
### Stick to 1 task for python multiprocessing and request
### CPUs using the --cpus-per-task option
#SBATCH --ntasks=1
### Set cpus-per-task below to the number of processes
#SBATCH --cpus-per-task=1
#SBATCH --nodes=1
#SBATCH --time=05:00:00
#SBATCH --partition=standard
#SBATCH --account=YOUR_GROUP

module load python/3.6
time python3 multiprocess.py
```

https://ua-researchcomputing-hpc.github.io/Python-Examples/Multiprocessing/

Parallel Computing References

Introduction to Parallel Computing Tutorial
Author: Blaise Barney, Livermore Computing (retired), Donald Frederick, LLNL
https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial##Overview

Recommended reading

"Introduction to Parallel Computing", Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar.
University of Oregon - Intel Parallel Computing Curriculum
https://ipcc.cs.uoregon.edu/curriculum.html
An Introduction to Linux - https://cvw.cac.cornell.edu/Linux/
Linux Tutorial for Beginners: Introduction to Linux Operating System
- https://www.youtube.com/watch?v=V1y-mbWM3B8
"Introduction to Linux" - Boston University
- https://www.bu.edu/tech/files/2018/05/2018-Summer-Tutorial-Intro-to-Lin…

https://www.machinelearningplus.com/python/parallel-processing-python/