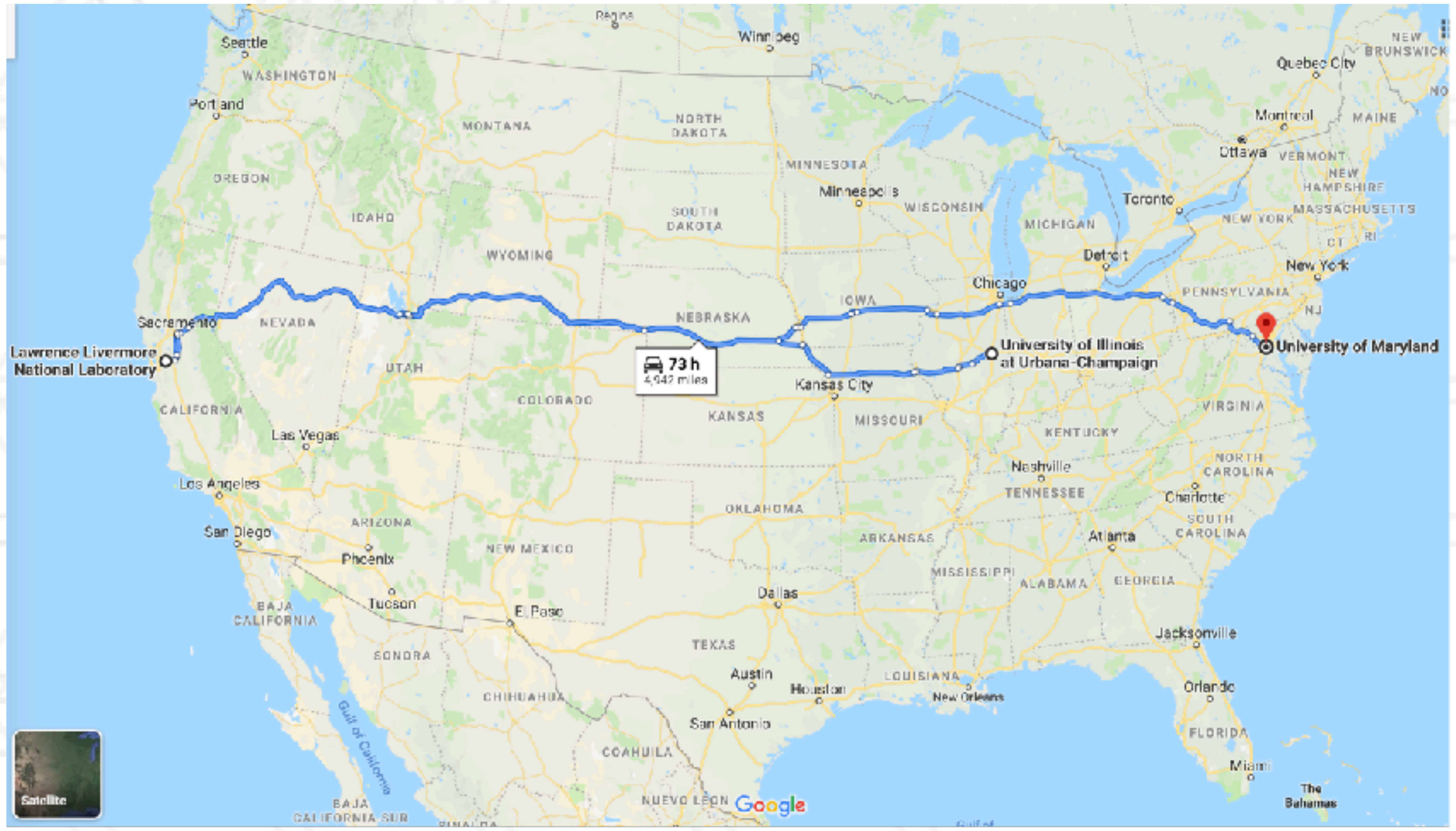
Course Overview

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

About the instructor

- Ph.D. from the University of Illinois
- Spent eight years at Lawrence Livermore National Laboratory
- Started at the University of Maryland in 2019
- Research areas:
  - High performance computing
  - Distributed AI



Introductions

- Name
- Junior/Senior
- Something interesting/ unique about yourself
- Why this course? (optional)

This course is

- An introduction to parallel computing
- 416: Upper Level CS Coursework / General Track / Area 1: Systems
- 616: Qualifying course for MS/PhD: Computer Systems
- Work expected:
  - Four to five programming assignments
  - Four quizzes
  - Midterm exam: in class on October 26 (tentative)
  - Final exam: on December 16 (10:30 am-12:30 pm)

Course topics

- Introduction to parallel computing (1 week)
- Distributed memory parallel programming (3 weeks)
- Shared memory parallel programming (2 weeks)
- Parallel algorithms (2 weeks)
- Performance analysis (1 week)
- Performance issues (2 weeks)
- Parallel simulation codes (2 weeks)

Tools we will use for the class

- Syllabus, lecture slides, assignment descriptions on course website:
  - http://www.cs.umd.edu/class/fall2023/cmsc416
- Programming assignment submissions on gradescope
- Quizzes on ELMS
- Discussions: Piazza
  - https://piazza.com/umd/fall2023/cmsc416/home
- If you want to contact the course staff outside of piazza, send an email to cmsc416-bhatele@cs.umd.edu

Zaratan accounts

- Zaratan is the UMD DIT cluster we'll use for the programming assignments
- The TAs will provide instructions for logging on to zaratan
- Helpful resources:
  - https://www.glue.umd.edu/hpcc/help/usage.html
  - https://missing.csail.mit.edu
  - https://gitlab.cs.umd.edu/mmarsh/quick-refs

COVID guideline

If you test positive for COVID, please isolate as per university/CDC guidelines.

Excused absence

Any student who needs to be excused for an absence from a single lecture, due to a medically necessitated absence shall make a reasonable attempt to inform the instructor of his/her illness prior to the class. Upon returning to the class, present the instructor with a self-signed note attesting to the date of their illness. Each note must contain an acknowledgment by the student that the information provided is true and correct. Providing false information to University officials is prohibited under Part 9(i) of the Code of Student Conduct (V-1.00(B) University of Maryland Code of Student Conduct) and may result in disciplinary action.

Self-documentation may not be used for Major Scheduled Grading Events (midterm and final exams) and it may only be used for one class meeting during the semester. Any student who needs to be excused for a prolonged absence (two or more consecutive class meetings), or for a Major Scheduled Grading Event, must provide written documentation of the illness from the Health Center or from an outside health care provider. This documentation must verify dates of treatment and indicate the timeframe that the student was unable to meet academic responsibilities. In addition, it must contain the name and phone number of the medical service provider to be used if verification is needed. No diagnostic information will ever be requested.

Use of LLMs

You can use LLMs such as ChatGPT as you would use Google for research. However, you cannot generate your solutions only using ChatGPT. You must demonstrate independent thought and effort.

If you use ChatGPT for anything class related, you must mention that in your answer/report. Please note that LLMs provide unreliable information, regardless of how convincingly they do so. If you are going to use an LLM as a research tool in your submission, you must ensure that the information is correct and addresses the actual question asked.

What is parallel computing?

- Serial or sequential computing: doing a task in sequence on a single processor
- Parallel computing: breaking up a task into sub-tasks and doing them in parallel (concurrently) on a set of processors (often connected by a network)
- Some tasks do not need any communication: embarrassingly parallel

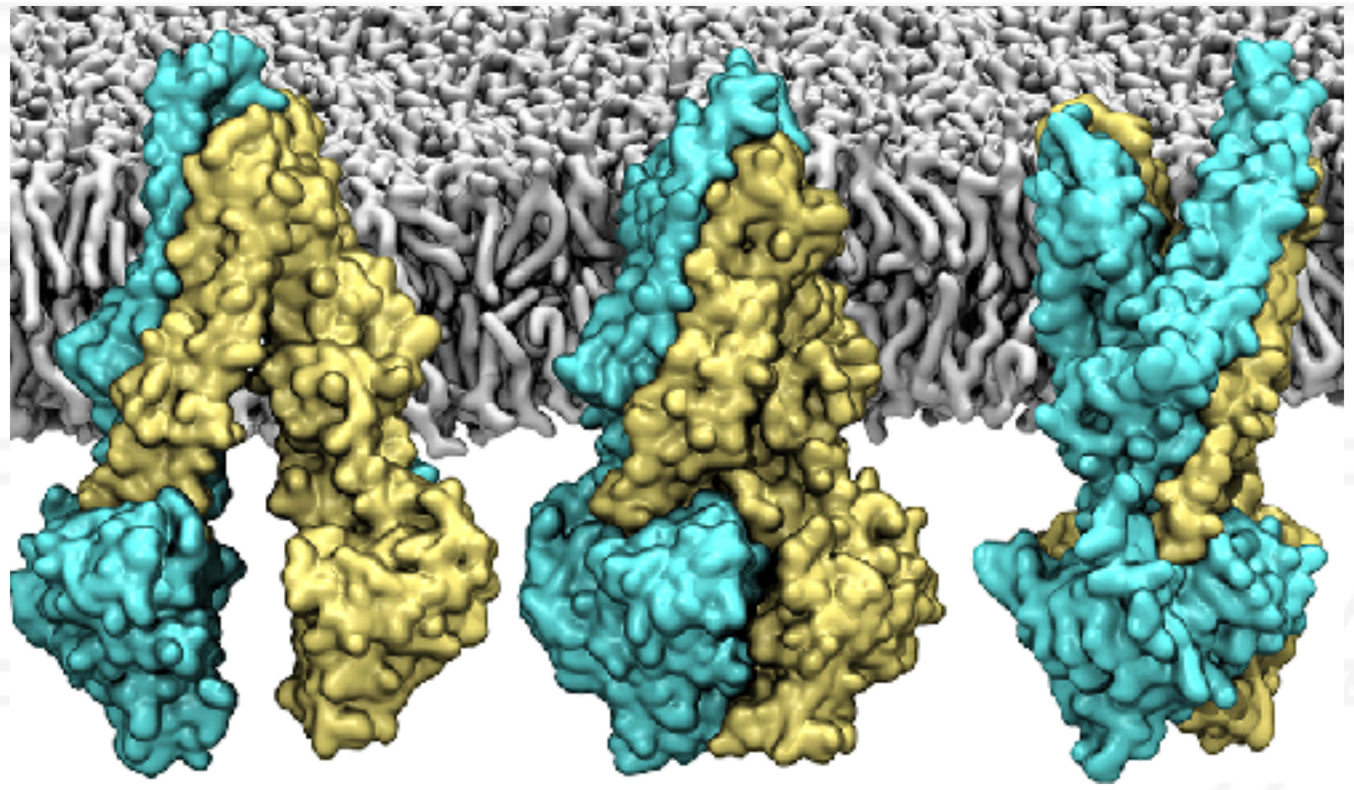
What is parallel computing?

- Does it include:
  - Grid computing
  - Distributed computing
  - Cloud computing
- Does it include:
  - Superscalar processors
  - Vector processors
  - Accelerators (GPUs, FPGAs)

The need for parallel computing or HPC

HPC stands for High Performance Computing

Drug discovery

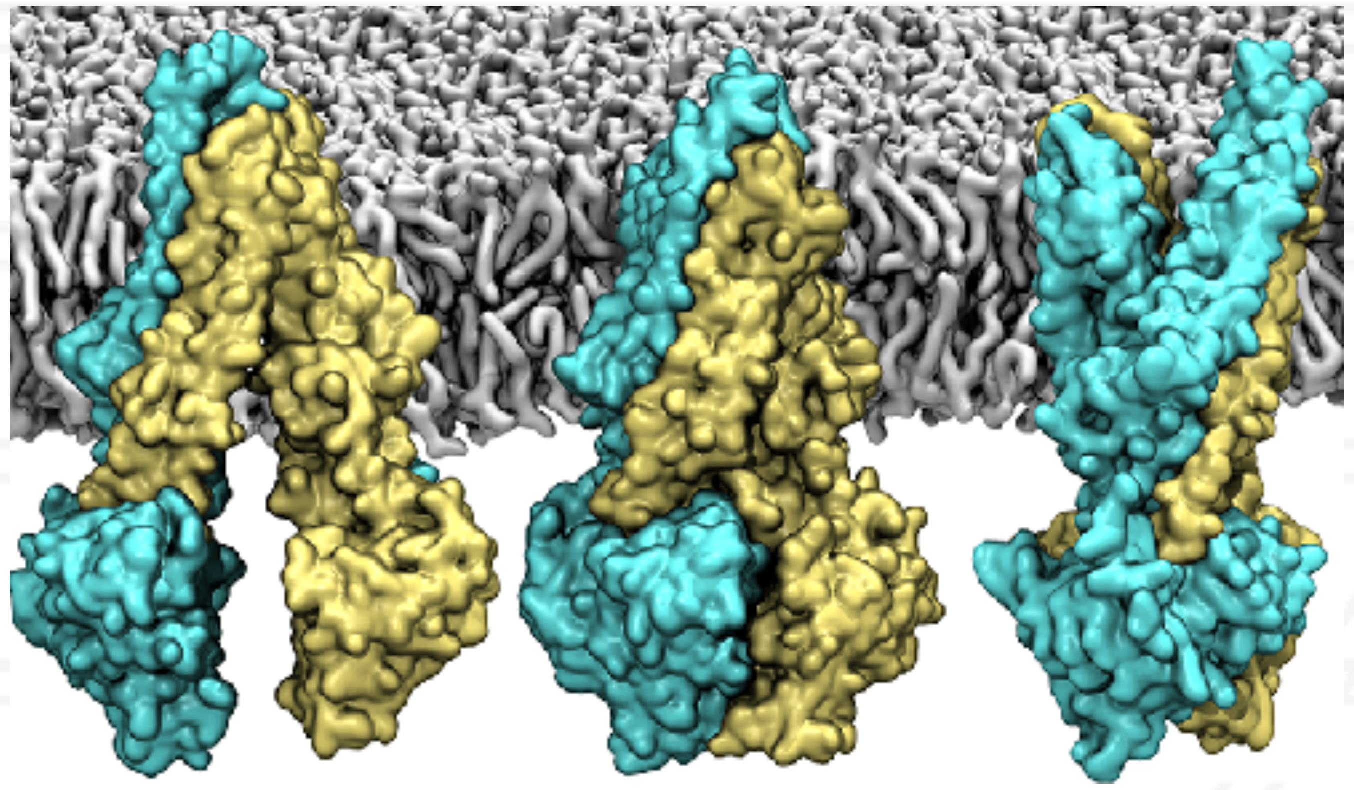


https://www.nature.com/articles/nature21414

The need for parallel computing or HPC

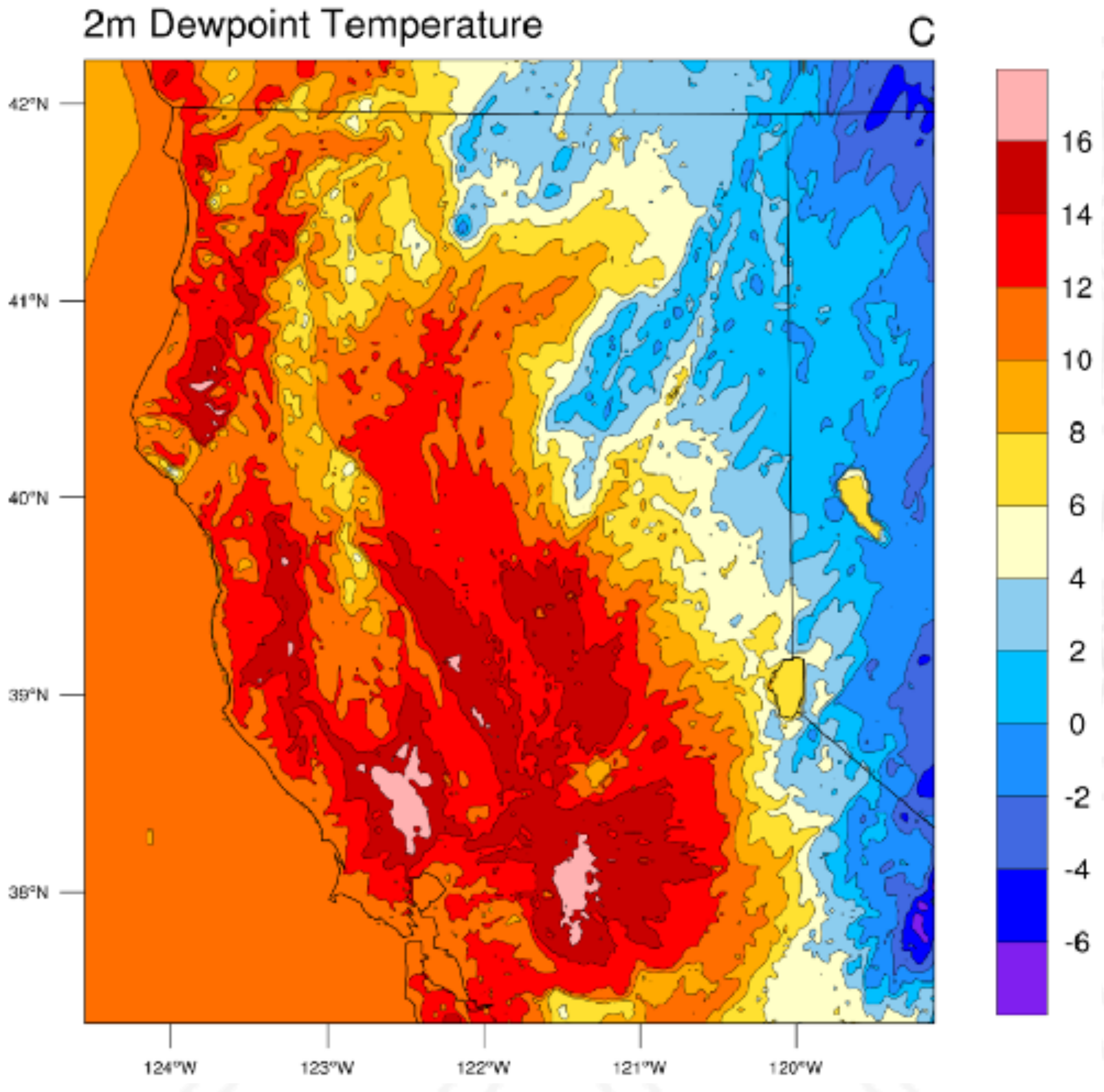
HPC stands for High Performance Computing

Drug discovery



https://www.nature.com/articles/nature21414

Weather forecasting

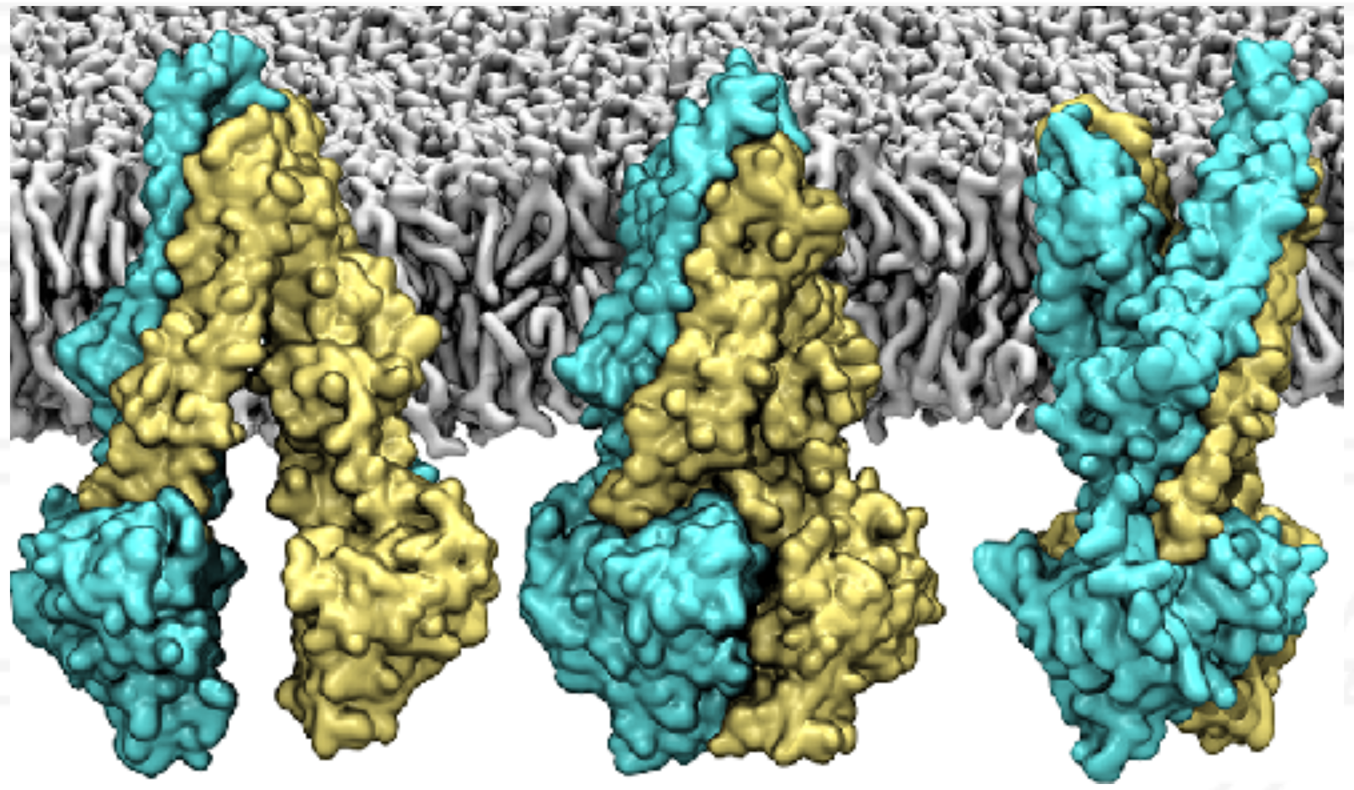


https://www.ncl.ucar.edu/Applications/wrf.shtml

The need for parallel computing or HPC

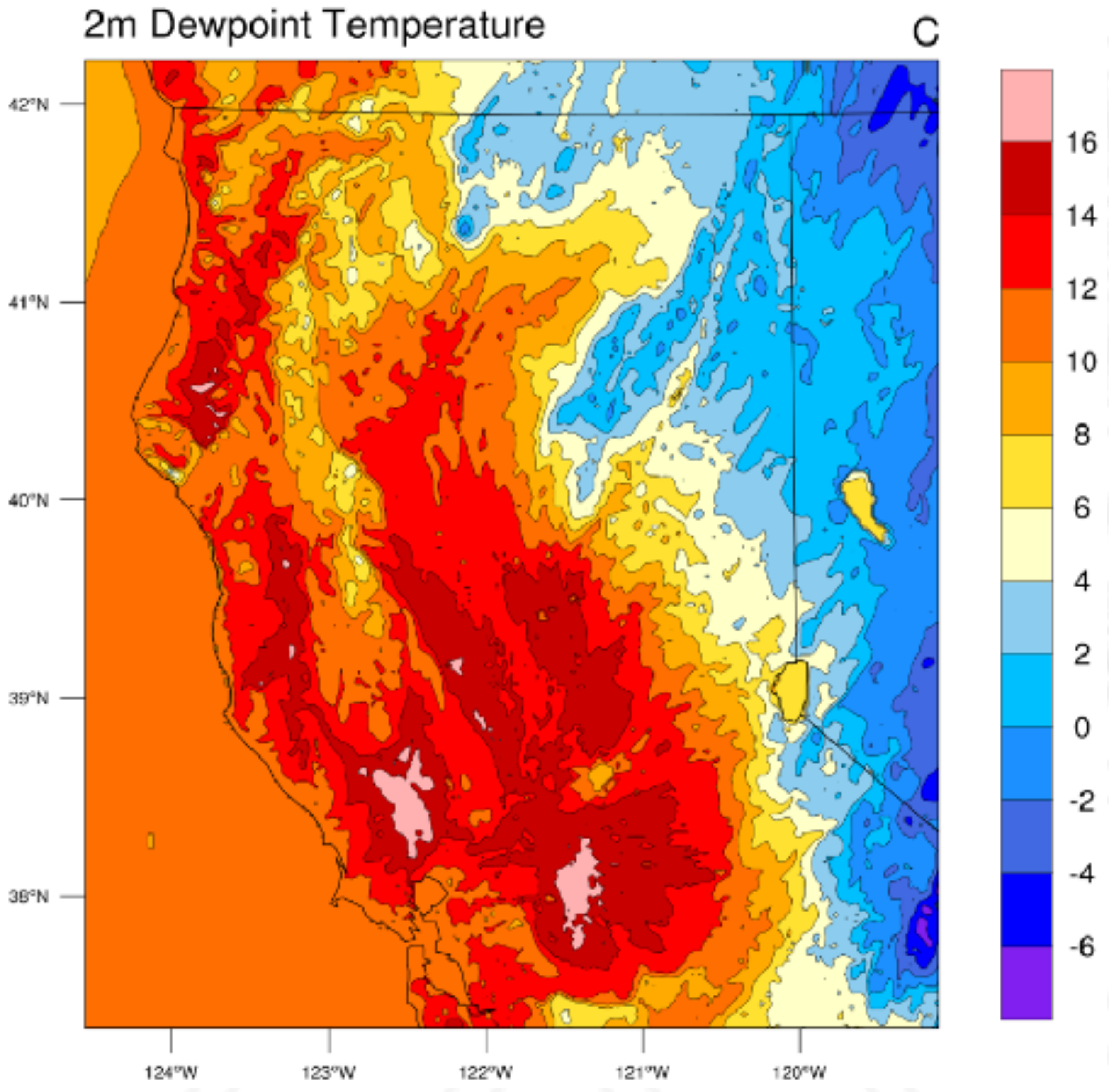
HPC stands for High Performance Computing

Drug discovery



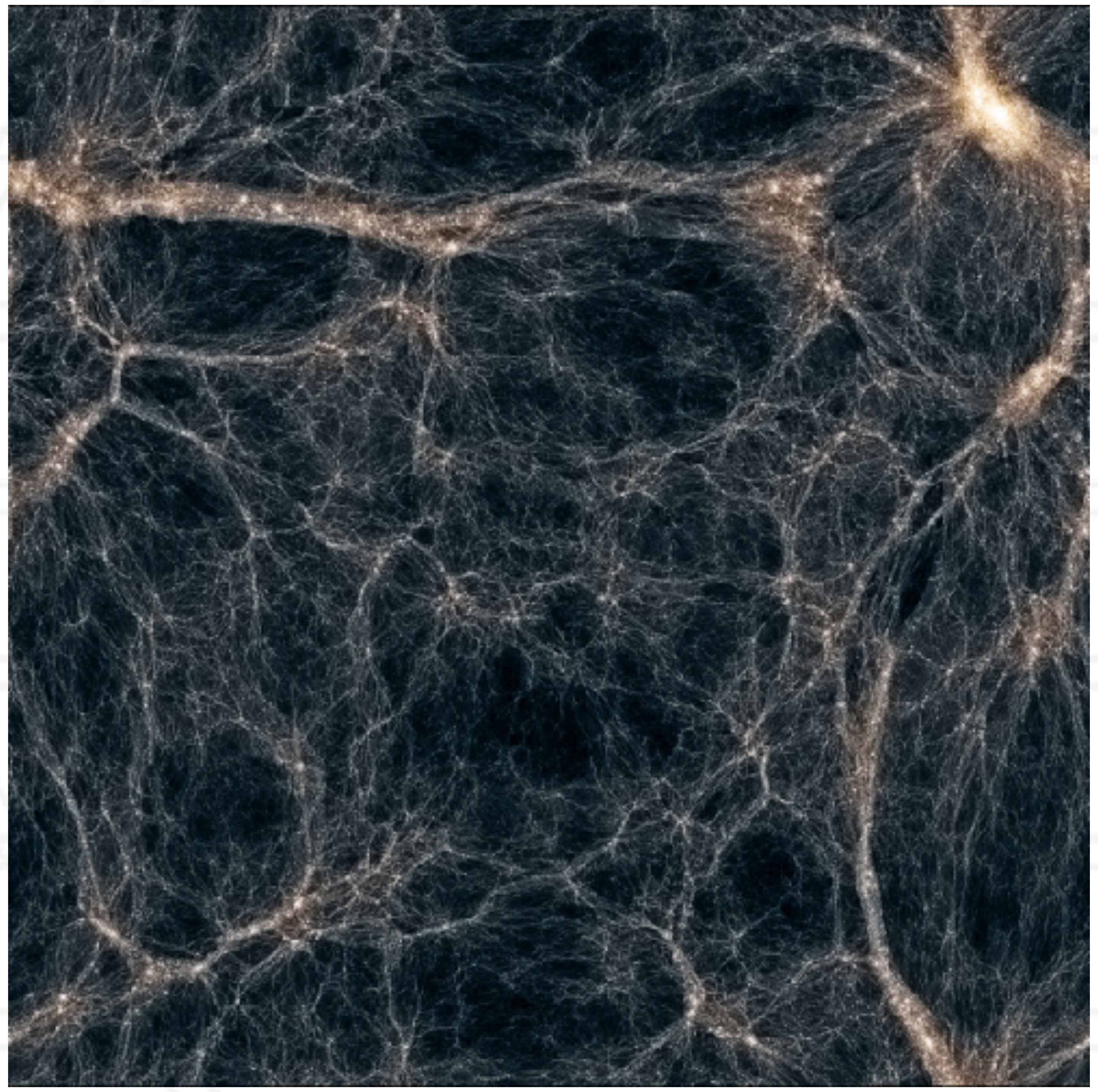
https://www.nature.com/articles/nature21414

Weather forecasting



https://www.ncl.ucar.edu/Applications/wrf.shtml

Study of the universe



https://www.nas.nasa.gov/SC14/demos/demo27.html

Why do we need parallelism?

- Make some science simulations feasible in the lifetime of humans
  - Typical constraints are speed or memory requirements
  - Made possible by using more than one core/processor
- Provide answers in realtime or near realtime

Large supercomputers

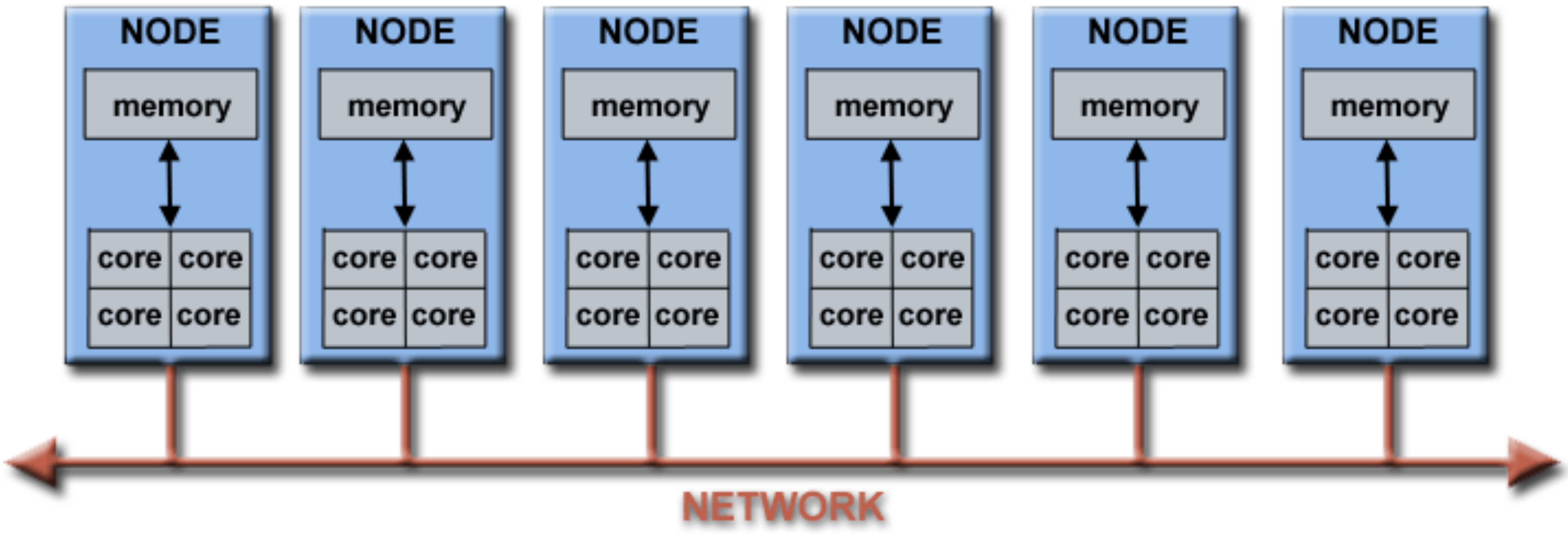
- Top500 list: https://top500.org/lists/top500/2023/06

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

https://www.olcf.ornl.gov/frontier

Parallel architecture

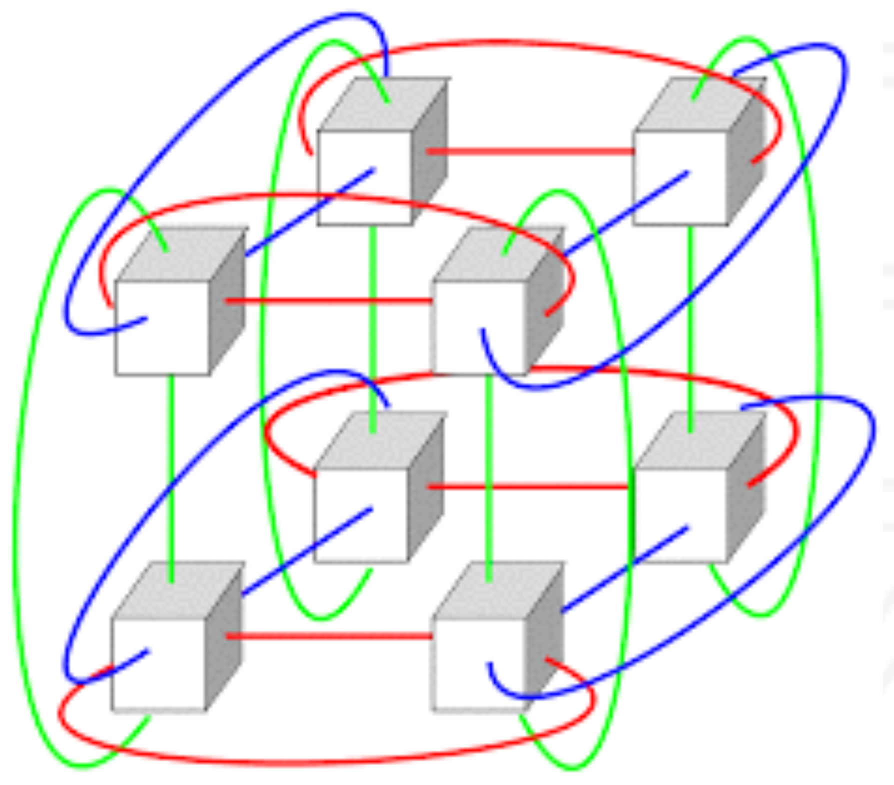
- A set of nodes or processing elements connected by a network.



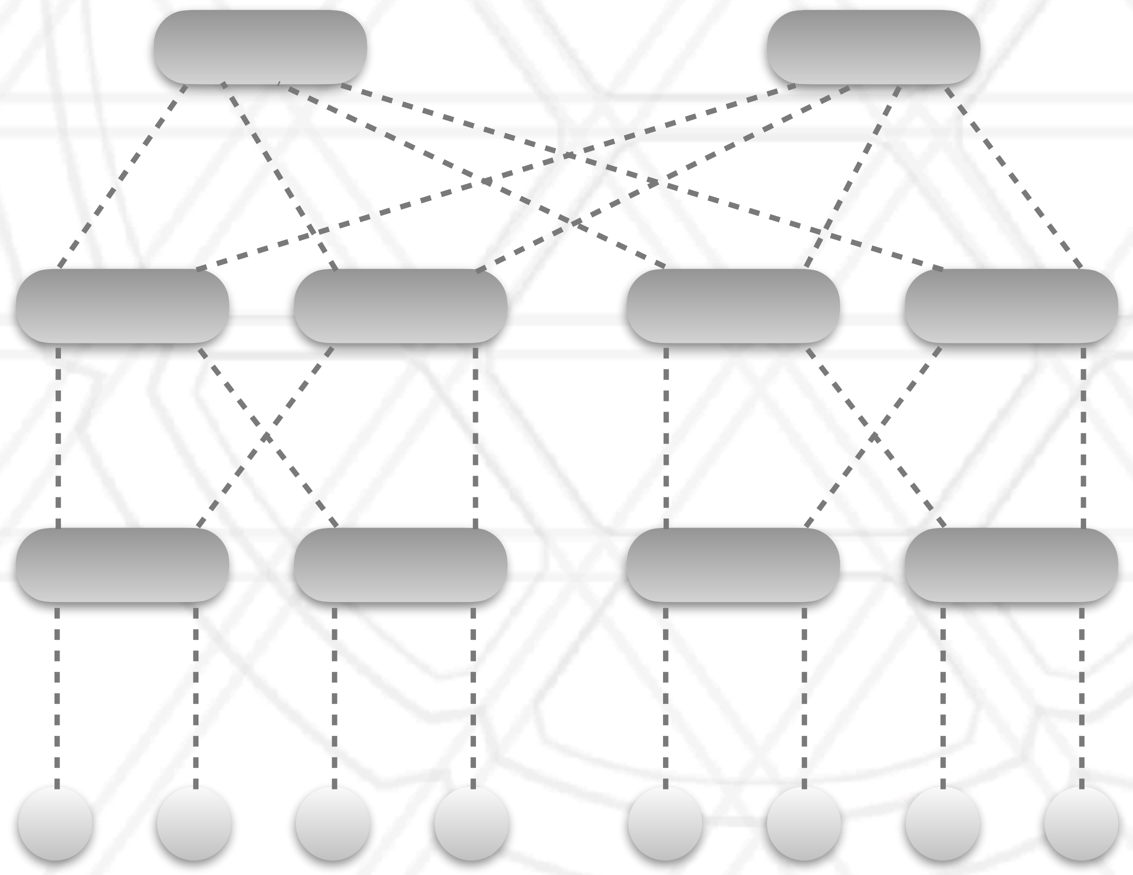
https://computing.llnl.gov/tutorials/parallel_comp

Interconnection networks

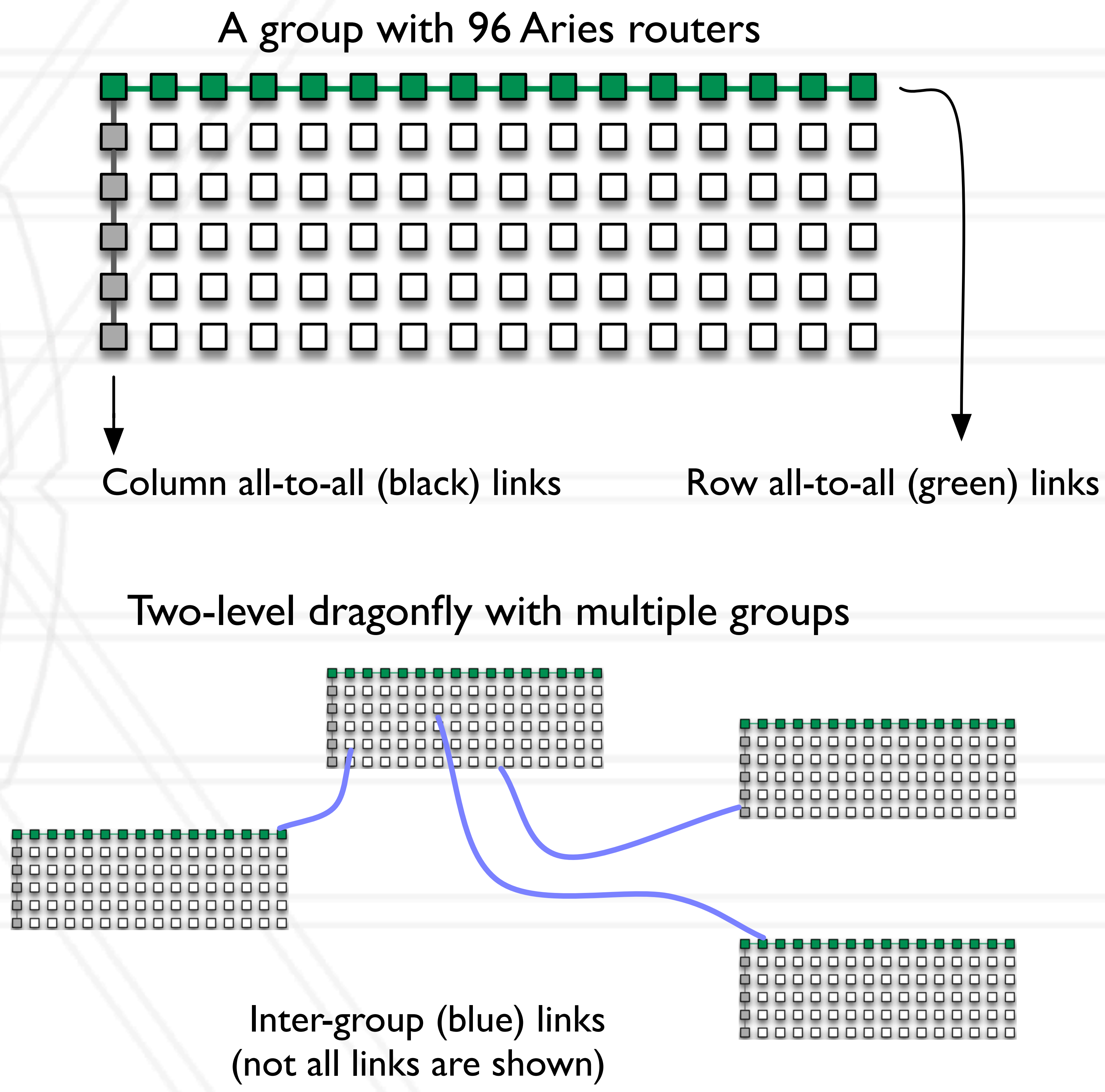
- Different topologies for connecting nodes together
- Used in the past: torus, hypercube
- More popular currently: fat-tree, dragonfly



Torus



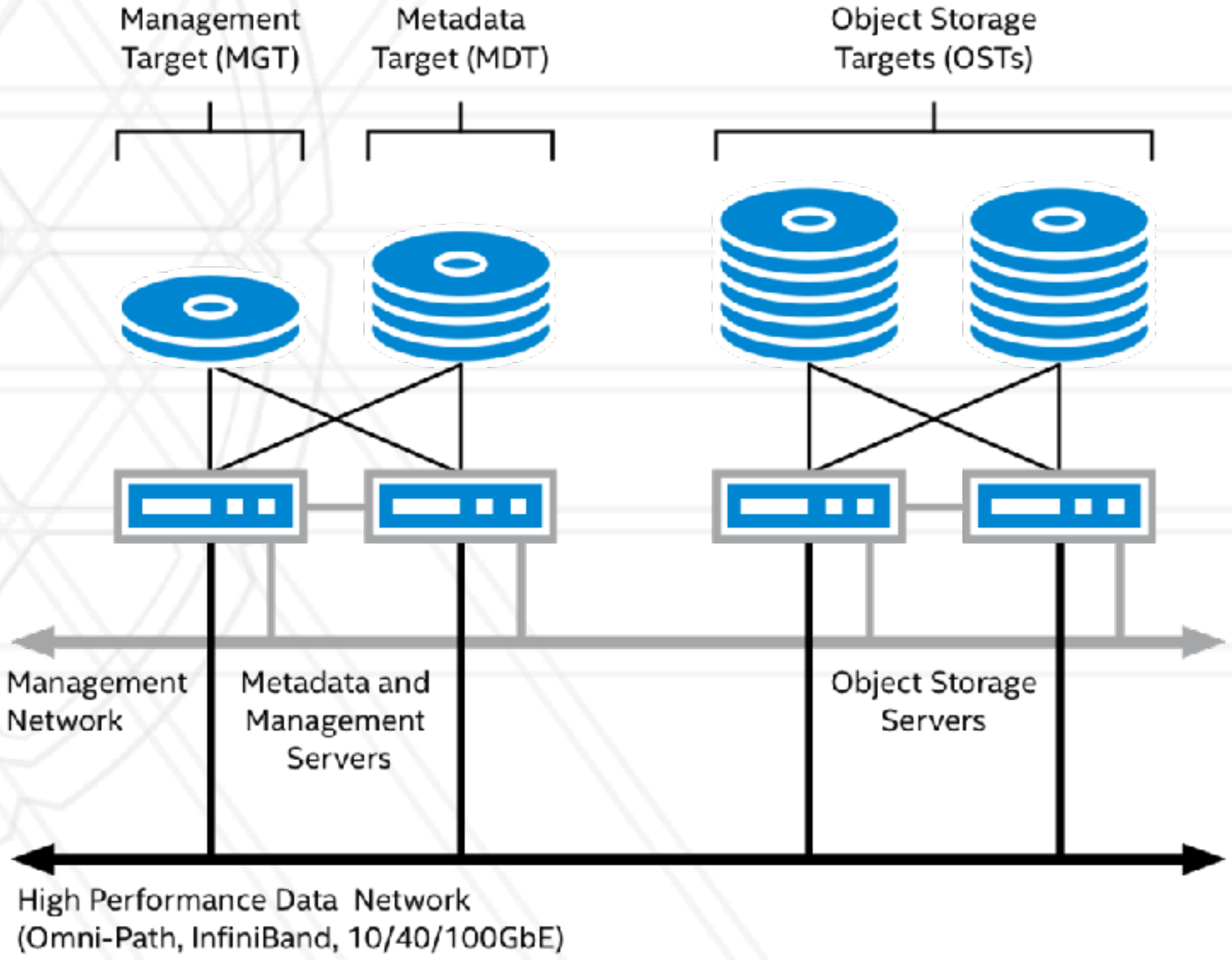
Fat-tree



Dragonfly

I/O sub-system / Parallel file system

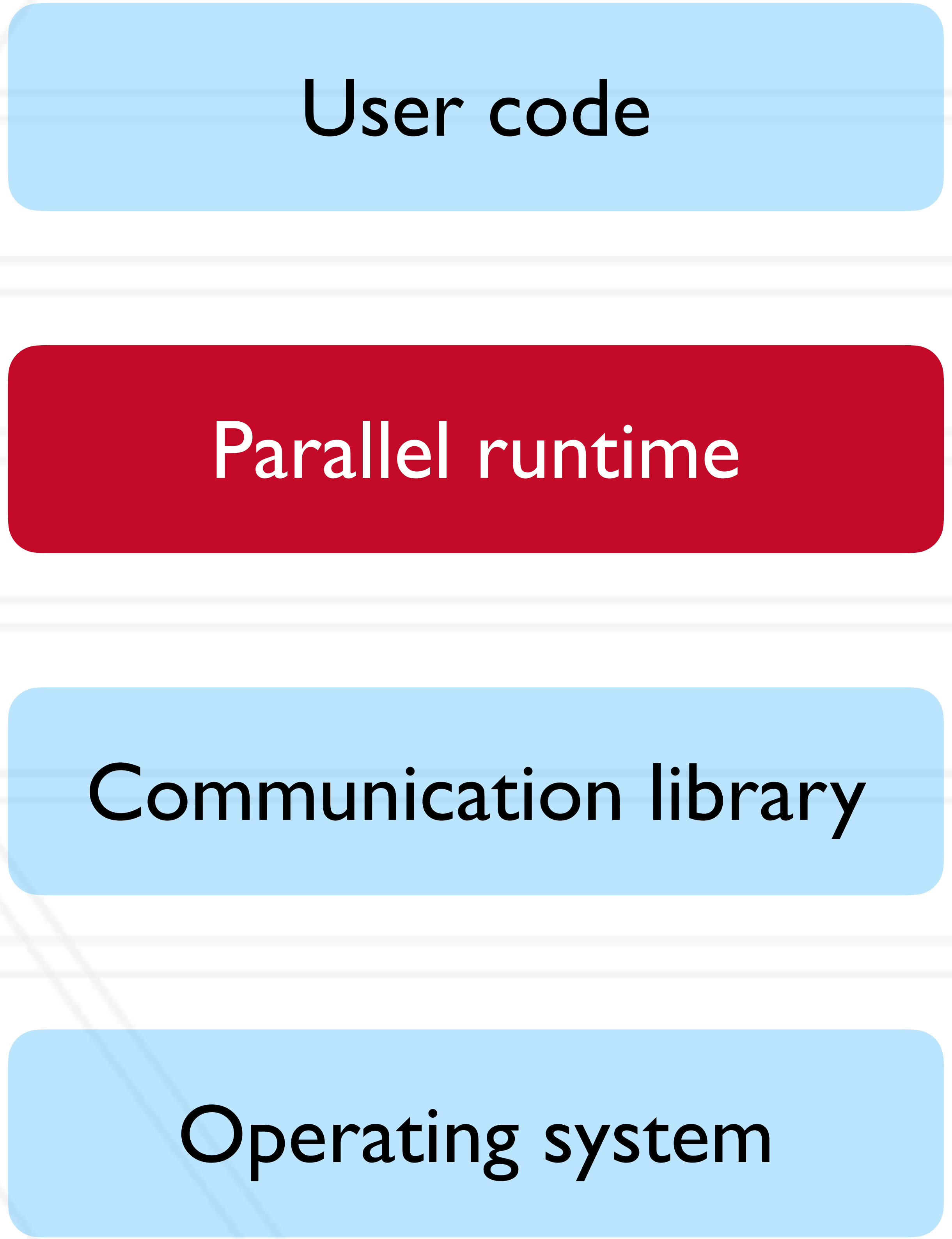
- Home directories and scratch space typically on a parallel file system
- Mounted on all login and compute nodes



http://wiki.lustre.org/Introduction_to_Lustre

System software: models and runtimes

- Parallel programming model
  - Parallelism is achieved through language constructs or by making calls to a library and the execution model depends on the model used.
- Parallel runtime [system]:
  - Implements the parallel execution model
- Shared memory/address-space
  - Pthreads, OpenMP, Chapel
- Distributed memory
  - MPI, Charm



Terminology and Definitions

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

- Quiz 0 has been posted on ELMS
- Zaratan accounts have been created for everyone

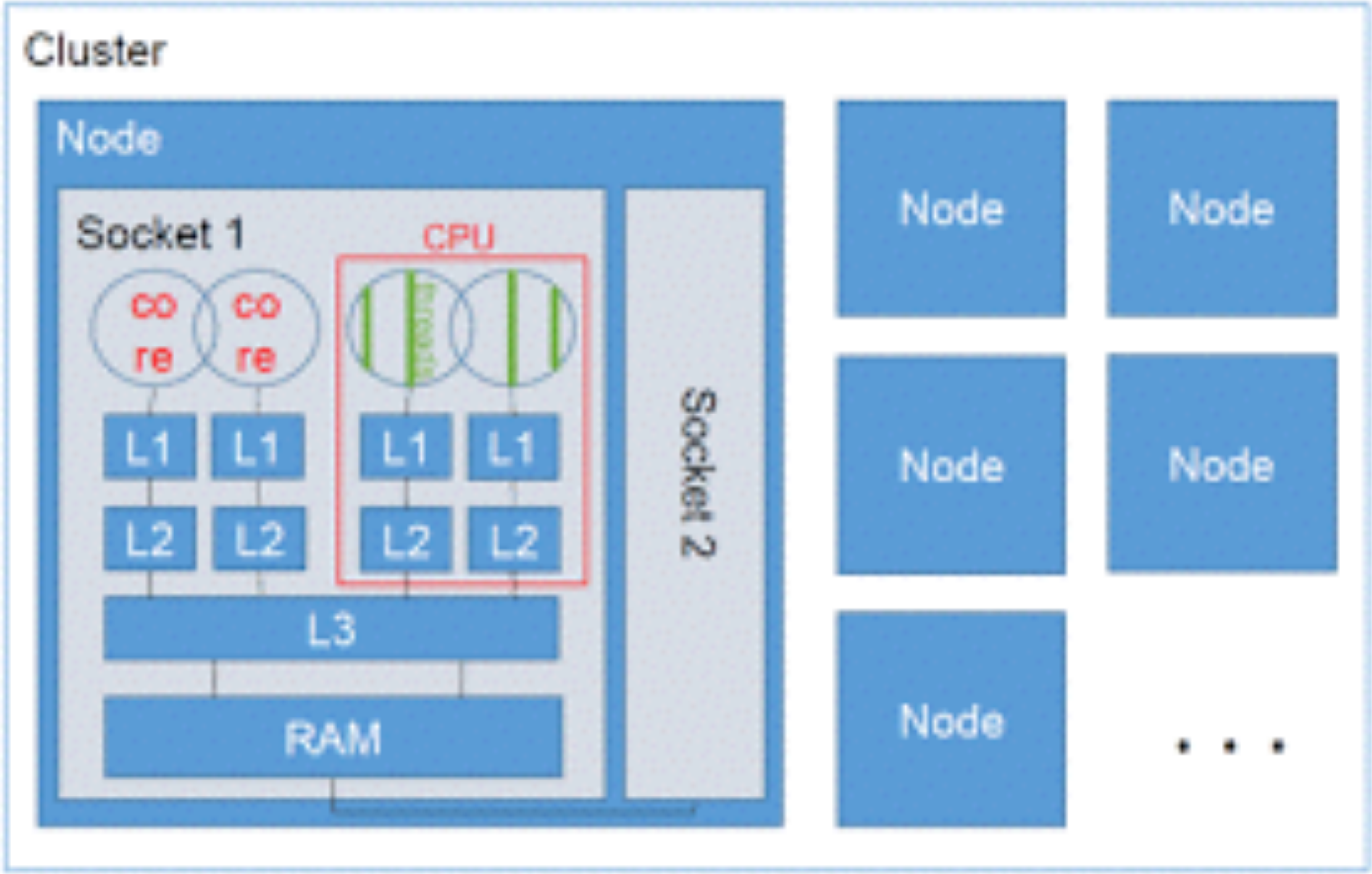
Getting started with zaratan

- Over 380 nodes with AMD Milan processors (128 cores/node)
- 20 nodes with four NVIDIA A100 GPUs

```
ssh username@login.zaratan.umd.edu
```

Cores, sockets, nodes

- Core: a single execution unit that has a private L1 cache and can execute instructions independently
- Processor: several cores on a single Integrated Circuit (IC) or chip are called a multi-core processor
- Socket: physical connector into which an IC/chip or processor is inserted.
- Node: a packaging of sockets - motherboard or printed circuit board (PCB) that has multiple sockets

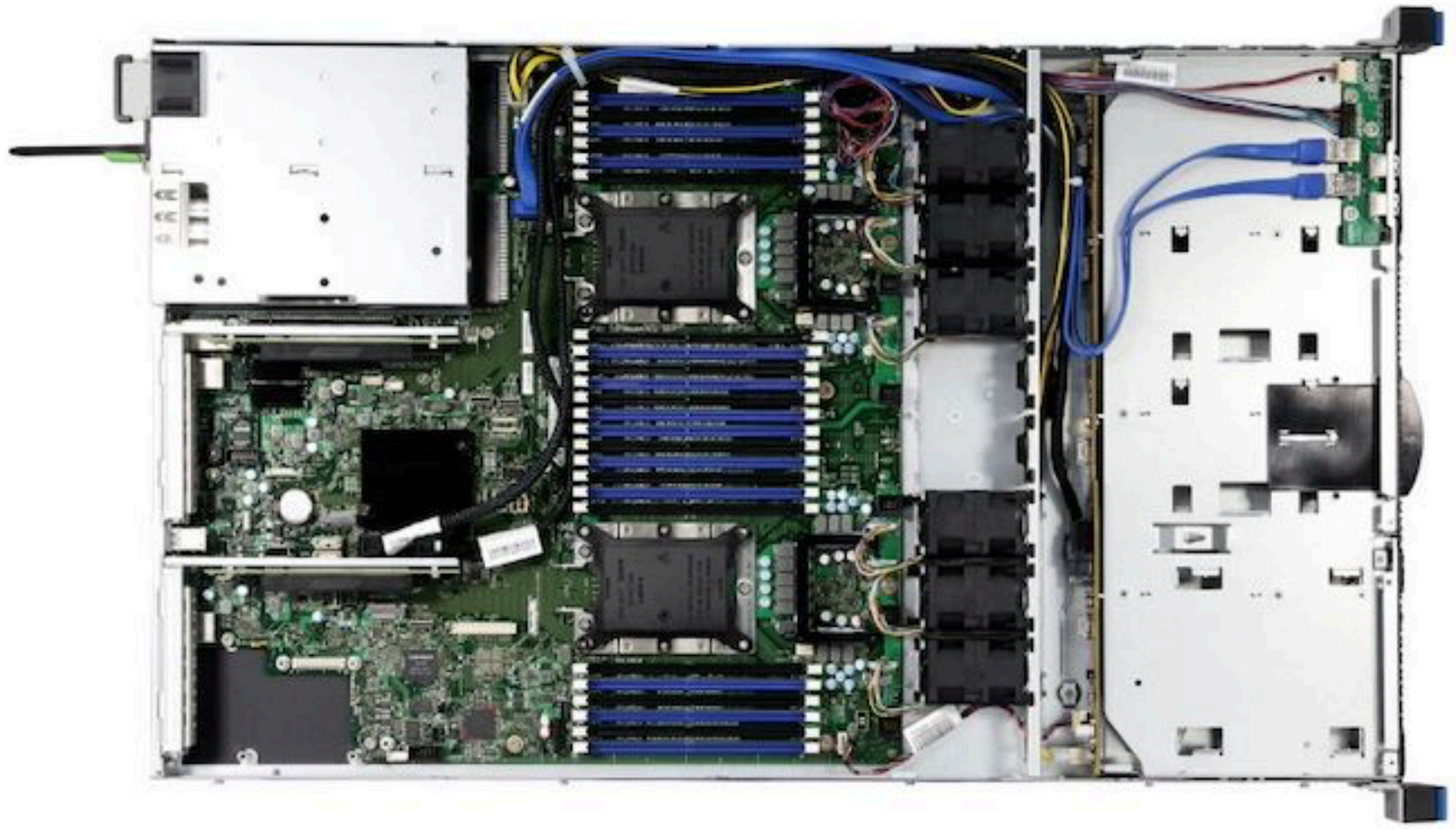


https://hpc-wiki.info/hpc/HPC-Dictionary

Rackmount servers

Rackmount servers

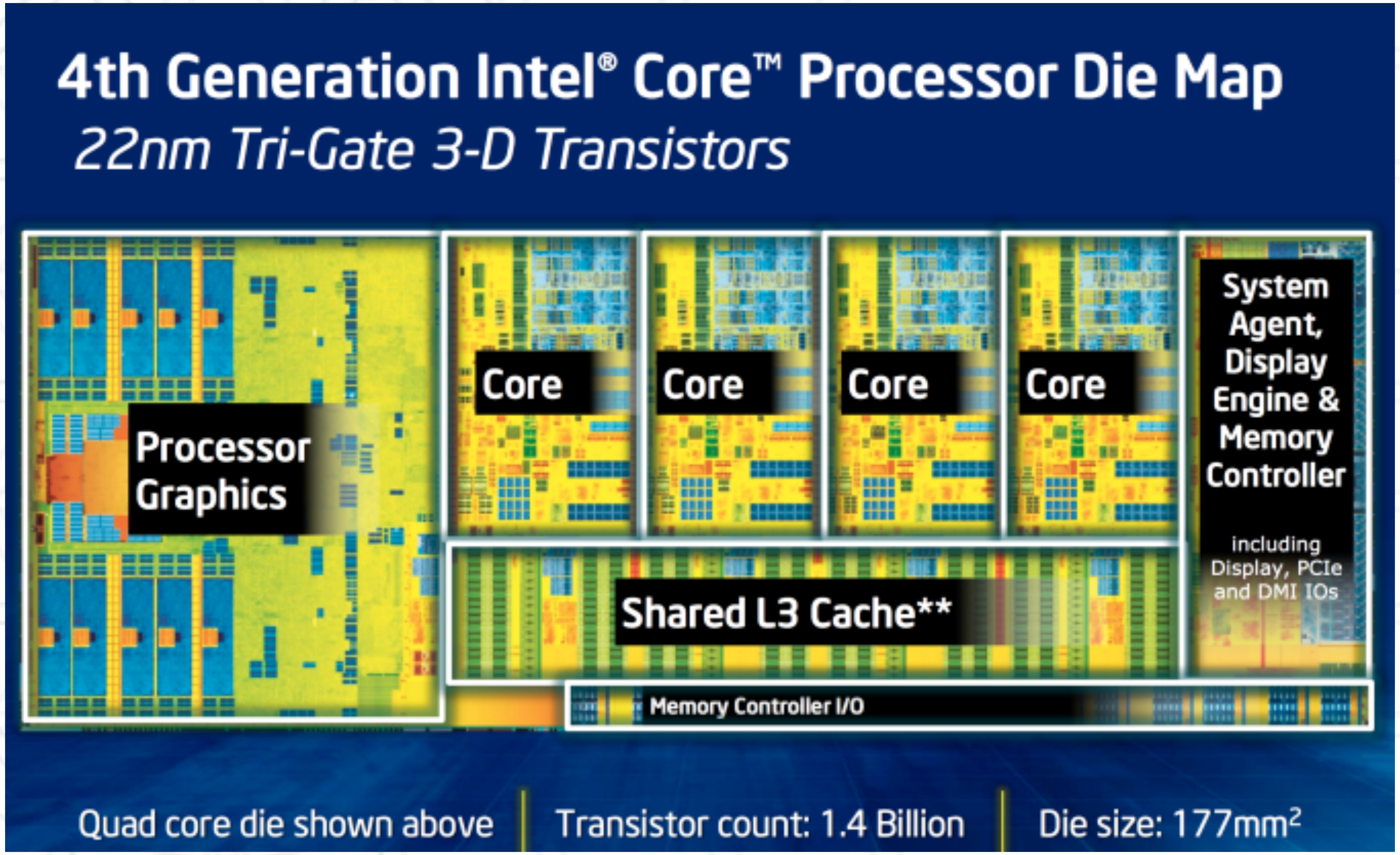
Rackmount server motherboard



https://www.anandtech.com/show/15924/chenbro-announces-rb13804-dual-socket-1u-xeon-4-bay-hpc-barebones-server

https://www.anandtech.com/show/7003/the-haswell-review-intel-core-i74770k-i54560k-tested

Rackmount server motherboard



https://www.anandtech.com/show/15924/chenbro-announces-rb13804-dual-socket-1u-xeon-4-bay-hpc-barebones-server

https://www.anandtech.com/show/7003/the-haswell-review-intel-core-i74770k-i54560k-tested

Job scheduling

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a “job” scheduler

Job Queue

	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a “job” scheduler
- The scheduler decides:
  - what job to schedule next (based on an algorithm: FCFS, priority-based, ….)
  - what resources (compute nodes) to allocate to the ready job

Job Queue

	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a “job” scheduler
- The scheduler decides:
  - what job to schedule next (based on an algorithm: FCFS, priority-based, ….)
  - what resources (compute nodes) to allocate to the ready job

- Compute nodes: dedicated to each job
- Network, filesystem: shared by all jobs

Job Queue

	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Compute nodes vs. login nodes

- Compute nodes: dedicated nodes for running jobs
  - Can only be accessed when they have been allocated to a user by the job scheduler
- Login nodes: nodes shared by all users to compile their programs, submit jobs etc.
- Service/managements nodes: I/O nodes, etc.

Supercomputers vs. commodity clusters

- Supercomputer refers to a large expensive installation, typically using custom hardware
  - High-speed interconnect
  - IBM Blue Gene, Cray XT, Cray XC
- Cluster refers to a cluster of nodes, typically put together using commodity (off-the-shelf) hardware

Serial vs. parallel code

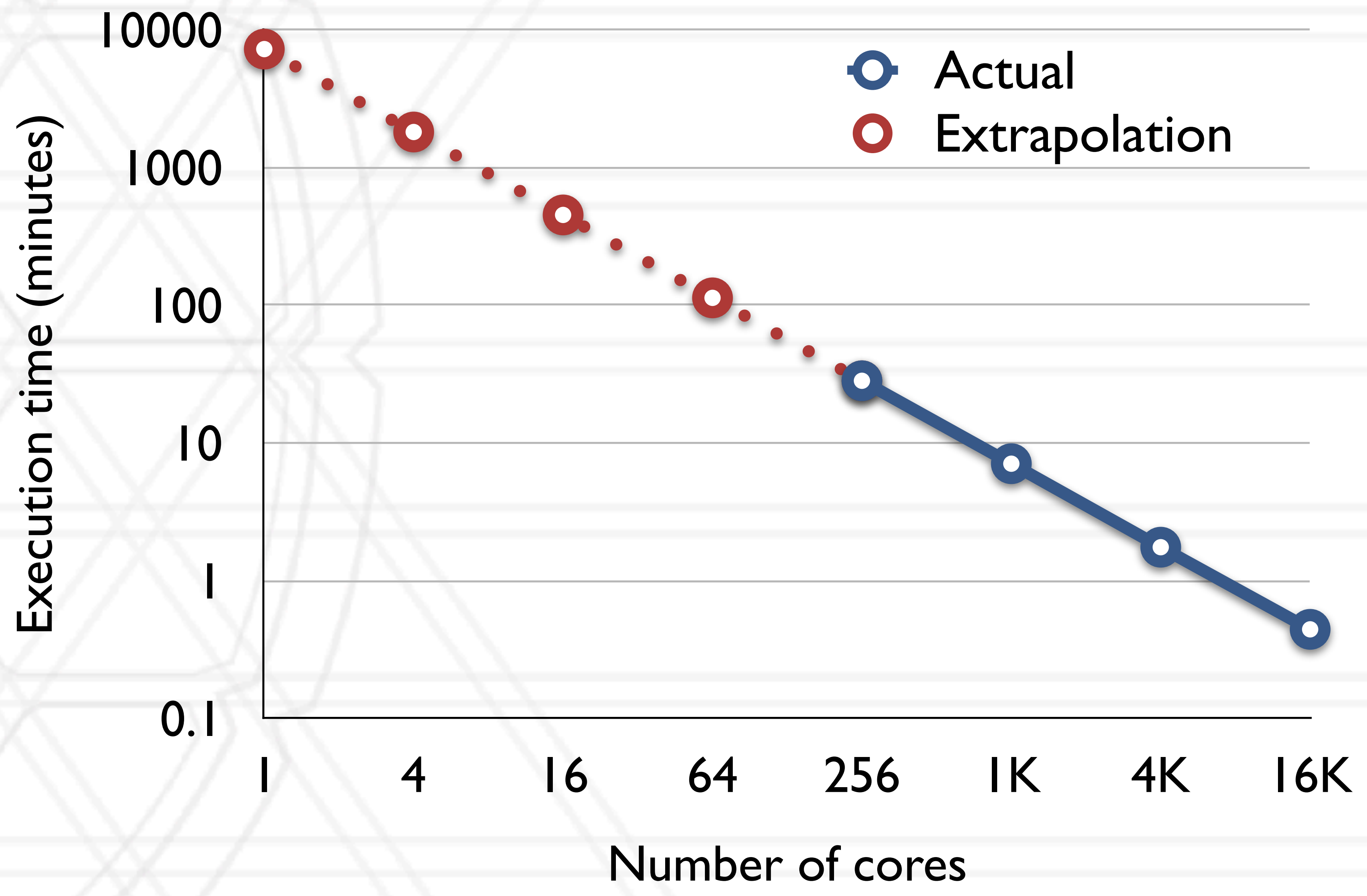
- Thread: a thread or path of execution managed by the OS
  - Threads share the same memory address space
- Process: heavy-weight, processes do not share resources such as memory, file descriptors etc.
- Serial or sequential code: can only run on a single thread or process
- Parallel code: can be run on one or more threads or processes

Scaling and scalable

- Scaling: running a parallel program on 1 to n processes
  - 1, 2, 3, … , n
  - 1, 2, 4, 8, …, n
- Scalable: A program is *scalable* if its performance improves when using more resources

Scaling and scalable

- Scaling: running a parallel program on 1 to n processes
  - 1, 2, 3, … , n
  - 1, 2, 4, 8, …, n
- Scalable: A program is *scalable* if its performance improves when using more resources



Weak versus strong scaling

- Strong scaling: *Fixed total* problem size as we run on more resources (processes or threads)
  - Sorting n numbers on 1 process, 2 processes, 4 processes, …
- Weak scaling: Fixed problem size per process but *increasing total* problem size as we run on more resources
  - Sorting n numbers on 1 process
  - 2n numbers on 2 processes
  - 4n numbers on 4 processes

Speedup and efficiency

- Speedup: Ratio of execution time on one process to that on p processes

$$\text{Speedup} = \frac{t_1}{t_p}$$

- Efficiency: Speedup per process

$$\text{Efficiency} = \frac{t_1}{t_p \times p}$$

Amdahl's law

- Speedup is limited by the serial portion of the code
  - Often referred to as the serial "bottleneck"
- Lets say only a fraction *f* of the code can be parallelized on *p* processes

$$\text{Speedup} = \frac{1}{(1-f) + f/p}$$

Amdahl's law

- Speedup is limited by the serial portion of the code
  - Often referred to as the serial "bottleneck"
- Lets say only a fraction *f* of the code can be parallelized on *p* processes

$$\text{Speedup} = \frac{1}{(1 - f) + f/p}$$

Amdahl's law

- Speedup is limited by the serial portion of the code
  - Often referred to as the serial "bottleneck"
- Lets say only a fraction *f* of the code can be parallelized on *p* processes

$$\text{Speedup} = \frac{1}{(1 - f) + f/p}$$

Amdahl's law

$$\text{Speedup} = \frac{1}{(1-f) + f/p}$$

```
fprintf(stdout,"Process %d of %d is on %s\n",
    myid, numprocs, processor_name);
fflush(stdout);

n = 10000;          /* default # of rectangles */
if (myid == 0)
startwtime = MPI_Wtime();
```

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

h   = 1.0 / (double) n;
sum = 0.0;
/* A slightly better approach starts from large i and works back */
for (i = myid + 1; i <= n; i += numprocs)
{
x = h * ((double)i - 0.5);
sum += f(x);
}
mypi = h * sum;

MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

Total time on 1 process = 100s
Serial portion = 40s
Portion that can be parallelized = 60s

$$f = \frac{60}{100} = 0.6$$

$$\text{Speedup} = \frac{1}{(1-0.6) + 0.6/p}$$

Communication and synchronization

- Each process may execute serial code independently for a while
- When data is needed from other (remote) processes, messaging occurs
  - Referred to as communication or synchronization or MPI messages
- Intra-node vs. inter-node communication
- Bulk synchronous programs: All processes compute simultaneously, then synchronize (communicate) together

Different models of parallel computation

- SIMD: Single Instruction Multiple Data
- MIMD: Multiple Instruction Multiple Data
- SPMD: Single Program Multiple Data
  - Typical in HPC

UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu