Introduction to Parallel Computing (CMSC416 / CMSC818X)

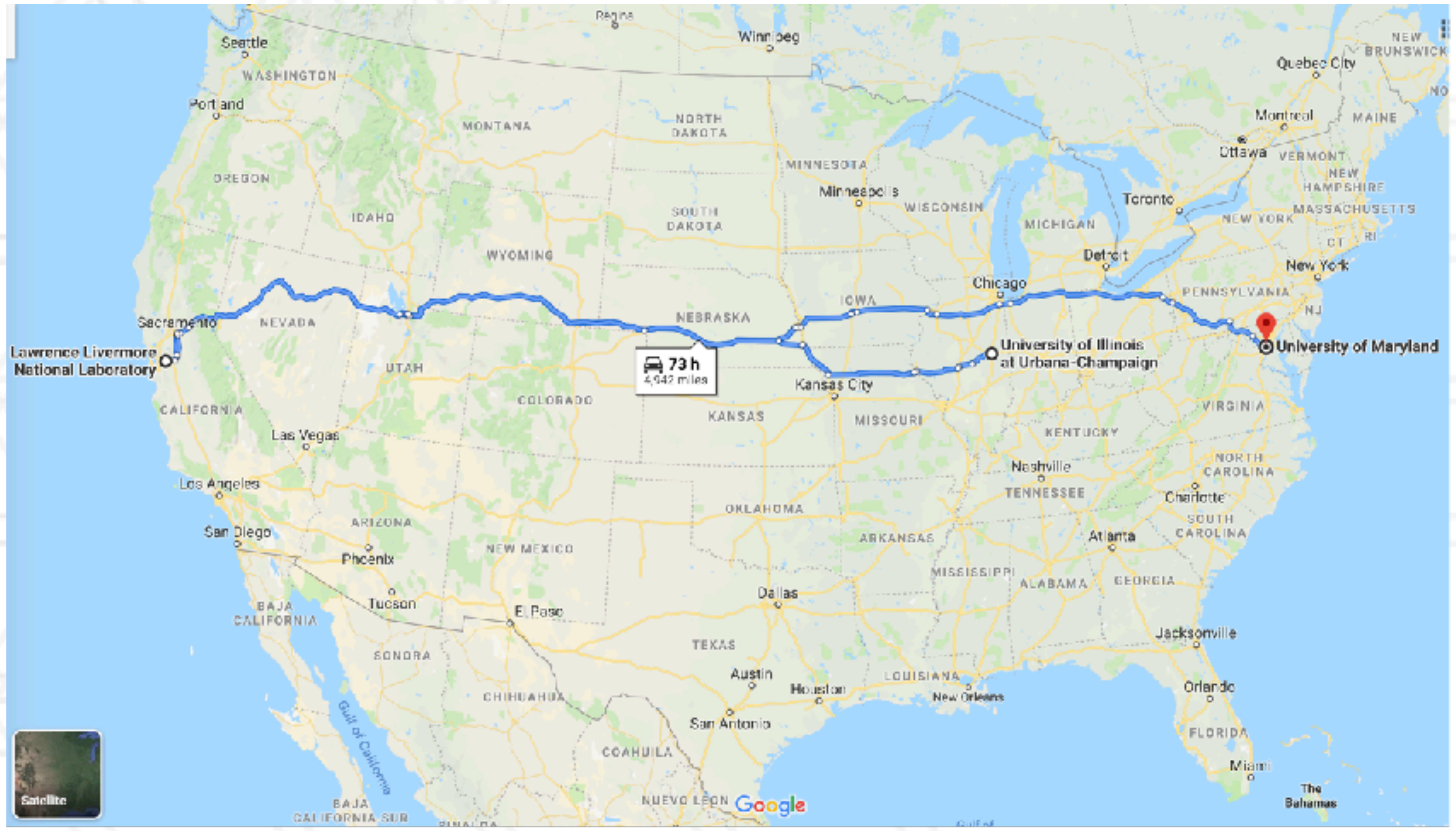
Course Overview

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



Masks are mandatory inside the classroom

President Pines provided clear expectations to the University about the wearing of masks for students, faculty, and staff. Face coverings over the nose and mouth are required while you are indoors at all times. There are no exceptions when it comes to classrooms, laboratories, and campus offices.

Students not wearing a mask will be given a warning and asked to wear one, or will be asked to leave the room immediately. Students who have additional issues with the mask expectation after a first warning will be referred to the Office of Student Conduct for failure to comply with a directive of University officials.

Introductions

- Name
- Junior/Senior/MS/PhD
- 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
- 818X: Qualifying course for MS/PhD: Computer Systems
- Work expected:
  - Four to five programming assignments
  - Four quizzes
  - Midterm exam: in class on October 26
  - Final exam: on December 15

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/fall2021/cmsc416
- Video recordings on Panopto
- Assignment submissions and quizzes on ELMS
- Discussions: Piazza
  - piazza.com/umd/fall2021/cmsc416cmsc818x
- If you want to send an email, cc both TAs and me

Deepthought2 accounts

- Shoken and Pooja will email your login/password for deepthought2
- Helpful resources:
  - http://www.cs.umd.edu/class/fall2021/cmsc416/deepthought2.shtml
  - https://www.glue.umd.edu/hpcc/help/usage.html
  - https://hpcbootcamp.readthedocs.io

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 two class meetings 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.

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

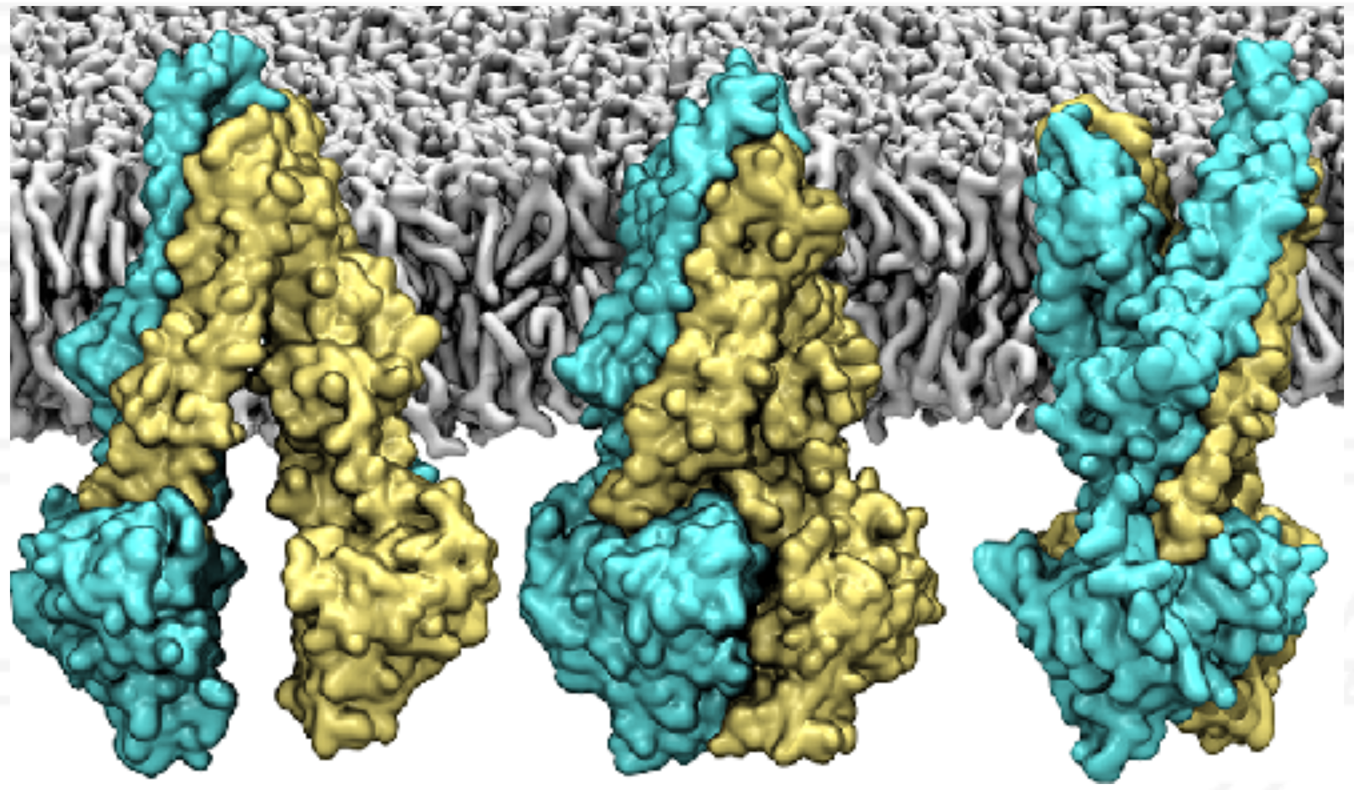
What is parallel computing?

- 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

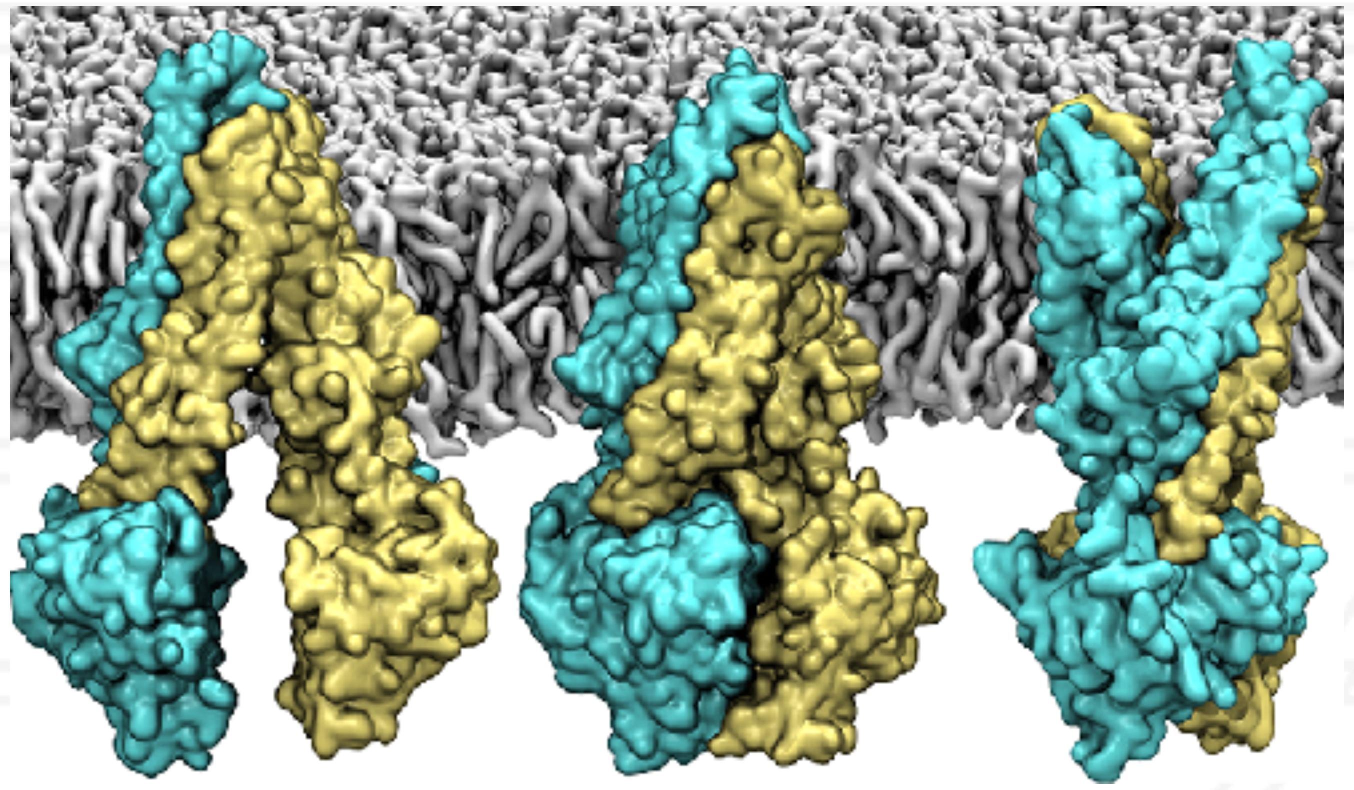


https://www.nature.com/articles/nature21414

The need for parallel computing or HPC

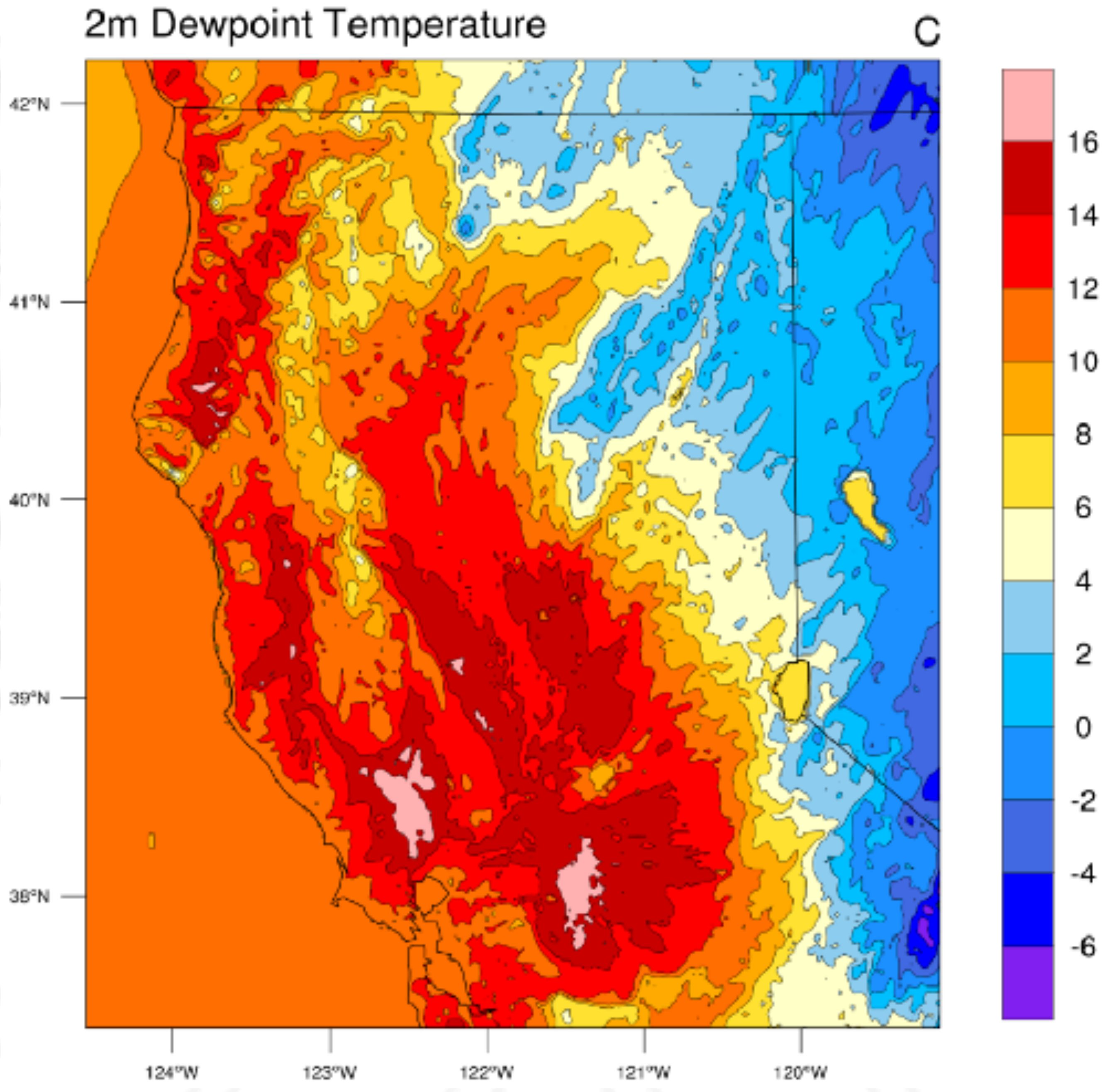
HPC stands for High Performance Computing

Drug discovery



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

Weather forecasting

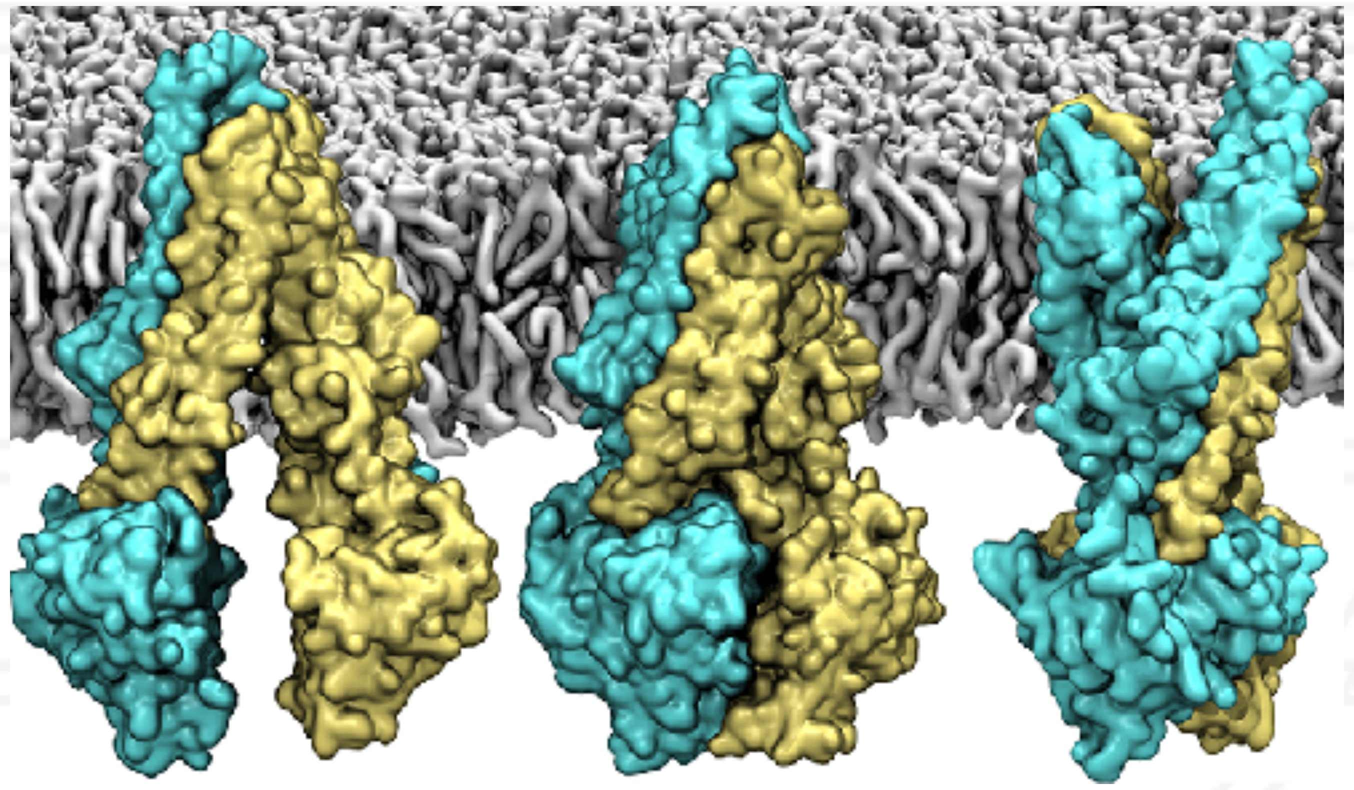


https://www.ncl.ucar.edu/Applications/wrf.shtml

The need for parallel computing or HPC

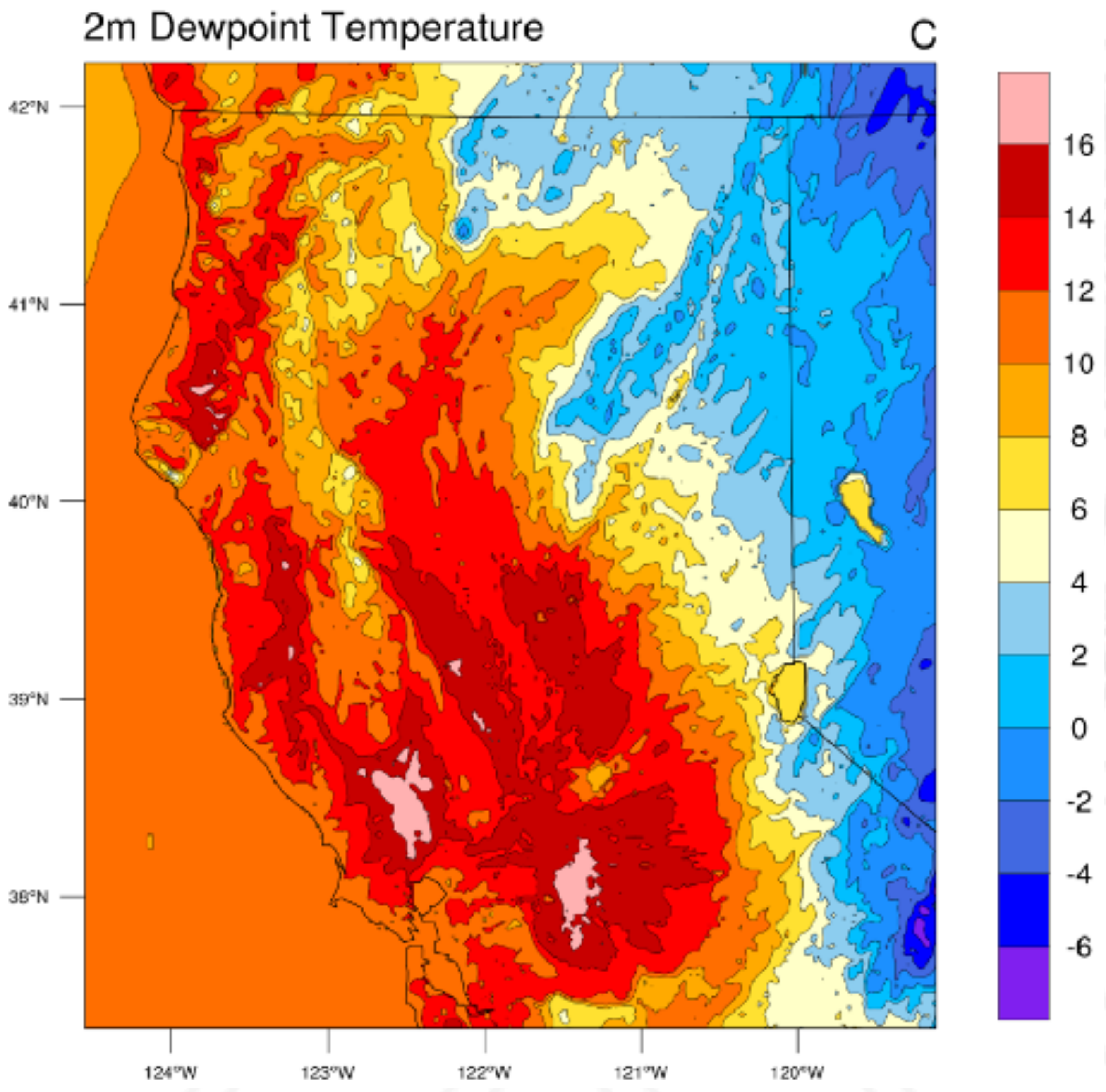
HPC stands for High Performance Computing

Drug discovery



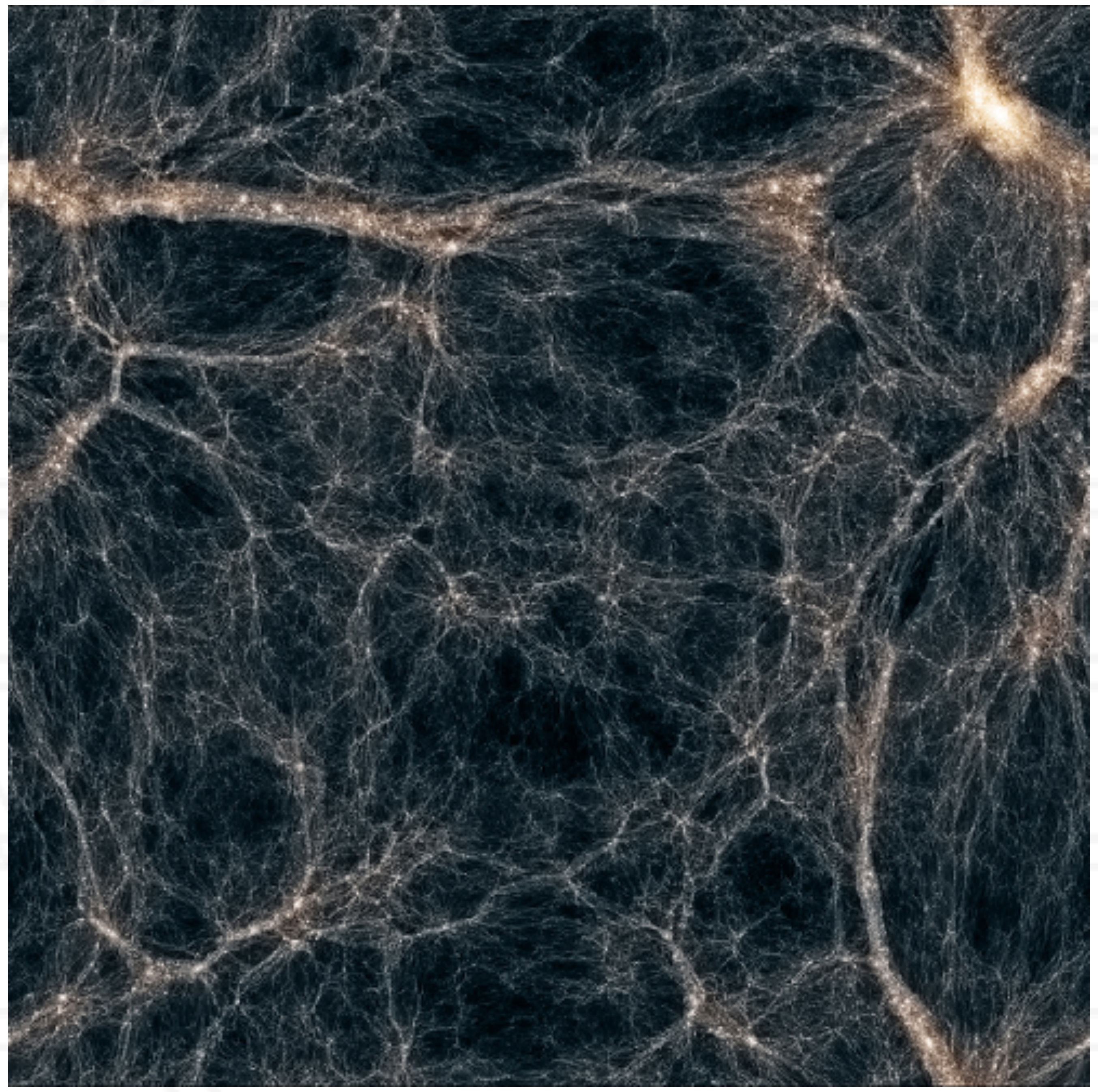
https://www.nature.com/articles/nature21414

Weather forecasting



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

Study of the universe



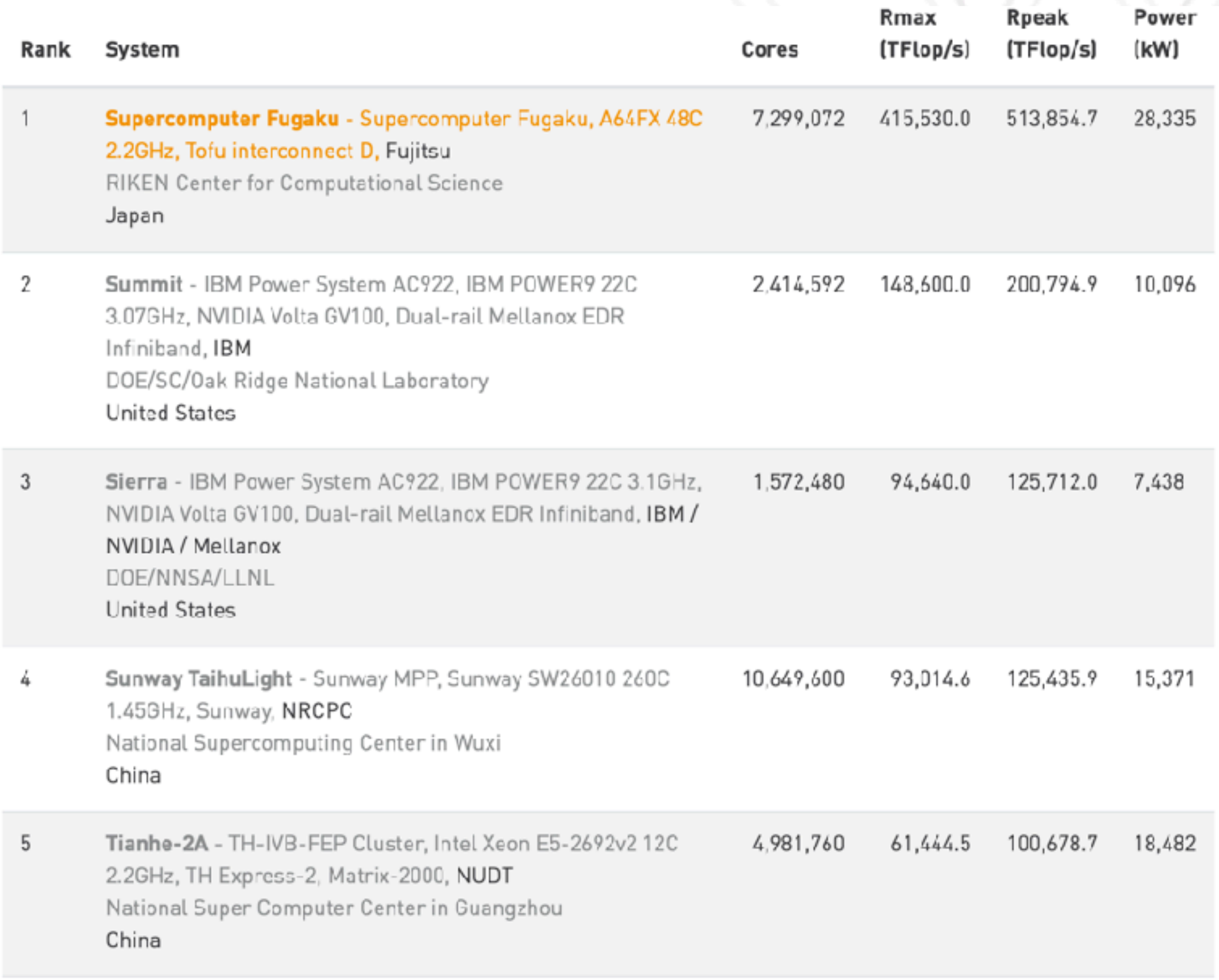
https://www.nas.nasa.gov/SC14/demos/demo27.html

Why do we need parallelism?

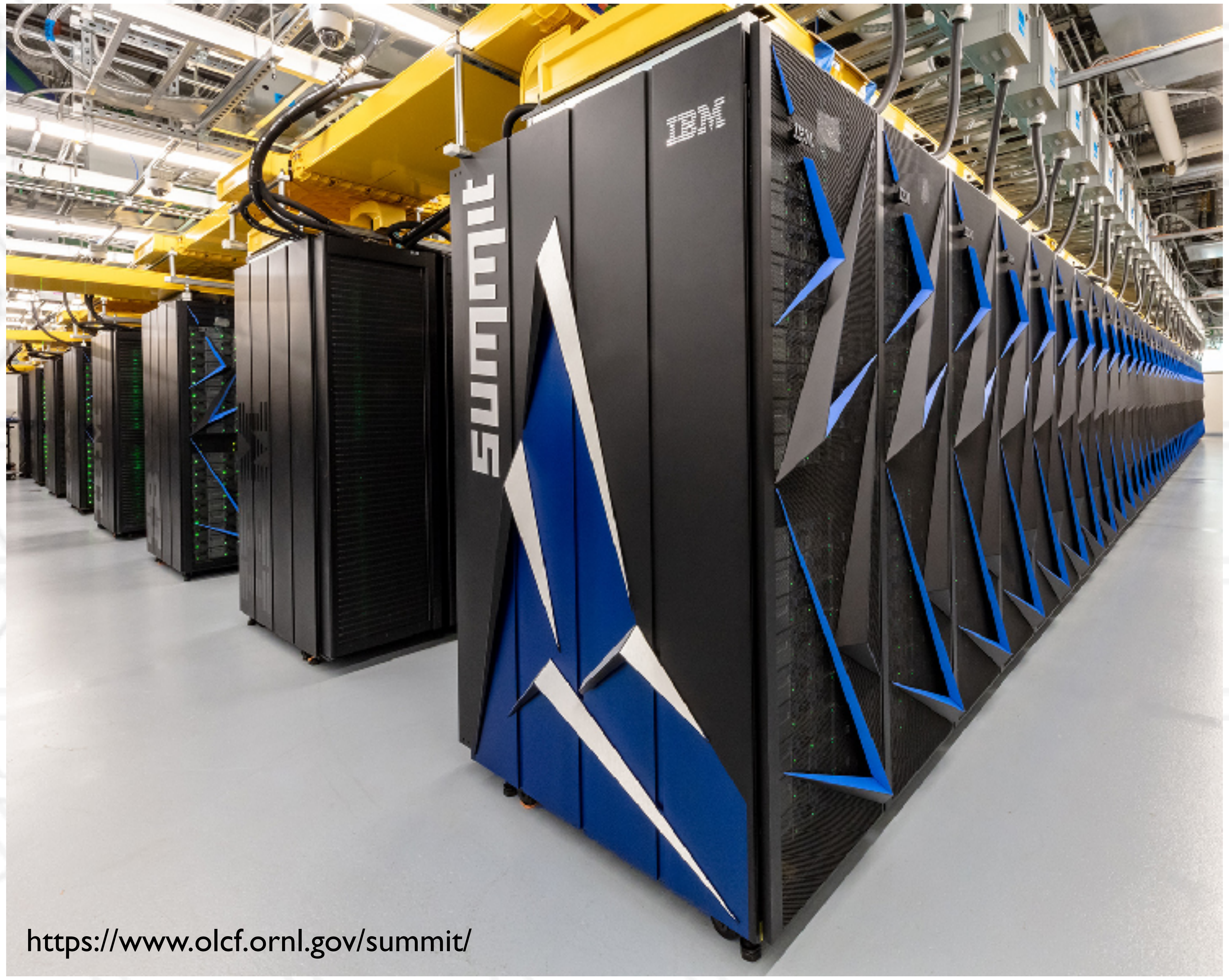
- Make some science simulations feasible in the lifetime of humans
    - Either due to speed or memory requirements
- Provide answers in realtime or near realtime

Large supercomputers

- Top500 list: https://www.top500.org/lists/top500/2020/06/



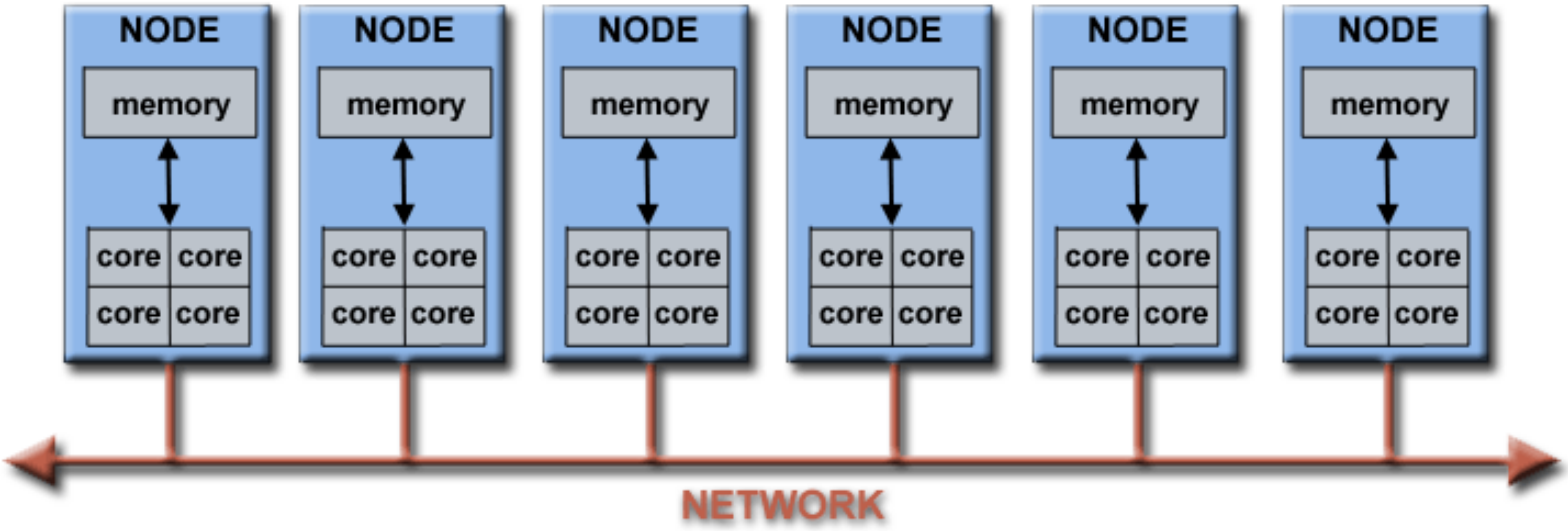
Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482



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

Parallel architecture

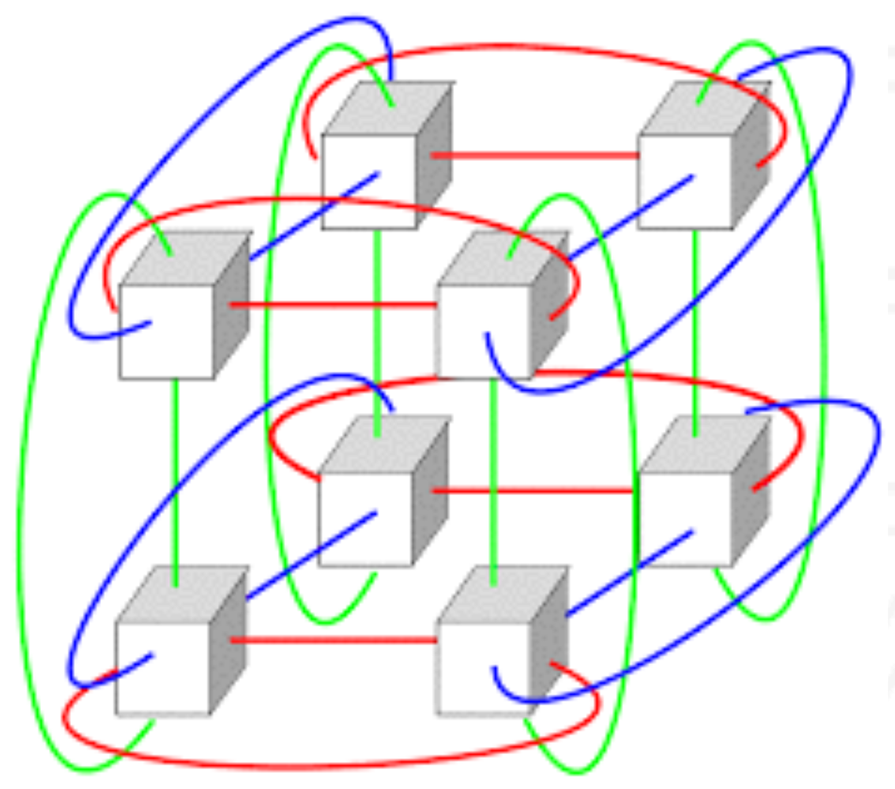
- A set of nodes or processing elements connected by a network.



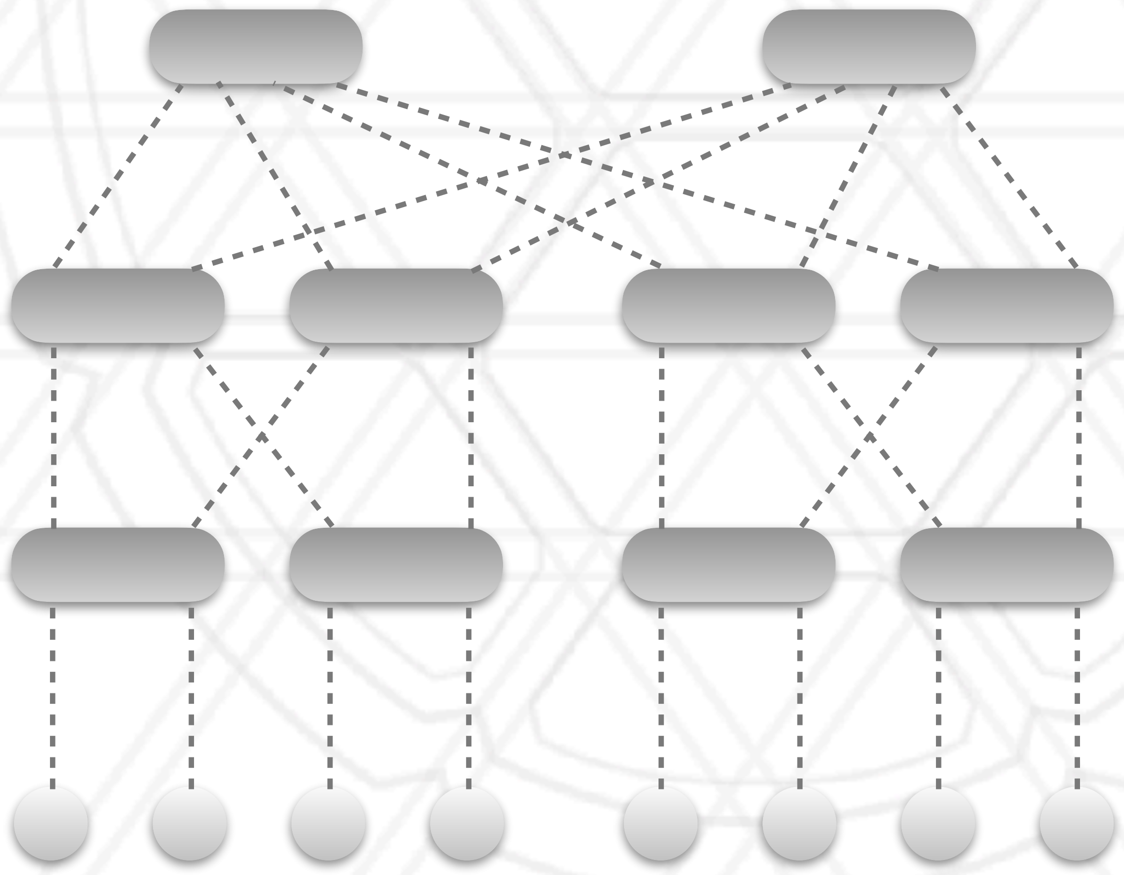
https://computing.llnl.gov/tutorials/parallel_comp

Interconnection networks

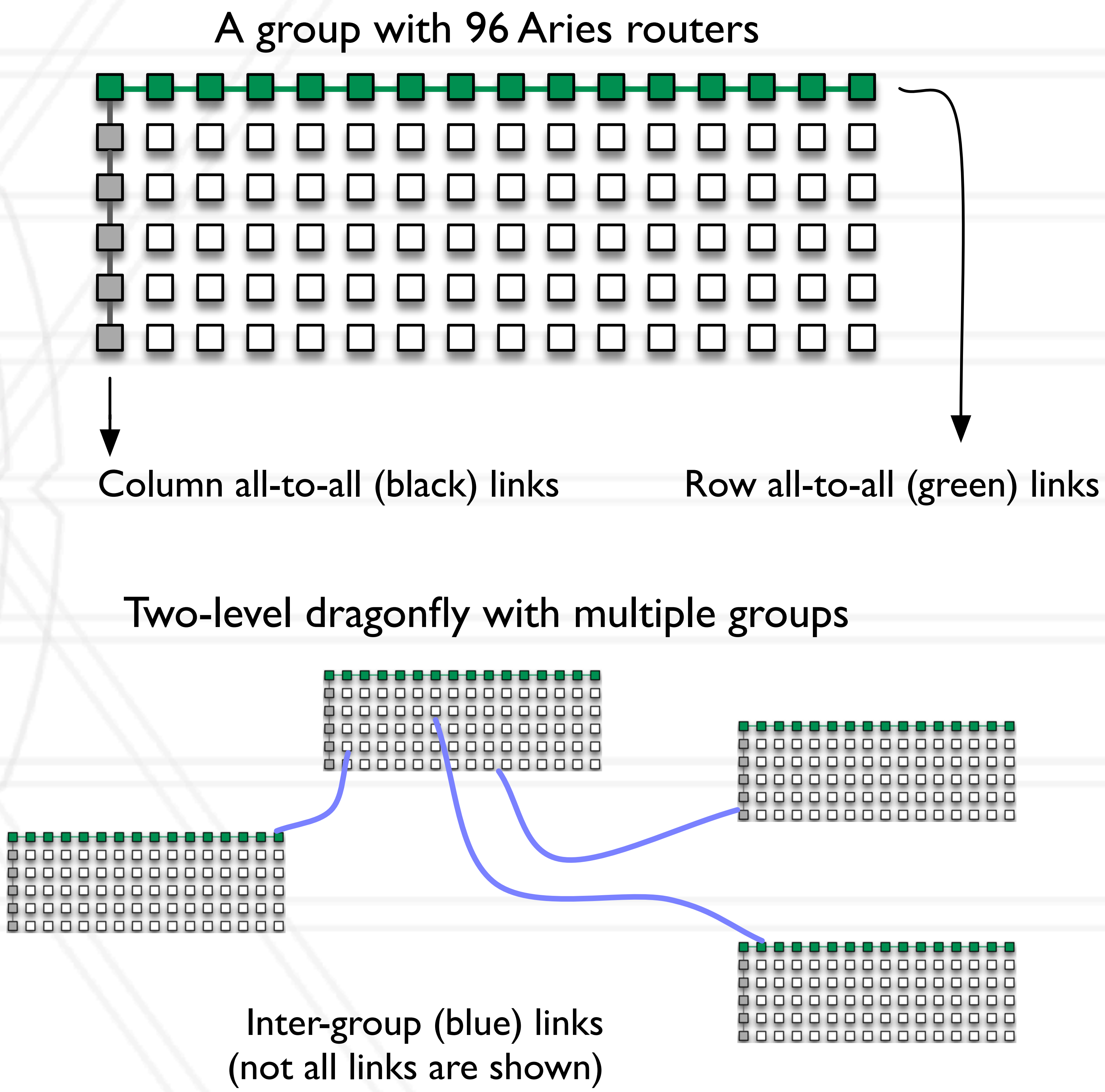
- Different topologies for connecting nodes together
- Used in the past: torus, hypercube
- More popular currently: fat-tree, dragonfly



Torus



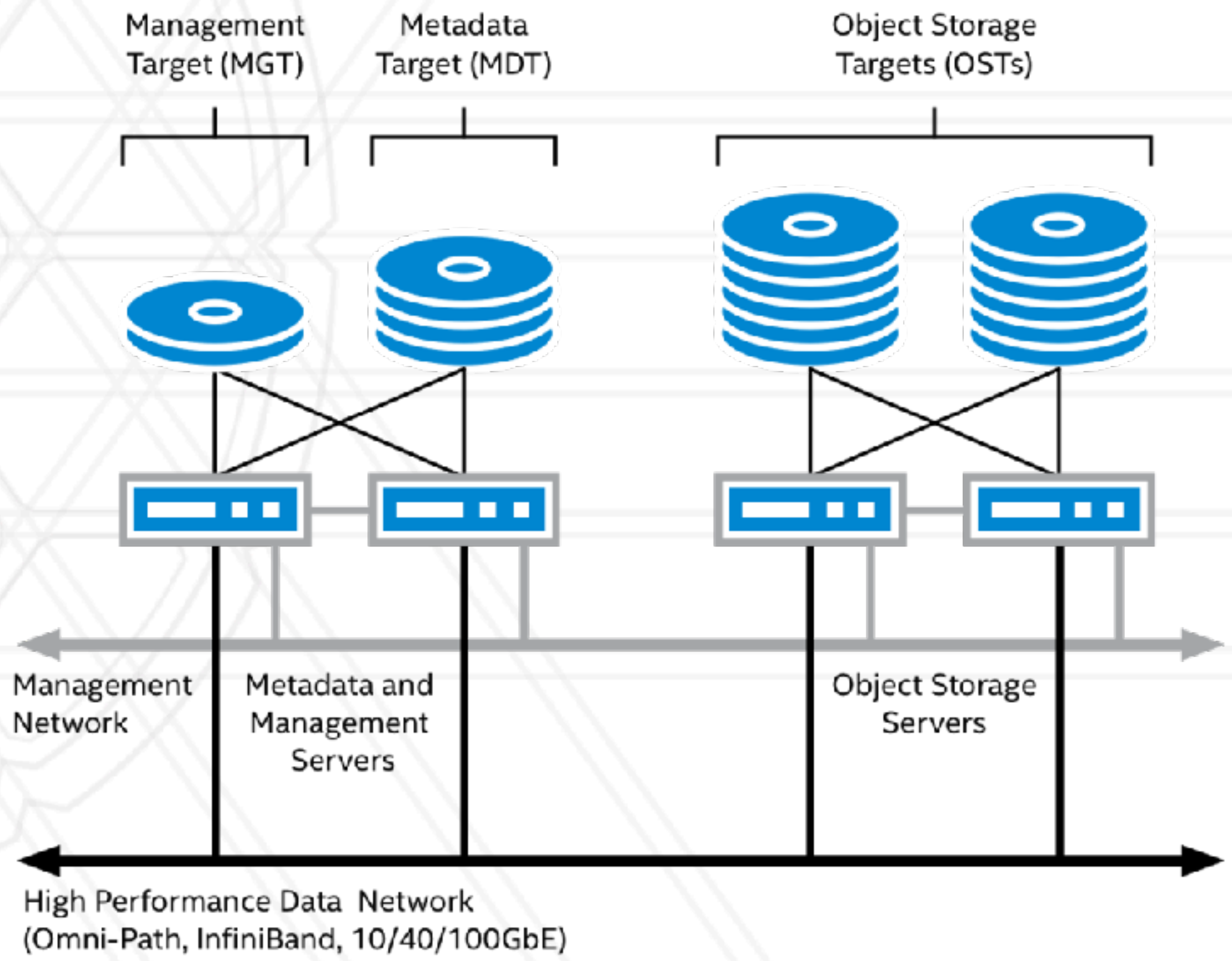
Fat-tree


A group with 96 Aries routers
Column all-to-all (black) links
Row all-to-all (green) links
Two-level dragonfly with multiple groups
Inter-group (blue) links
(not all links are shown)

Dragonfly

I/O sub-system / Parallel file system

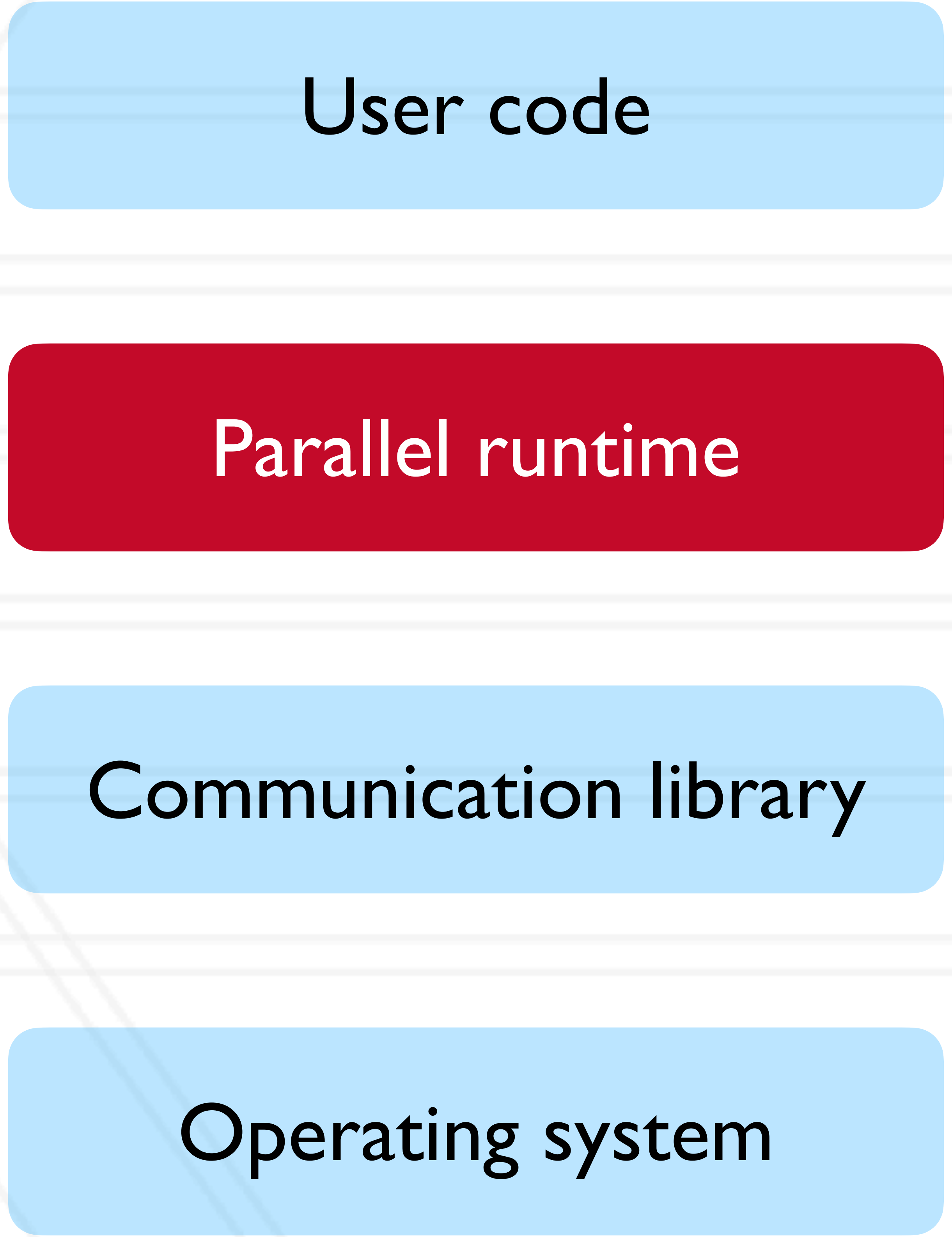
- Home directories and scratch space typically on a parallel file system
- Mounted on all login and compute nodes


Management Target (MGT)
Metadata Target (MDT)
Object Storage Targets (OSTs)
Management Network
Metadata and Management Servers
Object Storage Servers
High Performance Data Network (Omni-Path, InfiniBand, 10/40/100GbE)

http://wiki.lustre.org/Introduction_to_Lustre

System software: models and runtimes

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



Introduction to Parallel Computing (CMSC416 / CMSC818X)

Terminology and Definitions

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

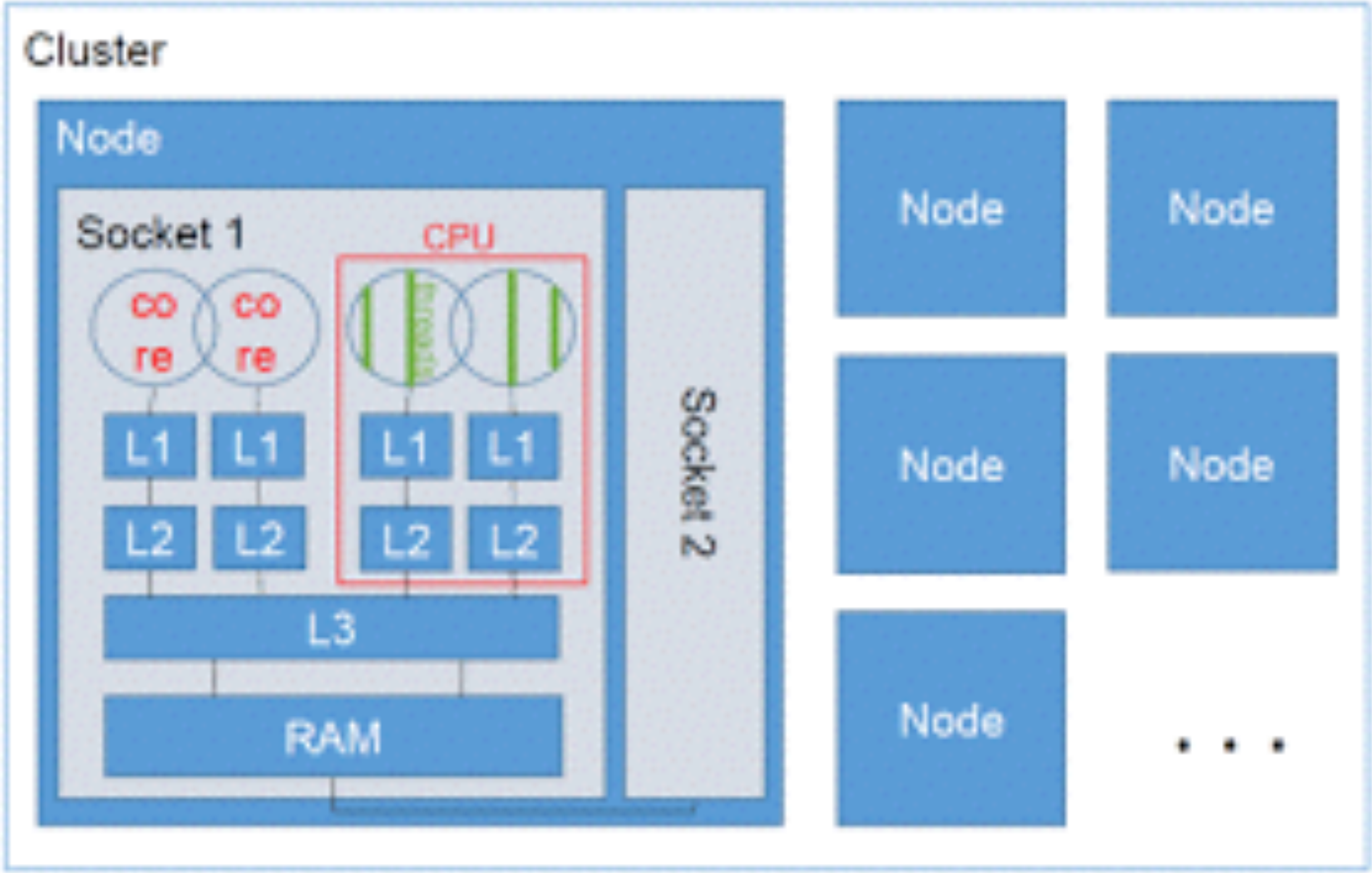
Cores, sockets, nodes

- 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

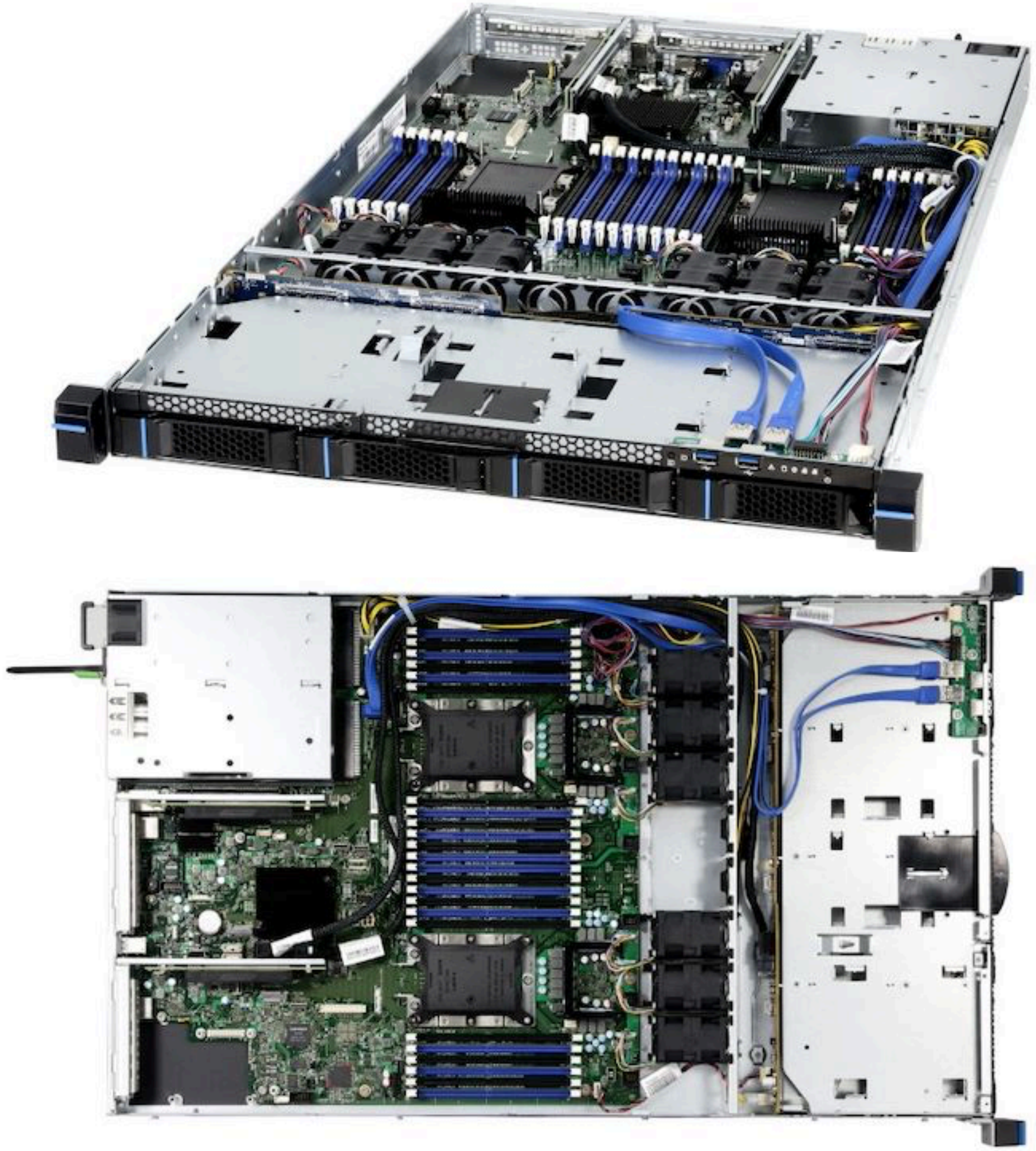


https://hpc-wiki.info/hpc/HPC-Dictionary

Rackmount servers

Rackmount servers

Rackmount server motherboard

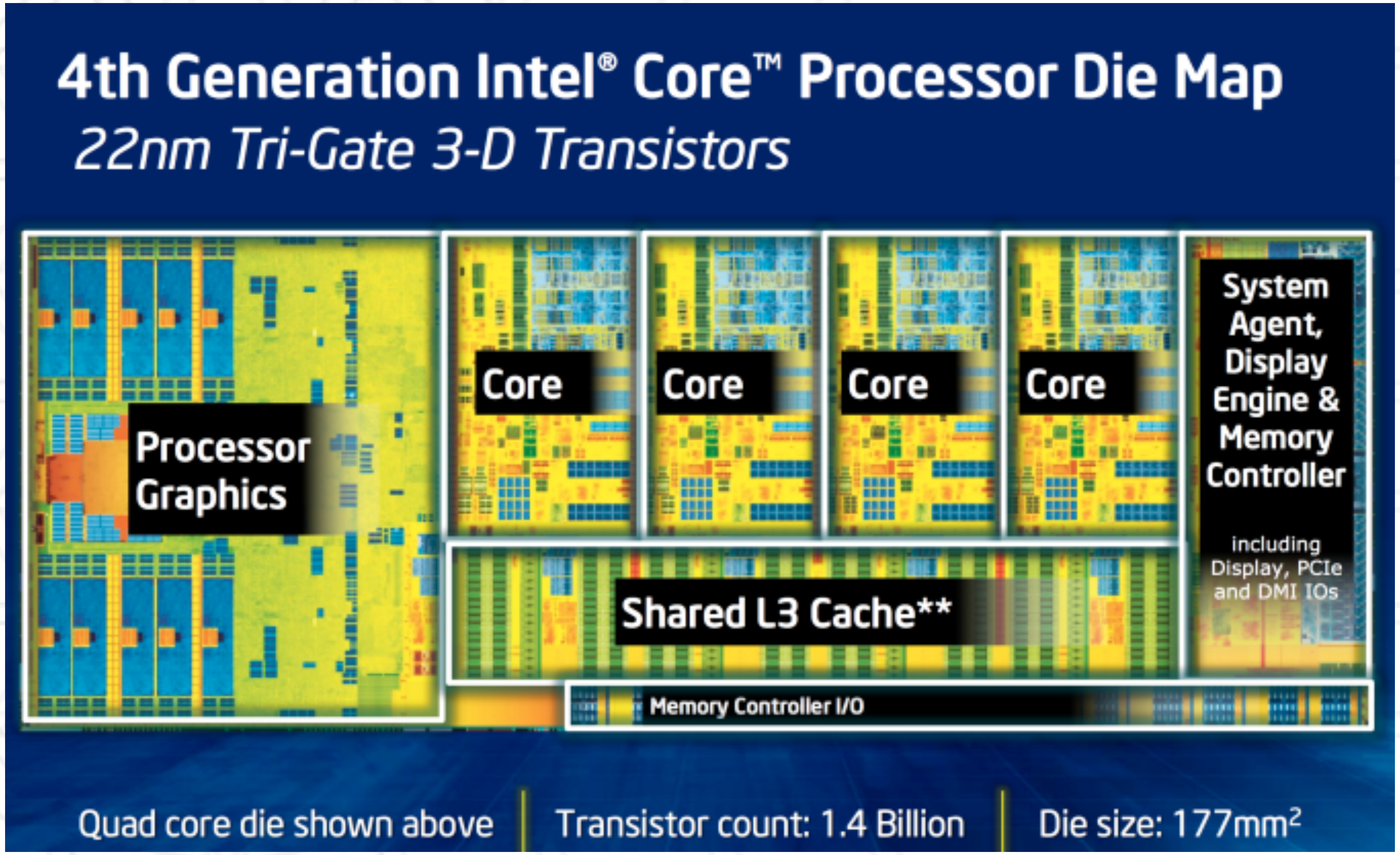


https://www.anandtech.com/show/15924/chenbro-announces-rb13804-dual-socket-1u-xeon-4-bay-hpc-barebones-server

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


4th Generation Intel® Core™ Processor Die Map
22nm Tri-Gate 3-D Transistors
Processor Graphics
Core
Core
Core
Core
System Agent, Display Engine & Memory Controller
including Display, PCIe and DMI IOs
Shared L3 Cache**
Memory Controller I/O
Quad core die shown above
Transistor count: 1.4 Billion
Die size: 177mm²

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

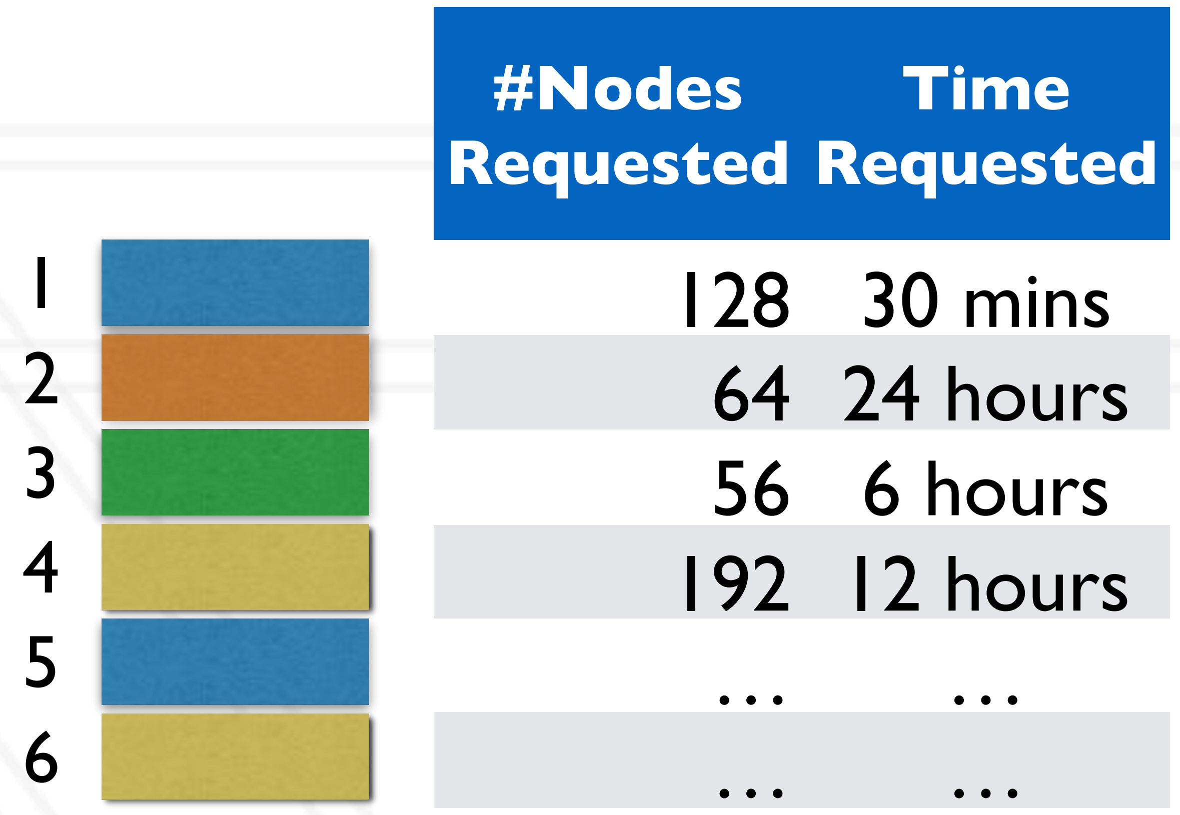
		#Nodes Requested	Time Requested
1	🟦	128	30 mins
2	🟧	64	24 hours
3	🟩	56	6 hours
4	🟨	192	12 hours
5	🟦	…	…
6	🟨	…	…

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.

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

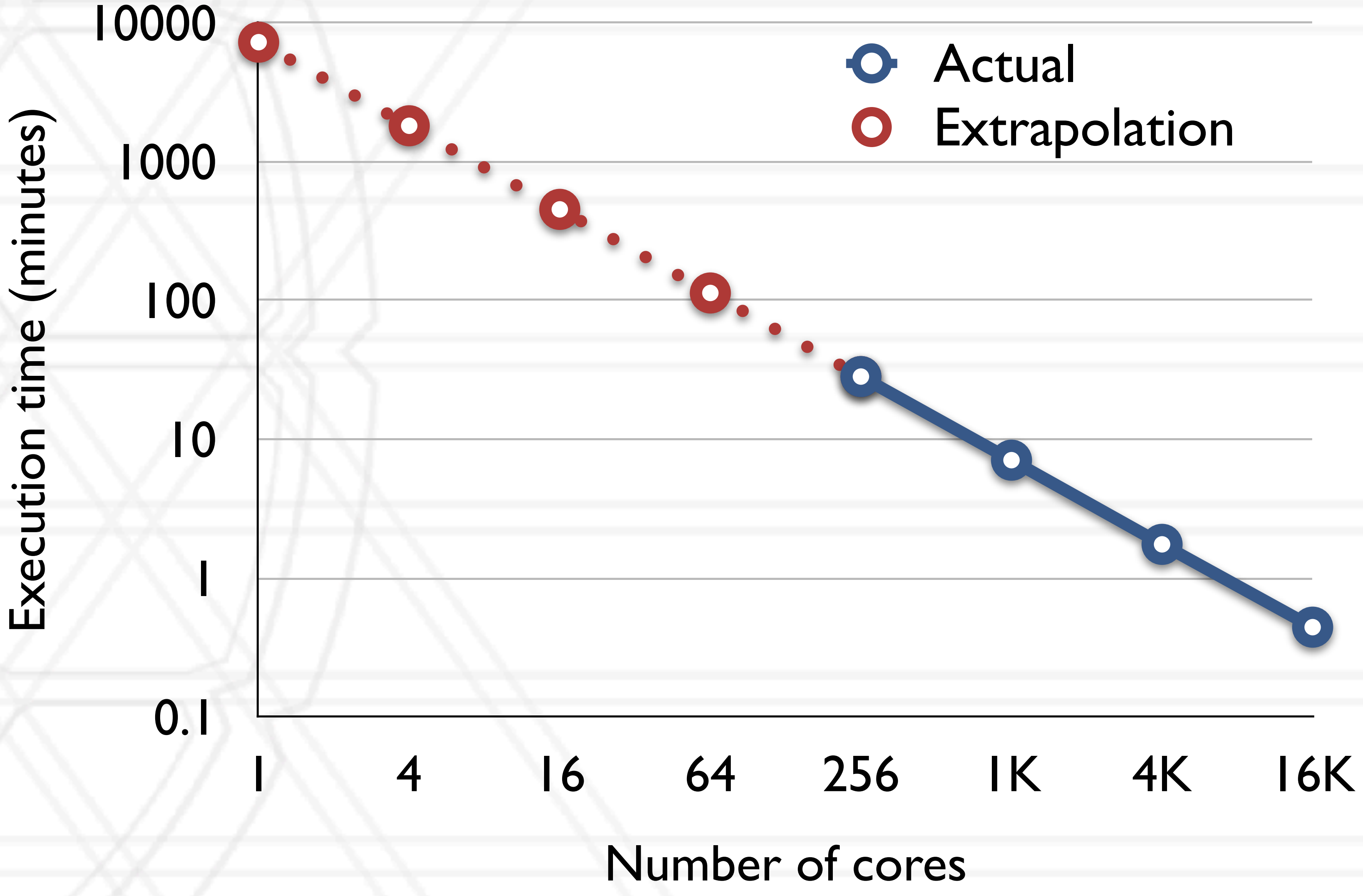
- Thread: a thread or path of execution managed by the OS
  - Threads share the same memory address space
- Process: heavy-weight, processes do not share resources such as memory, file descriptors etc.
- Serial or sequential code: can only run on a single thread or process
- Parallel code: can be run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



Weak versus strong scaling

- Strong scaling: *Fixed total* problem size as we run on more processes
  - 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 processes
  - 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

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

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

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

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 together

Different models of parallel computation

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

Getting started with deepthought2

- 444 Ivy Bridge nodes with 20 cores/node
- 40 Ivy Bridge nodes with 20 cores/node and 2 NVIDIA Tesla K20m GPUs

UNIVERSITY OF MARYLAND

Abhinav Bhatele

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

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