

CME 213, ME 339—Spring 2021  
Introduction to parallel computing using MPI, openMP, and CUDA

Eric Darve, ICME, Stanford



"Computers are useless. They can only give you answers." (Pablo Picasso)

# Homework 1

Pre-requisite homework

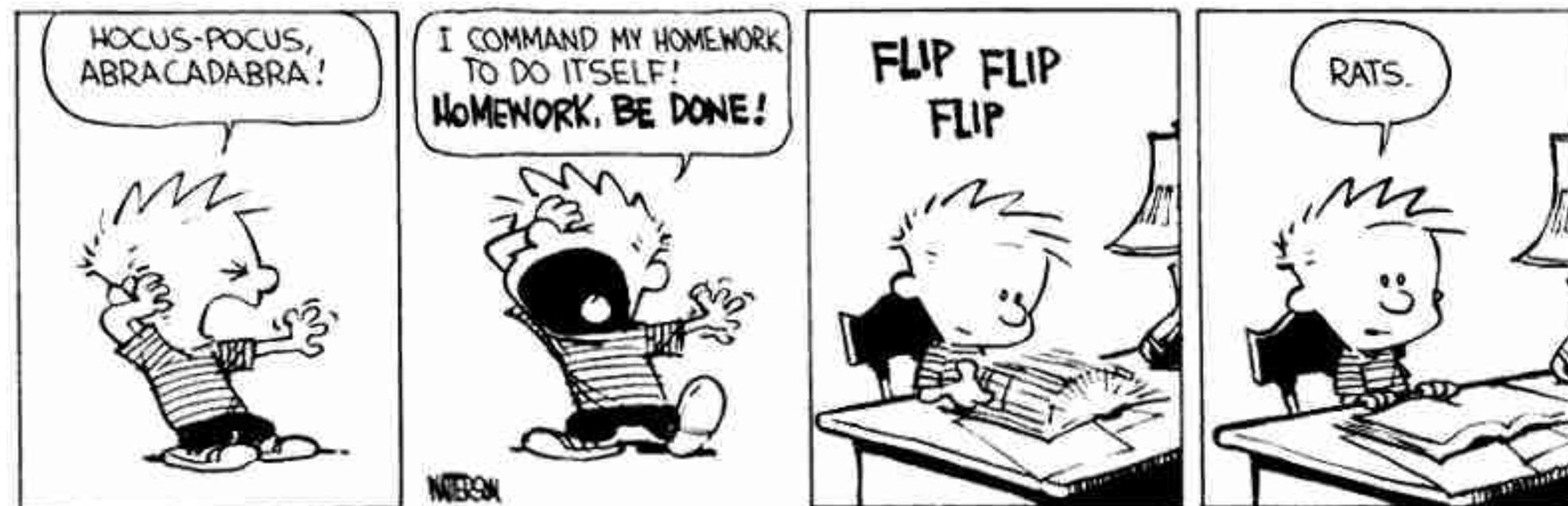
Topics:

- derived classes
- polymorphism
- standard library
- testing

# Submission

1. Submit your PDF on gradescope
2. For your computer code, copy your files to cardinal
3. Run a Python script it submit code

Grading is done on gradescope



**Deadline is Friday, March 9**

# Why parallel computing?



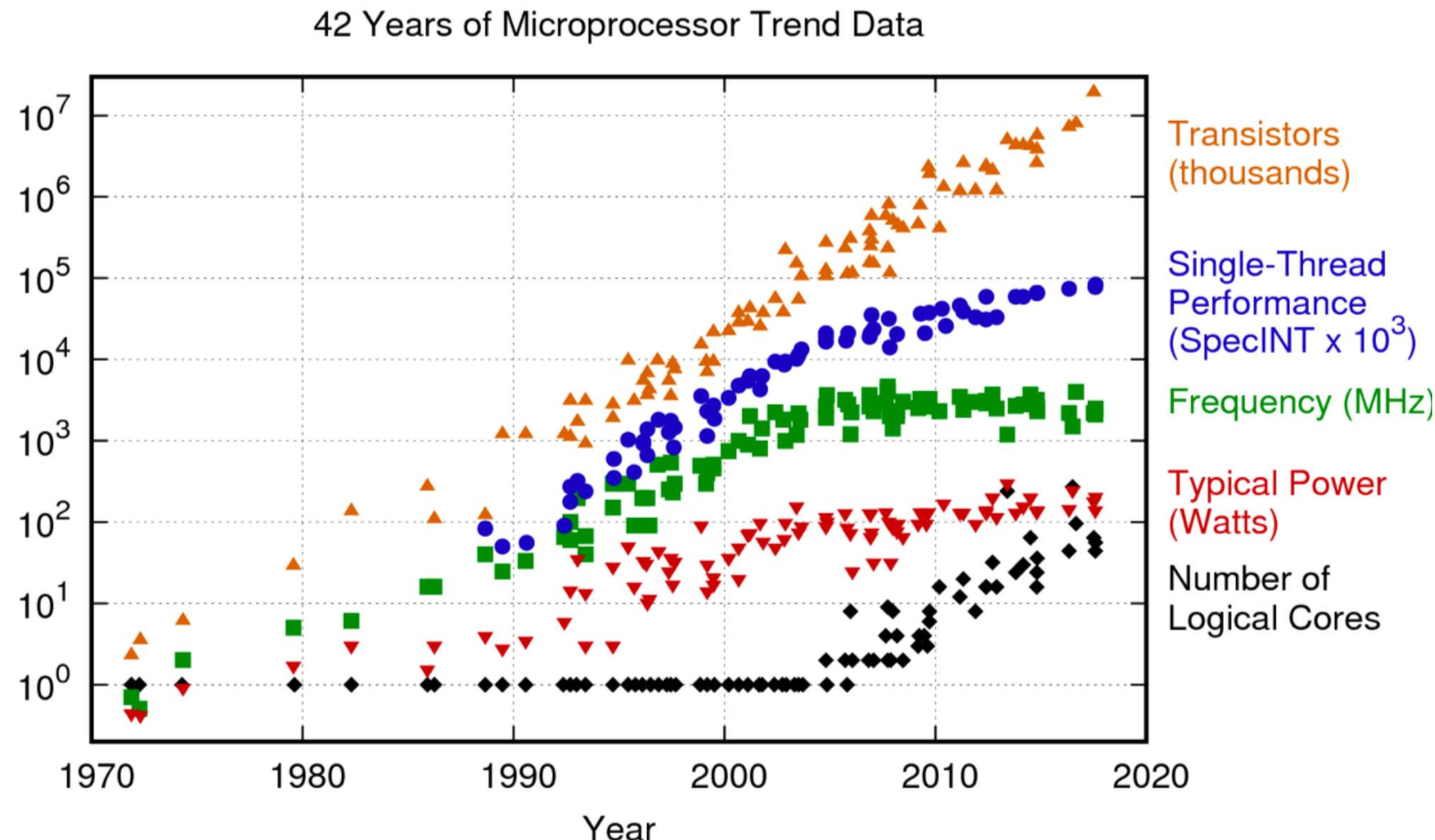
Parallel computing is omni-present

Any type of non-trivial computing requires parallel computing



- Gordon Moore 1965: "the number of transistors on a chip shall double every 18–24 months."
- Accompanied by an increase in clock speed

# Intel microprocessor trends



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten  
New plot and data collected for 2010-2017 by K. Rupp

But

Increase in transistor density is limited by:

- Leakage current increases
- Power consumption increases
- Heat generated increases

Memory access time has not been reduced at a rate comparable to the processing speed



Go parallel!

Multiple cores on a processor

## Multicore



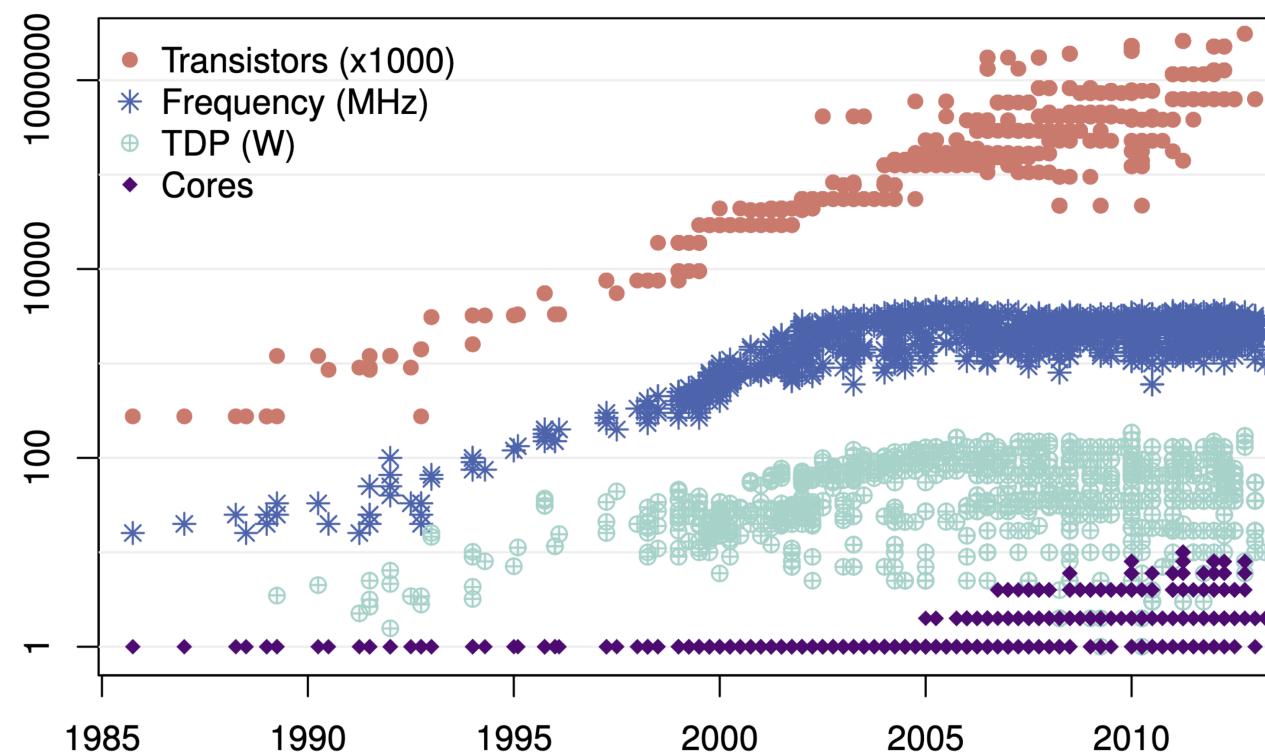
One/few  
but  
fast core(s)

## Manycore

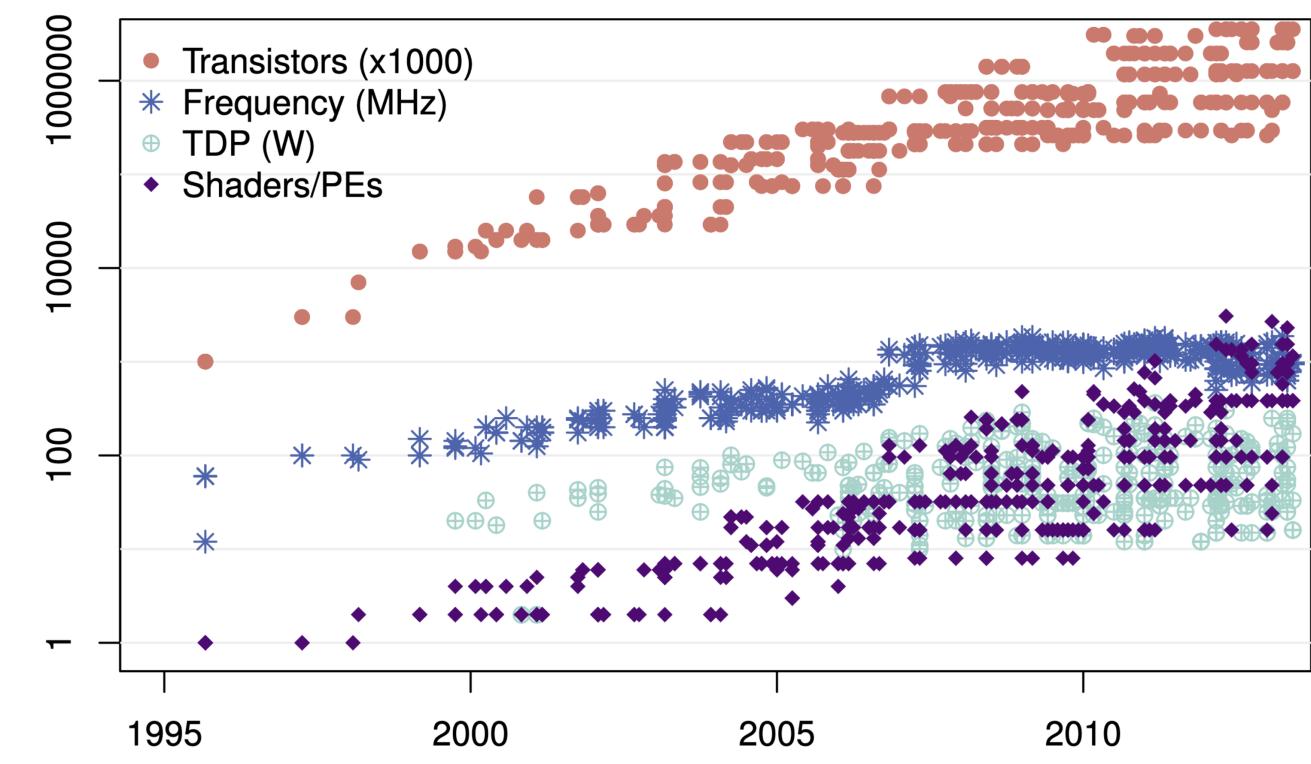


- Many, but slower cores
- GPUs

## Core increase; frequency plateau

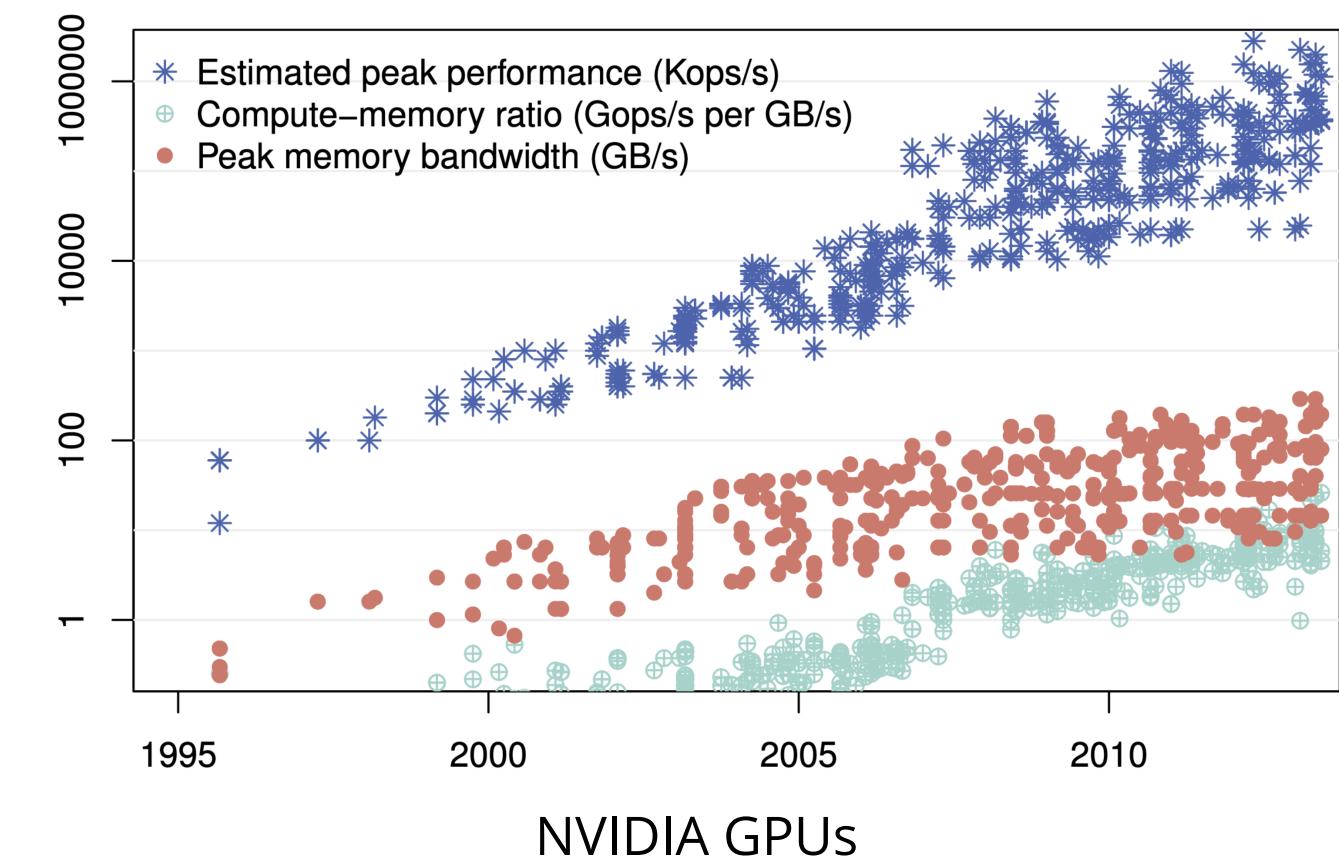
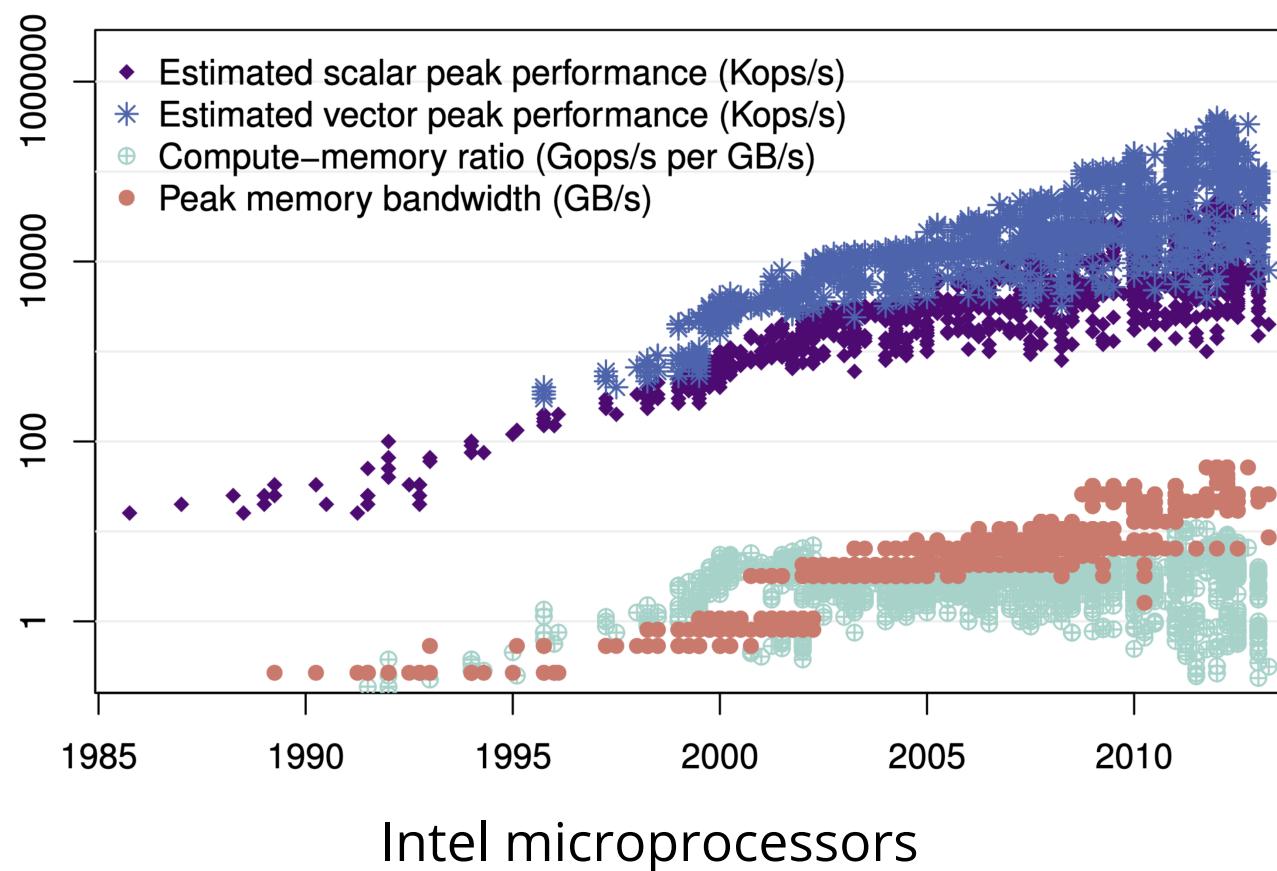


Historical data on 1403 Intel microprocessors



Historical data on 566 NVIDIA GPUs

## Memory wall; bandwidth and latency



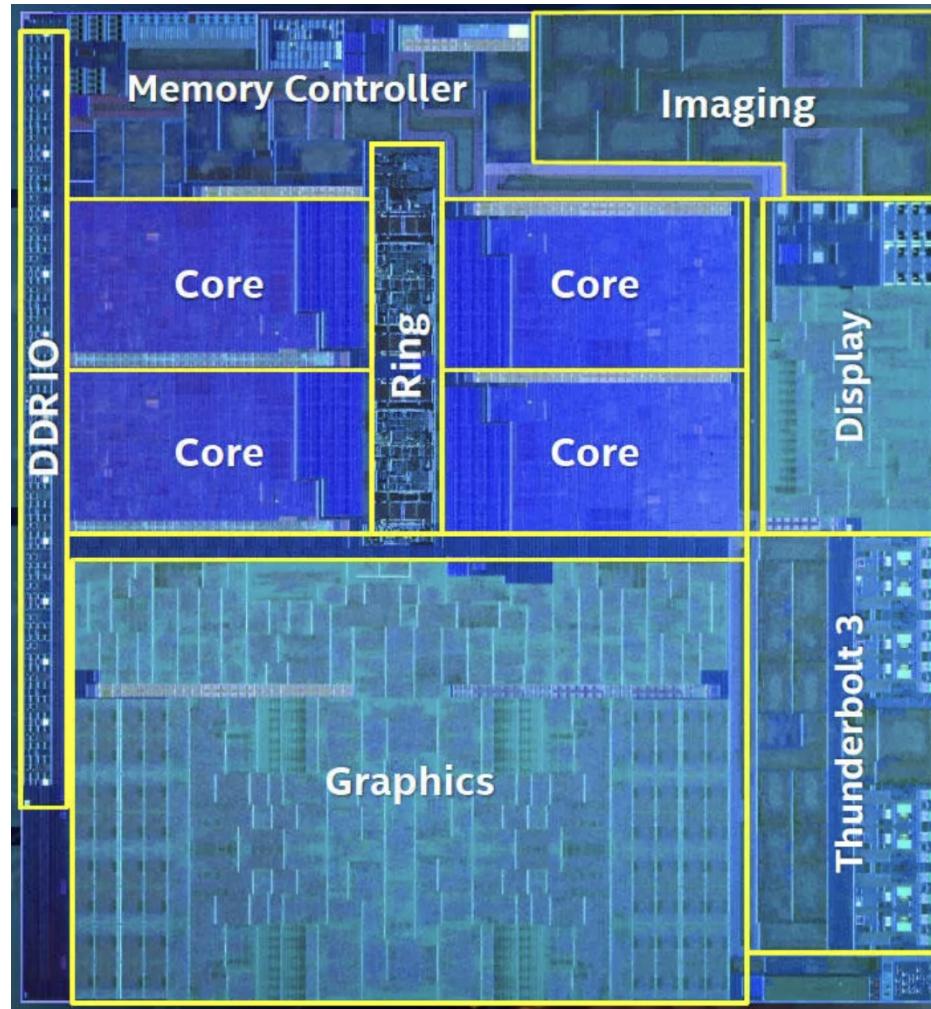
More info at

<https://pure.tue.nl/ws/portalfiles/portal/3942529/771987.pdf>

Parallel computing everywhere!



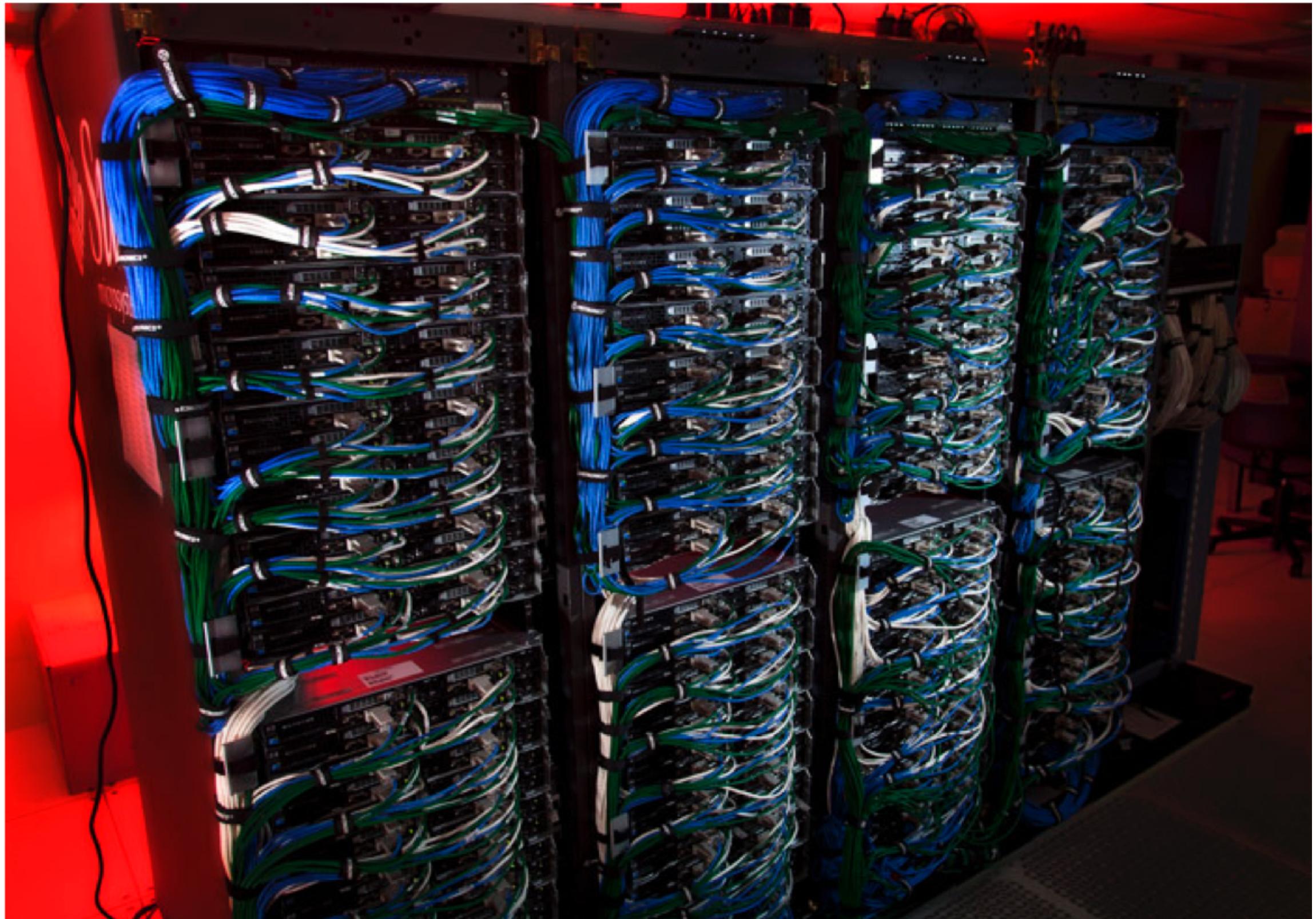
# Multi and many core processors

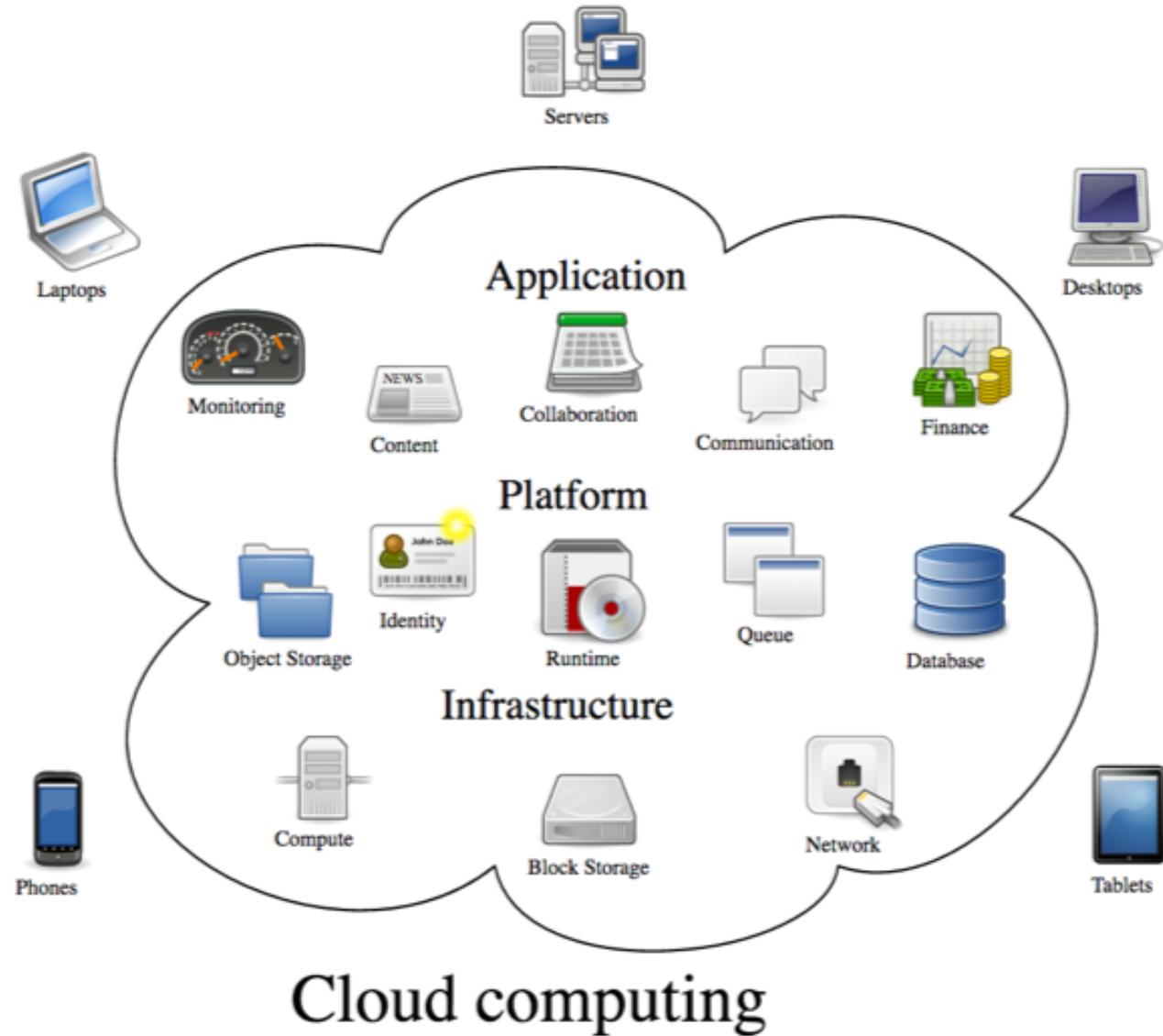


Intel Ice Lake 10 nm

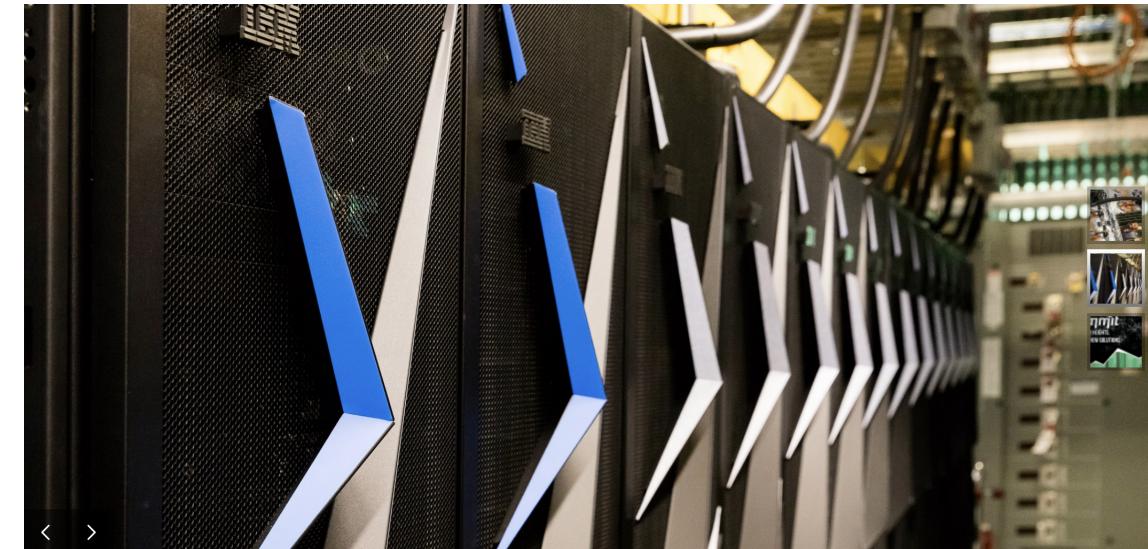


Turing TU102 architecture





## Summit—Oak Ridge National Laboratory's 200 petaflop supercomputer



**Processor:** IBM POWER9™ (2/node)

**GPUs:** 27,648 NVIDIA Volta V100s (6/node)

**Nodes:** 4,608

**Node Performance:** 42TF

**Memory/node:** 512GB DDR4 + 96GB HBM2

**NV Memory/node:** 1600GB

**Total System Memory:** >10PB DDR4 + HBM + Non-volatile

**Interconnect Topology:** Mellanox EDR 100G InfiniBand, Non-blocking

Fat Tree

**Peak Power Consumption:** 13MW

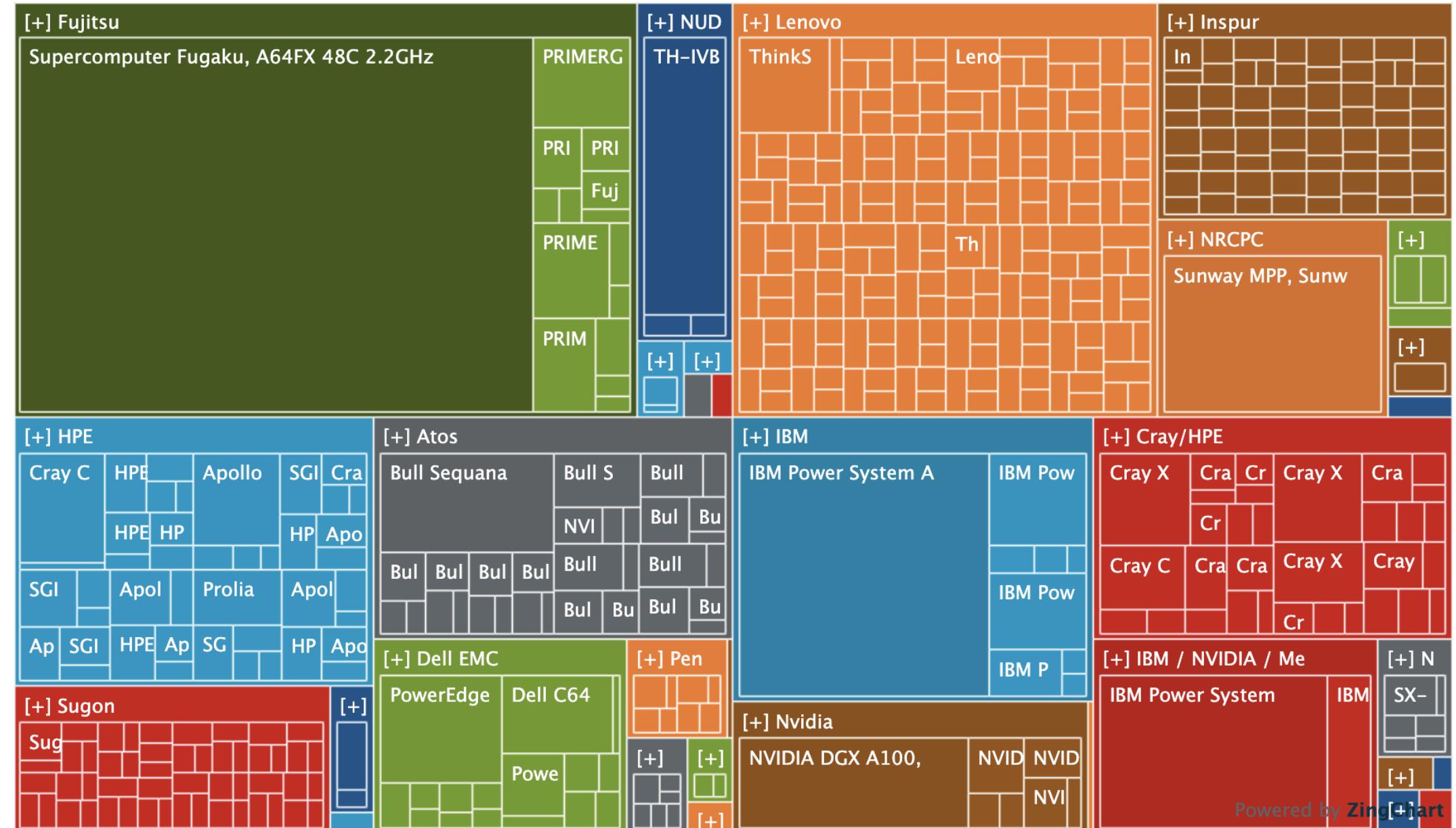
## Top 500 Supercomputers



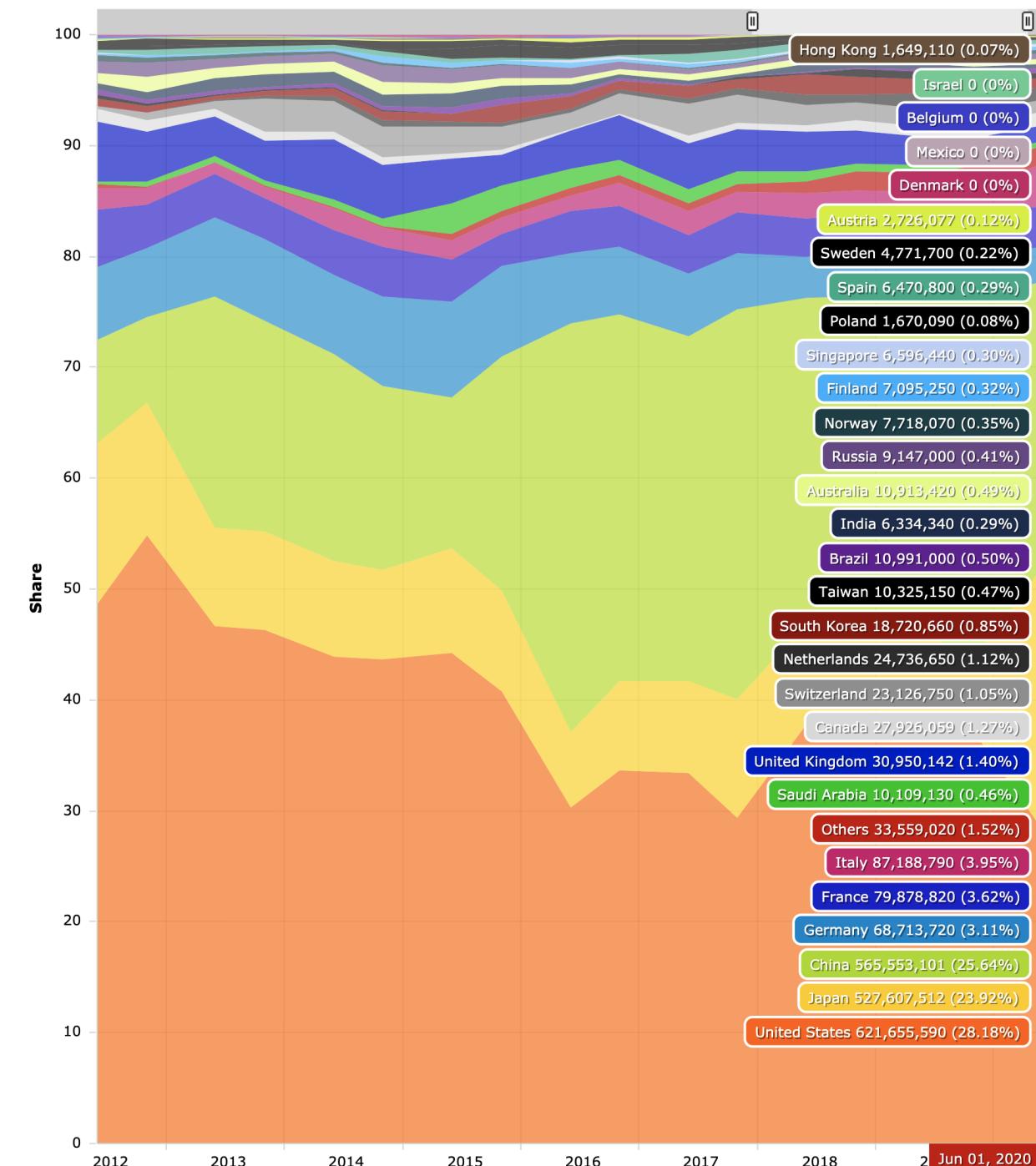
# Green 500

TOP500				Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
Rank	Rank	System					
1	170	<b>NVIDIA DGX SuperPOD</b> - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia	NVIDIA Corporation United States	19,840	2,356.0	90	26.195
2	330	<b>MN-3</b> - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks MN-Core, MN-Core DirectConnect, Preferred Networks	Preferred Networks Japan	1,664	1,652.9	65	26.039
3	7	<b>JUWELS Booster Module</b> - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos	Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	1,764	25.008
4	146	<b>Spartan2</b> - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox	HDR Infiniband, Atos	23,040	2,566.0	106	24.262
5	5	<b>Selene</b> - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia	NVIDIA Corporation United States	555,520	63,460.0	2,646	23.983

# Vendor shares



### Countries - Performance Share

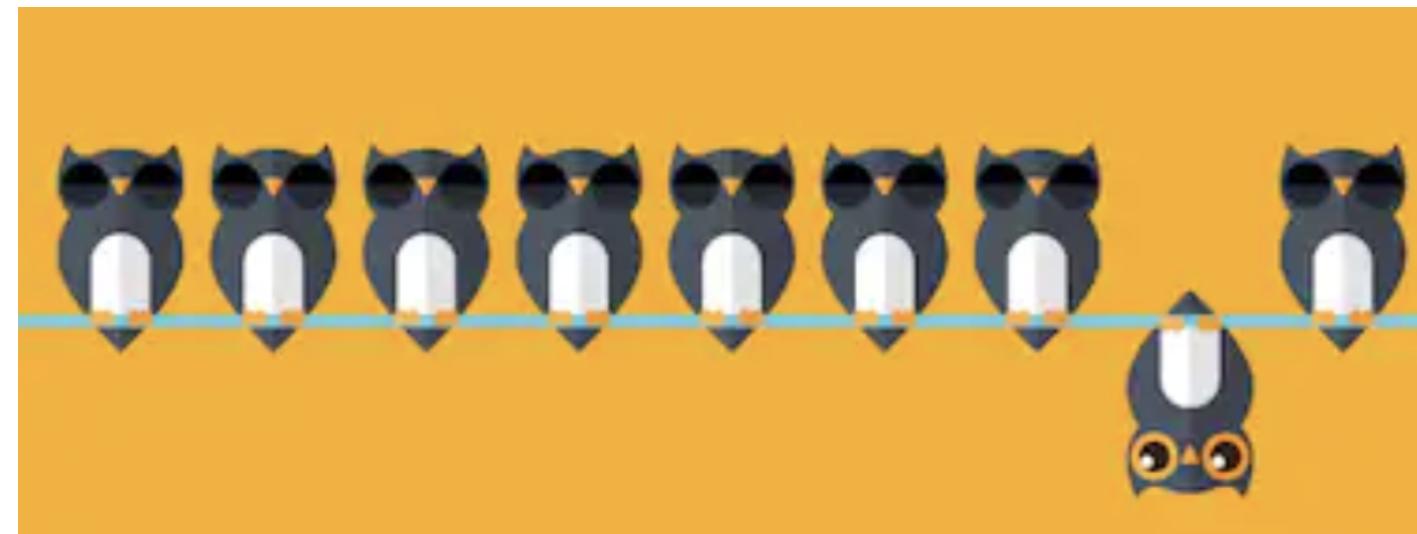


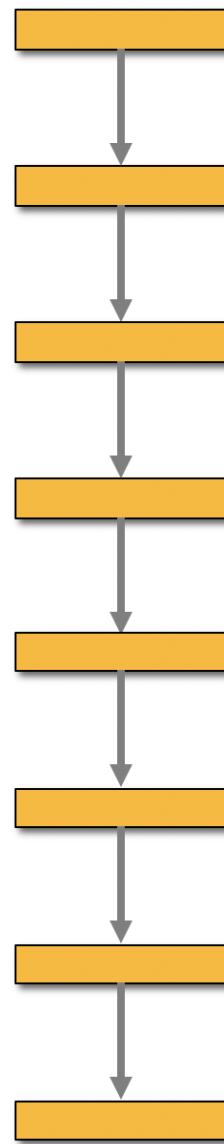
More at

<https://www.top500.org/>

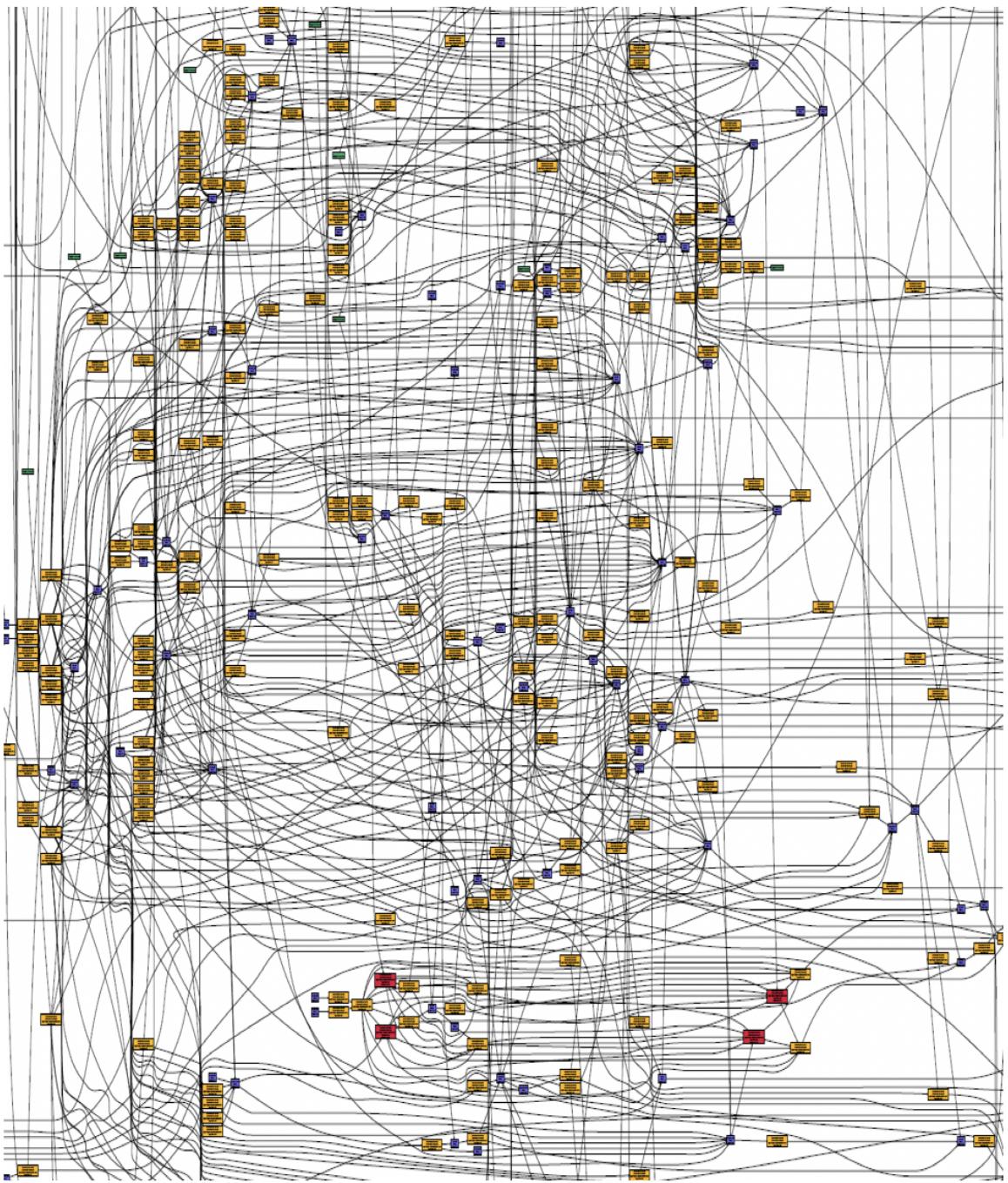
## Example of a Parallel Computation

Parallel programs often look very different from sequential programs





## Sequential



## Parallel

## Example: program to sum numbers

```
for (int i = 0; i < n; ++i)
{
    x = ComputeNextValue();
    sum += x;
}
```

We have  $\$p\$$  cores that can compute and exchange data

Can we accelerate our calculation by splitting the work among the cores?

```
int r; /* thread number */
int b; /* number of entries processed */
int my_first_i = r * b;
int my_last_i = (r + 1) * b;
for (int my_i = my_first_i; my_i < my_last_i; my_i++) {
    my_x = ComputeNextValue();
    my_sum += my_x;
}
```

Not that simple

Each core has computed a partial sum

All these partial sums need to summed up together

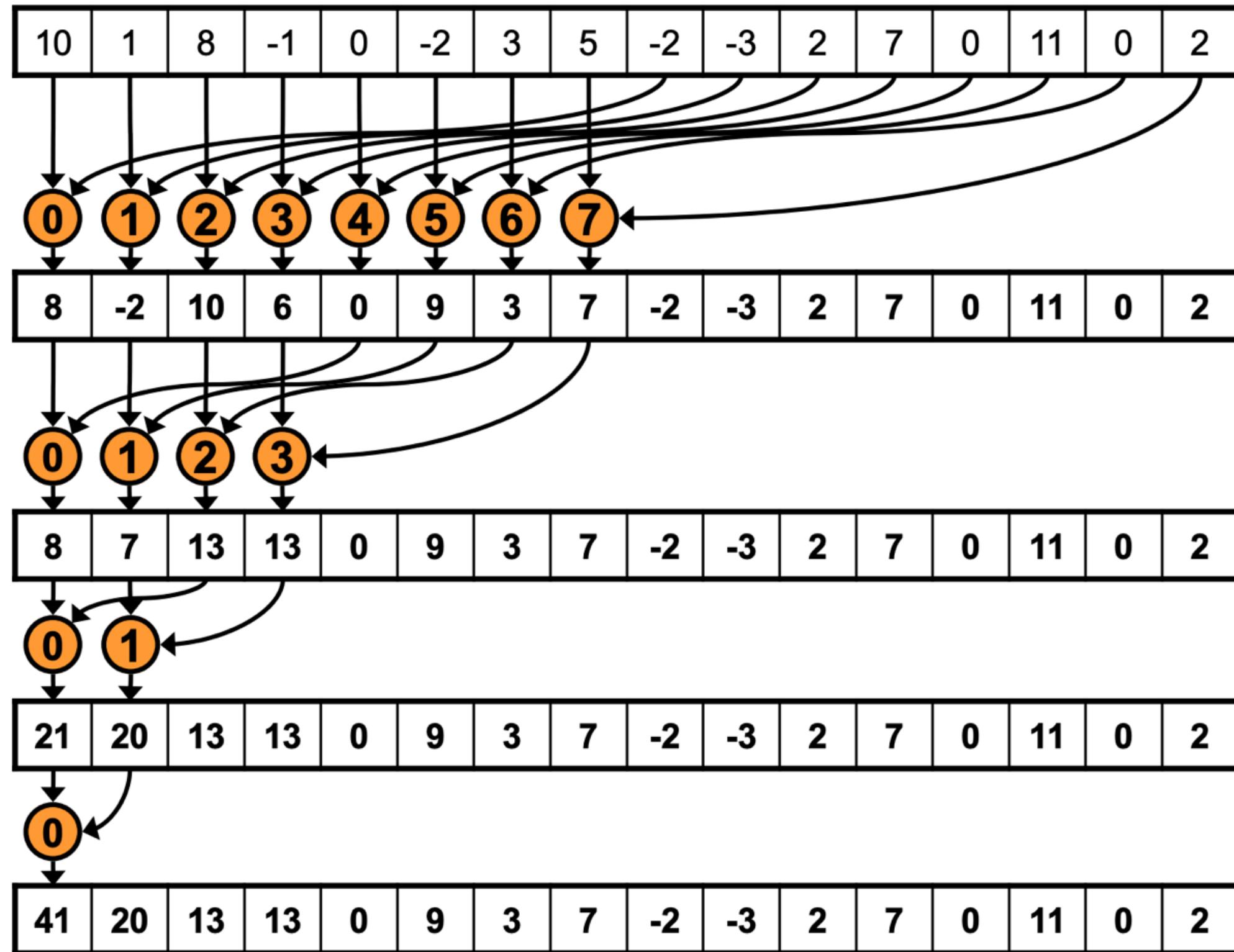
Simplest approach:  
have one "master" thread do all the work

```
if (r == 0) /* master thread */
{
    int sum = my_sum;
    for (int ro = 1; ro < p; ++ro)
    {
        int sum_ro;
        ReceiveFrom(&sum_ro, ro);
        sum += sum_ro;
    }
}
else /* worker thread */
{
    SendTo(&my_sum, 0);
}
```

That may not be enough

If we have many cores, this final sum may take a lot of time





This simple example illustrates the fact that it is difficult for a compiler to parallelize a program.

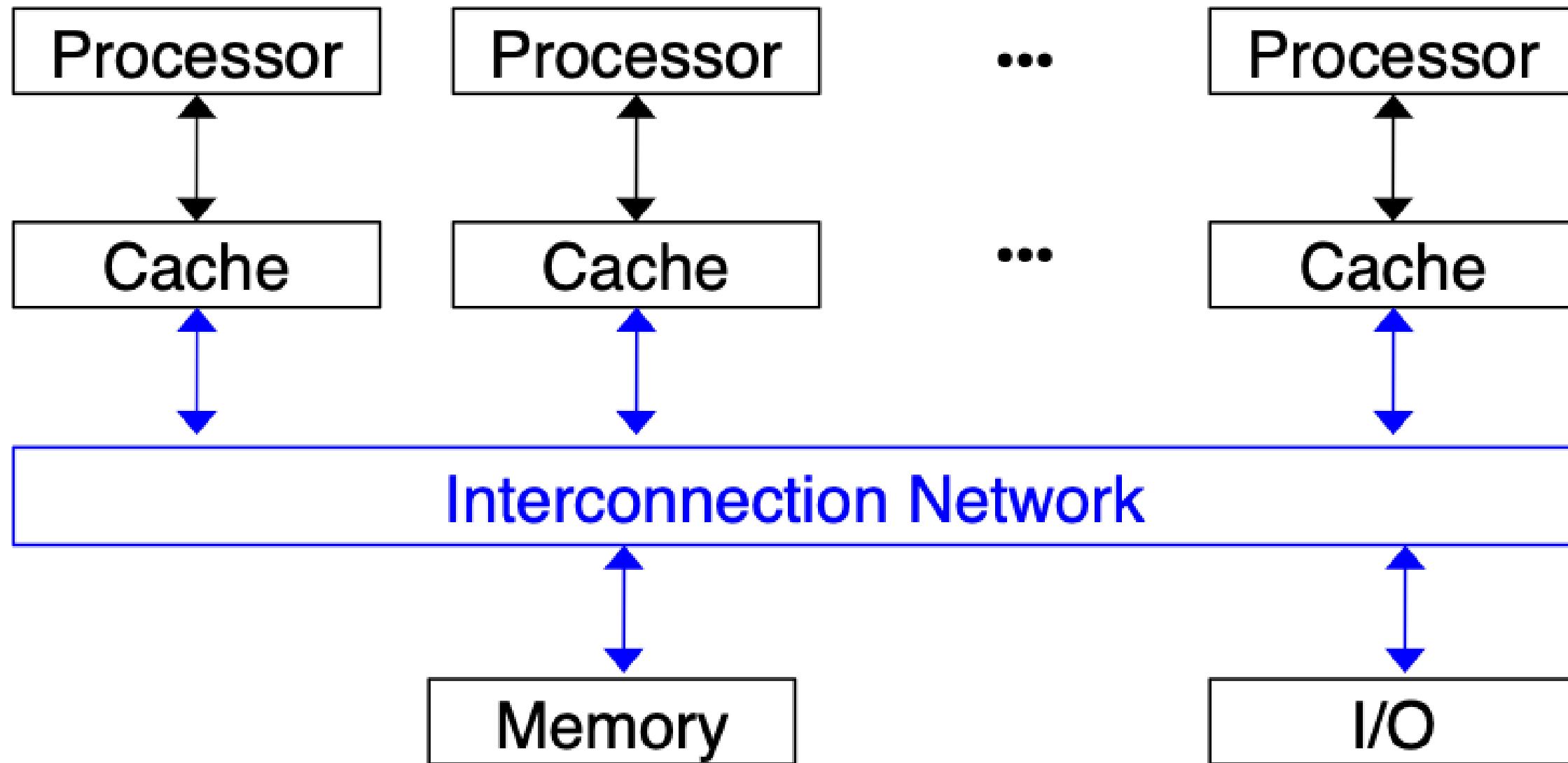
Instead the programmer must often re-write his code having in mind that multiple cores will be computing in parallel.

The purpose of this class is to teach you the most common parallel languages used in science and engineering.

# Shared Memory Processor

## Schematic

- A number of processors or cores
- A shared physical memory (global memory)
- An interconnection network to connect the processors with the memory



## Shared memory NUMA

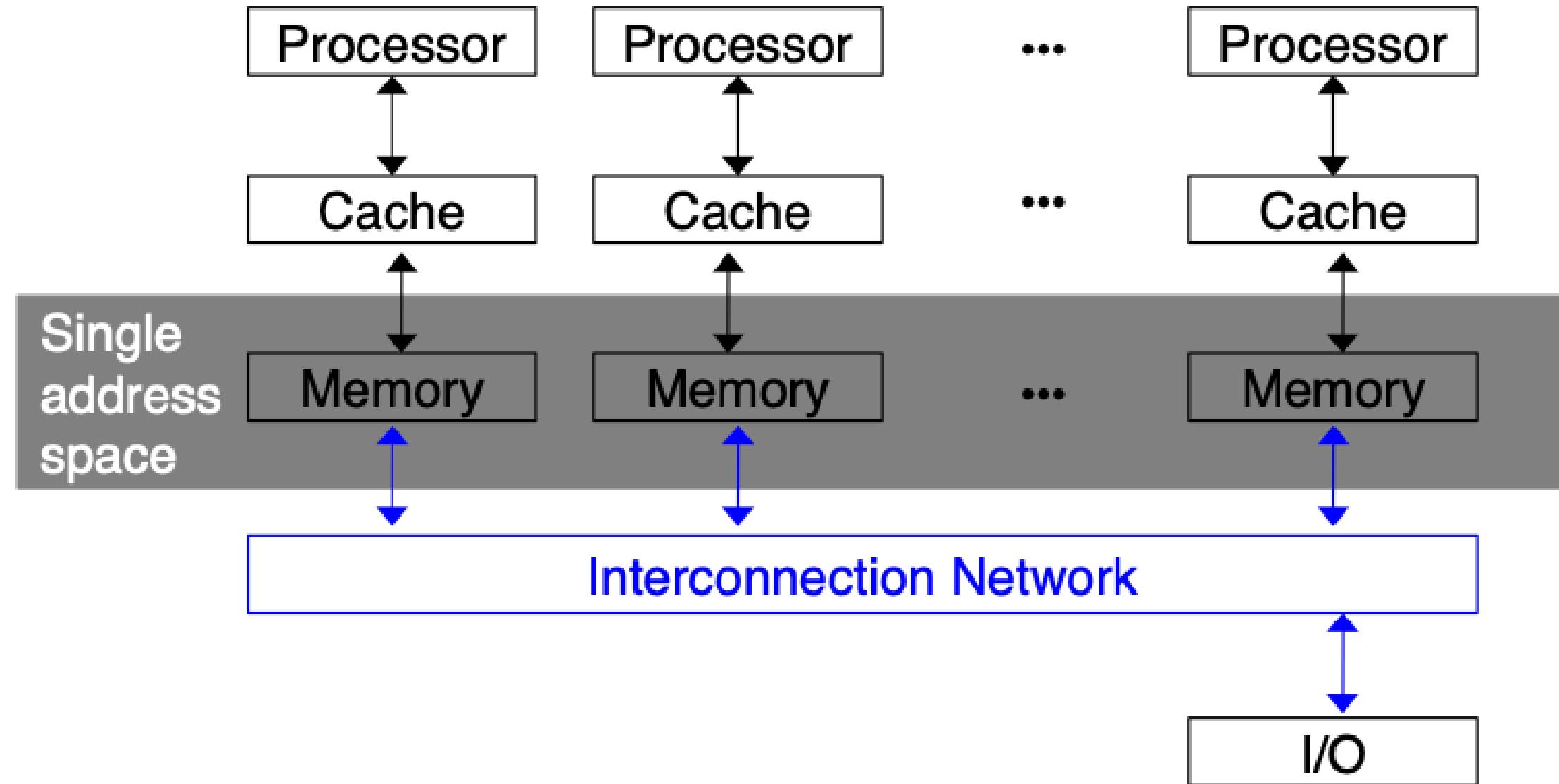
In many cases, the program views the memory as a single addressable space.

In reality, the memory is physically distributed.

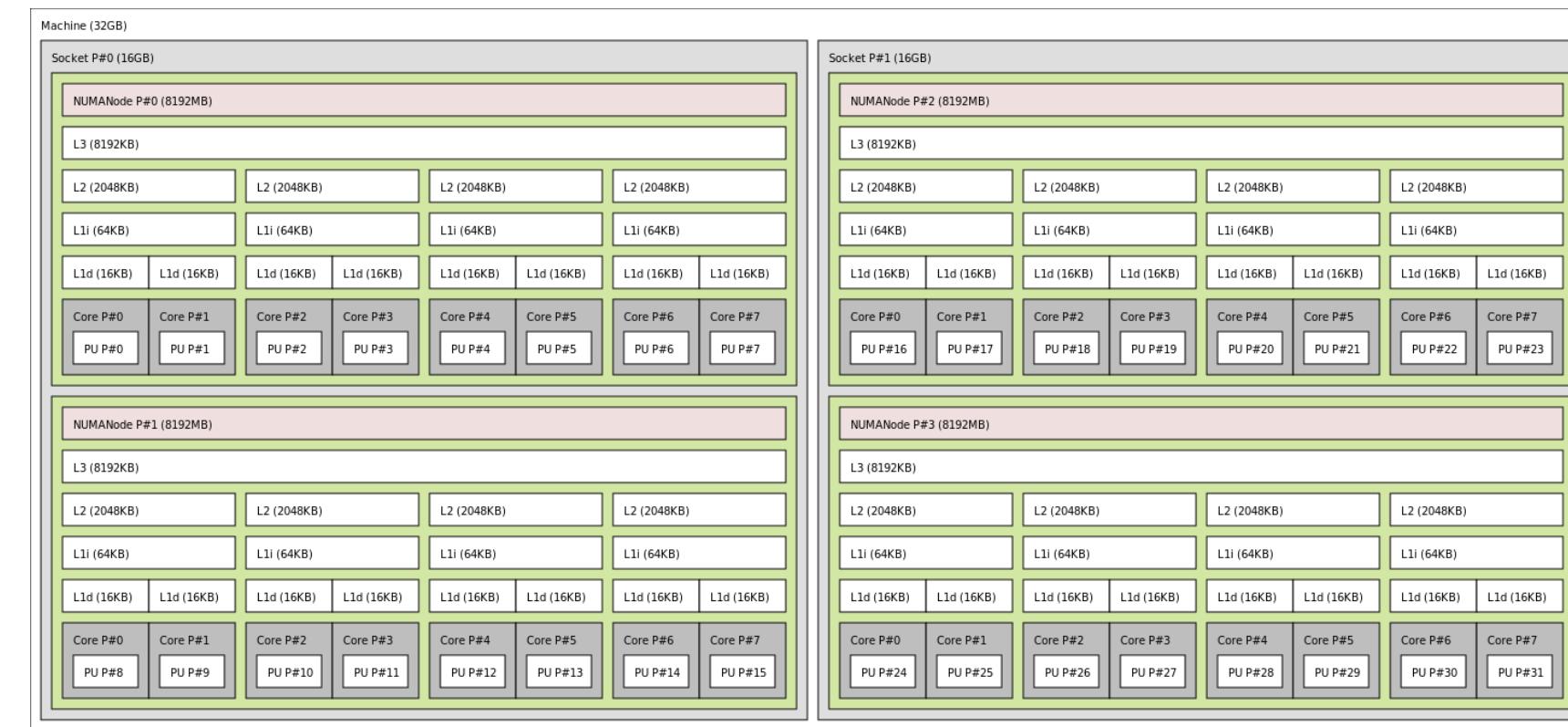
NUMA non-uniform memory access

Why? Faster access to memory

But, special hardware required to move data between memory banks

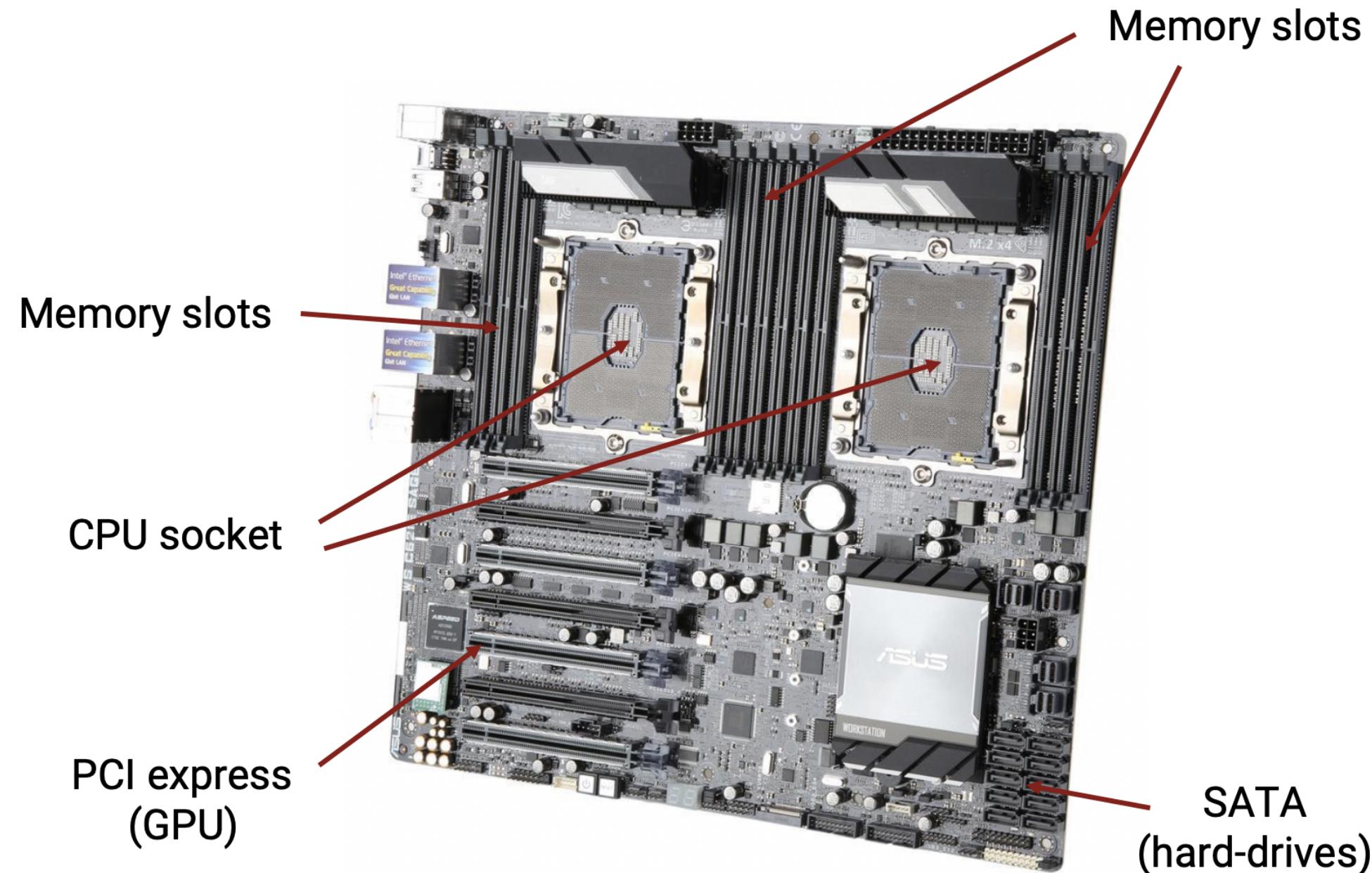


# Bulldozer server (AMD)



Cache coherent NUMA (ccNUMA) uses inter-processor communication between cache controllers to keep a consistent memory image when more than one cache stores the same memory location

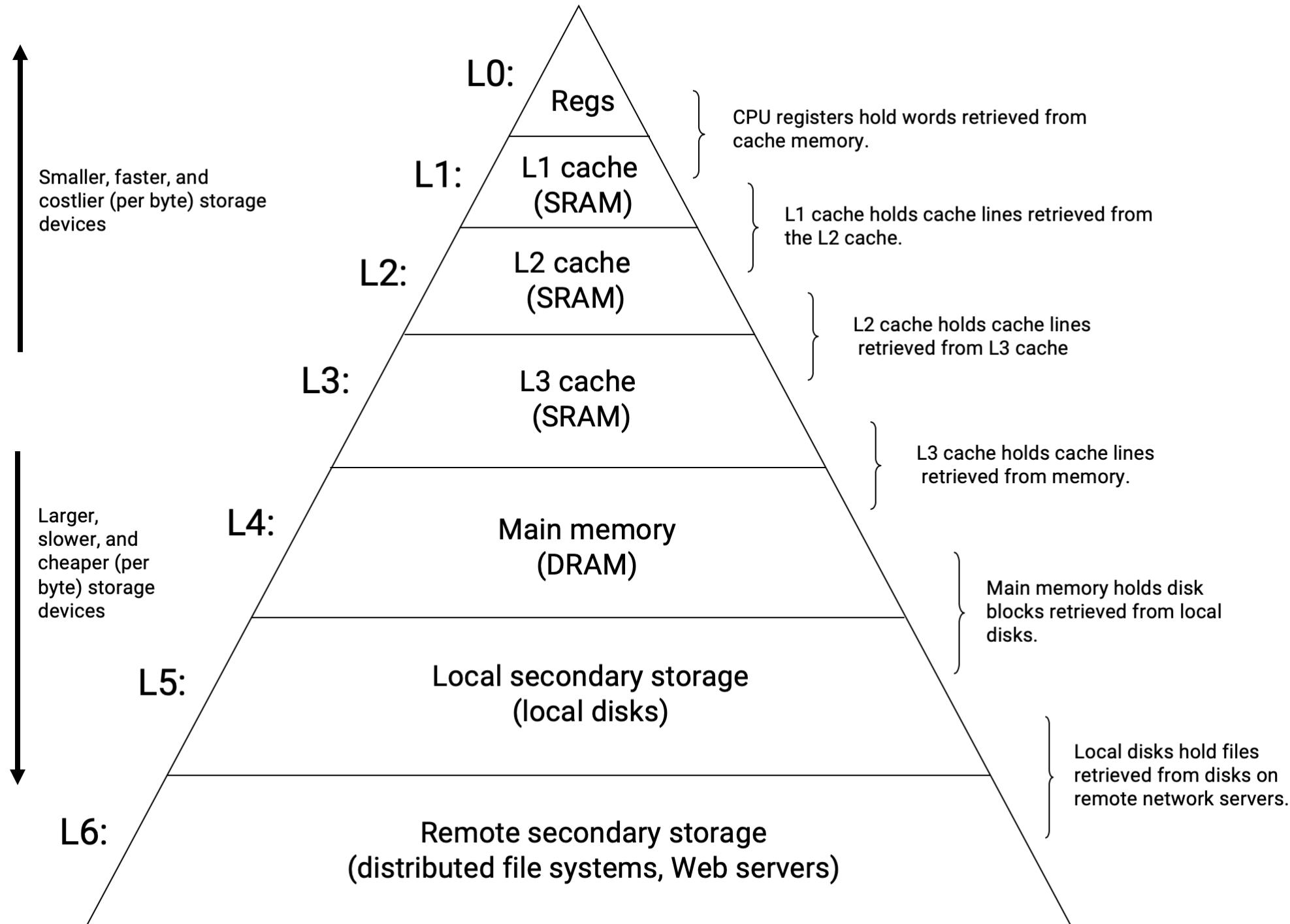
## Motherboard with 2 CPU sockets



## Performance tip on multicore

- Memory is key to developing high-performance multicore applications
- Memory traffic and time to access memory are often more important than flops
- Memory is hierarchical and complex





<b>Memory</b>	<b>Size</b>	<b>Latency</b>	<b>Bandwidth</b>
L1 cache	32 KB	1 nanosecond	1 TB/second
L2 cache	256 KB	4 nanoseconds	1 TB/second Sometimes shared by two cores
L3 cache	8 MB or more	10x slower than L2	>400 GB/second
MCDRAM		2x slower than L3	400 GB/second
Main memory on DDR DIMMs	4 GB-1 TB	Similar to MCDRAM	100 GB/second
Main memory on Intel Omni-Path Fabric	Limited only by cost	Depends on distance	Depends on distance and hardware
I/O devices on memory bus	6 TB	100x-1000x slower than memory	25 GB/second
I/O devices on PCIe bus	Limited only by cost	From less than milliseconds to minutes	GB-TB/hour Depends on distance and hardware