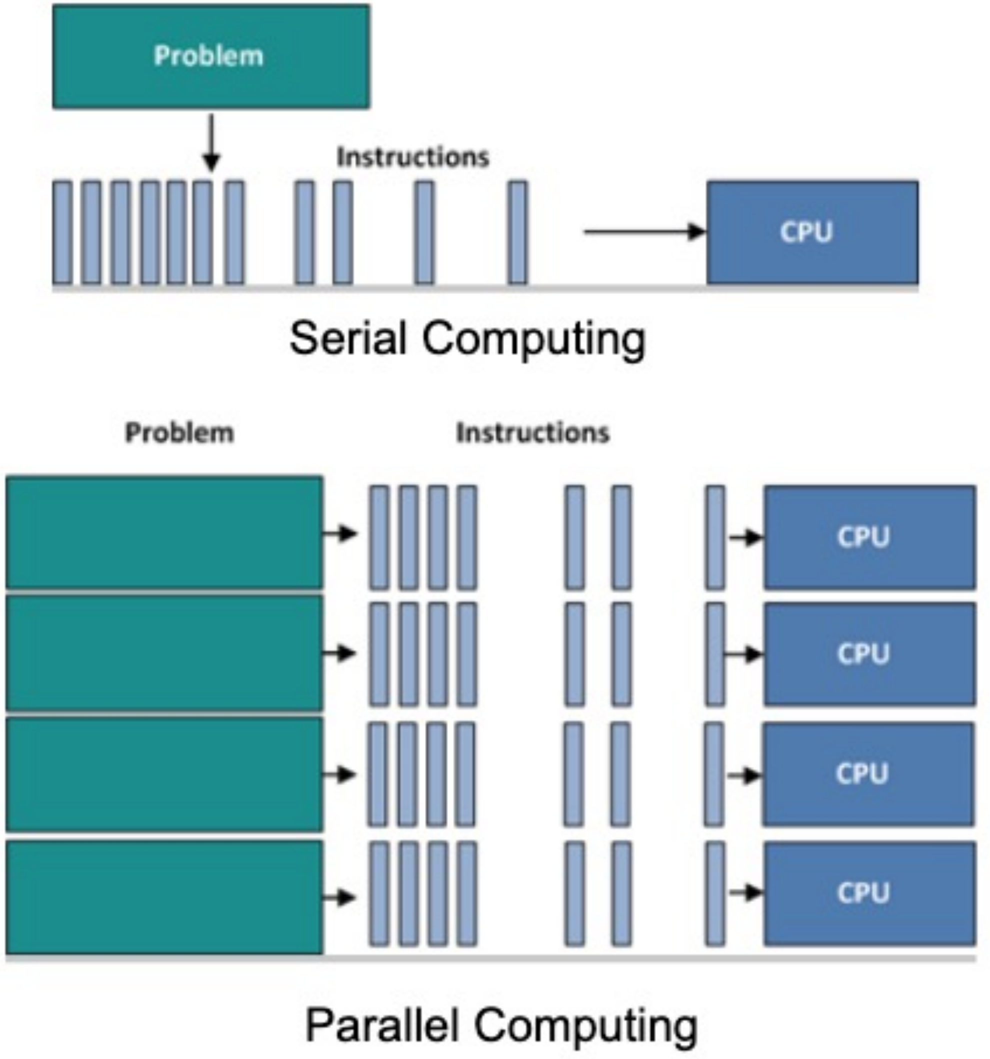
High Performance Computing (HPC)

What is HPC?

Aggregation of computing power for tasks too large for desktop PCs

At heart of HPC is task parallelisation

CPU speed is same as desktops but there are far more of them!



HPC architecture

Shared Memory Architecture

- All cores connected to single memory
- Process has direct access to all memory
- Most common HPC architecture 20 years ago

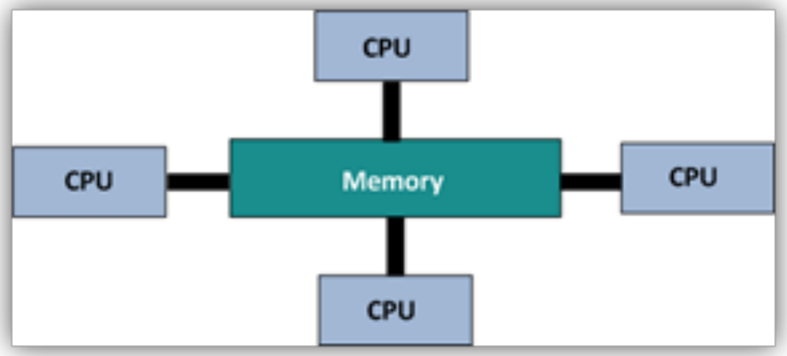
Distributed Memory Architecture

- Clusters of processors connected via interconnect
- Parallelisation achieved via interconnect
- Each node has OS
- No direct access to memory of other processes

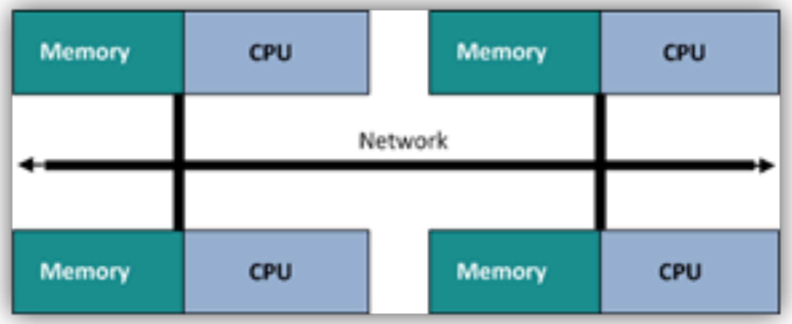
Hybrid Architecture

- Each node has multiple cores
- Each node has shared memory system
- Parallelisation achieved:
  - Within node (i.e. multi-core) – HPC
  - Between nodes (using interconnect) – super-computing

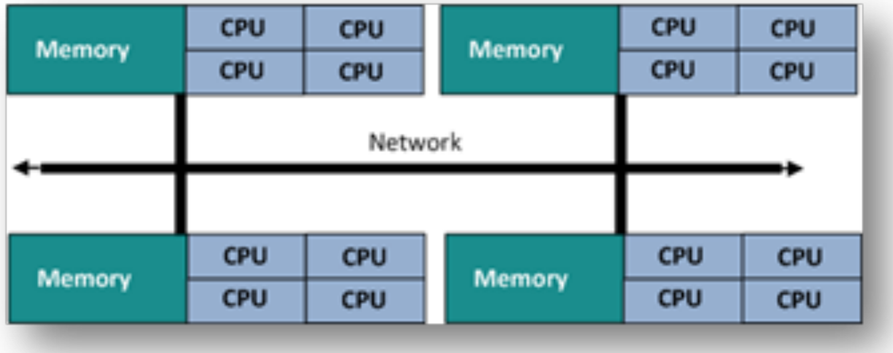
Shared Memory



Distributed Memory



Hybrid

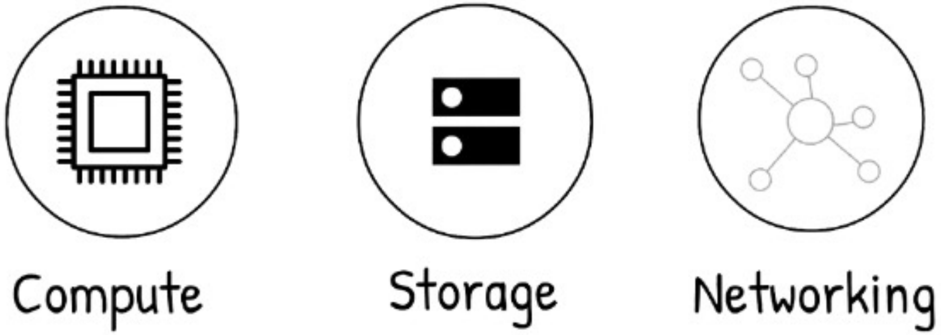


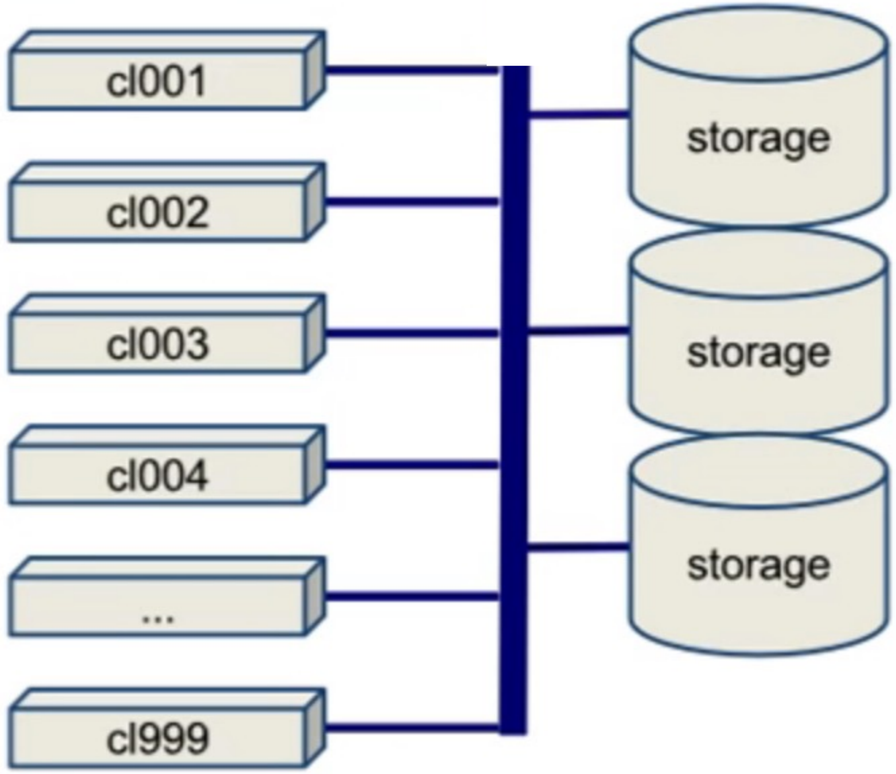
HPC architecture

Each node has RAM, OS, CPU

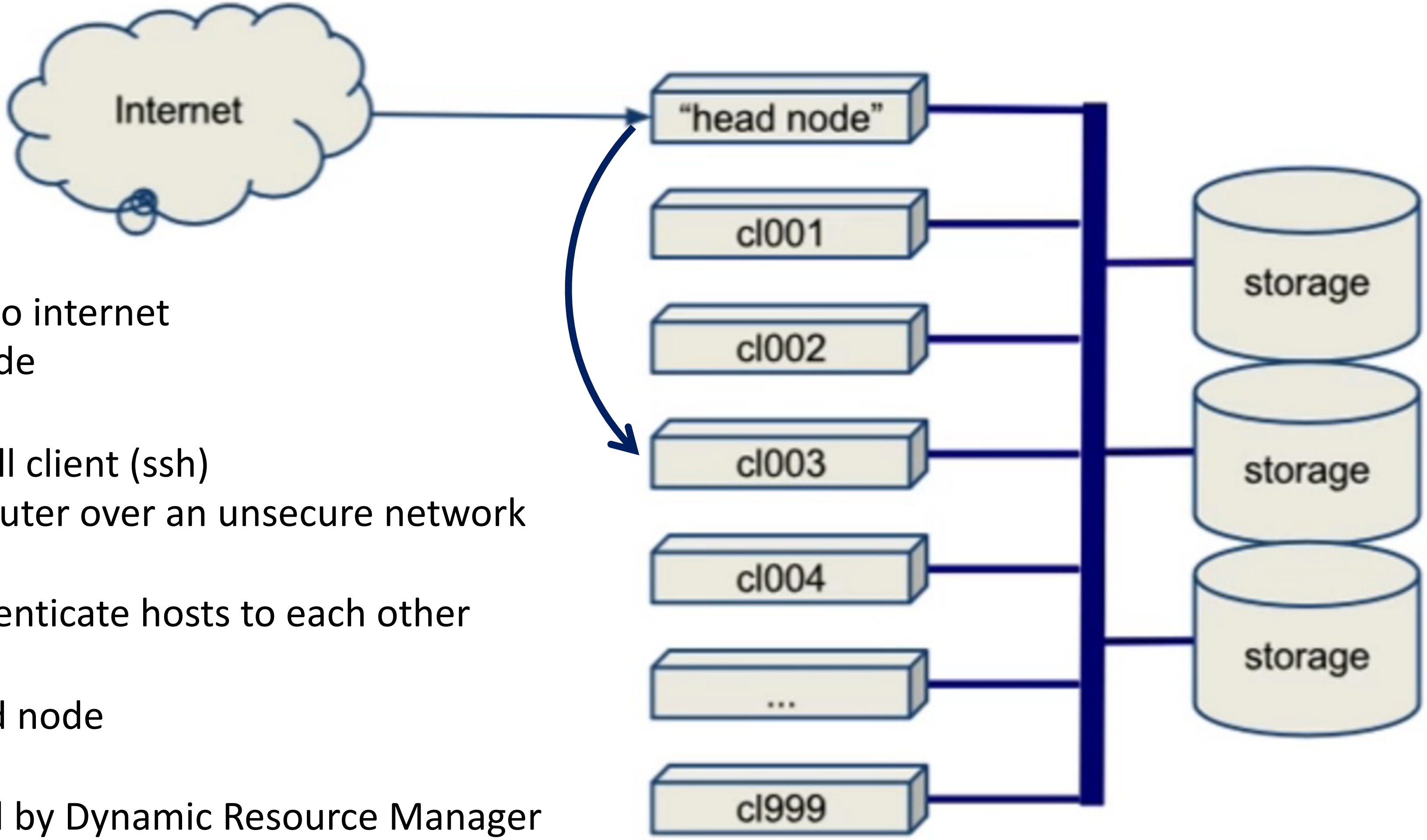
Files are stored on separate storage array
  - all nodes have access
  - doesn’t matter which node does calcs

Nodes and storage connected via network
  - ethernet or InfiniBand





Submitting Jobs



Most nodes are not connected to internet

- users log in to head/login node

Users access HPC via secure shell client (ssh)

- Secure way to access a computer over an unsecure network (i.e. internet)
- Uses public key pairs to authenticate hosts to each other

Calculations are not ran on head node

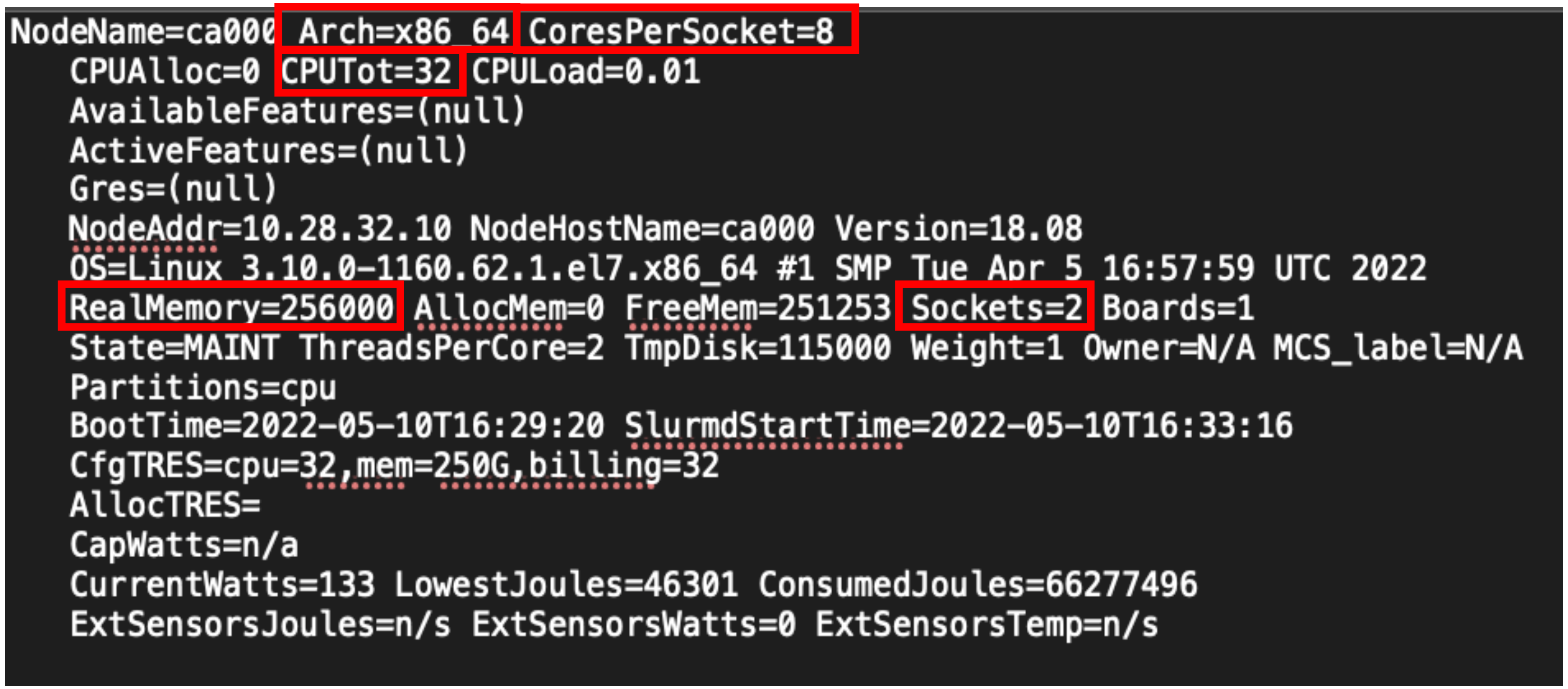
- many users logged in
- job submissions orchestrated by Dynamic Resource Manager (scheduler)
    - Grid Engine, SLURM

CAMP

Resources restricted to one node

190 CPU nodes – each with 32 virtual cores and 250GB RAM
4 high RAM CPU nodes – 96 virtual cores and 1500GB RAM
40 GPU nodes – each with 80 virtual cores and 750GB RAM

10 petabytes of data storage – 1PB = 1024TB



```
NodeName=ca000 Arch=x86_64 CoresPerSocket=8
   CPUAlloc=0 CPUTot=32 CPULoad=0.01
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=(null)
   NodeAddr=10.28.32.10 NodeHostName=ca000 Version=18.08
   OS=Linux 3.10.0-1160.62.1.el7.x86_64 #1 SMP Tue Apr 5 16:57:59 UTC 2022
   RealMemory=256000 AllocMem=0 FreeMem=251253 Sockets=2 Boards=1
   State=MAINT ThreadsPerCore=2 TmpDisk=115000 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=cpu
   BootTime=2022-05-10T16:29:20 SlurmdStartTime=2022-05-10T16:33:16
   CfgTRES=cpu=32,mem=250G,billing=32
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=133 LowestJoules=46301 ConsumedJoules=66277496
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

slurm

Open source job scheduler

- Allocates users resources (nodes) for jobs
- Framework for starting, executing and monitoring jobs
- Manages limited resouces (queue of pending jobs)

Slurmctld

- Centralised slurm manager which monitors resources and work

Slurmd

- Each compute node has slurm daemon
- Waits for work -> executes work -> returns status -> waits for work

sched/builtin

- Jobs submitted strictly in priority order

Sched/backfill

- Lower priority jobs submitted if does not delay start time of higher priority jobs
- Maximum efficiency

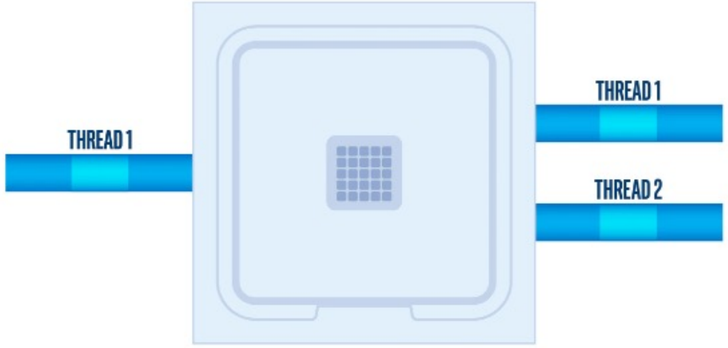
Intel Xeon

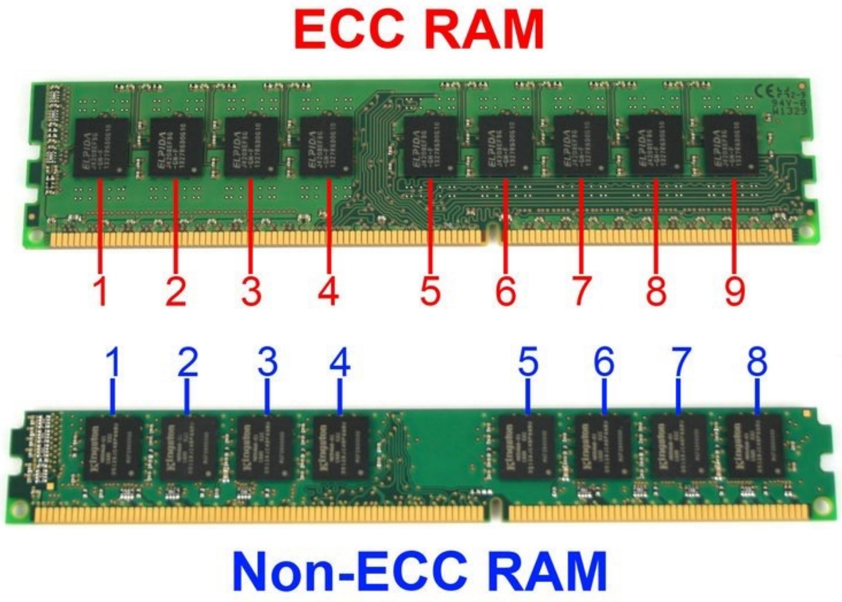
- Designed for servers

- Up to 28 cores
  - Typically slower clock speed than intel core processors – they generate more heat

- Hyperthreading capability
  - Enables two threads to be run by each core
  - OS recognises 2 logical cores per physical core
  - Scheduling technique to eliminate time CPU is idle

- Stability and lower energy usage is prioritised for servers
  - ECC (error correcting code) memory
    - Find and correct bit flips
    - Most common is SECDED Hamming code
      - Can CORRECT a single bit flip or DETECT two bit errors

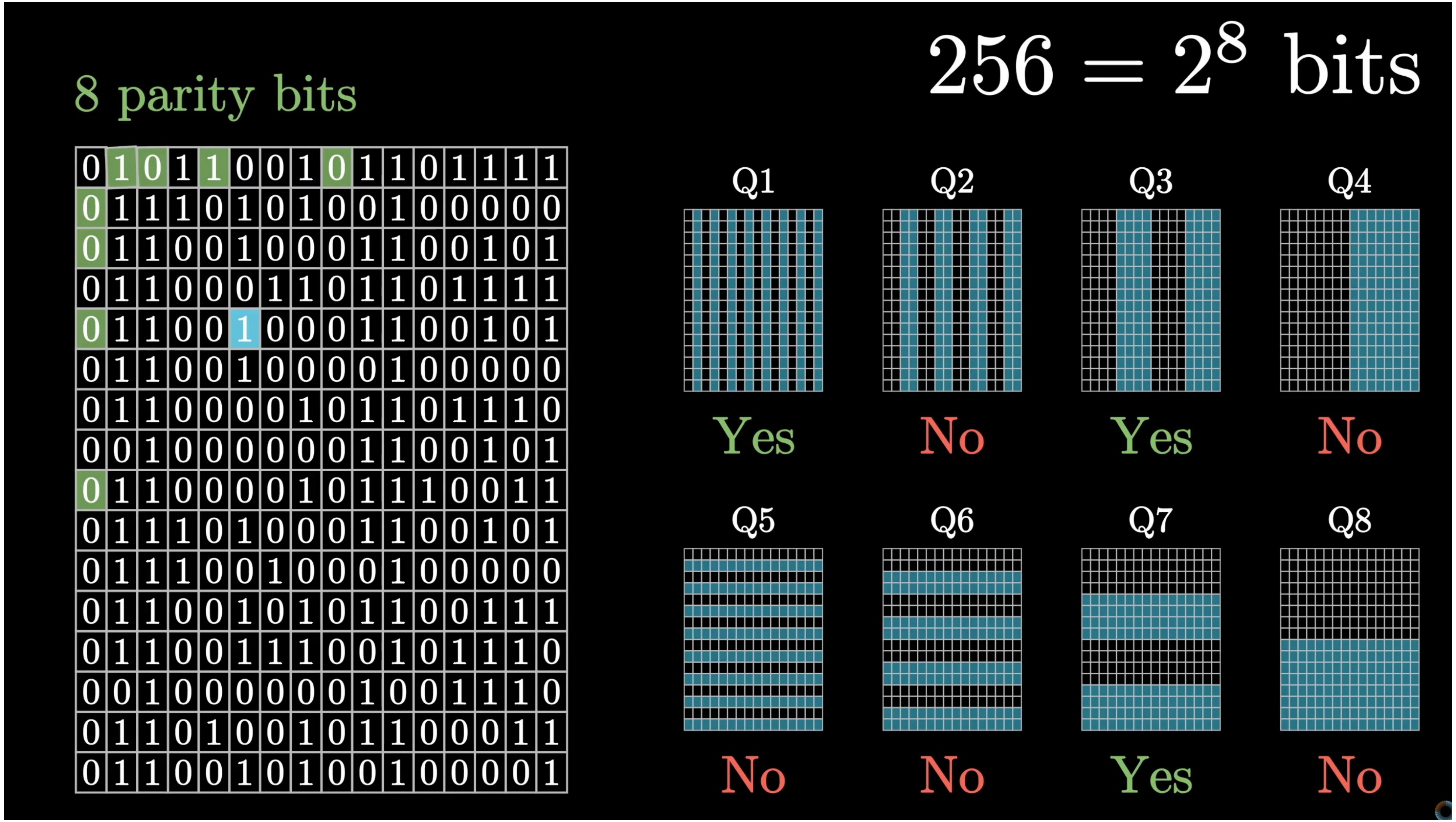




Hamming code

Part of memory are allocated as parity bits

- 16 bit message = 11 data bits (68.75%)
- 256 bit message = 247 data bits (96.48%)



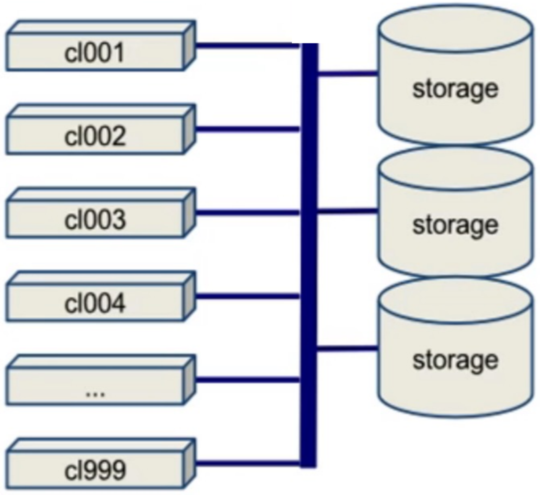
$256 = 2^8$ bits

8 parity bits

0	1	0	1	1	0	0	1	0	1	1	0	1	1	1	1
0	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0	1	1	0	1	1	1	1
0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	1
0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1
0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	1
0	1	1	1	0	1	0	0	0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	1
0	1	1	0	0	1	1	1	0	0	1	0	1	1	1	0
0	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0
0	1	1	0	1	0	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	0	1	0	0	1	0	0	0	0	1

https://harryli0088.github.io/hamming-code/

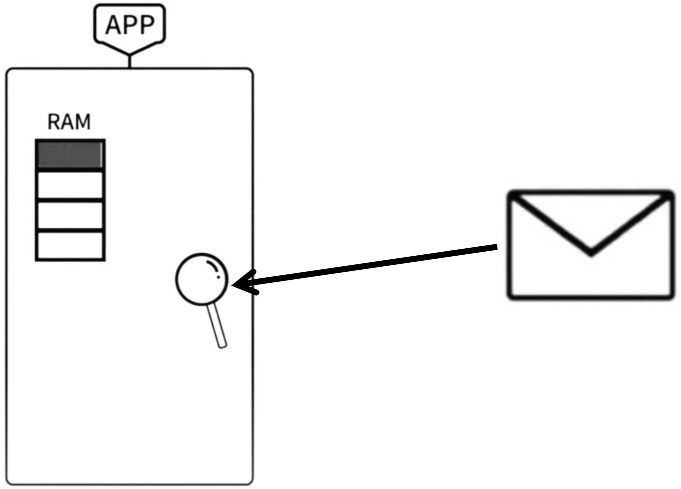
Server communication



CPU time can be lost whilst transporting or waiting for data

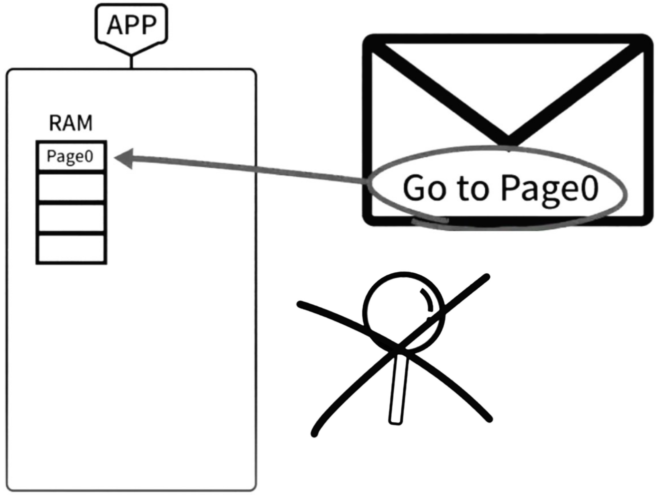
Two-sided communication

- Receiver has to accept data
- Receiver places data into memory
- Standard ethernet



RDMA (Remote Direct Memory Access)

- One sided data communication
- Little overhead on the CPU
- Sender includes destination memory address
- Hardware on receiver side places data directly into memory



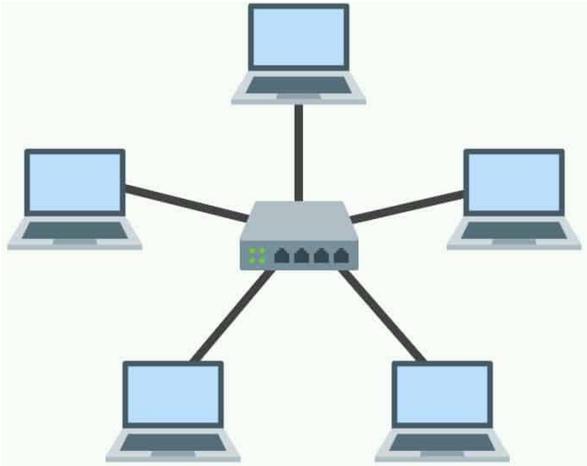
- InfiniBand has built in RDMA capabilities

Network topology

Network topology affects:

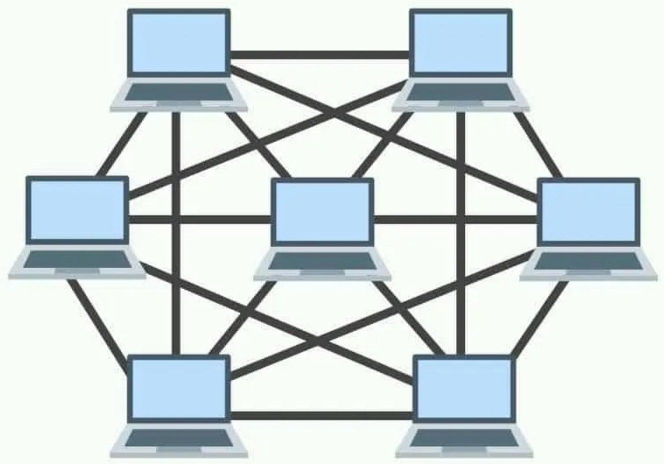
- Bandwidth
- Latency
- Scalability
- Fault tolerance

Star



- Easy to set up
- Not scalable
- Single point of failure

Mesh



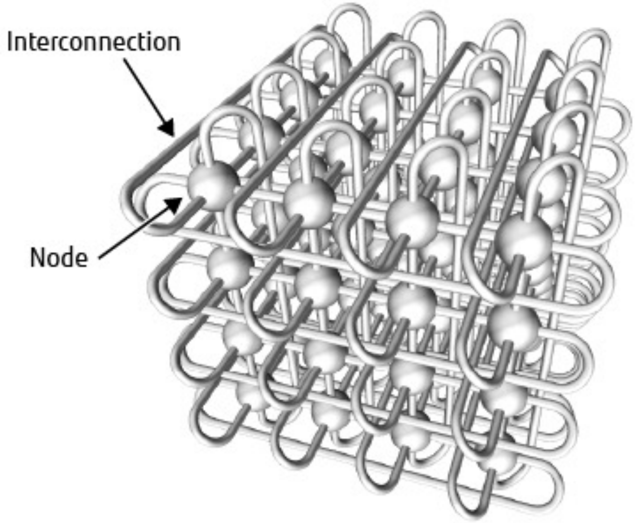
- Difficult to set up
- Extremely scalable
- Fault tollerant

Network topology

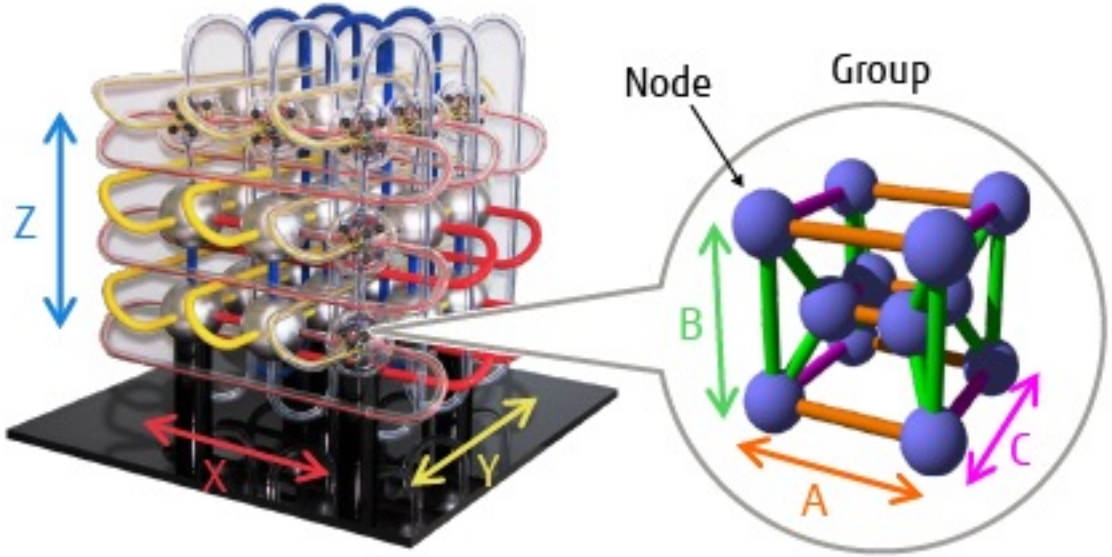
Supercomputers utilise inter-node communication

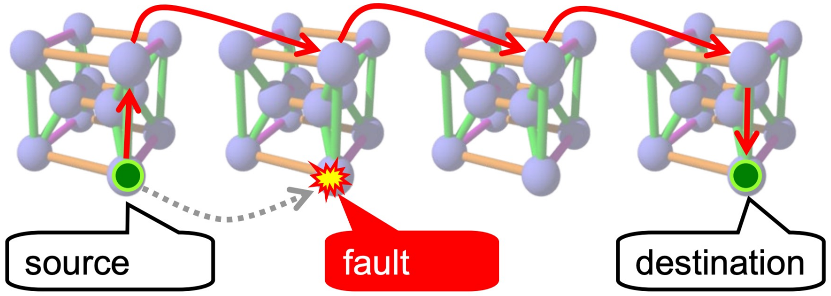
Topology is critical

3D Torus topology

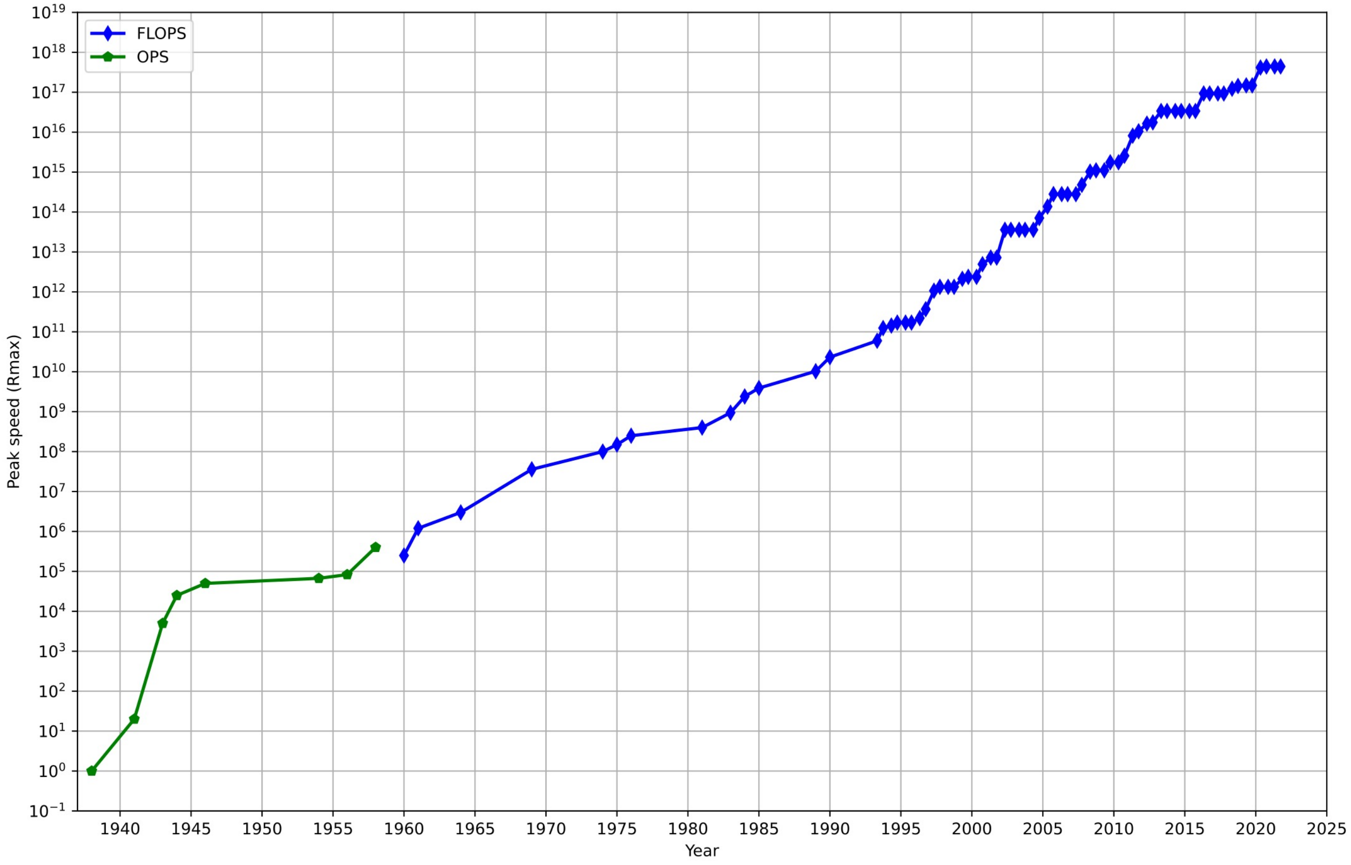


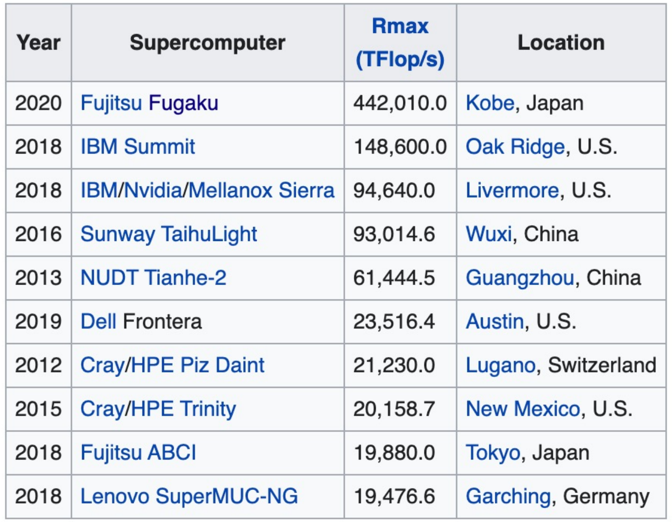
6D Torus topology



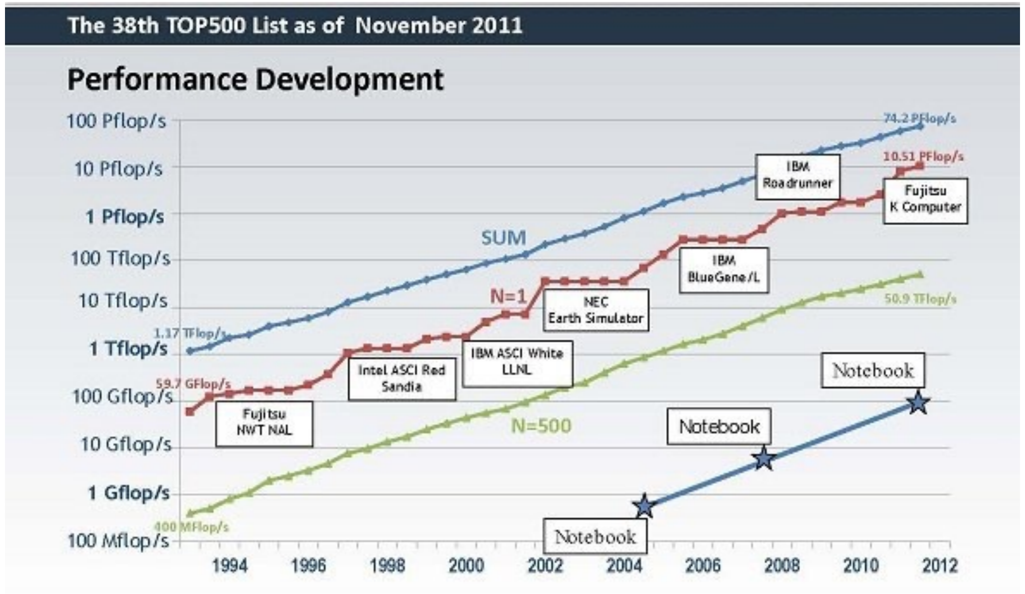


Supercomputers





Year	Supercomputer	Rmax (TFlop/s)	Location
2020	Fujitsu Fugaku	442,010.0	Kobe, Japan
2018	IBM Summit	148,600.0	Oak Ridge, U.S.
2018	IBM/Nvidia/Mellanox Sierra	94,640.0	Livermore, U.S.
2016	Sunway TaihuLight	93,014.6	Wuxi, China
2013	NUDT Tianhe-2	61,444.5	Guangzhou, China
2019	Dell Frontera	23,516.4	Austin, U.S.
2012	Cray/HPE Piz Daint	21,230.0	Lugano, Switzerland
2015	Cray/HPE Trinity	20,158.7	New Mexico, U.S.
2018	Fujitsu ABCI	19,880.0	Tokyo, Japan
2018	Lenovo SuperMUC-NG	19,476.6	Garching, Germany



- Exoscale speeds have been achieved
- China and US have 2/3 of global supercomputers