HPC User Training: Lecture 01: High Performance Computing Overview

Mary Thomas

San Diego Supercomputer Center (SDSC)
Univeristy of California San Diego (UCSD)

Posted: 01/22/21
Updated: 01/22/21

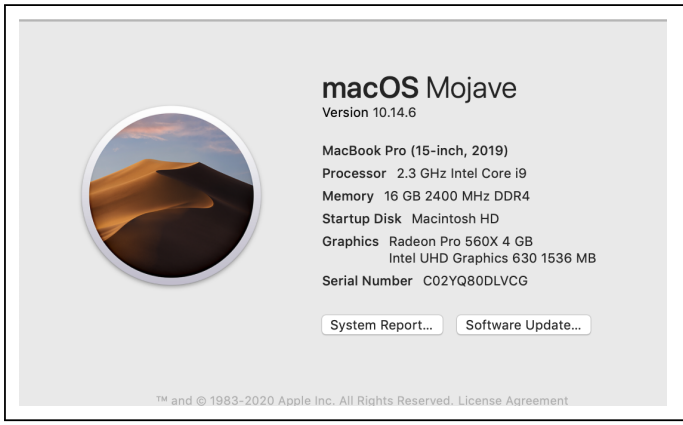
Table of Contents

Motivation for HPC

Motivation for HPC

My serial code is working, why should I parallelize it?

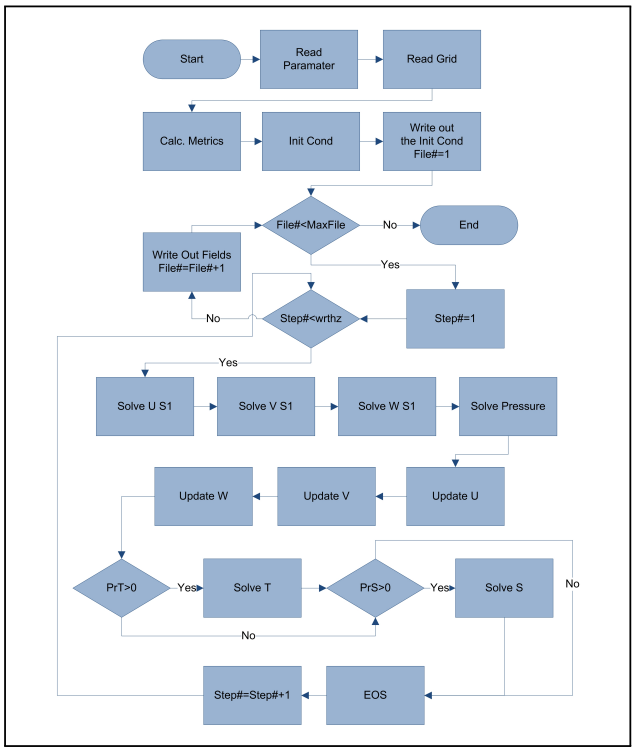
- Compute bound
  - Large number of iterations
  - Long time to converge
- Data bound
  - Memory footprint is larger than RAM (common for science apps)
  - Output may take FOREVER to save/dump/transfer
- contention for resources/other processes
  - What other processes are running on the machine?
  - What other data is running on the network?
  - Timeout problems?



Note: Intel Core i7 has 4 physical cores, I9 has 8.

General Curvilinear Environmental Model

- Simulate processes along the coast
- Serial version, Simple Seamount, appx 8k lines
  - 1 km /cell, 100x50x50 (small job)
  - large memory: 100 arrays $\tilde{O}(10^7)$
  - 200k iters $\simeq$ 6 hours simulation time
  - Twall $\simeq$ 70 hours $\simeq$ $O(10^{14})$ ops
- Typical simulations need to run for days/weeks/months/years
  - Jaguar (ORNL), $2x10^5$ cores ($\simeq$3GHz, 16GB), $1.75x10^{12}$ flops
  - Could run this job in $0.3x10^{-2}$ seconds (note: est. does not include time to fire up 200k cores)
  - Motivation for parallel computing!



Source: `http://www.xsede.org`

Porting Code to Parallel has Many Challenges

- I have 200k cores, now what do I do with them?
- How to distribute the data? Is it memory/IO bound)?
- How to distribute the computation? Is it compute bound?
- How do you synchronize the results?

What is Parallel Computing and Why do We Need it?

- Large problems:
  - Biz: BM has 433,000 employees; how do you update paycheck/empoyee data?
  - Science: Hurricane Predictions, GoM is appx 800x800 km = 6.4x10^5 points/array. Typically have 10^2 arrays. Forecast models use multiple models, multiple clusters, take 12 hours (wall clock) to forecast out 3-4 days on parallel computers (probably 500 nodes). See `http://www.srh.noaa.gov/ssd/nwpmodel/html/nhcmodel.htm`, f
- Large data: These models generate massive data sets (Tera-to-peta-bytes per simulation).
  - My coastal ocean model generates Vel(U,V,W), Pressure,Temp,Salinity files once each iteration at 2.5MB/file. 6 hour test run is 2x10^5 iterations = 3x10^6 MBytes (but we don't have room so we only store 100 slices).
  - Where do you store the data?
  - How do you move it? Sneaker Net?
- Fast networks are required
- Power consumption is big issue

Common HPC Computing Resources

- Compute Hardware: CPU, GPU, FPGA, Quantum
- Compute Systems: iPhone, Laptops; HPC systems & Clusters; Clouds;
  - iPhone, Laptops (MBP has 8 cores)
  - HPC Systems: 1000's to millions of cores
  - Cloud computing (Azure, Amazon, Google)
- Archival & Storage:
  - high speed parallel servers with large storage (Lustre)
  - Large, distributed (Open Storage Network, https://www.openstoragenetwork.org/)
  - Cloud storage (Amazon EC2, S3 Bucket)
- Networks: 400 GB/s now available
  - Internet2 (https://internet2.edu/)
  - CENIC- California Research and Education Network (CalREN, https://cenic.org/)
  - Pacific Research Platform
- Visualization
- Cyberinfrastructure

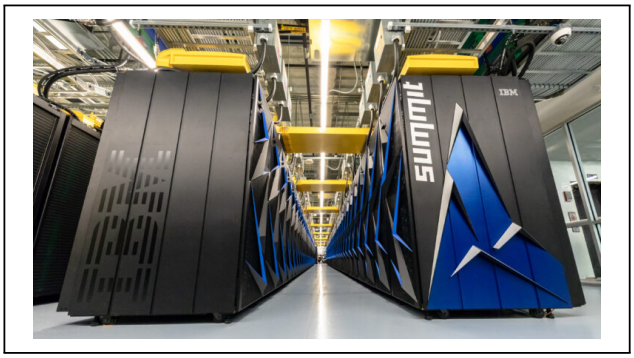
HPC Units of Measure

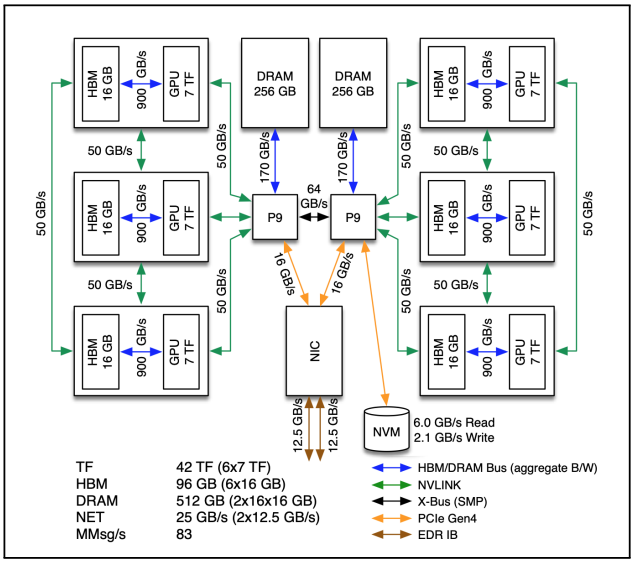
What is the scale of modern parallel resources & computations

Unit	Definition
Flops	floating point operations
Flop/s	floating point operations per second

Prefix	#Flops/sec 2	Bytes	10^n	2^n
Mega	Mflop/s	Mbyte	10^6	$2^{20} = 1{,}048{,}576$
Giga	Gflop/s	Gbyte	10^9	$2^{30} = 1{,}073{,}741{,}824$
Tera	Tflop/s	Tbyte	10^{12}	$2^{40} = 10{,}995{,}211{,}627{,}776$
Peta	Pflop/s	Pbyte	10^{15}	$2^{50} = 1{,}125{,}899{,}906{,}842{,}624$
Exa	Eflop/s	Ebyte	10^{18}	$2^{60} = 1.15292152^{18}$
Zetta	EZlop/s	Ebyte	10^{21}	$2^{70} = 1.1805916^{21}$
Yotta	Yflop/s	Ebyte	10^{24}	$2^{80} = 1.2089258^{24}$

Source: https://en.wikipedia.org/wiki/FLOPS

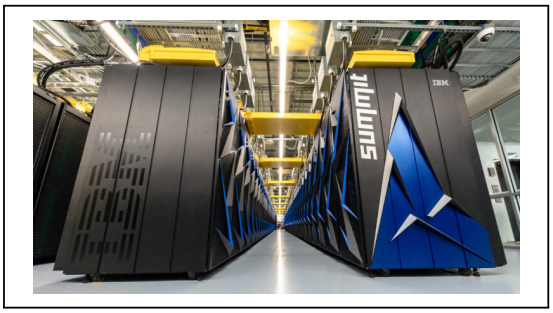


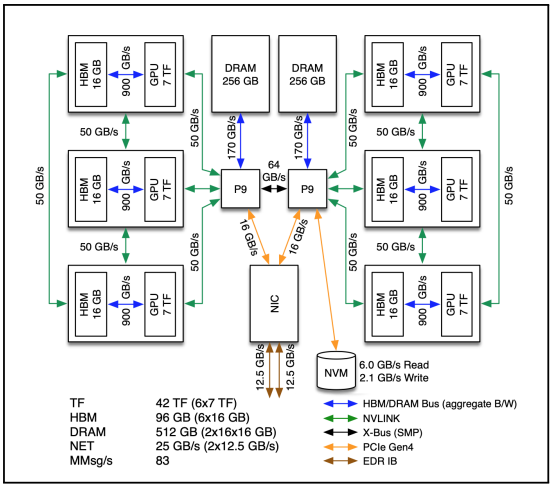


Source: `http://www.xsede.org`

Oak Ridge National Lab: Summit

- Architecture 9,216 POWER9 22-core CPUs
- 27,648 NVIDIA Tesla V100 GPUs[1]
- Power 13 MW[2]
- Operating system Red Hat Enterprise Linux (RHEL)[3][4]
- Storage 250 PB
- Speed 200 petaFLOPS (peak)
- Purpose Scientific research
- Web site www.olcf.ornl.gov/olcf-resources/compute-systems/summit/



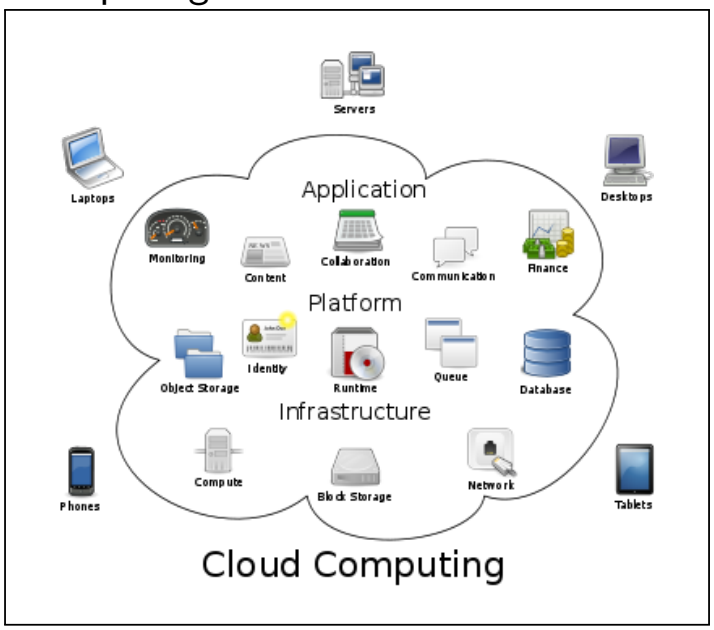


Source: `http://en.wikipedia.org/wiki/Cloud_computing`

Cloud Computing

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet).
There are many types of public cloud computing:

- Infrastructure as a service (IaaS),
- Platform as a service (PaaS),
- Software as a service (SaaS)
- Storage as a service (STaaS)
- Security as a service (SECaaS)
- Data as a service (DaaS)
- Business process as a service (BPaaS)
- Test environment as a service (TEaaS)
- Desktop as a service (DaaS)
- API as a service (APIaaS)
- Cyberinfrastructure



Source: `http://en.wikipedia.org/wiki/Cloud_computing`

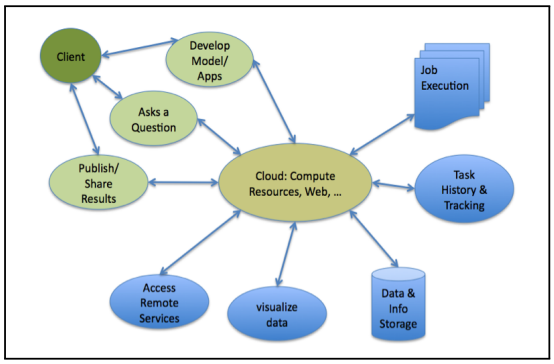
Cyberinfrastructure (CI)

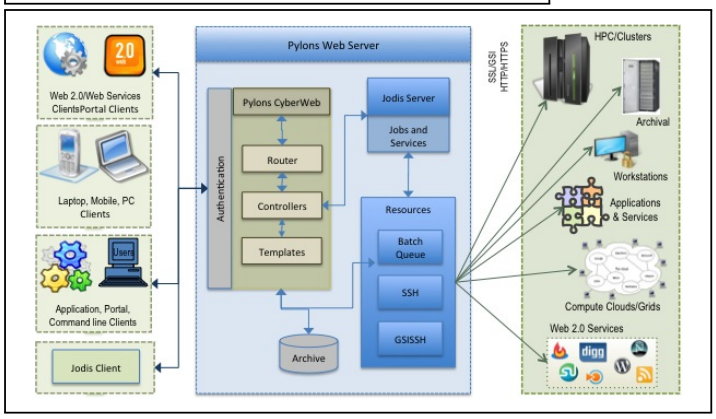
- Cyberinfrastructure - Research environment or infrastructure based upon distributed computer, information and communication technology. (From: Atkins PITCAC Report, 2003)
- Resources: data acquisition, data storage, data management, data integration, data mining, data visualization, computing and information processing services distributed over the Internet beyond the scope of a single institution.*
- Scientific Computing: a technological and sociological solution to the problem of efficiently connecting laboratories, data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge.*
- Cyberinfrastructure = HPC + Cloud/Grid + Networks

* Source: `http://en.wikipedia.org/wiki/Cyberinfrastructure`

Example of CI Project: CyberWeb (gateway)

- Provide a bridge between users and high-end resources, emerging technologies and cyberinfrastructure.
- Simplify use by using common/familiar Web and emerging technologies
- Facilitate access to and utilization of Science Applications.
- Apps run in heterogeneous environment
- Plug-n-play mode

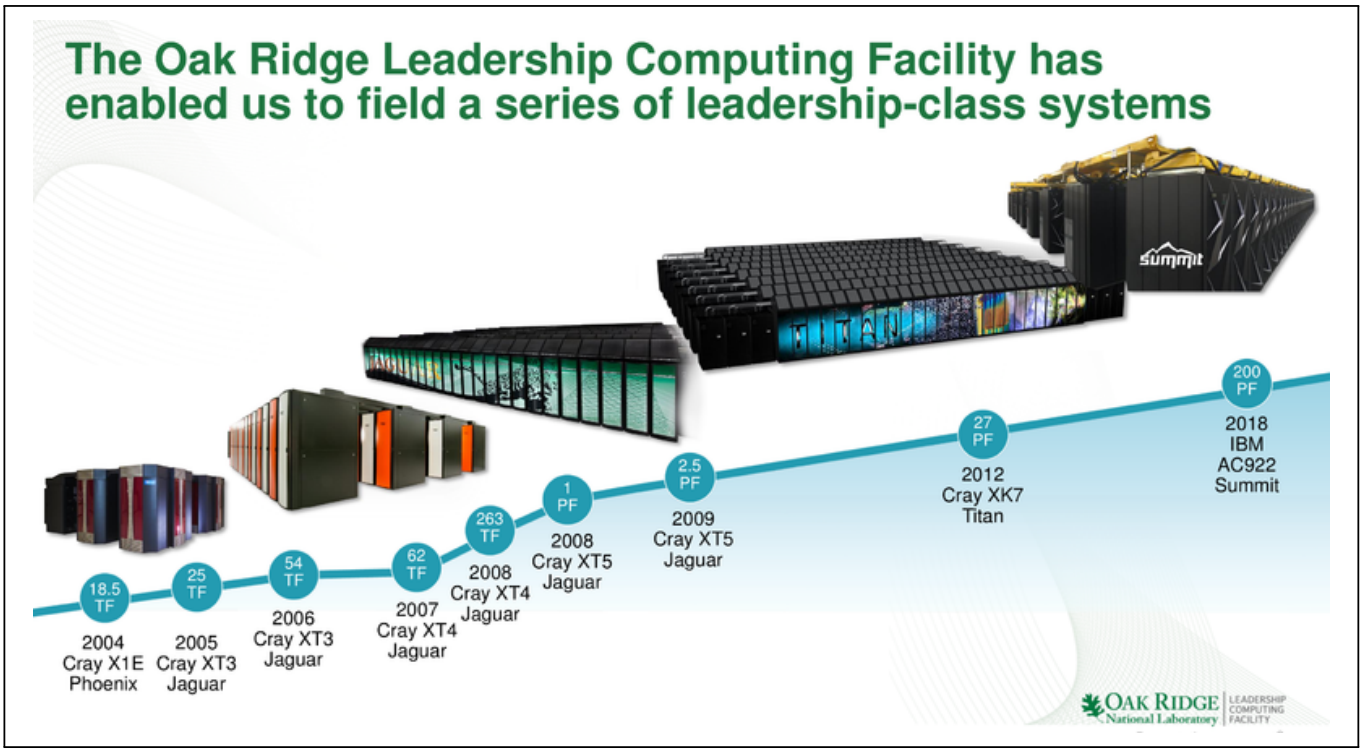




HPC Performance

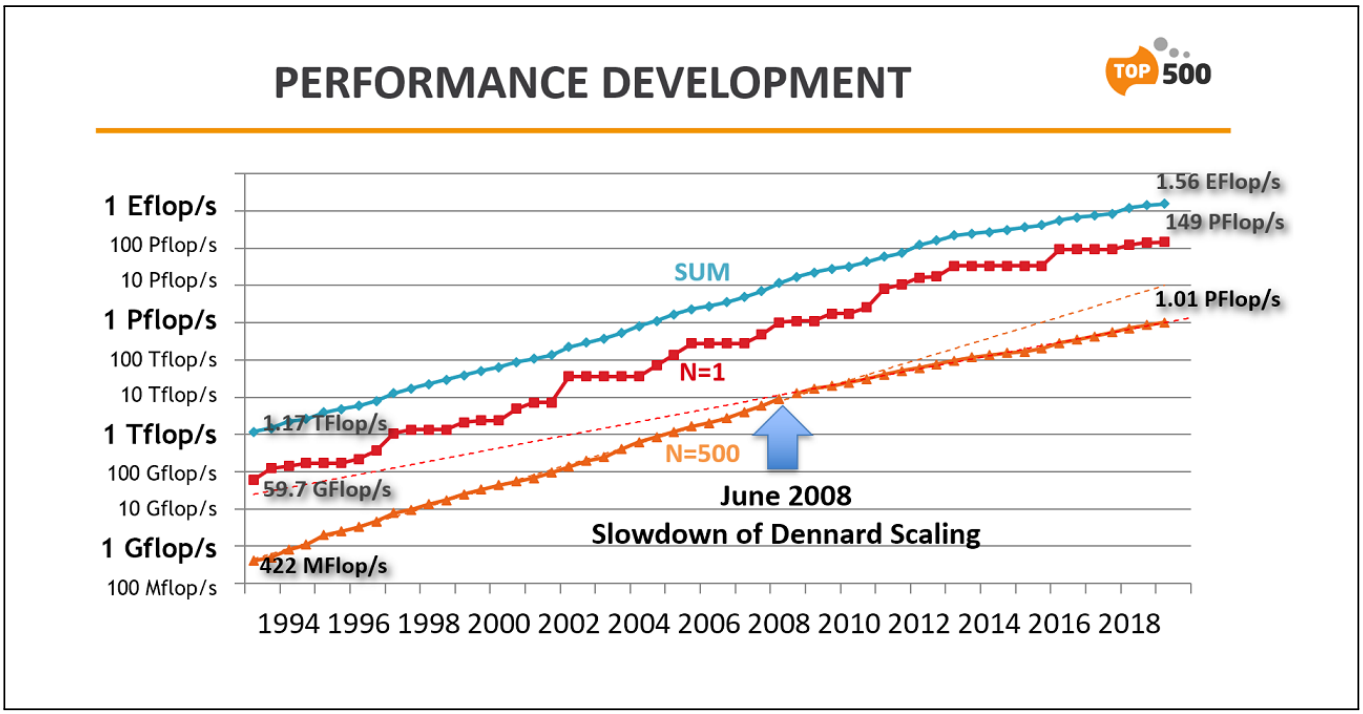
HPC Performance

Oak Ridge National Lab: HPC Systems Evolution



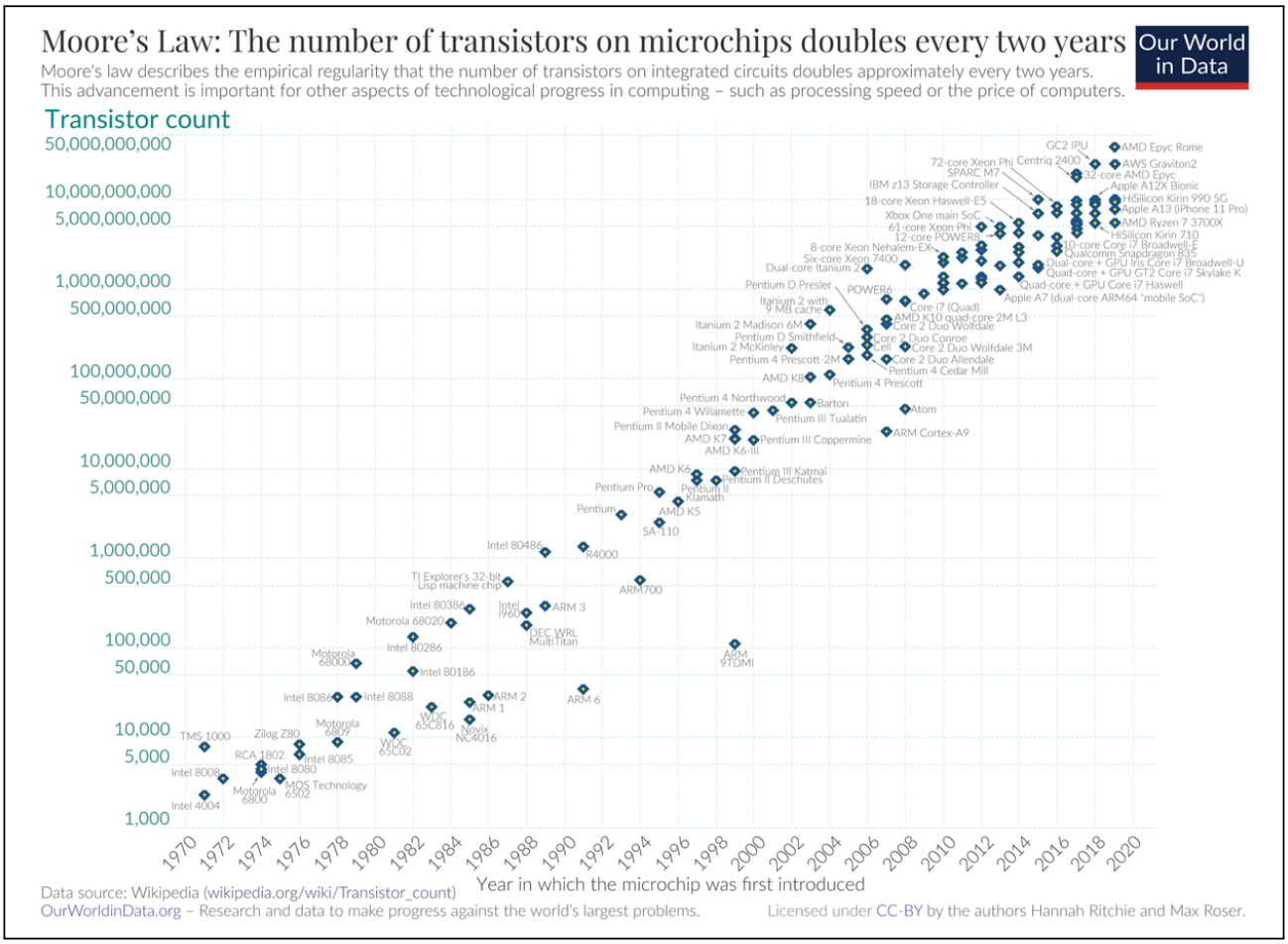
Source: `http://www.xsede.org`

Top 500 Dennard Scaling



Source: `http://www.xsede.org`

Moore's Law for Transistors



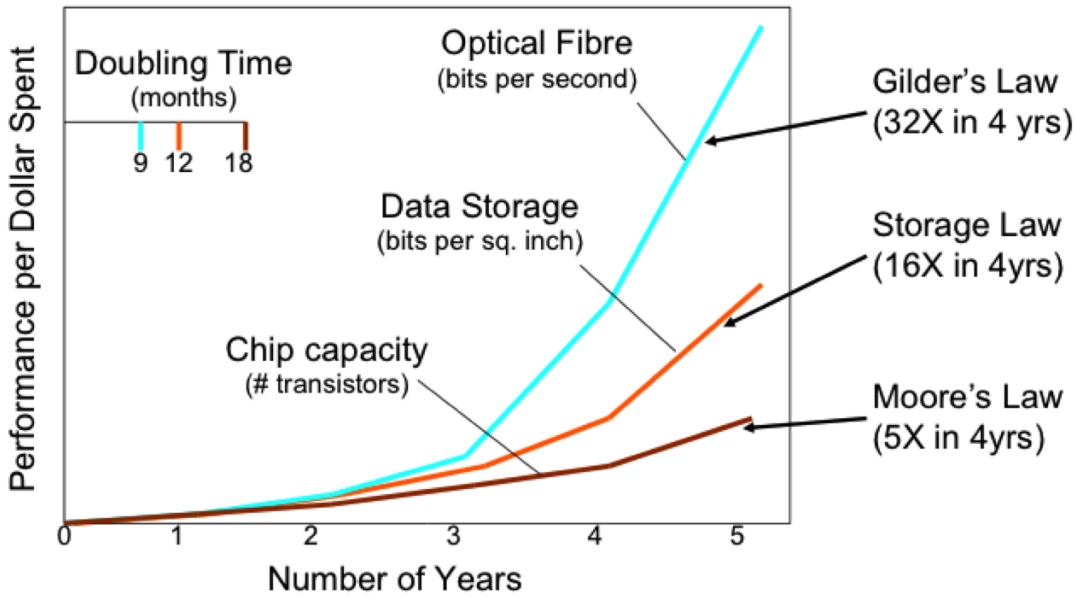
Source: `http://wikipedia.org/Transistor_count`

Technology Growth Laws

- **Moore's Law:** Processing power of a cpu/core doubles every 18 months; corollary, computers become faster and the price of a given level of computing power halves every 18 months (Gordon Moore,70's).
- **Gilder's Law:** Total bandwidth of communication systems triples every twelve months (George Gilder).
- **Metcalfe's Law:** The value of a network is proportional to the square of the number of nodes; so, as a network grows, the value of being connected to it grows exponentially, while the cost per user remains the same or even reduces (Robert Metcalfe, originator of Ethernet)
- **Kryder's Law:** The density of information on hard drives growing faster than Moore's law, and is doubling roughly every 13 months.

Source: `http://www.jimpinto.com/writings/techlaws.html`, and `http://www.scientificamerican.com/article/kryders-law/`

Technology Growth Laws



Source: G. Stix, Triumph of Light Scientific American. January 2001

Top500 Projected Performance

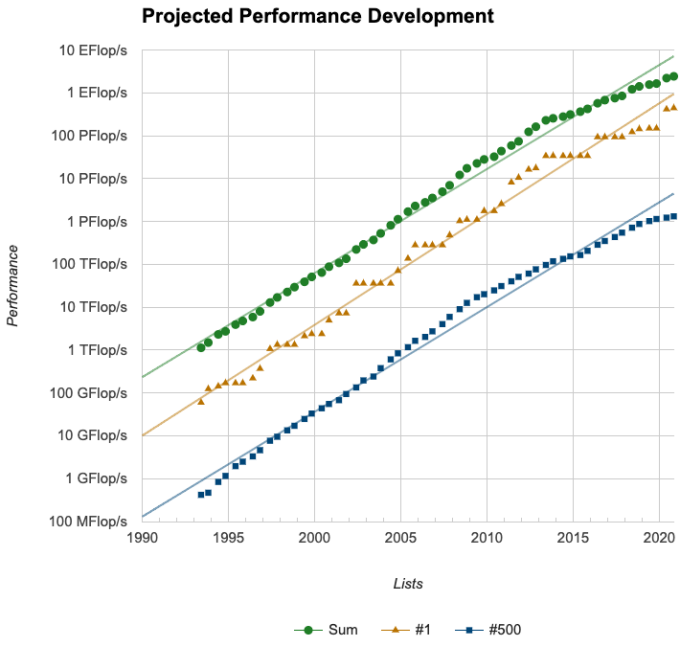


Figure: Source: `https://www.top500.org/statistics/perfdevel/`

Examples of Parallel Hardware and Architectures

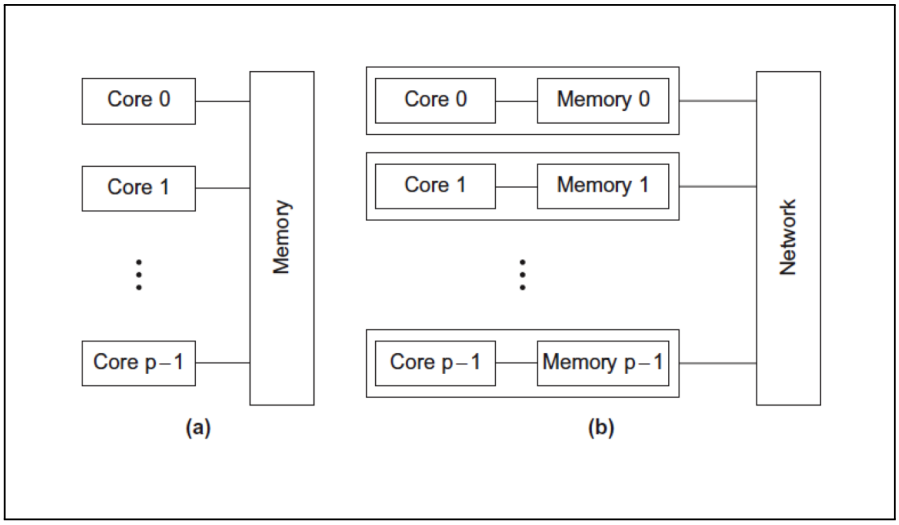
Examples of Parallel Hardware and Architectures

High Performance Computing - Key Components

- Parallel Systems (clustered, distributed)
- Large memory on device
- Fast Networks
- Large, fast data storage
- Parallel Software

Parallel Systems - Memory

- Shared-memory (a)
  - Cores share computer's memory
  - Cores access shared memory locations –¿ must synchronize.
- Distributed-memory (b)
  - Each core has its own, private memory.
  - Cores communicate explicitly by sending messages across a network.

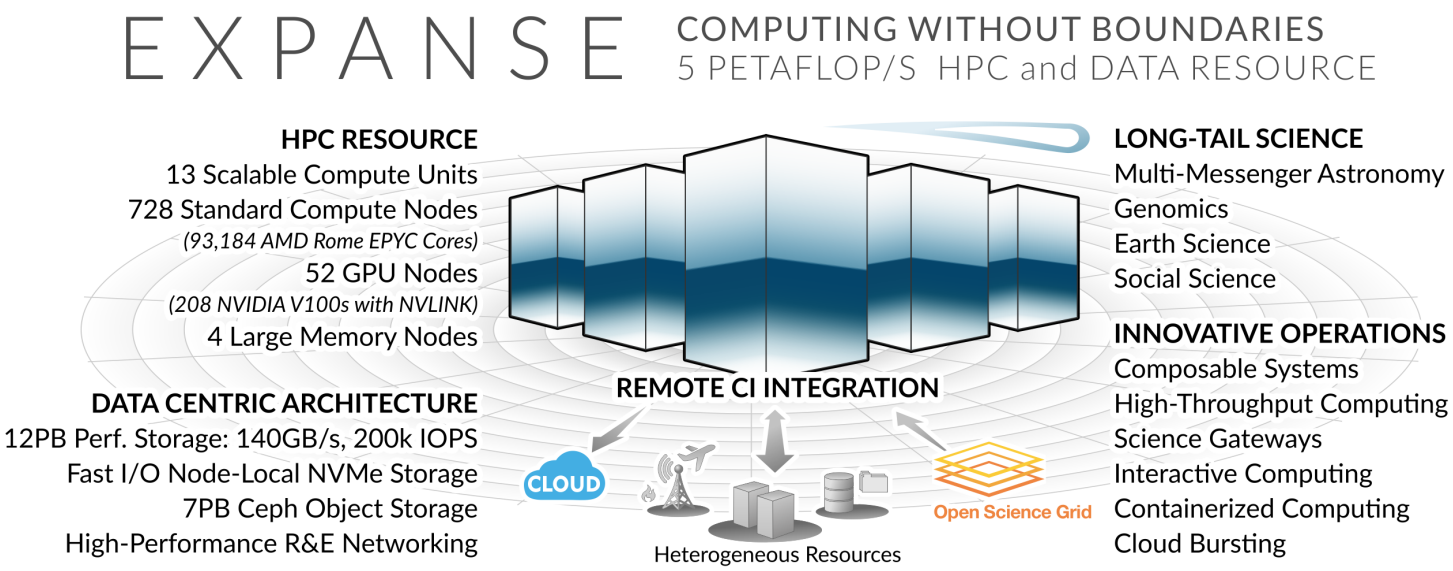


Source: Pacheco, 2011 Textbook

Parallel Computing Models

- **Concurrent computing:** a program is one in which multiple tasks can be in progress at any instant
  - Stock market transactions, embarrassingly parallel problems.
- **Parallel computing:** a program is one in which multiple tasks cooperate closely to solve a problem
  - CFD, chemical computations, tightly coupled
- **Distributed computing:** a program may need to cooperate with other programs to solve a problem
  - Seti@Home, the cloud, the grid, loosely coupled

HPC Hardware: SDSC E X P A N S E



Source: `http://expanse.sdsc.edu`

HPC Hardware: SDSC E X P A N S E - Specifications

Expanse System Specification

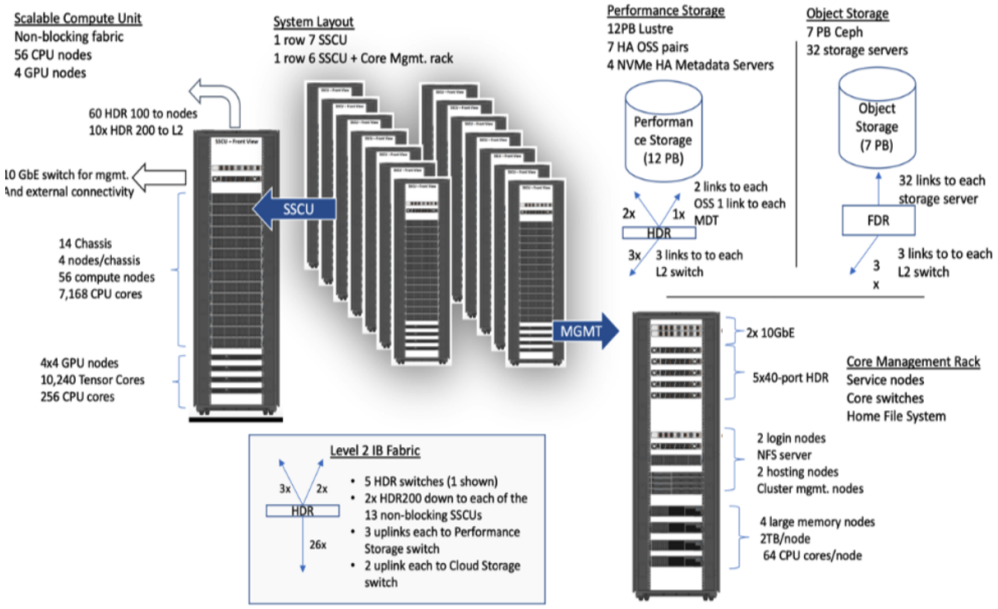
System Component	Configuration
AMD Rome Standard Compute Nodes	
Node count	728
Clock speed	2.25 GHz
Cores/node	128
DRAM/node	256 GB
NVMe/node	1 TB
NVIDIA V100 GPU Nodes	
Node count	52
CPU cores/node	40
CPU Type	6248 Xeon
CPU Clock speed	2.5 GHz
CPU DRAM/node	384 GB
GPUs/node	4
GPU Type	V100 SMX2
Memory/GPU	32 GB
Large-memory AMD Rome Nodes	
Node count	4
Clock speed	2.25 GHz
Cores/node	128
DRAM/node	2 TB
SSD memory/node	3.2 TB
Storage Systems	
File systems	Lustre, Ceph
Lustre Storage	12 PB
Ceph Storage	7 PB

Source: `http://expanse.sdsc.edu`

HPC Hardware: SDSC E X P A N S E - Architecture

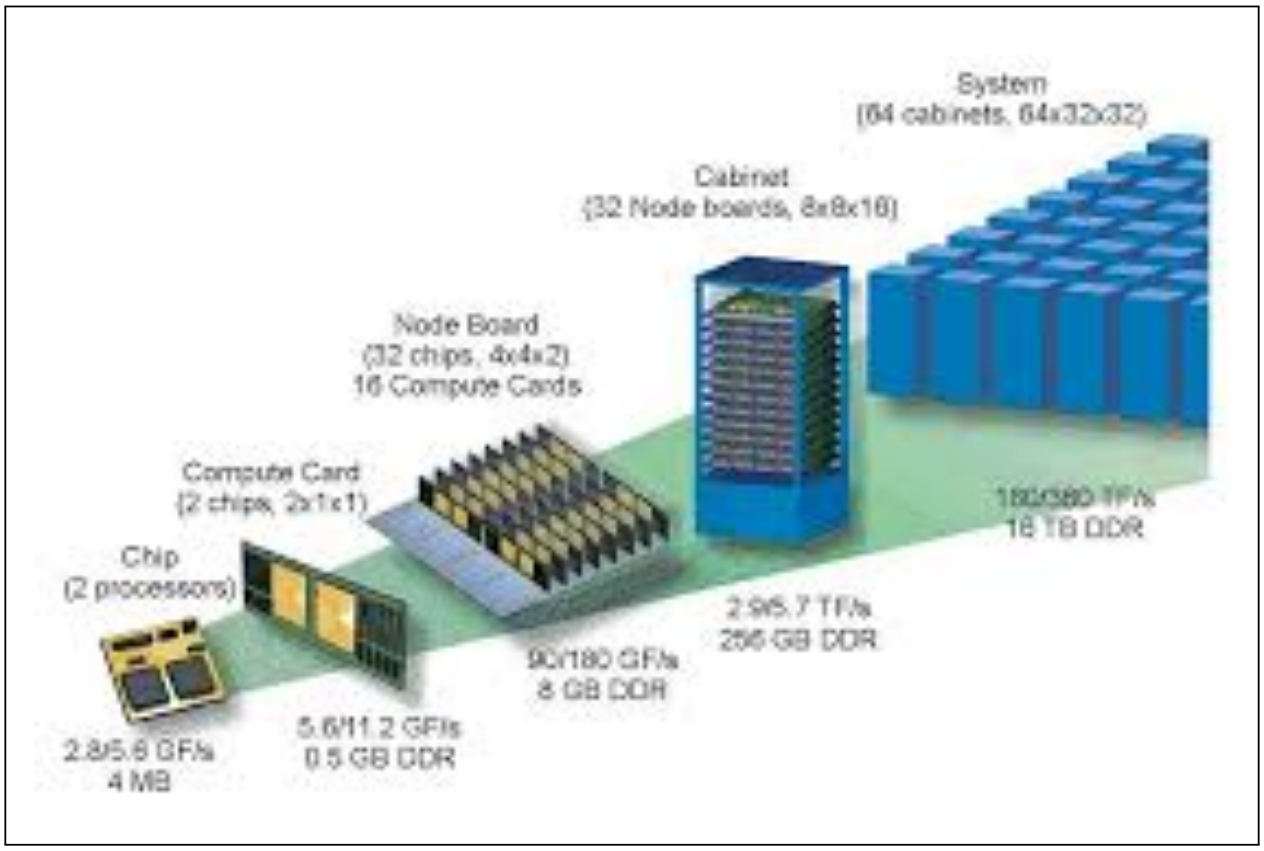
System Summary

- *13 SDSC Scalable Compute Units (SSCU)*
- *728 x 2s Standard Compute Nodes*
- *93,184 Compute Cores*
- *200 TB DDR4 Memory*
- *52x 4-way GPU Nodes w/NVLINK*
- *208 V100s*
- *4x 2TB Large Memory Nodes*
- *HDR 100 non-blocking Fabric*
- *12 PB Lustre High Performance*
- *Storage*
- *7 PB Ceph Object Storage*
- *1.2 PB on-node NVMe*
- *Dell EMC PowerEdge*
- *Direct Liquid Cooled*



Source: `https://hpc-training.sdsc.edu/expanse-101/`

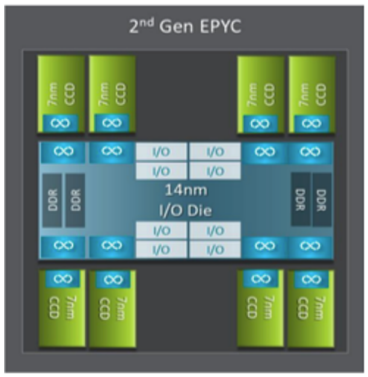
HPC Hardware: Blue Gene/L Hardware



HPC Hardware: AMD EPYC 7742 NUMA Architecture

AMD EPYC 7742 Processor Architecture

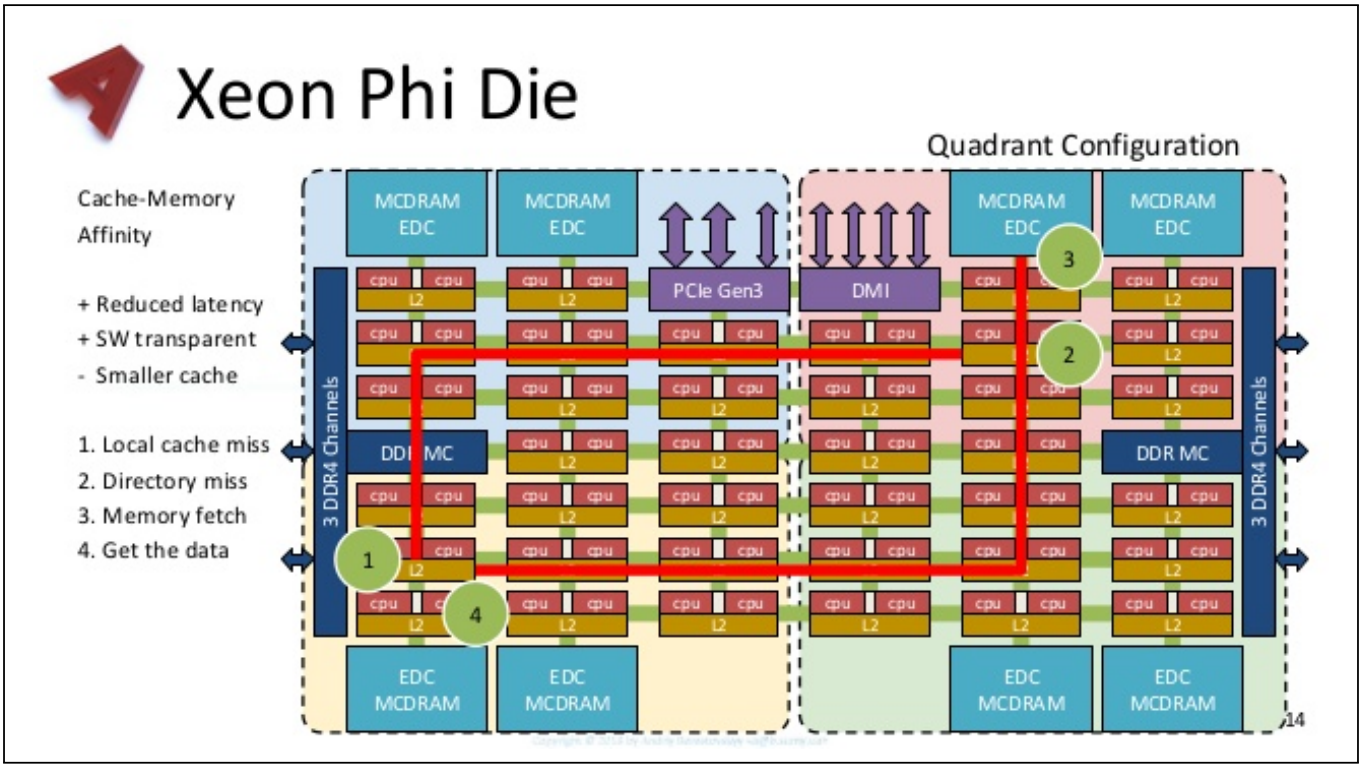
- 8 Core Complex Dies (CCDs).
- CCDs connect to memory, I/O, and each other through the I/O Die.
- 8 memory channels per socket.
- DDR4 memory at 3200MHz.
- PCI Gen4, up to 128 lanes of high speed I/O.
- Memory and I/O can be abstracted into separate quadrants each with 2 DIMM channels and 32 I/O lanes.
- 2 Core Complexes (CCXs) per CCD
- 4 Zen2 cores in each CCX share a16ML3 cache. Total of 16x16= 256MB L3 cache.
- Each core includes a private 512KB L2 cache.



[t]

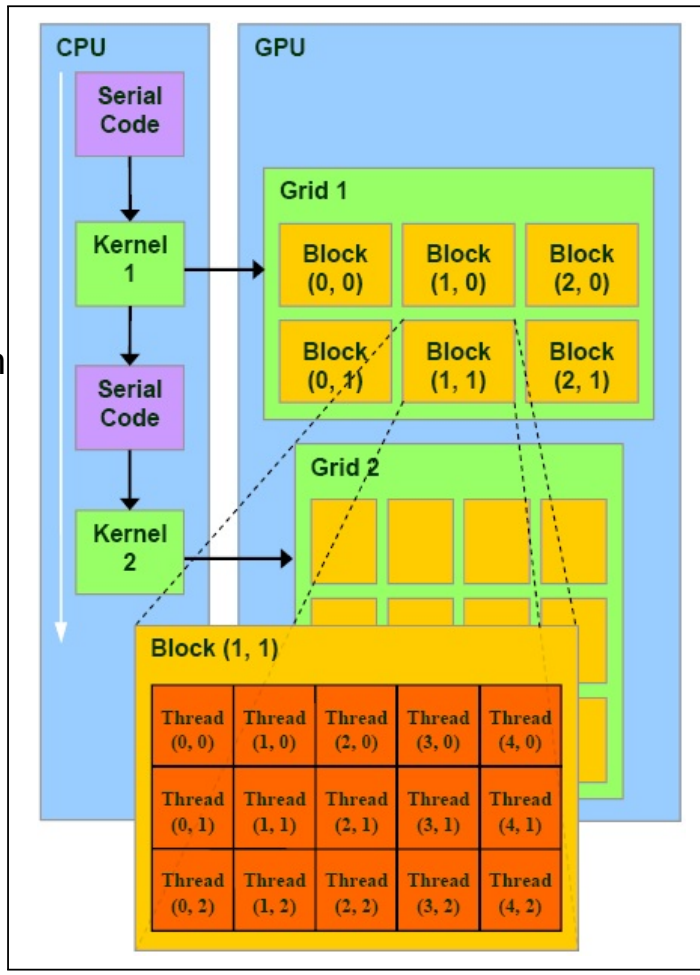
Source: https://hpc-training.sdsc.edu/expanse-101/

HPC Hardware: Intel Xeon Phi Processor with Many Integrated Core (MIC) architecture



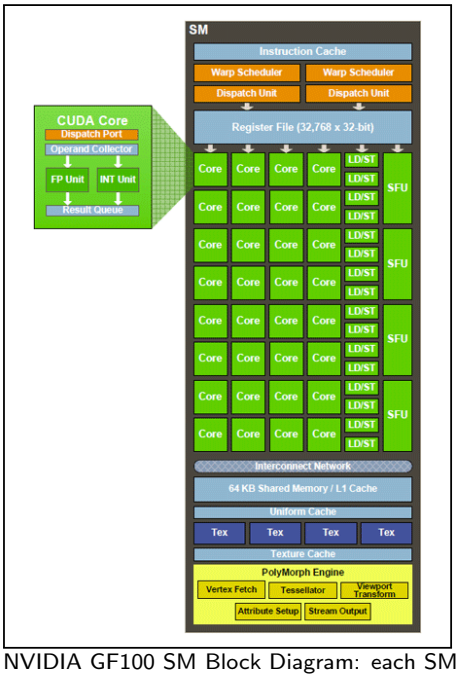
GPU Computing

- GPU is a highly threaded coprocessor to the host CPU and associated memory
- Kernels are sections of the application that are run on the GPU by a thread.
- A thread block is a batch of threads that can cooperate with each other by:
  - Sharing data through shared memory
  - Synchronizing their execution
  - Each block is organized as 3D array of threads:
  - (blockDim.x, blockDim.y, and blockDim.z)
- Threads within a thread block must:
  - execute the same kernel
  - share data, so they must be issued to thesame processor
- A grid is a collection of blocks:
  - A Grid is organized as a 2D array of blocks
  - (gridDim.x and gridDim.y)



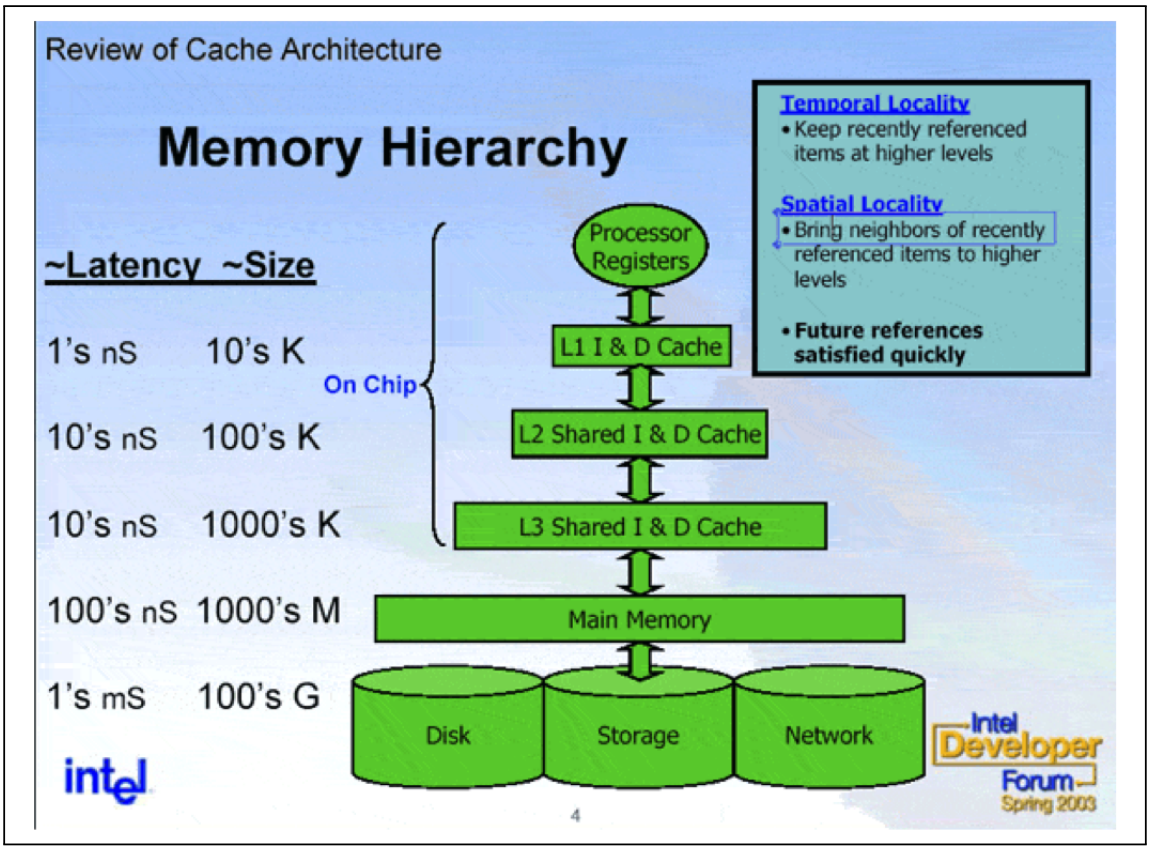
GPU Computing

NVIDIA GPU GF100 High-Level Block Diagram (2010): GPU device layout showing 4 Graphics Processing Clusters (GPC) with has 4 Streaming Multiprocessors (SMs); each SM has a block of Stream Processors (SPs) arranged in 8x4 layout.



NVIDIA GF100 SM Block Diagram: each SM has 32 cores.

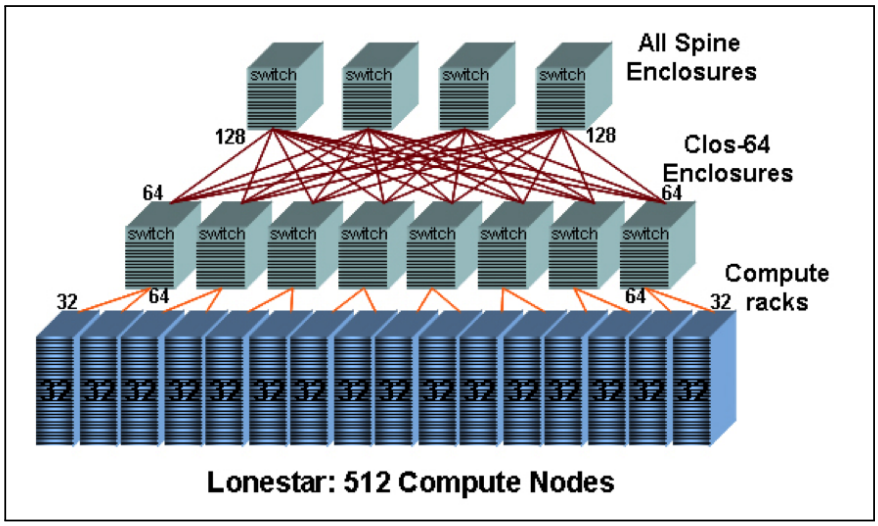
HPC Hardware: Memory



Source: `http://www17.tomshardware.com/cpu/20030811/images/intelgrafik.gif`

HPC Hardware: Interconnection Networks

- Myrinet network for 512 nodes
- 12 switches,each housing up to 16 leaf or spine cards
- each card has 8 fiber ports.
- leaf ports connect to hosts (1750 nodes)
- spine ports are used to form a hierarchical switch fabric.
- groups of 64 nodes are connected into 64 ports on each switch
- 8 ports x 8 leaf cards in 8 switches.

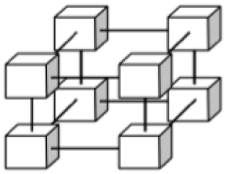


Sources: M. Thomas, CS596, 2007

HPC Hardware - Blue Gene/L Networks

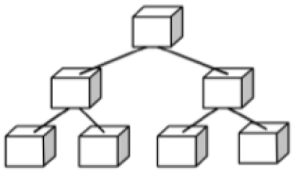
Blue Gene/L - The Networks

- 65536 nodes interconnected with three integrated networks



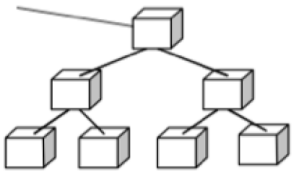
3 Dimensional Torus

- Virtual cut-through hardware routing to maximize efficiency
- 2.8 Gb/s on all 12 node links (total of 4.2 GB/s per node)
- Communication backbone
- 134 TB/s total torus interconnect bandwidth



Global Tree

- One-to-all or all-all broadcast functionality
- Arithmetic operations implemented in tree
- ~1.4 GB/s of bandwidth from any node to all other nodes
- Latency of tree traversal less than 1usec



Ethernet

- Incorporated into every node ASIC
- Disk I/O
- Host control, booting and diagnostics

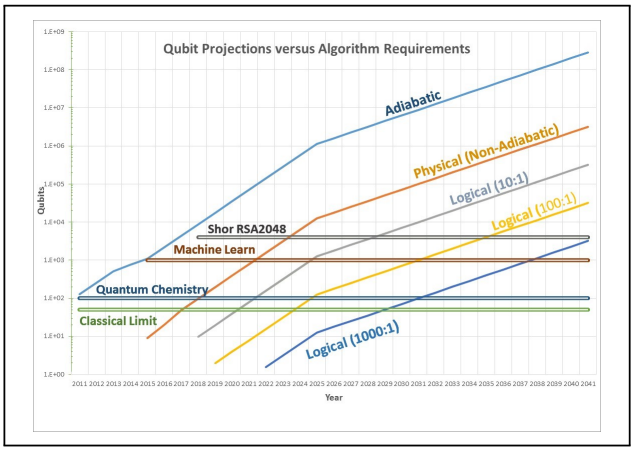
Aug-30-12

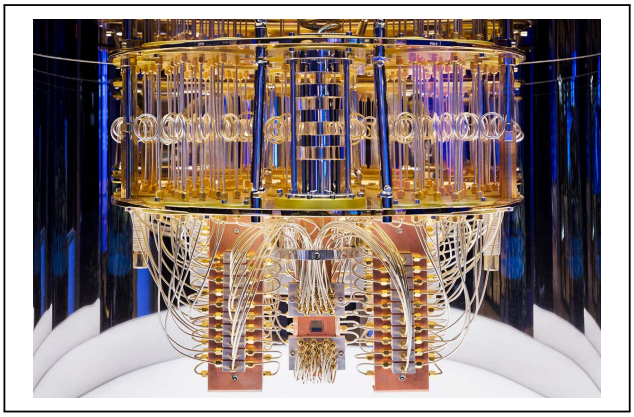
Source: http://www.research.ibm.com/bluegene/BG_External_Presentation_January_2002.pdf.

9

Source: M. Thomas, CS596, 2007

Quantum Computing





Ethernet Growth

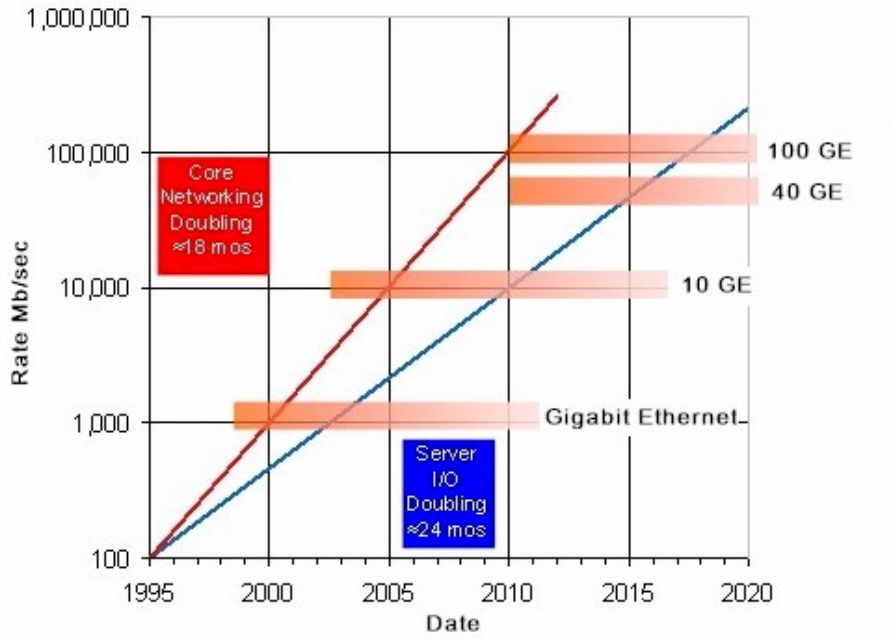


Figure: Source: `http://www.infinibandta.org/content/pages.php?pg=technology_overview`

InfiniBand

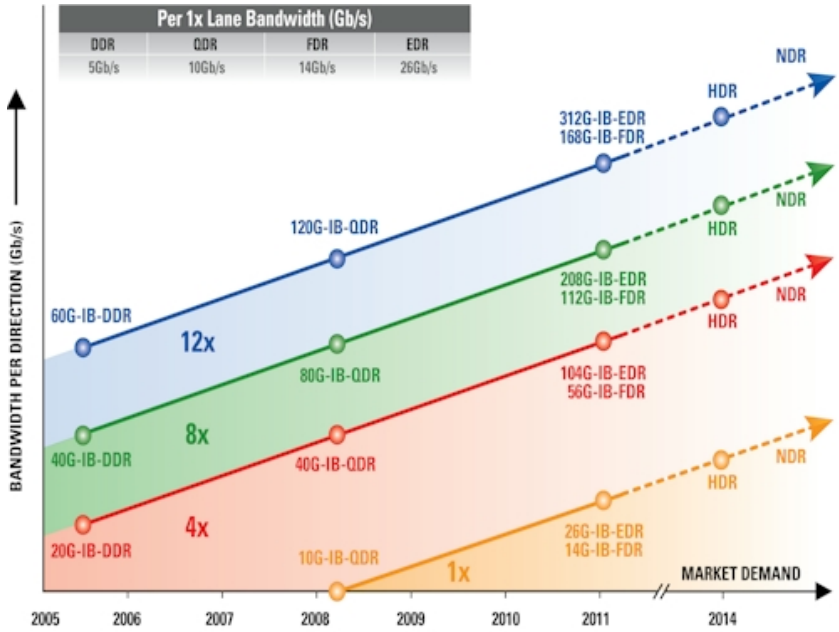


Figure: Source: `http://www.infinibandta.org/content/pages.php?pg=technology_overview`

Developing Parallel Algorithms

Developing Parallel Algorithms

Developing Parallel Algorithm Example: Serial Summation

Example

- Compute n values and add them together.
- Serial solution:

```
sum = 0;
for (i = 0; i < n; i++) {
   x = Compute_next_value(. . .);
   sum += x;
}
```


Simple Parallel Sum: Distribute Work to Different Cores

Example (cont.)

- We have p cores, p much smaller than n.
- Each core performs a partial sum of approximately n/p values.

```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
   my_x = Compute_next_value( . . .);
   my_sum += my_x;
}
```

Each core uses it's own private variables and executes this block of code independently of the other cores.

Simple Parallel Sum: Each Core Holds Its Results

- After each core completes execution of the code, is a private variable my_sum contains the sum of the values computed by its calls to Compute_next_value.
- Ex 8 cores, n = 24, then the calls to Compute_next_value return:
- [1,4,3] [9,2,8] [5,1,1] [5,2,7] [2,5,0] [4,1,8] [6,5,1] [2,3,9]
- Once all the cores are done computing their private my_sum, they form a global sum by sending results to a designated ÒmasterÓ core which adds the final result.

Source: Pacheco, 2011 Textbook

Simple Parallel Sum: Gather Results to One Core

Example (cont.)

```
if (I'm the master core) {
   sum = my_x;
   for each core other than myself {
      receive value from core;
      sum += value;
   }
} else {
   send my_x to the master;
}
```

Simple Parallel Sum: Data Distribution

Example (cont.)

Core	0	1	2	3	4	5	6	7
my_sum	8	19	7	15	7	13	12	14

Global sum

8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95

Core	0	1	2	3	4	5	6	7
my_sum	95	19	7	15	7	13	12	14

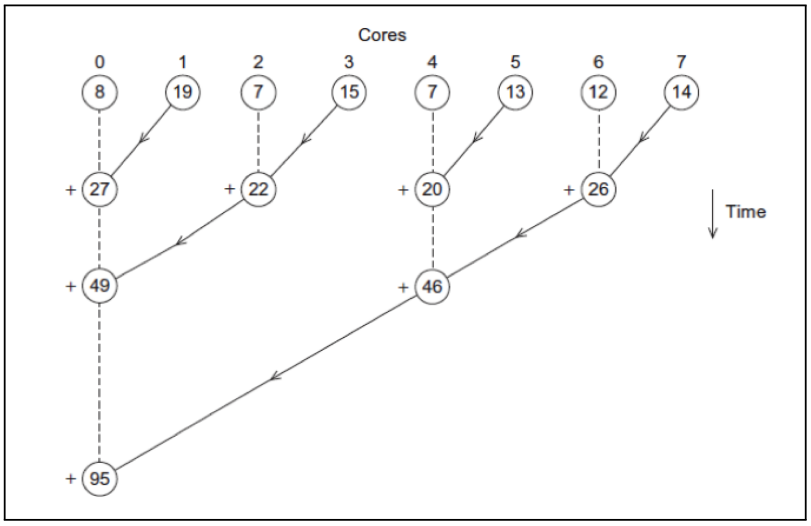
Better parallel algorithm: Reduce Master Core Workload

- After core completes execution, local variable my_sum contains the sum of the values computed by its calls to Compute_next_value.
  - Pair the cores so that core 0 adds its result with core 1's result.
  - Core 2 adds its result with core 3's result, etc.
  - Work with odd and even numbered pairs of cores.
  - Repeat the process now with only the evenly ranked cores.
  - Core 0 adds result from core 2.
  - Core 4 adds the result from core 6, etc.
- Now cores divisible by 4 repeat the process, and so forth, until core 0 has the final result.

Source: Pacheco, 2011 Textbook

Simple Parallel Sum: Tree-structured Addition

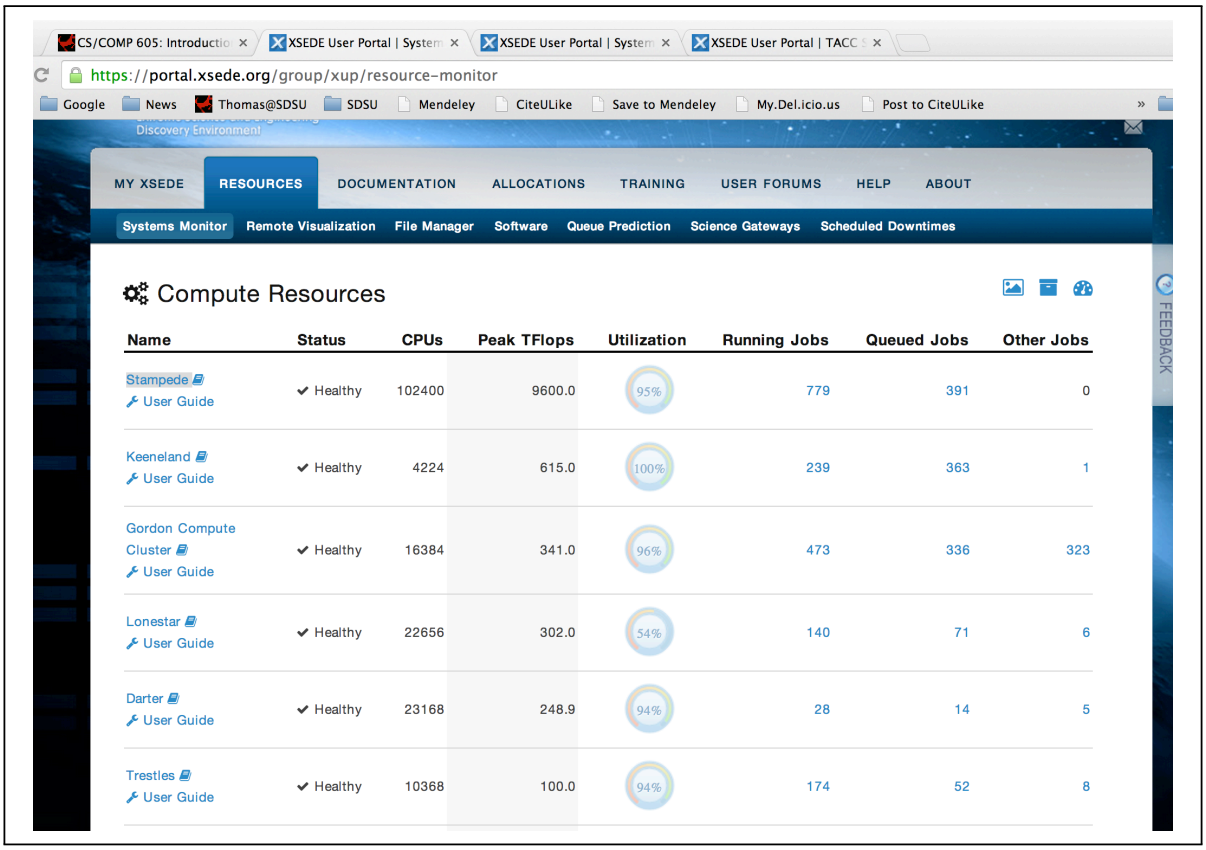
- Soltn 1: master core - 7 recvs & 7 adds.
- Soltn 1: master core - 3 recvs & 3 adds.
- Improvement greater than 2!
- for 10^3 cores,
  - Soltn 1: 999 recvs & 999 adds.
  - Soltn 1: 10 recvs & 10 adds.



Ways to Improve Parallel Algorithms

- Develop parallel version of serial problems
  - task-paralellism: distribute different tasks among cores (stock market transactions)
  - data-paralellism: distribute the data among cores, each core does the same computation
- Develop parallel hardware
  - fast core speed, large Ram memory, fast interconnections
  - develop parallel software libraries to simplify programming env.

HPC Hardware - NSF XSEDE Project



Name	Status	CPUs	Peak TFlops	Utilization	Running Jobs	Queued Jobs	Other Jobs
Stampede User Guide	Healthy	102400	9600.0	95%	779	391	0
Keeneland User Guide	Healthy	4224	615.0	100%	239	363	1
Gordon Compute Cluster User Guide	Healthy	16384	341.0	96%	473	336	323
Lonestar User Guide	Healthy	22656	302.0	54%	140	71	6
Darter User Guide	Healthy	23168	248.9	94%	28	14	5
Trestles User Guide	Healthy	10368	100.0	94%	174	52	8

Source: `http://www.xsede.org`

Next Time

- Next session: 01/29/21
- Topics: Unix basics/Using student cluster, etc., Introduction to HPC (Part 2)
- Homework 1 will be posted today.