"For over a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers"

Gene Amdahl, Engineer at IBM, 1967

Lecture A.1:
Parallel Processing Architectures

CS205: Computing Foundations for Computational Science
Dr. David Sondak
Spring Term 2021

HARVARD
School of Engineering and Applied Sciences

IACS
INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

Lectures developed by Dr. Ignacio M. Illorente

Before We Start

Where We Are

Computing Foundations for Computational and Data Science

How to use modern computing platforms in solving scientific problems

Intro: Large-Scale Computational and Data Science

A. Parallel Processing Fundamentals

- A.1. Parallel Processing Architectures
- A.2. Large-scale Processing on the Cloud
- A.3. Practical Aspects of Cloud Computing
- A.4. Application Parallelism
- A.5. Designing Parallel Programs

B. Parallel Computing

C. Parallel Data Processing

Wrap-Up: Advanced Topics

CS205: Contents

APPLICATION SOFTWARE

APPLICATION PARALLELISM

PARALLEL PROGRAM DESIGN

PROGRAMMING MODEL

Optimization

OpenACC

OpenMP

MPI

Spark

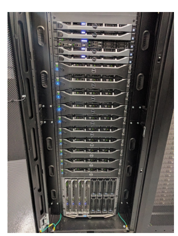
Map-Reduce

B. BIG COMPUTE

PLATFORM

C. BIG DATA

amazon web services



Open Nebula

FAS RC

CLOUD COMPUTING

FAS RC CANNON HARVARD'S LARGEST CLUSTER

PARALLEL ARCHITECTURES

Context

What Do They Have in Common?

Google Datacenter

100K cores

iPhone X

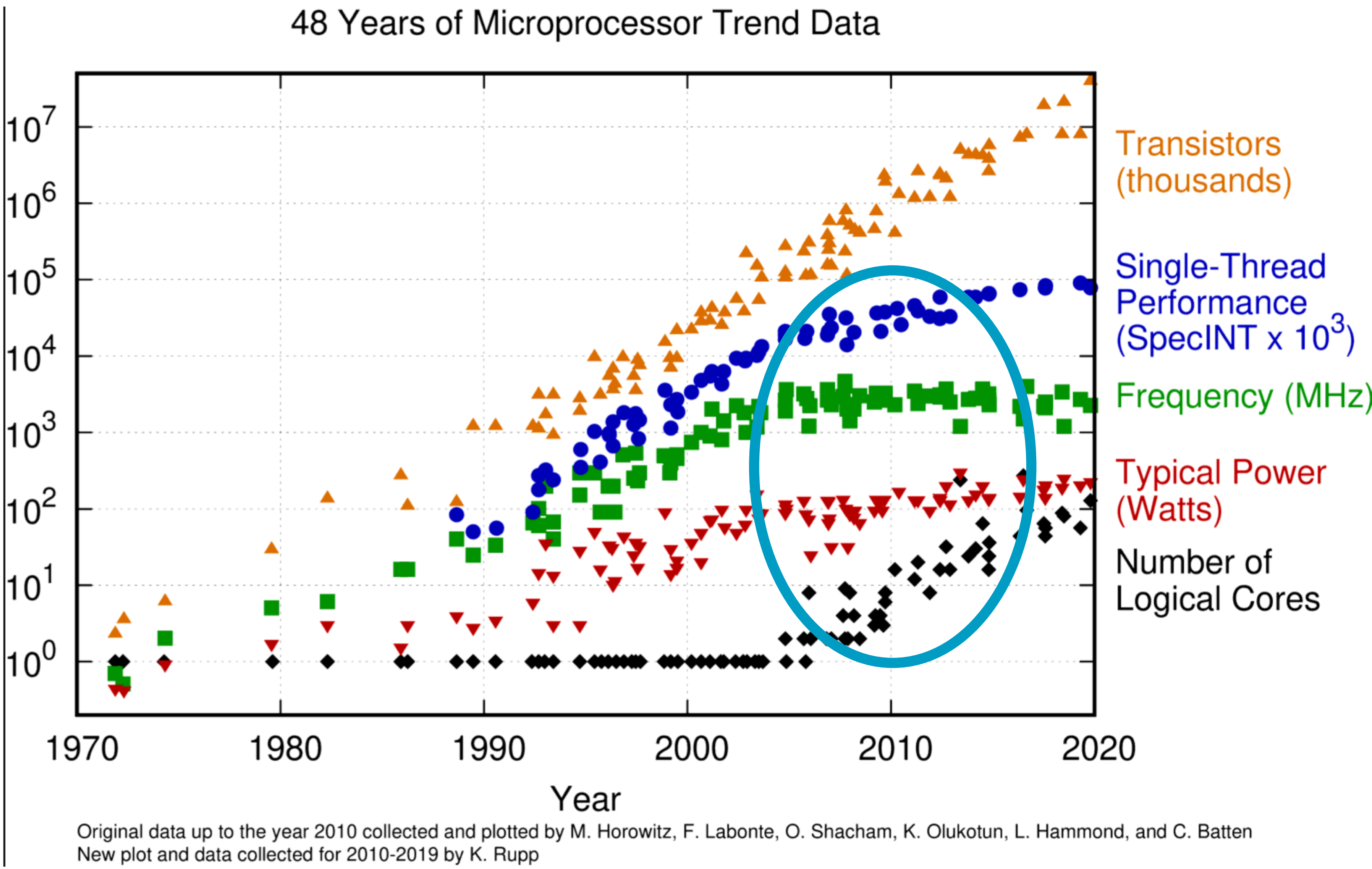
6 cores

MacBook Pro

10 cores

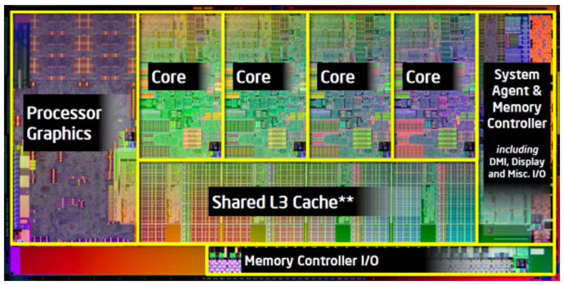
Context

Problem with Clocks



Context

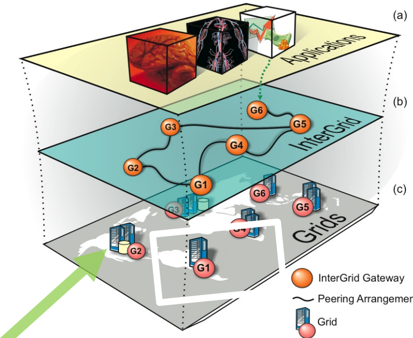
The Path to More Computing



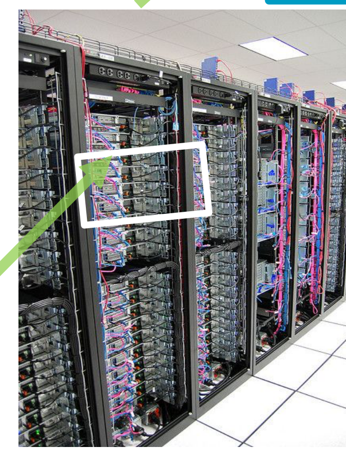
Multi-core

Multi-CPU

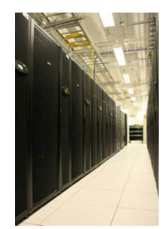
GPU (Many-core)



Multi-site

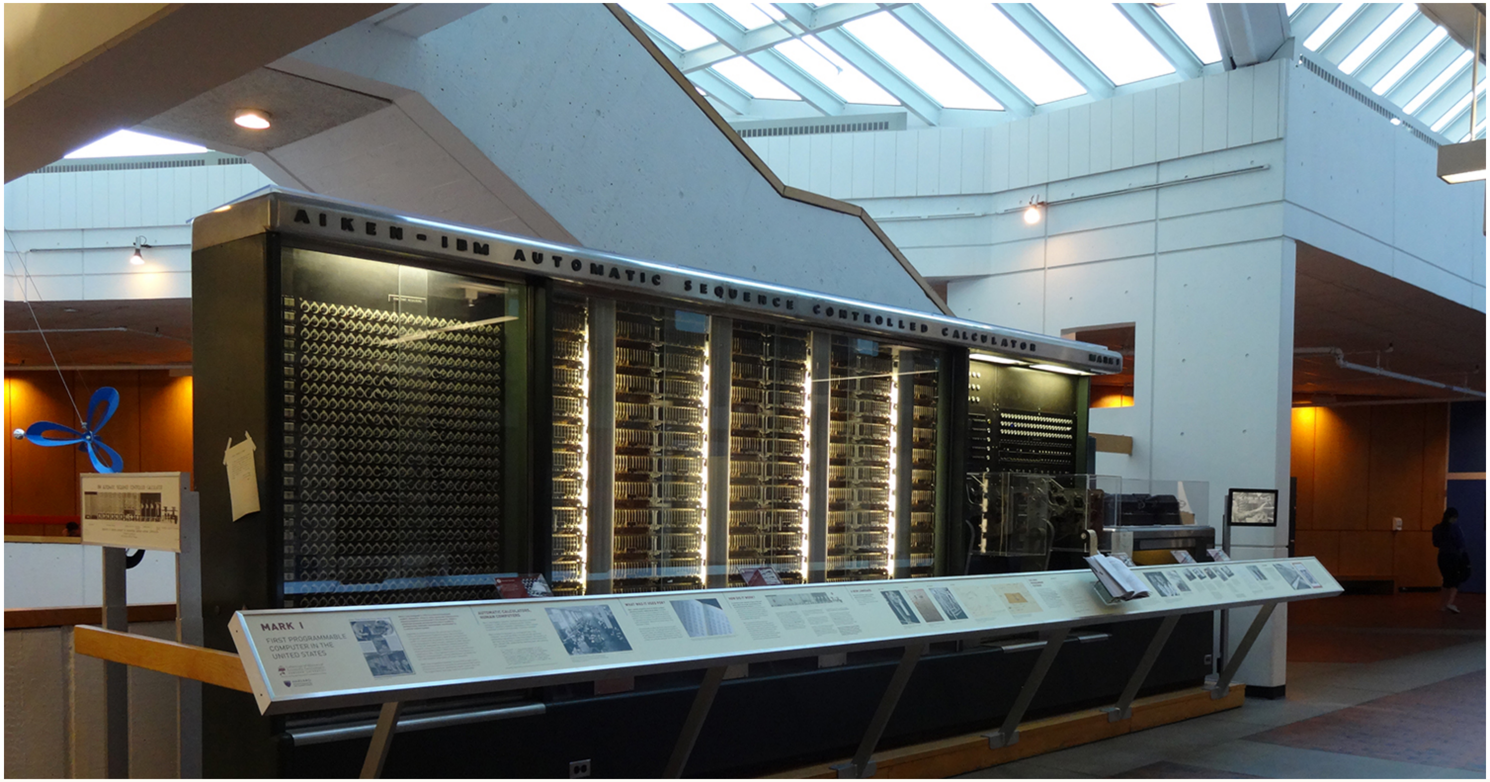


Multi-node



Context

Revolution Started HERE!



Mark I was designed in 1937 by a Harvard graduate student, Howard H. Aiken to solve advanced mathematical physics problems encountered in his research. Aiken's ambitious proposal envisioned the use of modified, commercially-available technologies coordinated by a central control system.

https://chsi.harvard.edu/harvard-ibm-mark-1

Roadmap

Parallel Processing Architectures

Shared-Memory Parallel Architectures

Accelerated Computing (GPUs)

Distributed-Memory Parallel Architectures

Benchmarking

Local Resource Managers

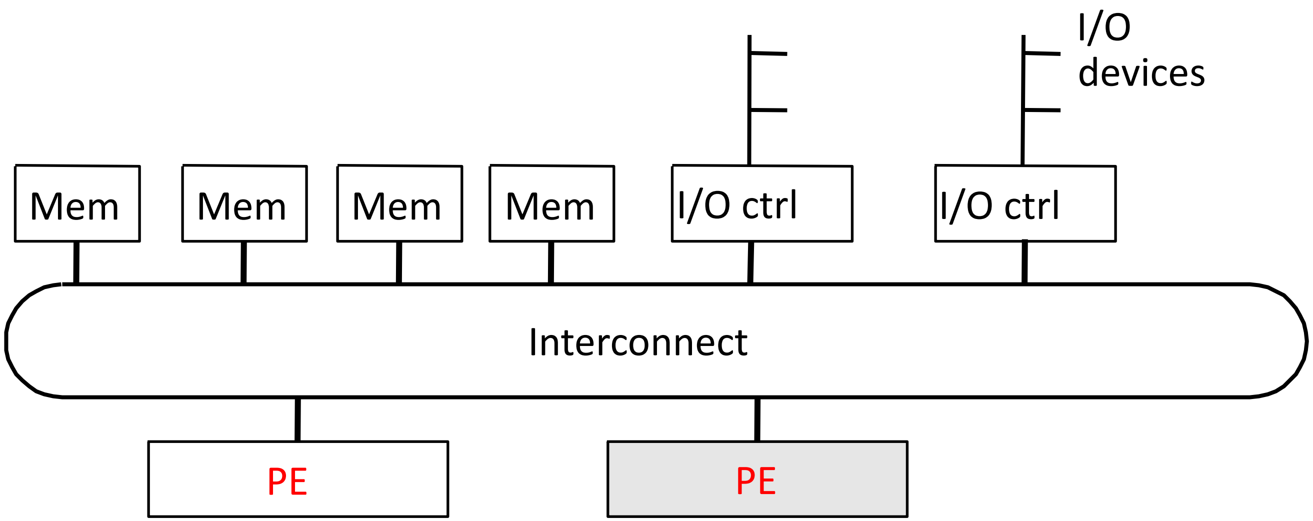
Grid Computing

Shared-Memory Parallel Architectures

The Natural Approach to Grow Performance

Easy Programming and Administration

- All the processing elements share the physical memory uniformly
- Single OS for the whole system, support both processes and threads

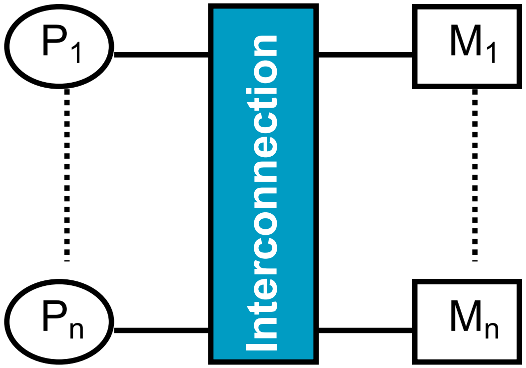


Shared-Memory Parallel Architectures

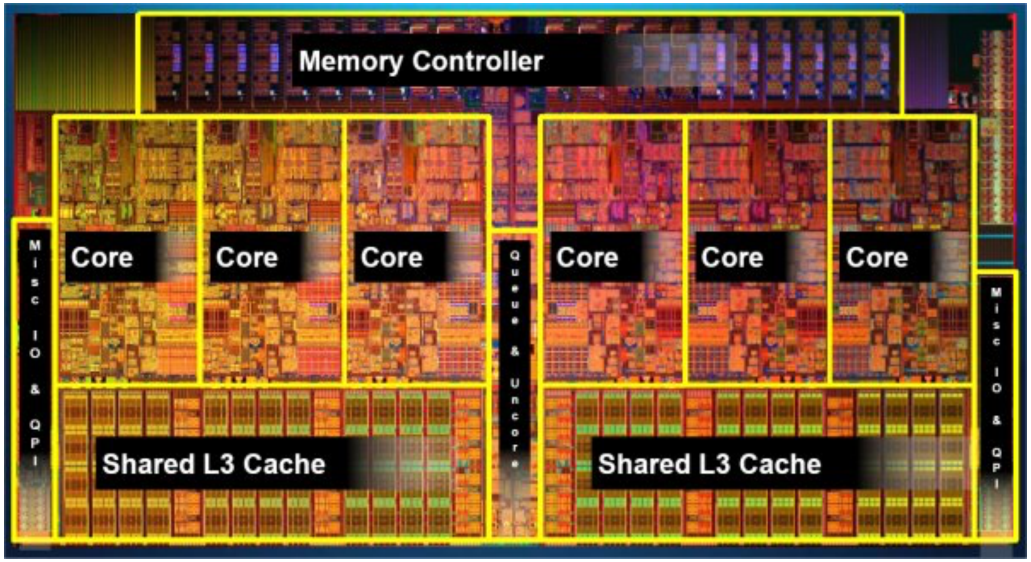
Uniform Memory Access

UMA

- Access time to a memory location is independent of which element makes the request or which memory chip contains the transferred data



UMA
(Uniform Memory Access)
SMP



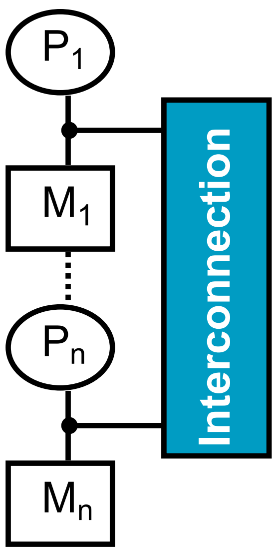
multi-core processors
Intel i7 980x (Extreme)
6 cores

Shared-Memory Parallel Architectures

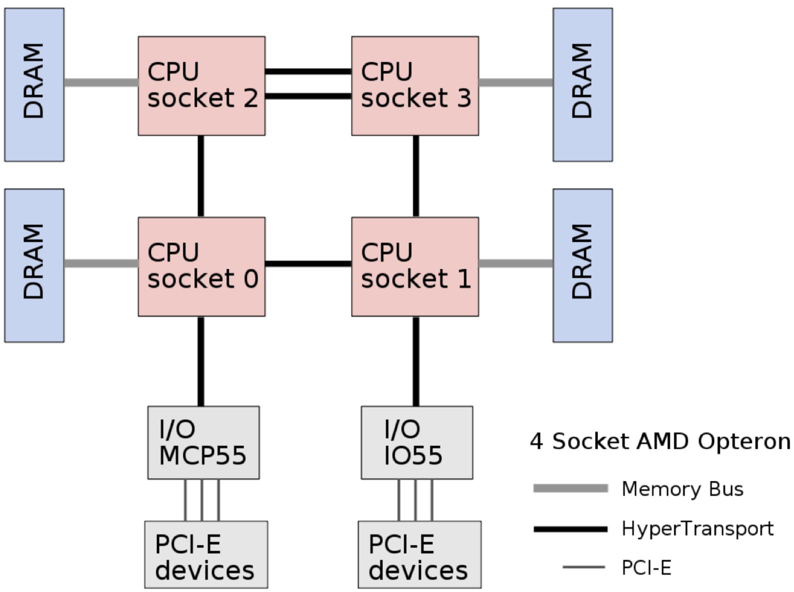
Non-Uniform Memory Access

NUMA

- Memory access time depends on the memory location relative to the processor
- Shared memory programming with tuning for data location



NUMA
(Non-Uniform Memory Access)



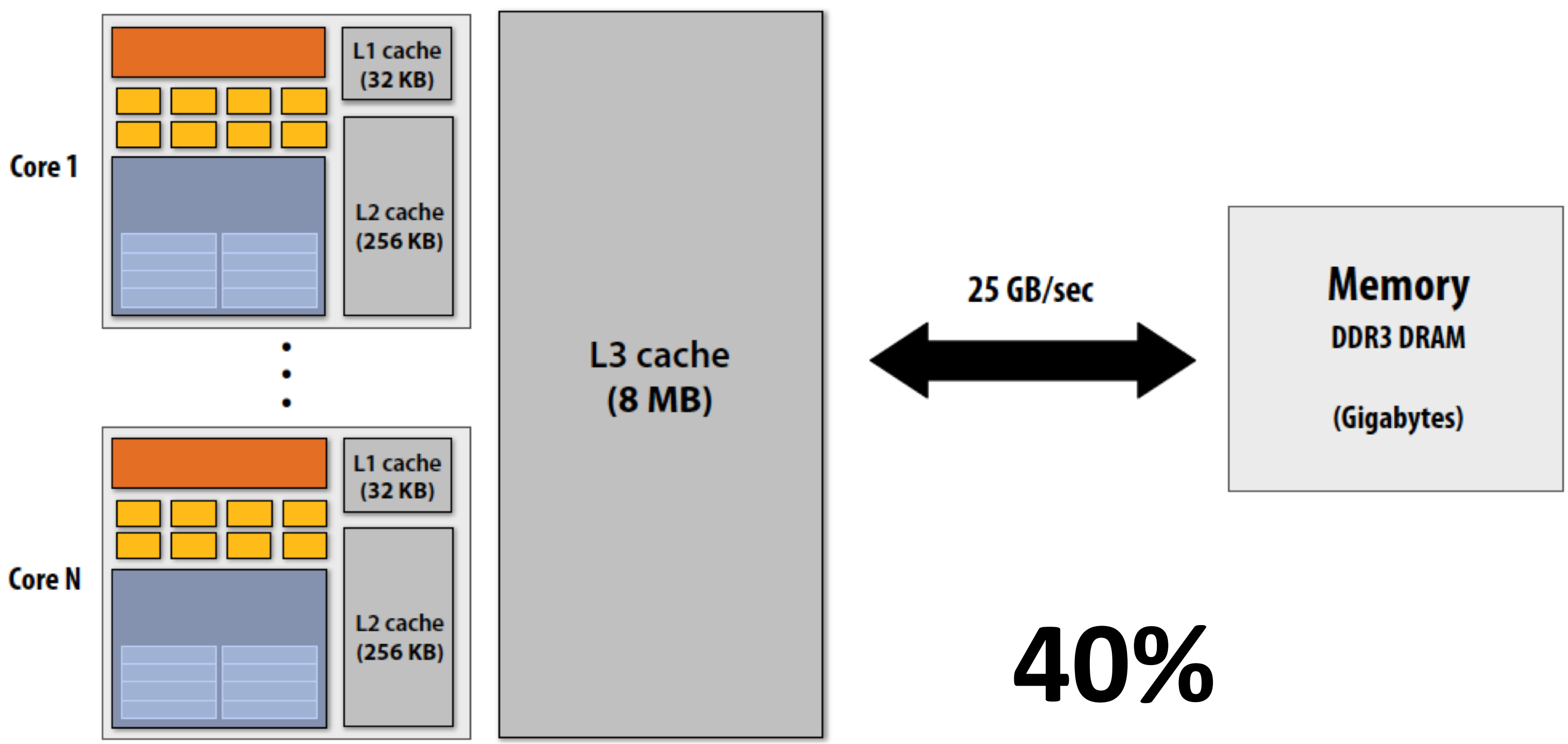
multi-processor motherboards

Shared-Memory Parallel Architectures

Main Downside

Scalability

- Add lots of cache: Hides latency!

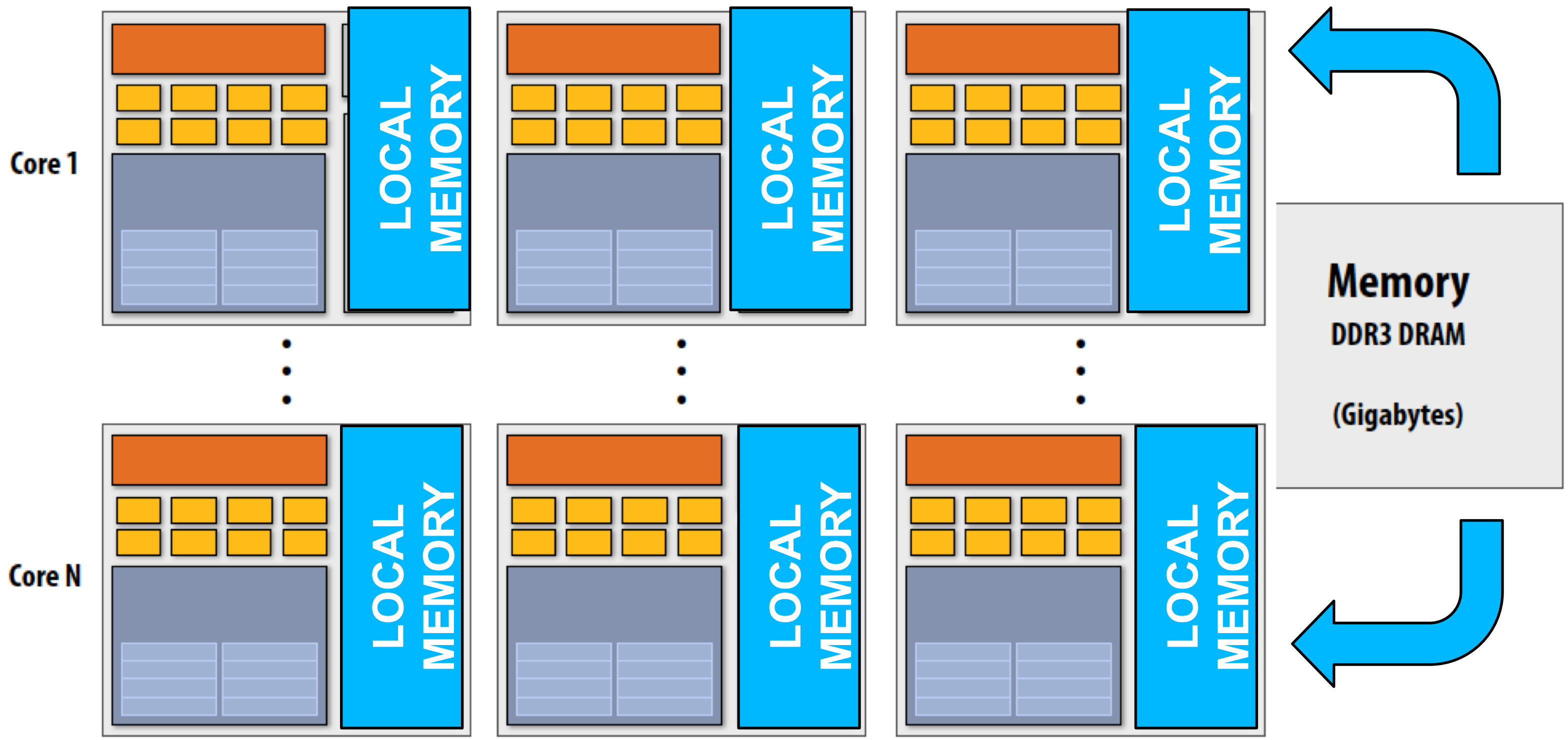


40%

Source: “From Shader Code to a Teraflop: How Shader Cores Work“, Kayvon Fatahalian, Stanford University

Accelerated Computing

Many-core



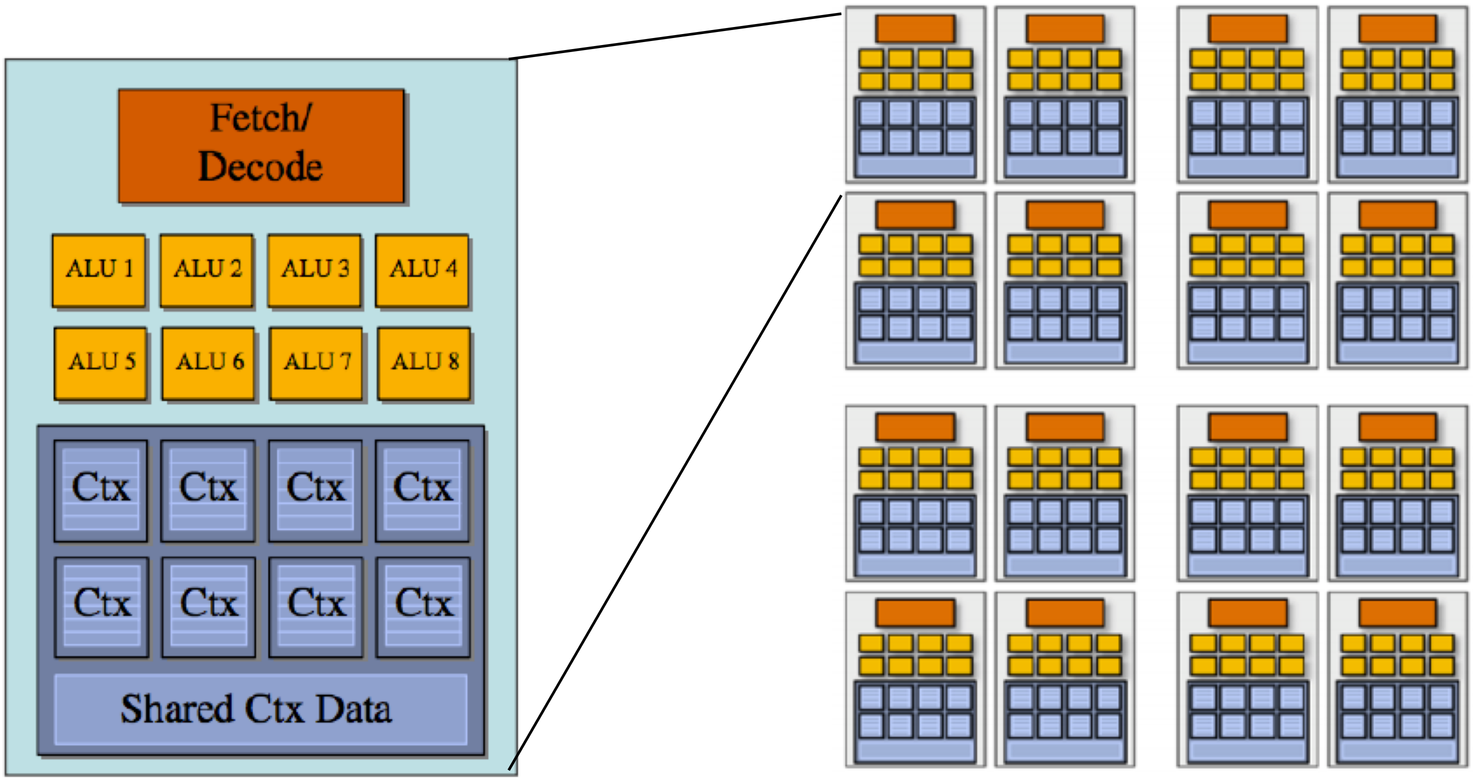
Source: “From Shader Code to a Teraflop: How Shader Cores Work“, Kayvon Fatahalian, Stanford University

Accelerated Computing

Many-core

Add SIMD Processing on Many Cores

- Amortize cost/complexity of managing an instruction stream across many ALUs

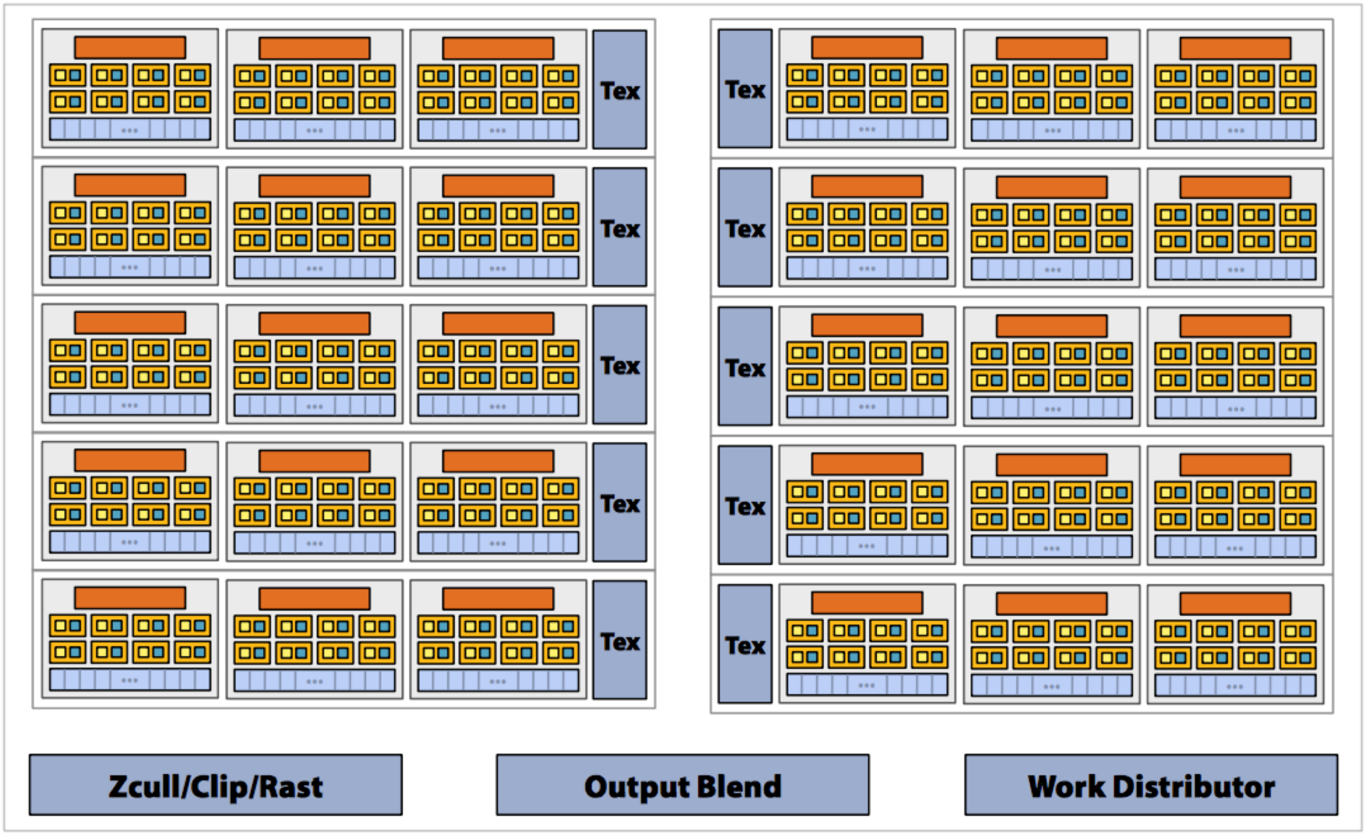


16 cores x 8 ALUs/core = 128 ALUs (mul-add) => 256 GFLOPs @1GHz

Accelerated Computing

An Example

Example: NVIDIA GTX 280



30 cores x 8 ALUs/core = 240 ALUs (3 FLOPS) => 933 GFLOPs @1.3GHz

Accelerated Computing

Main Downside

GPU-friendly Applications

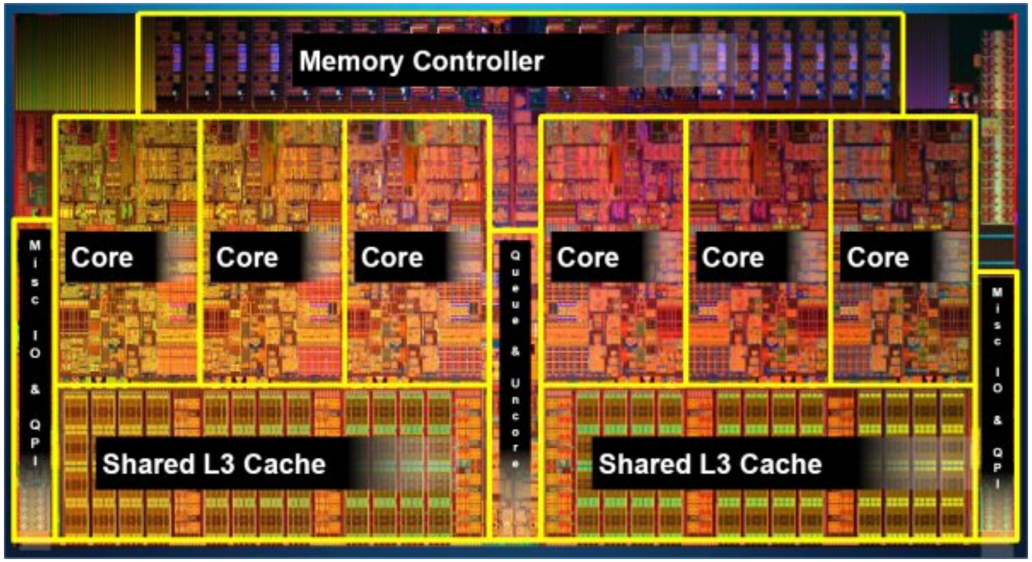
- Computer graphics
- Texture, rendering, image processing…
- Matrix operations
- Structured simulations (finite differences)

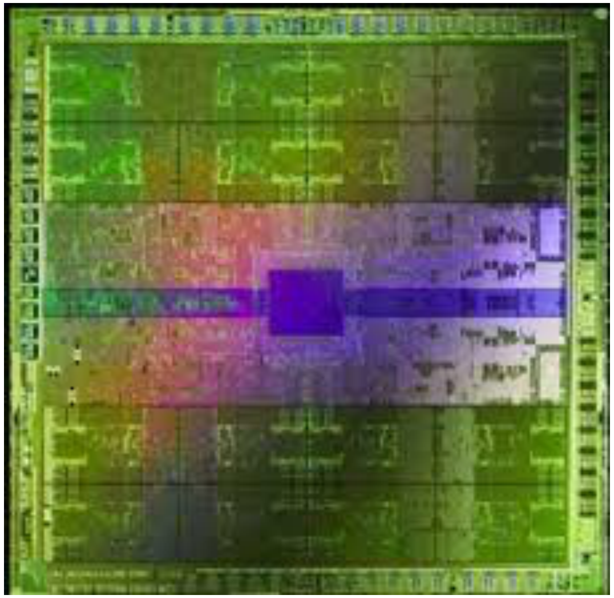
Downsides

- Not-general purpose CPUs
- Difficult to program
- Difficult to tune: Bandwidth vs. Compute vs. Context
- CPU-GPU link has been slow, historically (system bus)

Accelerated Computing

Multi-core vs Many-core





Intel i7 980x (Extreme)
6 cores
1.2B transistors

NVIDIA GTX 580 SC
512 cores
3B transistors

Cache and memory hierarchy vs more cores and ALUs

Optimized for low-latency access to cache
Complex control logic for ILP

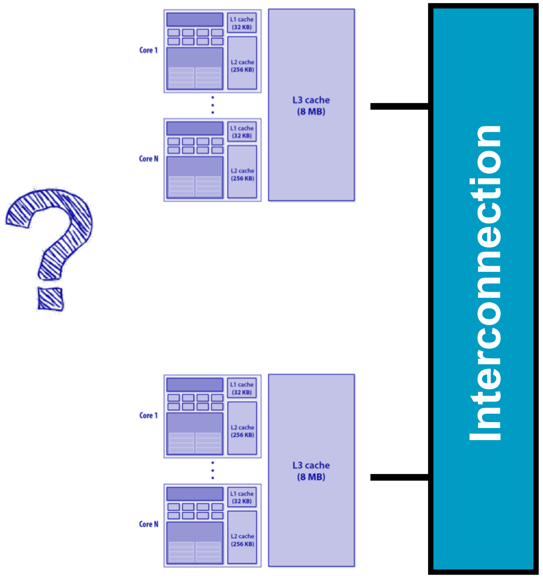
Optimized for data-parallel, throughput computation
More transistors for computation

Distributed-Memory Parallel Architectures

Scaling Shared-Memory Systems

Scalability

- Grow the system beyond a single shared-memory multi-processing node
- Local private memory and OS instance within each node
- **Require Job Manager** | DRMS (Distributed Resource Management System)



U.S. DOE Oak Ridge National Laboratory

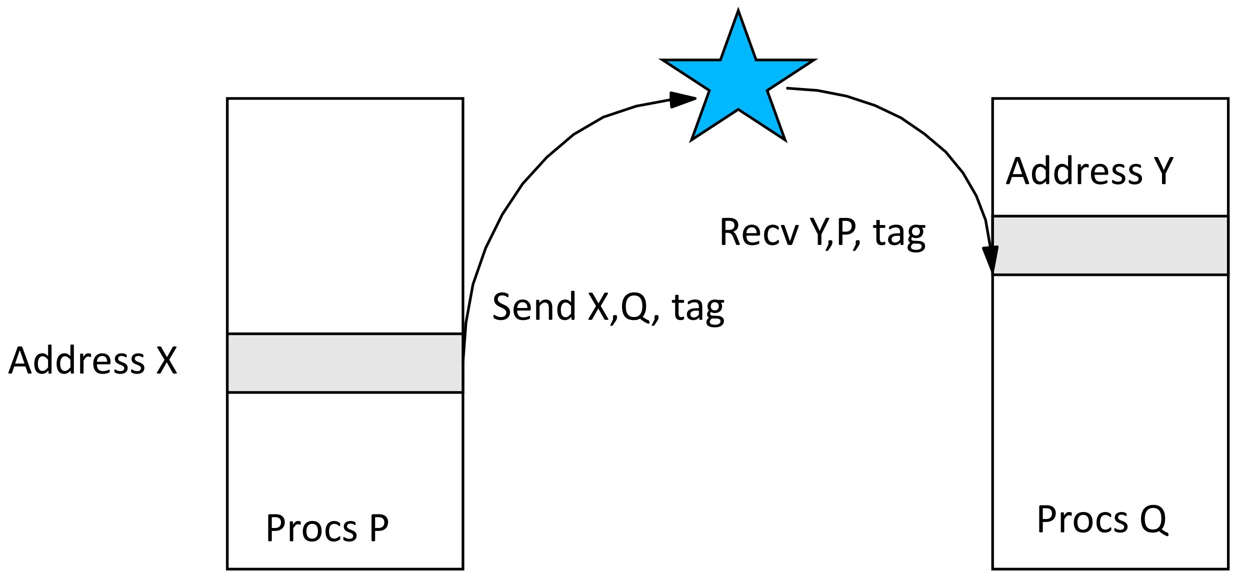
Harvard's Cannon

Distributed-Memory Parallel Architectures

Main Downside

Programming

- Communication through message passing



- **<u>Hybrid parallel programming</u>**

Distributed-Memory Parallel Architectures

Different Approaches

Centralized
Coupled

- **Network Links**
- **Administration**
- **Homogeneity**

Decentralized
Decoupled

MPP (Massive Parallel Processors)

Dedicated Clusters

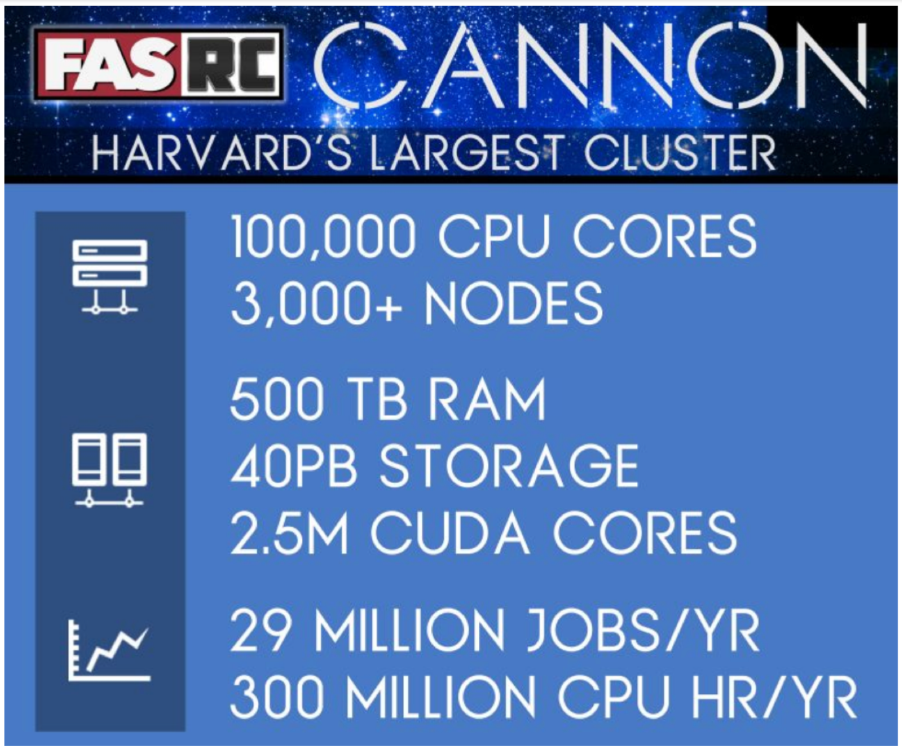
Network Systems Intranet/Internet

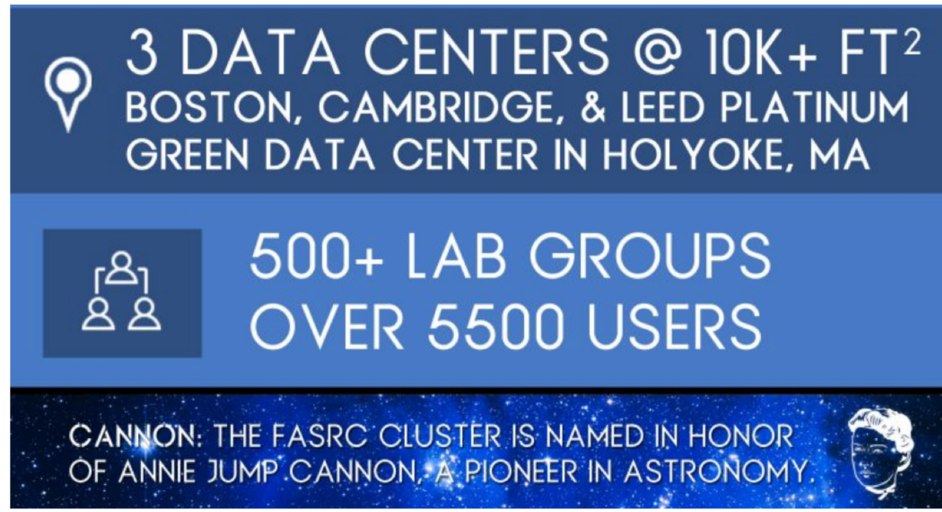
High Performance Computing

High Throughput Computing

Distributed-Memory Parallel Architectures

Example: Harvard's Cannon





Distributed-Memory Parallel Architectures

Example: Harvard’s Cannon

Interconnection Network

- Traditional TCP/IP network
- Low-latency 100 Gb/s InfiniBand network for inter-node parallel-computing and fast access to Lustre storage

Software

- SLURM
- CentOS
- Puppet
- 1000+ Scientific software tools and programs

Storage

- 40+ PB of storage spread out over various form factors with differing characteristics.
- Use case examples include: Robust home directories on enterprise storage, Lustre filesystem-based and performance driven scratch and research repositories, and middle tier laboratory storage using Gluster and NFS filesystems.

Compute

- Primarily comprised of 670 Lenovo SD650 NeXtScale servers
- Each chassis unit contains 2 nodes, each with 2 Intel 8268 “Cascade Lake” processors and 192 GM RAM per node
- GPUs: 16 Lenovo SR670 each w/ 32 CPU cores, 384 GB memory, 4 x V100s (@ 5,120 CUDA cores each)

Benchmarking

Top500

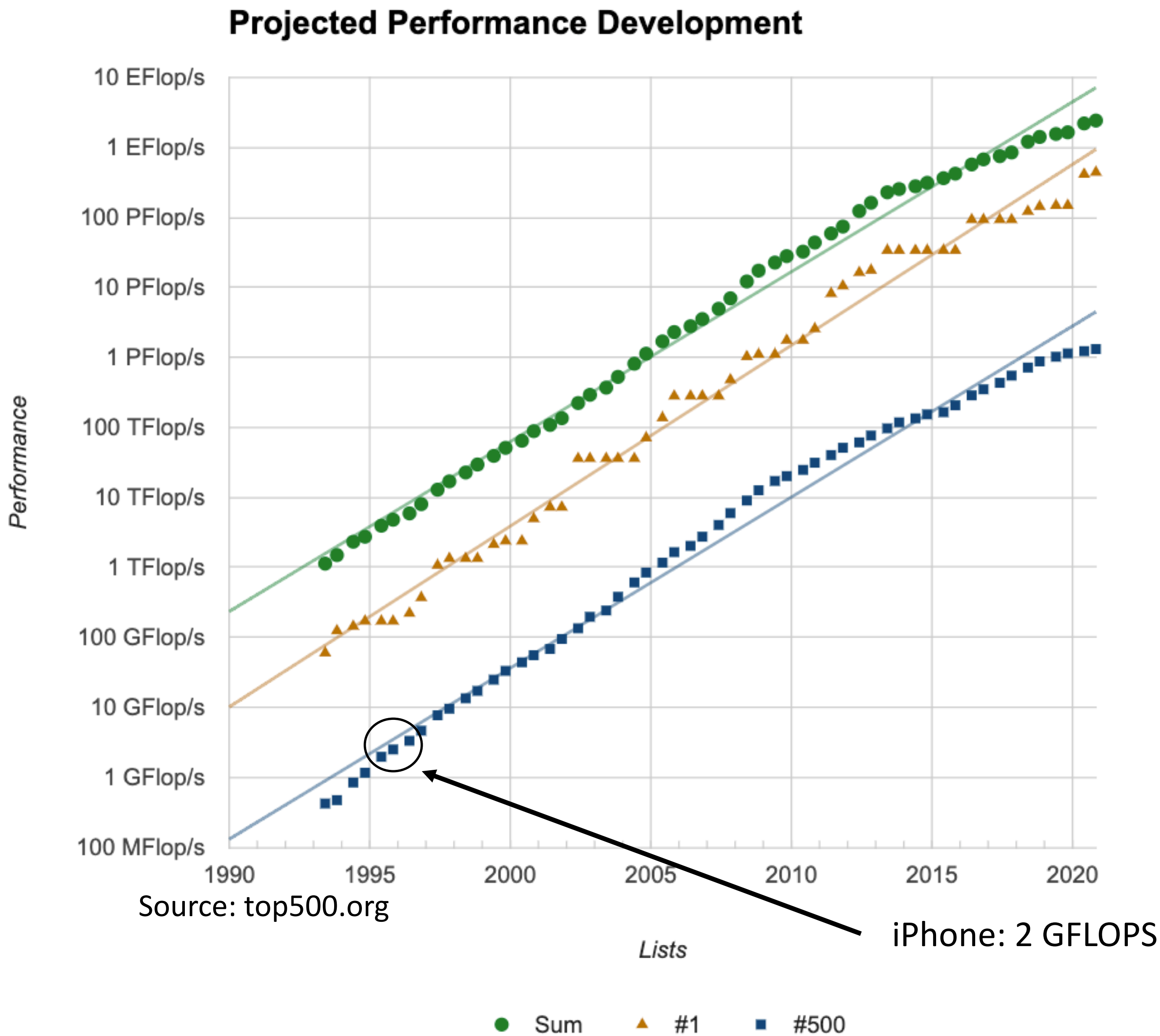
- Ranks the 500 most powerful parallel computers in the world
- Based on high-performance LINPACK benchmark (Fortran)
- Started in 1993 and updated list twice a year (SC in US and EU)

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	7,630,848	442,010.0	537,212.0	29,899
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	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646

November 2020 - source: top500.org

Benchmarking

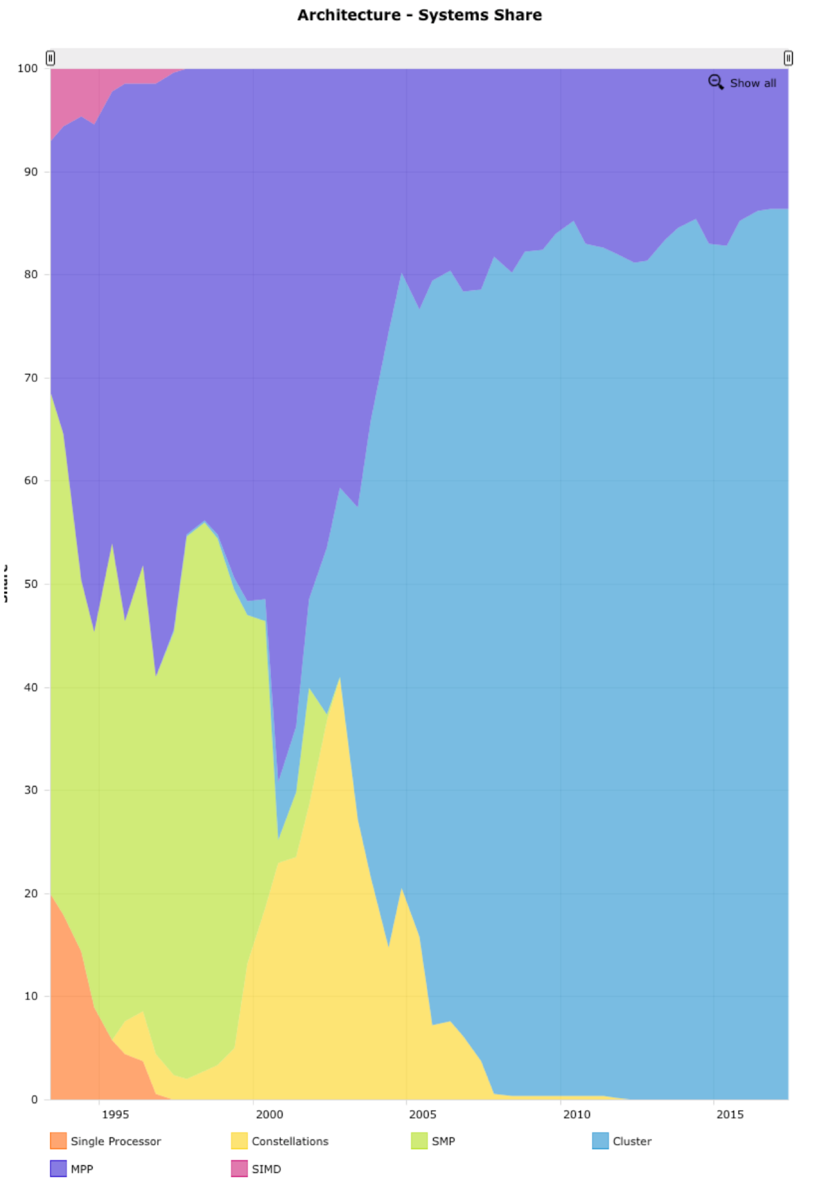
Future Performance of HPC (according to Top500)



Source: top500.org

Benchmarking

Historical Charts of Top500



Breakout Room

10 mins

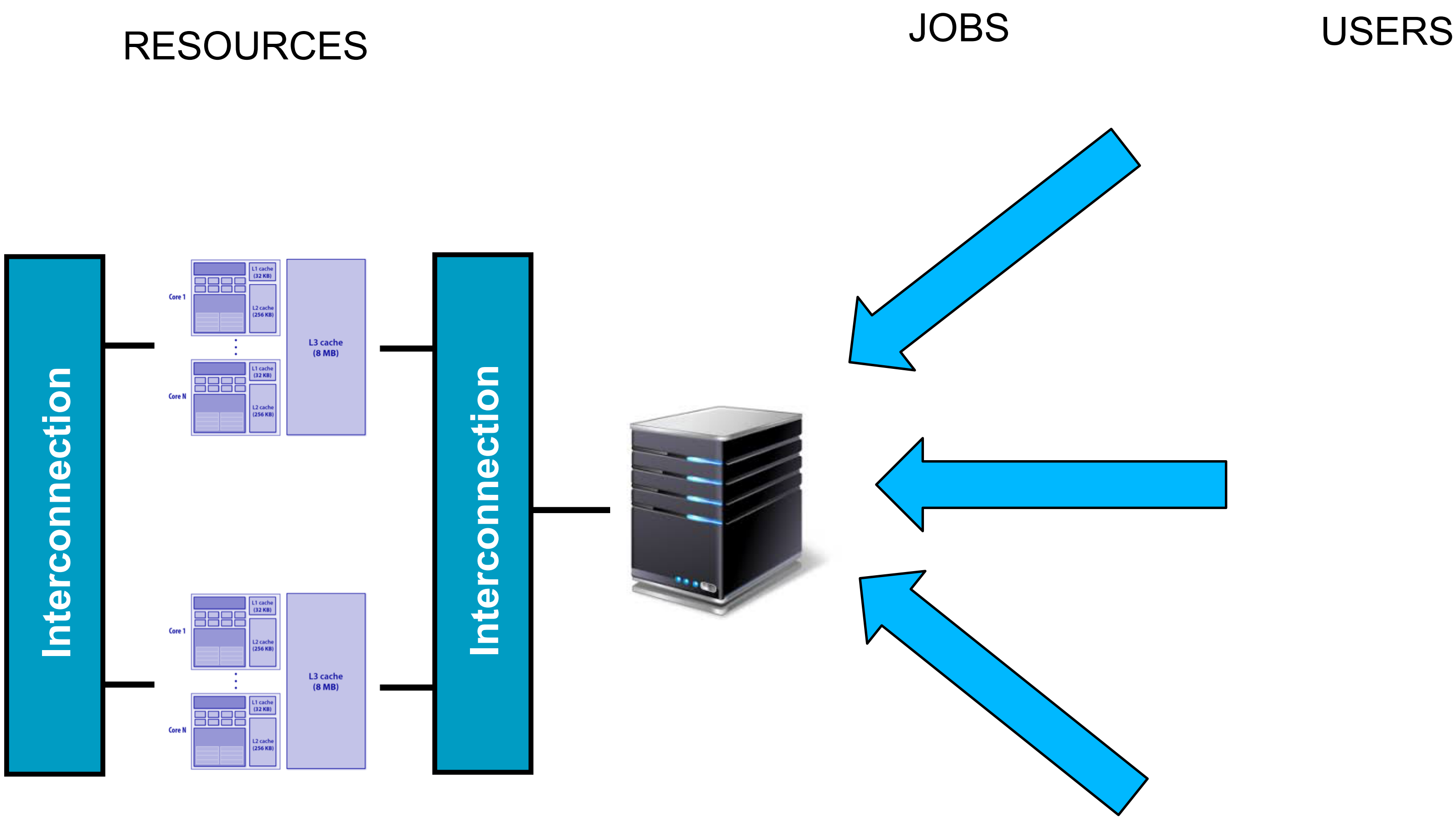
- Determine (roughly) who is closest to 0 degrees longitude and 0 degrees latitude

- Review some of the terminology from today:
  - Shared memory
  - Distributed memory
  - Accelerated computing
  - ALUs and how to calculate performance in FLOPs

- Visit and explore the Top500 website:
  - What is your favorite stat?

Local Resource Managers

How to Manage the Allocation of Resources to Jobs?



Local Resource Managers

Simple Linux Utility for Resource Management

- **Simple Linux Utility for Resource Management**
  User tasks (jobs) on the cluster are controlled by slurm and isolated in cgroups so that users cannot interfere with other jobs or exceed their resource request (cores, memory, time)
- **Basic SLURM commands:**
  - sbatch: submit a batch job script
    >$ sbatch [options for resource request] myscript
  - srun: submit an interactive test job
    >$ srun --pty [options for resource request] /bin/bash
  - squeue: contact slurmctld for currently running jobs
    >$ squeue
  - sacct: contact slurmdb for accounting stats after job ends
    >$ sacct
  - scancel: cancel a job(s)
    >$ scancel somejobnumber

FASRC

https://rc.fas.harvard.edu/resources/documentation/convenient-slurm-commands/

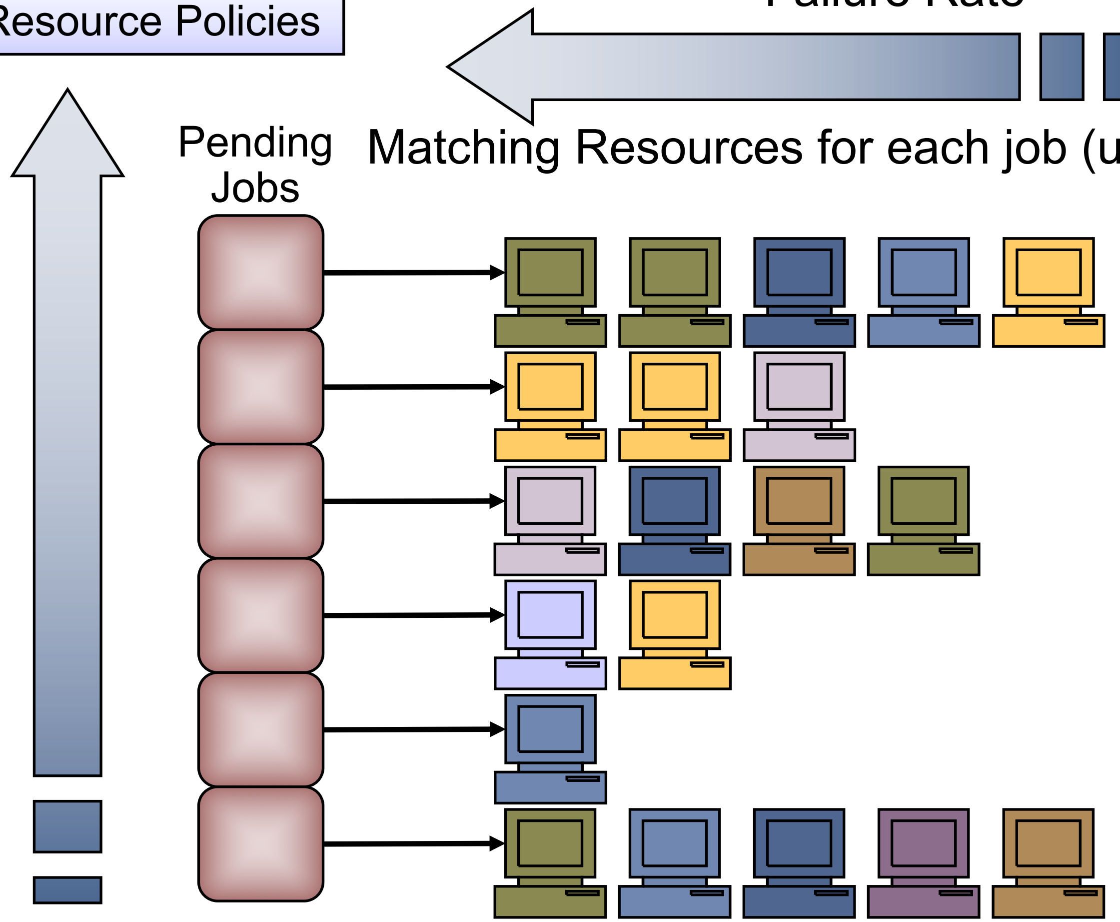
https://rc.fas.harvard.edu/resources/running-jobs/

Local Resource Managers

Simple Linux Utility for Resource Management

Resource Policies
- Rank Expressions
- Fixed Priority
- User Usage History
- Failure Rate

Scheduling = Job + Resource Policies

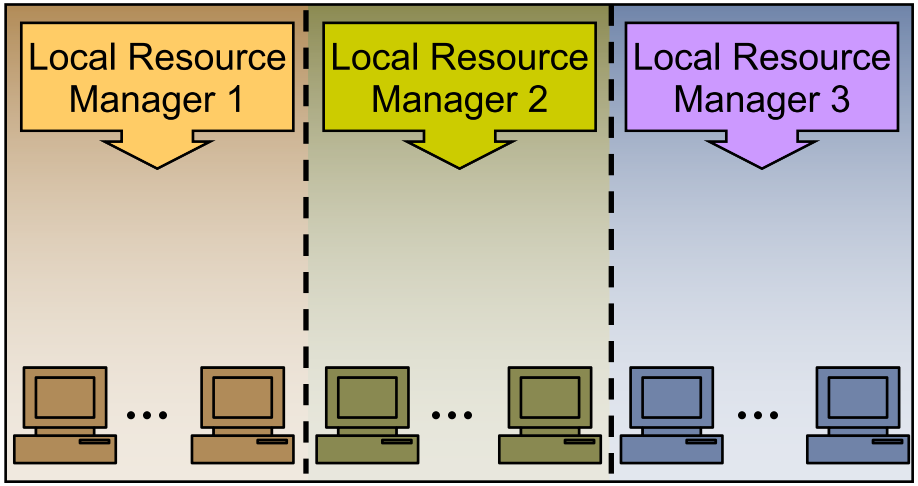


Job Policies
- Fixed Priority
- Urgent Jobs
- User Share
- Deadline
- Waiting Time

Local Resource Managers

Limitations for Interoperation

- Do not provide a common interface or security framework
- Based on proprietary protocols
- **Non-interoperable computing vertical silos** within a single organization
  - Requires specialized administration skills
  - Increases operational costs
  - Generates over-provisioning and global load unbalance



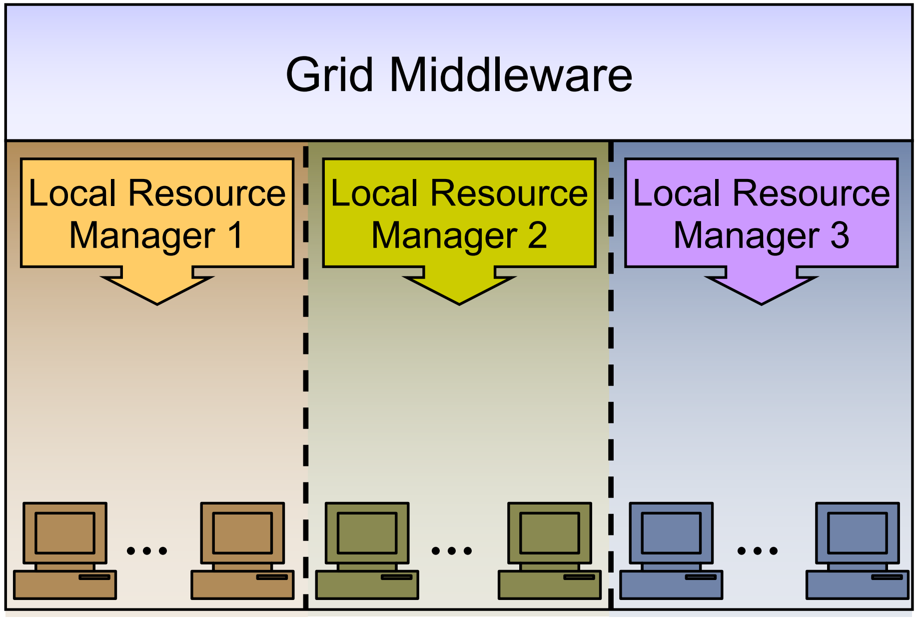
Only a small fraction of the infrastructure is available to the user

Infrastructure is fragmented in non-interoperable computational silos

Grid Computing

Integration of Different Administrative Domains

“A (*computational*) grid offers a common layer to integrate heterogeneous computational platforms (vertical silos) and/or administrative domains by defining a consistent set of abstraction and interfaces for access to, and management of, shared resources”



Common Interface for Each Type of Resources: User can access a wide set of resources.

Types of Resources: Computational, storage and network.

Grid Computing

Grid Infrastructures

- **Grid Services**
  - Security
  - Information & Monitoring
  - Data Management
  - Execution
  - Meta-scheduling
- **Based on Open Source Software**

Open Science Grid

- **> 300 million core hours per year**

egi

1,100,000 cores — High-Throughput Compute

19 cloud providers — Cloud Compute

500 PB — Online Storage

500 PB — Archive Storage

As of March 2020

Grid Computing

Grid is NOT Public Resource Computing

Public Resource Computing

- Volunteer computing
- Master-worker architecture using systems donated by owners to specific projects

https://boinc.berkeley.edu

BOINC computing power

Totals

24-hour average: 31.568 PetaFLOPS.
Active: 77,075 volunteers, 288,298 computers.
Daily change: +12 volunteers, -538 computers.

Next Steps

- Completed mandatory course survey?
  https://forms.gle/3GMsAsHgnbYuUXbdA

- Get ready for **next lecture**
  A.2. Large-scale processing on the cloud

- **Reading Assignments**

I. Sadooghi et al, *"Understanding the Performance and Potential of Cloud Computing for Scientific Applications"*,
IEEE TCC, Issue No. 02 - April-June (2017 vol. 5)

- **HW A** - Out on Monday

Questions

Parallel Processing Architectures

https://piazza.com/harvard/spring2021/cs205/home