Supercomputing Architectures and Development Methodologies

for AI and Big Data Applications

by: Tassadaq Hussain

Professor Department of Electrical Engineering

Namal University Mianwali

Collaborations:

Barcelona Supercomputing Center, Spain

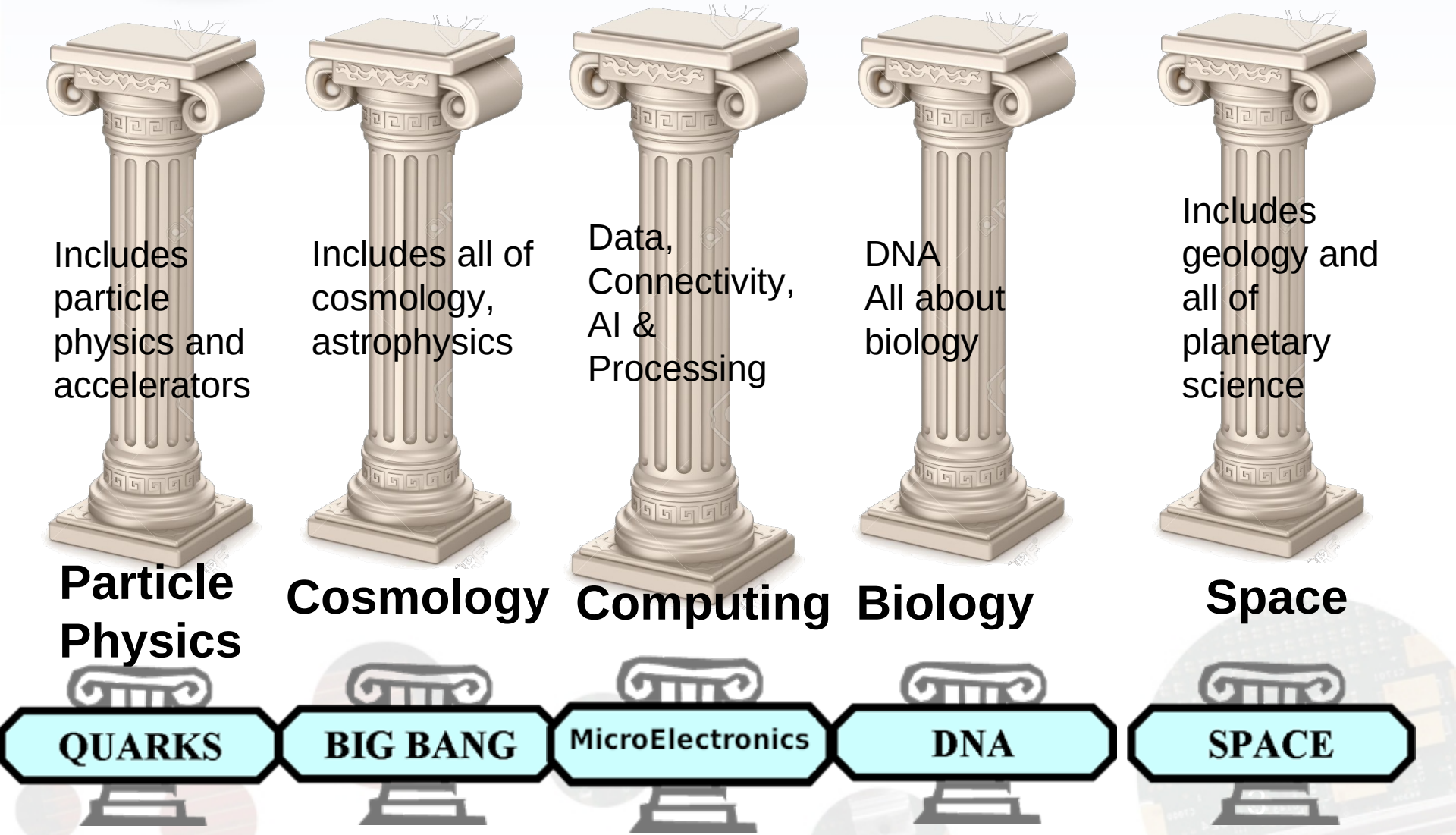
European Network on High Performance and Embedded Architecture and Compilation

Pakistan Supercomputing Center

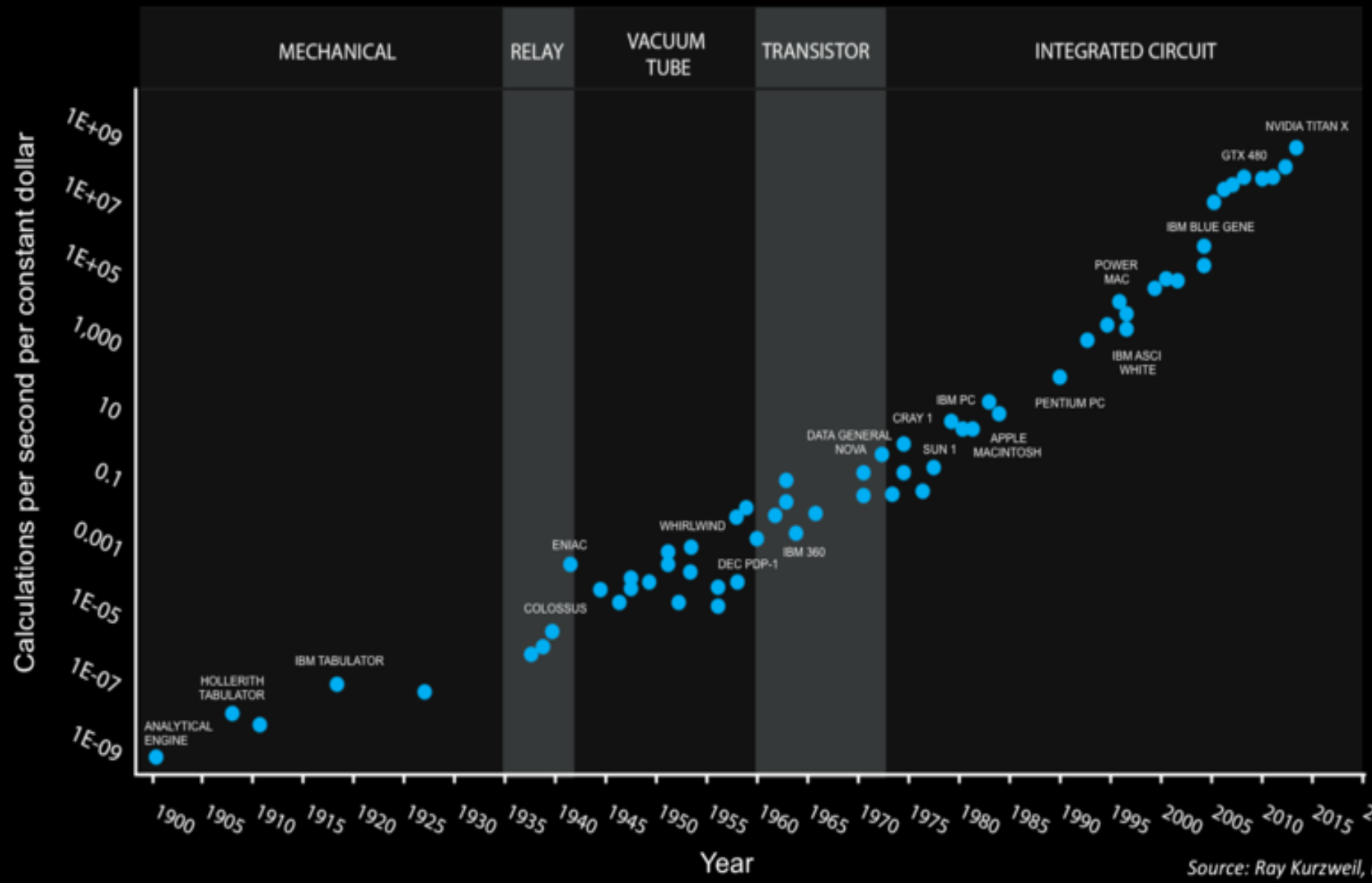
- **Importance of HPC**
- Types of Processing System
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

Pillars of Science

Fermi National Accelerator Laboratory



120 Years of Moore's Law



Source: Ray Kurzweil, DFJ

Life Science

Earth Science

Social Science

Science

175 ZByte @2025

80%
Data-Sciences

Data

100 ExaFLOPS
@2020

87.04 B$

234.6 B$ @2025

AI

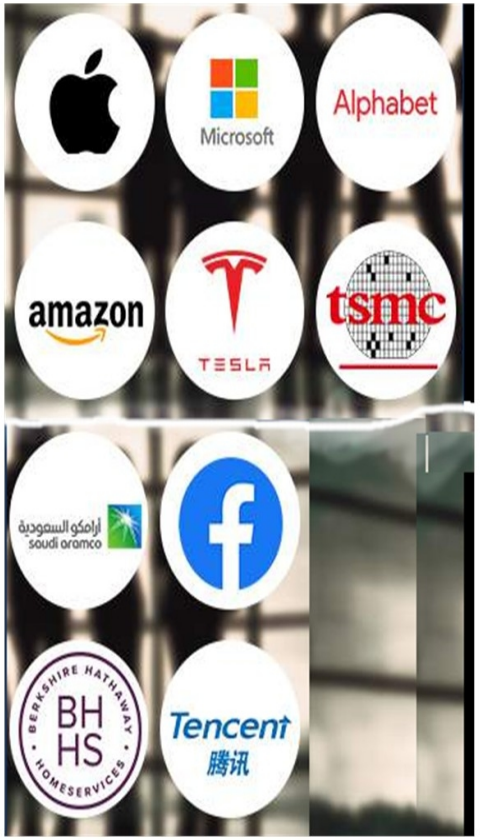
Top500 List
8 PetaFLOPS
@2022

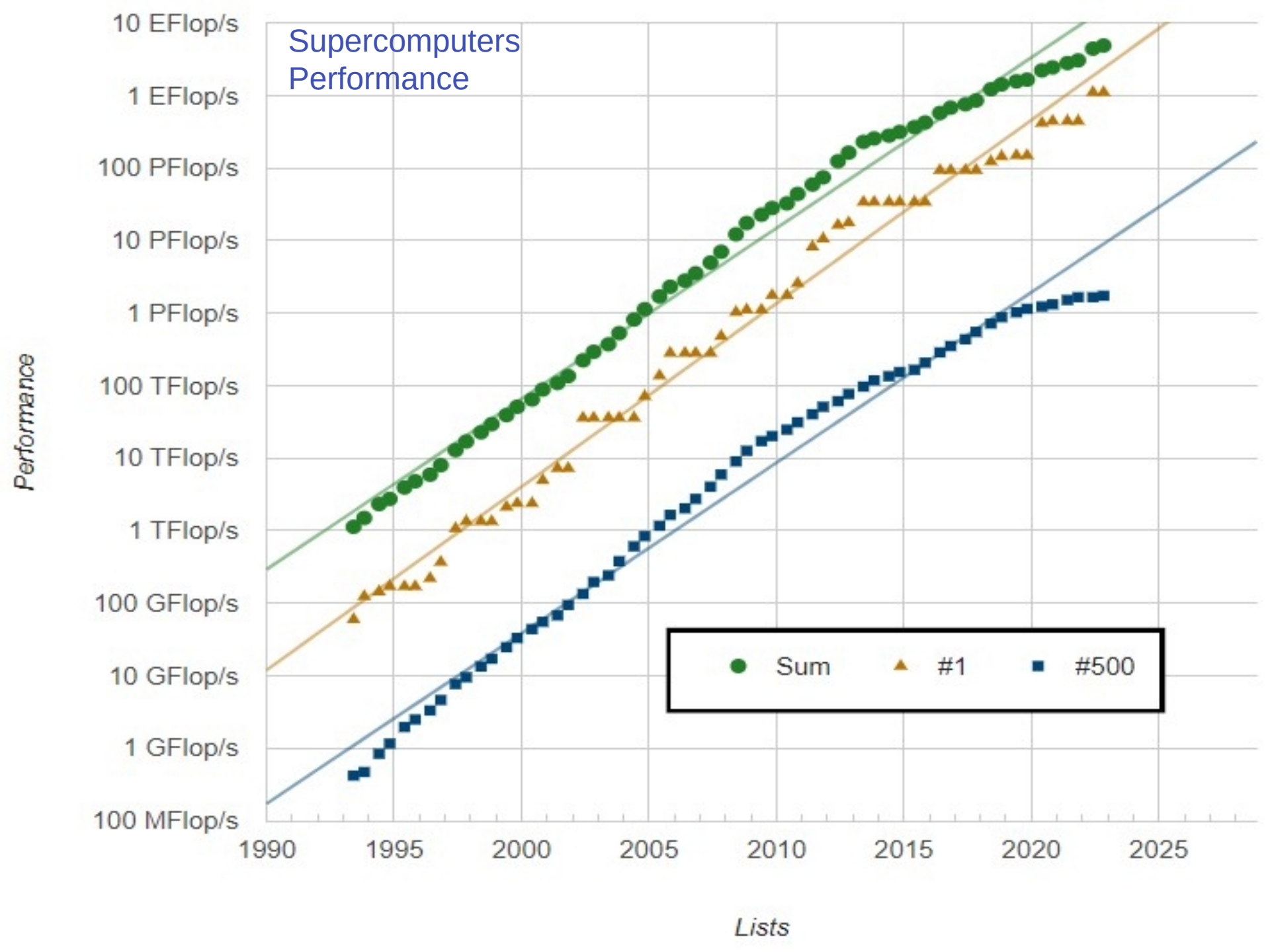
uProcessor
100 B$ @2020

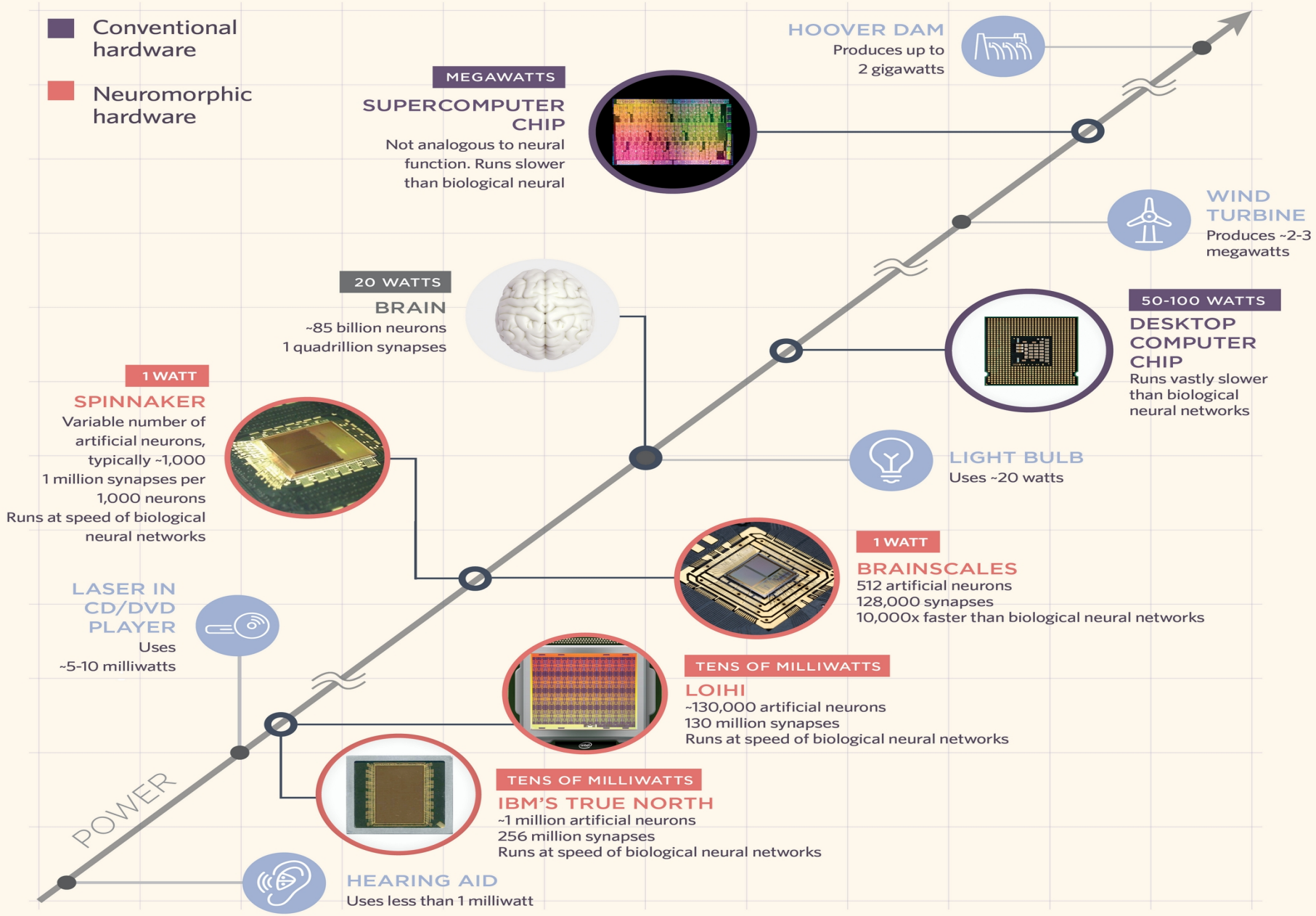
30% Cell Phone
20% Embedded
App
50 Servers, PCs etc.

Computing

Digital Industrial Age
5.5 Trillion $ Revenue@2021




Supercomputers
Performance
10 EFlop/s
1 EFlop/s
100 PFlop/s
10 PFlop/s
1 PFlop/s
100 TFlop/s
10 TFlop/s
1 TFlop/s
100 GFlop/s
10 GFlop/s
1 GFlop/s
100 MFlop/s
Performance
1990
1995
2000
2005
2010
2015
2020
2025
Lists
Sum
#1
#500


Conventional hardware
Neuromorphic hardware
POWER
HOOVER DAM
Produces up to 2 gigawatts
MEGAWATTS
SUPERCOMPUTER CHIP
Not analogous to neural function. Runs slower than biological neural
WIND TURBINE
Produces ~2-3 megawatts
20 WATTS
BRAIN
~85 billion neurons
1 quadrillion synapses
50-100 WATTS
DESKTOP COMPUTER CHIP
Runs vastly slower than biological neural networks
1 WATT
SPINNAKER
Variable number of artificial neurons, typically ~1,000
1 million synapses per 1,000 neurons
Runs at speed of biological neural networks
LIGHT BULB
Uses ~20 watts
1 WATT
BRAINSCALES
512 artificial neurons
128,000 synapses
10,000x faster than biological neural networks
LASER IN CD/DVD PLAYER
Uses ~5-10 milliwatts
TENS OF MILLIWATTS
LOIHI
~130,000 artificial neurons
130 million synapses
Runs at speed of biological neural networks
TENS OF MILLIWATTS
IBM'S TRUE NORTH
~1 million artificial neurons
256 million synapses
Runs at speed of biological neural networks
HEARING AID
Uses less than 1 milliwatt

Hype Cycle for Artificial Intelligence, 2020

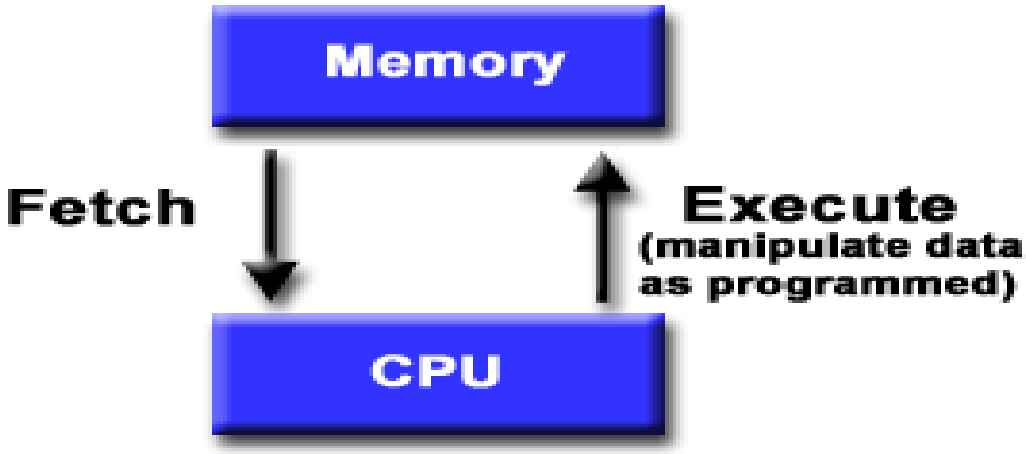
gartner.com/SmarterWithGartner

Gartner®

- Importance of HPC
- **Types of Processing System**
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

Basic Processor Architecture

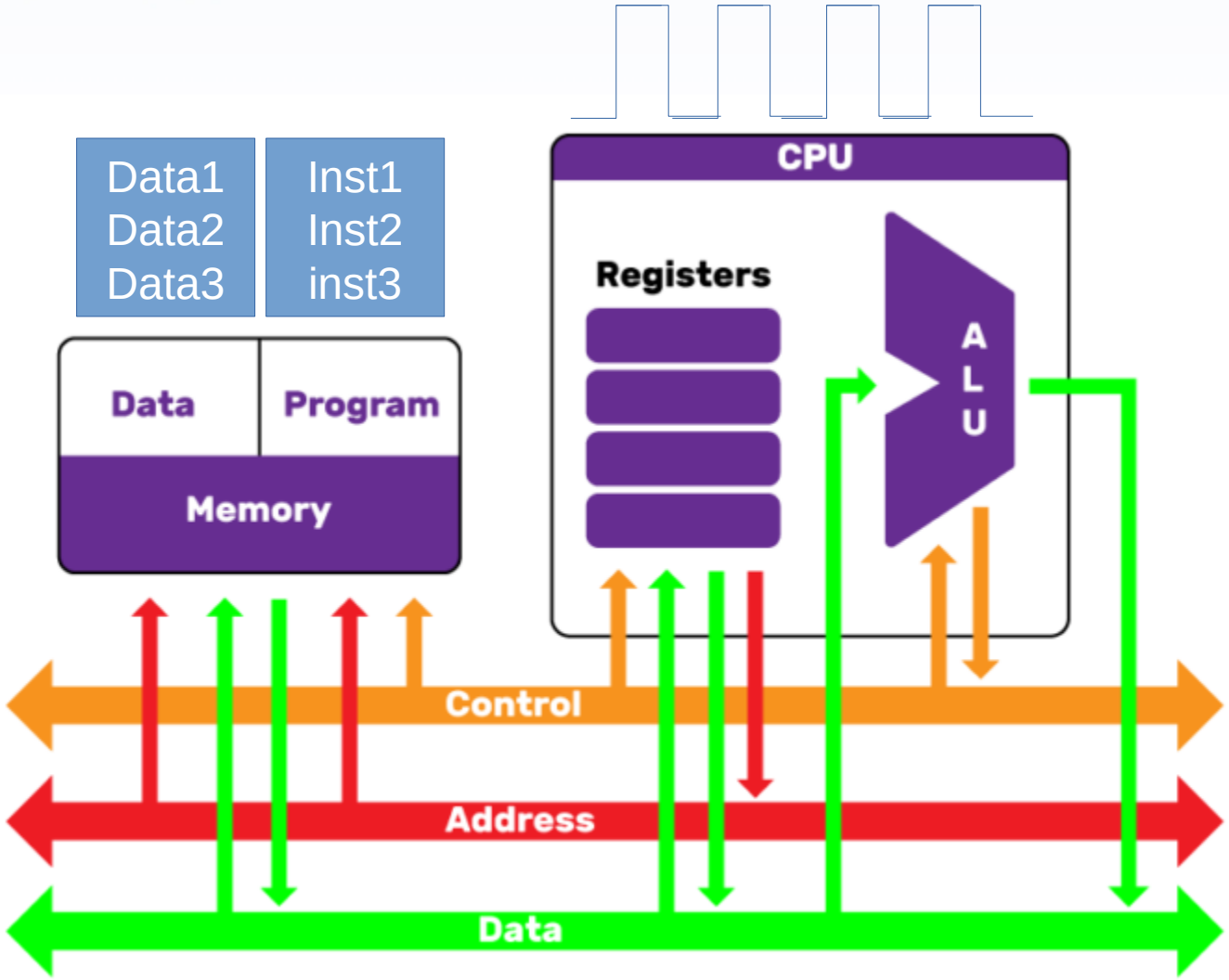
- A central processing unit (CPU) gets instructions and/or data from memory, decodes the instructions and then ***sequentially*** performs them.
- Memory is used to store both program and data instructions
  - Program instructions are coded data which tell the computer to do something
  - Data is simply information to be used by the program



Information and Computer

Permanent Storage Disk Drives

Main Memory



Processors for Supercomputers

Microprocessor development directions:

- Increasing of clock frequency and speed instruction stream processing
- Processing of large collection of data in single processor instruction - SIMD
- Control path multiplication – multi threading

Vector processors

- NEC SX-6
- Cray (Cray X1)

RISC processors

- MIPS
- IBM Power4
- Alpha

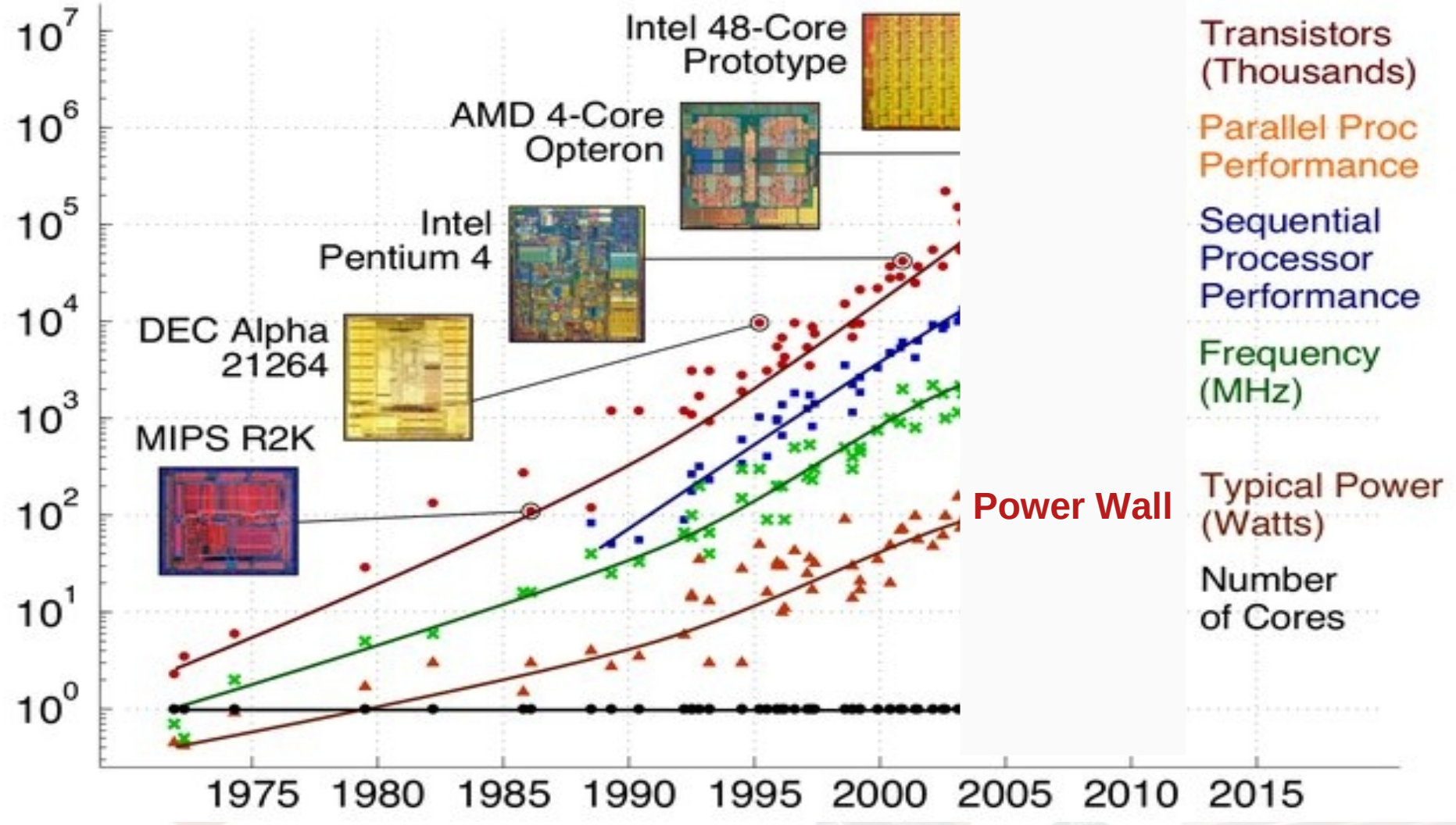
CISC processors

- IA32
- AMD x86-64

VLIW processors

- IA64

Performance: Transistor Count, Frequency, Power and Multi Processor



- Importance of HPC
- Types of Processing System
- **HPC System Architecture (Parallel Processors)**
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

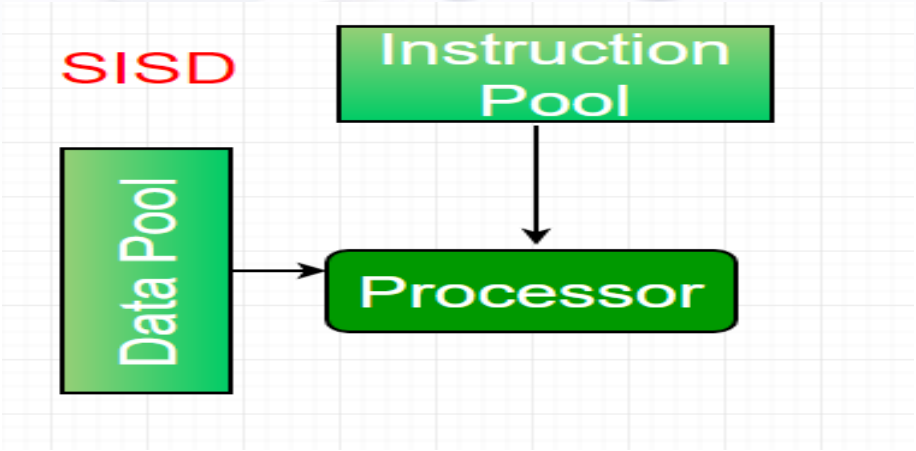
HPC System

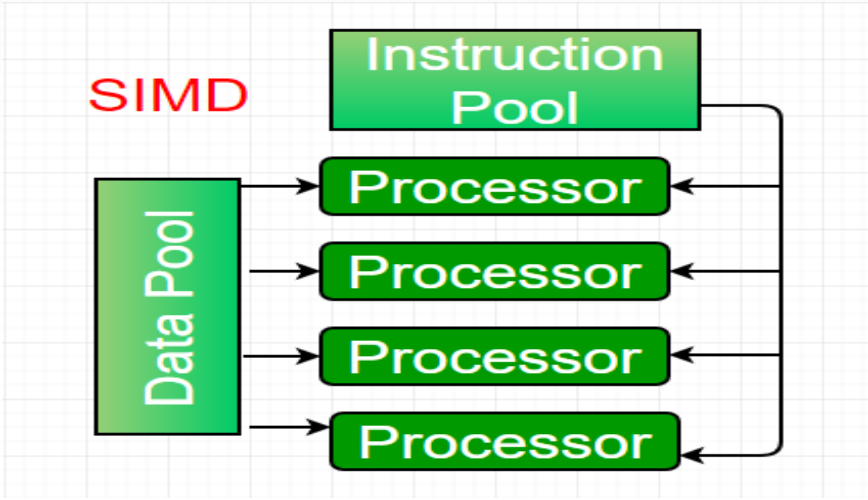
- Muti/Many Processor System Architecture
- Computer Architecture wrt Memory Arrangement
- Shared Storage Solutions
- Network Configuration

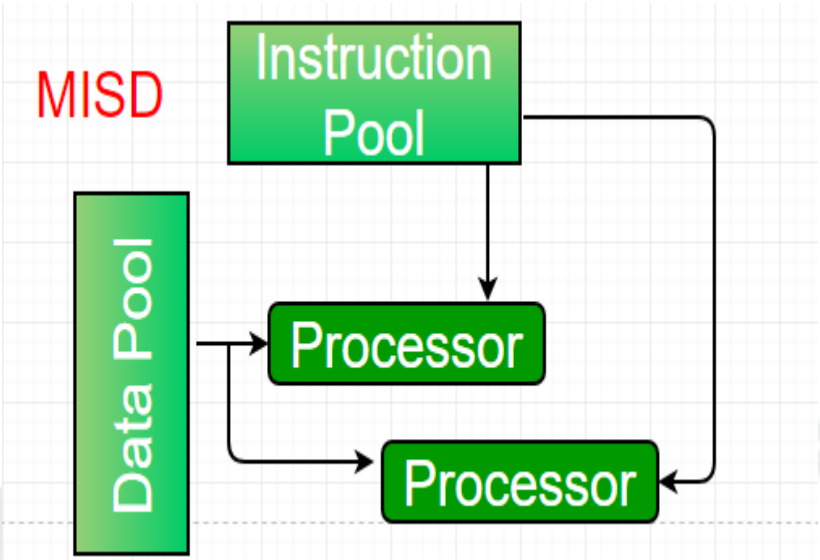
Flynn's Classical Taxonomy

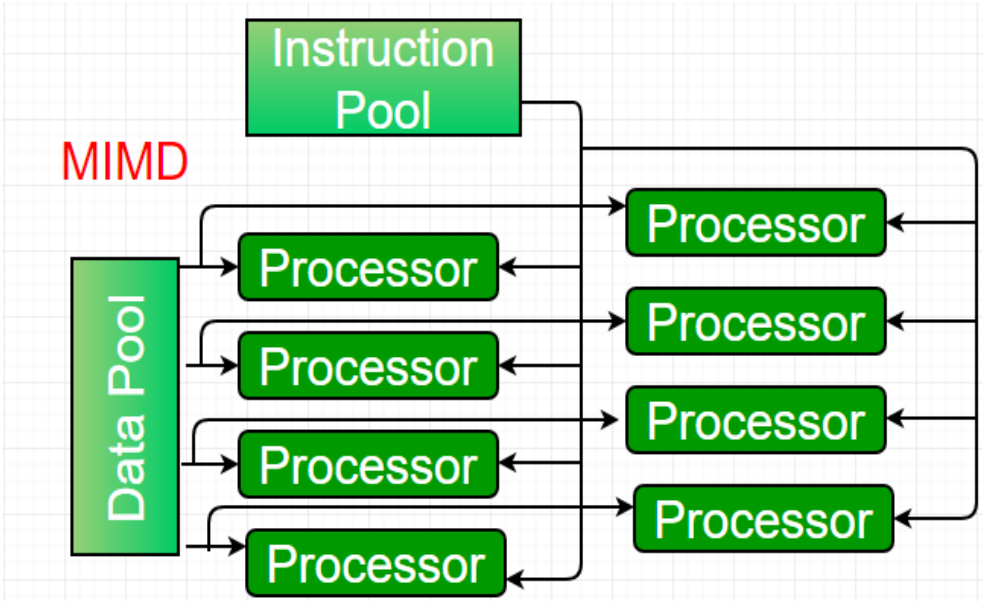
- There are different ways to classify parallel computers. One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.
- Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of ***Instruction*** and ***Data***.
- Each of these dimensions can have only one of two possible states: ***Single*** or ***Multiple***.

Processor Architectures







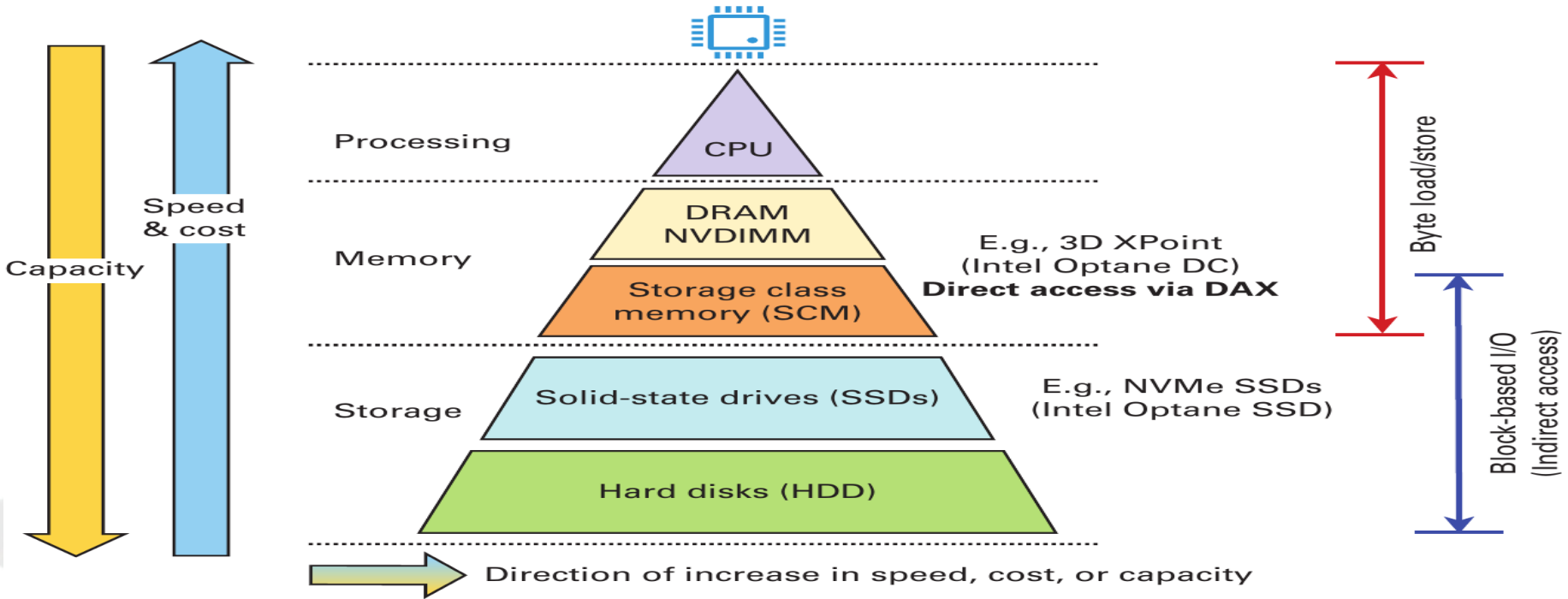


Computer Architecture wrt Memory Arrangement

Shared-memory

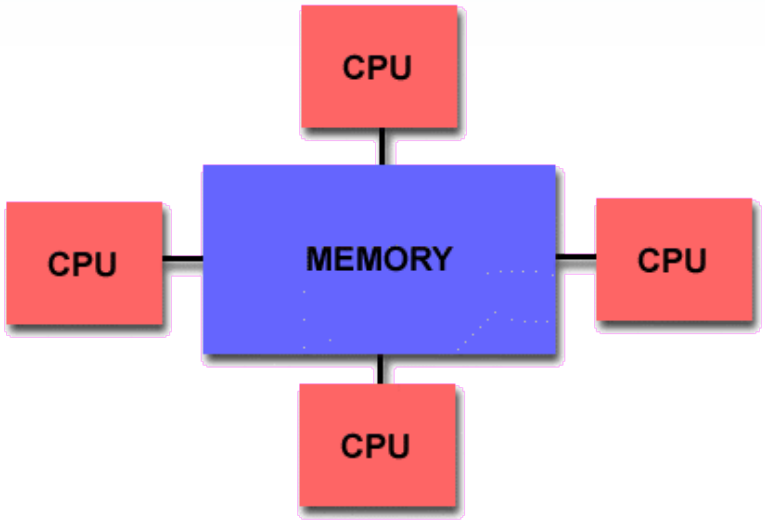
Distributed-memory

Distributed Shared-memory



Shared Memory

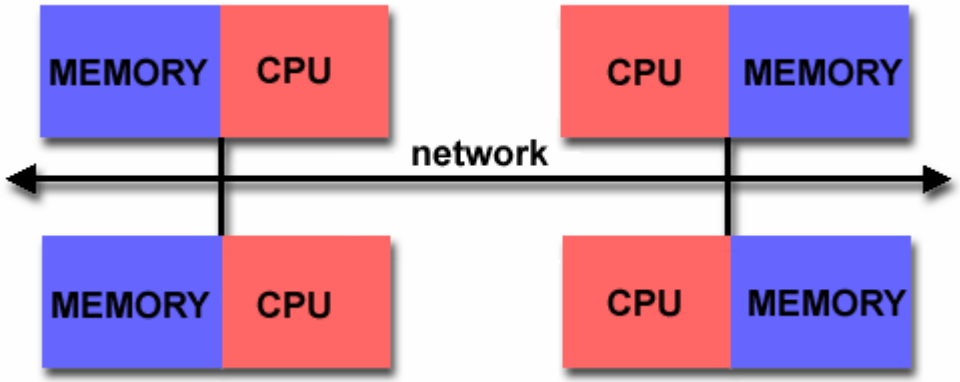
- Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.



- Multiple processors can operate independently but share the same memory resources.
- Changes in a memory location effected by one processor are visible to all other processors.
- Shared memory machines can be divided into two main classes based upon memory access times: ***UMA*** and ***NUMA***.

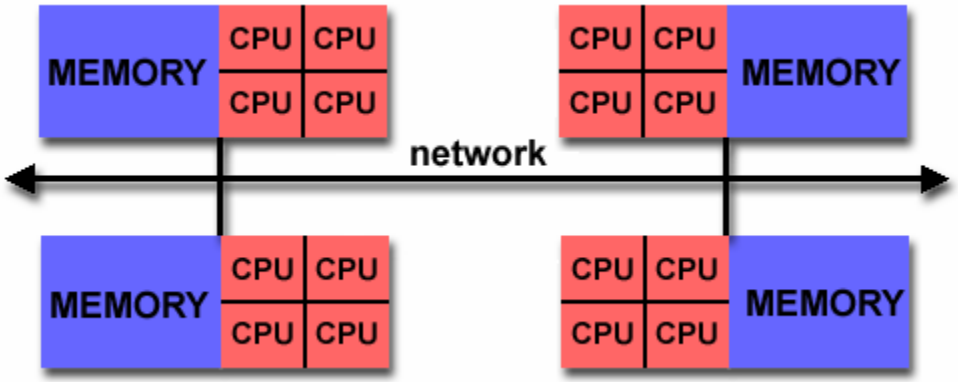
Distributed Memory

- Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory.
- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.
- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.
- The network "fabric" used for data transfer varies widely, though it can can be as simple as Ethernet.



Hybrid Distributed-Shared Memory

- The largest and fastest computers in the world today employ both shared and distributed memory architectures.



- The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global.
- The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.
- Current trends seem to indicate that this type of memory architecture will continue to prevail and increase at the high end of computing for the foreseeable future.
- Advantages and Disadvantages: whatever is common to both shared and distributed memory architectures.

Types: Memory Access

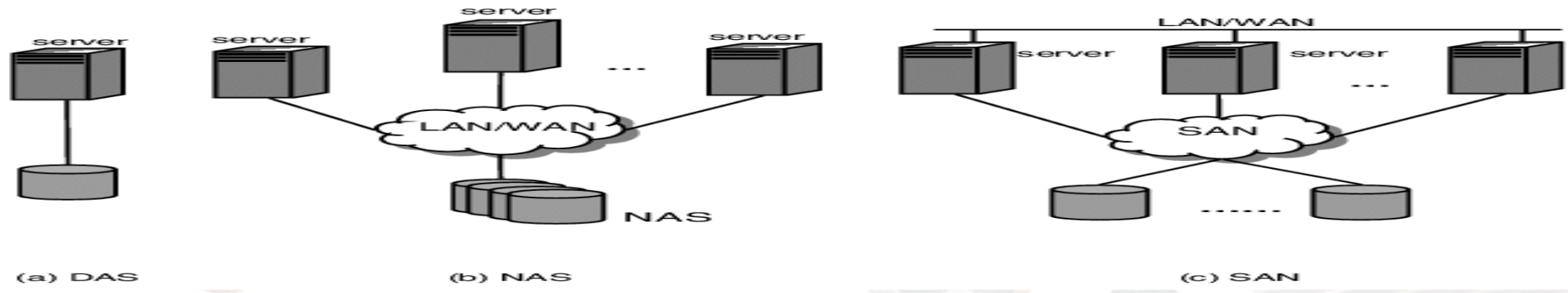
Unified Memory Access (UMA)

Non Unified Memory Access (NUMA)

- Cache-Coherent NUMA (ccNUMA)
- Distributed Memory (DM) / MPI
- In-Memory Computing
- NUMA Hybrid Architectures

Shared Disk Storage

- DAS (Direct Attached Storage)
- NAS (Network Attached Storage)
- SAN (Storage Area Network)
- Parallel File System Storage
- Object Storage:
- Tiered Storage: SSD and HDD
- Ceph : Object, Blocks, File Storage



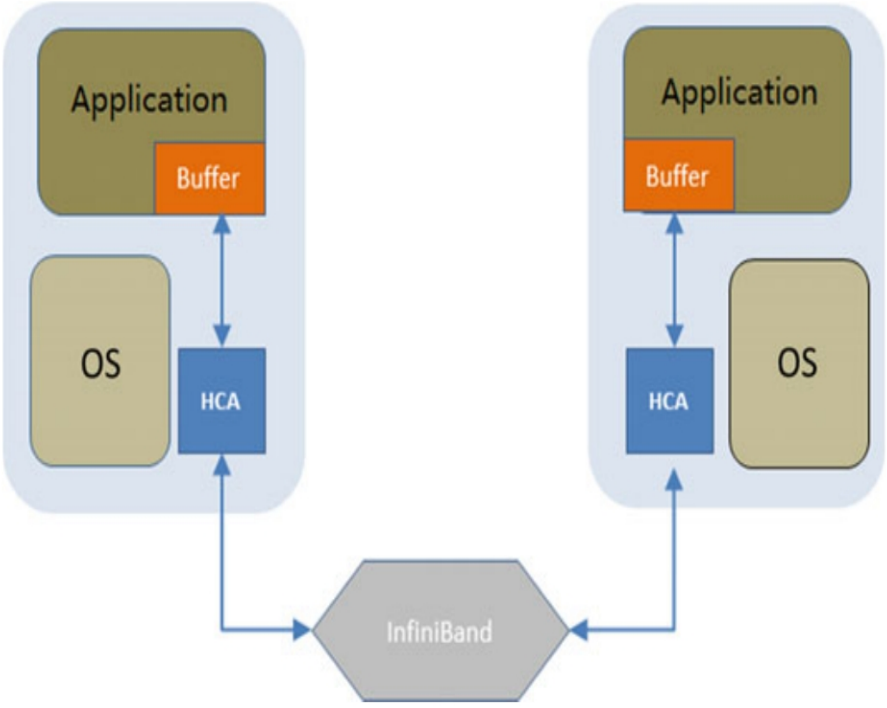
Comparison: SAN, NAS, PFS

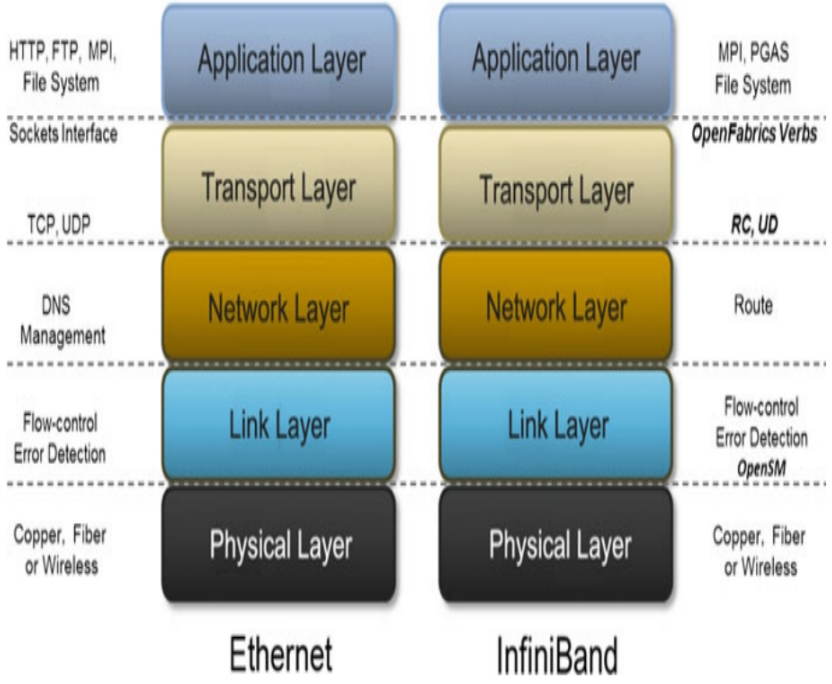
Criteria	Storage Area Network (SAN)	Network Attached Storage (NAS)	Parallel File System
Access Type	Block-level access	File-level access	File-level access with parallel capabilities
Performance	High-performance, low-latency	Varies based on NAS solution	High-performance, scalable
Workload Flexibility	Well-suited for demanding workloads	Suitable for general workloads	Designed for high-performance computing
Scalability	Scalable, can handle growth	Varies based on NAS solution	Highly scalable and optimized for large clusters
Complexity	Can be complex to set up and manage	Relatively simple to set up and manage	Can be complex to optimize for specific workloads
Isolation	Can provide isolated storage access	Shared access across clients	Shared access, optimized for parallelism
Hardware Requirement	Requires dedicated SAN hardware (FC)	NAS devices with Ethernet connectivity	Requires specialized parallel file system software
Cost	Higher cost due to dedicated infrastructure	Generally cost-effective	Costs can vary based on solution and scale
Management	May require specialized expertise	Generally easier to manage	Requires expertise in parallel file system management
Protocols Supported	Typically Fibre Channel, iSCSI, etc.	NFS, SMB, other file protocols	NFS, Lustre, GPFS, BeeGFS, etc.
Use Cases	High-performance applications	General-purpose file sharing	High-performance computing, data-intensive tasks

Network Communication Protocols

InfiniBand is a high-speed networking technology primarily designed for low-latency, high-bandwidth communication within data centers and high-performance computing (HPC) environments.

InfiniBand excels in short to medium-range communication within a data center or between closely located computing clusters.





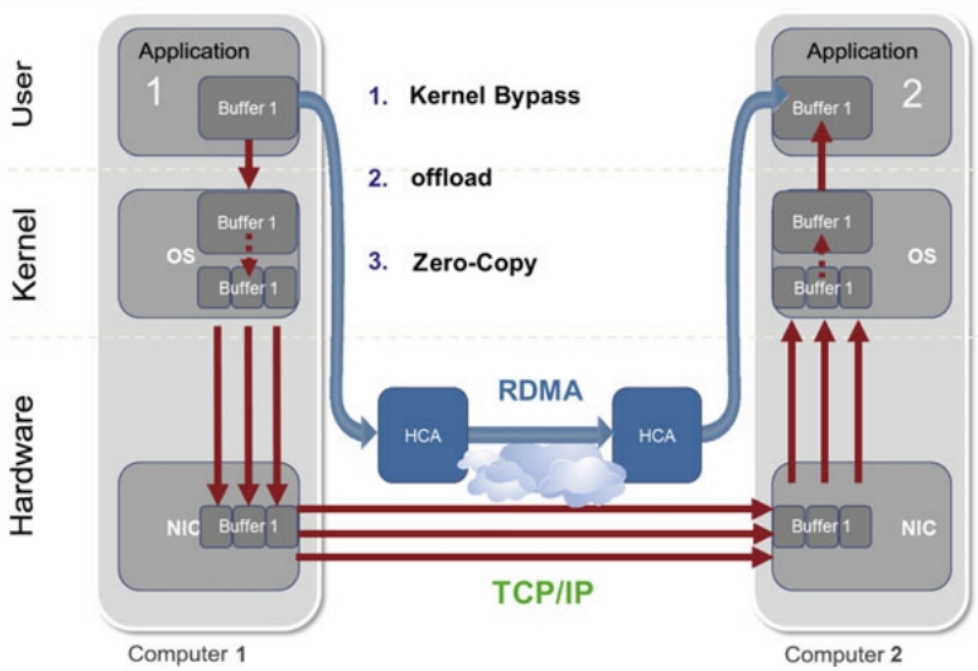
RDMA and HCA

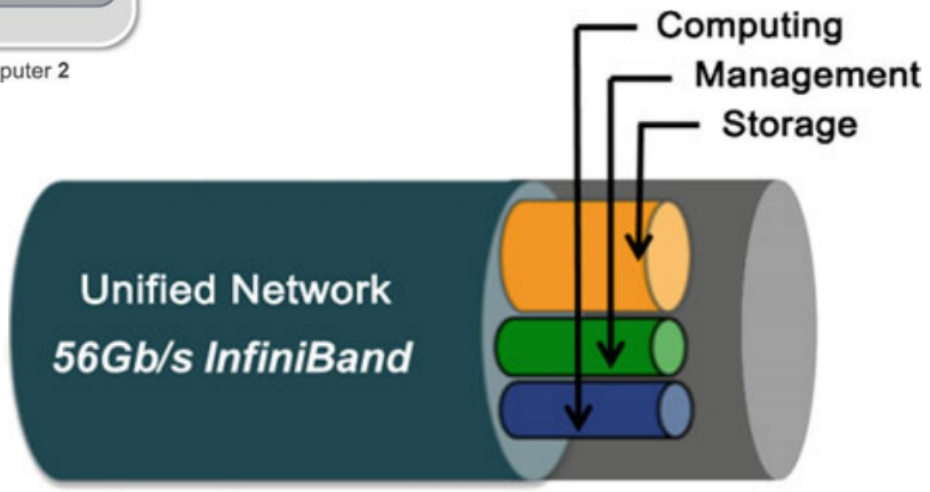
Remote Direct Memory Access (RDMA) and Host Channel Adapter (HCA) allows computers of Cluster to access each other's memory directly without involving the host CPU.

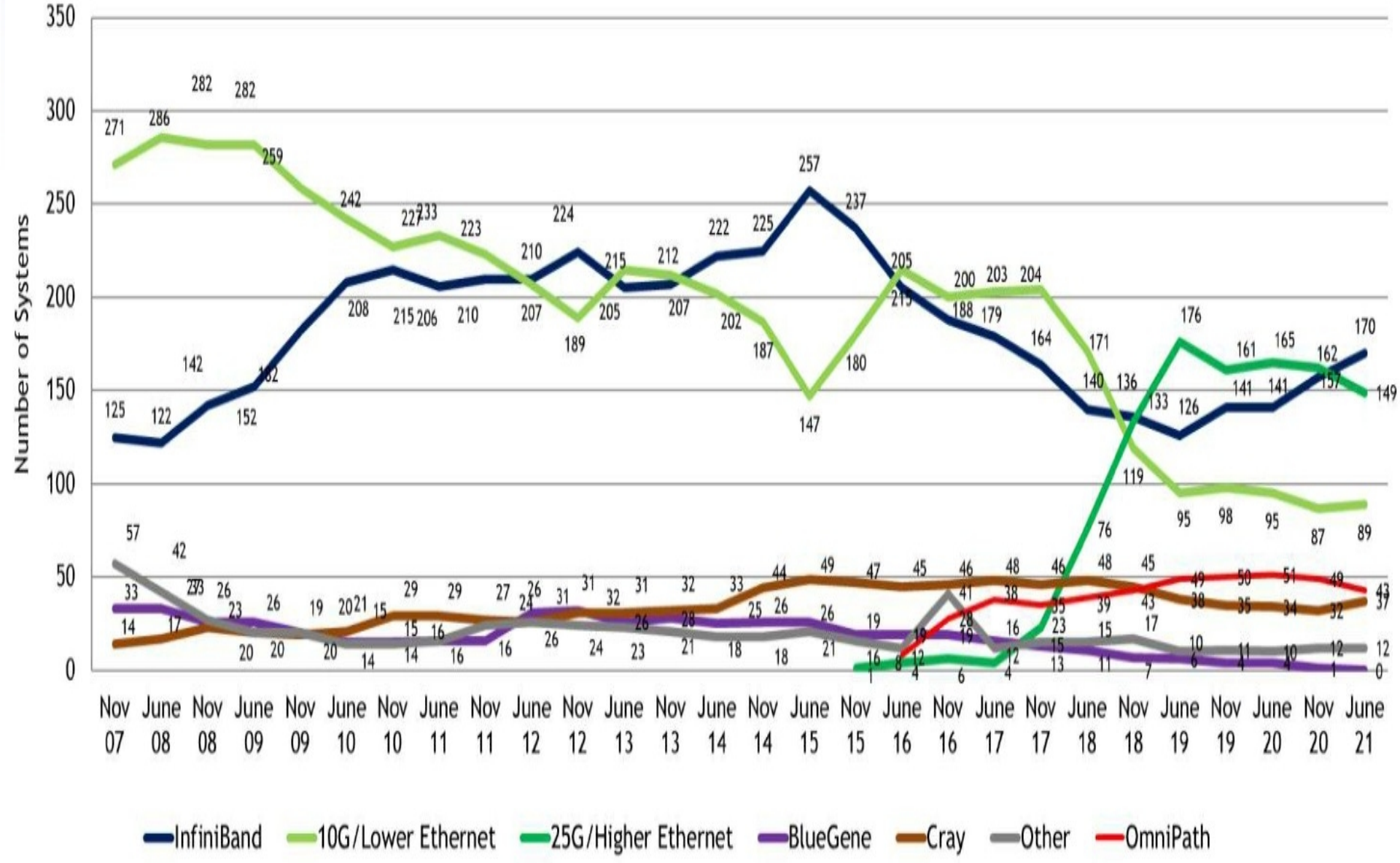
It executes **data offloading** and **data transfer tasks** from the host to dedicated network adapters that support **RDMA**, such as **InfiniBand or RoCE (RDMA over Converged Ethernet)**.

This bypasses the need for the usual network stack processing, resulting in reduced latency and improved data throughput.

TCP/IP and Remote Direct Memory Access

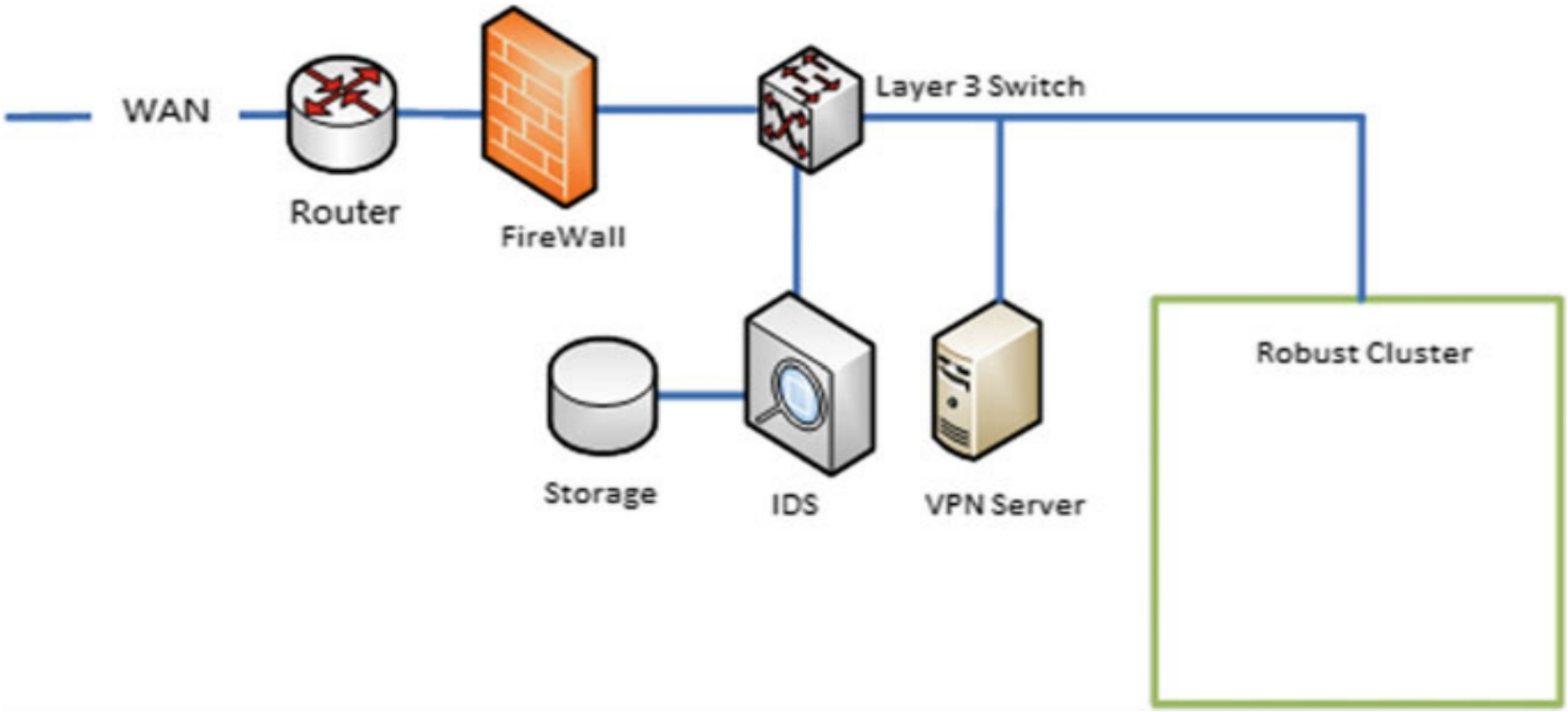





Number of Systems
350
300
250
200
150
100
50
0
Nov 07
June 08
Nov 08
June 09
Nov 09
June 10
Nov 10
June 11
Nov 11
June 12
Nov 12
June 13
Nov 13
June 14
Nov 14
June 15
Nov 15
June 16
Nov 16
June 17
Nov 17
June 18
Nov 18
June 19
Nov 19
June 20
Nov 20
June 21
InfiniBand
10G/Lower Ethernet
25G/Higher Ethernet
BlueGene
Cray
Other
OmniPath

Network Security

Soft Firewall: Zentyal, pfSense, IPFire, SmoothWall, ClearOS, and Iptables.



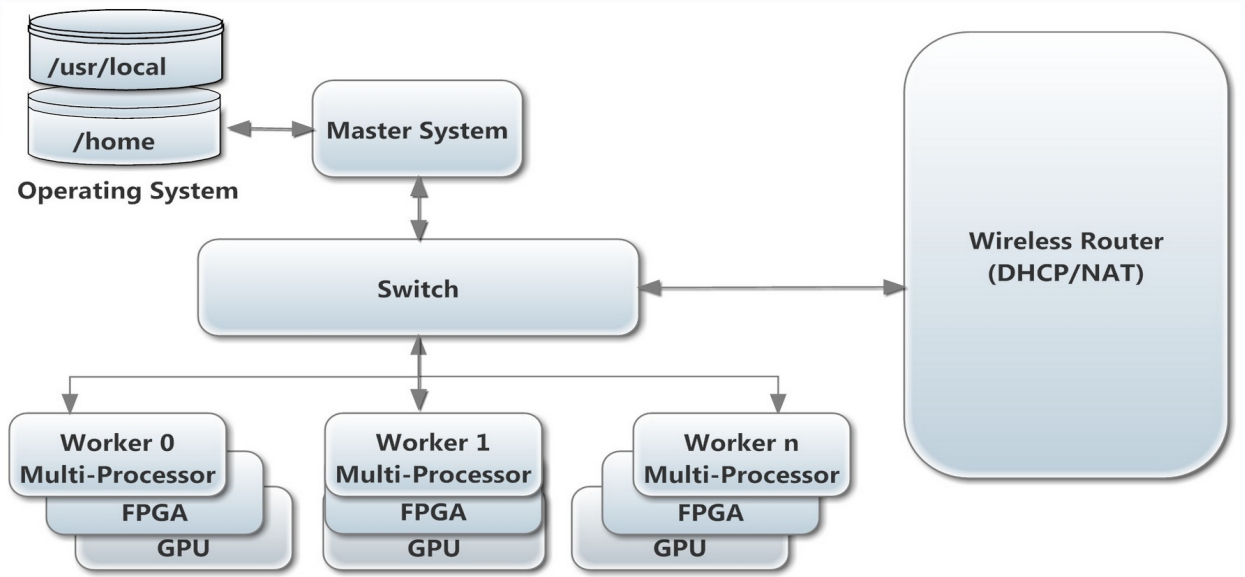
- Importance of HPC
- Types of Processing System
- HPC System Architecture
- **Supercomputing Classes and Infrastructure**
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

What is a Supercomputer?

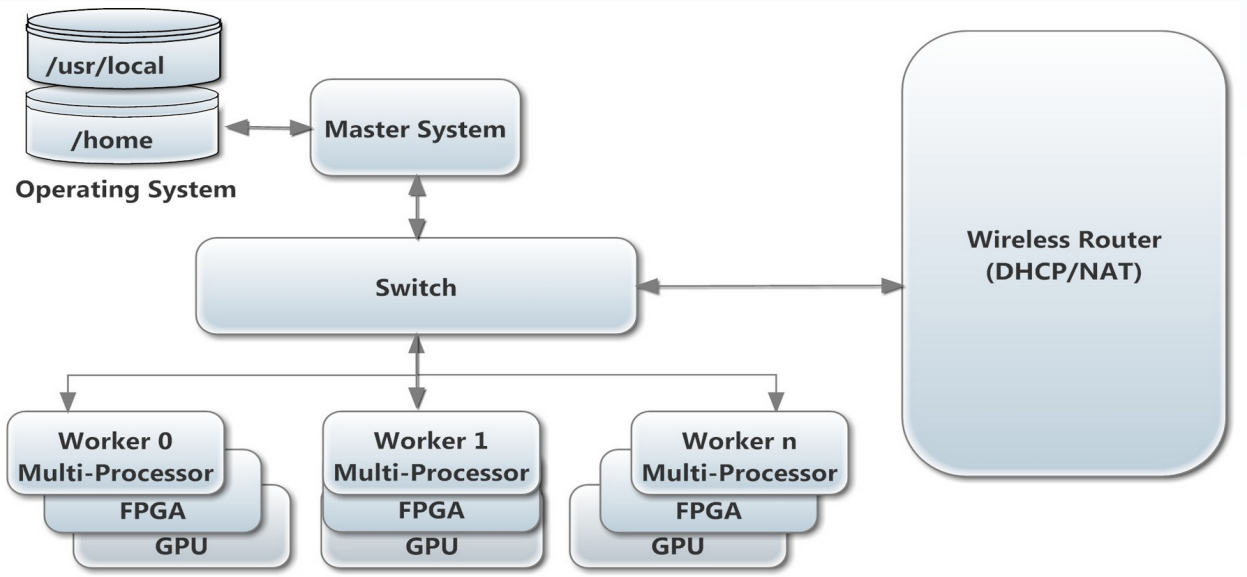
"A supercomputer is a device for turning compute-bound problems into I/O-bound problem" - Seymour Cray

A supercomputer is a computer system that leads the world in terms of processing capacity, particularly speed of calculations, at the time of its introduction. (Wikipedia)

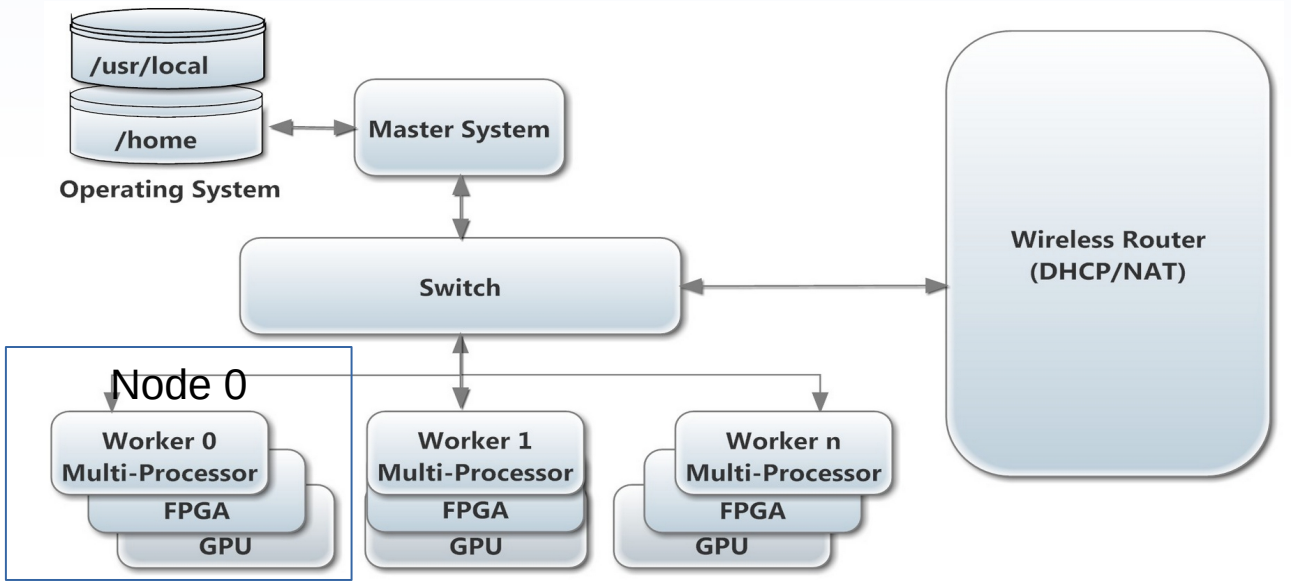
Basic Arch of Supercomputer



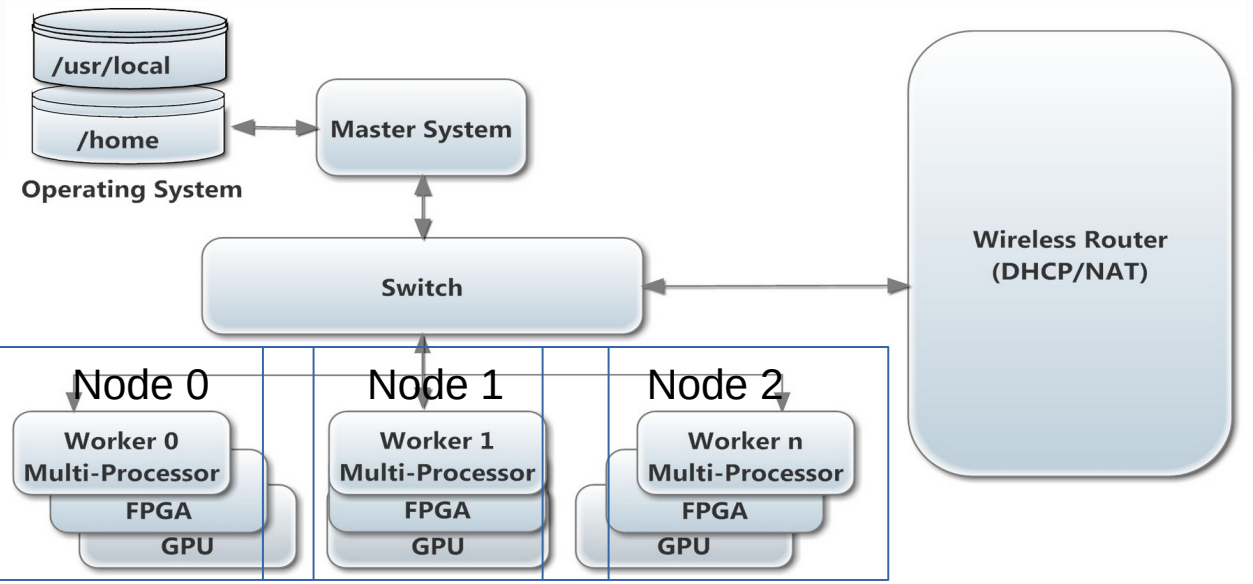
Basic Arch of Supercomputing

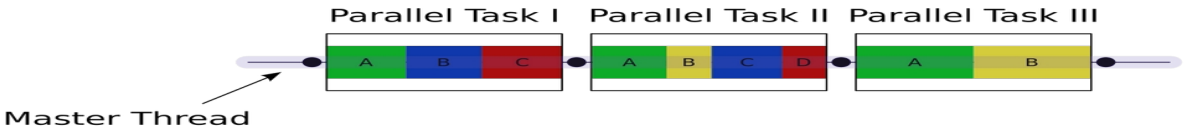


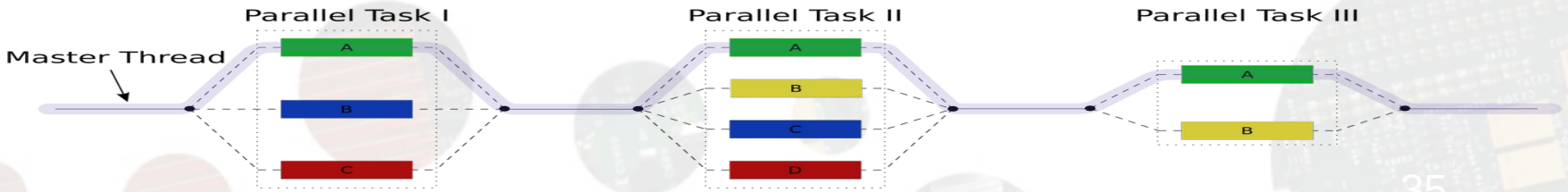
Basic Arch of Supercomputing



Basic Introduction of Supercomputing

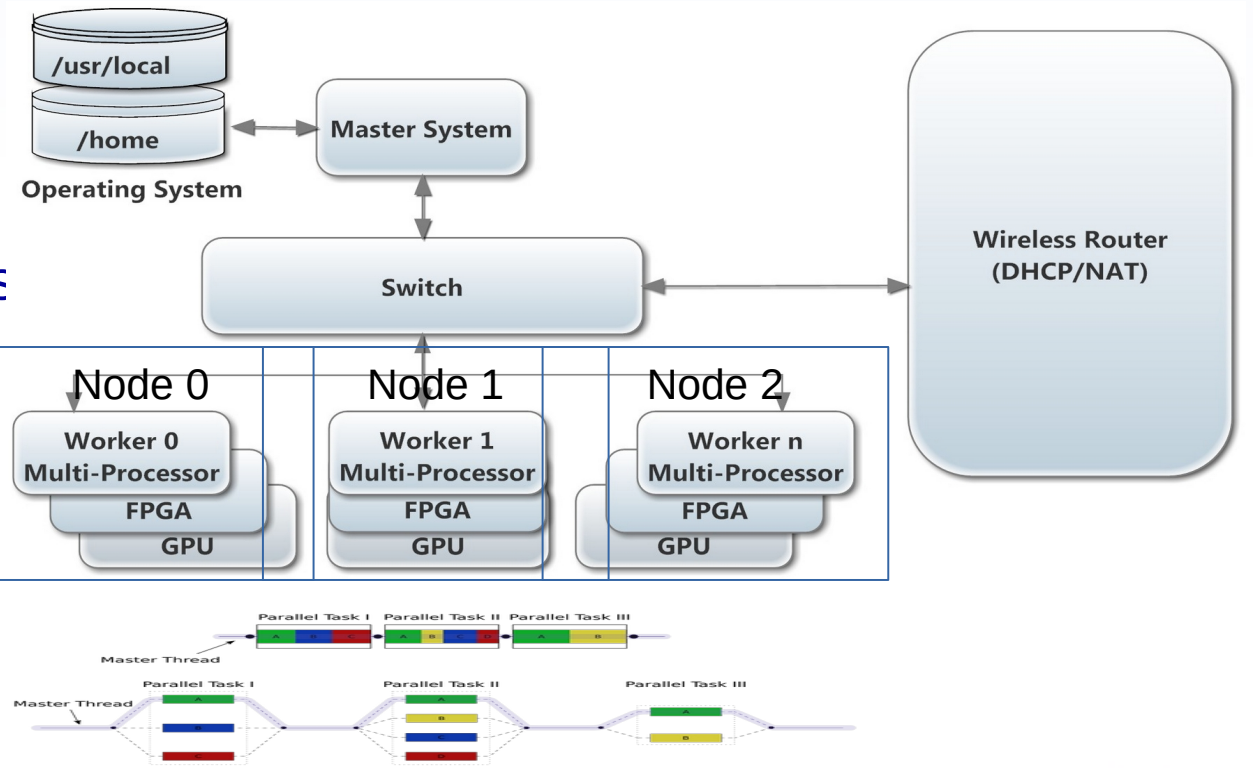






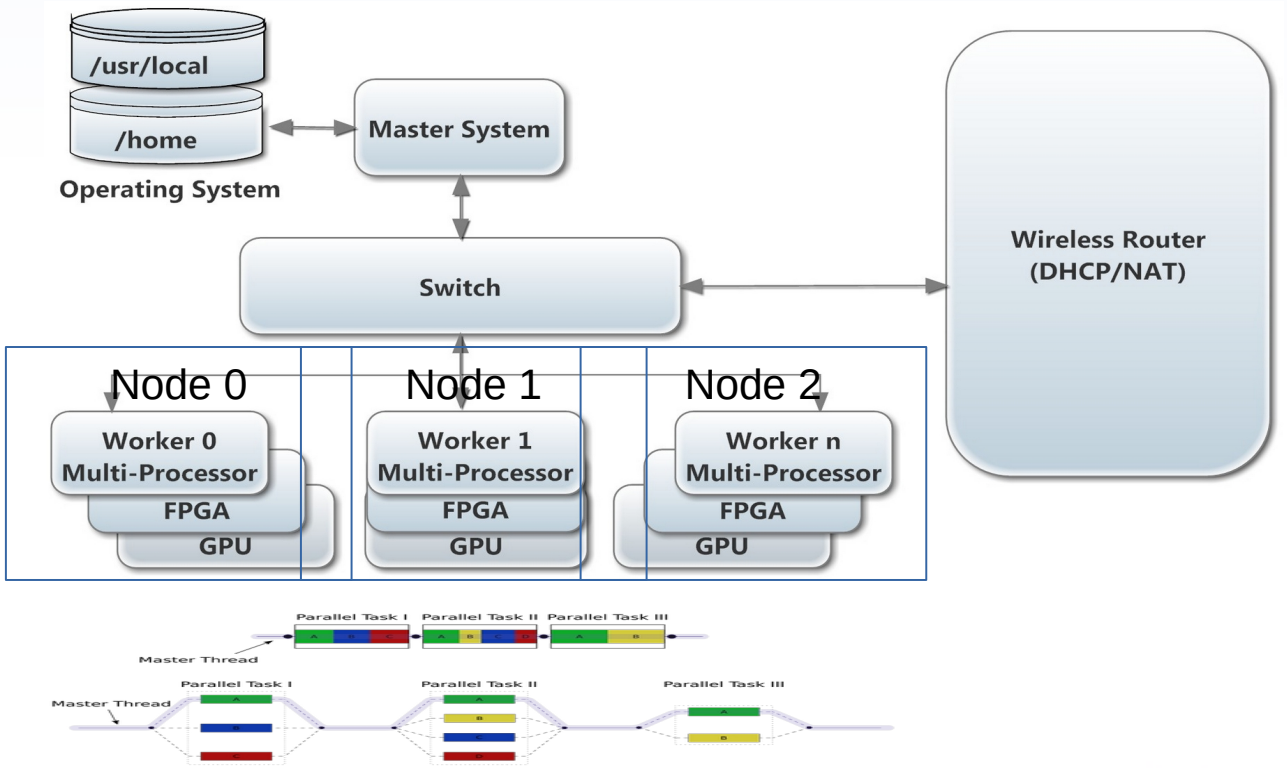
Constituents

- Processing
- Software
- Programming Models
- Storage
- Network
- Accessibility
- Power



Basic Introduction of Supercomputing

- Performance
- Programmability
- Portability
- Scalability
- Accessibility
- Usability
- Power
- Cost



FLOPS = **Fl**oating **P**oint **O**peration **P**er **S**econd
KFLOPS = 10^3
= One Thousand Computation Per Second
= 12.00 * 1212.222 * 21212 + 232323 .....

MFLOPS= 10^6 Million Computation Per Second
GFLOPS= 10^9 Billion
TFLOPS= 10^{12} Trillion
PFLOPS= 10^{15} Quadrillion
EFLOPS= 10^{18} Quintillion
ZFLOPS = 10^{21} Sextillion

History of Supercomputers

1945-50 - Manchester Mark I
1950-55 - MIT Whirlwind
1955-60 - IBM 7090 - **210 KFLOPS**
1960-65 - CDC 6600 -**10.24 MFLOPS**
1965-70 - CDC 7600 - **32.27 MFLOPS**
1970-75 - CDC Cyber **76 MFLOPS**
1975-80 - Cray-1 - **160 MFLOPS**
1980-85 - Cray X-MP - **500 MFLOPS**
1985-90 - Cray Y-MP - **1.3 GFLOPS**
1990-95 - Fujitsu Numerical Wind Tunnel - **236 GFLOPS**
1995-00 - Intel ASCI Red - **2.150 TFLOPS**
2000-02 - IBM ASCI White, SP Power3 375 MHz - **7.226 TFLOPS**
2002-03 - NEC Earth Simulator - **35 TFLOPS**

Supercomputing Generations

Gen I: 1970s. SIMD Array of Processors.

- USAs developed ILLIAC-IV processor array, consisting of an 8 x 8 array of 64 processors. 106 MFLOPS

Gen II: Cray-1 Pipelined Vector Machines 1976, 12 pipelined arithmetic units, 160 MFLOPS

Gen III: Shared-Memory multi-processor systems (MIMD Arch) 40 TFLOPS

Gen IV: Masive Parallel Processor having 10 to thousands of processors

Gen V: Cluster based (CPU+GPU+MIC)

Classes of Supercomputers

General Purpose

vector processing machines - the same operation carried out on a large amount of data simultaneously

Tightly connected cluster computers (NUMA) - communication oriented architectures engineered from ground up, based on high speed interconnects and large number of processors

Commodity clusters - collection of large number of commodity PCs (COTS) interconnected by high-bandwidth low-latency network

Special Purpose: high performance computing devices with a hardware architecture dedicated to solve a single problem (equipped with custom ASICS or FPGA chips)

Commodity Cluster

A cluster is a collection of connected, independent computers working in unison to solve a problem COTS technology nodes are interconnected by Ethernet LAN, Myrinet, QsNet ELAN etc. computation can be performed by using popular programming toolkits and frameworks: OpenMP, MPI clusters require dedicated management software.

Components of a Supercomputing System

- Compute Nodes
- Login/Access Nodes
- I/O Nodes
- Network Nodes (Switches/Routers)
- Management Nodes
- Storage Nodes
- Service Nodes

Compute Nodes:

These nodes perform the actual computational tasks and simulations.

Equipped with powerful CPUs and GPUs for parallel processing.

The number of compute nodes can vary based on the expected workload and desired level of parallelism.

Login/Access Nodes:

Provide remote access for users to connect to the cluster.

Users log in to these nodes to submit jobs, manage files, and interact with the cluster.

Not meant for computation but facilitate user interaction.

Management Nodes:

Host cluster management software, schedulers, and monitoring tools.

Manage resource allocation, job scheduling, system health monitoring, and overall cluster administration.

Storage Nodes:

Manage the storage infrastructure of the cluster.

Provide access to high-performance storage systems, such as parallel file systems or distributed storage solutions.

Store large datasets and simulation results.crucial for parallel processing and data sharing.

I/O Nodes:

Handle data input/output operations efficiently.

Optimize data access and transfer between compute nodes and storage systems.

Reduce I/O bottlenecks and latency for data-intensive applications.

Network Nodes (Switches/Routers):

High-speed interconnects (e.g., InfiniBand, Ethernet with RDMA) enable fast communication between compute nodes.

Low-latency, high-bandwidth communication is

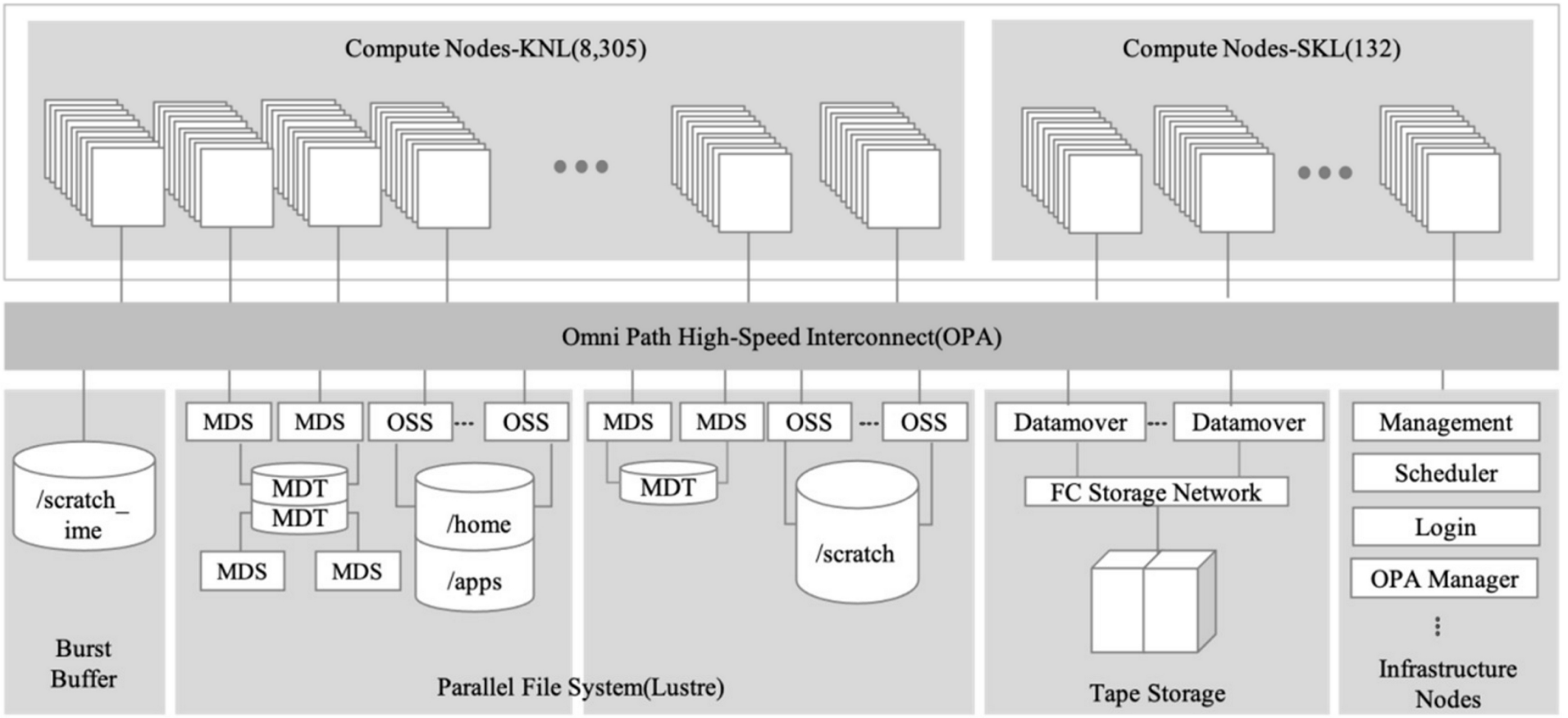
Service Nodes:

Host essential services for cluster operation and user interaction.

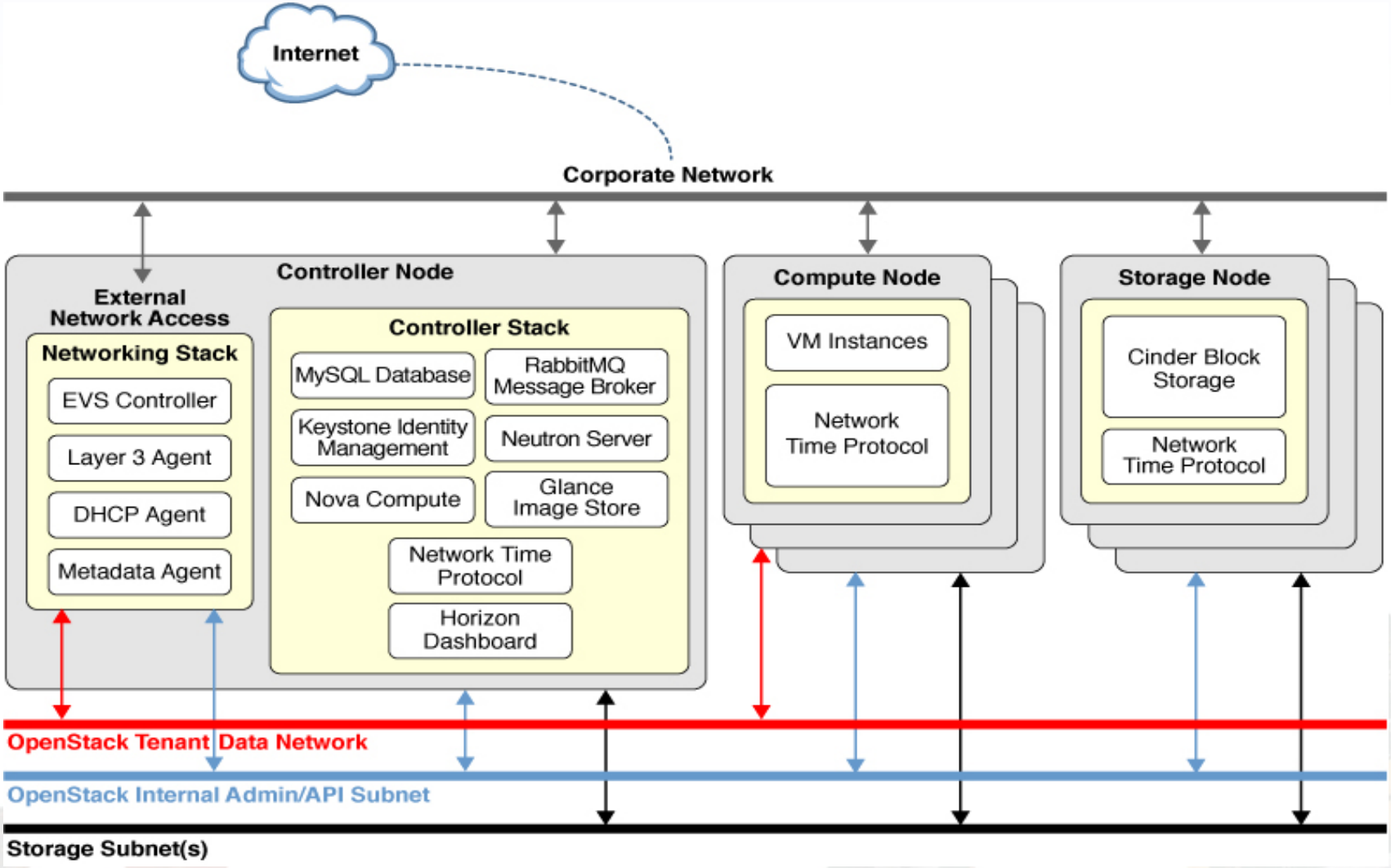
Services may include Light Weight Directory Access Protocol authentication, DNS, NTP (Network Time Protocol), and other services needed for system stability.

Compute Node Interconnect:

High-performance network interconnects between compute nodes facilitate communication for parallel processing and distributed computing.


Compute Nodes-KNL(8,305)
Compute Nodes-SKL(132)
Omni Path High-Speed Interconnect(OPA)
/scratch_
ime
Burst
Buffer
MDS
MDS
OSS
OSS
MDT
MDT
MDS
MDS
/home
/apps
MDS
MDS
OSS
OSS
MDT
/scratch
Parallel File System(Lustre)
Datamover
Datamover
FC Storage Network
Tape Storage
Management
Scheduler
Login
OPA Manager
Infrastructure
Nodes

OpenStack


Internet
Corporate Network
Controller Node
External Network Access
Networking Stack
EVS Controller
Layer 3 Agent
DHCP Agent
Metadata Agent
Controller Stack
MySQL Database
RabbitMQ Message Broker
Keystone Identity Management
Neutron Server
Nova Compute
Glance Image Store
Network Time Protocol
Horizon Dashboard
Compute Node
VM Instances
Network Time Protocol
Storage Node
Cinder Block Storage
Network Time Protocol
OpenStack Tenant Data Network
OpenStack Internal Admin/API Subnet
Storage Subnet(s)

Performance Measurement

2 login node and 52 compute nodes, each of them:

2 x IBM Power9 8335-GTH @ 2.4GHz (3.0GHz on turbo, 20 cores and 4 threads/core, total 160 threads per node)

- **160 x 2.4 G FLOPS = 384 GFLOPS**

512GB of main memory distributed in 16 dimms x 32GB @ 2666MHz

2 x SSD 1.9TB as local storage and 2 x 3.2TB NVME

4 x GPU NVIDIA V100 (Volta) with 16GB HBM2.

- **4 x 14.13 TFLOPS = 56.52**

GPFS via one fiber link 10 Gbit

Supercomputing System Performance = (56.52 + .384) x 52

2.959 PFLOPS

Performance Bottleneck

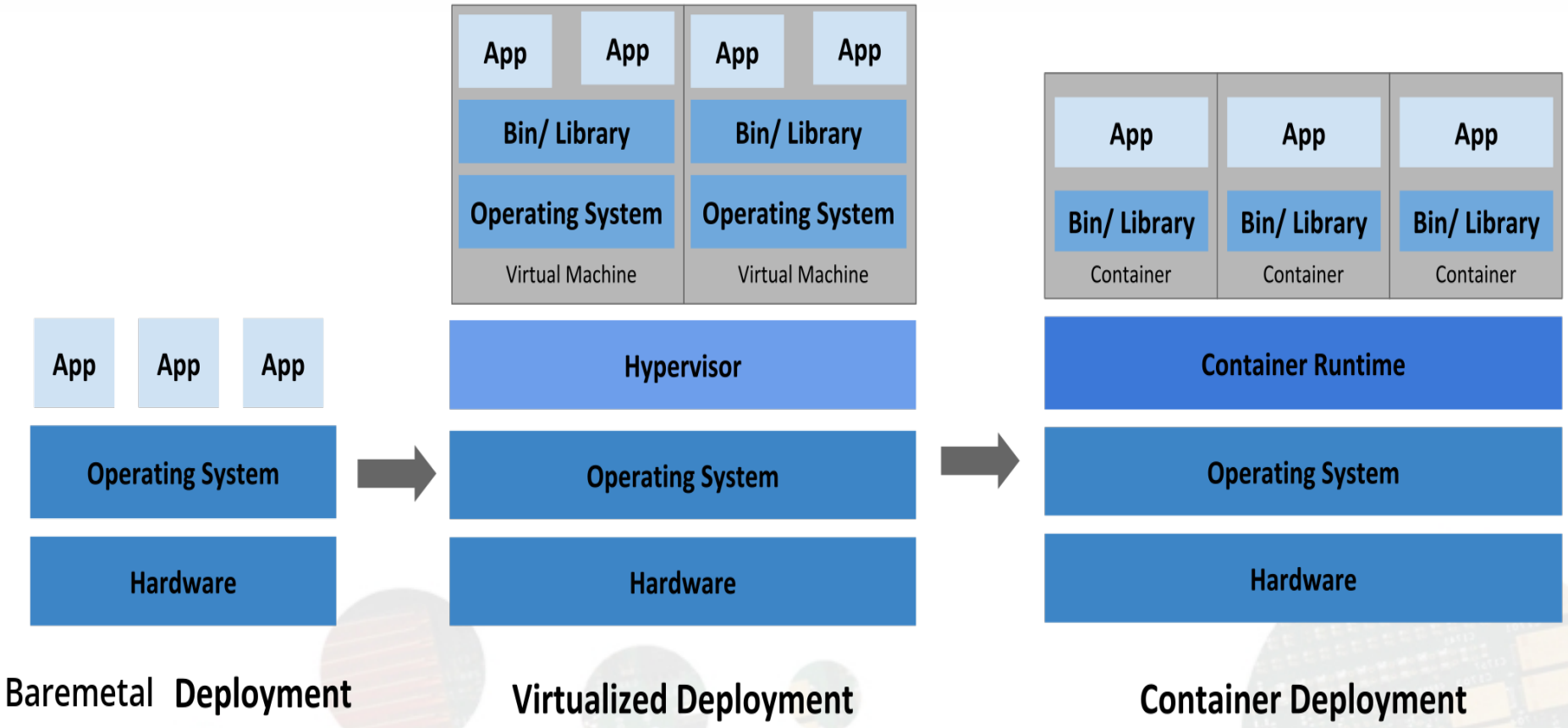
Main Memory Delay

Network (I/O) Delay

Disk Storage Delay

- Importance of HPC
- Types of Processing System
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- **Cluster Software Stacks**
- Programming Models
- Applications

Software Stack Structures and Usage



Cluster System Software Stack Configuration Tools

Linux Distributions

- Red Hat Enterprise Linux (RHEL) and CentOS
- SUSE Linux Enterprise Server (SLES)
- Ubuntu Server

HPC-Optimized Linux Distributions

- CentOS Stream for HPC
- OpenHPC

Specialized Linux Variants:

- Rocky Linux
- Scientific Linux

Environment Modules System

- Singularity containers in an HPC environment

Cluster Management Tools

- Automate the deployment and management of nodes

High-Performance Computing (HPC) Bare Metal

Slurm: A popular open-source workload manager and job scheduler for managing and optimizing HPC clusters.

Distributed computation environments are used for software development (OpenMP, MPI). OpenMPI: A widely used message-passing interface for developing parallel and distributed computing applications.

Intel oneAPI: A unified software stack from Intel for developing high-performance applications that can leverage CPU, GPU, and FPGA architectures.

GNU Compiler Collection (GCC): A set of compilers and libraries often used for building high-performance applications in various programming languages.

CUDA Toolkit: NVIDIA's software platform for developing applications that leverage NVIDIA GPUs for high-performance computing.

High-Performance Math Function Library: BLAS (Basic Linear Algebra Subprograms), LINPACK, LAPACK (Linear Algebra PACKage) Linux

Kubernetes

Docker: A platform for developing, shipping, and running applications using containerization.

Kubernetes: An open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications.

Helm: A package manager for Kubernetes that simplifies the deployment and management of applications.

Prometheus: An open-source monitoring and alerting toolkit designed for cloud-native environments.

Fluentd: An open-source data collector for unifying log management across different services.

OpenStack Platform

Nova: The compute service in OpenStack, responsible for managing virtual machines and instances.

Neutron: The networking service that provides networking resources and connectivity for OpenStack deployments.

Cinder: The block storage service that enables the creation and management of block storage volumes.

Swift: The object storage service designed to store and retrieve large amounts of unstructured data.

Horizon: The web-based dashboard interface for managing and monitoring OpenStack resources.

Heat: The orchestration service that allows users to define and automate the deployment of resources using templates.

HPC , AI and BigData Software Stack

Deep Learning and BigData Environment	**Frameworks**	Caffe, Caffe2, Caffe-MPI, Chainer, Microsoft CNTK, Keras, MXNet, Tenasorflow, Theano, PyTorch Apache Hadoop, Apache Spark, Apache Flink, Apache Storm, Apache				
	Libraries	cnDNN, NCCL, cuBLAS, Apache Hive, Apache Pig, Apache HBase, Spark SQL, MLlib, GraphX				
	User Access	**NVIDIA DIGITS, Apache Zeppelin**				
Programming Environment	**Development & Performance Tools**	Intel Parallel Studio XS Cluster Edition	PGI Cluster Development Kit		GNU Toolchain	NVIDIA CUDA
	Scientific and Communication Libraries	Intel MPI	MVAPICH2, MVAPICH		IBM Spectrum LSF	Open MPI
	Debuggers	Intel IDB	PGI PGDBG		GNU GDB	
Schedulers, File Systems and Management	**Resource Managment/Job Scheduling**	Adaptive Computative Moab, Maui TORQUE	SLURM	Altair PBS Professional	IBM Spectrum LSF	Grid Engine
	File Systems	Lustre	NFS	GPFS	Local (ext3, ext4, XFS)	
	Cluster Managment	Beowulf, xCat, OpenHPC, Rocks, Bright Cluster Manager for HPC including support for NVIDIA Data Center GPU Manager				
Operating Systems and Drivers	**Drivers & Network Mgmt.**	Accelerator Software Stack and Drivers		OFED, OPA		
	Operating Systems	Linux (RHEL, CentOS, SUSE Enterprise, Ubuntu, etc.)				

		Supercomputing(SC)	Deep Learning(DL)	Big Data Computing (BDC)
Apps...	Boundary Interaction Services	**In_house, commercial & OSS applications** [e.g. Paraview], **Remote desktop** [e.g. Virtual Network Computing (VNC)], **Secure Sockets Layer** [e.g. SSL certificates]	**Framework_dependent applications** [e.g. NLP, voice, image], **Web mechanisms**[e.g. Google & Amazon Web Services], **Secure Sockets Layer** [e.g. SSL certificates]	**Framework_dependent applications** [e.g. 2/3/4-D], **Secure Sockets Layer** [e.g. SSL certificates]
Middleware & MGMT	Processing Services	**Domain Specific frameworks** [e.g. PETSc], **Batch processing of large tightly coordinated parallel jobs** [100s - 10000s of processes communicating frequently with each other]	**DNN training & interference frameworks** [e.g. Caffe, Tensorflow, Theano, Neon, Torch], **DNN numerical libraries** [e.g. dense LA]	**Machine Learning** (traditional) [e.g. Mahout, Scikit-learn, BigDL], **Analytics/ Statistics** [e.g. Python, ROOT, R, Matlab, SAS, SPSS, Sci-Py], **Iterative** [e.g. Apache Hama], **Interactive** [e.g. Dremel, Drill, Tez, Impala, Shark, Presto, BlinkDB, Spark], **Batch / Map Reduce** [e.g. MapReduce, YARN, Sqoop, Spark], **Real-time / streaming** [e.g. Flink, YARN, Druid, Pinot, Storm, Samza, Spark]
	Model / Infromation Management Services	**Data Storage**: Parallel File Systems [e.g. Lustre, GPFS, BeeGFS, PanFS, PVFS], **I/O libraries** [e.g. HDF5, PnetCDF, ADIOS]	**Data Storage** [e.g. HDFS, Hbase, Amazon S3, GlusterFS, Cassandra, MongoBD, Hana, Vora]	**Serialization** [e.g. Avro], **Meta data** [e.g. HCatalog], **Data Ingestion & Integration** [e.g. Flume, Sqoop, Apache Nifi, Elastic Logstash, Kafka, Talend, Pentaho], **Data Storage** [e.g. HDFS, Hbase, Amazon S3, GlusterFS, Cassandra, MongoBD, Hana, Vora], **Cluster Mgmt** [e.g. YARN, MESO]
	Communication Services	**Messaging & Coordination** [e.g. MPI/PGAS, direct fabric access], **Threading** [e.g. OpenMP, task-based models]	**Messaging & Coordination** [e.g. Machine Learning Scaling library(MLSL)]	**Messaging** [e.g. Apache Kafka (streaming)]
	Workflow / Task Services	**Conventional compiled languages** [e.g. C/C++/Fortran], **Scripting languages** [e.g. Python, Julia]	**Scripting languages** [e.g. Python]	**Workflow & Scheduling** [e.g. Oozie], **Scripting languages** [e.g. Leras, Mocha, Pig, JAQL, Python, Java, Scala
System SW	System Management & Security Services	**Domain numerical libraries** [e.g. PETSc, ScalLAPACK, BLAS, FFTW,...], **Performance & debugging** [e.g. DDT, Vampire], **Accelerator APIs** [e.g. CUDA, OpenCL, OpenACC] **Data Protection** [e.g. System SSS, OS/PFS file access control] **Batch scheduling** [e.g. SLURM], **Cluster management** [e.g. OpenHPC], **Container Virtualization** [e.g. Docker], **Operating System** [e.g. Linux OS Variant]	**Batching for training** [built into DL frameworks], **Reduced precision** [e.g. interference engines], **Load distribution layer** [e.g. Round robin/load balancing for interference], **Accelerator APIs** [e.g. CUDA, OpenCL], **Hardware Optimization Libraries** [e.g. cuDNN, MKL-DNN, etc.] **Virtualisation** [e.g Dockers, Kubernetes, VMware, Xen, KVM, HyperX], **Operating System** [e.g. Linux (RedHat, Ubuntu, etc.), Windows]	**Distributed Coordination** [e.g. ZooKeeper, Chubby, Paxos], **Provisioning, Managing & Monitoring** [e.g. Ambari, Whirr, BigTop, Chukwa], **SVM systems** [e.g. Google Sofia, libSVM, svm-py,...], **Hardware Optimization Libraries** [e.g. DAAL, DPDK, MKL, etc.] **Vitualization** [e.g. Dockers, Kubernetes, VPware, Xen, KVM, HyperX], **Operating System**[e.g. Linux (PedHat, Ubuntu, etc.), Windows]
Hardware	Infrastructure	**Local storage** [e.g. Storage & I/O nodes, NAS] **Servers**[e.g. CPU & Memory [Gen Purpose CPU nodes, GPUs, FPGAs] **Network** [e.g. Infiniband & OPA fabrics]	**Local storage** [e.g. Local storage or NAS/SAN] **Services** [e.g. CPU & Memory [Gen Purpose CPU + GPU/FPGA, TPU] **Natwork** [e.g. Ethernet]	**Local Storage** [e.g. Direct attached Storage] **Services** [e.g. CPU & Memory, [Gen Purpose CPU hyper-convergent nodes] **Natwork** [e.g. Ethernet fabrics]

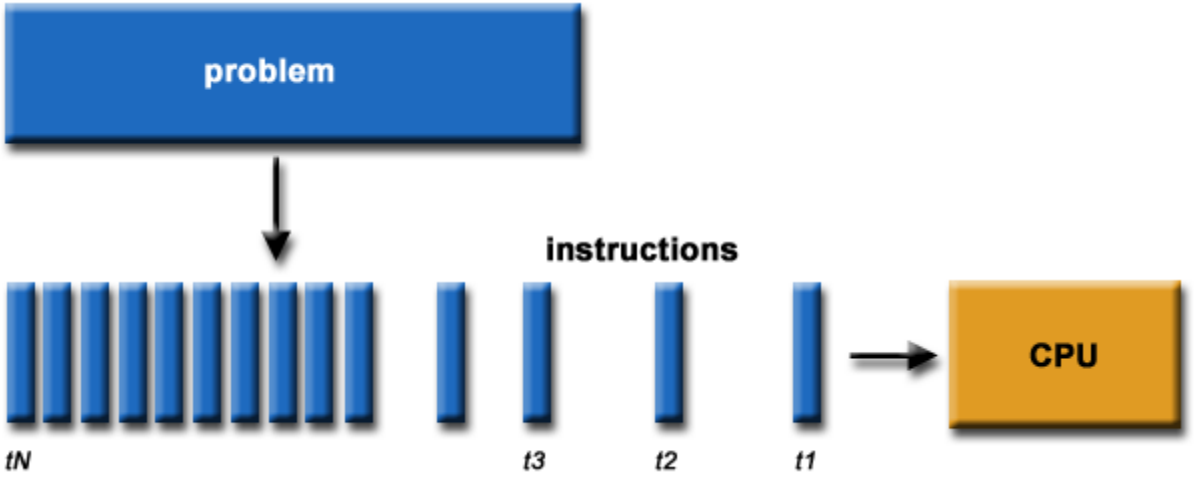
BigData Framework: Infiniband Support

- Big data Hadoop (inherent support for RDMA);
- Oracle, IBM DB2, Microsoft SQL server database;
- Block storage (support RDMA storage protocol iSER and SRP);
- Parallel file system (GPFS, Lustre, Ceph, etc.); and
- Enterprise applications, such as finance, futures, and animations.
- High-Performance Computing (HPC) Clusters:
- Data Analytics and Big Data Platforms:
- AI and ML

- Importance of HPC
- Types of Processing System
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- **Parallel Programming Models**
- Demonstration (Supercomputing System)

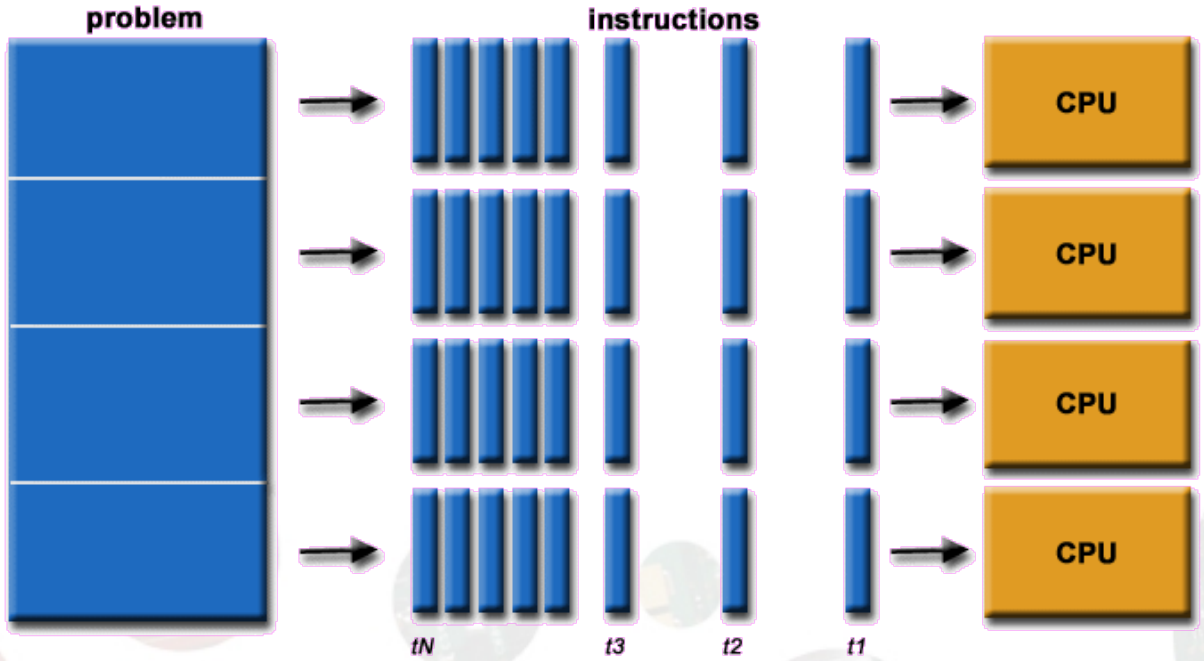
Parallel Programming ?

- Traditionally, software has been written for ***serial*** computation:
  - To be run on a single computer having a single Central Processing Unit (CPU);
  - A problem is broken into a discrete series of instructions.
  - Instructions are executed one after another.
  - Only one instruction may execute at any moment in time.



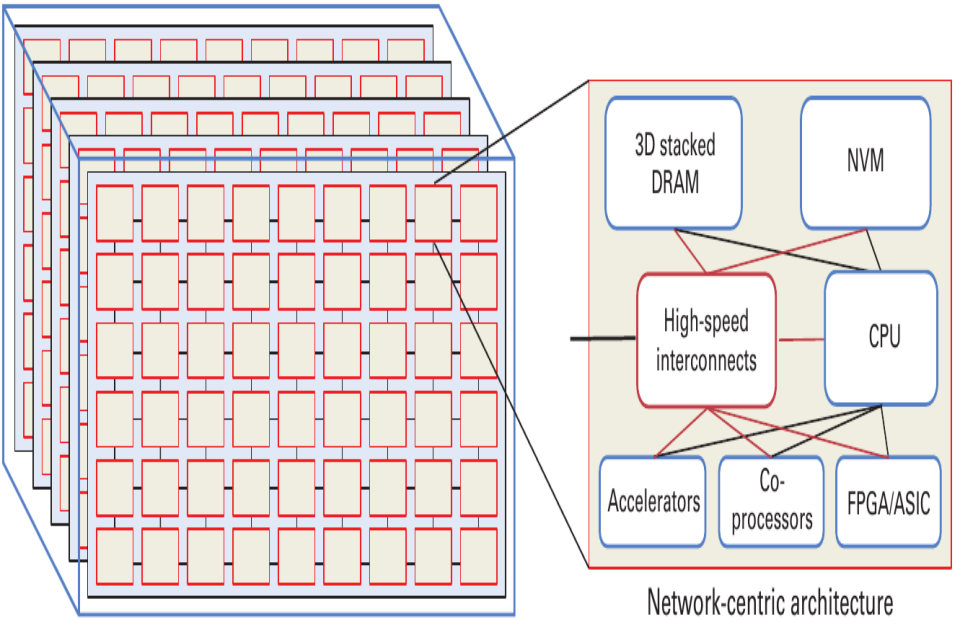
Parallel Computing

- In the simplest sense, ***parallel computing*** is the simultaneous use of multiple compute resources to solve a computational problem.
  - To be run using multiple CPUs
  - A problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different CPUs



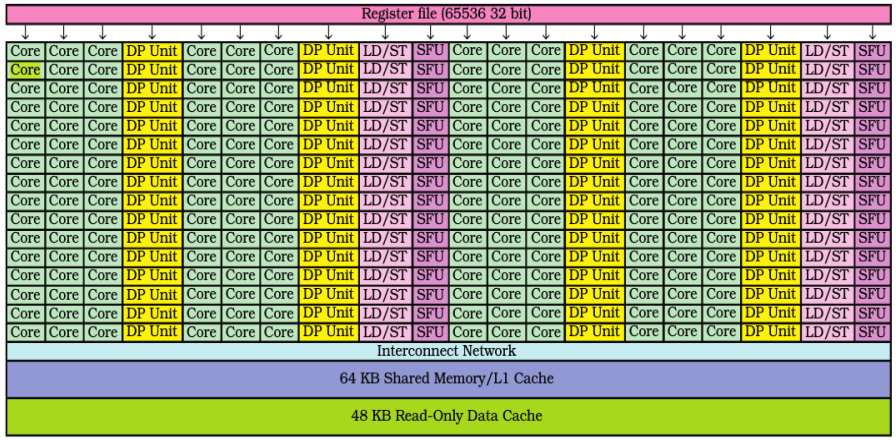
Parallel Computing: Resources

- The compute resources can include:
  - A single computer with multiple processors;
  - A single computer with (multiple) processor(s) and some specialized computer resources (GPU, FPGA …)
  - An arbitrary number of computers connected by a network;
  - A combination of both.



Terabit network connected system

Network-centric architecture



192 **Core** : single-precision cores
64 **DP Unit** : double -precision cores
32 **LD/ST** : load/store units
32 **SFU** : Special Function Units

Parallel Computing: The Computational Problem

- The computational problem usually demonstrates characteristics such as the ability to be:
  - Broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Solved in less time with multiple compute resources than with a single compute resource.
  - Amdahl's Law
    - Speedup = 1 / [(1 - P) + (P / N)]

Parallel Computing: what for? (1)

- Traditionally, parallel computing has been considered to be "the high end of computing" and has been motivated by numerical simulations of complex systems and "Grand Challenge Problems" such as:
  - weather and climate
  - chemical and nuclear reactions
  - biological, human genome
  - geological, seismic activity
  - mechanical devices - from prosthetics to spacecraft
  - electronic circuits
  - manufacturing processes

The future

- During the past 10 years, the trends indicated by ever faster networks, distributed systems, and multi-processor computer architectures (even at the desktop level) clearly show that ***parallelism is the future of computing***.
- It will be multi-forms, mixing general purpose solutions (PC) and very speciliazed solutions as IBM Cells, ClearSpeed, GPGPU from Nvidia etc.

How Parallel Processing?

Instruction Level Parallelism

- Pipelining and Superscalar Execution

Data Level Parallelism

- Vector Instruction or Specilizaed Accelerator

Thread Level Parallelism

- Execute multiple function on different cores

Task Level Parallelism

- Breaking multiple tasks (Memory, Input Output etc) into subtasks and execute using TLP

Writing Parallel Application

A=10;

B=20;

C=A+B;

D=A*B;

E=C*A;

F=C*B;

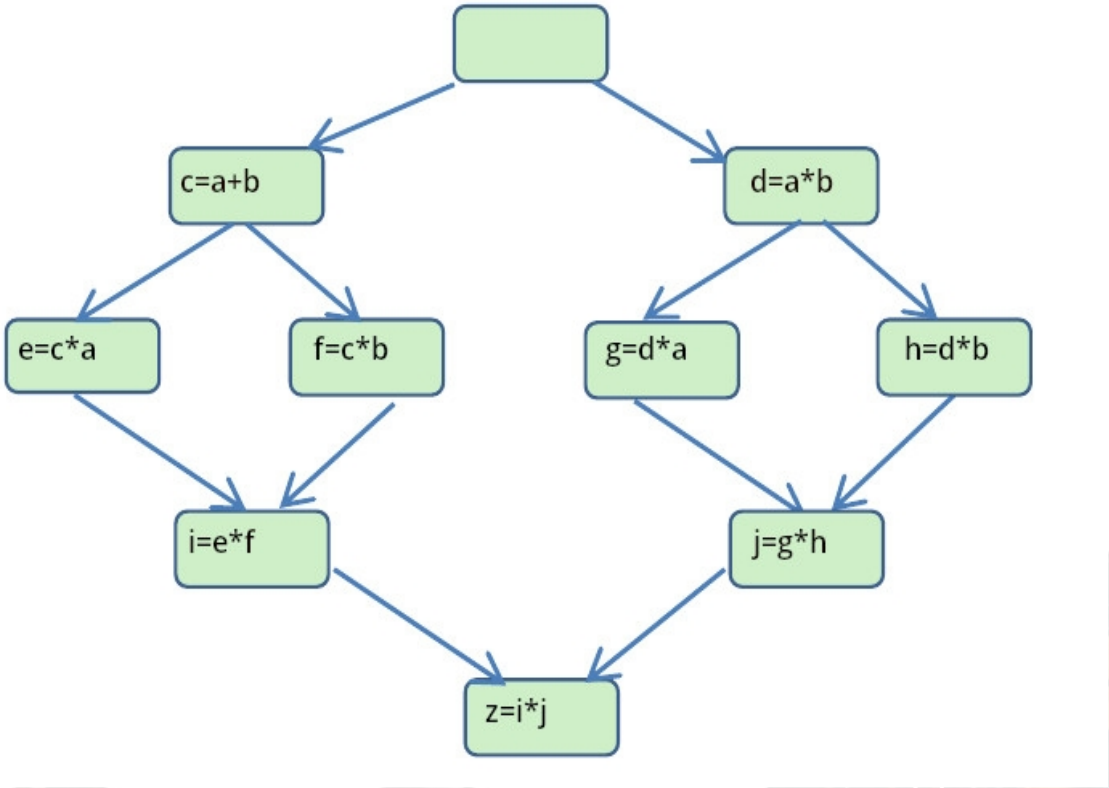
G=D*A;

H=D*B;

I=E*F;

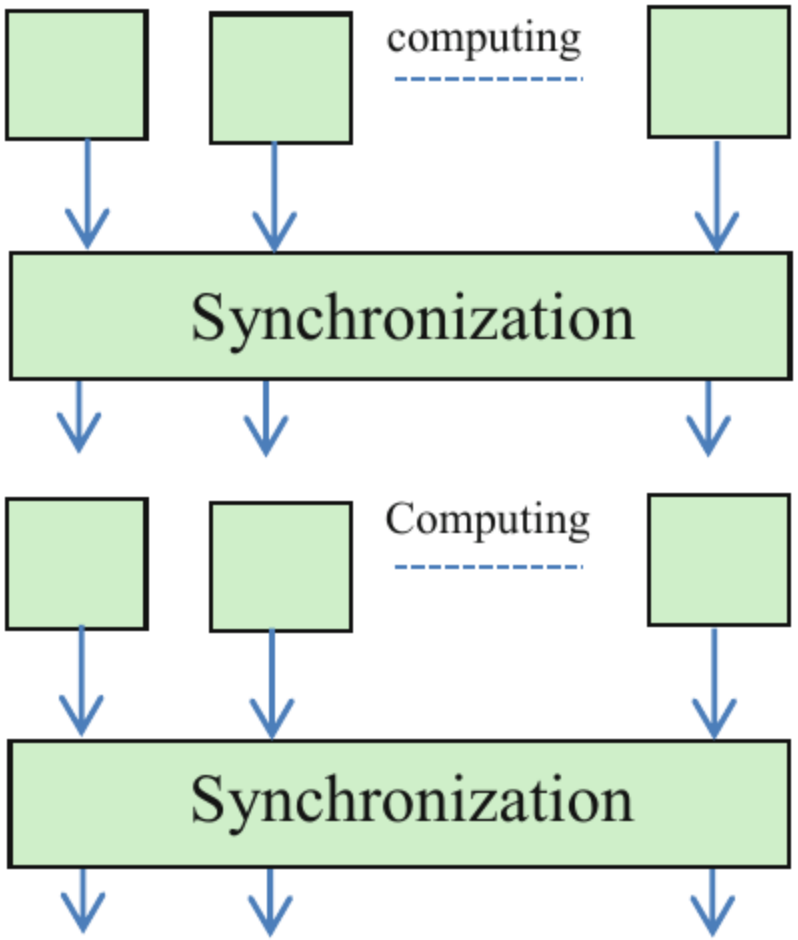
J=G*H;

Z=I*J;

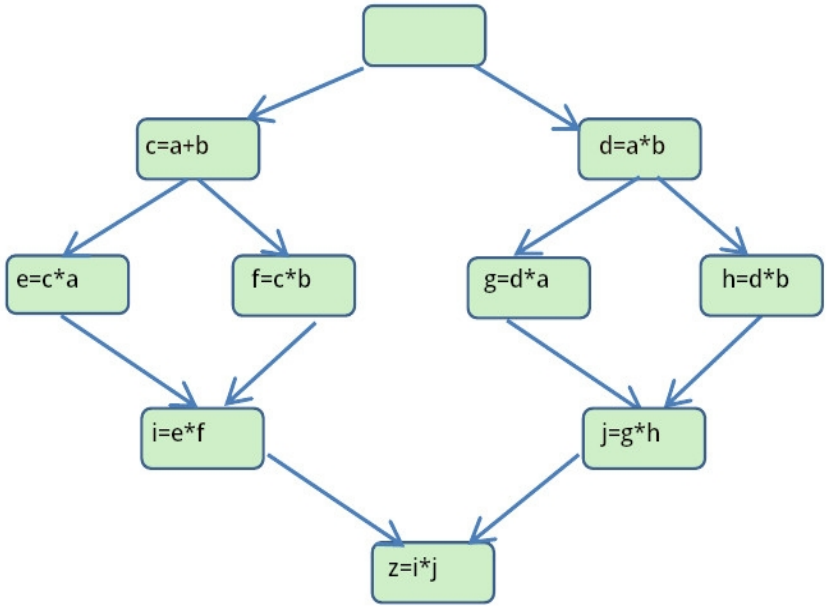


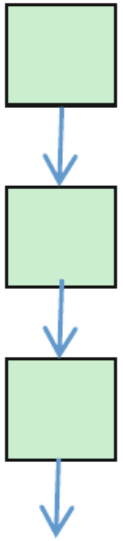
Considerations for Parallel Programs

- Automatic vs. Manual Parallelization
- Understand the Problem and the Program
- Data Dependencies
- Partitioning
- Communications
- Synchronization
- Load Balancing
- Granularity
- I/O
- Limits and Costs of Parallel Programming
- Performance Analysis and Tuning

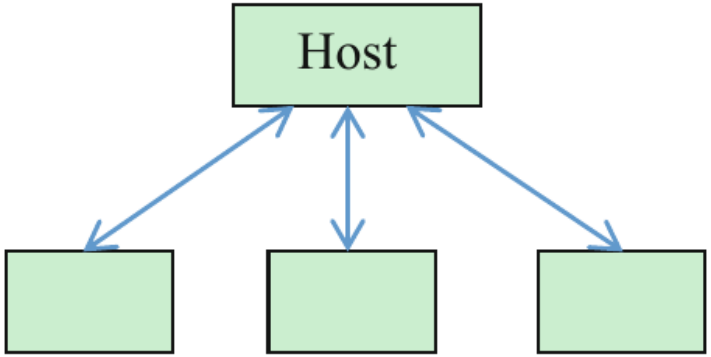


Development and Application of Supercomputing

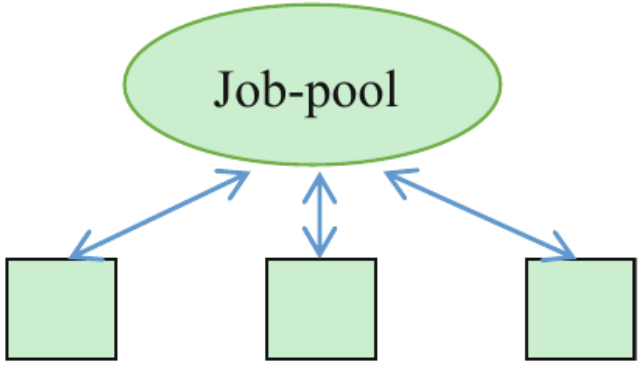




macro-pipeline



master-slave



job-pool

Frameworks for Parallel Programming

Scientific Computing

Big Data Processing

AI and Machine Learning

Parallel Programming Models

Shared Memory: Multi-threading programming languages: such as POSIX Threads, Java Threads, OpenMP.

- OpenMP: Parallelize loops and sections of code using compiler directives.

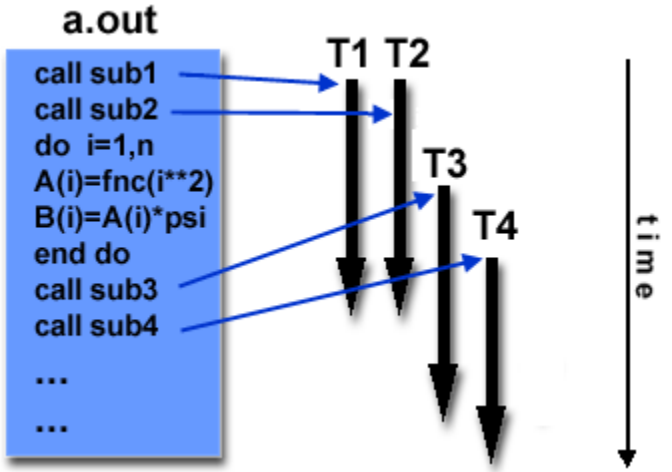
MPI (Message Passing Interface): A widely used standard for writing parallel programs that execute on distributed memory systems. It's commonly used in high-performance computing for scientific simulations and calculations.

OpenACC: OpenACC is a directive-based approach to parallel programming that focuses on using high-level directives to guide the compiler in parallelizing code for accelerators like GPUs.

CUDA: NVIDIA's parallel computing platform and programming model that enables developers to use GPUs for accelerating scientific computations.

BLAS (Basic Linear Algebra Subprograms)

Threads Model

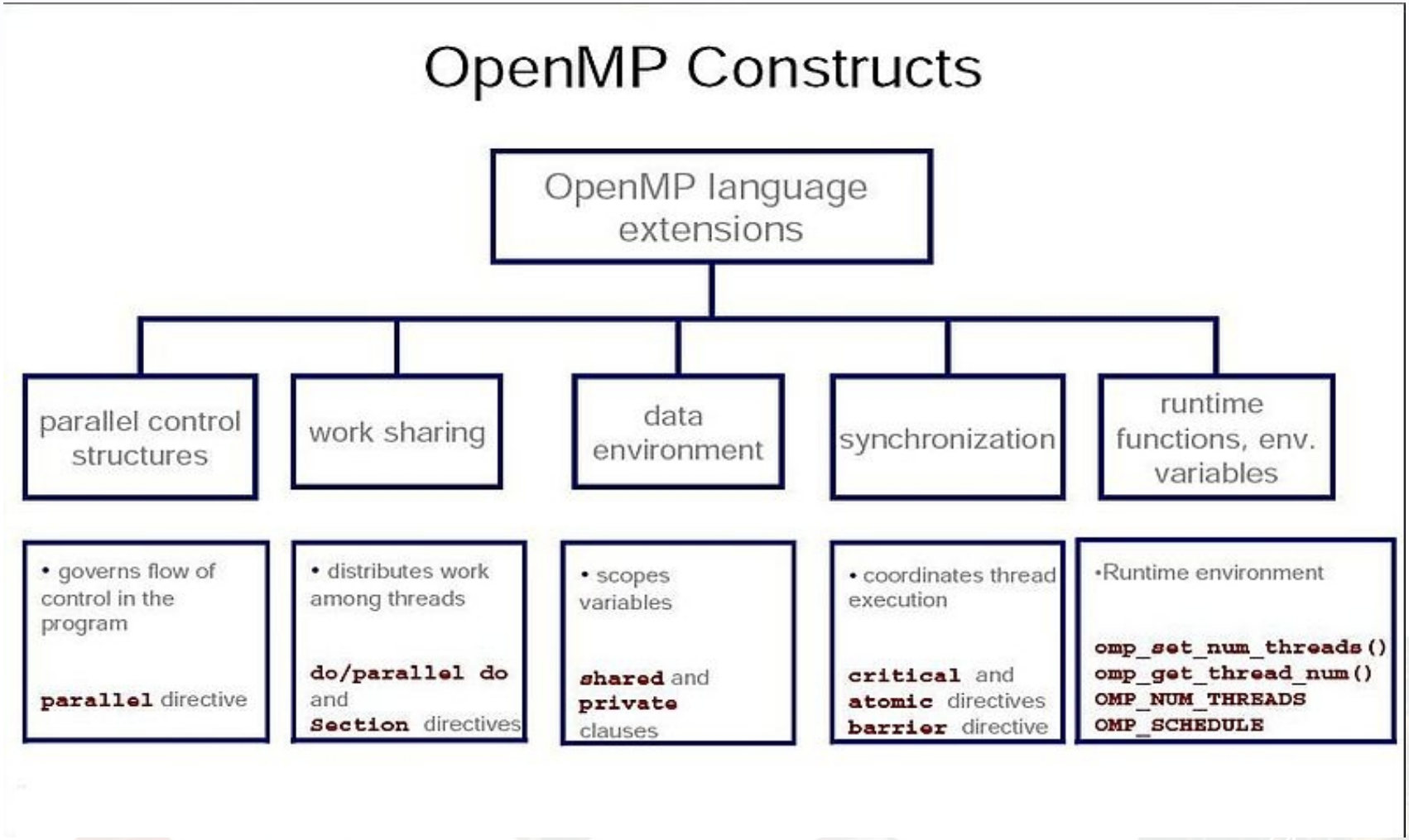


- In the threads model of parallel programming, a single process can have multiple, concurrent execution paths.
- Perhaps the most simple analogy that can be used to describe threads is the concept of a single program that includes a number of subroutines.
- Threads are commonly associated with shared memory architectures and operating systems.

Shared Memory Programming Model: OpenMP

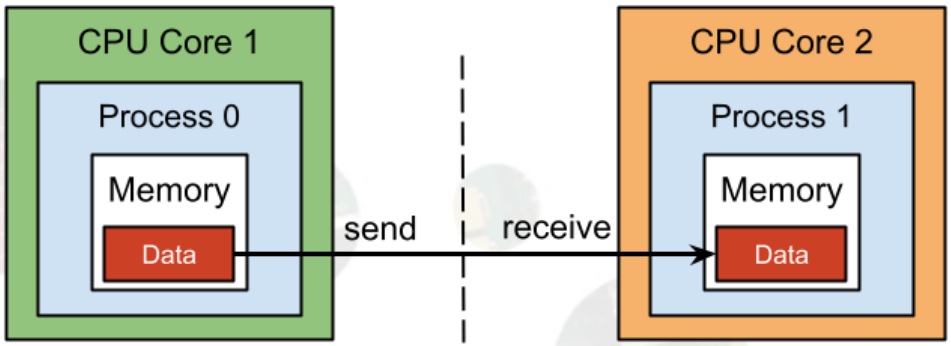
- **OpenMP**
  - Compiler directive based; can use serial code
  - Jointly defined and endorsed by a group of major computer hardware and software vendors. The OpenMP Fortran API was released October 28, 1997. The C/C++ API was released in late 1998.
  - Portable / multi-platform, including Unix and Windows NT platforms
  - Available in C/C++ and Fortran implementations
  - Can be very easy and simple to use - provides for "incremental parallelism"
- Microsoft has its own implementation for threads, which is not related to the UNIX POSIX standard or OpenMP.

Example:


OpenMP Constructs
OpenMP language extensions
parallel control structures
work sharing
data environment
synchronization
runtime functions, env. variables
• governs flow of control in the program
parallel directive
• distributes work among threads
do/parallel do and Section directives
• scopes variables
shared and private clauses
• coordinates thread execution
critical and atomic directives
barrier directive
•Runtime environment
omp_set_num_threads()
omp_get_thread_num()
OMP_NUM_THREADS
OMP_SCHEDULE

Message Passing Model

- The message passing model demonstrates the following characteristics:
  - A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine as well across an arbitrary number of machines.
  - Tasks exchange data through communications by sending and receiving messages.
  - Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.



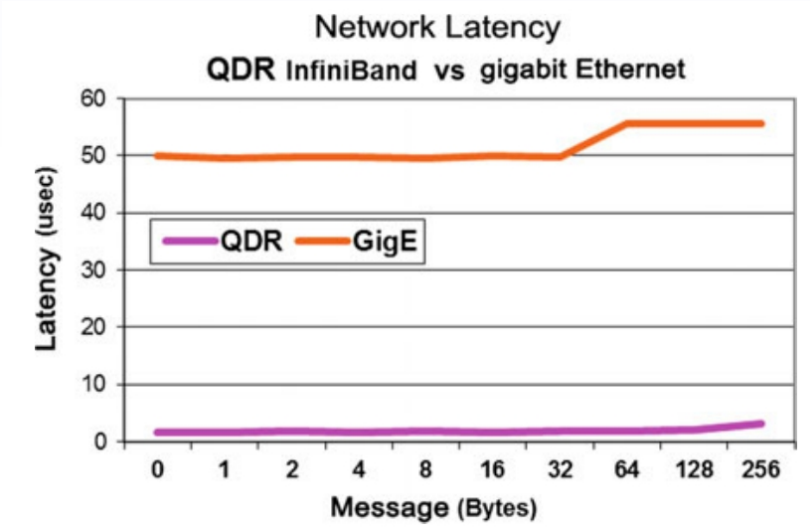
Point-to-Point Communication

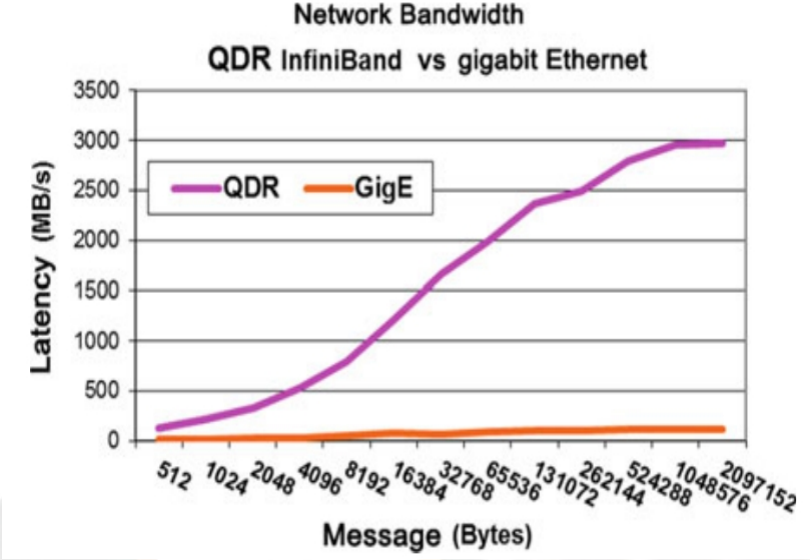
MPI defines more than 35 point-to-point communication methods, including

MPI_Send, MPI_Recv, MPI_Sendrecv, MPI_Isend, MPI_Irecv, MPI_Probe, MPI_Iprobe, MPI_Test, MPI_Testall, MPI_Wait, and MPI_Waitall.

Collective Communication

These include: MPI_Allgather, MPI_Allgatherv, MPI_Allreduce, MPI_Alltoall, MPI_Alltoallv, MPI_Barrier, MPI_Bcast, MPI_Gather, MPI_Gatherv, MPI_Reduce, MPI_Scatter, and MPI_Scatterv.





MPI include file

Declarations, prototypes, etc.

Program Begins

.
.
.

Serial code

Initialize MPI environment *Parallel code begins*

.
.
.

Do work & make message passing calls

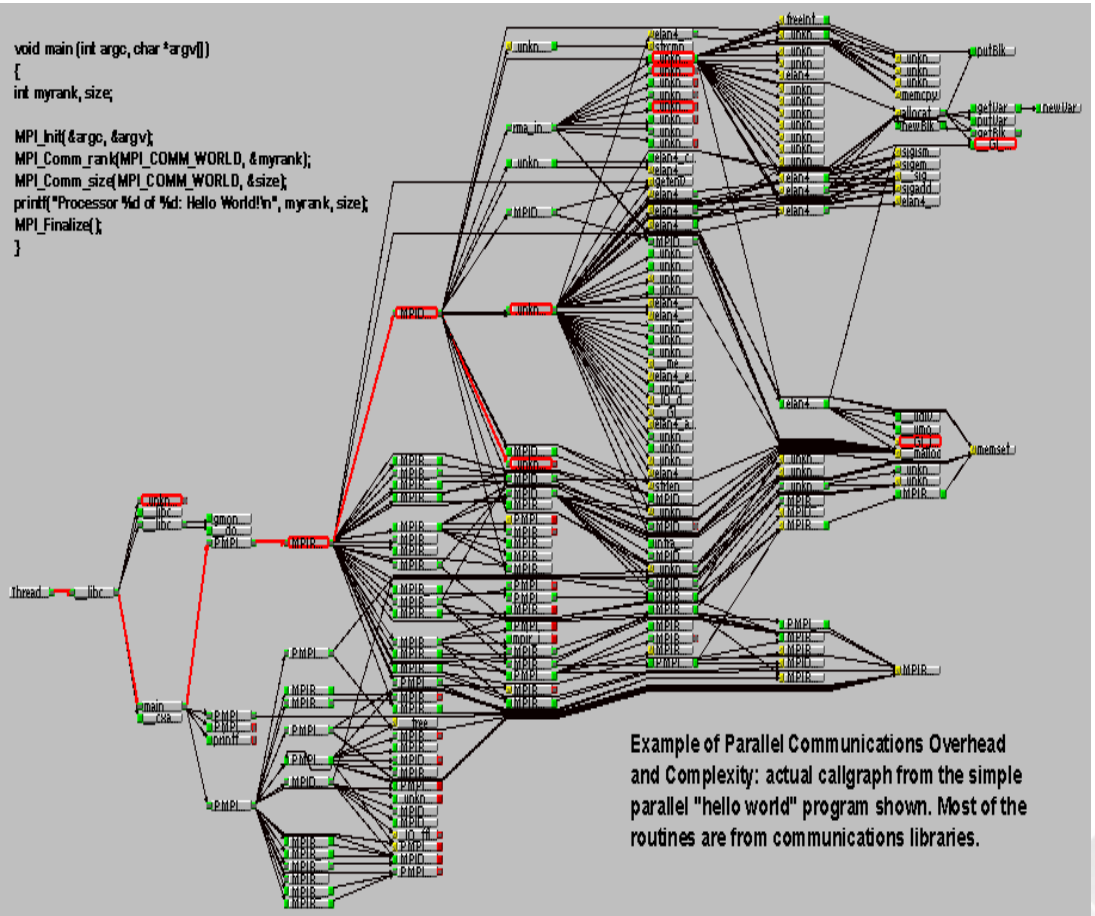
.
.
.

Terminate MPI environment *Parallel code ends*

.
.
.

Serial code

Program Ends



Example of Parallel Communications Overhead and Complexity: actual callgraph from the simple parallel "hello world" program shown. Most of the routines are from communications libraries.

Hybryd

- In this model, any two or more parallel programming models are combined.
- Currently, a common example of a hybrid model is the combination of the message passing model (MPI) with either the threads model (POSIX threads) or the shared memory model (OpenMP). This hybrid model lends itself well to the increasingly common hardware environment of networked SMP machines.
- Another common example of a hybrid model is combining data parallel with message passing. As mentioned in the data parallel model section previously, data parallel implementations (F90, HPF) on distributed memory architectures actually use message passing to transmit data between tasks, transparently to the programmer.

Big Data Processing

Apache Hadoop: An open-source framework for distributed storage and processing of large datasets across clusters of computers using the MapReduce programming model.

Apache Spark: A fast and general-purpose cluster computing system that provides in-memory data processing for big data analytics.

Apache Flink: A stream processing framework for real-time event-driven data processing with support for batch processing as well.

Apache Beam: A unified programming model for both batch and stream data processing, supporting multiple backends like Apache Spark, Google Cloud Dataflow, and more.

AI and Machine Learning:

TensorFlow: An open-source deep learning framework developed by Google that supports both CPU and GPU acceleration for training neural networks.

PyTorch: An open-source machine learning framework developed by Facebook's AI Research lab, known for its dynamic computation graph and ease of use.

Apache MXNet: A flexible and efficient deep learning framework that supports both symbolic and imperative programming paradigms.

Horovod: A distributed deep learning framework designed to scale TensorFlow and PyTorch training across multiple GPUs and nodes.

Distributed TensorFlow: An extension of TensorFlow that enables distributed training of machine learning models across multiple devices and machines.

- Importance of HPC
- Types of Processing System
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- Programming Models
- **Demonstration (Supercomputing System)**

Can the bottlenecks be alleviated with new designs by taking advantage of HPC technologies?

Can RDMA-enabled high-performance interconnects benefit big data processing and deep learning?

Can HPC clusters with high-performance storage systems (e.g., SSD, parallel file systems) benefit big data and deep learning applications?

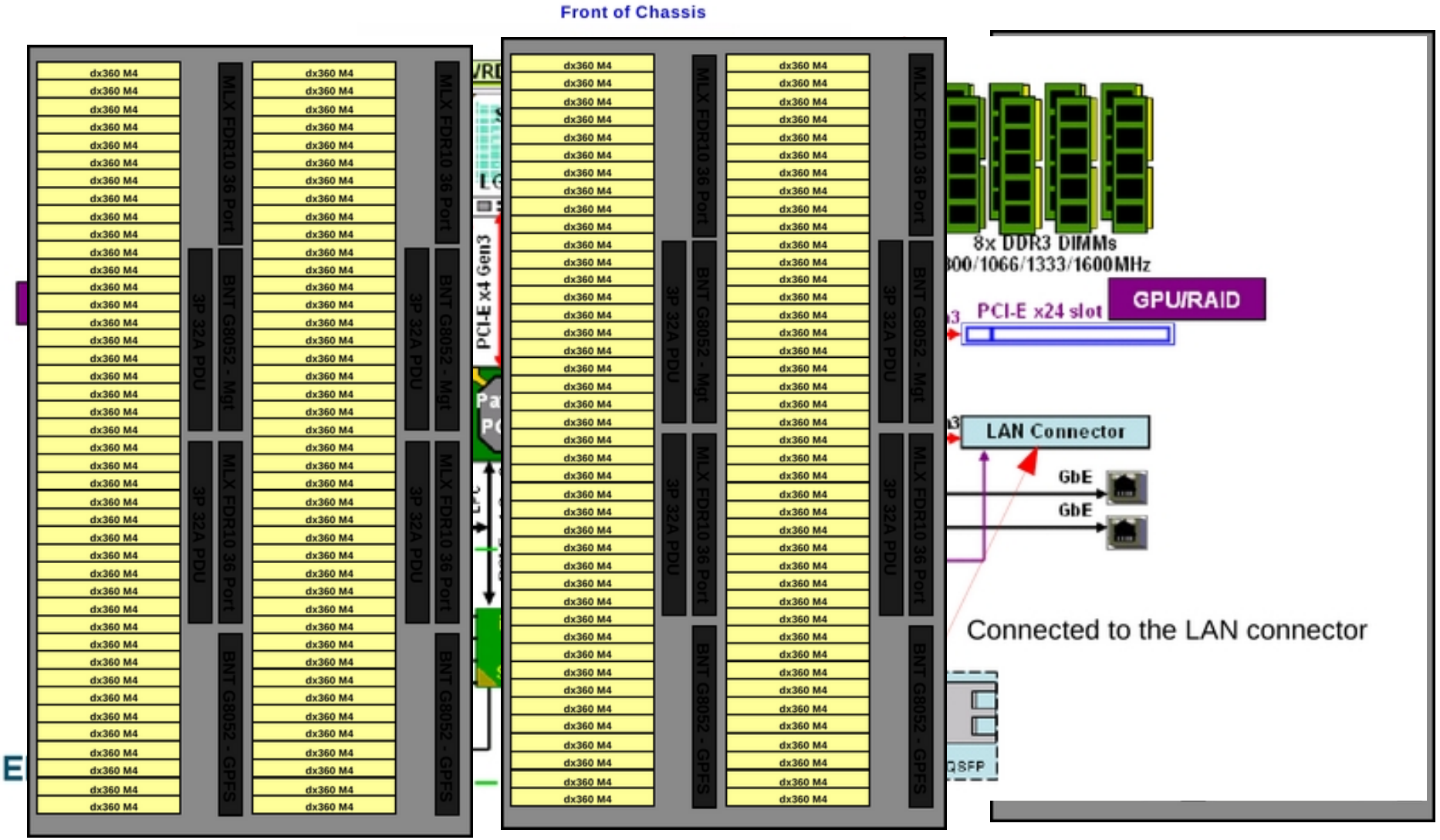
How much performance benefits can be achieved through enhanced designs?

What are the major bottlenecks in current big data processing and deep learning middleware (e.g., Hadoop, Spark)?

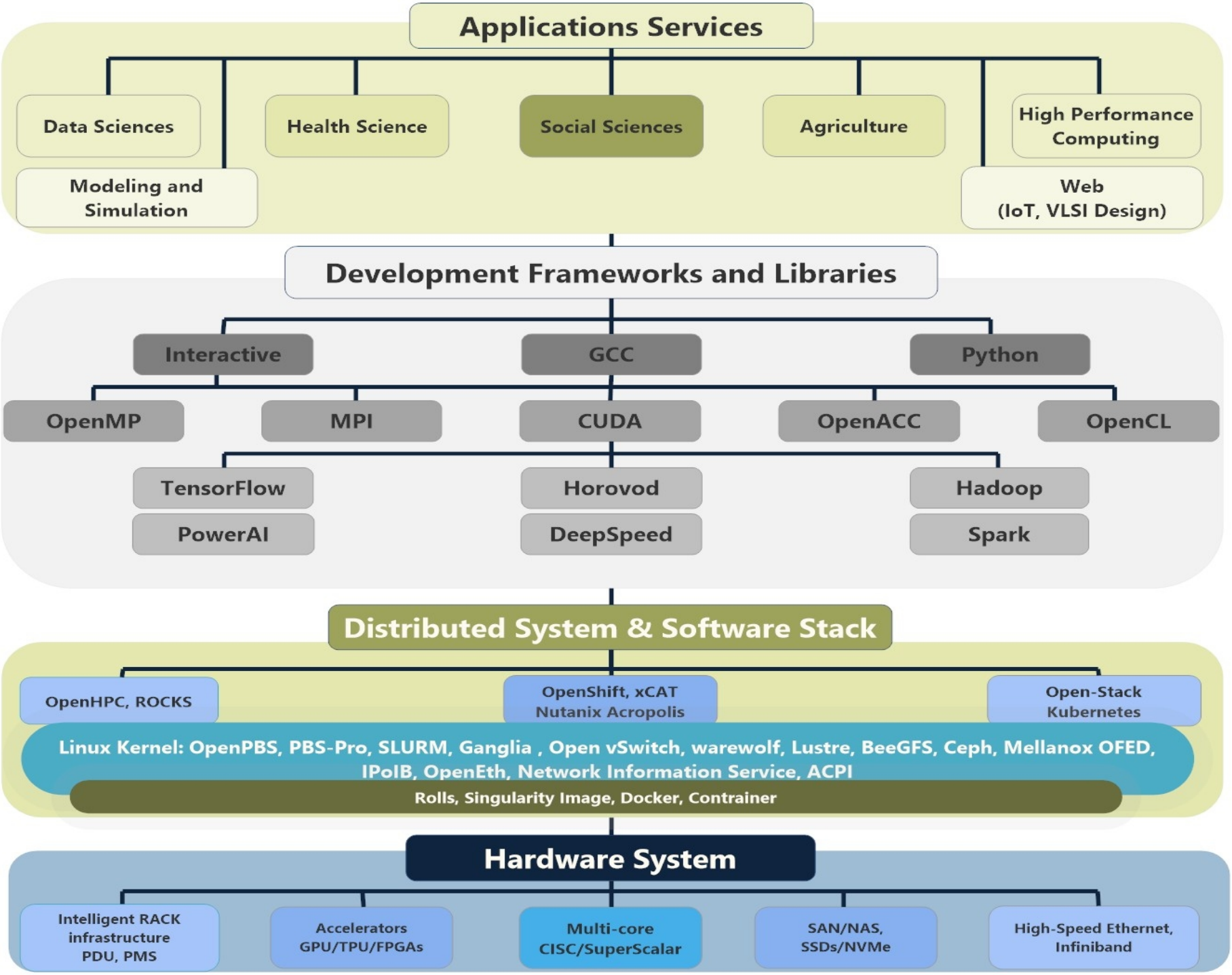
How to design benchmarks for evaluating the performance of big data and deep learning middleware on HPC clusters?

Bring HPC, big data processing, and deep learning into a "convergent trajectory"!

Bottom-up Design Approach



https://www.bsc.es/marenostrum-support-services/marenostrum-system-architecture/compute-racks


Applications Services
Data Sciences
Health Science
Social Sciences
Agriculture
High Performance Computing
Modeling and Simulation
Web (IoT, VLSI Design)
Development Frameworks and Libraries
Interactive
GCC
Python
OpenMP
MPI
CUDA
OpenACC
OpenCL
TensorFlow
Horovod
Hadoop
PowerAI
DeepSpeed
Spark
Distributed System & Software Stack
OpenHPC, ROCKS
OpenShift, xCAT Nutanix Acropolis
Open-Stack Kubernetes
Linux Kernel: OpenPBS, PBS-Pro, SLURM, Ganglia , Open vSwitch, warewolf, Lustre, BeeGFS, Ceph, Mellanox OFED, IPoIB, OpenEth, Network Information Service, ACPI
Rolls, Singularity Image, Docker, Contrainer
Hardware System
Intelligent RACK infrastructure PDU, PMS
Accelerators GPU/TPU/FPGAs
Multi-core CISC/SuperScalar
SAN/NAS, SSDs/NVMe
High-Speed Ethernet, Infiniband

Challenges

How to Write Highly Scalable and Portable Parallel Programs?

How to Enable Automated Use of Effective Parallel Programming Techniques When Writing Parallel Programs?

How to Enforce the Use of Parallel Programming Design Principles During the Programming Process?

How to Employ Suitable Optimization Techniques?

How to Promote Interdisciplinary Collaboration?

Barcelona Supercomputing Center

Linux Operating system,

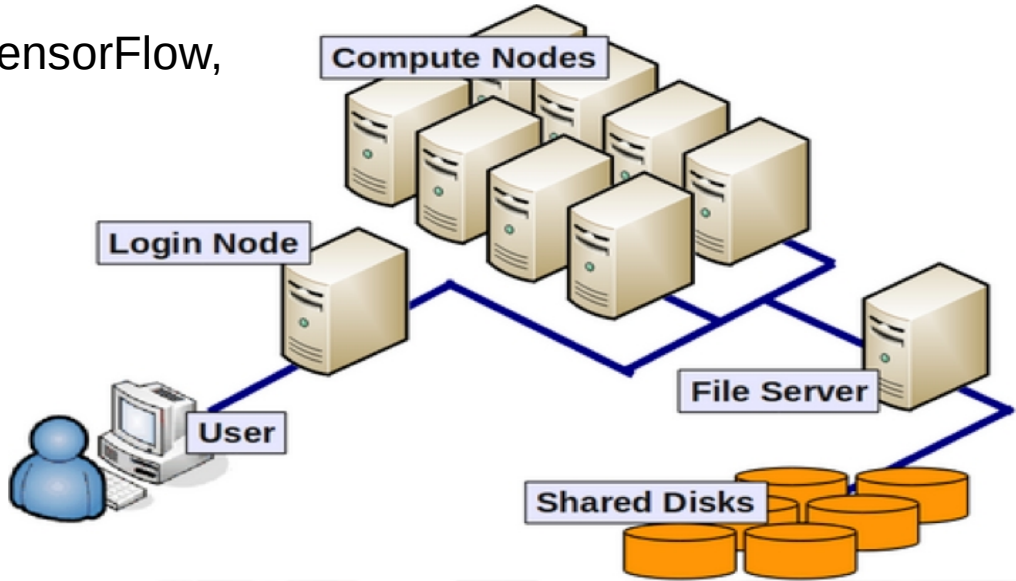
54 Servers

Two Power 9 processors, 4 NVIDIA GPUs with 512 GB of main memory

Interconneciton InfiniBand is an industry-standard to interconnect servers that allows the local memory of one server to be accessed from remote servers speedily.

The software stack

Horovod, from Uber. Horovod Plugs into TensorFlow, PyTorch, and MXNet.



Supercomputing Architectures and Development Methodologies

for AI and Big Data Applications

by: Tassadaq Hussain

Professor Department of Electrical Engineering

Namal University Mianwali

Collaborations:

Barcelona Supercomputing Center, Spain

European Network on High Performance and Embedded Architecture and Compilation

Pakistan Supercomputing Center