Summer School on Supercomputing & AI for Tech Entrepreneurs

29-31 August 2025 Namal University Mianwali

From Edge to Bare-Metal: HPC Hardware and Software Architectures for AI Product Development

by: Tassadaq Hussain

Professor Department of Electrical Engineering

Namal University Mianwali

Affiliated Partners:

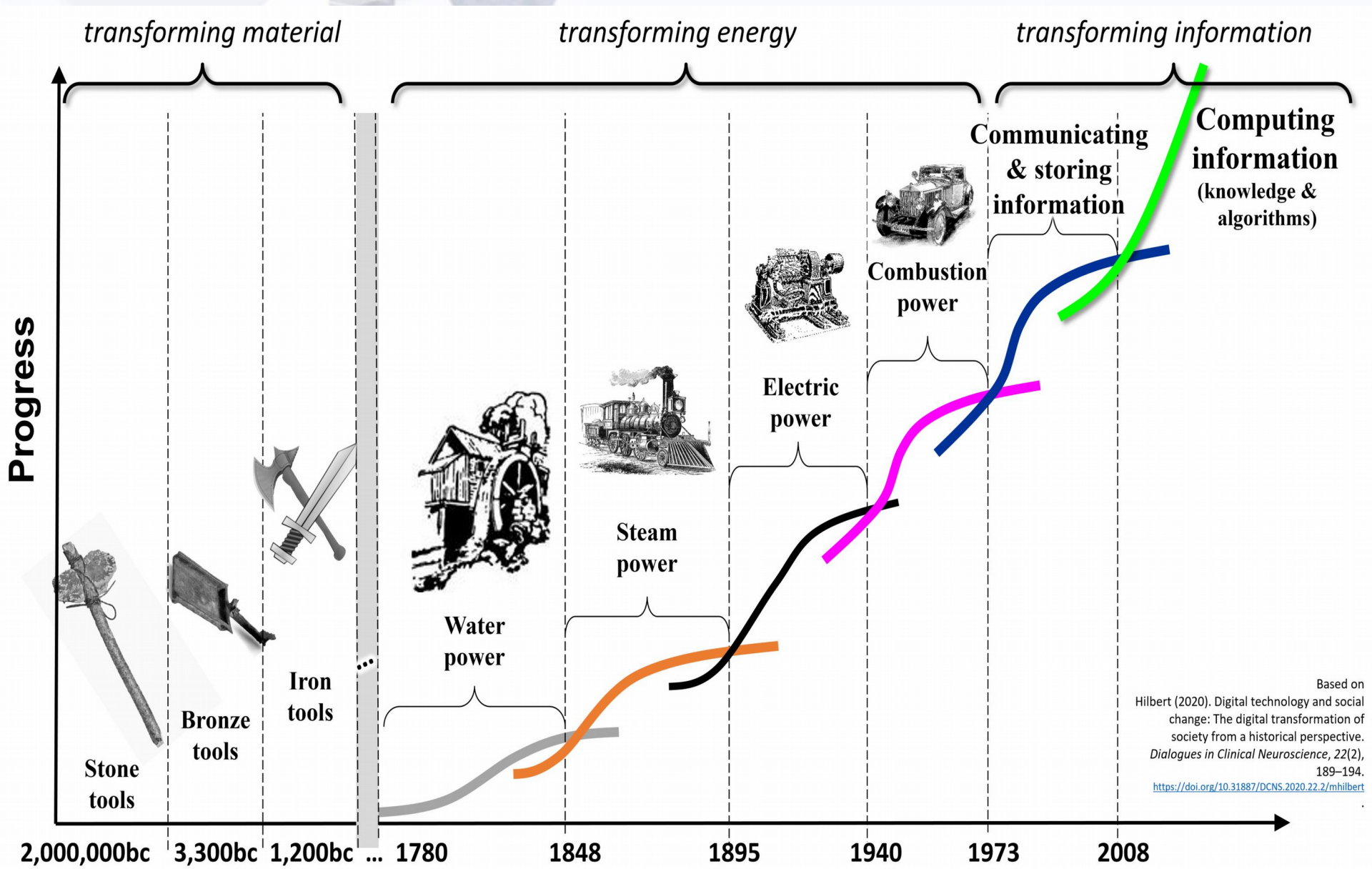
Barcelona Supercomputing Center, Spain

European Network on High Performance and Embedded Architecture and Compilation

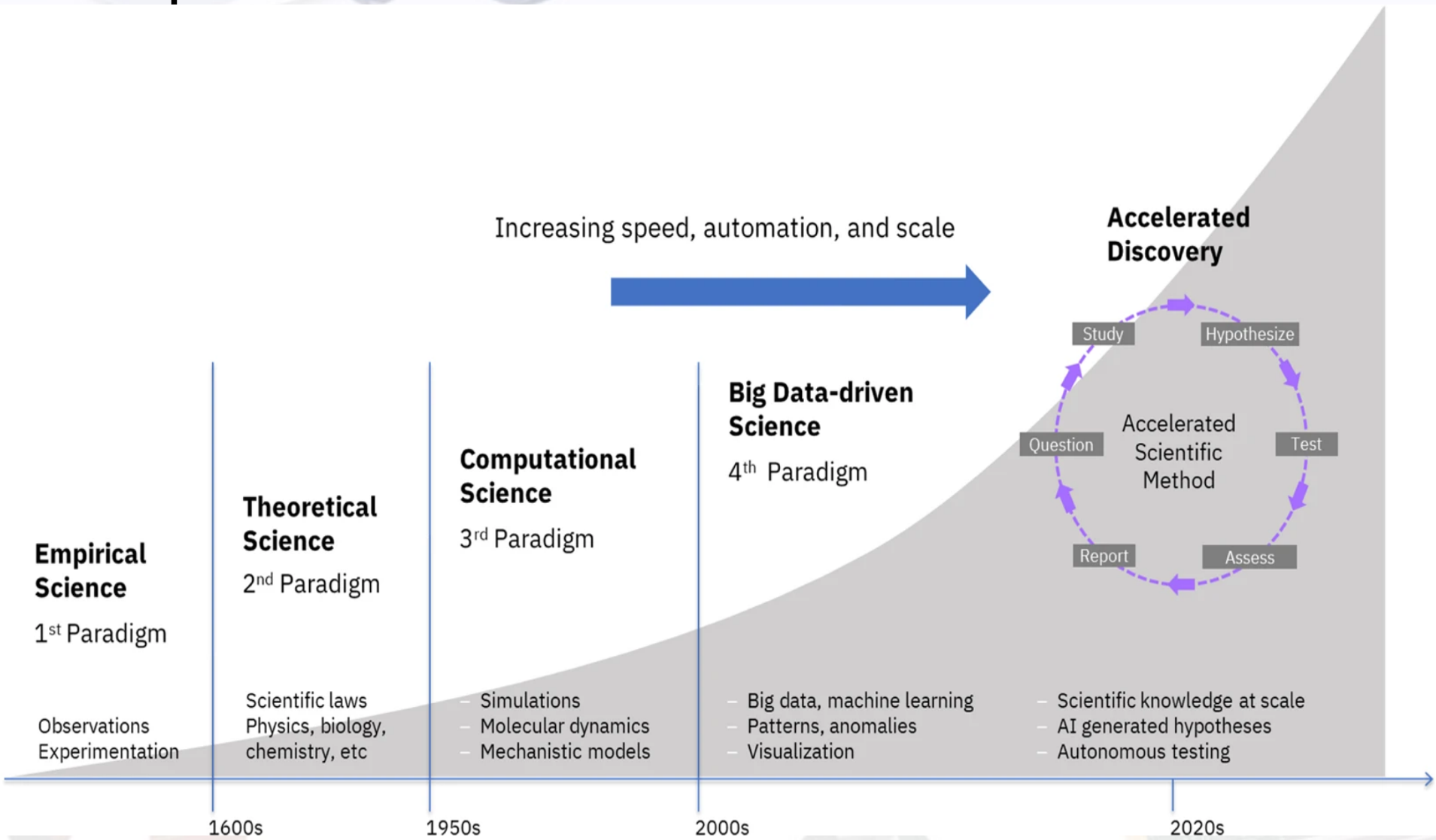
Pakistan Supercomputing Center

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

Human Progress: From the Stone Age to the Data Age

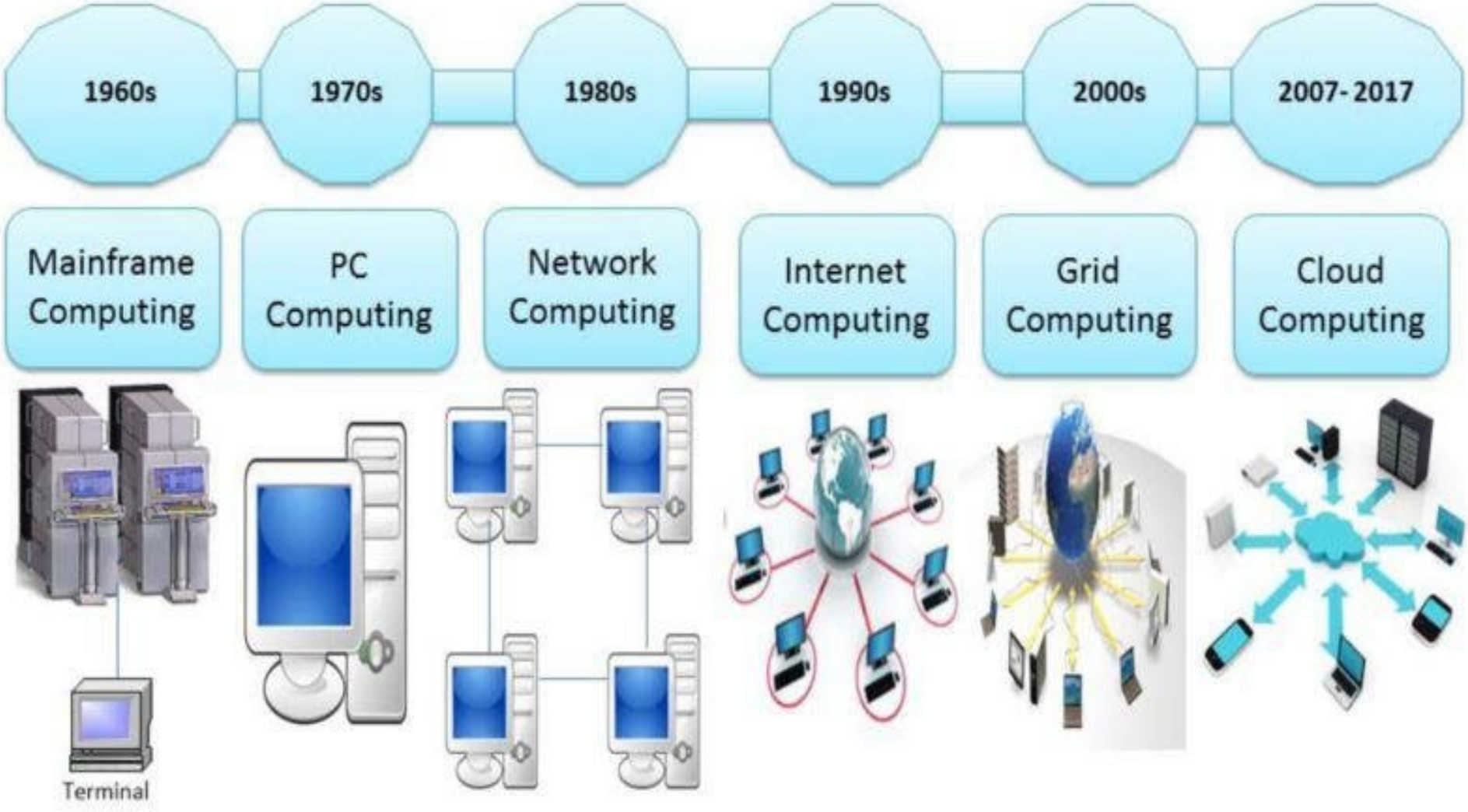


From Age of Empirical Science to Computational-Science



https://www.nature.com/articles/s41524-022-00765-z/figures/1

Journey of Computing: Six Decades of Transformation



Ecosystem of Modern Industry

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

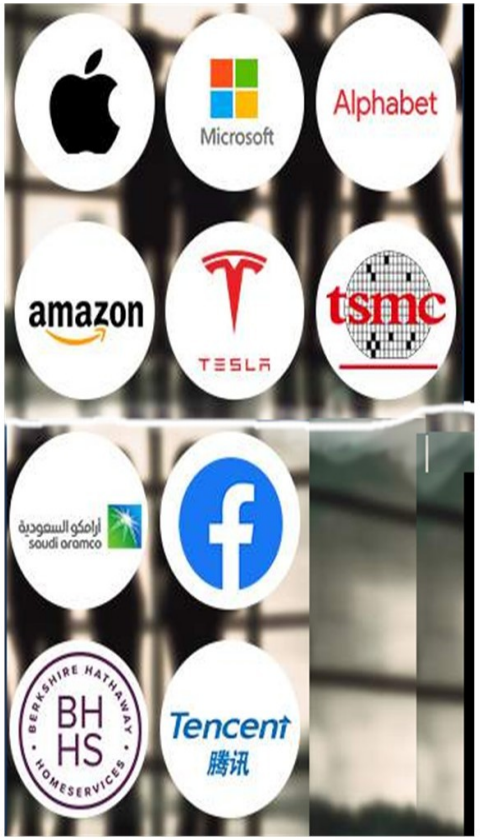
Top500 List
8 PetaFLOPS
@2022

uProcessor
100 B$ @2020

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

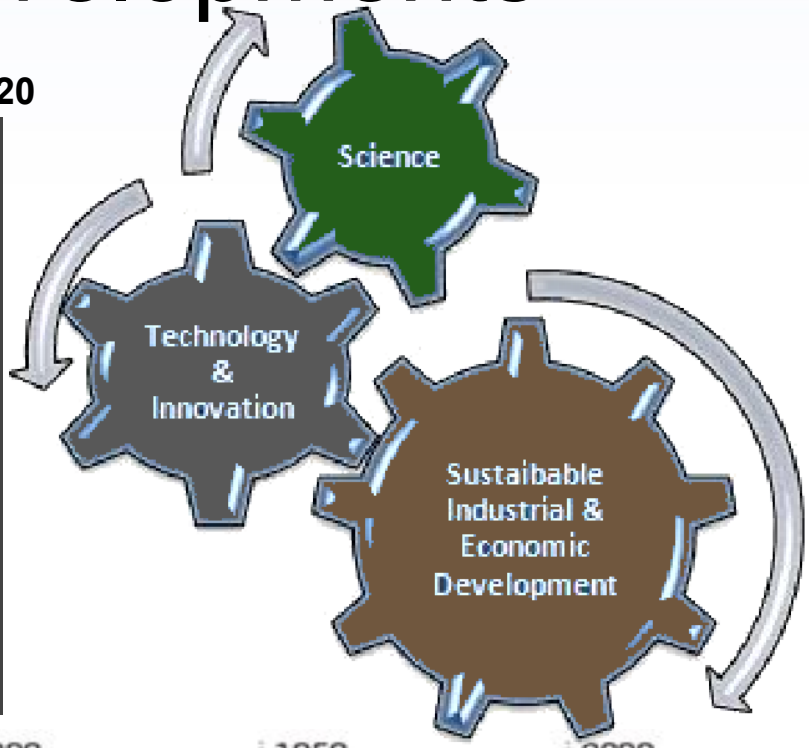
Computing

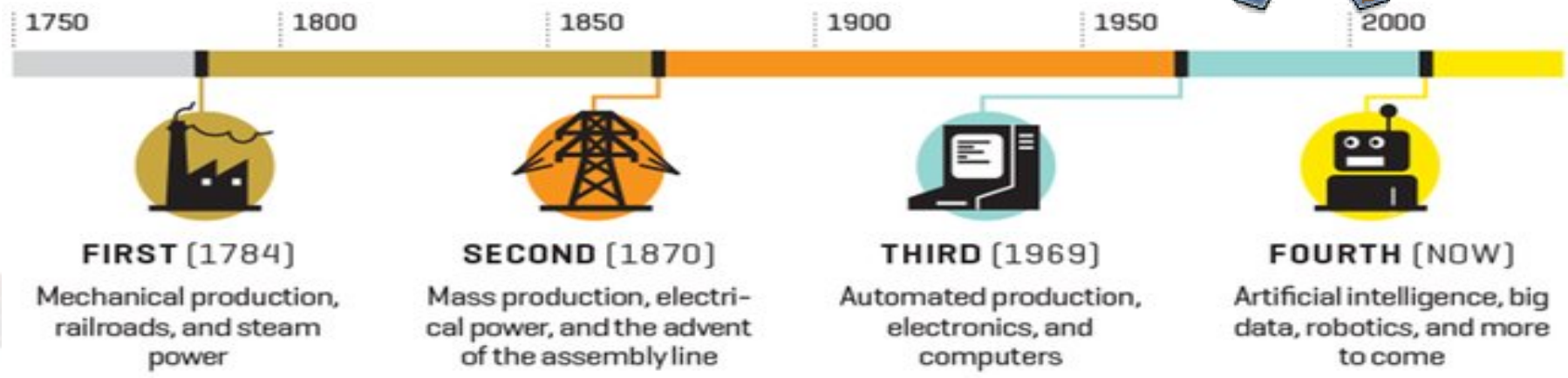
Digital Industrial Age
5.5 Trillion $ Revenue@2021



Industrial Revolutions and Sustainable Developments

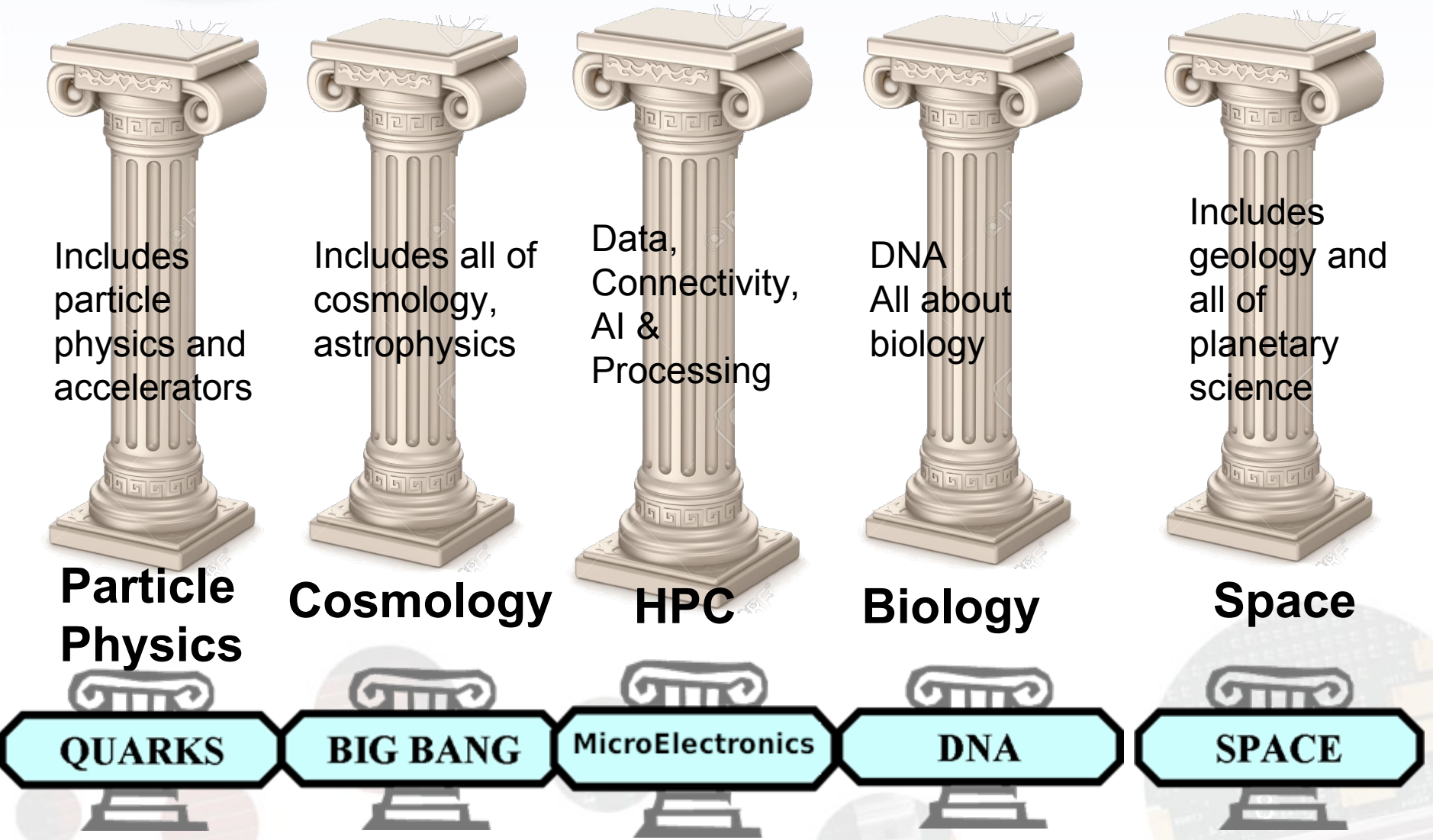
Top 10 Biggest Companies By Market Size From 2010 - 2020





Pillars of Science

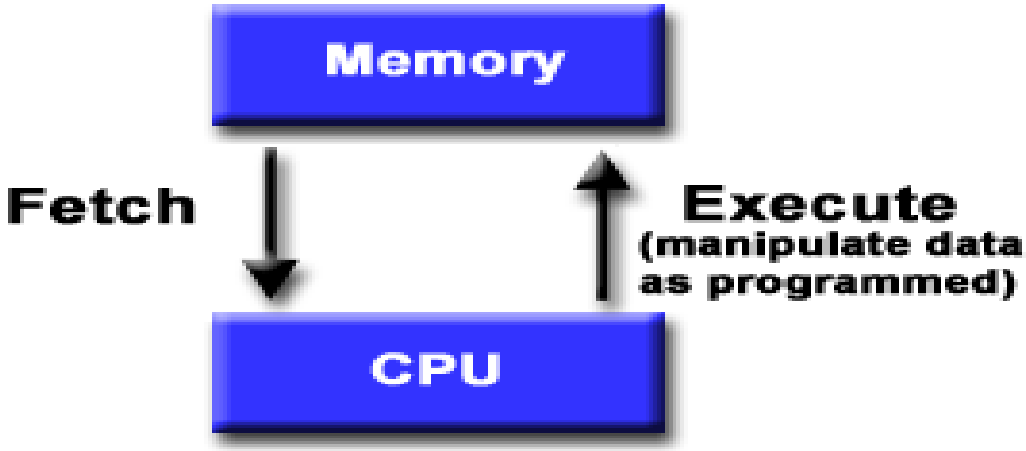
Fermi National Accelerator Laboratory



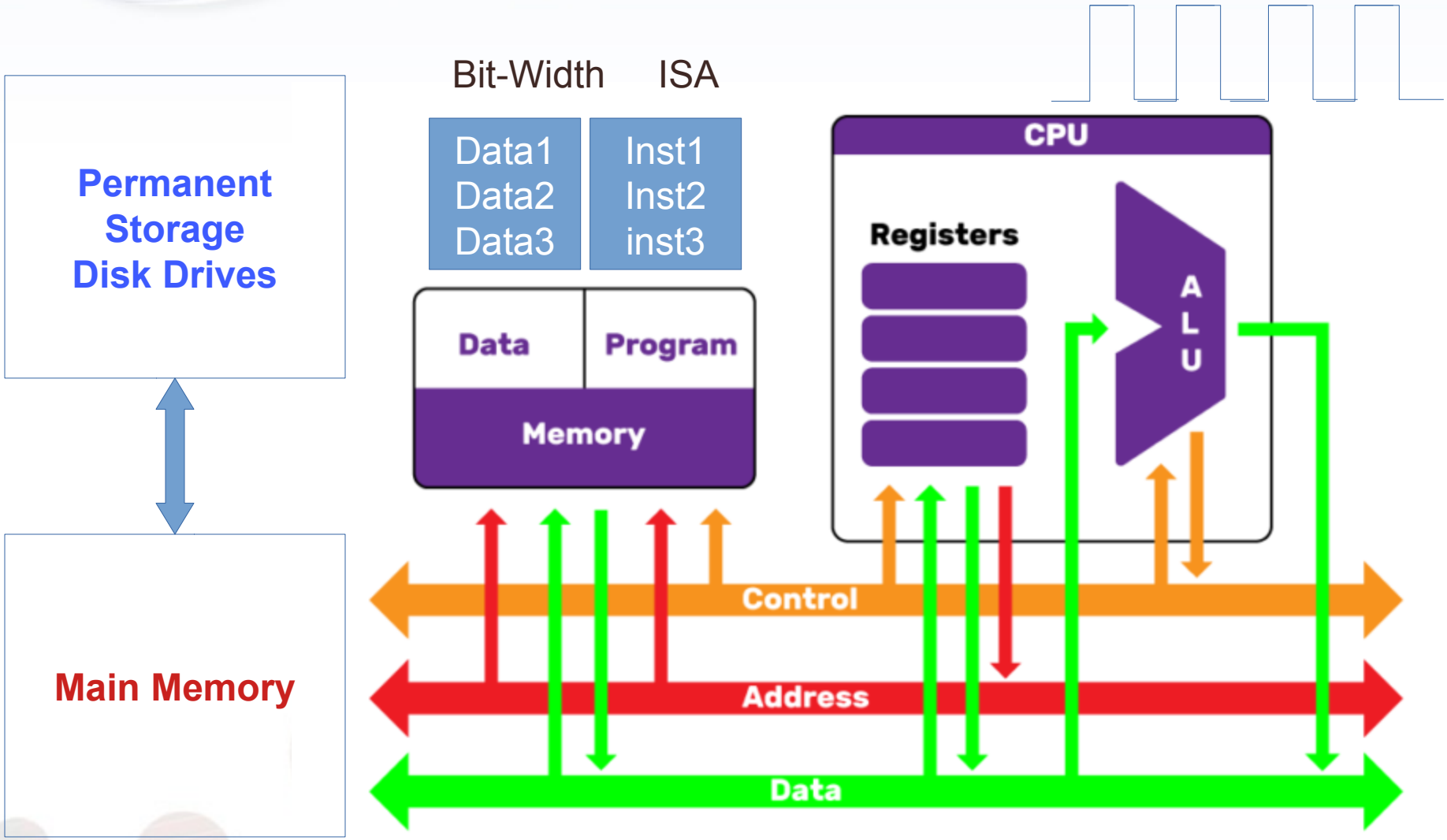
- Importance of HPC
- **Types of Processing System**
- HPC System Architecture
- Supercomputing Classes and Infrastructure
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

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



Generations of Classical Microprocessors:

First-generation –

From 1971 to 1972 the era of the first generation came which brought microprocessors like INTEL 4004 Rockwell international PPS-4 INTEL 8008 etc.

Second generation –

The second generation marked the development of 8-bit microprocessors from 1973 to 1978. Processors like INTEL 8085 Motorola 6800 and 6801 etc came into existence

HPC Microprocessors

6th "Parallel & Multi-core Era" (mid-2000s onwards) (Intel Xeon, AMD Opteron):

- Widespread adoption of massively parallel processors. SIMD/vector extensions (SSE, AVX) and NUMA architectures.

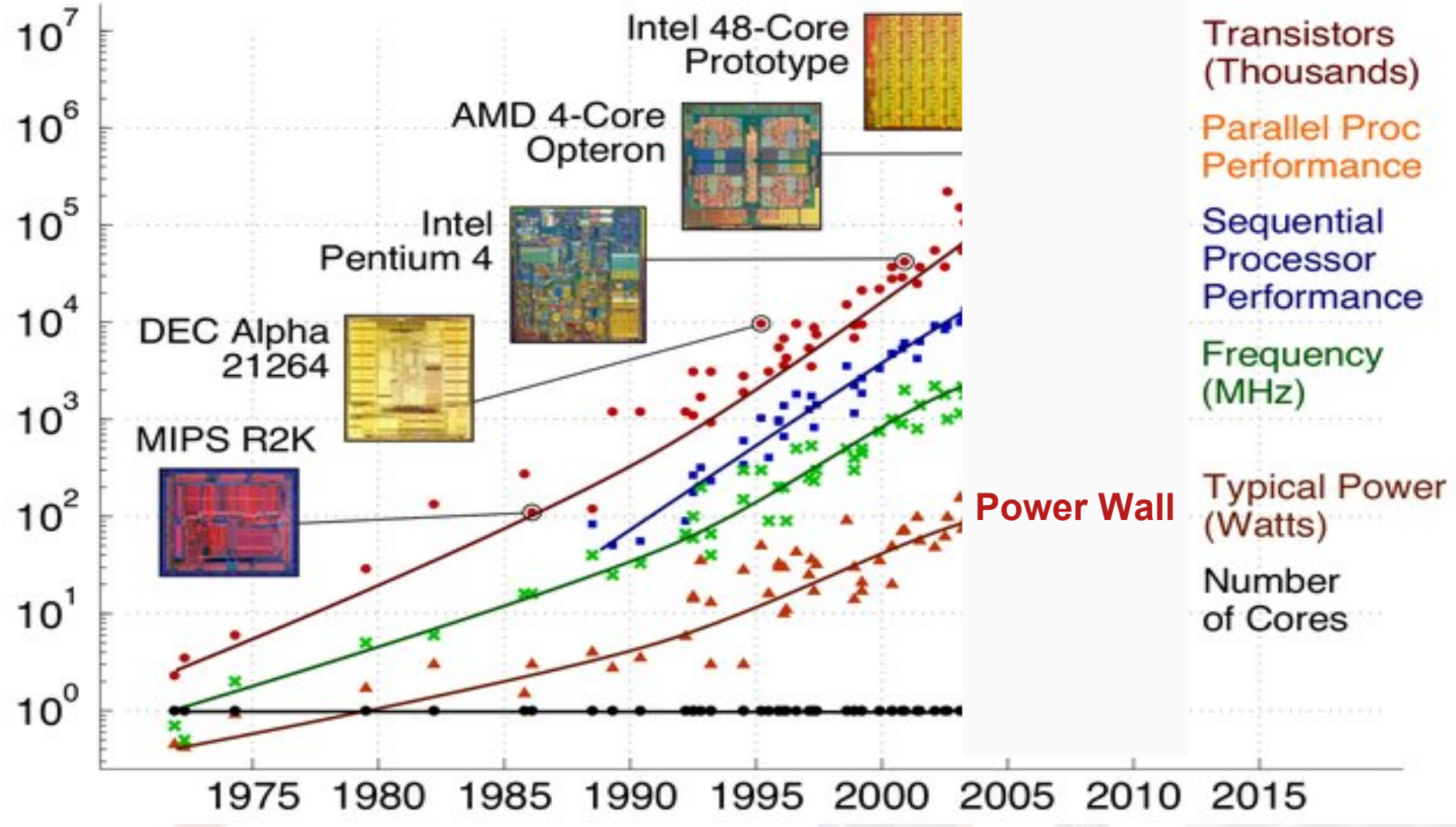
7th "Accelerator Era" (2010 onwards) Co-Processor Xeon Phi (Many Integrated Core, MIC):

- Integration of GPUs (NVIDIA CUDA, AMD ROCm) into HPC clusters. Heterogeneous

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

Performance: Transistor Count, Frequency, Power and Multi Processor



Conventional hardware

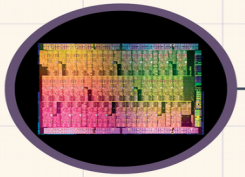
Neuromorphic hardware

POWER

HOOVER DAM
Produces up to 2 gigawatts

MEGAWATTS

SUPERCOMPUTER CHIP
Not analogous to neural function. Runs slower than biological neural



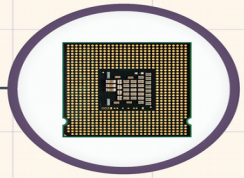
WIND TURBINE
Produces ~2-3 megawatts

20 WATTS

BRAIN
~85 billion neurons
1 quadrillion synapses

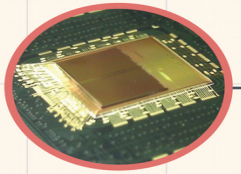
50-100 WATTS

DESKTOP COMPUTER CHIP
Runs vastly slower than biological neural networks

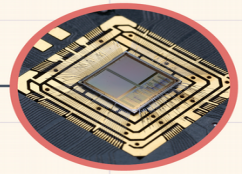


1 WATT

SPINNAKER
Variable number of artificial neurons, typically ~1,000
1 million synapses per 1,000 neurons
Runs at speed of biological neural networks



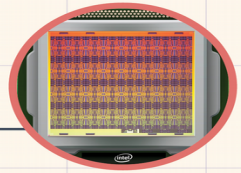
LIGHT BULB
Uses ~20 watts



1 WATT

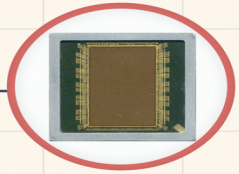
BRAINSCALES
512 artificial neurons
128,000 synapses
10,000x faster than biological neural networks

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

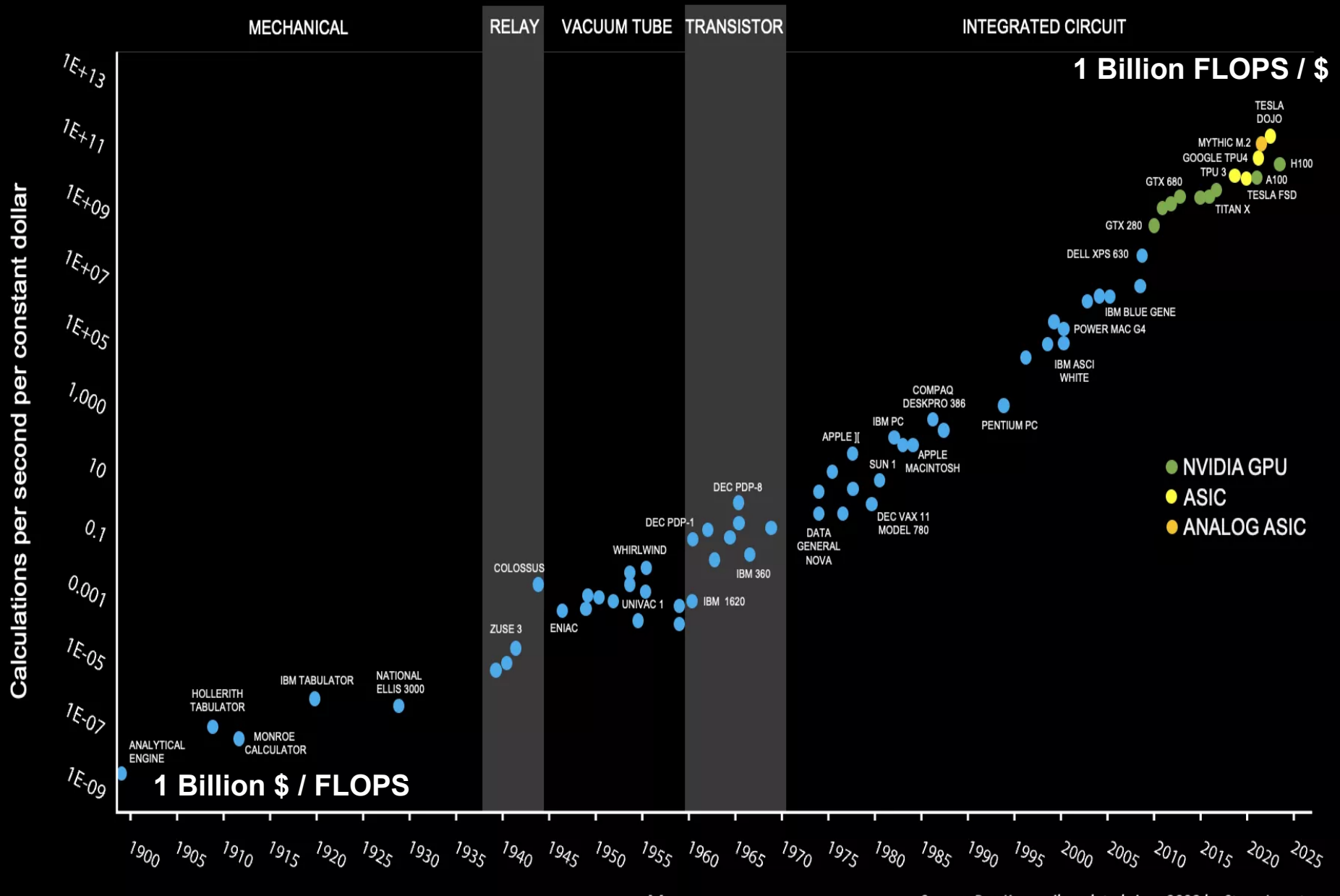


TENS OF MILLIWATTS

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

125 YEARS OF MOORE'S LAW



Source: Ray Kurzweil, updated since 2008 by Steve Jurvetson

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

Criteria of HPC (in FLOPS)

Traditional Definition (1990s–2000s):

- HPC was defined as performance beyond what a typical desktop/workstation could achieve.
- Roughly > 1 GFLOP (10^9 FLOPS) was considered HPC in the 1990s.

Cluster Era (~2000s):

- Systems with sustained > 1 TFLOP (10^{12} FLOPS) entered the HPC category.

HPC Computer Architectures

By System Composition

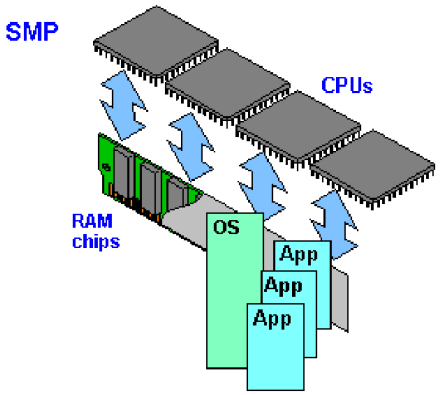
By Instruction and Data

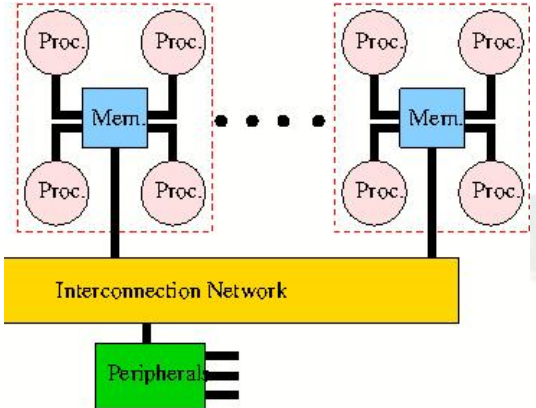
By Memory Structure and Architecture

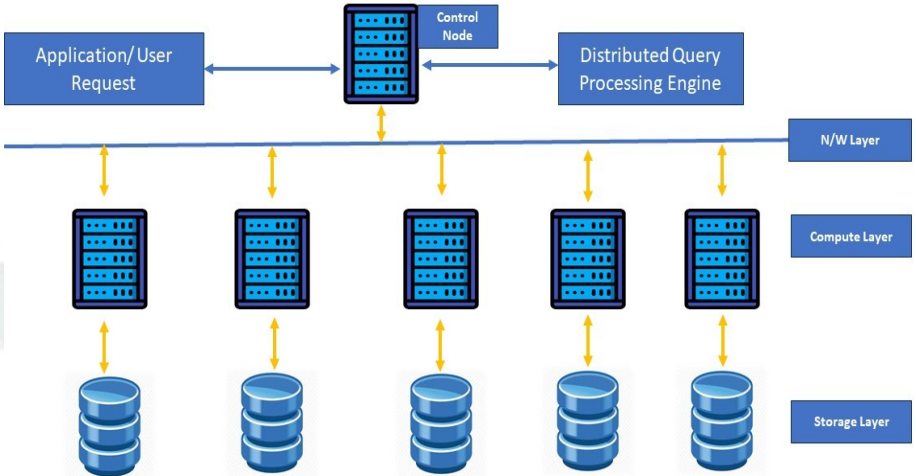
By Network / Interconnect

By System Composition

- SMP (Symmetric Multi-Processing) — one server, many CPUs share one memory. Small/medium shared-memory codes; simple to manage.
- ccNUMA (cache-coherent NUMA) — big server; each CPU socket has “local” RAM. Large in-node parallel jobs; use NUMA pinning for speed.
- Cluster / MPP (Massively Parallel Processing) — many nodes, each with its own memory; talk over a fast network. Most supercomputers; MPI simulations, weather/CFD/FEA.
- GPU-Accelerated (Heterogeneous) — CPU + multiple GPUs per node. AI training/inference, dense math, MD codes.
- Many-core / Vector — very wide SIMD or specialty chips. Niche codes tuned for vectorization (e.g., weather kernels).

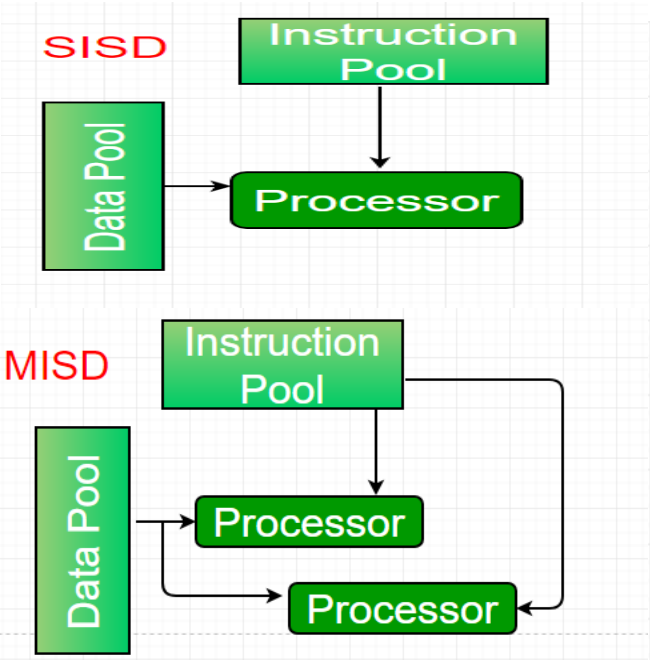


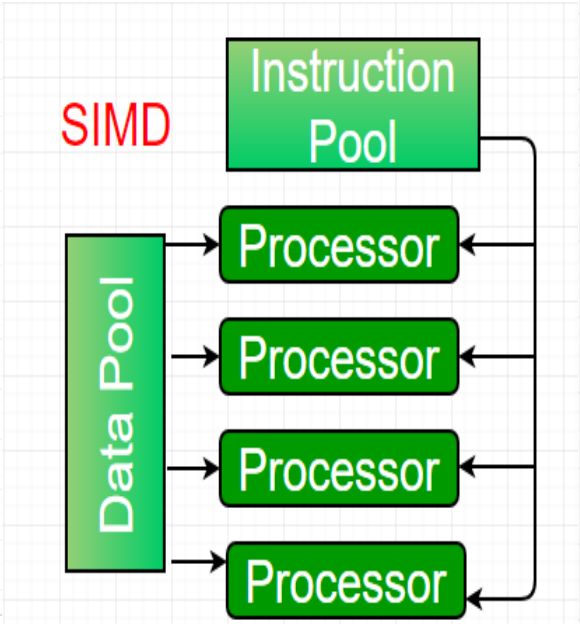


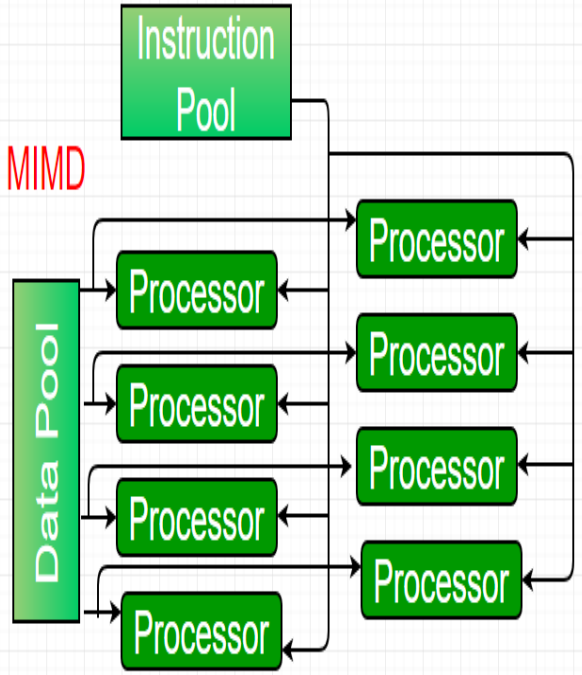


Inst and Data: Flynn's Classical Taxonomy

- There are different ways to classify HPC 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***.



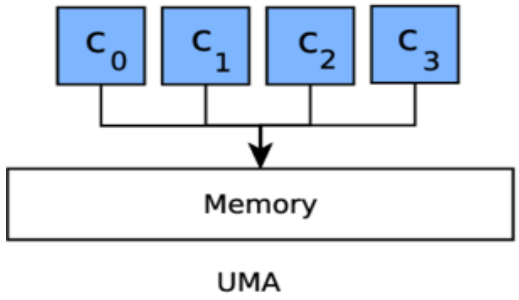


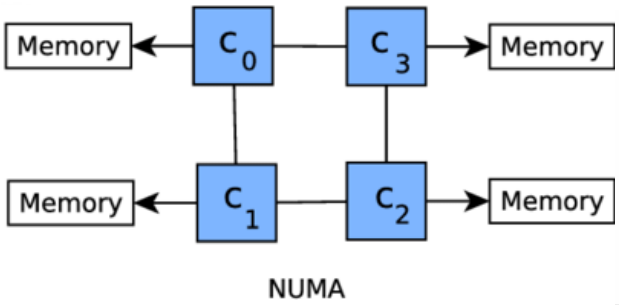


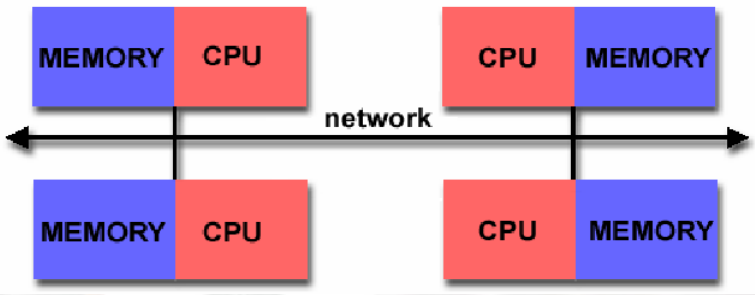
Computer Architecture wrt Memory Arrangement

Shared Memory (UMA): all CPUs equal distance to RAM. Simple, doesn't scale huge.

Shared Memory (NUMA): faster local RAM, slower remote RAM. Bind CPU+memory on

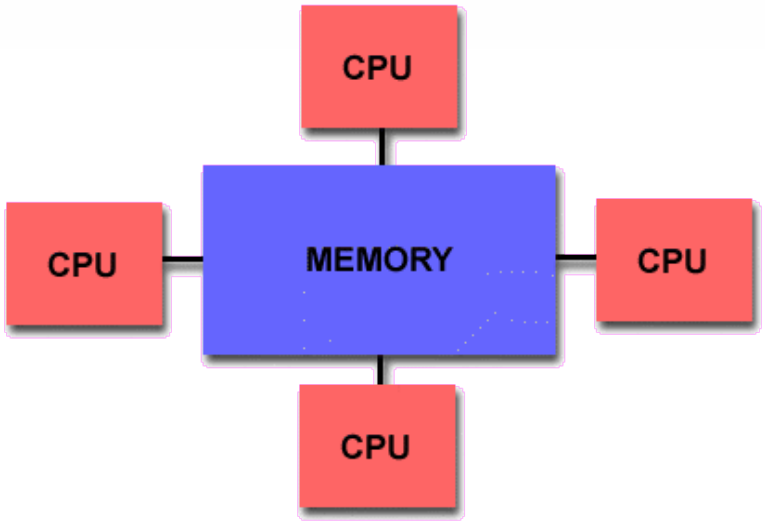






Shared Memory

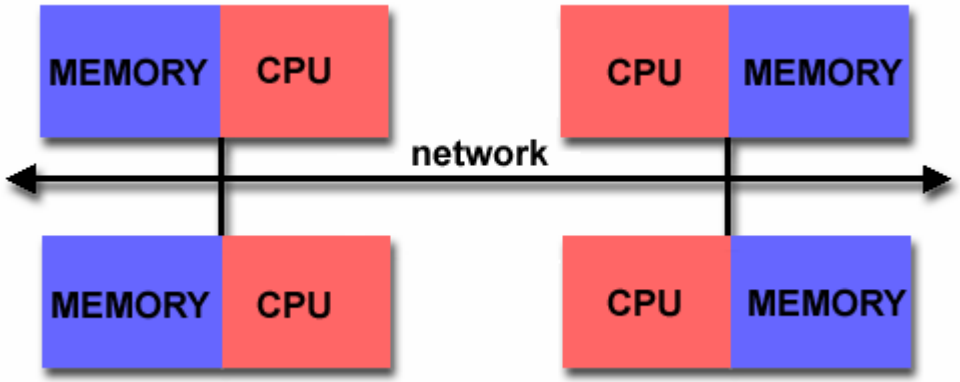
- Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.



- 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***.

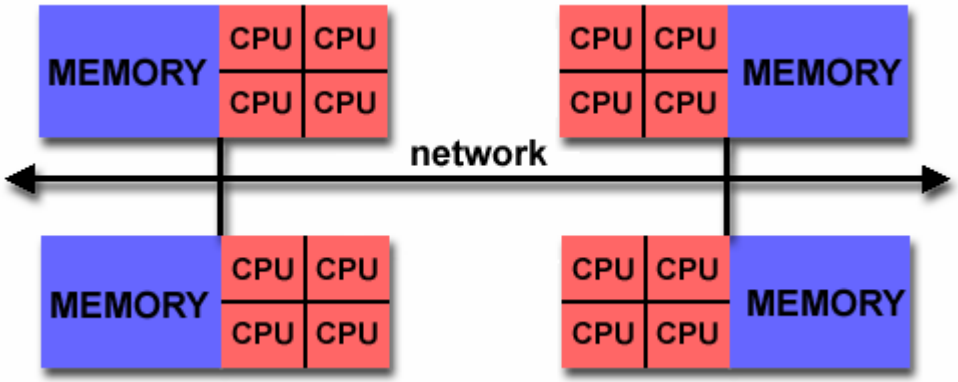
Distributed Memory

- 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.

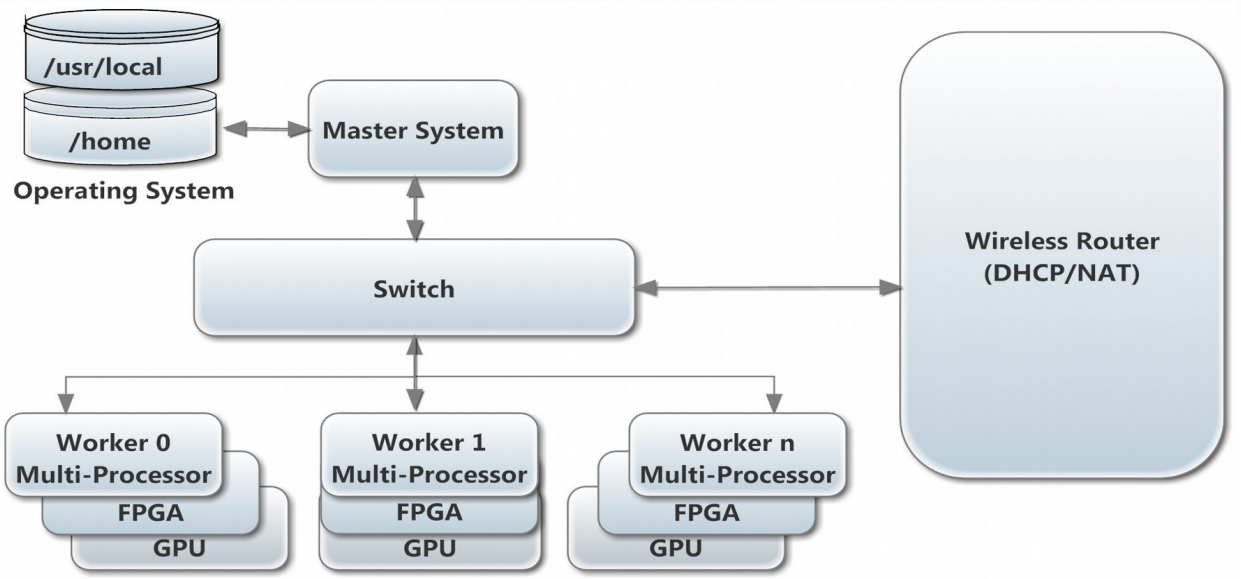
- Importance of HPC
- Types of Processing System
- HPC System Architecture
- **Supercomputing Classes and Infrastructure**
- Cluster Hardware Architecture
- Cluster Software Stacks
- Programming Models
- Demonstration (Supercomputing System)

What is a Supercomputer?

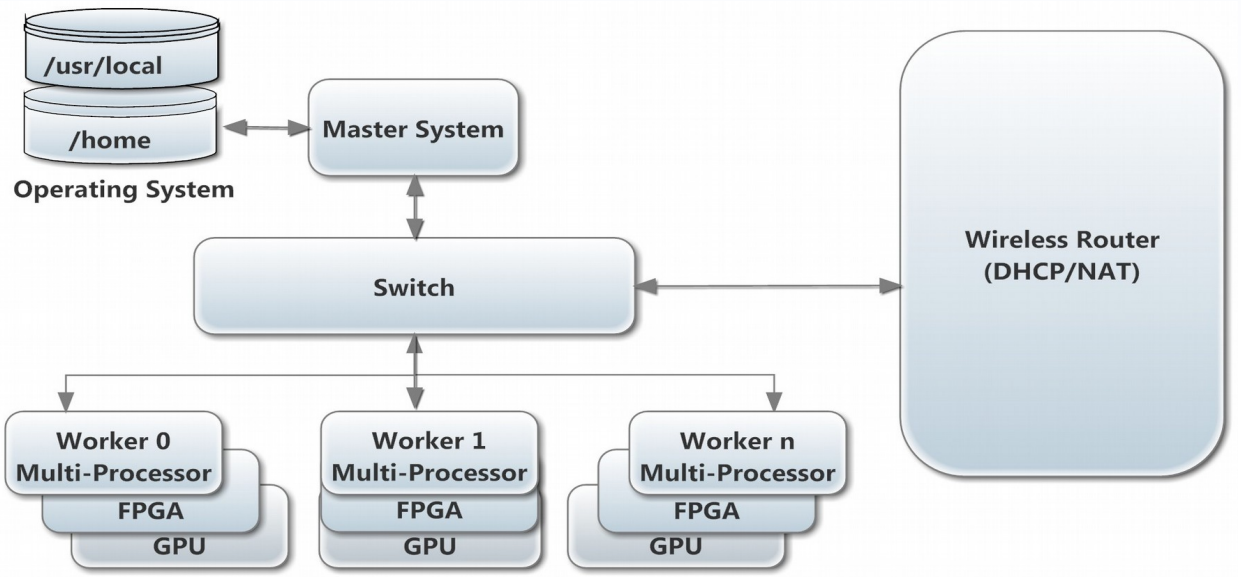
"A supercomputer is a device for turning compute-bound problems into I/O-bound problem" - Seymour Cray

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)

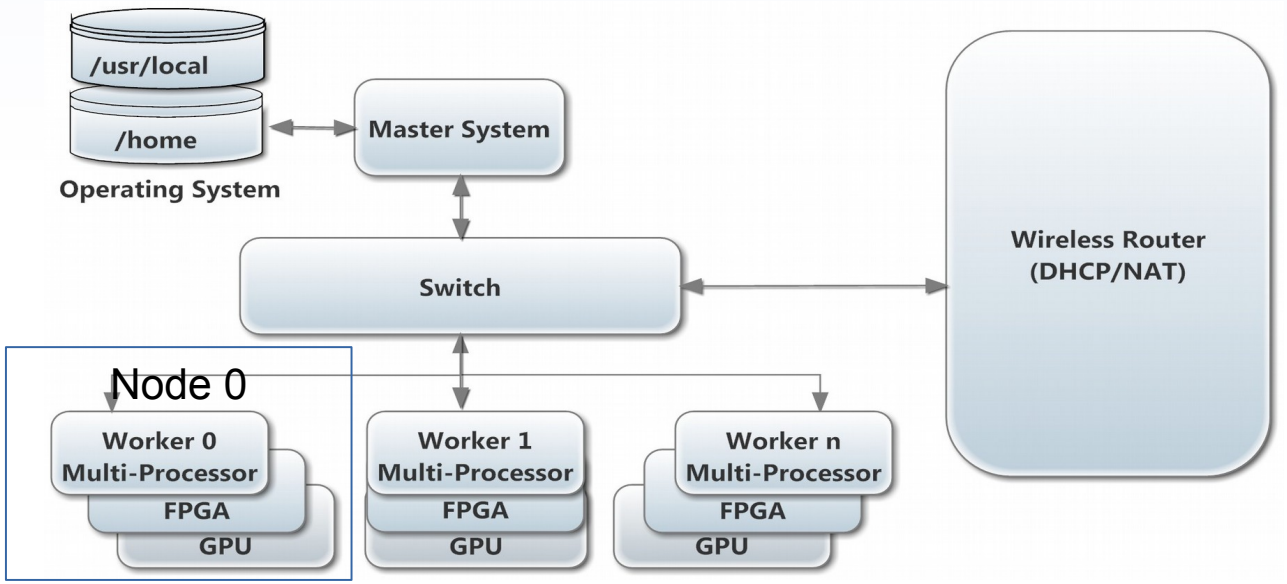
Basic Arch of Supercomputer



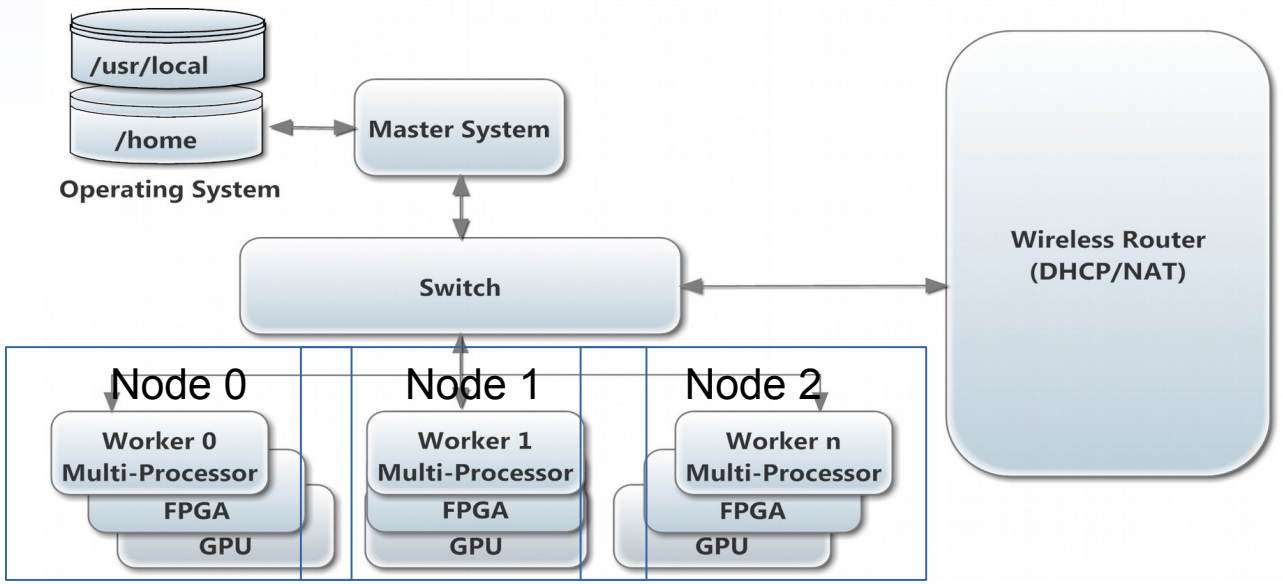
Basic Arch of Supercomputing

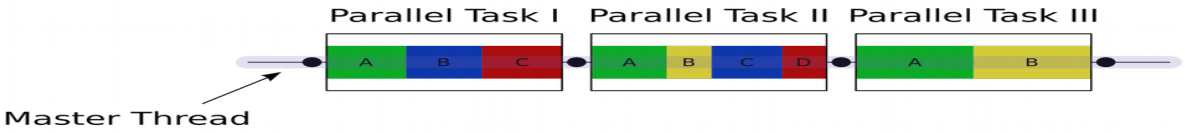


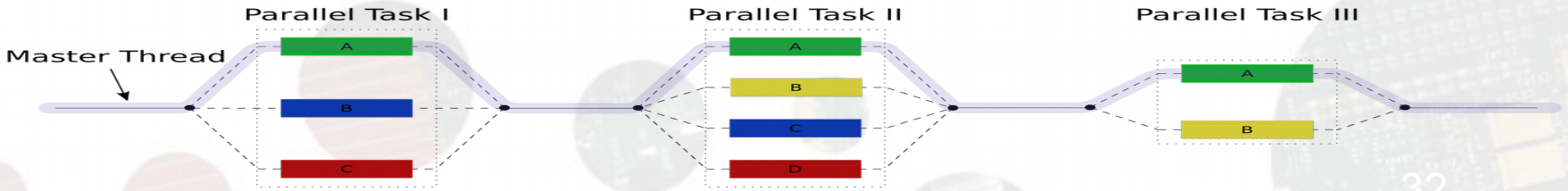
Basic Arch of Supercomputing



Basic Introduction of Supercomputing

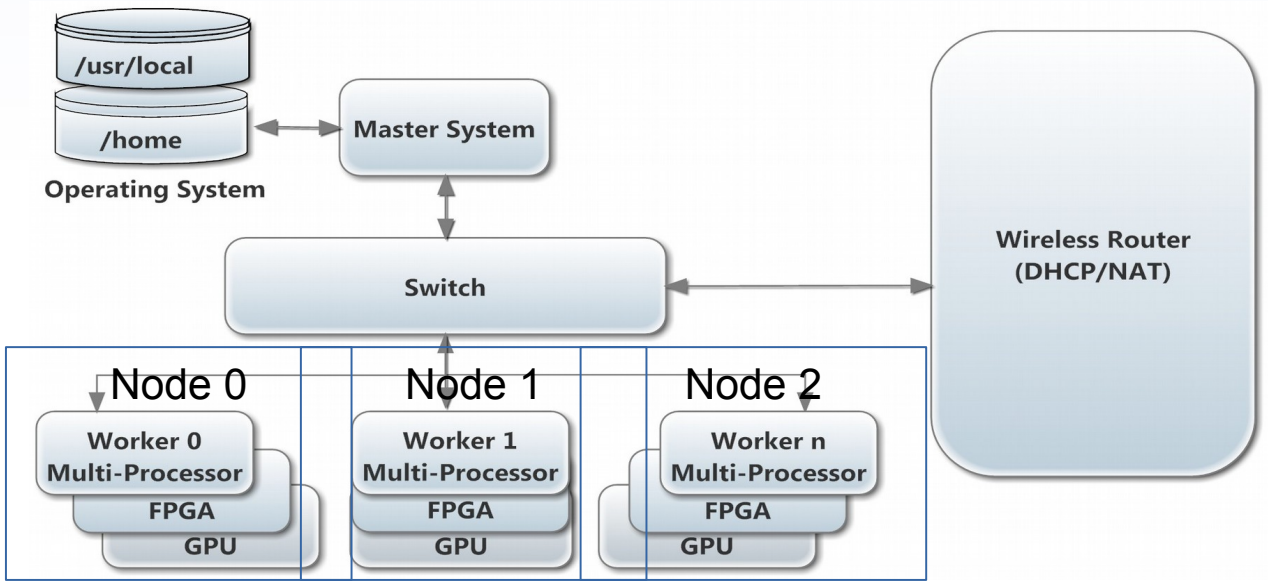


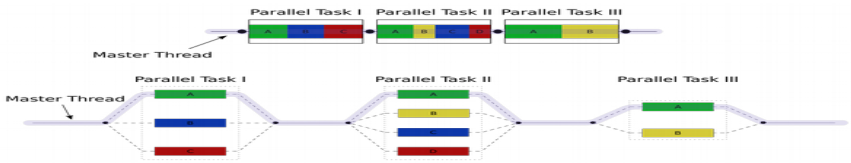




Constituents

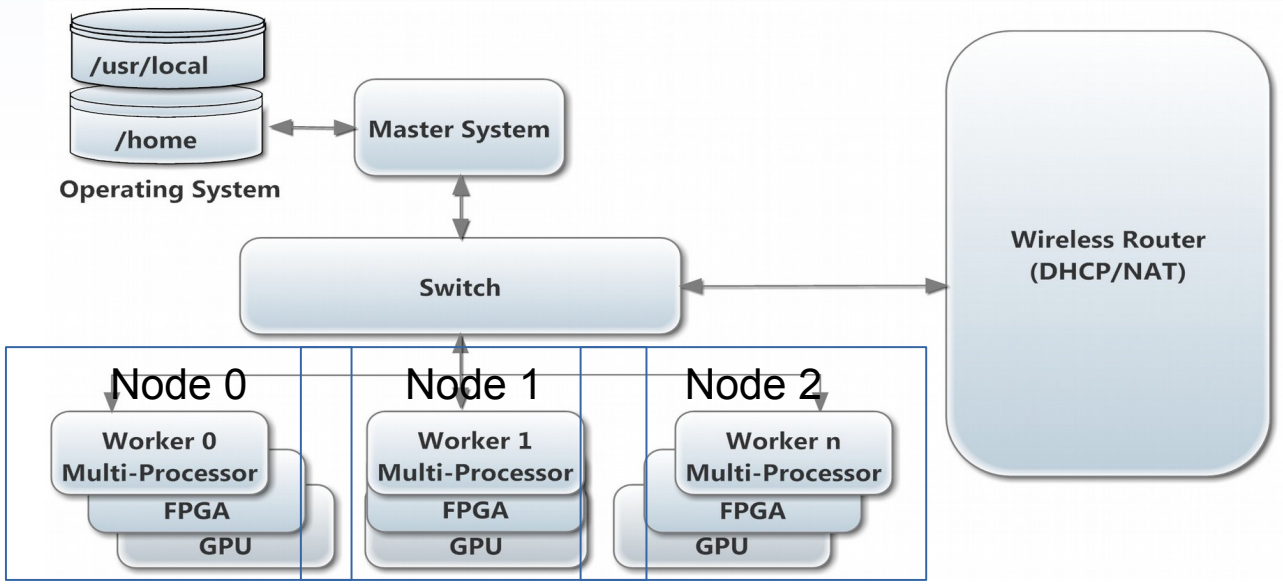
- Processing
- Software
- Programming Models
- Storage
- Network
- Accessibility
- Power

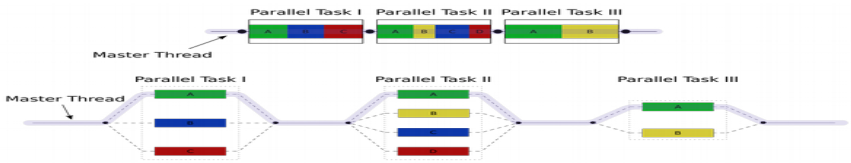




Basic Introduction of Supercomputing

- 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**

Source: Wikipedia

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: 1976 Cray-1 Pipelined Vector Machines:

- 12 pipelined arithmetic units, 160 MFLOPS

Gen III: Shared-Memory multi-processor systems

- (MIMD Arch) 40 TFLOPS

Gen IV: Masive Parallel Processor:

- 10 to thousands of processors

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

Classes of Supercomputers

- **Vector Supercomputers:** Process vectors (arrays of data) in a single instruction. Very fast for scientific workloads involving large matrix and vector operations Limitation: Specialized hardware, expensive, less flexible for general workloads.
- **Tightly Coupled Cluster Supercomputers (Massively Parallel Processors – MPP):** Thousands of processors connected via a high-speed custom interconnect, working in parallel. Extremely high scalability and performance, used in most modern TOP500 systems.
- **Commodity Cluster Supercomputers (Beowulf Clusters):** Built from COTS (Commercial Off-The-Shelf) hardware (standard PCs/servers)

Commodity Cluster

A commodity cluster is a collection of interconnected, independent, and inexpensive computers (COTS – Commercial Off-The-Shelf hardware) that work together as a unified system to solve computational problems.

- Interconnection: Nodes are linked using standard networking technologies such as Ethernet LAN, Myrinet, or QsNet/ELAN.
- Programming: Computation is carried out using widely adopted parallel programming

Performance Benchmarks

Performance benchmark refers to a standardized set of tests used to measure and evaluate how efficiently and effectively the system performs under specific workloads or tasks.

What It Measures:

CPU performance (e.g., FLOPS – floating-point operations per second)

Memory bandwidth and latency

Disk I/O throughput

Network latency and bandwidth (for cluster

Performance Benchmarks

CPU Performance Test: To check the CPU's performance, we used a tool called **sysbench**. The test runs the CPU with multiple tasks to see how it handles them.

sysbench cpu --threads=4 --time=60 run

Memory Performance Test: The memory test checks how well the system's memory works. **sysbench** is again used to check how fast the memory can perform read and write operations.

sysbench memory --threads=4 --time=60 run

Disk I/O Performance Test: Disk I/O (Input/Output) is how fast the system reads and writes data to the storage (disk). To test this, we used “fio”, a tool that checks random read/write speeds and input/output operations per second (IOPS).

fio --name=randreadwrite --ioengine=libaio --rw=randwrite --bs=4k --numjobs=1 --size=10G --time_based --runtime=60m –output=fio_results.log

Network Performance Test: Network performance checks how well the system can send and receive data over the network. We used “iperf”, a tool that measures network bandwidth (how much data can be sent in a given time), latency (how long it takes for data to travel), and packet loss.

Server (Master Node): iperf3 -s

Client (Worker Node): iperf3 -c <server_ip>

GPU Stress Test: To test the GPU (Graphics Processing Unit), we used a tool called “GpuBurn”. This test checks how well the GPU works under heavy use.

./GpuBurn -s 60

System Stress Test: The system was tested using “stress-ng” to simulate high workloads on the CPU, memory, and I/O systems. This test helps check if the system can handle heavy load without breaking down.

stress-ng --cpu 4 --io 2 --vm 2 --vm-bytes 1G --timeout 60s

BSC: SUpercomputer

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

Namal Supercomputer

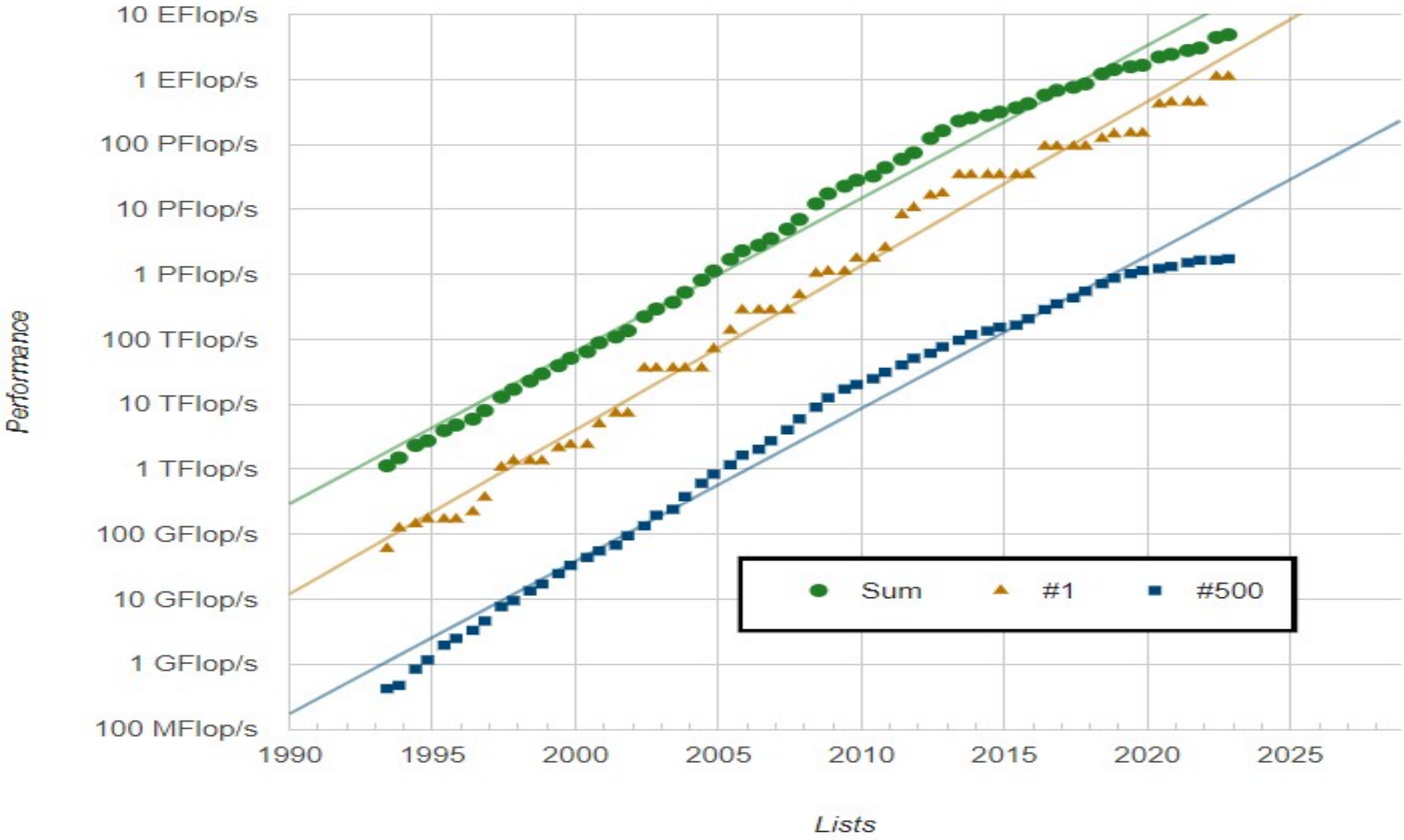
20 compute nodes, each of them:

- 2 × Intel Xeon E5-2673 v4 @ 2.30 GHz (20 cores/socket, 2 threads/core → 80 threads per node)

80 × 2.30 G FLOPS = 184 G FLOPS

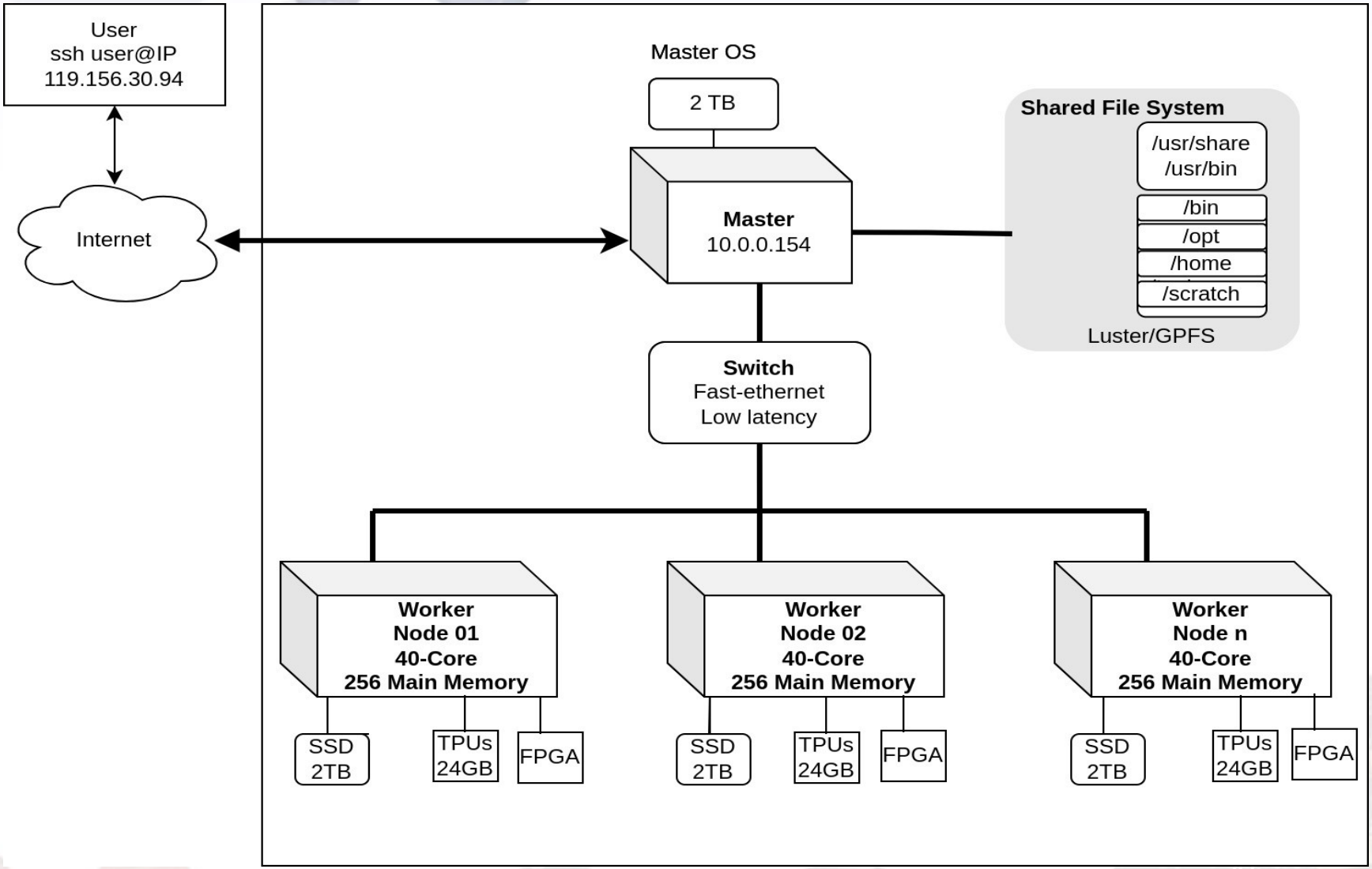
- 256 GB of main memory
- 1 × 4 TB SSD as local storage
- 2 × GPU NVIDIA GeForce RTX 4070 Ti (12 GB)

Supercomputers Performance



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

Supercomputer Architecture



Cluster Hardware Components

- **Login/Access**
- **Compute Node**
- **Storage and File System**
- **Network Communication**
- Management Nodes
- Service Nodes

Compute Node

Processing System:

CPU: Dual Intel Xeon, high core count, strong FP64/FP32 performance.

GPU: Dual NVIDIA RTX, massive parallel FP32 for AI/ML/simulation.

FPGA: Tang Mega for reconfigurable, task-specific acceleration.

I/O & Storage: PCIe Gen 3.0, NVMe SSD, high-speed networking.

Shared Memory:

Large-capacity DDR4 ECC memory per node

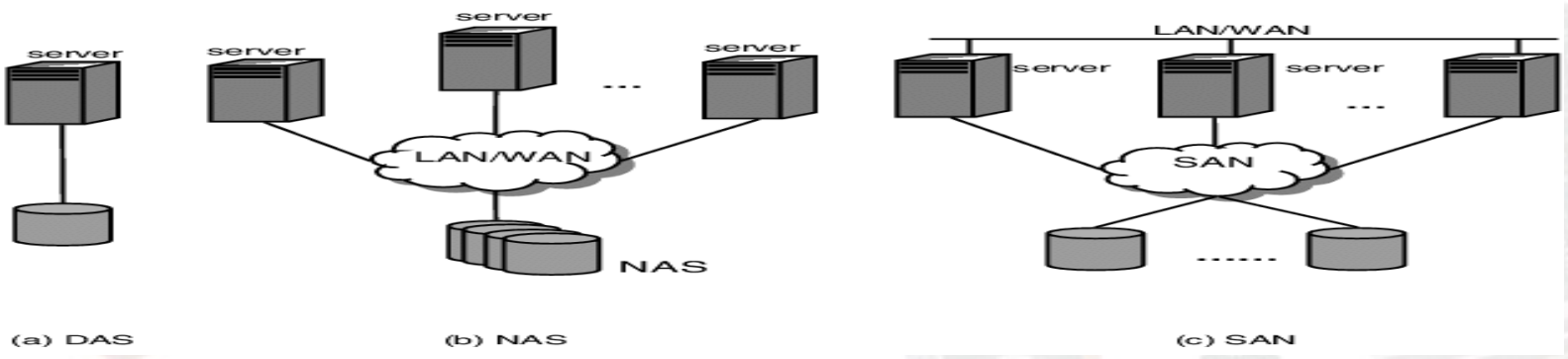
High bandwidth for fast data access

Supports large-scale, memory-intensive workloads

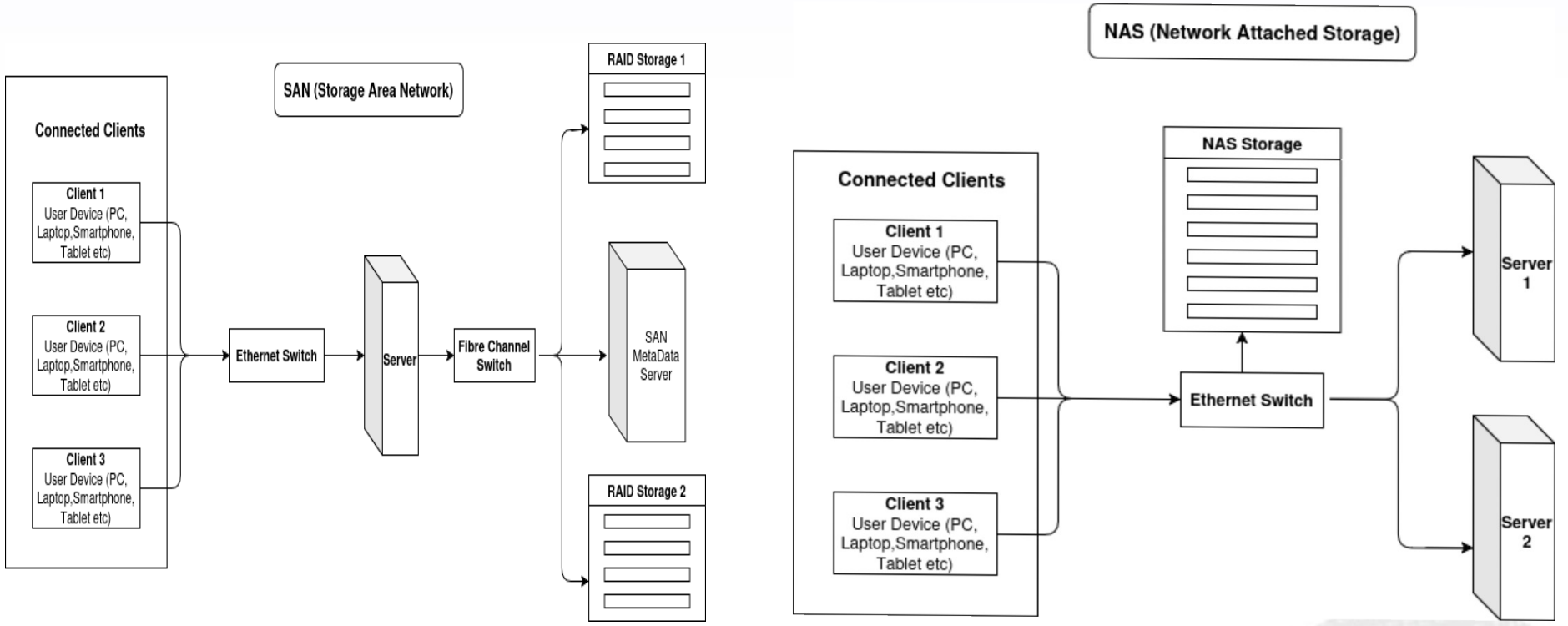
ECC technology for reliable, error-free computation

Storage and File System

- Storage: SSD
  - DAS (Direct Attached Storage)
  - NAS (Network Attached Storage): A file-level storage device connected to a standard network, accessible like a shared folder.
  - SAN (Storage Area Network): A high-speed dedicated network that provides block-level storage access to servers.
    - Parallel File System Storage
    - Ceph : Object, Blocks, File Storage



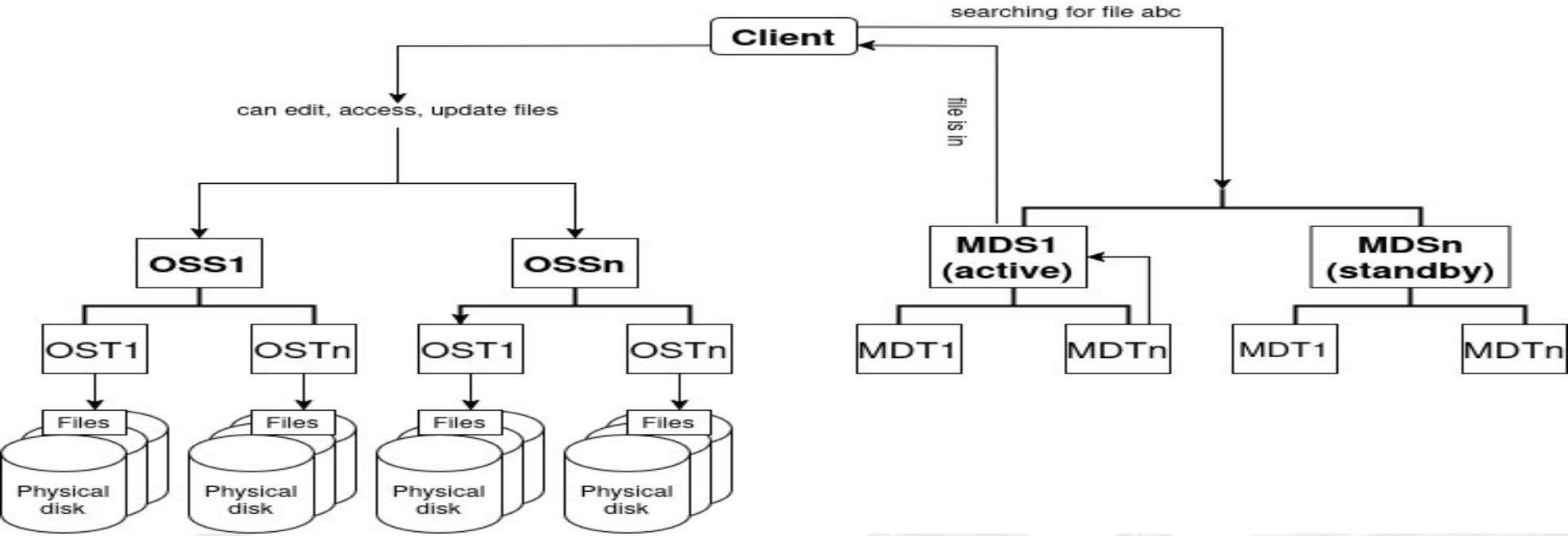
SAN & NAS (Diagrams Added)



Lustre parallel file system architecture

Lustre splits file metadata (handled by MDS) from actual file data (handled by OSS/OST), enabling high-performance, parallel access to data for HPC clusters.

- MDS (Metadata Server) → Manages metadata (file names, directories, permissions).
- OSS (Object Storage Server) → Manages actual file data.



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

The network communication fabric is one of the most critical hardware components in a commodity cluster, because it directly impacts latency (communication delay) and bandwidth (data transfer rate).

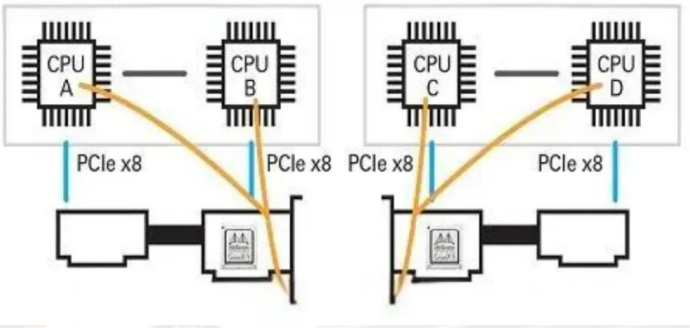
Fast Ethernet (100M – 10Gbps)

- Low cost (commodity switches and NICs), Easy to deploy and High Latency

RoCE (RDMA over Converged Ethernet)

InfiniBand (200 Gbps)

- A high-performance network interco built for HPC 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.

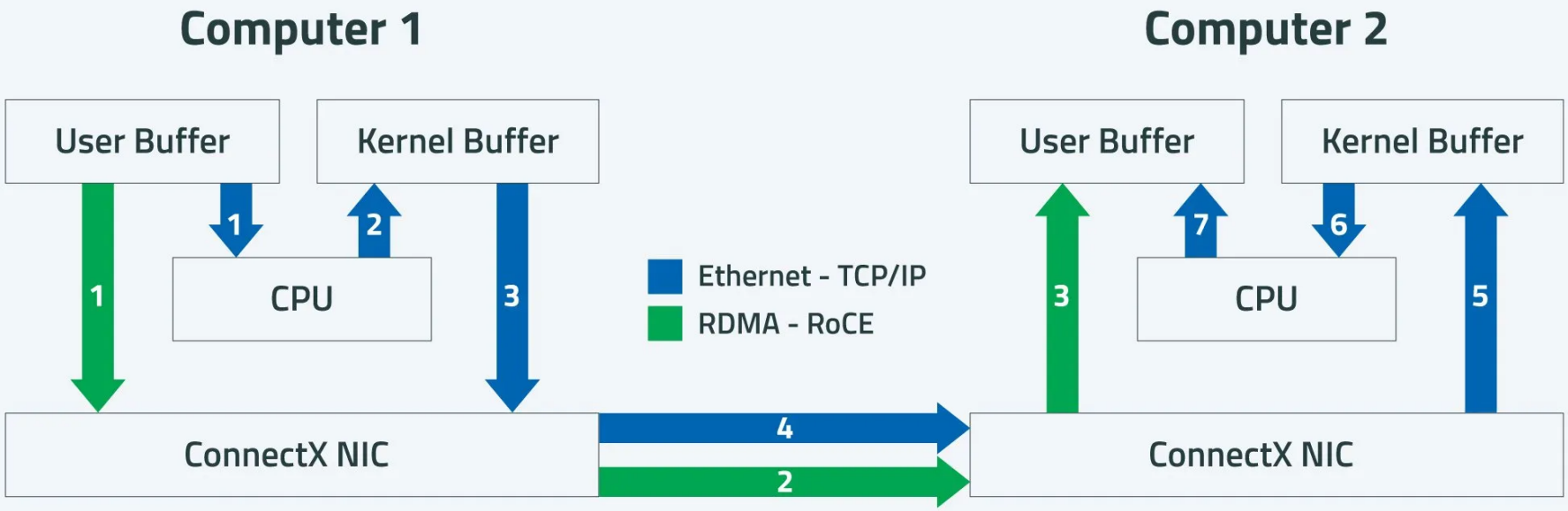
It executes **data offloading** and **data transfer tasks** from the host to dedicated network adapters that support **RDMA**, such as **InfiniBand or RoCE (RDMA over Converged Ethernet)**.

This bypasses the need for the usual

Fast Ethernet & RoCE

RoCE (RDMA over Converged Ethernet)

- RoCE is a high-performance networking protocol that enables Remote Direct Memory Access (RDMA) over Ethernet networks, allowing low-latency, high-throughput data transfer.
- RoCE is ideal for accelerating data transfer within Ethernet-based data centers and HPC clusters, especially when minimizing CPU overhead is important.

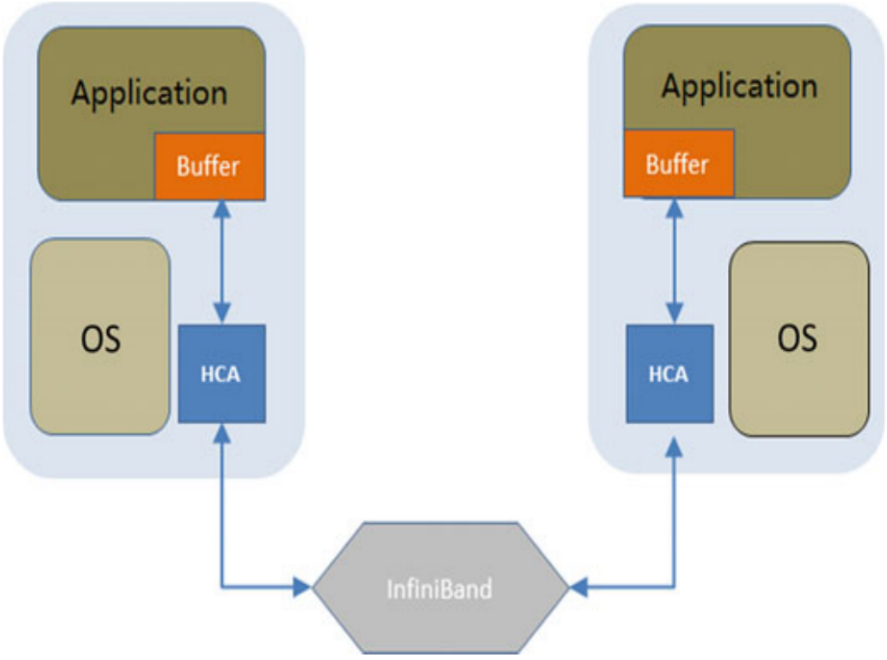


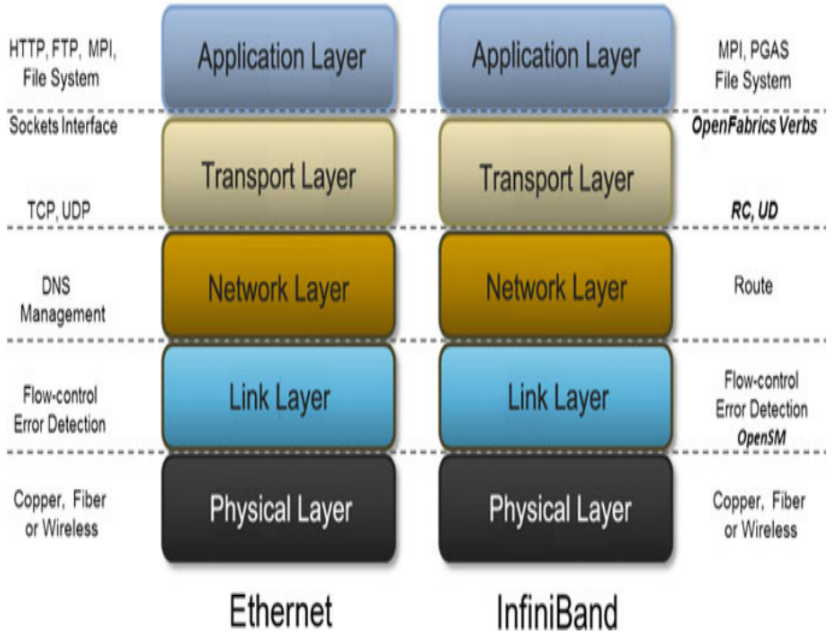
InfiniBand

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

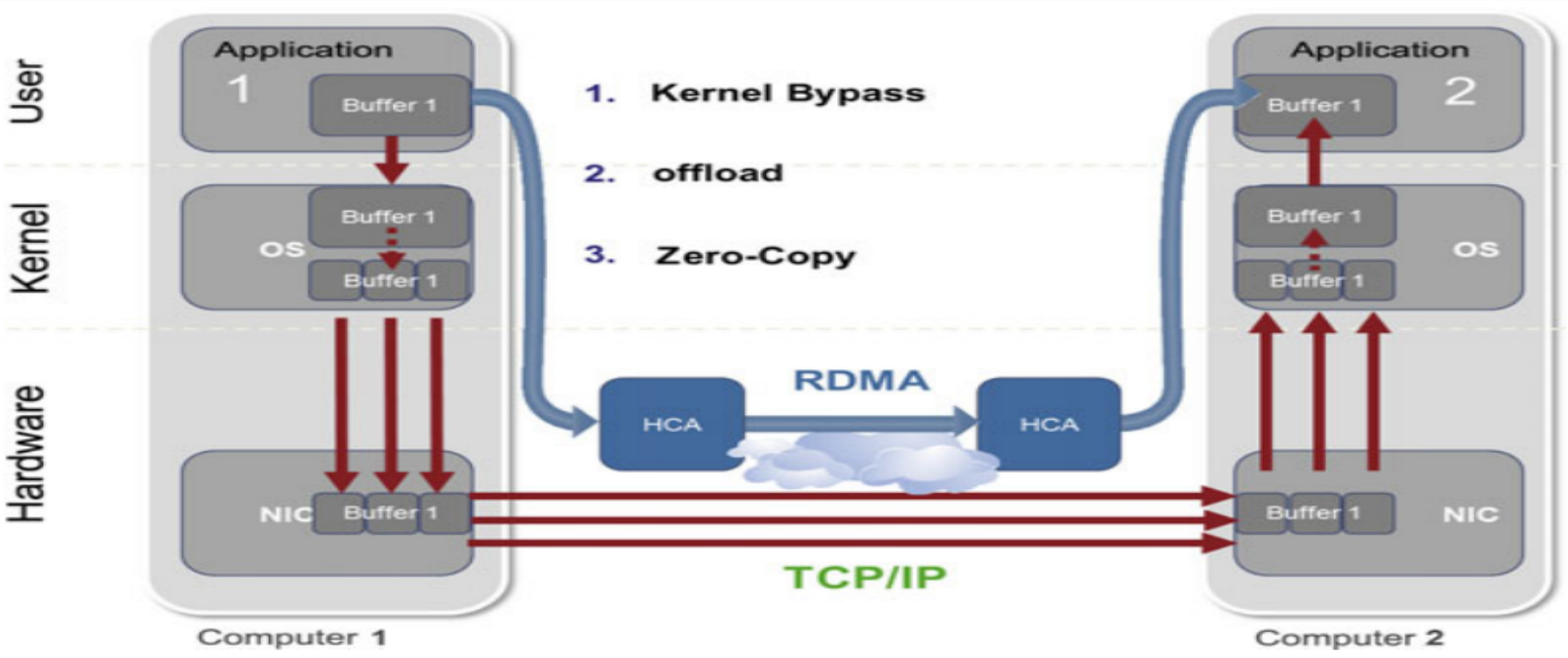
InfiniBand Communication Model: RDMA (Remote Direct Memory Access) is natively supported by InfiniBand by default.

A node can directly read/write into the memory of another node without involving the remote CPU or OS kernel.

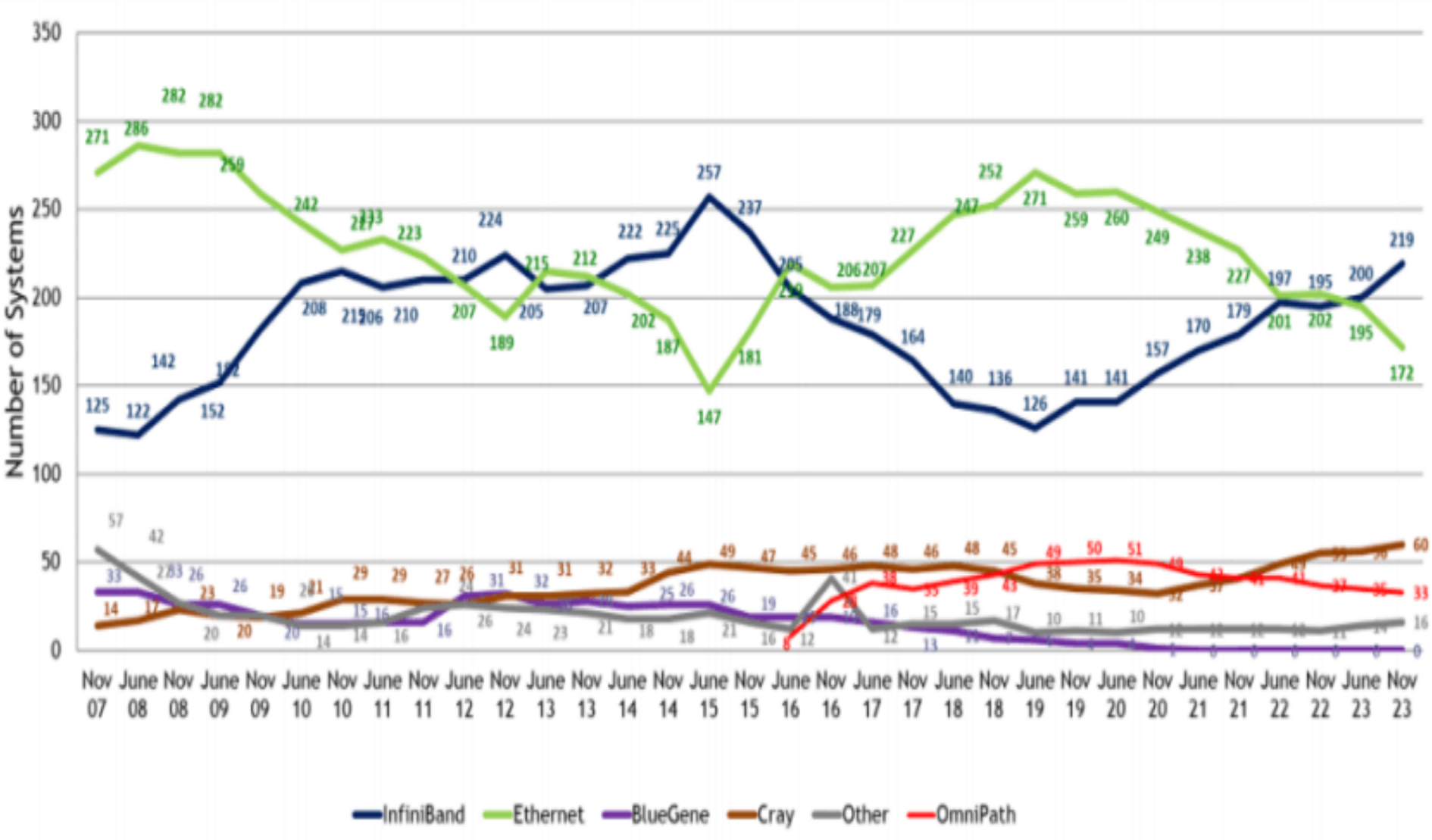




TCP/IP and Remote Direct Memory Access



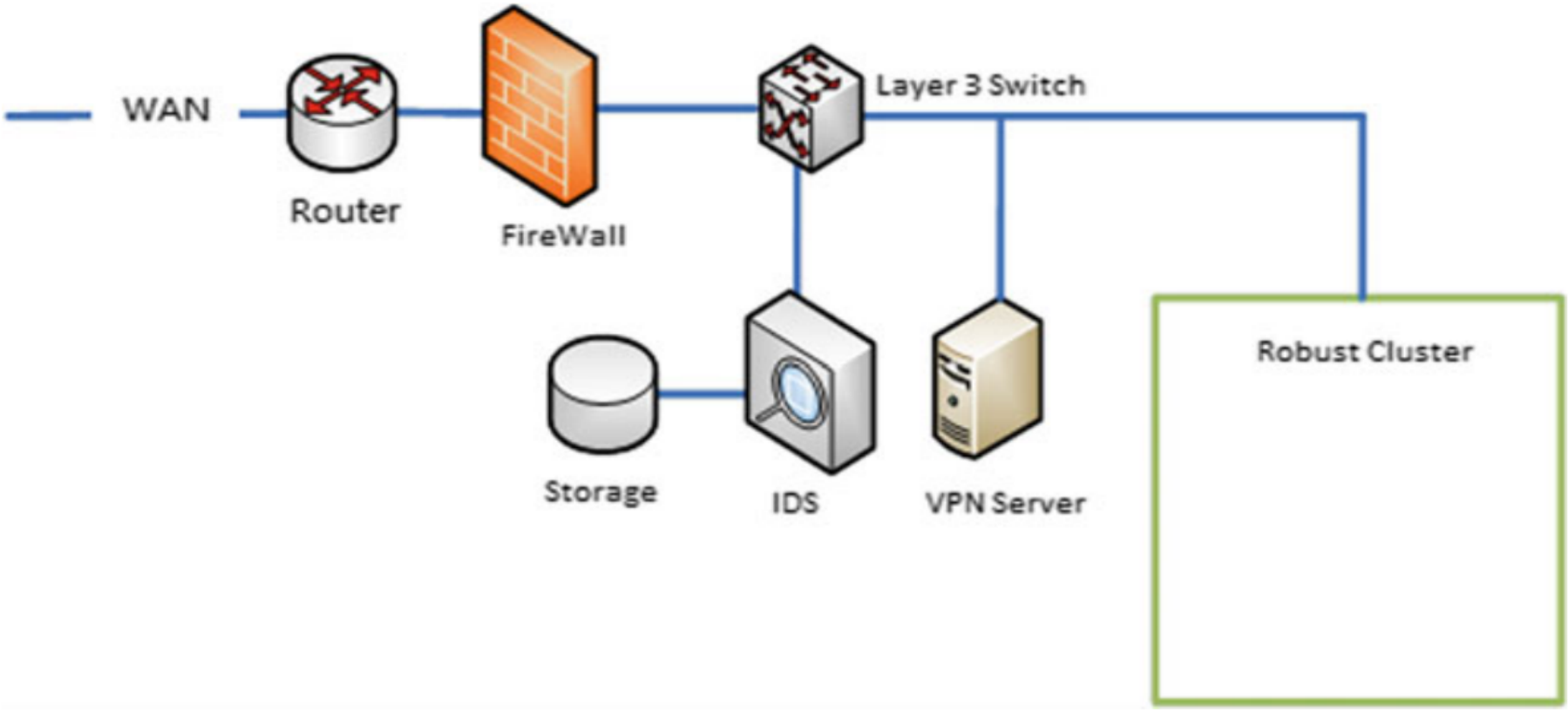
InfiniBand and Ethernet In HPC



Network Security

Network Security refers to a set of rules and configurations designed to protect the network infrastructure from:

Unauthorized access, misuse, modification, destruction and improper disclosure.



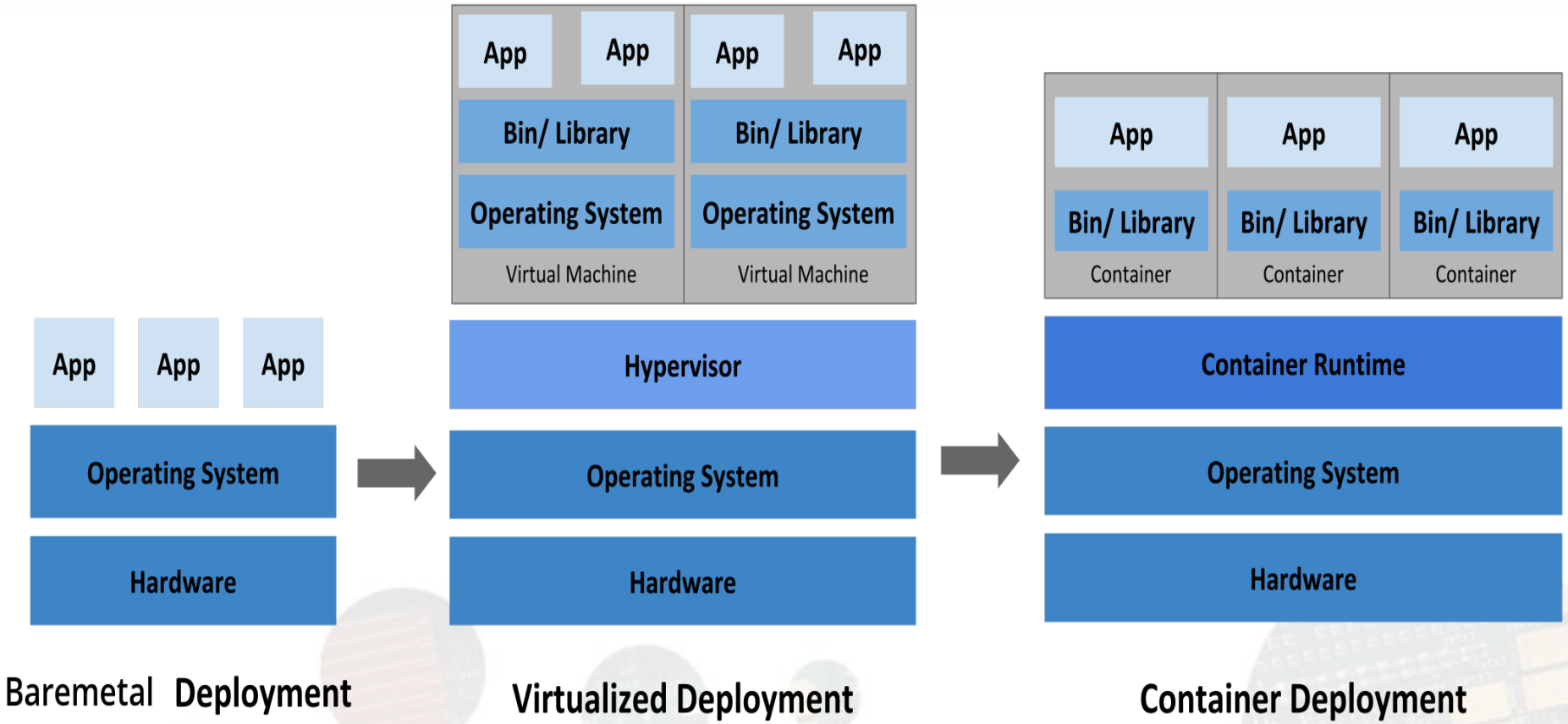
Soft Firewall

Soft firewalls are software-based solutions that control and monitor incoming/outgoing traffic on a system or network to enhance security. Soft firewalls include Zentyal, pfSense, IPFire, SmoothWall, ClearOS, and Iptables.

Soft Firewall	Description
Zentyal	Linux server management tool with integrated firewall and domain services.
pfSense	Powerful, FreeBSD-based firewall/router software with web GUI.
IPFire	Hardened, open-source Linux firewall focused on performance and modularity.
SMoothWall	User-friendly firewall with GUI, ideal for small networks.
ClearOS	Enterprise-level firewall with multiple network security features.
Iptables	CLI-based Linux firewall tool for creating custom traffic rules.

- 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



Baremetal 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

HPC Software Stack

Network:

- Drivers and Libraries: RDMA / RoCE drivers (mlx4, mlx5, rxe), InfiniBand drivers, libibverbs, rdma-core.
- MPI Libraries: OpenMPI, MPICH, Intel MPI → provide message-passing interface for parallel programs.

Storage / Disk

- Parallel File Systems:
- Lustre – widely used in large supercomputers

Development & Runtime Environment

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

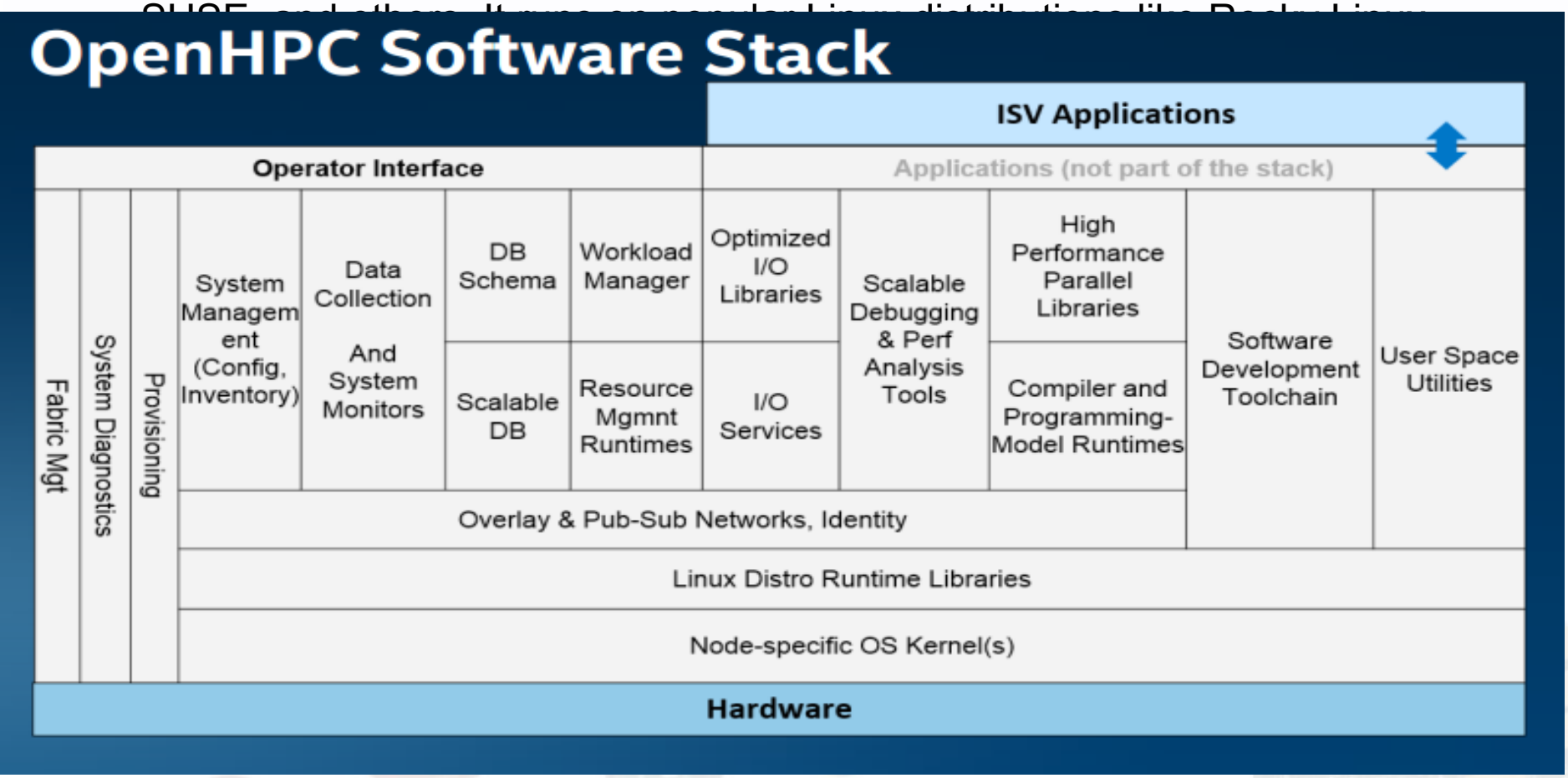
Monitoring & Management

Cluster Monitoring: Ganglia, Nagios, Prometheus + Grafana.

Configuration Management: Ansible, xCAT, Bright Cluster Manager.

OpenHPC

OpenHPC stands for Open High Performance Computing. It offers a modular stack of tools, libraries, and services to make building and managing HPC systems easier. It's community-driven, supported by organizations like Intel, SUSE, and others. It runs on popular Linux distributions like Rocky Linux



OpenHPC Platform

SLURM (Scheduler): Schedules and manages jobs across compute nodes.

Lmod / Environment Modules: Handles dynamic software environment setup.

Warewulf / xCAT / Cobbler: Node provisioning and management tools.

Ganglia / Prometheus / Grafana: Monitoring and visualization of cluster performance.

OpenMPI / MPICH: Libraries for distributed parallel computing (MPI).

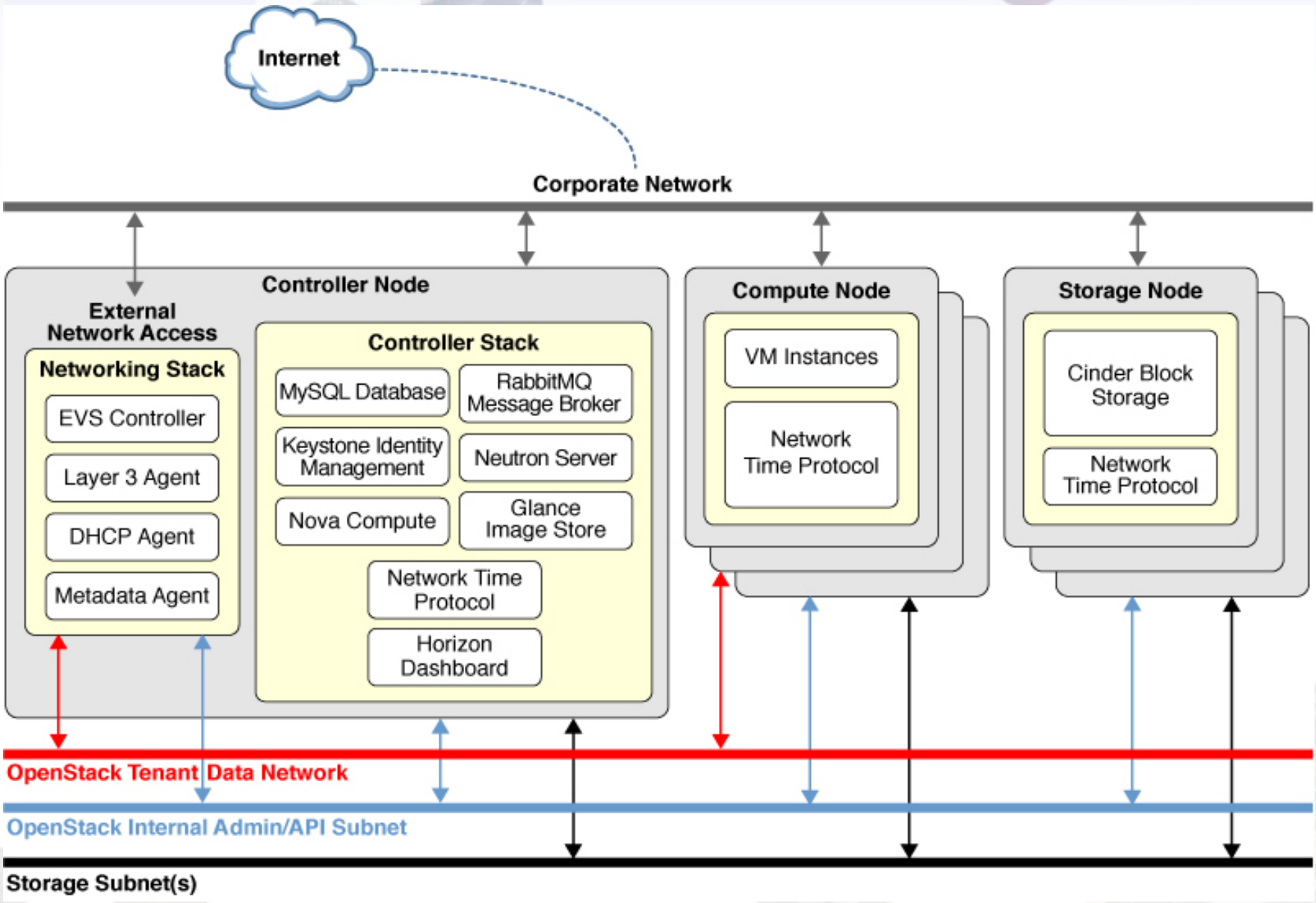
Compilers (GCC, Intel, NVIDIA): Compilers optimized for HPC Development.

Storage and File Systems (Lustre, NFS, BeeGFS): High Performance parallel and shared storage systems.

I/O Libraries and Development Tools: Scientific data management for parallel I/O, Debugging and Profiling of HPC applications.

Security and Access (SSH, Firewalls, LDAP): Manages user access and cluster security.

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)

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

Kubernetes

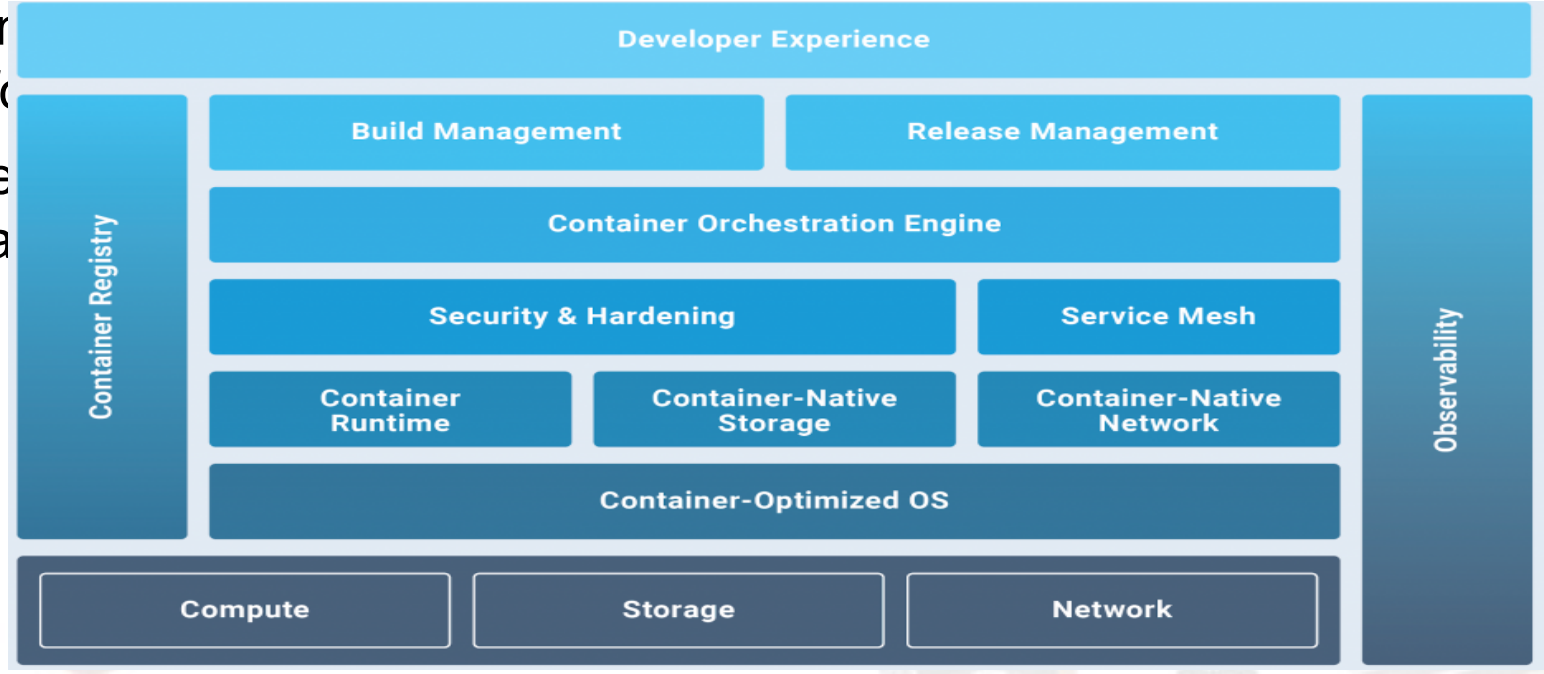
Docker: A platform for developing, shipping, and running applications using containerization.

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.

Pror
f

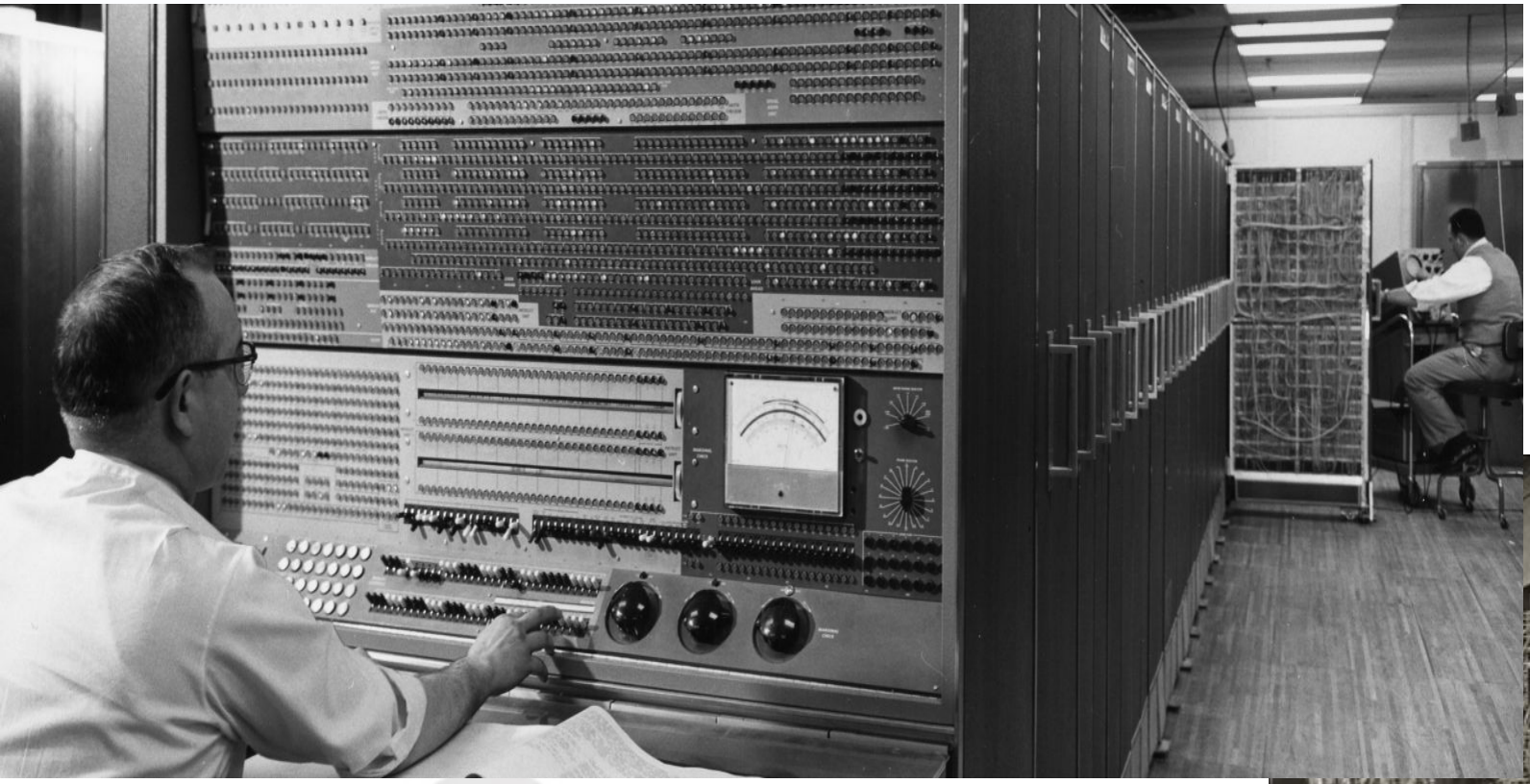
Flue
a



		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]

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

Programming in the Early Mainframe Era



Before keyboards and screens: Early mainframes were programmed by toggling switches and monitoring lights.

Programming HPC

Mainframe Era (1940s–1960s)

- Programming via switches, punched cards, assembly language

Vector & Scientific Supercomputers (1970s–1980s)

- Programming with Fortran, vectorization

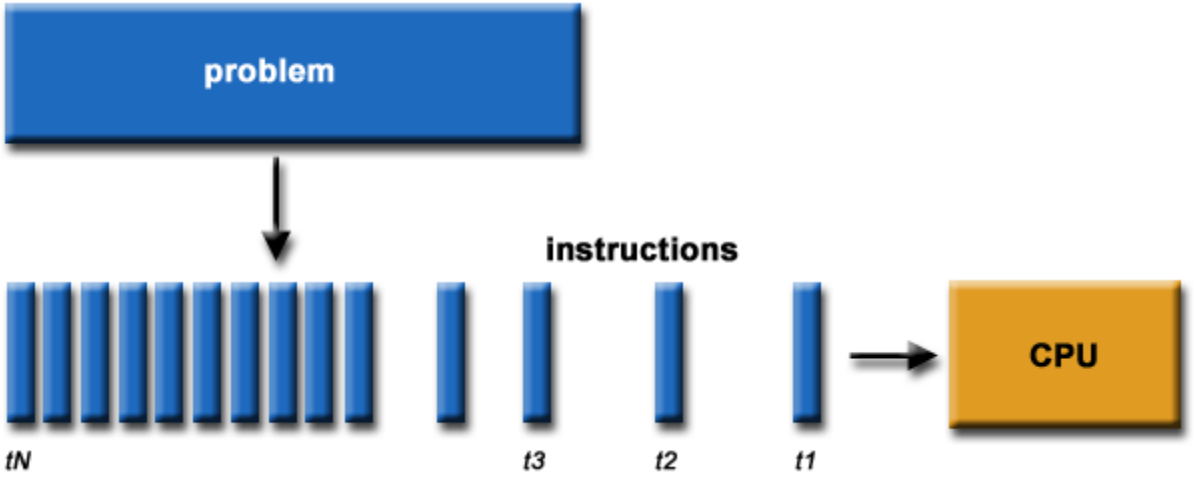
Cluster & Parallel Era (1990s–2000s)

- MPI (Message Passing Interface) and OpenMP became standard

Petascale & Specialized Acceleration

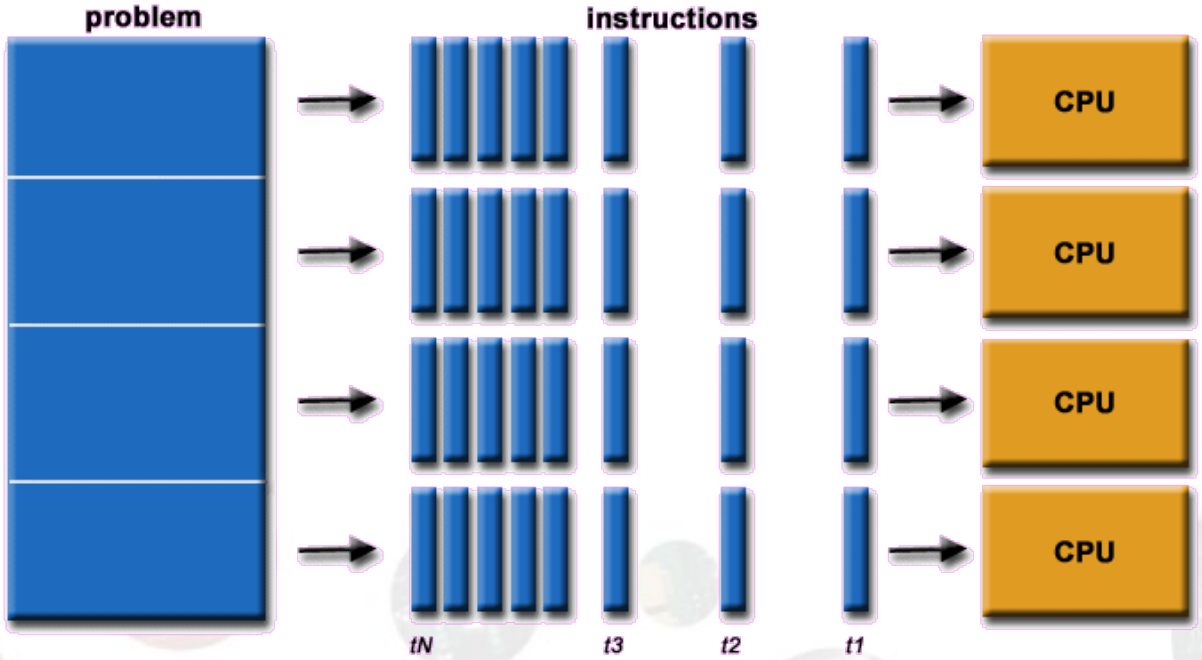
Parallel Programming ?

- 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.



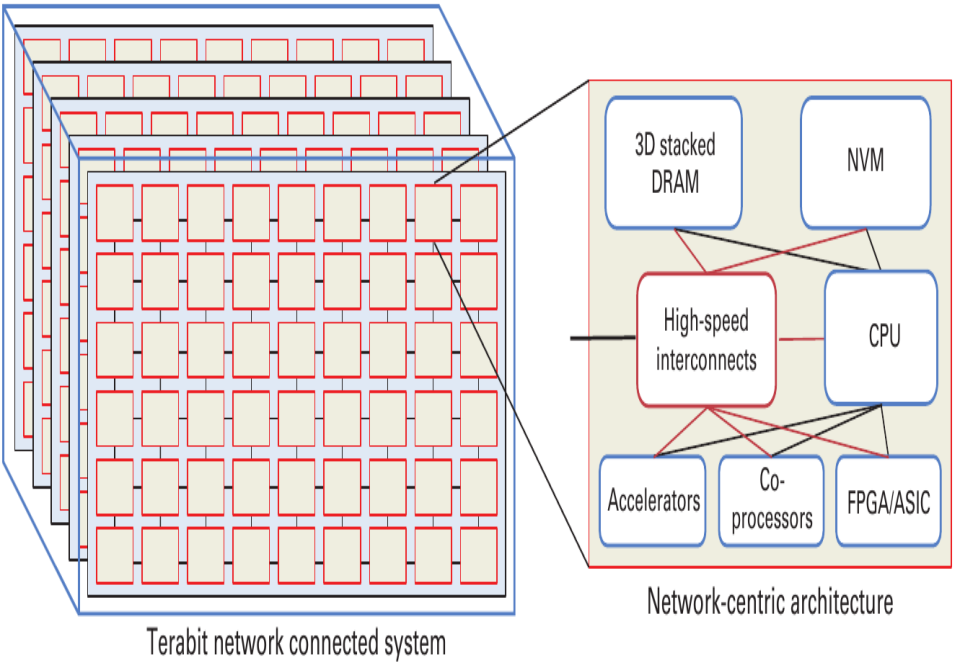
Parallel Computing

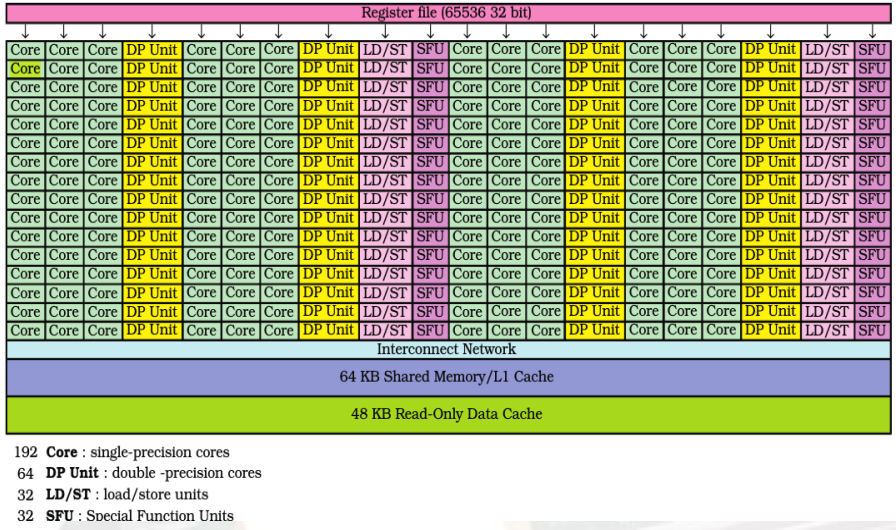
- In the simplest sense, ***parallel computing*** is the simultaneous use of multiple compute resources to solve a computational problem.
  - To be run using multiple CPUs
  - A problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions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





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

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)

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

Distributed AI Programming Models

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

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

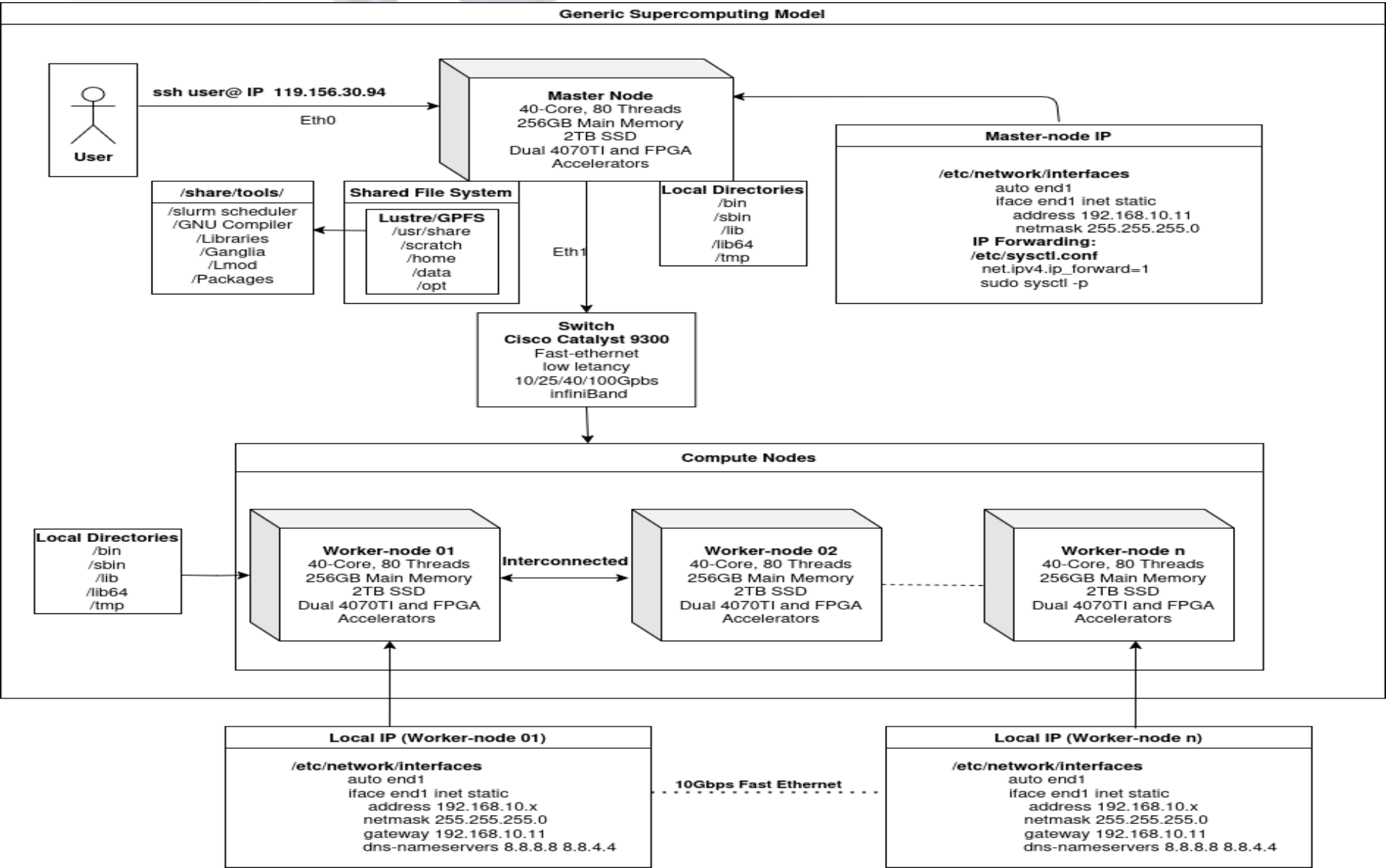
Centre for AI and BigData: PakistanSupercomputing

- **Hardware Architecture**
- Software Configuration and Deployment
- Parallel Programming Model
- Specificaitons and Use

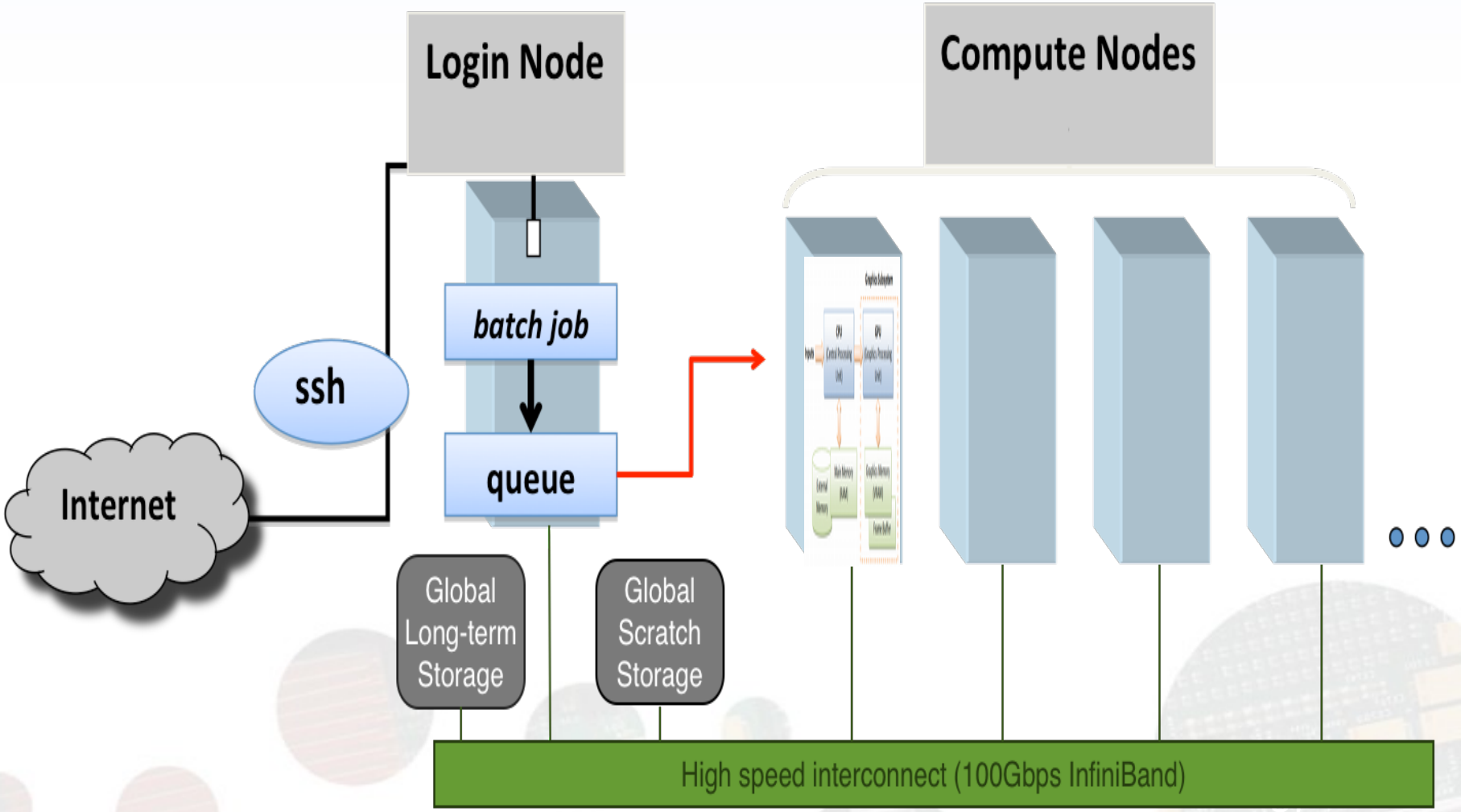
Hardware Architecture

- **Compute Nodes**
- **Networking Infrastructure**
- **Storage Systems**
- **Rack System**

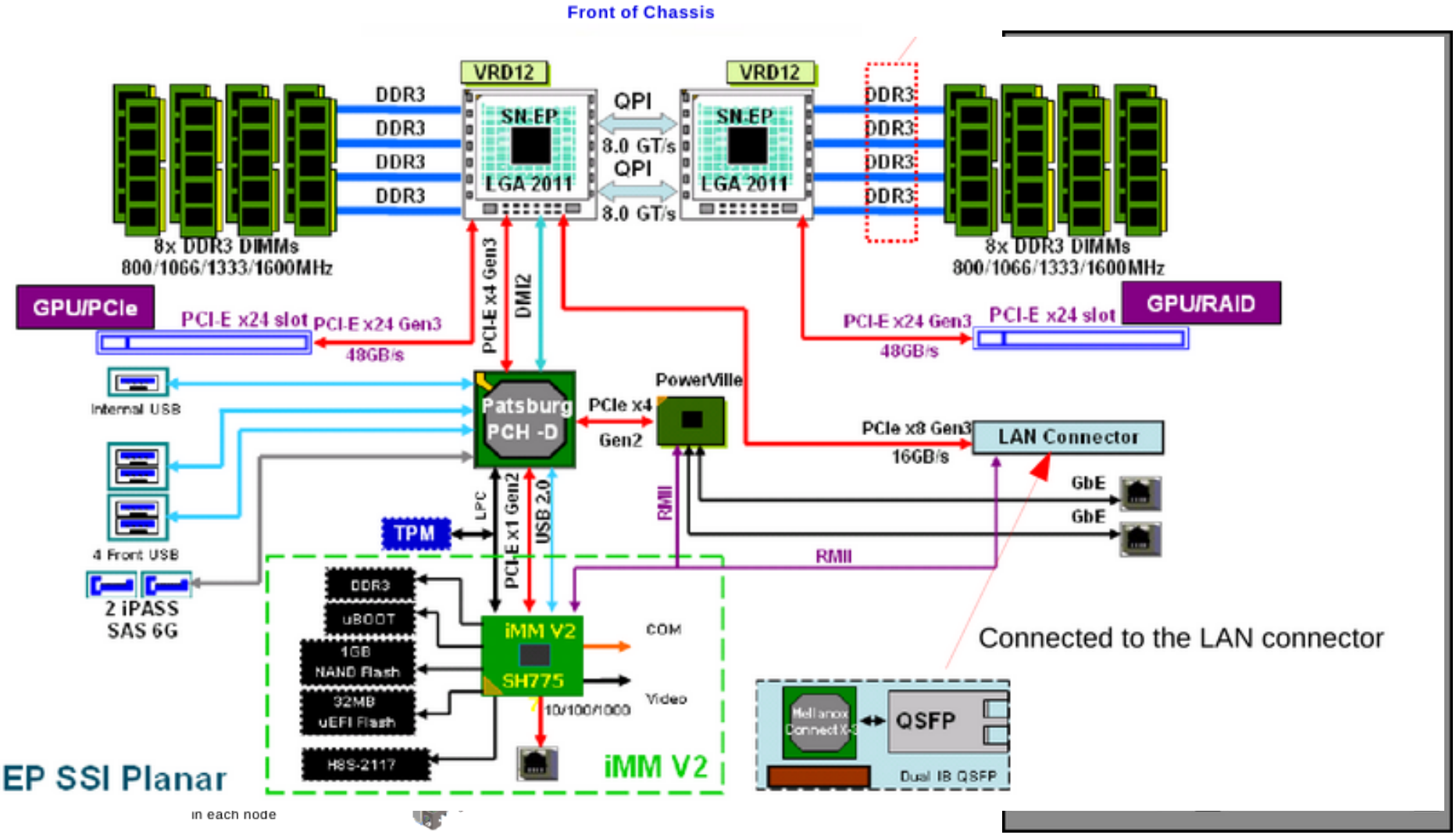
Hardware Architecture



Working Example



Bottom-up Design Approach



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

Hardware Stack...

Processing System:

- CPU: Dual Intel Xeon, high core count, strong FP64/FP32 performance.
- GPU: Dual NVIDIA RTX, massive parallel FP32 for AI/ML/simulation.
- FPGA: Tang Mega for reconfigurable, task-specific acceleration.
- I/O & Storage: PCIe Gen 3.0, NVMe SSD, high-speed networking.

Shared Memory:

- Large-capacity DDR4 ECC memory per node
- High bandwidth for fast data access
- Supports large-scale, memory-intensive workloads
- ECC technology for reliable, error-free computation

Bottom-up Design Approach

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

Hardware Stack...

Storage File System:

- High-speed NVMe SSDs for fast local data access
- Centralized Storage Area Network (SAN) with parallel file system like Lustre, BeeGFS, GLuster etc.
- Supports high-throughput, data-intensive workloads
- RAID configuration for fault tolerance and reliability

Scalable Network Switch:

- High-speed, low-latency network for compute node communication
- Scalable architecture to support growing bandwidth needs
- Optimized for efficient data transfer in HPC

Bottom-up Design Approach

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

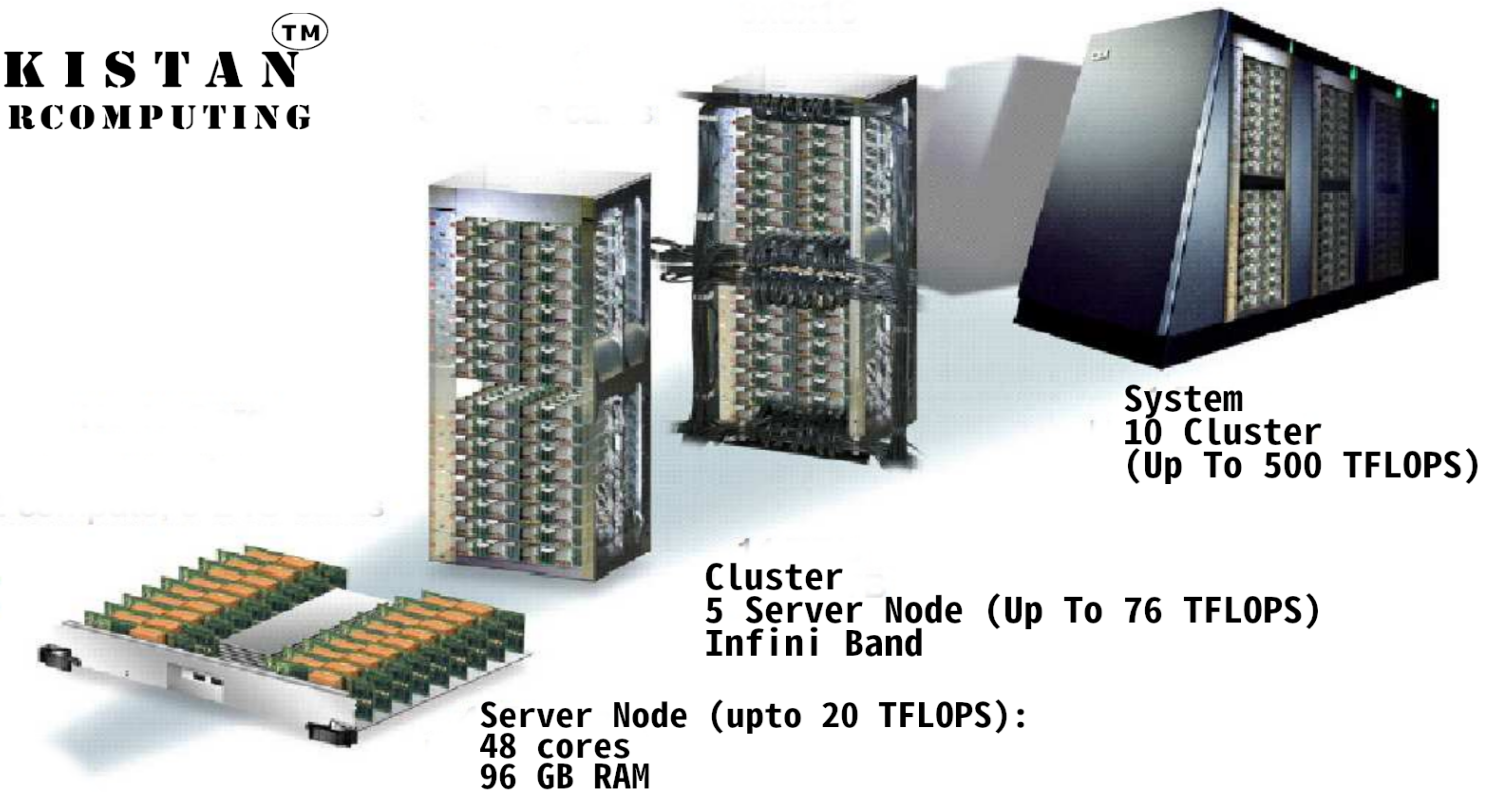
Hardware Stack...

Rack System:

- Optimized for space, thermal efficiency, and reliability
- Active cooling for stable operation under heavy workloads
- Redundant PDUs and UPS for uninterrupted power supply
- Supports high power capacity with fast failover protection
- Smart monitoring and airflow design for efficient temperature control
- Centralized power management for balanced load distribution

Developing Supercomputing for AI

PAKISTAN SUPERCOMPUTING™



System
10 Cluster
(Up To 500 TFLOPS)

Cluster
5 Server Node (Up To 76 TFLOPS)
Infini Band

Server Node (upto 20 TFLOPS):
48 cores
96 GB RAM
1 TB Disk
2 GPUs

CentOS Linux

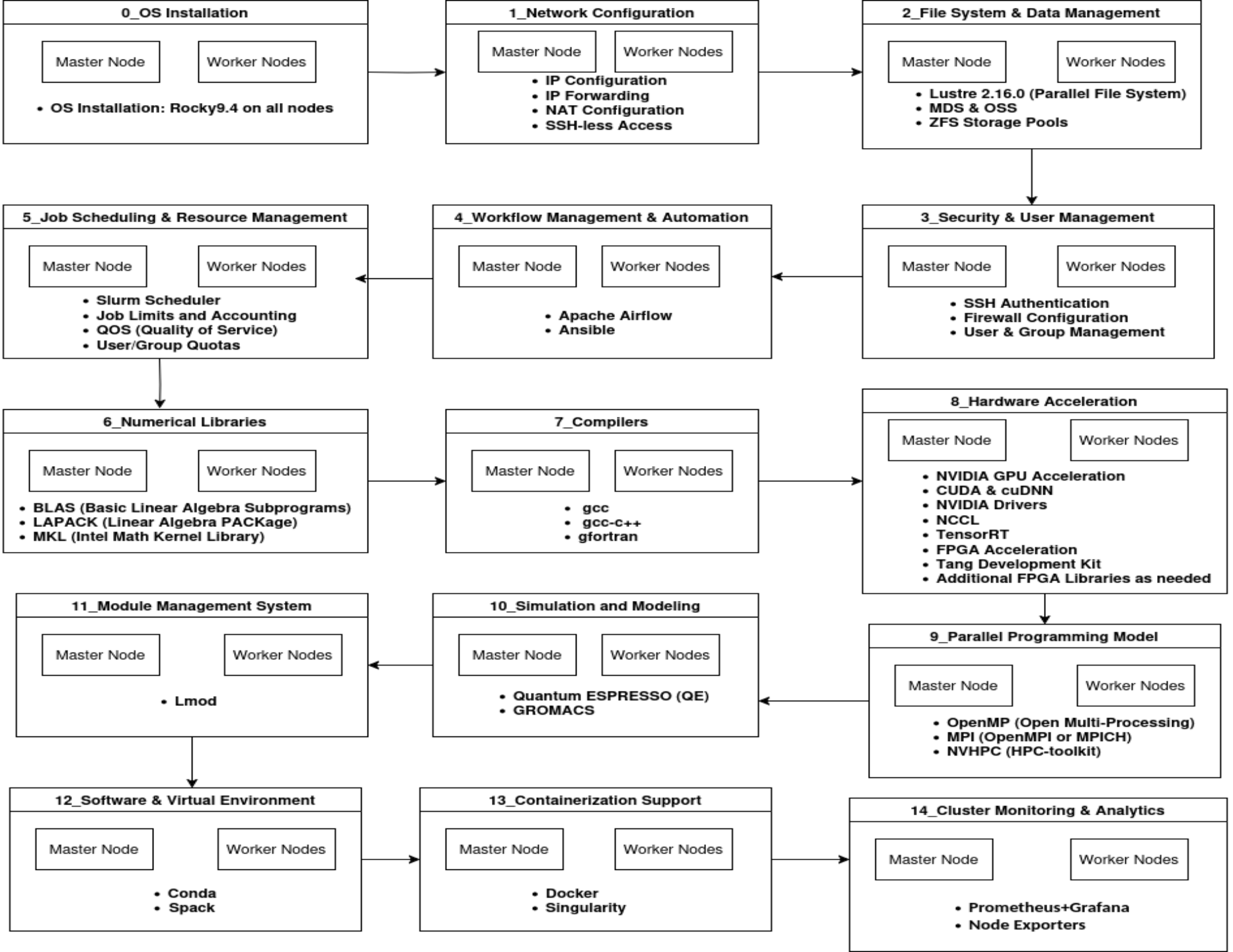
Chip
4 cores

XEON Processor

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Centre for AI and BigData: PakistanSupercomputing

- Hardware Architecture
- **Software Configuration and Deployment**
- Parallel Programming Model
- Specificaitons and Use


0_OS Installation
Master Node
Worker Nodes
• OS Installation: Rocky9.4 on all nodes
1_Network Configuration
Master Node
Worker Nodes
• IP Configuration
• IP Forwarding
• NAT Configuration
• SSH-less Access
2_File System & Data Management
Master Node
Worker Nodes
• Lustre 2.16.0 (Parallel File System)
• MDS & OSS
• ZFS Storage Pools
3_Security & User Management
Master Node
Worker Nodes
• SSH Authentication
• Firewall Configuration
• User & Group Management
4_Workflow Management & Automation
Master Node
Worker Nodes
• Apache Airflow
• Ansible
5_Job Scheduling & Resource Management
Master Node
Worker Nodes
• Slurm Scheduler
• Job Limits and Accounting
• QOS (Quality of Service)
• User/Group Quotas
6_Numerical Libraries
Master Node
Worker Nodes
• BLAS (Basic Linear Algebra Subprograms)
• LAPACK (Linear Algebra PACKage)
• MKL (Intel Math Kernel Library)
7_Compilers
Master Node
Worker Nodes
• gcc
• gcc-c++
• gfortran
8_Hardware Acceleration
Master Node
Worker Nodes
• NVIDIA GPU Acceleration
• CUDA & cuDNN
• NVIDIA Drivers
• NCCL
• TensorRT
• FPGA Acceleration
• Tang Development Kit
• Additional FPGA Libraries as needed
9_Parallel Programming Model
Master Node
Worker Nodes
• OpenMP (Open Multi-Processing)
• MPI (OpenMPI or MPICH)
• NVHPC (HPC-toolkit)
10_Simulation and Modeling
Master Node
Worker Nodes
• Quantum ESPRESSO (QE)
• GROMACS
11_Module Management System
Master Node
Worker Nodes
• Lmod
12_Software & Virtual Environment
Master Node
Worker Nodes
• Conda
• Spack
13_Containerization Support
Master Node
Worker Nodes
• Docker
• Singularity
14_Cluster Monitoring & Analytics
Master Node
Worker Nodes
• Prometheus+Grafana
• Node Exporters

Software Stack

1) Operating system & base

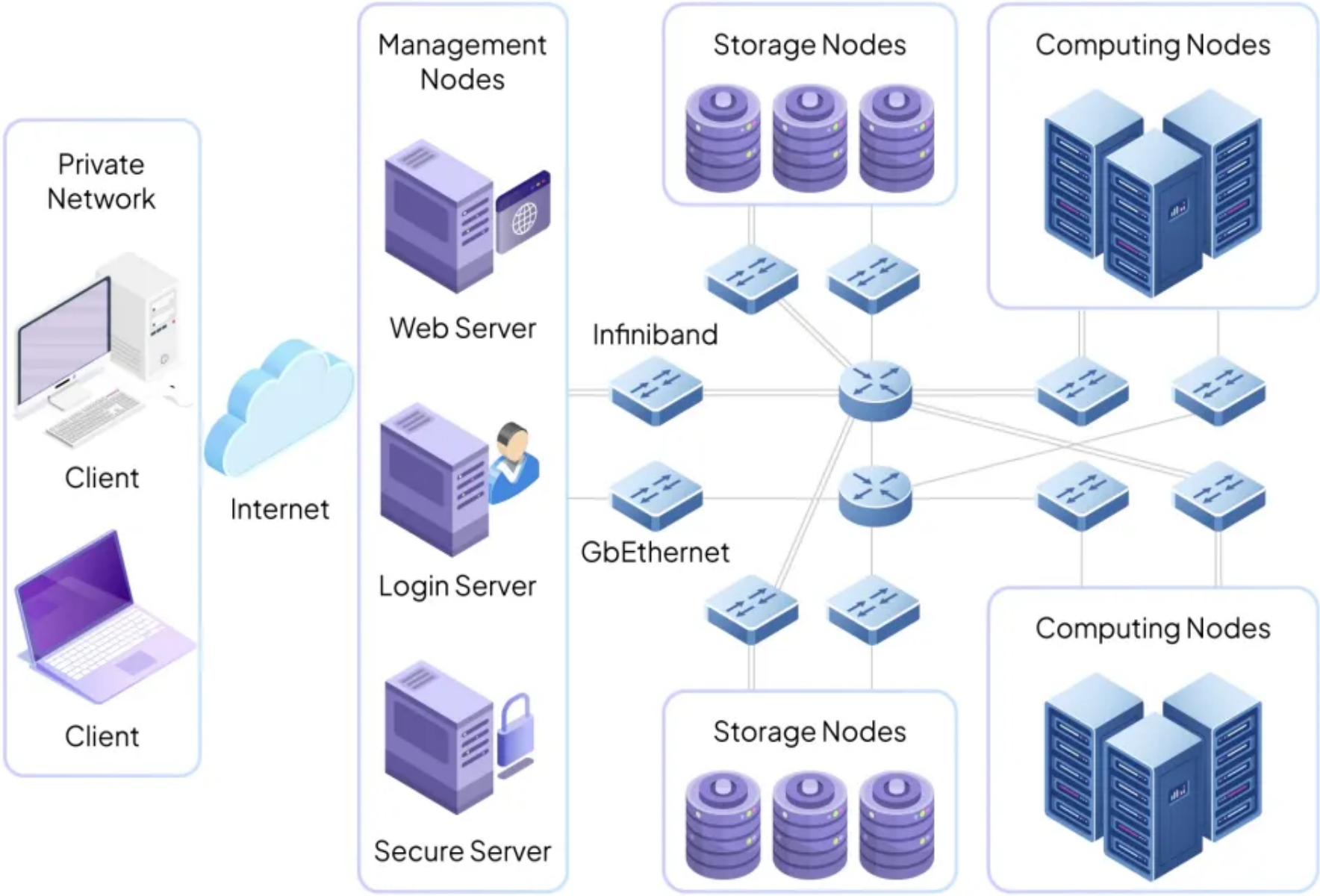
- Linux (RHEL/CentOS/Alma, Rocky, Ubuntu, SLES) with NUMA, hugepages, cpufreq governor=performance.

2) Network stack (fabric + comms)

- Fabric/NIC drivers: Mellanox/NVIDIA OFED (InfiniBand), RoCEv2, or high-speed Ethernet.
- RDMA stack: rdma-core, verbs, SR-IOV (if virtualized), PFC/ECN (for RoCE).

3) Resource & job management

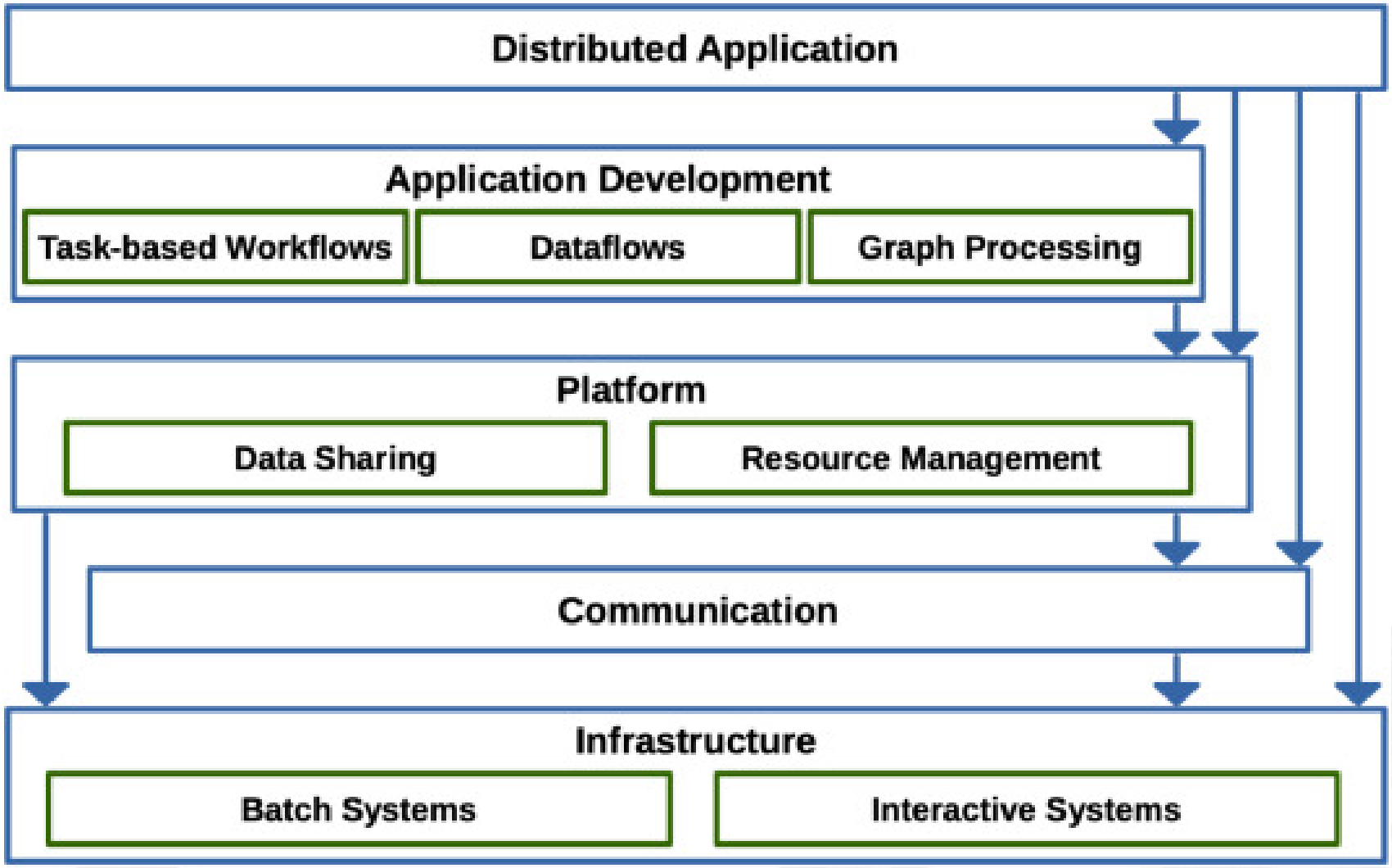
HPC Infrastructure



High Performance Computing

- Basics
- Hardware Stack
- Software Stack
- **Programming Model**

Data Processing Stack: Distributed Programming Models

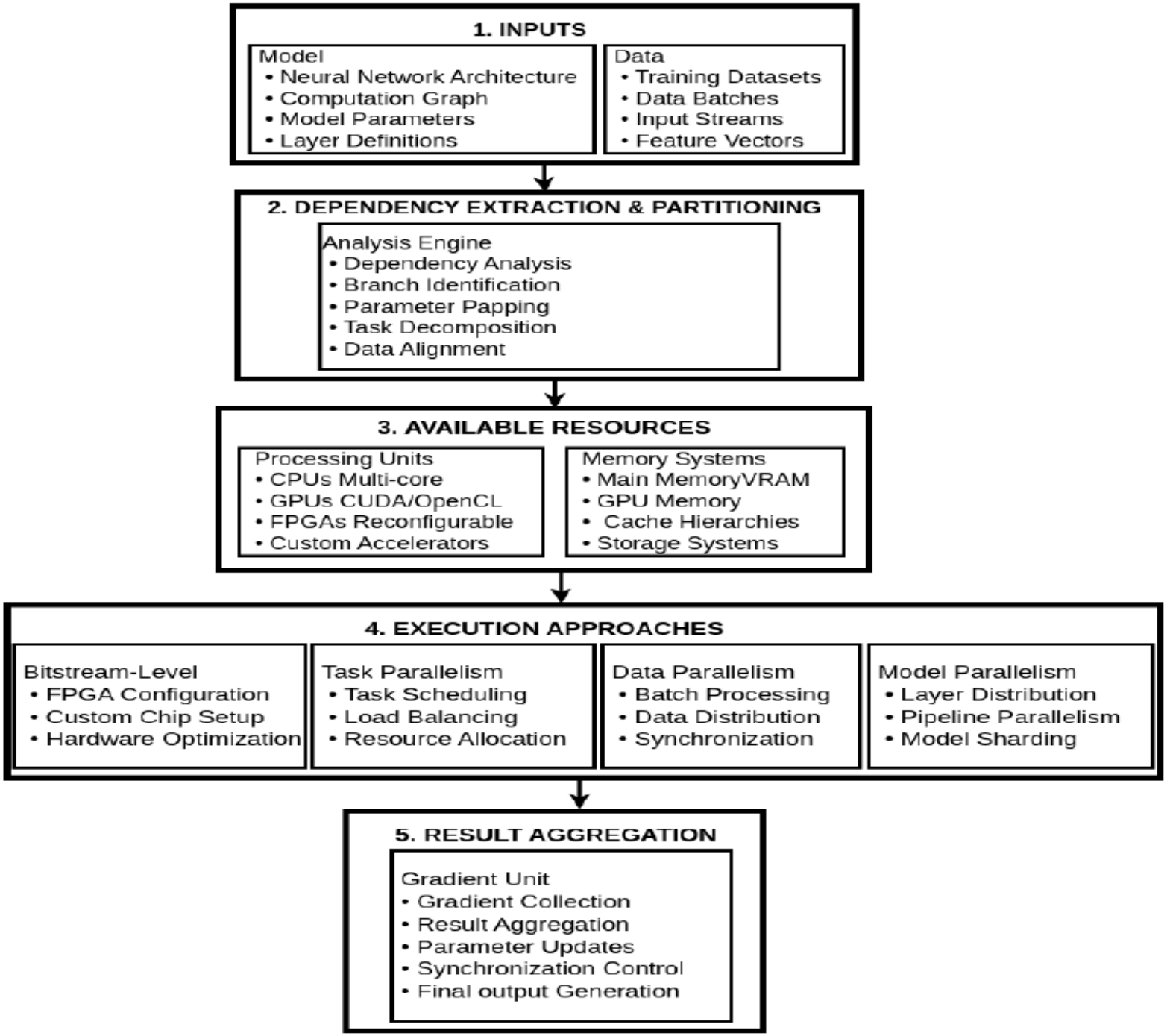


		HPC	AI	BigData
Apps...	Boundary Interaction Services	- **Remote access**: X2Go, Apache Guacamole, OpenVNC - **Visualization**: ParaView, VisIt (via X11 forwarding or browser) - **Secure access**: OpenSSL, SSH with key-based login - **User portals**: Cockpit, Slurm-Web	- **Model serving**: FastAPI, Gradio, Streamlit (self-hosted) - **Web access**: JupyterHub, VS Code Server - **Authentication**: OpenSSL/TLS, OAuth2 via open-source IdPs (e.g., Keycloak)	- **Dashboards**: Grafana, Superset, Apache ECharts - **Remote data access**: WebHDFS, OpenNMS - **Visualization tools**: Apache Zeppelin, D3.js, Plotly Dash - **Secure access**: OpenSSL, SSH
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]

High Performance Computing	Cloud Computing	Big Data, Edge Computing
Access & Security		
X2Go, Guacamole, OpenVNC, SSH, Cockpit, Slurm-Web	FastAPI, Gradio, Triton, JupyterHub, Keycloak (OAuth2), VS Code Server	Grafana, Superset, Zeppelin, WebHDFS, SSH + Keycloak
Frameworks and Libraries		
PETSc, OpenHPC, Spack, SLURM, OpenMPI	PyTorch, TensorFlow, ONNX, JAX, MLFlow, HuggingFace, Kubeflow	Spark, Flink, Mahout, MLlib, H2O.ai, Scikit-learn, Druid
Storage and Messaging		
Lustre, BeeGFS, GlusterFS, CephFS, HDF5, ADIOS2, NetCDF	HDFS, MinIO, MongoDB, Redis, Ceph, DVC, Weights & Biases	HDFS, Iceberg, Delta Lake, MinIO, Kafka, Pulsar, Zookeeper
Programming & Languages		
C, C++, Fortran, Python, Julia, OpenMP, Spack	Python, Julia, R, YAML (Kube), TensorBoard, Torch Profiler	Java, Scala, Python, Pig, HiveQL, Airflow DAGs
DevOps, MLOps & CI/CD		
Spack, EasyBuild, Singularity, Ansible, Warewulf, Podman	Kubeflow, MLFlow, Metaflow, DVC, Docker, K3s, KServe	Jenkins, ArgoCD, GitLab CI, NiFi, StreamSets, Airflow
Containerization & Orchestration		
Podman, Singularity, Apptainer, KVM, SLURM	Docker, Kubernetes, K3s, Kubeflow Pipelines	Docker, Kubernetes, Helm, MicroK8s, Oozie
OS, Virtualization, Networking		
Rocky Linux 9.x, AlmaLinux, KVM, Infiniband, RoCEv2	Ubuntu 22.04+, Debian 12+, KVM, K3s, RDMA, Ethernet	Ubuntu, Debian, Kubernetes CNI (Calico, Flannel), Ethernet
Monitoring & Performance		
Prometheus, DCGM, Ganglia, Grafana, Valgrind, Vampir, DDT	Prometheus, Grafana, TensorBoard, PyTorch Profiler	Prometheus, Grafana, Alertmanager, Kafka Monitor
Hardware & Services		
CPU, GPU, FPGA, NAS/SAN, NVMe, High-speed fabrics	CPU, GPU, TPU, Edge Devices (Jetson), NVMe, Ceph	CPU, GPU, Hyper-converged nodes, MinIO, Ceph, HDFS

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.)				


1. INPUTS
Model
• Neural Network Architecture
• Computation Graph
• Model Parameters
• Layer Definitions
Data
• Training Datasets
• Data Batches
• Input Streams
• Feature Vectors
2. DEPENDENCY EXTRACTION & PARTITIONING
Analysis Engine
• Dependency Analysis
• Branch Identification
• Parameter Papping
• Task Decomposition
• Data Alignment
3. AVAILABLE RESOURCES
Processing Units
• CPUs Multi-core
• GPUs CUDA/OpenCL
• FPGAs Reconfigurable
• Custom Accelerators
Memory Systems
• Main MemoryVRAM
• GPU Memory
• Cache Hierarchies
• Storage Systems
4. EXECUTION APPROACHES
Bitstream-Level
• FPGA Configuration
• Custom Chip Setup
• Hardware Optimization
Task Parallelism
• Task Scheduling
• Load Balancing
• Resource Allocation
Data Parallelism
• Batch Processing
• Data Distribution
• Synchronization
Model Parallelism
• Layer Distribution
• Pipeline Parallelism
• Model Sharding
5. RESULT AGGREGATION
Gradient Unit
• Gradient Collection
• Result Aggregation
• Parameter Updates
• Synchronization Control
• Final output Generation

Problem

Digital

Data
Software
Hardware
Front-end

Low-Performance

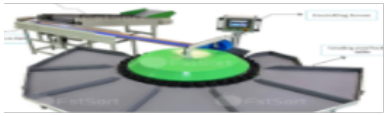
Technology

High-Performance

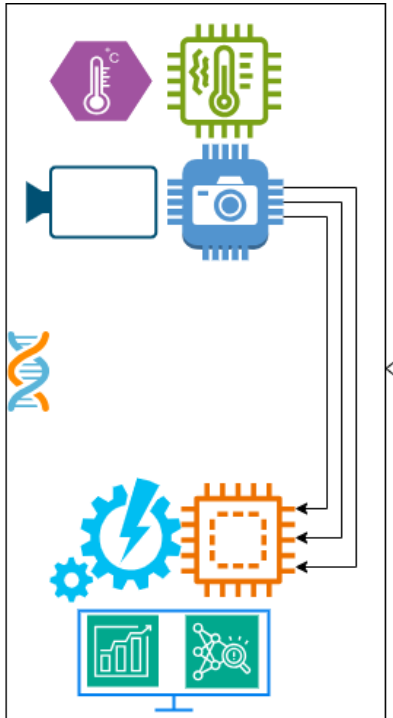
Embedded/Edge

Cloud

Bare-Metal

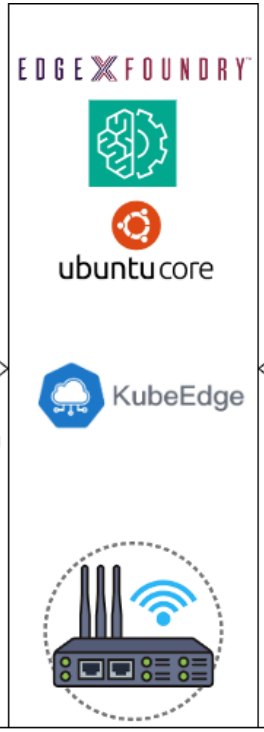


Sensors Netowrk, IoT, Automation Fields, Farms, Processing Units and Research Labs



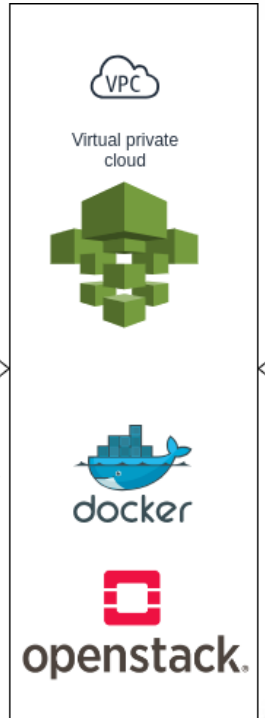
Real Time Computing
Decision Support System

Edge Computing Network



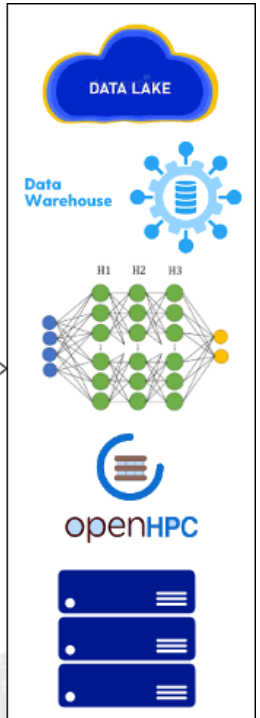
Edge Computing
Basic Analysis, Pre-Processing
Weak-AI, Machine Learning

Edge-to-Cloud Connectivity



Cloud Computing
Classification, Prediction
Deeap Learning

Cloud-to-Bare Metal Network.

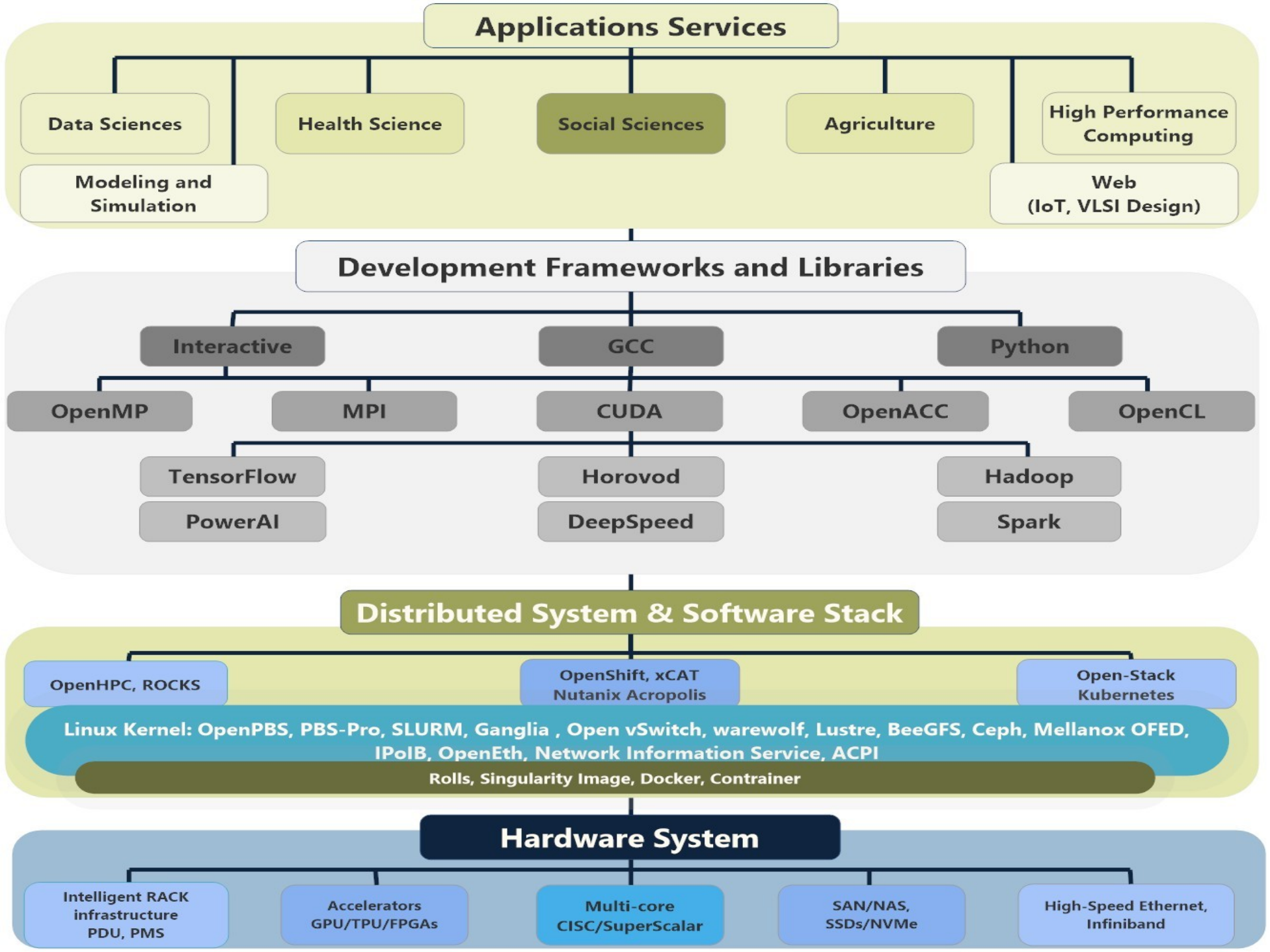


Supercomputing
HPC, Big-Data Processing
Deep/Reinforcement Learning

Baremetal: ssh namal-hpc@hpc.computingpark.com
Cloud Application: http://cloud.computingpark.com/
Data Center: https://data.computingpark.com

AI and BigData Applications

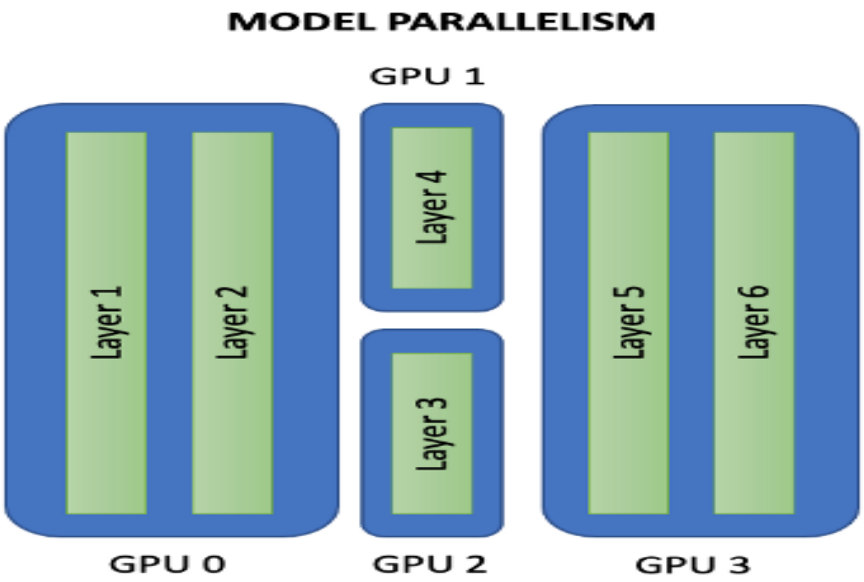
- **Bare-Metal and Containerized Cluster Infrastructure:**
  - Distributed Hardware Interfacing, Network Configuration and Distributed Computing Software Deployment
- **Data Center and Cloud Infrastructure:**
  - Storage systems, networking equipment, and software configuration
- **AI Applications for Scientific and Engineering Problems**
  - Distributed AI applications for multi-node bare-metal system
- **HPC Application Parallel Programming**
  - Heterogeneous multi-node parallel processing using parallel programming models

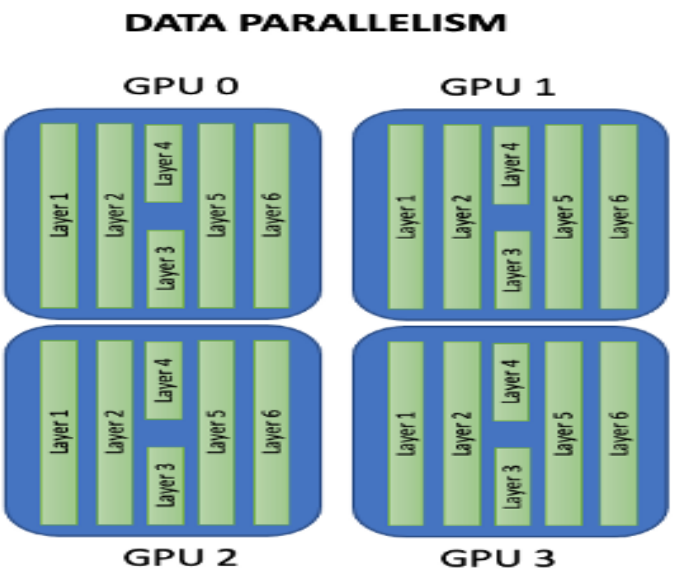

Applications Services
Data Sciences
Health Science
Social Sciences
Agriculture
High Performance Computing
Modeling and Simulation
Web (IoT, VLSI Design)
Development Frameworks and Libraries
Interactive
GCC
Python
OpenMP
MPI
CUDA
OpenACC
OpenCL
TensorFlow
Horovod
Hadoop
PowerAI
DeepSpeed
Spark
Distributed 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

Parallel Processing

- **Model Parallelism**
  Different layers of the network distributed across different devices
- **Data Parallelism**

Same model in every one of the GPUs, each processing a separate piece of the data, a separate portion of the mini-batch.





Visit us: ssh username@hpc.pakistansupercomputing.com

Pakistan's Number One High Performance Computing Facility:

The hardware architecture includes: 20 Nodes, 1600 Processor Core, 5 Tera Byte Main Memory, 40 TeraByte SSD, 10 Gigabit Fast Ethernet, Low Latency Switch, 40 4070TI GPU for Distributed Acceleration.

The Supercomputer is build on Rocky Linux 9.4 and features an advanced software stack including RoCE-enabled networking, Lustre parallel file system, Slurm workload manager, distributed AI and parallel programming models, with Grafana and Prometheus for real-time monitoring, and Ansible for automated deployment and management.

Namal Supercomputing Facility

Linux Operating system
(Ubuntu/Rocky)

20 Nodes

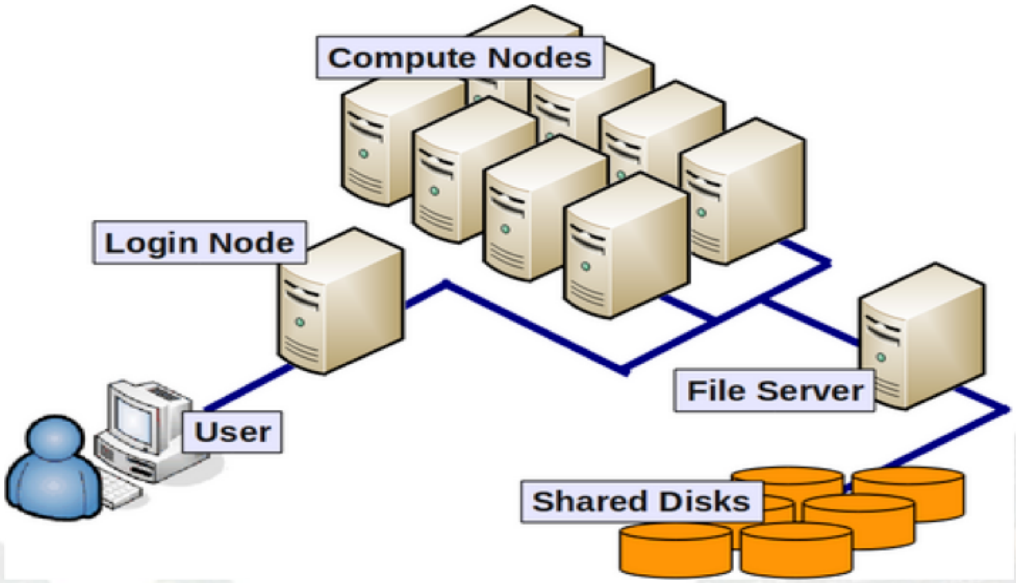
1600 Processor Cores

5 Tera Byte Main Memory

40 Tera Byte SSD

10 Gigabit Fast Ethernet

Low Latency Switch



HPC Scheduler (Updated)

Slurm Script:

```
#!/bin/bash
#SBATCH --job-name=gpu_test
#SBATCH --output=%j.out
#SBATCH --error=%j.err
#SBATCH --nodes=6              # Set the number of GPU nodes you want
#SBATCH --ntasks=6             # Each Task on A node
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:1
#SBATCH --distribution=block
# Single srun will launch in parallel on all allocated nodes
srun bash -c '
    echo "=== Running on $(hostname) ==="
    nvidia-smi
    nvidia-smi --query-gpu=uuid –format=csv
```

Explanation: This script submits a job to Slurm to test GPU availability and status across 6 nodes, each running 1 task using 1 GPU and 4 CPUs.

Slurm Commands

Command	Description
srun	Runs a job interactively or in a script.
sbatch <file>	Submit a job script to Slurm.
scancel <jobID>	Cancels a submitted job.
squeue	Displays job queue and status.
sinfo	Shows status of nodes and partitions.
scontrol show job <jobID>	Detailed info about a running job.
sacct	Shows job accounting and completion history.