Infrastructure Comparison Tables

Martina Castellucci

Big Data Processing Distributed Computing: Teaching Summary, Conceptual Links, and Questions

—

1

Quick Technical Recap by Topic (Memorization Notes)

1.1

From Architecture to Containers: Full Logical Progression

Modern computing infrastructures are built on a layered and increasingly abstract architecture, each level adding new capabilities, flexibility, or scalability. This progression not only reflects the evolution of technologies but also models the complexity of today's big data and scientific computing environments.

We begin with **computer architecture**, which defines the hardware backbone: processors (CPUs), memory (RAM), buses, cache, and physical storage. This level determines raw performance, memory hierarchy, and instruction processing. It's the ground layer upon which all higher layers operate.

From hardware, we move to **networking**, which connects nodes together using standardized protocols and layered models like OSI and TCP/IP. LANs, WANs, switches (Layer 2), and routers (Layer 3) enable machines to communicate and share resources. MAC addresses identify interfaces; IP addresses identify logical positions in a network. VPNs allow secure tunneled access across public infrastructure. DNS maps human-readable names to IPs.

On this connected infrastructure, we build **computing farms or clusters**: sets of networked machines that work together to process tasks. These systems use shared filesystems and coordinated job submission interfaces. They enable parallelism and high-throughput workloads but require resource management.

To manage these farms, **batch systems** are introduced. These allow non-interactive job submission using schedulers like SLURM, HTCondor, PBS. Jobs enter a queue and are dispatched based on policies like fair-share (historical use), priority, and availability. Concepts like *backfilling* and *reservation escape* increase resource utilization.

As data volumes grow, we need scalable **storage systems**:

- **DAS** (Direct Attached Storage): local to the machine, fast, but not network-shared.
- **NAS** (Network Attached Storage): shared file-based access over Ethernet; easy to mount.
- **SAN** (Storage Area Network): shared block-level storage over Fibre Channel/iSCSI; presented as local disk.

In large environments, *parallel file systems* (e.g., Lustre) enable multiple nodes to read/write simultaneously. Tiered storage strategies assign data to hot (SSD), warm (HDD), or cold (tape/archive) tiers depending on access frequency.

Cloud computing emerged to abstract and automate resource provisioning. It offers *Infrastructure-as-a-Service* (e.g., EC2), *Platform-as-a-Service* (e.g., Heroku), and *Software-as-a-Service* (e.g., Gmail). Key cloud features include elasticity (scale out/in), on-demand provisioning, pay-per-use billing, multi-tenancy, stateless design, and rapid deployment.

To support flexible deployment, **virtualization** was introduced: running multiple OS instances on a hypervisor (e.g., KVM, VMware). Virtual Machines (VMs) provide full isolation but have slow boot and high overhead. **Containers** (e.g., Docker) emerged as lightweight alternatives, sharing the host kernel, starting instantly, and requiring fewer resources. Containers are ideal for microservices, CI/CD, and reproducible pipelines.

Before cloud, **grid computing** federated compute resources across institutions. It is designed for *HTC* scenarios: users submit many independent jobs (e.g., simulations, Monte Carlo, BLAST). Grids require middleware (e.g., gLite), digital certificates, VO policies, and workload management systems (WMS).

We then differentiate **HTC vs HPC**. *HTC (High Throughput Computing)* emphasizes total jobs over time, suitable for independent tasks. *HPC (High Performance Computing)* focuses on tightly coupled jobs requiring parallel execution and synchronization, often with MPI or OpenMP. HPC needs high-bandwidth interconnects, fast memory, and shared state.

Finally, we reach **containers and orchestration**. Docker enables defining, building, and running containerized apps using Dockerfiles, volumes, and networks. Images are pushed to or pulled from registries like Docker Hub. Systems like `docker-compose` or `Kubernetes` handle multi-container orchestration. `Udocker` enables non-root container execution, useful in academic/batch environments.

This path—from hardware to networks, clusters, batch processing, storage, cloud, virtualization, grid, compute paradigms, and containers—describes the full ecosystem of modern distributed and data-intensive computing.

—

1.2

1. Distributed vs Parallel Systems HTC is *distributed* with independent tasks. HPC is *parallel*, tightly coupled. Grid is HTC, not cluster. Cluster = centralized; grid = federated.

1.3

2. Speedup, Efficiency, and Limits Speedup $S = T_{serial}/T_{parallel}$; Efficiency $E = S/N$. Amdahl's Law limits speedup. Crunching Factor = CPU time vs wall time. PUE = total power / IT power.

1.4

3. Virtualization vs Cloud vs Containers VMs = heavy, isolated. Containers = fast, share kernel. Cloud = abstraction + elasticity. Cloud-native apps = stateless, resilient.

1.5

4. Batch Systems and Scheduling HTCondor: pull-based scheduler. Batch is static. Cloud is dynamic. Backfill = optimize idle nodes. Fair-share = historical priority. Job = queued, non-interactive.

1.6

5. Containers: Docker Udocker Dockerfile defines image. CMD = default args; ENTRYPOINT = fixed behavior. Docker Daemon runs images. Compose = multi-container. Udocker = no root.

1.7

6. Storage Systems (DAS, NAS, SAN) DAS = local, not shared. NAS = file-level over network. SAN = block-level remote disk. Lustre = parallel FS. Tiered: hot (SSD), warm (HDD), cold (tape).

1.8

7. Cloud Characteristics and Terms Elasticity, multi-tenancy, stateless. Horizontal = more instances; vertical = stronger machines. Pay-per-use. Resource pooling. Failover = auto recovery.

1.9

8. Deployment Models and Isolation Public = shared, Private = org-owned, Hybrid = mix. Isolation: Bare Metal ¿ VM ¿ Container. Multi-tenant = shared infra, logical separation.

1.10

9. Miscellaneous Subtle Points Grid Cloud: Grid = VO, certs. Cloud = billing, users. Stateful = preserves context. Stateless = easy to scale. Cloud-aware = resilient design.

1.11

10. Key Docker Terms `docker build`, `docker run`, `push/pull`, `compose`. Daemon = runs containers. Udocker = user-level.

1.12

11. File System and Data Access NAS = NFS (file), SAN = iSCSI (block), S3 = object. Parallel FS = shared across nodes. Tiered = SSD/HDD/tape.

1.13

12. Edge, Fog, Cloud Layers Edge = low latency (on-device). Fog = gateway-level processing. Cloud = central. Hierarchy: Edge ¡ Fog ¡ Cloud.

1.14

13. Metrics and Units Speedup, efficiency = ratios. Time = sec, CPU-hrs. Power = PUE. Billing = $/hr, $/req. Storage = GB/TB. Latency = ms. Bandwidth = Gbps.

1.15

14. Networking Details Switch = MAC, Layer 2. Router = IP, Layer 3. VPN = secure tunnels. DNS = name resolution. IP = logical; MAC = physical.

1.16

15. Commands to Know `ping`: test reachability. `ifconfig`, `ip a`: interfaces. `traceroute`: packet path. `netstat`: open sockets. `nslookup`: DNS check. `bwa aln/samse`: sequence alignment.

—

These notes now include a more detailed and discursive overview, following a layered progression from hardware to modern distributed and cloud-native architectures.

Comparison Tables

Docker vs Docker Compose vs Udocker

Feature	Docker	Docker Compose	Udocker
Root Access	Required for daemon	Depends on setup	Not required; runs in user space
Use Case	Running individual containers	Orchestrating multi-container apps	Running containers in HPC without root
Security	Host-level access required	Shares Docker context	User-level, no root access
Deployment	CLI or GUI	Uses YAML config files	CLI compatible with Docker images

Cloud Models

Model	Description and Responsibility
IaaS (Infrastructure as a Service)	Provides virtualized computing resources over the internet. The user manages the operating system, storage, deployed applications, and runtime. Examples include AWS EC2, Google Compute Engine, Microsoft Azure VM.
PaaS (Platform as a Service)	Provides a platform allowing customers to develop, run, and manage applications without dealing with infrastructure. The provider handles OS, middleware, and runtime. Examples: Heroku, Google App Engine.
SaaS (Software as a Service)	Offers ready-to-use software applications over the web. The provider manages everything from infrastructure to software. Users only interact with the interface. Examples: Gmail, Dropbox, Microsoft 365.

Cooling Techniques

Method	Description
Air Cooling	Uses fans and heat sinks to remove heat from components. It is the most economical method, ideal for general-purpose systems, but becomes inefficient in high-density setups.
Liquid Cooling	Employs a coolant fluid circulated via tubes and blocks to draw heat away from processors and components. Offers superior thermal performance and quieter operation.
Immersion Cooling	Submerges entire components or servers in a dielectric fluid. It ensures uniform cooling, reduces noise, and is used in high-performance, energy-efficient data centers.

File Systems

File System	Usage and Characteristics
ext4	Widely used local file system on Linux. Supports journaling, fast performance, good for general-purpose servers.
xfs	High-performance journaling FS for parallel I/O. Efficient for large files and used in enterprise Linux systems.
NFS	Network File System. Enables sharing files over network. Suitable for shared storage in clusters.
GPFS (IBM Spectrum Scale)	Parallel file system for HPC. Provides scalability and high-throughput for large workloads.
Lustre	Designed for large-scale parallel file systems in supercomputers and clusters. Open source and scalable.

Kubernetes vs Docker

Aspect	Docker	Kubernetes
Function	Tool to build, run, and manage containers on a single host	Orchestration system to deploy, scale, and manage containers across multiple hosts
Scalability	Limited to a single host (or Docker Swarm)	Designed for scaling across many machines and fault tolerance
Networking	Provides simple bridge and host networking	Built-in service discovery and load balancing via Services
Use Case	Local development, small deployments	Production-grade orchestration of microservices at scale

Cloud vs Virtualization

Aspect	Description
Virtualization	Virtualization allows multiple operating systems to run on the same physical hardware by abstracting the hardware using a hypervisor (e.g., KVM, VMware ESXi). It forms the basis for cloud computing but can be used independently in on-prem setups.
Cloud Computing	A service delivery model that provides access to computing resources (VMs, storage, databases, etc.) over the internet. Built on top of virtualization, it adds elasticity, scalability, APIs, billing, and managed services.

Recall vs Backfill vs Reservation

Term	Meaning
Recall	The process of reclaiming allocated but unused or idle resources (e.g., from preemptible or low-priority jobs) so they can be reassigned. Useful in opportunistic environments.
Backfill	A scheduling strategy where smaller jobs are temporarily scheduled in idle gaps between larger reservations, improving overall resource utilization without delaying high-priority jobs.
Reservation	Explicit allocation of compute resources for a specific user, group, or job at a future time. Ensures resource availability for time-sensitive workloads.

OSI vs TCP/IP Models

Layer	OSI Model	TCP/IP Model
Layer 7	Application	Application
Layer 6	Presentation	– (merged into Application)
Layer 5	Session	– (merged into Application)
Layer 4	Transport	Transport
Layer 3	Network	Internet
Layer 2	Data Link	Network Access
Layer 1	Physical	Network Access

TOR (The Onion Router) Architecture

Layer	Function
Entry Node	First relay in the TOR network. It knows the user's IP address and establishes an encrypted path to the next node.
Relay Node	Intermediate node that passes traffic in encrypted form. It does not know the source or final destination.
Exit Node	Final node in the circuit. It decrypts the last layer and forwards the request to the destination server. It sees the destination but not the source.

HTCondor vs SLURM

Feature	HTCondor	SLURM (HPC Scheduler)
Scheduling Model	Opportunistic scheduling with late-binding	Resource-reservation with partitioned scheduling
Target Use	High Throughput Computing (HTC), grid environments	High Performance Computing (HPC), tightly coupled tasks
Fault Tolerance	Supports checkpointing and job migration	Limited fault tolerance; job restarts on failure
Resource Match	Uses ClassAds to match jobs with resources	Uses queues and partitions, less dynamic
Scalability	Scales across heterogeneous systems in a grid	Scales efficiently in tightly integrated clusters

INFN CNAF vs Other Sites

Location	Description
INFN CNAF	National computing center (Tier-1) in Bologna. Offers large-scale storage, computing, and data services for LHC and Italian research. Connected to GRID, provides access to HPC and HTC resources.
Other Sites	Regional centers (Tier-2 or Tier-3). Focused on localized support, smaller infrastructure, often used for preprocessing or analysis tied to larger collaborations.

Amdahl's Law: Speedup vs Efficiency

Metric	Formula and Explanation
Speedup (S)	$S = \frac{1}{(1-P)+\frac{P}{N}}$ — quantifies how much faster a program runs when parallelized, where P is the parallelizable portion and N the number of processors.
Efficiency (E)	$E = \frac{S}{N}$ — evaluates how effectively the parallel resources are being used; close to 1 means optimal utilization.

Power vs Storage

Aspect	Metric / Description
Power	Measured in Watts. TDP (Thermal Design Power) indicates the max heat a component generates; influences cooling requirements.
Storage	Measured in GB/TB. Performance determined by latency, throughput, and IOPS (Input/Output Operations Per Second).

Node vs Job

Entity	Description
Node	A compute resource — physical or virtual — in a cluster that executes assigned jobs. Often includes CPUs, RAM, and access to shared storage.
Job	A user-defined task or process submitted to a scheduler for execution on one or more nodes. Can be serial, parallel, or array-based.

Server vs Data Center vs Distributed

System Type	Scope and Characteristics
Server	A single machine offering computing or storage services. Can be physical or virtual and may serve multiple users or applications.
Data Center	A facility that houses many servers with supporting infrastructure (cooling, power, security). Centralized and managed IT operations.
Distributed Computing	A computing model in which components located on networked computers communicate and coordinate to achieve a common goal. Supports scalability and fault tolerance.

CPU vs GPU

Feature	CPU (Central Processing Unit)	GPU (Graphics Processing Unit)
Purpose	General-purpose computing	Parallel processing, mainly for graphics and large-scale computations
Cores	Fewer (2–64), optimized for sequential tasks	Hundreds to thousands of cores, optimized for parallel tasks
Latency	Low latency per task	Higher latency per task
Throughput	Lower overall throughput	Very high throughput
Use Case	OS management, general logic, low-latency tasks	Deep learning, image rendering, matrix operations

Single-core vs Multi-core CPU

Aspect	Single-core CPU	Multi-core CPU
Definition	One execution core per CPU	Multiple cores in a single CPU chip
Parallelism	Executes one thread at a time	Executes multiple threads simultaneously
Performance	Slower for multitasking	Better multitasking and parallel computing
Power Consumption	Typically lower	More power, but better performance/watt
Use Cases	Embedded devices, old systems	Modern desktops, servers, smartphones

The 5 Vs of Big Data

V	Description
Volume	Refers to the huge amounts of data generated every second (e.g., terabytes, petabytes)
Velocity	Speed at which data is generated, processed, and analyzed
Variety	Different types of data: structured, semi-structured, unstructured
Veracity	Quality and trustworthiness of data; presence of noise and uncertainty
Value	The usefulness of the data once processed and analyzed

DAS vs NAS vs SAN

Type	Description
DAS (Direct Attached Storage)	Storage attached directly to a server or workstation without a network
NAS (Network Attached Storage)	File-level storage shared over a network; uses protocols like NFS, SMB
SAN (Storage Area Network)	Block-level storage accessed over a high-speed network, typically via Fibre Channel or iSCSI

Edge vs Fog vs Cloud Computing

Model	Description
Edge Computing	Processing done near the data source (e.g., sensors); reduces latency and bandwidth
Fog Computing	Intermediate layer between edge and cloud; distributes computing/storage closer to edge
Cloud Computing	Centralized processing in data centers; offers scalability, redundancy, and global access

AWS and IaaS Example

Service	Description
AWS EC2	Elastic Compute Cloud; provides resizable compute capacity in the cloud (IaaS)
AWS S3	Simple Storage Service; object storage with web interface, often used with EC2
AWS VPC	Virtual Private Cloud; isolated network for AWS resources
Relation to IaaS	AWS provides IaaS by allowing users to manage VMs, storage, and networking infrastructure

Batch Computing vs Cloud Computing

Feature	Batch Computing	Cloud Computing
Job Execution	Scheduled in queues, managed by local scheduler	On-demand job execution via cloud APIs
Resource Allocation	Static, predefined	Elastic and scalable
Cost Model	Fixed, shared among users	Pay-per-use (metered billing)
Infrastructure	On-premise or academic clusters	Hosted and managed by provider (e.g., AWS, Azure)

Batch Computing vs Grid Computing

Aspect	**Batch Computing**	**Grid Computing**
Scope	Typically within a single organization or cluster	Federated, across multiple administrative domains
Resource Sharing	Centralized, static	Decentralized, dynamic sharing
Middleware	Simple batch schedulers (e.g., SLURM, PBS)	Grid middleware (e.g., gLite, Globus, ARC)
Security	Based on local users	Often uses certificates and federated identities

Scheduler vs Cloud Provider

Aspect	**Scheduler (e.g., SLURM, HTCondor)**	**Cloud Provider (e.g., AWS, Azure)**
Responsibility	Manages job queueing, prioritization, and execution on a cluster	Provides infrastructure and services for compute, storage, and networking
Control	Full control over resource management	Limited to APIs and services offered
Location	Local clusters or HPC/HTC systems	Remote, virtualized data centers
Scalability	Bound to physical resources	Virtually unlimited scaling

Dockerfile vs Docker Image

Concept	**Dockerfile**	**Docker Image**
Definition	Script with instructions to build an image	Executable snapshot of a container environment
Use	Written by developer to define build steps	Created after Dockerfile is built; used to run containers
Example `COPY . /app`	`FROM python:3.8` `myimage:v1` (result of `docker build`)	

Digital Twins

Term	Description
Digital Twin	A digital representation of a real-world object or system that is updated with real-time data. Used for monitoring, simulation, and optimization (e.g., industrial IoT, smart cities).

IP vs MAC Address + Common IPs

Term	Description
IP Address	Logical address used for routing on network layer (e.g., 192.168.0.1)
MAC Address	Physical address tied to network interface; layer 2 identifier (e.g., 00:1A:2B:3C:4D:5E)
127.0.0.1	Loopback address (localhost); used for internal testing
0.0.0.0	Non-routable address; binds to all available interfaces

Checksums

Term	Description
Checksum	A value derived from data to detect errors or integrity loss. Common algorithms: MD5, SHA-1, SHA-256.
Use Cases	File verification, network transmission, package downloads

BLASTn vs BWA

Tool	BLASTn	BWA
Purpose	Nucleotide sequence similarity search	Read alignment to reference genome
Algorithm	Seed-and-extend; local alignment	Burrows-Wheeler Transform; global or semi-global
Speed	Slower, especially with large datasets	Very fast, optimized for NGS data
Use Case	Annotating genes, database comparison	DNA-seq data preprocessing, variant calling

HTT (High Throughput Technologies)

Term	Description
High Throughput Technologies	Techniques that allow simultaneous processing of a large number of samples or data points (e.g., microarrays, NGS, mass spectrometry). Key in omics and screening platforms.

WMS (Workflow Management Systems)

System	Description
WMS	Tools that help define, manage, and execute data pipelines (e.g., Snakemake, Nextflow, Galaxy). Support reproducibility, scalability, and parallelism in bioinformatics and big data workflows.

Vocabulary: Key Infrastructure Terms

Term	Definition
Switch	Network device that connects devices within a LAN and forwards data based on MAC addresses.
Router	Device that connects multiple networks and routes packets based on IP addresses. Often links LANs to WANs.
Hub	Basic network device that broadcasts data to all connected devices; no intelligence like a switch.
LAN (Local Area Network)	Network covering a small geographic area (e.g., office, home), typically high-speed.
WAN (Wide Area Network)	Network covering a large area, such as the internet or inter-office networks.
Hardware	The physical components of a computer or infrastructure (CPU, memory, disk, etc.).
Software	Programs and operating systems that run on hardware and control tasks.
Kernel	Core of the operating system that manages system resources and communication between hardware and software.
Host	A device (physical or virtual) that provides services and has an operating system installed.
OS (Operating System)	System software that manages computer hardware, software resources, and provides services for applications.
HDD (Hard Disk Drive)	Traditional spinning magnetic storage device, high capacity, slower access.
SSD (Solid State Drive)	Fast, non-volatile flash memory storage device with no moving parts.
NFS (Network File System)	Protocol that allows file access over a network as if local; used in shared UNIX/Linux environments.
NUMA (Non-Uniform Memory Access)	Memory architecture where access time depends on the memory's location relative to the processor.
UMA (Uniform Memory Access)	Architecture where all processors share physical memory uniformly.
L1 Cache	The smallest and fastest memory level, located closest to the CPU core; typically stores instructions and data.
L2 Cache	Larger than L1, slower but still fast; stores data used less frequently. Often shared among cores.
L3 Cache	Shared among all cores in a processor; bigger but slower than L2, reduces memory latency at higher levels.
RAID (Redundant Array of Independent Disks)	A method of storing data on multiple hard disks for redundancy and performance, using various levels (RAID 0, 1, 5, etc.)

Vocabulary: Key Infrastructure Terms 2)

Tiered Storage	Storage architecture that uses multiple storage media (e.g., SSD, HDD, tape) ranked by speed/cost, and data is moved accordingly.
Hyperthreading	Intel technology that allows one physical CPU core to execute two threads simultaneously to improve parallelism.
Throughput	The amount of data processed in a given time, often measured in bits per second or IOPS in storage.
Grid Computing	Distributed computing model where resources across multiple locations are federated and used collectively.
Cloud Computing	Delivery of computing services (compute, storage, networking) over the internet on demand.
Edge Computing	Processing data close to where it is generated (e.g., IoT sensors) to reduce latency.
Fog Computing	Extends cloud to be closer to edge devices, offering computation, storage, and networking.
Stream Processing	Real-time processing of continuous data streams (e.g., Kafka, Flink).
ALU (Arithmetic Logic Unit)	Part of the CPU that performs arithmetic and logical operations.
Control Unit	CPU component that directs operations of the processor; it fetches, decodes, and executes instructions.
I/O (Input/Output)	Interface between the computer and external devices; includes peripherals and network input/output.
Container	Lightweight, portable, and self-sufficient unit that includes everything needed to run a piece of software.
Docker	Popular container platform that automates deployment of applications inside containers.

Common Command Reference Table

Command	Function / Description
docker pull <image>	Download an image from a Docker registry (e.g., Docker Hub)
docker push <image>	Upload an image to a Docker registry
docker run <image>	Create and start a container from an image
docker create <image>	Create a container without starting it
docker build -t <name> .	Build an image from a Dockerfile in the current directory
docker volume create <name>	Create a named volume (persistent storage for containers)
docker-compose up	Start all services defined in a docker-compose.yml file
docker-compose down	Stop and remove services, networks, volumes defined in the compose file
udocker run <container>	Execute a container without requiring root privileges (HPC environments)
condor_submit <file>	Submit a job to HTCondor using the specified job description file
condor_q	View the status of submitted jobs in the queue
condor_status	Show the status of machines in the Condor pool
cat /proc/cpuinfo	Display detailed CPU architecture and core information
free -m	Show memory usage in megabytes
df -h	Show available disk space on mounted file systems (human readable)
mount	List all currently mounted filesystems
lsblk	Show all block devices and partitions
md5sum <file>	Generate or verify MD5 checksum for a file
time <command>	Measure the time taken to execute a command (e.g., time blastn)
blastn -query file -db nt -out res	Run a nucleotide BLAST search
ping	A networking command that sends ICMP echo requests to check if a remote host is reachable and measures round-trip time.
ifconfig	A system command (Linux/macOS) used to view or configure network interface settings such as IP addresses and MAC addresses (now mostly replaced by ip addr).
bwa aln	A command from the BWA (Burrows-Wheeler Aligner) tool that aligns short sequencing reads to a reference genome and produces a .sai intermediate file.
bwa samse	A BWA command that converts .sai alignment files (from bwa aln) into single-end SAM format, which contains the alignment information in a readable text format.

Docker Workflow Overview

```
Dockerfile
    |
    v
docker build
    |
    v
Docker Host (Docker Daemon)
    |
    |-------------------------> docker push
    |                               |
    v                               v
docker pull  <-----------------  Docker Hub (Registry)
    |
    v
Run container

[Optional]
    ^
    |
docker search
```

Basic Networking Diagram

```
[Device A]        [Device B]
   |                 |
   |  MAC Address     |  MAC Address
   v                 v
+---------+       +---------+
| Switch |-------| Switch |     ← operates using MAC addresses (Layer 2)
+---------+       +---------+
      |               |
      v               v
   [ LAN ]       [ LAN ]          ← Local Area Network
        \         /
         \       /
          \     /
         +------------+
         |  Router   |            ← operates using IP addresses (Layer 3)
         +------------+
               |
               v
            [ WAN ]               ← Wide Area Network (e.g., the Internet)
```

Storage Architectures

```
              +-------------------------+
              |     Client / Server     |
              +-------------------------+
                         |
       +----------------+----------------+
       |                                 |
   [DAS]                          [Network Storage]
Direct-Attached             +-----------+-----------+
   Storage                  |                       |
 (e.g., local SSD)        [NAS]                   [SAN]
                           |                        |
               File-level access         Block-level access
                over Ethernet            over Fibre/iSCSI
```

Cloud service models

```
+-------------------------+
|      SaaS               |  ← Software as a Service
|  (e.g. Gmail, Dropbox)  |  - End-user applications
+-------------------------+
|      PaaS               |  ← Platform as a Service
|  (e.g. Heroku, GCP App) |  - App dev/runtime, DBs
+-------------------------+
|      IaaS               |  ← Infrastructure as a Service
|  (e.g. AWS EC2, Azure)  |  - VMs, networks, storage
+-------------------------+
|      Bare Metal         |  ← Physical hardware
```

Edge – Fog – Cloud Computing

```
[ Data Source (e.g., IoT Sensor) ]
               ↓
        Edge Computing
     (on-device, gateway)
               ↓
         Fog Computing
   (local/near-site processing)
               ↓
        Cloud Computing
     (centralized datacenters)
```

Computing Farm and Batch system

```
User → Job Queue → Scheduler → Compute Nodes (Farm)
                                 ↓
                           HTCondor / SLURM
```

AWS

```
[ User / DevOps ]
       |
       v
   IAM Authentication
       |
       v
   Create EC2 Instance
       |
       +--> Attach EBS Volume (block storage)
       |
       +--> Access VPC (isolated network)
       |
       v
   Deploy App (manually or via ECS/EKS)
       |
       +--> Use S3 for object storage (e.g., upload files, models)
       |
       +--> Use RDS for DB storage (optional)
       |
       v
   Configure Route 53 (DNS) for domain access
       |
       v
   Monitor & Log with CloudWatch
       |
       v
     [ User Accesses App via Internet ]
```

HTC

```
[ User submits many independent jobs ]
                |
                v
        Job Scheduler (e.g., HTCondor)
                |
                v
    Jobs distributed across many nodes
        (no communication between them)
                |
                v
       Each job runs independently
                |
                v
    Output files collected asynchronously
```

HPC

```
[ User submits one large parallel job ]
                |
                v
     Job Scheduler (e.g., Slurm, PBS)
                |
                v
     Job runs across tightly-coupled nodes
         (often via MPI/OpenMP)
                |
                v
     Nodes communicate in real-time
                |
                v
     Synchronized parallel output/results
```

Big Data Processing Distributed Computing: Teaching Summary, Conceptual Links, and Questions

...

existingsectionsunchanged]...

—

2

Key Concepts Summary: Cloud, Distributed Systems, Virtualization

2.1

HTC vs HPC: Classification

- **HTC (High Throughput Computing)**: Distributed system for independent tasks.
- **HPC (High Performance Computing)**: Parallel system for tightly-coupled tasks.

2.2

Grid vs Cluster

- **Grid**: Heterogeneous, distributed, federated. *A form of HTC.*
- **Cluster**: Centralized, homogeneous, tightly-coupled. *Used in HPC.*

2.3

Parallel Performance Metrics

- **Speedup (S)**: $S = \frac{T_{serial}}{T_{parallel}}$
- **Efficiency (E)**: $E = \frac{S}{N}$
- **Crunching Factor**: Ratio between compute and wall-clock time.
- **PUE (Power Usage Effectiveness)**: $\text{PUE} = \frac{\text{Total Power}}{\text{IT Power}}$

2.4

Cloud Computing Characteristics

- **Multi-tenant**: Multiple users share infrastructure securely.
- **Cloud-aware**: Applications designed to scale in cloud environments.
- **Stateful / Stateless**: Whether the application maintains session state.
- **Pay-per-use**: Billing based on actual resource consumption.

- **Failover**: Automatic recovery in case of failure.
- **Elasticity**: Dynamic scaling of resources up/down.
- **Scalability**: Ability to handle increased workload.
- **Horizontal scalability**: Adding more instances.
- **Vertical scalability**: Increasing resource capacity of single instance.
- **Isolation**: Logical/physical separation of tenants or processes.
- **Static vs Dynamic**: Fixed vs on-demand resource provisioning.
- **Resource Pooling**: Shared resources dynamically allocated.

Feature	**Virtualization**	**Cloud**	**Batch System**	**Container**
Abstraction	VM with OS	Services/API	Static resource	App + deps
Isolation	Full OS per VM	Varies (multi-tenant)	None (shared nodes)	Shared kernel (namespaces)
Scalability	Manual	Auto-scale (elastic)	Fixed policy	Dynamic, light
Startup Time	Slow (boot OS)	Medium/Fast	Scheduled	Instant
Pay-per-use	No	Yes	No (quotas)	No (indirect)
Deployment	Manual VM config	On-demand via UI/API	Via scheduler queue	Dockerfile / Compose
Use Case	OS testing, legacy apps	SaaS, PaaS, IaaS workloads	HPC/HTC jobs	Microservices, CI/CD
Tools	KVM, VMware, VirtualBox	AWS, GCP, Azure	HTCondor, SLURM	Docker, Singularity

Table 1: Compact comparison of Virtualization, Cloud, Batch System, and Containerization

2.5

Feature	**HTC**	**HPC**
Job Coupling	Independent	Tightly Coupled
Communication	Minimal	Intensive (MPI/OpenMP)
Scheduler	HTCondor	Slurm, PBS, LSF
Goal	Maximize throughput	Maximize speed/performance

2.6

Deployment and Isolation Models

- **VM**: Full OS, strong isolation via hypervisor.
- **Container**: Lightweight, shared kernel, fast startup.
- **Bare Metal**: Direct use of hardware, no virtualization.
- **Serverless**: Abstracted infra, runs code without server management.

2.7

Units of Measurement

- Compute Time: Seconds, CPU hours
- Speedup / Efficiency: Ratios or percentages
- Storage: Bytes, GB, TB
- Network Latency: Milliseconds (ms)
- Power Efficiency: Watts, Joules, PUE
- Billing: $/hour, $/request, $/GB