BDP1

Martina Castellucci

Glossary

This glossary summarizes key terms and technologies discussed throughout the BDPI1 course, providing concise definitions and context for quick reference.

Big Data Extremely large datasets that require advanced techniques for storage, processing, and analysis due to their Volume, Variety, Velocity, Veracity, and Value (5Vs).

Structured Data Data organized in predefined formats such as tables (e.g., SQL databases).

Semi-structured Data Data with some organizational properties but not fixed schema (e.g., JSON, XML).

Unstructured Data Raw data with no predefined structure (e.g., text, images, audio).

CPU Central Processing Unit: Executes instructions in computing systems.

GPU Graphics Processing Unit: Optimized for parallel processing, useful in AI and simulations.

RAM Random Access Memory: Temporary memory used to store active processes.

Cache Small high-speed memory close to the CPU used for frequently accessed data.

Virtual Memory Hard disk space used as extension of RAM when physical RAM is full.

Hyperthreading Intel technology allowing a single physical core to act as two logical cores.

Latency Time delay between data request and response.

Bandwidth Amount of data transferred in a given time.

LAN Local Area Network, typically confined to a building.

WAN Wide Area Network, covering large geographical areas.

MAC Address Hardware identifier for a network interface card (NIC).

IP Address Unique address assigned to a device in a network.

DNS Domain Name System: Resolves domain names to IP addresses.

RAID Redundant Array of Independent Disks: Combines multiple disks for redundancy/performance.

Filesystem Organizes and manages data on storage media.

DAS Direct Attached Storage: Physically connected storage to a computer.

NAS Network Attached Storage: Shared storage accessed over a network at file level.

SAN Storage Area Network: Block-level network storage shared across multiple servers.

DFS Distributed File System: Spans files across multiple machines but appears as unified.

POSIX Standard interface for Unix-like file operations.

Parallel File System High-performance file access across multiple nodes (e.g., GPFS, Lustre).

Tiered Storage Storage strategy using devices with different cost/speed (e.g., SSD vs HDD).

PUE Power Usage Effectiveness: , lower is better.

Docker Platform for containerized applications using images and layers.

Docker Compose Tool to define and run multi-container Docker apps via YAML.

Virtualization Technology allowing multiple OSs to run on one machine.

Hypervisor Software creating and managing virtual machines (Type 1 or Type 2).

Cloud Computing Remote, scalable computing services delivered over the internet.

IaaS Infrastructure as a Service: Provides virtualized hardware.

PaaS Platform as a Service: Includes OS, runtime, and environment.

SaaS Software as a Service: End-user applications hosted in the cloud.

AWS Amazon Web Services: Popular cloud platform offering compute, storage, and tools.

EC2 Elastic Compute Cloud: AWS service for virtual machines.

S3 Simple Storage Service: Object storage in AWS.

Lambda Serverless AWS compute service triggered by events.

HTCondor Job scheduling system for distributed computing environments.

ClassAds Condor mechanism for matching job/resource requirements.

Edge Computing Computation done at/near data source (e.g., IoT).

Fog Computing Adds local nodes between cloud and edge for intermediate processing.

Digital Twin Virtual replica of a physical system for simulation and monitoring.

NUMA Non-Uniform Memory Access: Memory access time varies by CPU location.

1 COMPUTATIONAL CHALLENGE

goal: find substrings within very large strings.

key tools:

- brute force
  loops comparing one position at a time
- BLASTn
  efficient in small dataset and used for comparing divergent sequences
  calculates also statistical significance
- BWA
  faster than BLASTn but works on low divergence.

BWA outputs:

- .sai (aln): not human-readable
- .sam (samse): SAM format, human-readable

2 INTRODUCTION TO BIG DATA

Big data: large and complex dataset that traditional software tools are insufficient for processing.

Types:

- Structured (more for traditional data)
- Semi-structured
- Unstructured

5V:

- Volume
- Velocity
- Variety
- Veracity

- Value

Analysis on Big Data:

- Descriptive
- Predictive
- Prescriptive

3 FROM PC TO DATACENTER

goal: understand how computing infrastructure scales from a PC to a datacenter.

BIT (b):

Most basic unit that can have two values (0, 1). It's the fundamental language.

BYTE (B):

1 B = 8 b (e.g., a single character).

PROCESSING UNITS

- CPU (Central Processing Units)
  Main processor (brain). Executes instructions by working sequentially.
  - Arithmetic logic unit (ALU): for arithmetic and logical operations
  - Processor registers: store data temporarily
  - Control unit: coordinates the operations of alu, registers and other components.

  The amount of data it processes at once depends on its bit-width (e.g., 32-bit, 64-bit).
- GPU(Graphics Processing Unit)
  Originally created for graphics rendering but now used for parallel computations. Ideal for image processing, machine learning and other high-performance tasks.

- MULTI-CORE CPU Contains two or more separate processing units (cores), independent so they work simultaneously.

  Cache memory → individual memory for each cpu core.

STORAGE SYSTEM

Memory hierarchy:

- Caches (L1-3): faster but smaller
  L1: smallest and fastest, located inside each core holds the most frequently accessed data and instruction CPU is working on.
  L2: located inside each core, stores less frequently used data.
  L3: slowest, scared among all cores. Holds data that might be needed by any of the CPU cores. Still faster than RAM.

- RAM: main memory, slower but larger (temporary, stores data CPU is actively working on).
  Used when data are not found in cache.

- Virtual memory (non-volatile) uses part of the hard drive when RAM is full, but it's much slower

Non-volatile memory:

- Hard drives (HDD): cheap and slow

- Solid-state drives (SSD, NVMe) are much faster due to their lack of moving parts.

- Tapes are used mainly for backups and archiving because they're cheap and energy-efficient, but very slow.

To increase reliability and speed, disks can be combined in RAID systems.

Hyper-Threading Technology (HTT)

Intel's implementation that enables each physical CPU core to function as two virtual cores, to execute two threads simultaneously on a single core.

Limitations → potential degradation can occur due to threads competing for limited cache resources. Solution → thread scheduling and resource management.

SYSTEM MEMORY

- LATENCY

  Latency → time it takes for the CPU to access data or perform an operation (clock cycles).

  Disk seek time → delay in HDD to move in the correct location on the disk patters where required data is stored.

  Round-Trip Time (RTT) → total time from source to destination and return to source. Measured by the tool ping.

  One-Way Delay (OWD) → half of RTT, measures the delay in one direction assuming it's the same.

  Latency focuses on measuring delay.

- MEMORY BANDWIDTH

  Rate at which a processor can read from or write to memory (bps).

  Memory bandwidth focuses on the amount of data transferred per second.

  Calculated multiplying:

  - base clock frequency
  - data transfer per clock cycle
  - memory bus width
  - number of interfaces

- VIRTUAL MEMORY

  It expands the usable memory by using a proportion of the hard drive (HDD) as temporary storage when physical RAM is full.

  It is persistent but slower.

NETWORK

goal: share resources between nodes

Data transmission → physical or wireless (bps)

Network nodes → devices responsible for originating, routing and terminating. Identified by address.

- OSI MODEL

  Conceptual model that standardizes how different parts of a computer network communicate by breaking down the process of data transmission into seven district layers.

  It defines different transport protocols classes, for example:

  - Class 0 (TP0): fewest features, no error recovery. Used when errors are rare.
  - Class 4 (TP4): adds error detection and retrotransmission.

- LAN VS WAN

  Types of network

  LAN → localized, used for smaller network. Faster and more secure.

  WAN → for larger areas, connect multiple LANs via public or private connection.

- PACKETS

  For more efficient transmission. 2 main components:

  - Control information: info to reach correct destination
  - Used data: data being transmitted

  Maximum transmission unit (MTU) → largest size of a single packet that can be sent in one go.

- MAC ADDRESS

  Media Access Control Address → unique identifier for the network card.

  Important in layer 2.

  Format → 6 pairs of decimal assigned by network card's manufacturer. Stored in network card's hardware.

- IP ADDRESS

  Internet Protocol → numerical label assigned to each device. Useful for identification and location.

  Dotted-decimal notation of 32 bit divided in 4 sections (octets 0-255).

  Subnet → way to divide a larger network in smaller parts.

  Subnet mask → way to determine which part of an IP address is for the network and which is for the host.

  CIDR (Classless Inter-Domain Routing) → notation that simplifies the address (IP/n where n is how many bits of the address are used for network).

  - Public: unique across the world and visible from the internet
  - Private: used within local networks

  Domain Name System (DNS) → system used to map domain names (example.com) to their corresponding IP.

- NETWORK PROTOCOLS

  Set of rules that govern how devices communicate.

  Organized in layers where each layer depends on the layer beneath it to function.

- HUB-SWITCH-ROUTER

  HUB → device used to connect multiple devices in a LAN (layer 2/3).

  When a data packet arrives at one port of the hub, it is duplicated and

sent to all other ports (every device receives the same data).

SWITCH → Intelligent manage data transmission.

ROUTER → device that connect different networks ensuring data packets reach the correct destination (layer 3).

In data centers, servers are arranged vertically in racks. Top-of-the-Rack (ToR) switching place a network switch at the top pf each rack to handle all internal and external connection.

Advantages of ToR:

- better organization
- reduce cable clutter
- enhanced performance (reducing latency and improving network speed)

COMPUTING FARM

Group of servers (nodes) working together to perform complex computing tasks. These nodes are interconnected through switches and routers for internal and external communication.

- BATCH SYSTEM

  When the number of nodes is too high, a batch system is used to manage job execution following some consideration.

  Users submit their job to batch system rather than accessing servers directly.

  It manages priority and assigns jobs to the most suitable nodes.

  The results are then returned to the user.

  Benefits → efficient usage (optimize computer resources), share and priority management.

  Types of jobs:

  - Single Batch Job.

    Simple job using one resource (tipically a single CPU core).

- DAG Workflow (Directed Acyclic Graph)

  A sequence of dependent jobs, ideal for tasks that require a specific order.

- Collection

  A group of similar but independent jobs executable in parallel.

- Parametric Jobs

  A collection where each job differs by a parameter.

- Parallel Jobs

  Jobs that need multiple CPU cores simultaneously.

BASH COMPUTING → Users submit tasks that run in the background (batch jobs). These don't need the user to interact while running.

- FAIR SHARE SCHEDULING

  Ensures that computer resources are distributed fairly among users.

  In contrast to First Come First Served (FCFS) scheduling, fair share scheduling allows dynamic rebalancing based on usage and priority.

  Key → dinamic priority.

  RESERVATION → reserve resources in advance for a big job that will start later. But it can cause idle time if no jobs are allowed in those cores.

  BACKFILL → A technique to improve efficiency by allowing smaller jobs to run in reserved slots, if they can finish before the big job starts.

- JOB COMPONENTS

  Key components:

  Job type

  Prologue → preparation phase before the main job starts (e.g., downloading data).

  Input Sandbox → contains all input files needed for the job's execution.

Requirements → describes what resources the job needs.

Executable → script or program that performs the actual job.

Standard Output/Error (stdout/stderr).

Output Sandbox → stores files generated by the job.

Epilogue → handles tasks after the job ends (e.g., cleaning up temporary files).

Error Recovery → specifies what happens if the job fails.

STORAGE

Types:

Spinning Disks → traditional, with magnetic platters that spin to process data. Slower than modern alternatives.

Solid-State Drives (SSDs) → no moving parts. Faster and more reliable.

Non-volatile.

NVMe (Non-Volatile Memory Express) → A type of SSD that uses a faster protocol.

Tapes → used for long-term data archiving and backups.

Slower for random access as data is stored sequentially High capacity but low performance.

Metrics:

Bandwidth

IOPS (Input/Output Operations Per Second) → Measures how many read/write operations per second for small or random dataset.

SSDs and NVMe drives offer much higher IOPS than HDDs.

Capacity → Total data a device can store.

- RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS)

  Storage that combines multiple physical hard drives into a single system

to improve performance and reliability.

Benefits:

- Data redundancy
- Improved performance
- Dual purpose (performance and reliability)

RAID 0:

- Speed Boost: splits data across disks for faster transfers.
- Storage: use full combined capacity of all disks.
- No redundancy

RAID 1:

- Data Backup: creates exact copies on two or more disks.
- Faster Read: data can be read from either disk.
- Redundant: survives one disk failure without data loss.

RAID 5:

- Balanced Speed + Safety: Spreads data and parity (error-checking code) across all disks.
- Minimum Disks: Requires at least 3 disks.
- Fault Tolerant: If one disk fails, data can be rebuilt from the remaining disks.

- FILE SYSTEM

  Software technology that organizes, stores, and accesses data on storage devices like HDDs or SSDs. It keeps files in order, tracks storage space, and ensures data can be reliably stored and retrieved.

  Functions:

  - Space management
  - Unique filenames

  - Directories
  - Metadata (stores important info of each file such as permission)

  Posix File Systems → standardized method that defines how file operations should behave across different systems.

  Benefits:

  - Uniform set of rules
  - Cross-System Compatibility
  - Commands and Tools are standard

- DAS (DIRECT ATTACHED STORAGE)

  Storage system connected directly to a computer via cable. Not sharable over a network.

  Block level access → the system interacts directly with the raw data blocks. Data is stored and accessed in small units called blocks. This allows low-level, fast access to the storage, but doesn't support shared use or file-level structure by default.

- NAS (NETWORK ATTACHED STORAGE)

  Storage system that connects to a network, allowing multiple users/devices to access it.

  It provides file-level access, meaning you interact with folders and files just like with a local drive.

  It often uses RAID to protect data if one disk fails (redundancy).

  Offloading → NAS handles file storage, freeing up other servers for their main tasks.

- SAN (STORAGE AREA NETWORK)

  Network system that stores and accesses large amount of data quickly, connecting multiple storage devices directly to multiple servers using a

specialized network.

Architecture:

- Host layer: contains the servers. Each server uses an HBA (Host Bus Adapter) to connect to the SAN and communicate with storage devices.
- Fabric layer: includes networking hardware like switches and routers.
- Storage layer: contains the storage devices (such as hard drives) often organized in RAID for data protection and fault tolerance.

Lun:

Logical Unit Number → unique identifier for each storage devices, to organize storage, track data and control access.

Tan:

Tape Area Network → part of a SAN that uses tape storage.

TAN connects tape drives/libraries to servers and is optimized for managing large archival data volumes.

- PARALLEL FILE SYSTEM

  Store data across multiple networked servers, allowing multiple clients to read and write simultaneously.

  Key concept:

  - Global Namespace: users don't need to know where data is physically stored
  - Metadata Server: store info to coordinate access

  Key features:

  - High availability
  - Mirroring: copies data across server to prevent data loss
  - Snapshots: capture data for backup or recovery

  Benefits:

  - Scalability: easily add storage or speed without stopping the system.
  - Access to large files
  - Simultaneous access

  IBM Spectrum Scale (GPFS) → used in enterprises and research to manage large datasets.

  LUSTREc → common in HPC (high-performance computing) for handling massive data and parallel tasks.

- DISTRIBUTED FILE SYSTEM (DFS) It allows users to access files stored across multiple servers in a network, but it makes them appear as if they're stored locally on your own computer.

  Key features:

  - Global Namespace
  - Local Interfaces: users interact with remote files just like they would with local ones.
  - Network Protocols: used to manage communication between machines.

  Advantages:

  - Scalability
  - Fault Tolerance
  - Efficiency: users interact with remote files just like local ones, no delays and no complexity.

  Google File System (GFS) → used by Google to store vast amounts of data across thousands of machines (Gmail).

  Hadoop Distributed File System (HDFS) → common in big data projects, machine learning, and analytics — great for managing huge datasets.

- TIERED STORAGE

  Organizes data into different levels (tiers) based on different attributes.

  Attributes:

  - Price: faster storage costs more
  - Performance: refers to data access speed
  - Capacity: total data that can be stored
  - Function: different tiers are used for different tasks

  SSD vs. HDD → SSDs: higher tiers (fast, costly); HDDs: lower tiers (slow, cheap).

  Benefits:

  - Cost optimization: use cheaper storage for less important data
  - Better performance: use fast storage for high-priority data
  - Quality of Service (QoS): assign storage based on the importance and urgency of the data.

  Migration → moving files to faster storage (disk) to slower long-term storage (tape).

  Recall → the opposite, retrieving a file when it's needed.

  QoS → guarantees the right speed and access for each application, using metrics like IOPS. It balances Performance and reliability.

MODERN DATACENTER FOR BIGDATA APPLICATION

Modern data centers are evolving to support Big Data applications by combining CPU, storage, and networking into a single, unified system.

Key features:

- Hyper-Converged Architecture: combines everything in a single system, simplifying setup and menagement of data center infrastructure.
- Scalability

Modern data centers are excellent for rapid data analysis and high-volume data handling.

Limitations → Not all scientific research tasks work well with hyper-converged systems, but may require traditional architectures.

PROVISIONING AND MONITORING

Modern data centers manage thousands of servers. To simplify this, administrators use automation tools like Foreman and Puppet.

Foreman → includes a web interface for easy monitoring all servers. Works well with tools like Puppet for configuration consistency. Manages servers from creation to maintenance.

Puppet → maintains consistent server configurations. Uses a Puppet master to define the desired state, and Puppet agents on servers to report and apply changes.

Monitoring → tracking system performance (CPU, memory, disk usage).

Alarming → triggered by events like crashes, ensures fast responses before issues get worse

Event handling → takes automatic actions when alarms are triggered, minimizing downtime and restoring normal operations with minimal manual work.

POWER AND COOLING

Power infrastructures → supply electricity to all components in a data center.

Sources:

- UPS (Uninterruptible Power Supply): backup power source to prevent downtime
- Battery-based Systems: provide instant power
- Inertial Generators: use stored energy
- Engine Generators: use fuel to generate power

Cooling infrastructures → keeps the data center at safe temperatures and avoids equipment overheating.

Methods:

- Free Cooling: outside air
- Forced Air Flow: blow air over equipment
- Liquid Cooling
- Liquid Submersion
- Heat Pipes: transfer heat using phase-changing liquid.

- PUE (POWER USAGE EFFECTIVENESS)

  PUE is a metric that measures how efficiently a data center uses energy.

  $$PUE = \frac{TotalFacilityEnergy}{ITEquipmentEnergy} \qquad PUE = 1 + \frac{Non - ITFacilityEnergy}{ITEquipmentEnergy}$$

  Values:

  - PUE = 1.0: Ideal (100% of energy goes to IT equipment)
  - PUE greater 1.0: Real-world scenario (some energy used for cooling, lighting, etc.)

  Why it matters:

  - Cost: high PUE = higher electricity bills
  - Inefficiency: indicates outdated infrastructure or energy waste
  - Environmental Impact: more energy = more emissions
  - Improvement: helps identify ways to increase energy efficiency

  Related metric → DCIE (Data Center Infrastructure Efficiency), inverse of PUE.

4 Virtualization

Virtualization enables the abstraction of hardware resources, allowing multiple operating systems to run concurrently on the same physical hardware. It is a core component in cloud infrastructures, resource management, and testing environments.

4.1

Types of Virtualization

- **Full Virtualization**: Simulates the entire hardware stack. The guest OS runs unmodified, unaware of being virtualized. Requires significant overhead. *Example: VMware, QEMU*
- **Paravirtualization**: The guest OS is modified to communicate with the hypervisor through hypercalls, improving efficiency. *Example: Xen*
- **Hardware-assisted Virtualization**: Uses CPU extensions (e.g., Intel VT-x, AMD-V) to provide isolation and control. Reduces overhead compared to full virtualization. *Example: KVM (Kernel-based Virtual Machine)*

4.2

Hypervisors

- **Type 1 (bare-metal)**: Installed directly on the host's hardware. Offers better performance and security. *Examples: VMware ESXi, Xen, Hyper-V*
- **Type 2 (hosted)**: Runs on top of a conventional OS, used for testing or development. *Examples: VirtualBox, VMware Workstation*

4.3

Advantages of Virtualization

- Higher resource utilization through hardware consolidation
- Greater isolation and fault tolerance between virtual machines

- Support for snapshots and quick rollback
- Simplified disaster recovery and failover
- Enables safe and isolated software testing environments

4.4

Examples of Use

- Running multiple OS instances for testing on a developer machine
- Consolidating multiple services on a single physical server in data centers
- Live migration of VMs to balance workloads (e.g., vMotion, virsh migrate)

5 Containers and Docker

Containers are lightweight, isolated environments that run applications and include all necessary dependencies, libraries, and configuration files. Unlike virtual machines, they share the host operating system kernel.

Containers vs Virtual Machines

- **Isolation Level**: VMs virtualize hardware; containers virtualize the OS.
- **Startup Time**: Containers launch in seconds; VMs take longer.
- **Resource Usage**: Containers use fewer resources due to lack of full OS.
- **Portability**: Containers are easily portable across environments.
- **Performance**: Containers offer near-native performance in many scenarios.

Docker Architecture

- **Docker Engine**: The core client-server architecture that runs containers.
- **Docker Images**: Immutable, read-only templates used to create containers.
- **Docker Containers**: Runnable instances of images that encapsulate an application and its environment.
- **Dockerfile**: A script that defines the steps to build a Docker image.
- **Docker Hub**: A public registry for sharing images.
- **Volumes**: Used for persistent storage.
- **Networks**: Isolate communication between containers.

Example Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Benefits of Docker

- Portability across systems and environments ("build once, run anywhere")
- Consistency between development, testing, and production
- Isolation of services (useful for microservices architecture)
- Integration with CI/CD pipelines
- Efficient resource usage

Use Case in Bioinformatics

Bioinformatics tools often require complex dependencies. Docker allows researchers to package tools like BLAST, BWA, or Bowtie with all dependencies, ensuring reproducibility and reducing installation time.

6 Edge and Fog Computing

Edge and fog computing are emerging paradigms designed to move computation and data processing closer to the data source (e.g., IoT devices), reducing latency and network load compared to centralized cloud computing.

Edge Computing

- Processes data at or near the data source (e.g., sensors, gateways).
- Minimizes latency by eliminating the need to send all data to the cloud.
- Reduces bandwidth usage and enhances real-time responsiveness.

Examples: smartwatches, self-driving vehicles, manufacturing robots.

Fog Computing

- Introduces intermediate nodes (fog nodes) between edge and cloud.
- Performs storage, compute, and analytics at the network edge.
- Acts as a bridge between local devices and remote data centers.

Example: A smart city infrastructure where fog nodes aggregate traffic data before sending summaries to the cloud.

Edge vs Fog: Comparison

- **Edge**: Processing happens directly on the device or gateway.
- **Fog**: Involves intermediate layers (fog nodes) for preprocessing.
- **Latency**: Both reduce latency; edge is generally lower.

- **Architecture**: Fog computing offers a more scalable and manageable structure for larger networks.

Benefits

- Low latency for time-sensitive applications (e.g., health monitoring)
- Bandwidth optimization through local processing
- Enhanced data privacy and security by limiting cloud dependency
- Offline processing capabilities in case of network failure

7 HTCondor and Batch Scheduling

HTCondor is a specialized workload management system designed for compute-intensive jobs. It is commonly used in academic and research environments to efficiently manage and schedule batch jobs across a distributed pool of compute resources.

Architecture and Components

- **Submit Node**: The node where users write and submit job scripts.
- **Execute Node**: The machines where the jobs are executed.
- **Central Manager**: Responsible for resource matchmaking and global scheduling decisions.
- **ClassAds**: Both jobs and resources are described using "classified advertisements" to match requirements and capabilities.

Job Types in HTCondor

- **Vanilla**: Runs standard executable scripts without modification.
- **Standard**: Requires linking with HTCondor's I/O libraries, enabling features like remote I/O.

- **MPI Jobs**: Supports parallel execution using MPI for high-performance computing.
- **Grid Jobs**: HTCondor can submit to remote grid systems using Globus or ARC.

Job Submission Example

```
Executable = my_program.sh
Arguments  = input1.txt
Output     = output.log
Error      = error.log
Log        = job.log
Queue 100
```

This example submits 100 instances of a job (parametric job). Each instance can receive unique arguments using variables (e.g., ($Process$)).

Useful Commands

- `condor submit job.sub { Submit a job.`
- `condor q { Show jobs in the queue.`
- `condor status { Display available machines and their states.`
- `condor rm  { Remove a job from the queue.`
- `condor history { List completed jobs.`

Matchmaking Mechanism

`HTCondor uses a matchmaking system based on ClassAds:`

- `Jobs describe what they need (e.g., memory, cores, architecture).`

- Resources advertise their availability and constraints.
- The central manager matches job requirements to resources.

Scheduling Policies

- Fair Share Scheduling: Balances usage among users or groups.
- Priority-Based Scheduling: Jobs are ranked by user or submission time.
- Backfilling: Short jobs are allowed to run ahead in reserved slots if they don't delay longer jobs.

Example Use Cases

- Submitting a collection of genome alignments or simulations.
- Executing hundreds of independent bioinformatics pipelines in parallel.
- Running parametric sweeps for hyperparameter tuning in machine learning.

8 Amazon Web Services (AWS)

Amazon Web Services (AWS) is a leading cloud computing platform offering scalable, reliable, and cost-effective infrastructure and software services on demand. It supports a wide range of applications, from simple websites to complex machine learning pipelines.

Core Services

- EC2 (Elastic Compute Cloud): Provides resizable compute capacity in the cloud (virtual machines).
- S3 (Simple Storage Service): Scalable object storage with high durability and availability.

- RDS (Relational Database Service): Managed relational databases like MySQL, PostgreSQL, Oracle.
- Lambda: Serverless computing that runs code in response to events.
- IAM (Identity and Access Management): Securely controls access to AWS services and resources.

Key Features and Benefits

- Elasticity: Automatically scales resources up or down based on demand.
- Pay-as-you-go: Only pay for what you use, ideal for dynamic workloads.
- High Availability: Built-in redundancy across regions and availability zones.
- Global Reach: Infrastructure spans multiple geographic locations.
- Integration: Easily integrates with DevOps, CI/CD, and ML frameworks.

Launching an EC2 Instance

Steps to launch and access a virtual machine (EC2):

1. Log into AWS Management Console.
2. Choose an Amazon Machine Image (AMI), e.g., Ubuntu Server 20.04.
3. Select instance type (e.g., t2.micro for free-tier).
4. Configure instance details (network, IAM role).
5. Add storage volume.
6. Configure firewall (security group) rules.
7. Generate or upload SSH key pair.
8. Launch instance.

9. Connect via SSH:

   ```
   ssh -i mykey.pem ec2-user@
   ```

Using S3 for Storage

Upload and download files to AWS S3:

```
aws s3 cp mydata.txt s3://mybucket/
aws s3 cp s3://mybucket/mydata.txt ./
```

S3 buckets can be private or public, and support versioning, encryption, and lifecycle policies.

Lambda: Serverless Functions

Lambda allows users to execute code in response to events without managing servers:

- Upload code (e.g., Python script).
- Define triggers (e.g., API Gateway, file upload to S3).
- AWS runs the function on demand, charging only for execution time.
- Ideal for microservices, real-time processing, automation.

Academic and Research Use

- AWS offers credits for students and researchers.
- Enables cloud-based bioinformatics pipelines (e.g., genome assembly, transcriptomics).
- Facilitates large-scale machine learning and data analysis.

Quiz Questions and Answers

Question 1: HTC vs HPC

Question:
High-Performance Computing (HPC) focuses on solving complex computational problems that require significant processing power and are typically executed on supercomputers, while High-Throughput Computing (HTC) focuses on executing a large number of independent tasks over a longer period of time, often using distributed computing resources.
Answer: True
Explanation: HTC focuses on job throughput and independent tasks, while HPC targets tightly-coupled problems requiring high performance on supercomputers.

Question 2: Amdahl's Law

Question:
Which of the following statements are true regarding Amdahl's Law?

a) Amdahl's Law suggests that increasing the number of processors will linearly increase the system's efficiency.

b) Amdahl's Law is used to calculate the execution speed of a program on a single processor.

c) According to Amdahl's Law, the maximum speedup achievable is inversely proportional to the fraction of the program that can be parallelized.

d) Amdahl's Law states that the improvement in performance of a system is limited by the portion of the system that cannot be parallelized.

Answer: d
Explanation: Amdahl's Law shows that the serial portion of a program limits the maximum speedup achievable, even with infinite processors.

Question 3: Storage Area Network (SAN)

Question:
Which of the following statements are true regarding Storage Area Networks (SAN)?

a) SANs are typically used for file sharing and collaboration among users.

b) SANs provide block-level storage that can be accessed by servers over a network.

c) SANs offer high availability and redundancy features to ensure data integrity and uptime.

d) SANs are primarily designed for personal and small office use.

Answer: a, b, c
Explanation: SANs are enterprise-level block storage solutions that provide high performance, availability, and redundancy. They support shared storage across servers and are used in large-scale data centers.

Question 4: AWS

Question:
The AWS instances we created during the exercises...

a) Did not have IPs but just a DNS name.

b) Had a private IP and no public IP.

c) Had a public IP because private IP does not exist.

d) Had a public IP and no private IP.

e) Had a public IP and a private IP.

f) Had a public IP we used for reaching a web server, had a private IP we used for SSH to access them from our laptop.

Answer: e
Explanation: AWS EC2 instances can have both a private and public IP address. The public IP allows access from outside the VPC, while the private IP is used internally.

Question 5: Cloud Computing

Question:
What are the main characteristics of cloud-aware applications?

a) Stateful

b) Dependence on specific hardware

c) Monolithic

d) Stateless

e) There are no cloud aware applications; avoid using commercial clouds.

f) Horizontal scalability

g) Fail-over in the application

h) All applications are cloud-aware.

Answer: d, f, g
Explanation: Cloud-aware applications are stateless, support horizontal scalability, and have built-in failover mechanisms for resilience.

Question 6: IaaS

Question:
Which of the following statements are true regarding IaaS?

a) IaaS requires users to manage the underlying physical hardware.

b) IaaS typically provides the user with services such as virtual machines, storage, and networking.

c) IaaS provides virtualized computing resources over the internet.

d) The cloud provider installs the operating system.

e) IaaS eliminates the need for users to manage operating systems and middleware.

Answer: b, c, d
Explanation: IaaS provides on-demand infrastructure resources (compute, storage, networking). Although providers often install a base OS, users are responsible for configuring, securing, and maintaining it.

9

BDPI1 Exam Sample Questions The following is a list of 26 sample questions from the BDPI1 exam (70 minutes total). These questions are based on theoretical and practical content discussed during the course:

1. Differences between the batch system and cloud computing: Batch systems are used in HPC environments and schedule jobs on local clusters with fixed resources. Cloud computing provides on-demand access to scalable and virtualized resources via the internet. ‘‘‘

2. Cooling techniques of the datacenter: Include free cooling (outside air), forced air flow, liquid cooling, liquid submersion, and heat pipes to manage thermal loads and prevent hardware overheating.

3. Checksums: A checksum is a calculated value that verifies the integrity of data. Used in networking and storage to detect transmission or file corruption.

4. LAN: Local Area Network connects computers in a small area (e.g., office or lab), offering high speed and low latency.

5. Meaning of 0.0.0.0 in IP: Refers to all IPv4 addresses on the local machine, often used to bind a service to all available interfaces.

6. SAN (Storage Area Network): A high-speed network providing block-level storage accessible by servers. Used in large-scale enterprise environments.

7. DAS (Direct Attached Storage): Storage device directly attached to a single computer or server without a network.

8. PaaS (Platform as a Service): Cloud model that provides a platform (OS, runtime, tools) for developers to build applications without managing infrastructure.

9. Cloud-aware applications: Applications designed to leverage elasticity, scalability, and distributed nature of cloud infrastructure.

10. GRID: Distributed computing infrastructure enabling resource sharing across multiple administrative domains for complex tasks.

11. Efficiency (e.g., PUE): Efficiency is often evaluated using Power Usage Effectiveness (PUE = Total Energy / IT Energy). Lower values indicate better energy efficiency.

12. CMD in Docker: CMD defines the default command that runs when a container starts. It can be overridden when running the container.

13. Docker Compose: A tool for defining and running multi-container Docker applications using a YAML configuration file.

14. Edge and Fog Computing: Edge computing processes data at the device level; Fog introduces intermediate nodes between edge and cloud for processing.

15. Digital Twins: Virtual replicas of physical devices/systems used to simulate, monitor, and predict their behavior in real-time.

16. HPC vs HTC: High-Performance Computing (HPC) runs tightly coupled parallel jobs (e.g., simulations); High-Throughput Computing (HTC) runs many independent tasks over time (e.g., genome analysis).

17. Infrastructure diagram (e.g., NUMA): NUMA (Non-Uniform Memory Access) is a memory design where memory access time depends on the memory location relative to the processor.

18. Public vs Private IP address: Public IPs are globally unique and routable on the internet; Private IPs are used in local networks and not accessible from the internet. ‘‘‘

Note: For practical questions, students should be familiar with recognizing infrastructure configurations and distinguishing between networking elements like public/private IPs.

---

10 High Performance vs High Throughput Computing

High Performance Computing (HPC)

HPC systems are used to solve complex, tightly-coupled computational problems. These require large amounts of memory and processing power across multiple nodes simultaneously.

Examples:

- Climate modeling
- Computational fluid dynamics
- Protein folding simulations

Characteristics:

- Low-latency interconnects (e.g., Infiniband)
- Use of MPI (Message Passing Interface)
- Tight synchronization across nodes
- Requires scheduling via tools like SLURM or PBS

High Throughput Computing (HTC)

HTC is used to process large numbers of loosely-coupled tasks over long periods. Each job is typically independent.

Examples:

- Genome sequencing pipelines
- Data analysis of large experimental datasets
- Parametric studies in bioinformatics

Characteristics:

- Jobs are independent (embarrassingly parallel)
- Focus on maximizing job throughput over time
- Workload managed using HTCondor or GridEngine
- Can use heterogeneous and opportunistic computing resources

Comparison Table

Feature	HPC	HTC
Workload Type	Tightly coupled	Independent jobs
Scheduling	SLURM, PBS	HTCondor, GridEngine
Interconnect	Low-latency (e.g., Infiniband)	Not required
Performance Goal	Short execution time	High total throughput
Examples	Simulations	Large-scale analyses

11 Scalability, Elasticity, and Reliability

Scalability

Scalability refers to the system's ability to handle increased workloads by adding resources.

Types:

- Vertical Scalability (Scale-Up): Increasing capacity of a single machine (e.g., more CPU or RAM).

- Horizontal Scalability (Scale-Out): Adding more machines or nodes to a system.

Examples:

- Vertical: upgrading from t2.micro to t2.large on AWS EC2.
- Horizontal: adding more EC2 instances to handle increased web traffic.

Elasticity

Elasticity is the capability of a system to automatically adjust its resources (scale up or down) based on current demand.

Key features:

- Dynamic provisioning and de-provisioning
- Avoids over-provisioning and reduces cost
- Typically implemented with autoscaling groups in cloud environments

Example:

- A web application that automatically adds instances during peak hours and removes them at night.

Reliability and Availability

Reliability is the system's ability to function correctly and continuously over time without failure.

Availability is the percentage of time the system is operational and accessible.

High Availability (HA) is achieved through:

- Redundant components (e.g., multiple web servers)
- Load balancers to distribute traffic
- Automatic failover and backup systems

Service Level Agreements (SLA)

Cloud providers define availability targets in their SLA.
Example:

- SLA = 99.9% uptime → maximum 43 minutes downtime per month

Comparison Table

Concept	Description
Scalability	Ability to handle increased load by adding resources
Elasticity	Dynamic adaptation to changing demand
Reliability	Consistent correct operation over time
Availability	Probability that system is up and running
High Availability	Design strategy to minimize downtime

12 Practical Example: HTCondor Job Submission

HTCondor is ideal for submitting many independent jobs (e.g., sequence analysis, simulations).

Goal

Submit 10 jobs in parallel, each executing a script with different input parameters.

Step 1: Prepare the Bash script

```
#!/bin/bash
echo "Running job with parameter: $1"
sleep 5
```

Save it as ‘run.sh‘ and make it executable:

```
chmod +x run.sh
```

Step 2: Create HTCondor submission file

```
Executable   = run.sh
Arguments    = $(Process)
Output       = output_$(Process).out
Error        = error_$(Process).err
Log          = log_$(Process).log
Queue 10
```

Save this as ‘job.sub‘.

Step 3: Submit to HTCondor

```
condor_submit job.sub
```

Result

HTCondor will launch 10 jobs, each with a different argument from 0 to 9. Output files will be generated for each job instance.

Useful commands

- condor_q | see job queue
- condor_status | see node availability
- condor_rm <jobID> | remove a job
- condor_history | check finished jobs

13 Practical Example: Docker for Bioinformatics

Docker simplifies software installation, ensuring portability and reproducibility.

Goal

Create a container to run a Python script using BioPython.

Step 1: Create project structure

```
bio-docker/
 Dockerfile
 requirements.txt
 script.py
```

Step 2: Write Python script (‘script.py‘)

```
from Bio.Seq import Seq
seq = Seq("AGTACACTGGT")
print("Reverse complement:", seq.reverse_complement())
```

Step 3: Define requirements (‘requirements.txt‘)

```
biopython
```

Step 4: Create the Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "script.py"]
```

Step 5: Build and Run the Container

```
docker build -t bioinfo .
docker run --rm bioinfo
```

Expected Output

```
Reverse complement: ACCAGTGTACT
```

Benefits

- Eliminates dependency conflicts

- Portable across systems
- Quick to deploy in cloud or clusters

14 Udocker: Containers Without Root Privileges

Udocker is a tool that allows users to run Docker containers in user space, without requiring root access or Docker daemon.

Why Udocker?

- Most HPC systems disallow root access and Docker daemon for security.
- Udocker enables execution of containerized applications in shared environments like clusters and supercomputers.
- Useful for researchers and students working on HPCs with user-level permissions.

Key Features

- No root privileges required
- Emulates container behavior using PRoot, runC, or Fakechroot
- Compatible with standard Docker images
- Portable: can be installed in home directory

Installation

```
curl -O https://raw.githubusercontent.com/indigo-dc/udocker/master/udocker.py
chmod u+x udocker.py
./udocker.py install
```

Basic Usage

```
udocker pull python:3.9-slim
udocker create --name=mycontainer python:3.9-slim
udocker run mycontainer
```

Limitations

- No full OS virtualization
- Reduced isolation compared to Docker
- Performance slightly lower in some cases

Typical Use Case in Bioinformatics

- Running complex tools (e.g., BWA, GATK, BLAST) without compiling or installing dependencies manually.
- Ensures reproducibility across environments (cluster, workstation, laptop).

15 Storage Comparison: SAN vs NAS vs DAS

Storage systems can be classified based on access type, architecture, and use case.

Comparison Table

Feature	DAS	NAS	SAN
Access Type	Block	File	Block
Networked?	No	Yes	Yes
Protocol	SATA/SCSI	NFS, SMB/CIFS	iSCSI, Fibre Channel
Use Case	Single server	File sharing	High-speed apps
Scalability	Low	Medium	High
Cost	Low	Medium	High
Performance	Good (local)	Moderate	High
Example	External HDD	Synology NAS	Enterprise database backend

Summary

- DAS (Direct Attached Storage): Attached directly to one host. Fast and simple but not sharable.
- NAS (Network Attached Storage): File-based access shared over a network. Good for multiple users and backups.
- SAN (Storage Area Network): Enterprise-grade, high-speed block access across network. Ideal for databases and VMs.

16 Advanced Distributed File Systems

Distributed File Systems (DFS) allow data storage across multiple servers, while appearing as a unified storage space.

Google File System (GFS)

- Proprietary system developed by Google.
- Optimized for large files and write-once, read-many workloads.
- Stores files as chunks (64 MB), each replicated on multiple chunk servers.
- Central metadata managed by a master node.

Use Case: Search indexing, Gmail attachments, YouTube video storage.

Hadoop Distributed File System (HDFS)

- Open-source DFS from Apache Hadoop ecosystem.
- Similar to GFS in architecture (NameNode + DataNodes).
- High fault tolerance and suitable for batch processing of large datasets.
- Optimized for streaming reads of big files rather than random access.

Use Case: Big data analytics, machine learning pipelines.

CephFS

- Fully POSIX-compliant, distributed file system.
- Provides block, file, and object storage.
- Uses CRUSH algorithm to distribute data with no single point of failure.
- Decentralized metadata and high scalability.

Use Case: Scientific computing, HPC, OpenStack integration.

Comparison Summary

Feature	GFS	HDFS	CephFS
Open Source?	No	Yes	Yes
Chunk Size	64 MB	128 MB (default)	Dynamic
Metadata Mgmt	Centralized Master	NameNode	Decentralized
POSIX-Compliant	No	No	Yes
Target Use	Web scale	Big Data	HPC, Cloud

17 Monitoring, Logging, and Alarming

Modern datacenters and cloud infrastructures require constant monitoring to ensure performance, availability, and security.

Monitoring

Tracks key system metrics over time:

- CPU usage
- RAM and disk usage
- Network bandwidth
- IOPS and latency

Prometheus:

- Open-source monitoring system
- Pull-based model: scrapes metrics from exporters
- Includes built-in alert manager

Visualization and Dashboards

Grafana:

- Visualizes metrics from Prometheus (and others)
- Custom dashboards for real-time observability
- Supports threshold-based alerting

Log Management

Why? Logs help detect issues, audit operations, and trace system behavior.

Common Tools:

- journalctl, syslog | local log viewers
- ELK Stack (Elasticsearch, Logstash, Kibana) | powerful log aggregation and visualization

Alarming and Event Handling

Nagios:

- Real-time infrastructure monitoring
- Defines hosts/services to check and alert policies
- Sends email, SMS, or script-based alerts on failure

Event Handling:

- Detect issue (e.g., CPU overload)
- Trigger alarm (e.g., via Prometheus AlertManager)

- Automatic action (e.g., restart service or migrate VM)

Goals:

- Minimize downtime
- Enable proactive maintenance
- Ensure SLA compliance

BDPI1 Recap and Key Concepts

Big Data Fundamentals

- 5V: Volume, Variety, Velocity, Veracity, Value
- Data Types: Structured, Semi-structured, Unstructured
- Analysis: Descriptive, Predictive, Prescriptive

Computing Architecture

- CPU vs GPU: CPU handles sequential logic, GPU handles parallel tasks
- Memory Hierarchy: Cache (L1-L3) $\rightarrow$ RAM $\rightarrow$ Virtual Memory
- Latency vs Bandwidth: Time delay vs transfer rate
- Hyperthreading: Two threads on one core

Networking Basics

- LAN vs WAN: Local vs Wide area networks
- IP Addressing: Public, Private, Subnet, CIDR
- Protocols: TCP/IP, DNS, OSI layers
- Hardware: Hub, Switch, Router, ToR

Storage Systems

- DAS: Direct, local storage (block-level)
- NAS: Networked file storage (file-level)
- SAN: High-speed block-level shared storage
- RAID Levels: RAID 0 (speed), RAID 1 (mirroring), RAID 5 (speed + redundancy)
- File Systems: POSIX, GPFS, LUSTRE, HDFS, Ceph

Batch Computing with HTCondor

- Job Types: Vanilla, Standard, MPI, Grid
- Scheduling: Fair Share, Priority, Backfilling
- Job Files: Executable, Arguments, Sandbox, Log
- ClassAds: Matchmaking for resources and jobs

Virtualization

- Types: Full, Paravirtualization, Hardware-assisted
- Hypervisors: Type 1 (bare metal), Type 2 (hosted)
- Benefits: Isolation, Snapshots, Resource sharing

Containers and Docker

- Concept: OS-level virtualization, portable and lightweight
- Components: Dockerfile, Images, Volumes, Networks
- Tools: Docker Compose for orchestration
- Use Case: Bioinformatics pipelines (e.g., BLAST, BWA)

Udocker

- Purpose: Run containers without root or Docker daemon
- Usage: Useful in HPC environments

Edge and Fog Computing

- Edge: Computation at data source
- Fog: Intermediate layer before cloud
- Advantages: Low latency, bandwidth savings, local privacy

Cloud and AWS

- Models: IaaS, PaaS, SaaS
- AWS Services: EC2, S3, Lambda, IAM
- Features: Elasticity, High Availability, Pay-as-you-go

HPC vs HTC

- HPC: Tightly coupled, low-latency, parallel jobs (e.g., simulations)
- HTC: Many independent jobs over time (e.g., bioinformatics pipelines)

Monitoring and Provisioning

- Tools: Prometheus (metrics), Grafana (dashboard), Nagios (alerts)
- Automation: Puppet, Foreman
- Alarming: Event handling and system recovery

Power and Cooling

- Power: UPS, Generators, Battery systems
- Cooling: Free cooling, Forced air, Liquid immersion
- PUE: Power Usage Effectiveness (ideal = 1.0)

BDPI1: Logical Pipeline of Concepts

1. Big Data Foundations
    - Understand 5Vs and data types (structured, semi-structured, unstructured)
    - Define processing challenges and analysis types
2. From Single Machine to Datacenter
    - Learn architecture: CPU, GPU, RAM, Cache, Memory hierarchy
    - Scale from PC to multi-node infrastructure
3. Storage Systems
    - Classify DAS, NAS, SAN
    - Understand RAID levels and POSIX file systems
    - Learn about Distributed FS (HDFS, Ceph, GPFS)
4. Networking and Communication
    - Explore IP addressing, MAC, DNS, OSI model
    - Distinguish LAN/WAN, switches, routers, ToR
5. Batch and Parallel Computing
    - Learn how HTCondor manages jobs
    - Use scheduling strategies (FCFS, fair share, backfilling)
    - Submit jobs (vanilla, parametric, DAG workflows)
6. Virtualization and Containers
    - Understand virtualization: Type 1/2 hypervisors
    - Use containers for reproducible environments (Docker, Udocker)
7. Cloud Computing and AWS
    - Learn IaaS, PaaS, SaaS
    - Use AWS services: EC2, S3, Lambda, IAM

- Build scalable, elastic, highly available applications

8. Edge and Fog Computing
   - Reduce latency with edge/fog models
   - Apply in real-time systems and IoT contexts
9. Monitoring and Provisioning
   - Automate infrastructure setup (Foreman, Puppet)
   - Monitor with Prometheus, visualize with Grafana
   - Trigger alarms and auto-recovery (Nagios, AlertManager)
10. Energy, Cooling and Efficiency
    - Understand datacenter power systems (UPS, generators)
    - Cooling methods (air, liquid, immersion)
    - Use PUE to evaluate efficiency
11. Integration and Exam Preparation
    - Combine Docker + Condor + Cloud + FS in workflows
    - Prepare using quizzes, command-line examples, diagrams
    - Focus on HPC vs HTC, cloud-aware design, and reproducibility