Big Data Processing Infrastructures: Comprehensive Guide

Martina Castellucci

May 30, 2025

Contents

Glossary of Definitions

CPU: Central Processing Unit — the main processor responsible for executing instructions.

GPU: Graphics Processing Unit — accelerates parallel tasks like rendering and scientific computation.

RAM: Random Access Memory — fast, volatile memory used for immediate data access.

Cache: Small, fast memory (L1, L2, L3) that stores frequently used data to speed up processing.

Kernel: Core component of the OS that manages system resources, hardware communication, and processes.

Operating System (OS): Software managing hardware, system resources, and application execution.

File System: Organizes and manages files on storage devices.

Cloud Computing: On-demand delivery of computing resources over the internet.

IaaS: Infrastructure as a Service — cloud model offering virtualized hardware (e.g. servers, storage).

PaaS: Platform as a Service — provides an environment for application development and deployment.

SaaS: Software as a Service — cloud-based applications fully managed by providers.

Virtual Machines (VMs): Emulated computers running a guest OS on top of a host OS.

Containers: Lightweight, portable application environments sharing the host OS kernel.

NGS: Next-Generation Sequencing — high-throughput genomic sequencing technology.

FM-index: Data structure enabling efficient substring searching.

BWA: Burrows-Wheeler Aligner — a tool for mapping sequencing reads.

BLASTn: Tool for local alignments of nucleotide sequences.

HPC: High-Performance Computing — focuses on fast, tightly coupled computations.

HTC: High-Throughput Computing — focuses on processing many independent tasks.

HTCondor: A batch system for job scheduling and workload management.

Amdahl's Law: Principle describing the speedup limit of parallel processing.

Encryption: Data security through encoding to prevent unauthorized access.

Authentication: Verifying identity of users or systems.

Authorization: Defining access rights to resources.

Data Lifecycle: Managing data from creation to archiving or deletion.

SSH: Secure Shell — encrypted network protocol for remote system access.

LAN: Local Area Network — connects devices in a small geographic area.

WAN: Wide Area Network — connects devices across broader regions.

Infiniband: High-speed network technology used in HPC clusters.

Throughput: Amount of work performed by a system over time.

Latency: Time delay between request and response.

GFLOPS: Giga Floating Point Operations Per Second — performance metric for HPC.

Job Scheduling: Managing the order and allocation of computational tasks.

Workflow: Defined sequence of tasks with dependencies in data processing.

Virtualization: Creating virtual representations of resources (e.g. VMs).

Chapter 1

Computational Challenge

Definition

The computational challenge in Next-Generation Sequencing (NGS) involves finding a substring (read) in a long string (reference genome). This is known as the **String Matching Problem**.

Key Concepts

- **BLASTn**: Tool for aligning short reads to a reference sequence using local alignments.
- **BWA**: Uses Burrows-Wheeler Transform (BWT) and FM-index for efficient alignment.

Example

```
bwa index reference.fasta
bwa mem reference.fasta reads.fastq > alignment.sam
```

Review Questions

1. What is the String Matching Problem in NGS?
2. How does BWA differ from BLASTn?
3. What is the FM-index and why is it important?

Quiz

- **True or False:** BWA uses exhaustive search to align reads. (*False*)
- **Multiple Choice:** What is the output format of BWA-MEM?
  1. .bam
  2. .sam
  3. .vcf

  (*Correct answer: .sam*)

Links

- See GitHub BDP1_2025 for hands-on exercises and example notebooks.
- Refer to Chapter 5 for information on shared file systems required for data management.

Summary

The computational challenge in NGS lies in efficiently aligning millions of short reads to a reference genome. Traditional brute-force methods are too slow, so tools like BLASTn and BWA use advanced algorithms (e.g. FM-index) to optimize this task. Understanding the difference between real time and user time helps interpret performance measurements. Mastering these concepts is foundational to building scalable NGS pipelines and effective big data solutions.

Chapter 2

Big Data

Definition

Big Data refers to datasets so large and complex that traditional software tools cannot efficiently store, manage, or process them within a reasonable timeframe.

Key Concepts

- **Volume**: The size of the dataset.
- **Variety**: Different data formats and types.
- **Velocity**: The speed of data generation and processing.
- **Variability**: Data inconsistency affecting management.
- **Veracity**: Data quality and reliability.

Example

Analyzing real-time social media data streams requires handling massive volumes of unstructured and semi-structured data at high velocity.

Review Questions

1. What are the five V's of Big Data?
2. Why is unstructured data a challenge for Big Data analytics?

Quiz

- **True or False:** Big Data only refers to structured data. (*False*)
- **Multiple Choice:** Which of the following is not considered a Big Data characteristic?
  1. Volume
  2. Variety
  3. Value
  4. Virtualization

  (*Correct answer: Virtualization*)

Links

- See GitHub BDP1_2025 for Big Data exercises and practical examples.
- Refer to Chapter 3 for infrastructure details on datacenters and distributed storage.

Summary

Big Data encompasses the management and analysis of huge, varied, and fast-growing datasets. Its defining features are known as the 5 V's, which represent the scale, diversity, and complexity of modern data. Successful Big Data solutions must integrate multiple data sources, handle high-speed data flows, and address issues of quality and reliability. This chapter highlights the foundational concepts that underpin data-driven innovation in fields ranging from genomics to social media analytics.

Chapter 3

From the PC to the Datacenter

Definition

This chapter explores the evolution from standalone personal computers to modern datacenters that support high-throughput and large-scale computing.

Key Concepts

- **CPU and GPU**: CPUs are optimized for general-purpose tasks; GPUs excel at parallel processing.
- **RAM**: Volatile memory that stores active processes and data.
- **Storage**: Persistent memory like HDDs and SSDs.
- **Cache Memory**: Fast memory layers (L1, L2, L3) that reduce data access time.
- **Networking**: Includes switches, routers, and protocols that connect systems.

Example

Modern datacenters use Top-of-the-Rack (ToR) switches to efficiently route traffic among thousands of servers.

Review Questions

1. How does cache memory hierarchy (L1, L2, L3) improve performance?
2. Explain the differences between LAN and WAN in a network.

Quiz

- **True or False:** The GPU is mainly used for graphics and cannot be used for scientific computing. (*False*)
- **Multiple Choice:** Which cache level is shared between all cores in a CPU?

  1. L1
  2. L2
  3. L3

  (*Correct answer: L3*)

Links

- Check out GitHub BDP1_2025 for examples of datacenter architectures.
- Refer to Chapter 4 for cloud computing and virtualization insights.

Summary

Datacenters represent the backbone of modern computing, enabling vast amounts of data to be stored, processed, and analyzed. From powerful multi-core CPUs to specialized GPUs, each component plays a crucial role. Networks and file systems link these resources, while cache hierarchies accelerate performance. Understanding how these systems work together is essential for building robust and efficient infrastructures capable of supporting Big Data and high-performance computing workloads.

Chapter 4

Cloud Computing

Definition

Cloud computing delivers computing services (servers, storage, databases, networking, software) over the Internet. It enables on-demand access to resources without direct management by the user.

Key Concepts

- **Self-Service, On-Demand:** Users can provision resources when needed.
- **Network Access:** Resources are available over the internet from any device.
- **Resource Pooling:** Shared resources across users.
- **Elasticity:** Resources can scale automatically as needed.
- **Pay-Per-Use:** Users pay only for what they consume.

Example

Amazon Web Services (AWS) offers Infrastructure as a Service (IaaS), allowing users to rent virtual servers and storage.

Review Questions

1. What are the core characteristics of cloud computing?

2. Explain the difference between IaaS, PaaS, and SaaS.

Quiz

- **True or False:** Cloud computing always requires a virtual machine. (*False*)
- **Multiple Choice:** Which is not a cloud characteristic?
  1. Self-service
  2. Fixed resource allocation
  3. Pay-per-use

  (*Correct answer: Fixed resource allocation*)

Links

- Refer to GitHub BDP1_2025 for examples of deploying apps in the cloud.
- See Chapter 5 for parallel computing infrastructures.

Summary

Cloud computing revolutionizes IT by offering scalable, on-demand resources. Its key features—self-service, network-based access, resource pooling, elasticity, and pay-per-use—enable efficient management and rapid deployment of services. Understanding its service models (IaaS, PaaS, SaaS) is critical to implementing flexible, cost-effective solutions.

Chapter 5

Infrastructures for Parallel Computing

Definition

Parallel computing infrastructures enable the simultaneous processing of multiple computational tasks, accelerating data analysis and complex workflows.

Key Concepts

- **HTC vs. HPC:** HTC maximizes throughput over time; HPC minimizes time per task.
- **GFLOPS:** A measure of computational performance.
- **Amdahl's Law:** Theoretical speedup limit in parallel processing.
- **Distributed Systems:** Multiple interconnected systems sharing workloads.

Example

High-Performance Computing (HPC) clusters run simulations requiring tightly coupled nodes and fast interconnects.

Review Questions

1. Explain the differences between HTC and HPC.

2. How does Amdahl's Law impact parallel efficiency?

Quiz

- **True or False:** HTC jobs always require fast interconnects. (*False*)
- **Multiple Choice:** Which metric measures computational performance?
  1. GBps
  2. GFLOPS
  3. RPM

  (*Correct answer: GFLOPS*)

Links

- See GitHub BDP1_2025 for scheduling strategies.
- Refer to Chapter 6 for containerized computing environments.

Summary

Parallel computing infrastructures leverage multiple systems to solve large, complex problems faster. HTC focuses on throughput, while HPC prioritizes task speed and efficiency. Understanding metrics like GFLOPS and principles like Amdahl's Law is essential for optimizing performance and selecting the right infrastructure.

Chapter 6

Containers and Docker

Definition

Containers are lightweight, portable environments that package applications and their dependencies together, enabling consistent operation across different platforms.

Key Concepts

- **Docker Engine:** Runs and manages containers.
- **Docker Image:** A snapshot containing app code and dependencies.
- **Docker Hub:** A registry to store and share images.
- **Isolation:** Containers share the host OS but isolate application environments.

Example

Using Docker, a user can package a web app into a container and deploy it on any Linux server.

Review Questions

1. What's the difference between a Docker image and a Docker container?
2. How do containers differ from traditional virtual machines?

Quiz

- **True or False:** Containers contain a full guest OS. (*False*)
- **Multiple Choice:** Which command starts a Docker container?
    1. docker push
    2. docker pull
    3. docker run

  (*Correct answer: docker run*)

Links

- Refer to GitHub BDP1_2025 for container exercises.
- See Chapter 7 for HPC and HTC infrastructures.

Summary

Containers simplify software deployment by packaging applications and dependencies into isolated environments. Docker enables rapid, consistent deployment across systems, reducing overhead compared to VMs. Mastery of Docker images, containers, and related commands is key to efficient DevOps and big data processing.

Chapter 7

HPC and HTC

Definition

High-Performance Computing (HPC) focuses on solving computationally intensive problems using powerful resources and fast interconnects, while High-Throughput Computing (HTC) emphasizes maximizing the number of tasks completed over time.

Key Concepts

- **HPC:** Often involves tightly coupled tasks requiring low-latency communication.
- **HTC:** Supports independent tasks (e.g. Monte Carlo simulations) with less stringent interconnect requirements.
- **Cluster Computing:** A collection of interconnected computers working together.
- **Supercomputers:** Extremely fast, large-scale systems for highly demanding tasks.

Example

Weather prediction simulations run on HPC systems due to their tight coupling and need for rapid communication among nodes.

Review Questions

1. What are the primary differences between HPC and HTC?
2. Why do some applications require low-latency interconnects?

Quiz

- **True or False:** HTC always requires a high-speed interconnect. (*False*)
- **Multiple Choice:** Which of the following is best suited for HPC?
    1. Weather simulation
    2. Email processing
    3. Log analysis

  (*Correct answer: Weather simulation*)

Links

- Explore GitHub BDP1_2025 for HPC cluster examples.
- Refer to Chapter 8 for scheduling and workflow management.

Summary

HPC and HTC cater to different computational needs. HPC focuses on speed and tightly coupled tasks with low-latency interconnects, while HTC emphasizes large numbers of independent tasks over time. Understanding these differences is crucial for selecting the right infrastructure for different data processing and analysis challenges.

Chapter 8

Scheduling and Workflows

Definition

Scheduling and workflows manage the execution order of computational tasks across resources, balancing efficiency and priority.

Key Concepts

- **Batch Systems:** Tools like HTCondor manage queues and prioritize jobs.
- **Fairshare Scheduling:** Allocates resources based on user or group priorities.
- **Dependency Management:** Defines task order to ensure correct execution.
- **Workflows:** Encapsulate complex analysis pipelines with task dependencies.

Example

HTCondor can manage thousands of jobs, scheduling them based on available resources and user priorities.

Review Questions

1. What's the difference between a batch system and a workflow manager?

2. Why is dependency management important in workflows?

Quiz

- **True or False:** All workflows are linear without dependencies. (*False*)
- **Multiple Choice:** Which tool is commonly used for batch job scheduling?
  1. Docker
  2. HTCondor
  3. Kubernetes

  (*Correct answer: HTCondor*)

Links

- Refer to GitHub BDP1_2025 for HTCondor configuration examples.
- See Chapter 9 for insights on security and data management.

Summary

Scheduling and workflows coordinate computing tasks across resources, ensuring efficient execution and reproducibility. Batch systems like HTCondor manage queues, while workflows define task order and dependencies. Mastering these systems is key to scaling up data processing pipelines in Big Data and HPC environments.

Chapter 9

Security and Data Management

Definition

Security and data management in big data infrastructures focus on ensuring data integrity, privacy, and controlled access while managing large, distributed datasets.

Key Concepts

- **Authentication:** Verifying user identity.
- **Authorization:** Controlling user access levels.
- **Encryption:** Protecting data in transit and at rest.
- **Data Backup:** Preventing data loss.
- **Data Lifecycle:** Managing data from creation to deletion.

Example

Using SSH keys for secure access and setting file permissions to control read-/write access on shared file systems.

Review Questions

1. What's the difference between authentication and authorization?
2. Why is data encryption important in distributed systems?

Quiz

- **True or False:** Encryption is only necessary during data transfer. (*False*)
- **Multiple Choice:** Which process verifies a user's identity?
    1. Authorization
    2. Authentication
    3. Encryption

  (*Correct answer: Authentication*)

Links

- See GitHub BDP1_2025 for examples of securing cluster access.
- Refer to Chapter 10 for final conclusions and advanced topics.

Summary

Security and data management are critical components of big data infrastructures. Protecting data integrity and ensuring appropriate access are essential for regulatory compliance and reliable analysis. Techniques like encryption, authentication, and data lifecycle management provide the foundation for a secure and trustworthy system.

Chapter 10

Summary and Final Thoughts

Comprehensive Overview

This chapter provides an extensive synthesis of the entire course, integrating all chapters to give you a holistic understanding of Big Data Processing Infrastructures. It covers definitions, key concepts, interconnections, and practical implications, ensuring you're equipped to tackle real-world challenges.

1. Computational Challenge

At the core of NGS analysis is the **String Matching Problem**—finding a short sequence within a larger genome. Tools like **BLASTn** use local alignment, while **BWA** leverages the Burrows-Wheeler Transform (BWT) and FM-index for efficient indexing and searching.

- **Key Points:**
  - Exact vs approximate string matching
  - Real time vs user time in performance evaluation

2. Big Data

Big Data is characterized by the 5 V's: **Volume, Variety, Velocity, Variability,** and **Veracity**. Handling unstructured data and streaming inputs requires specialized architectures and workflows.

3. From the PC to the Datacenter

Transitioning from personal computers to data centers introduces:

- **CPU and GPU roles**
- **Memory hierarchy (L1, L2, L3 caches)**
- **Networking (LAN/WAN)**

Datacenters leverage these technologies to scale up computation.

4. Cloud Computing

Cloud computing enables on-demand, elastic resources delivered via models like:

- **IaaS**: Infrastructure as a Service
- **PaaS**: Platform as a Service
- **SaaS**: Software as a Service

Key features include self-service, resource pooling, and pay-per-use.

5. Infrastructures for Parallel Computing

Distinguishing between **HPC** and **HTC** is vital:

- HPC = tightly coupled, low-latency communication
- HTC = high throughput of independent tasks

Amdahl's Law reminds us that speedup is limited by serial portions.

6. Containers and Docker

Containers (via Docker) simplify deployment by packaging apps and dependencies:

- Share the host OS kernel but remain isolated.
- Enable reproducibility and consistency across platforms.

7. HPC and HTC

HPC uses supercomputers or clusters with fast interconnects (e.g. Infiniband) for tightly coupled tasks. **HTC** focuses on maximizing job throughput, typically on clusters with simpler networking.

8. Scheduling and Workflows

Schedulers like **HTCondor** manage job queues and enforce priorities. Workflows define task dependencies to ensure correct execution order, supporting reproducibility.

9. Security and Data Management

Critical in distributed systems:

- **Authentication** (identity verification)
- **Authorization** (access control)
- **Encryption** (data protection)
- **Data Lifecycle** (management from creation to deletion)

Comparison Tables

HPC vs HTC

Feature	**HPC**	**HTC**
Task Coupling	Tightly coupled	Loosely coupled
Interconnect	High-speed (e.g. Infiniband)	Commodity Ethernet
Use Case	Simulations (e.g. weather)	Independent tasks (e.g. Monte Carlo)
Performance Metric	Speed per task	Throughput (tasks/time)

Containers vs Virtual Machines

Feature	**Containers**	**Virtual Machines**
OS Layer	Share host kernel	Include guest OS
Resource Usage	Lightweight	Heavyweight
Startup Time	Seconds	Minutes
Portability	High	Lower

Cloud Service Models

Model	Description
IaaS	Virtual servers, storage, networking
PaaS	Development platforms, middleware
SaaS	Fully managed applications

Integrated Example

Imagine a genomics lab:

- Uses BWA for alignment (String Matching Problem).
- Runs on an HPC cluster with HTCondor scheduling.
- Deploys Docker containers for reproducibility.
- Leverages cloud resources for burst workloads (IaaS).
- Manages data with secure encryption and strict access controls.

Final Thoughts

Big Data Processing Infrastructures require a blend of skills across computing architectures, software engineering, and security. Understanding:

- The computational challenges (String Matching, FM-index)
- The scale of Big Data and its V's
- Datacenter components (CPU, GPU, RAM, cache)
- Cloud service models
- HPC and HTC trade-offs
- Containers and Docker
- Scheduling and workflows
- Security and data management

— empowers you to design, implement, and manage powerful data pipelines. This integrated knowledge is essential for transforming raw data into meaningful insights in modern science and industry.

Links

- Review all exercises and code at GitHub BDP1_2025.

Appendix A

Command Recap

Introduction

This appendix provides a comprehensive list of practical commands relevant to Big Data infrastructures, NGS analysis, containerization, job scheduling, networking, and more. Each command is accompanied by a brief comment explaining its purpose.

```
# ====================================
# BWA (NGS alignment)
# ====================================
bwa index reference.fasta          # Index the reference
   genome
bwa mem reference.fasta reads.fq > alignment.sam  # Align
    reads

# ====================================
# Docker (Containers)
# ====================================
docker pull hello-world            # Download a test
   container
docker run hello-world             # Run a simple
   container
docker build -t myapp .            # Build an image from
   Dockerfile
docker ps                          # List running
   containers
docker stop container_id           # Stop a container
docker exec -it container_id bash  # Access shell inside
   container
```

```
# ====================================
# HTCondor (Job Scheduling)
# ====================================
condor_submit myjob.sub            # Submit a batch job
condor_q                           # View job queue
condor_q -long                     # Detailed job info
condor_rm job_id                   # Remove a job
condor_status                      # Check cluster status

# ====================================
# SSH (Secure Shell)
# ====================================
ssh user@hostname                  # Connect to remote
   host
scp file.txt user@hostname:/path/  # Copy file to remote
scp user@hostname:/path/file.txt ./  # Copy file from
   remote

# ====================================
# General Linux/Unix
# ====================================
ls -l                              # List files with
   details
cd /path/to/dir                    # Change directory
pwd                                # Print working
   directory
cat file.txt                       # Display file
   contents
less file.txt                      # View file with
   scrolling
df -h                              # Disk usage (human-
   readable)
du -sh folder/                     # Disk usage summary
   for a folder
mkdir new_folder                   # Create a new
   directory
rm file.txt                        # Delete a file
rm -r folder/                      # Delete a directory
   recursively
chmod +x script.sh                 # Make a script
   executable

# ====================================
# NFS (Network File System)
# ======================================
showmount -e servername            # List exported shares
    from NFS server
mount servername:/share /mnt/share # Mount NFS share
umount /mnt/share                  # Unmount NFS share

# ======================================
# Virtual Machines (VirtualBox)
# ======================================
VBoxManage list vms                # List all VMs
VBoxManage startvm "VM␣Name"       # Start a VM
VBoxManage controlvm "VM␣Name" poweroff  # Force-stop a
   VM
VBoxManage snapshot "VM␣Name" take snapshot1  # Take a
   snapshot

# ======================================
# HTCondor Submit File Example
# ======================================
# File: myjob.sub
universe = vanilla
executable = myscript.sh
output = output.txt
error = error.txt
log = log.txt
queue
```

Conclusion

This appendix serves as a quick reference for common commands used throughout the course. Refer back to this list whenever you need to recall syntax or clarify the purpose of a specific command.

All Chapter Pipelines

Appendix A

Pipeline Summary

Introduction

This appendix summarizes the pipelines for all course chapters using simple, text-based arrows (–>). Each pipeline highlights the key steps, tools, and concepts covered in each chapter.

Chapter 1: Computational Challenge

Reads (FASTQ) –> Index Reference (BWA) –> Align Reads –> Output (SAM)

Chapter 2: Big Data

Data Sources –> Ingest Data –> Process Analyze –> Store Results

Chapter 3: From PC to Datacenter

User PC –> LAN/WAN –> Datacenter –> Shared Storage

Chapter 4: Cloud Computing

User Request –> Cloud Provider –> IaaS/PaaS/SaaS

Chapter 5: Parallel Computing

Tasks –> Scheduler –> Cluster Nodes –> Results

Chapter 6: Containers and Docker

Build Image –> Test Image –> Deploy Container

Chapter 7: HPC and HTC

Job Submission –> Cluster –> Compute –> Output

Chapter 8: Scheduling and Workflows

Define Dependencies –> Submit Jobs –> Monitor Jobs –> Collect Results

Chapter 9: Security and Data Management

Authenticate –> Authorize –> Encrypt –> Manage Data Lifecycle

Chapter 10: Summary

All Steps Integrated –> Analysis Pipelines –> Results and Insights

Conclusion

These text-based pipelines provide a quick reference to the key workflows in each chapter, showing how concepts connect and build towards real-world Big Data solutions.