Fundamentals of Edge Computing

Chapter 2

Outline

❑ Distributed Computing

- Middleware, grid, mobile agent
- Hadoop/HDFS, MapReduce
- Spark, Storm

❑ Basic Concepts & Characteristics

- Basic concepts
- 5G/6G
- Storage
- Lightweight libraries
- Edge programming models

❑ Edge vs Cloud

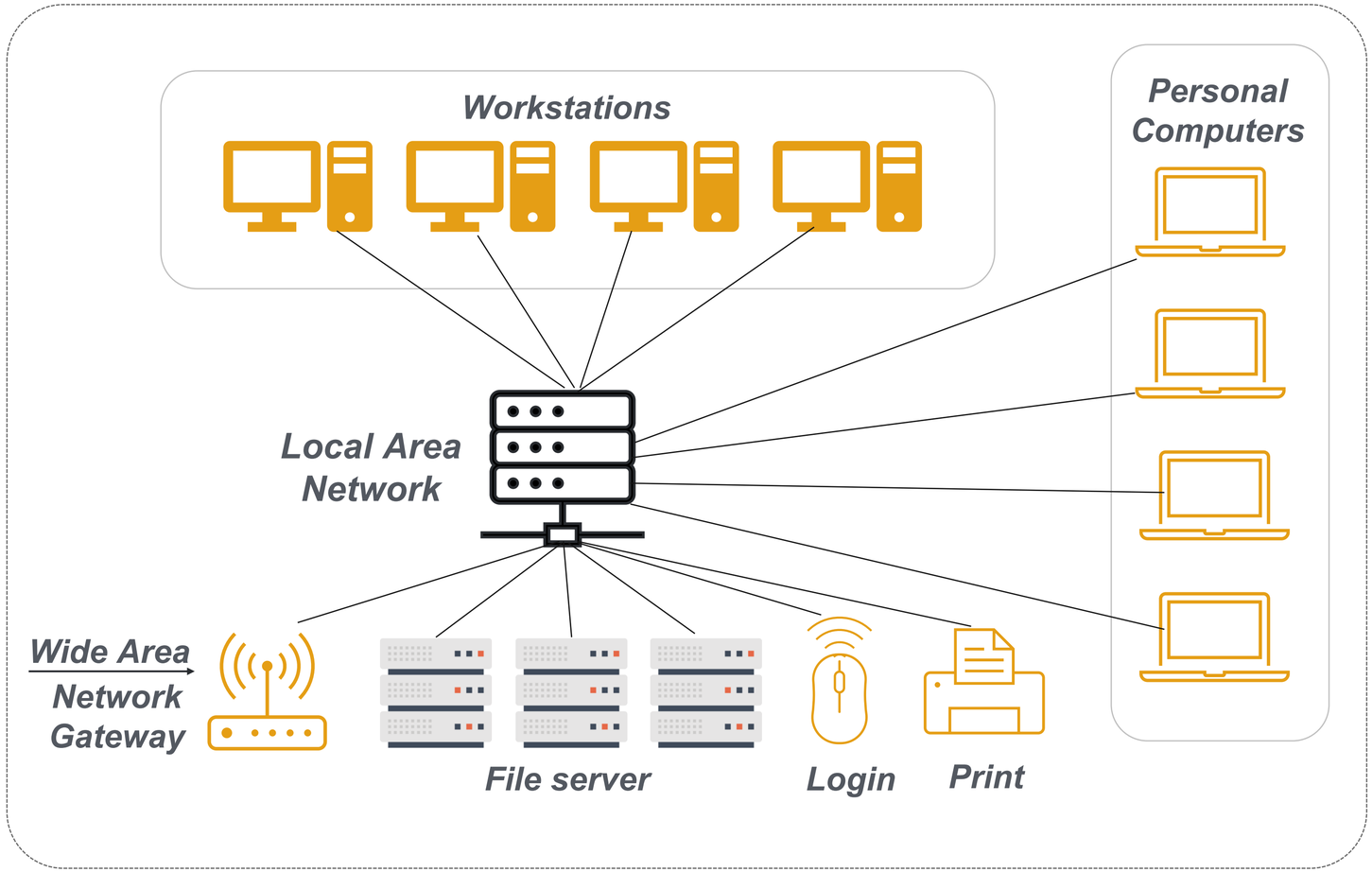
- IaaS / PaaS / SaaS
- Edge computing vs cloud computing
- Four challenge categories

❑ Summary & Practice

- Summary
- Practice

Distributed Computing

Connecting numerous computer nodes via the Internet to break a large task into subtasks distributed across machines. Results are integrated into the final output.



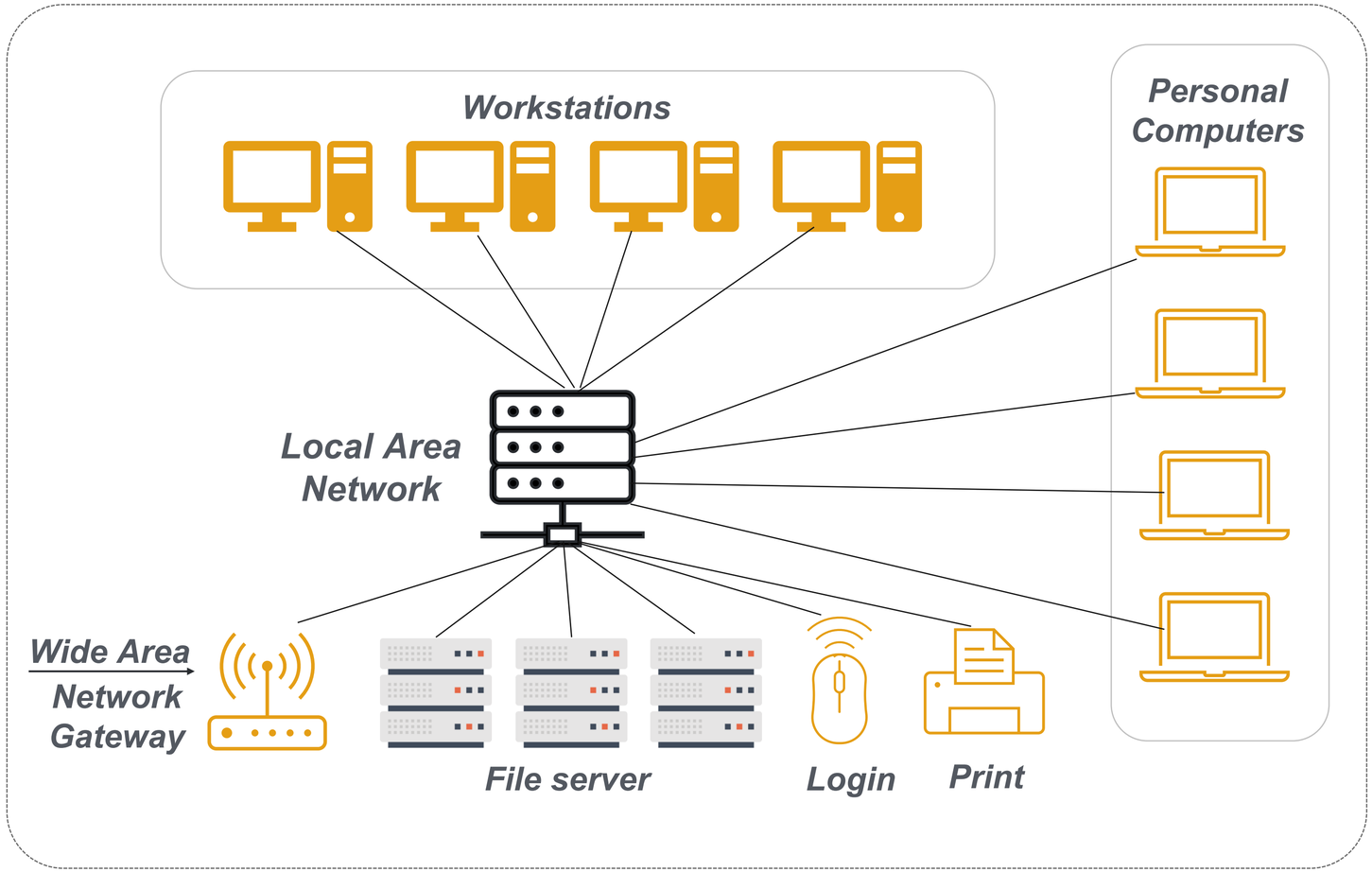
An overview of distributed system

Characteristics

- High-performance computation over high-speed networks
- Meets user demand & resource availability
- Enables resource sharing among nodes

Distributed Computing

Connecting numerous computer nodes via the Internet to break a large task into subtasks distributed across machines. Results are integrated into the final output.



An overview of distributed system

Main challenges

- **Heterogeneity**
- **Scalability**
- **Fault tolerance**
- **Concurrency**

Edge computing builds on this: it applies distributed-computing principles to process data closer to where it is generated.

Distributed Computing Technologies

Five core technologies underpin distributed computing.

Middleware

Sits between the OS and distributed apps to mask heterogeneity of operating systems and network protocols. Now mostly object-oriented middleware.

Grid Computing

Integrates geographically dispersed hardware & software over high-speed networks — e.g. Globus five-layer hourglass, or service-centric OGSA.

Mobile Agent

Autonomously migrates across heterogeneous networks, locating resources and performing tasks on behalf of clients, spawning subagents as needed.

P2P

Links terminal devices and pools idle resources for maximum sharing — open & self-organizing, but faces copyright, management & security issues.

Web Service

Extends object/component tech over the web. A platform- and language-independent layer so apps on different platforms interoperate.

Distributed System Platforms

Big-data environments drove three widely used platforms

Hadoop

Apache framework. Core: HDFS (storage), MapReduce (programming model), HBase (structured table) — the open-source take on Google's GFS, MapReduce & Bigtable.

Writes intermediate results to disk → slow on iterative jobs

Spark

UC Berkeley AMP Lab. In-memory computing via Resilient Distributed Datasets (RDDs) — avoids disk I/O between MapReduce steps.

Up to 100× faster than Hadoop, but very memory-intensive

Storm

Twitter/X open-source real-time stream processor. Topologies of spouts (read) and bolts (process) arranged in a DAG; multi-language.

Online ML, real-time analytics, distributed RPC, ETL

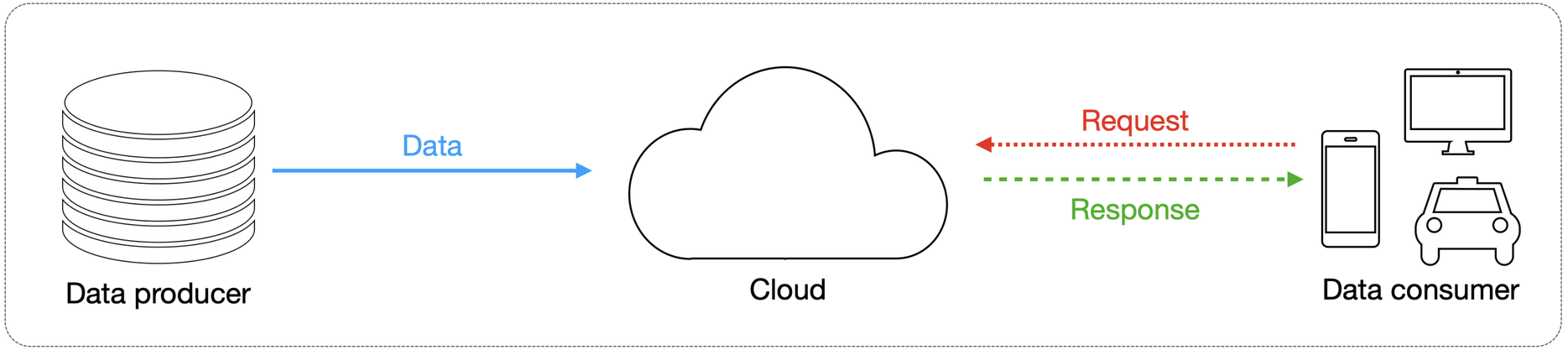
Tom White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.

Matei Zaharia et al. "Spark: Cluster computing with working sets". In: 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10). 2010.

Apache Software Foundation. Apache Storm. https://storm.apache.org/. Accessed: 2024-07-31.

Traditional Cloud Computing Model

In the IoE, source data flows to the cloud and consumers send requests back but this model strains under edge-scale data.



Traditional cloud computing model

➢ Why it falls short in the IoE

Bandwidth overload

Sending all edge data to the cloud wastes bandwidth & compute

Privacy risk

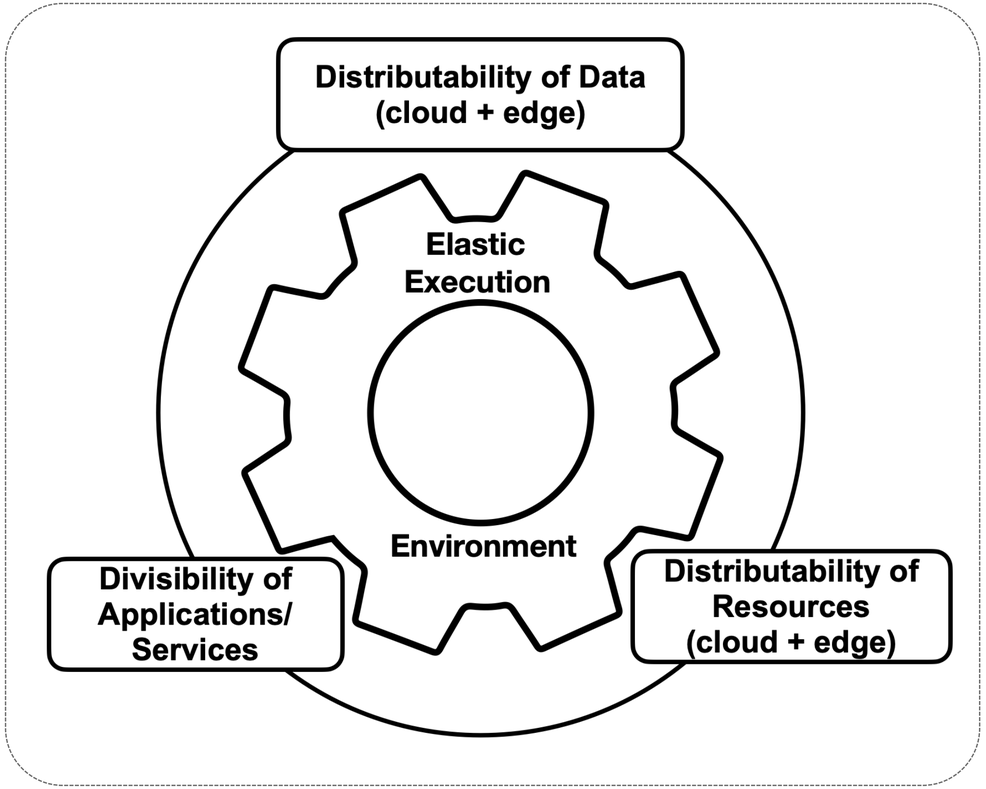
Centralizing data raises protection challenges in IoE

Energy cost

Edge nodes have limited power; GSM/Wi-Fi radios drain a lot

Edge Computing Model

Edge computing runs computation on the network path, on a bidirectional flow: devices are both data producers and consumers.



Edge computing model characteristics

Three defining requirements

Divisibility of applications/services

Tasks must be decomposable into migratable subtasks that can run at the edge.

Distributability of data

Data must be splittable across sources; without it, the model collapses back to centralized cloud.

Distributability of resources

Compute, storage & communication resources must themselves be distributed to process data at the edge.

Key Characteristics

Five technologies make the edge computing model work.

Compute Migration

Minimize data transmitted across the network — preprocess at the edge, allocate tasks by current load, balance energy vs latency vs data volume.

5G & 6G Communication

Higher speed, ultra-low latency, massive capacity. Cuts transmission delay when edge devices still must upload to the cloud.

Advanced Storage

NVM (NAND Flash, PCRAM, RRAM) gives low-latency, high-density storage but software stacks & reliability lag the hardware.

Lightweight Libraries & Kernels

Edge devices can’t run heavy software; containerization (Docker) shares the host kernel for efficient, isolated deployment.

Edge Programming Model

New models needed for heterogeneous nodes — the “computation stream” and Firework model (manager + nodes, shared data views).

Key Characteristics: Compute Migration

The goal is to minimize the data transmitted across the network

The strategy

Preprocess at the edge

Partial or complete preprocessing where data is collected — filter out unnecessary data to cut bandwidth demand.

Allocate by current load

Dynamically distribute tasks by each device's computational load, so no single device is overloaded.

Balance three factors

Find the optimal balance of energy consumption, edge computational latency, and volume of data transmitted.

Key migration decisions

1. Which tasks are suitable for migration?
2. What migration strategy to use?
3. Which specific tasks to migrate?
4. Partial or complete migration?

Also decide: can the app be migrated · is the required data volume known · can tasks be synchronized post-migration.

Key Characteristics: 5G & 6G Communication

Both aim to deliver higher speeds, ultra-low latency, more reliability, massive capacity, and a more uniform user experience.

5G

- Standard development began worldwide in 2019 — the successor to 4G
- Predicted 1.7B+ subscribers worldwide by 2025 (GSMA)
- vs 4G: more speed & responsiveness, far more devices at high data rates, lower latency
- Benefits autonomous driving, virtual reality and the IoT

6G

- Still early — expected to succeed 5G
- Even higher speeds & lower latency
- Integrates AI and advanced satellite networks
- Enables AR, high-fidelity mobile holograms, blended physical + digital realities

Why it matters for edge: edge devices still upload intermediate/final data to the cloud. However, 5G/6G cut that transmission delay. Both need scalable architecture with SDN & NFV.

Key Characteristics: Advanced Storage Systems

The speed gap between storage and processors is a bottleneck. NVM (NAND Flash, PCRAM, RRAM) outperforms mechanical drives for edge data.

NVM advantages: high density · low energy consumption · low latency · high read/write speeds — mitigating the I/O limits of existing storage systems.

➢ Three integration challenges at the edge

Tech dev vs software support

NVM hardware advances faster than its software stacks — a “software bottleneck.” Stacks designed for mechanical drives don’t exploit NVM’s full speed.

Diverse application needs

Edge demands many storage architectures. Research must maximize performance, energy efficiency & capacity, and simplify management in complex edge settings.

Reliability in harsh settings

Robust read/write and high data reliability under resource constraints — facing NVM consistency issues, malicious wear attacks, and media lifespan/failure.

Key Characteristics: Lightweight Libraries & Kernels

Edge devices are constrained by hardware capacity and often cannot run heavy software applications.

The problem

Heavy frameworks don’t fit

Running Apache Spark optimally needs ≥ 8-core CPU + 8 GB memory; the lightweight Apache Quarks does only basic tasks, unsuitable for advanced analytics.

Device heterogeneity

The edge is full of diverse devices from many makers with varying performance, making application deployment complex.

VMs are too heavy

Traditional VM-based virtualization is too resource-intensive and slow for edge environments, where swift responses are crucial.

Solution: containerization

- Docker virtualizes at the OS level and shares the host system’s kernel. No full OS per instance.
- Far more resource-efficient than VMs; isolated environments. with minimal resources
- Boosts scalability & deployment speed; consistent environments across dev, test & production.
- Ideal for resource-constrained edge devices.

Key Characteristics: Edge Programming Model

Cloud apps are written once and run on provider servers. Edge nodes are heterogeneous, so traditional programming models are inadequate.

“Computation stream”

The data transmission path plus the sequence of computations applied along it at source device, edge nodes and cloud.

New runtime libraries

Implement language functions & provide runtime support, offering simple application interfaces for the diverse, dynamic edge.

The Firework Model

A programming model designed specifically for edge computing, with two components:

Manager — decomposes service requests into subtasks and distributes them; each runs locally on a participant’s device.

Nodes — offer predefined functional interfaces; register datasets & functions as data views shared across all participants.

Cloud Computing: IaaS / PaaS / SaaS

Cloud computing delivers scalable compute, network & storage from data centers over a network via three service models.

IaaS

Infrastructure as a Service

Raw compute, storage & networking: manage the OS, runtime and apps.

PaaS

Platform as a Service

A managed platform to build & deploy on the provider runs the infrastructure.

SaaS

Software as a Service

Ready-to-use applications delivered over the Internet — nothing to manage.

Cloud characteristics

- **Large-scale servers**
- **High reliability**
- **Scalability**
- **Virtualization**

Edge Computing vs Cloud Computing

	Edge Computing	Cloud Computing
Target Application	Mobile applications and IoT	General Internet applications
Server Node Location	Edge of the networks (e.g., gateways, WiFi access points, cellular base stations)	Data centers
Server & Client Communication	Wireless local area network, 4G/5G/6G, etc.	Wide area network
Number of served devices	Billions	Millions
Service Type	Services based on local information	Services based on global information

Benefits

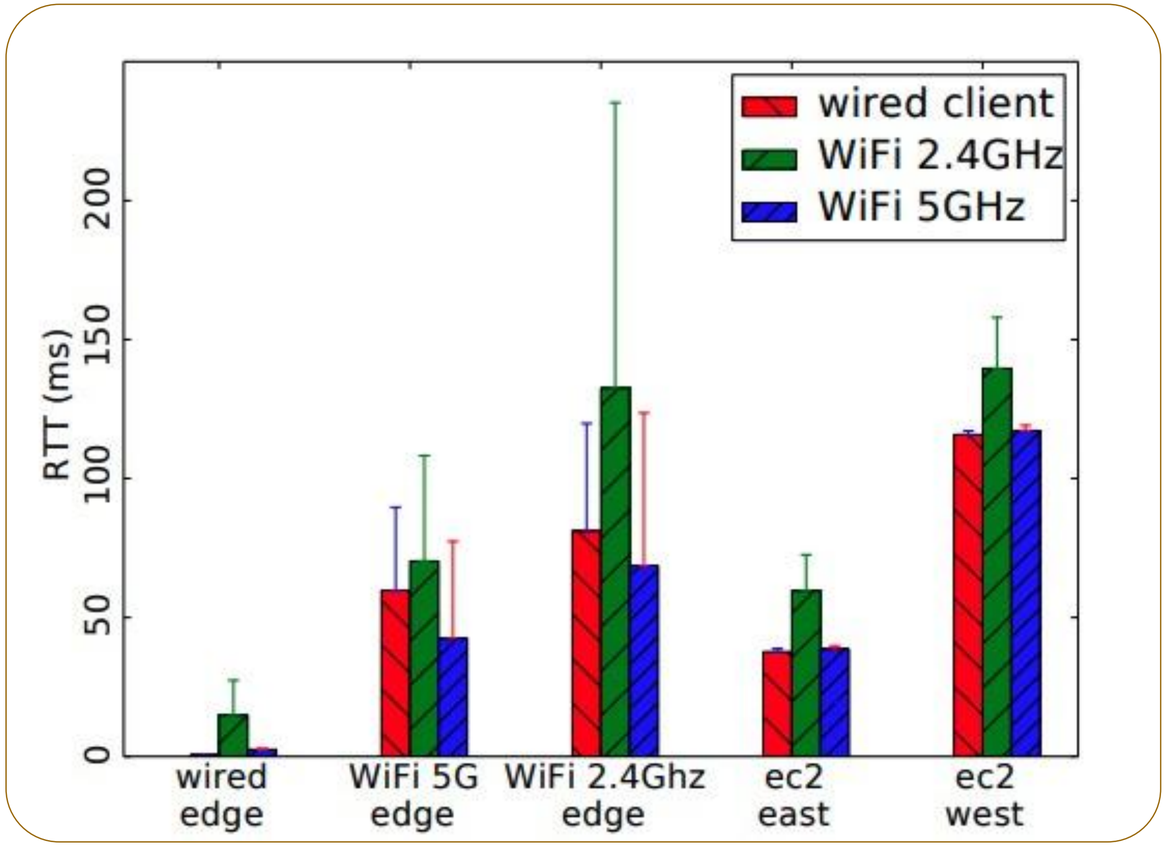
➢ Cloud Computing

- **Lower upfront cost**
  - Get applications to market quickly
- **Flexible pricing**
  - Pay-as-you-go
- **Limitless compute on demand**
  - React and adapt to changing demands instantly
- **Simplified IT management**
  - Users can access IT support and focus on the business's core needs
- **Easy updates**
  - Get the latest hardware, software, and services
- **Reliability**
  - Data backup, disaster recovery, business continuity
- **Save time**
  - No more configuring private servers and networks

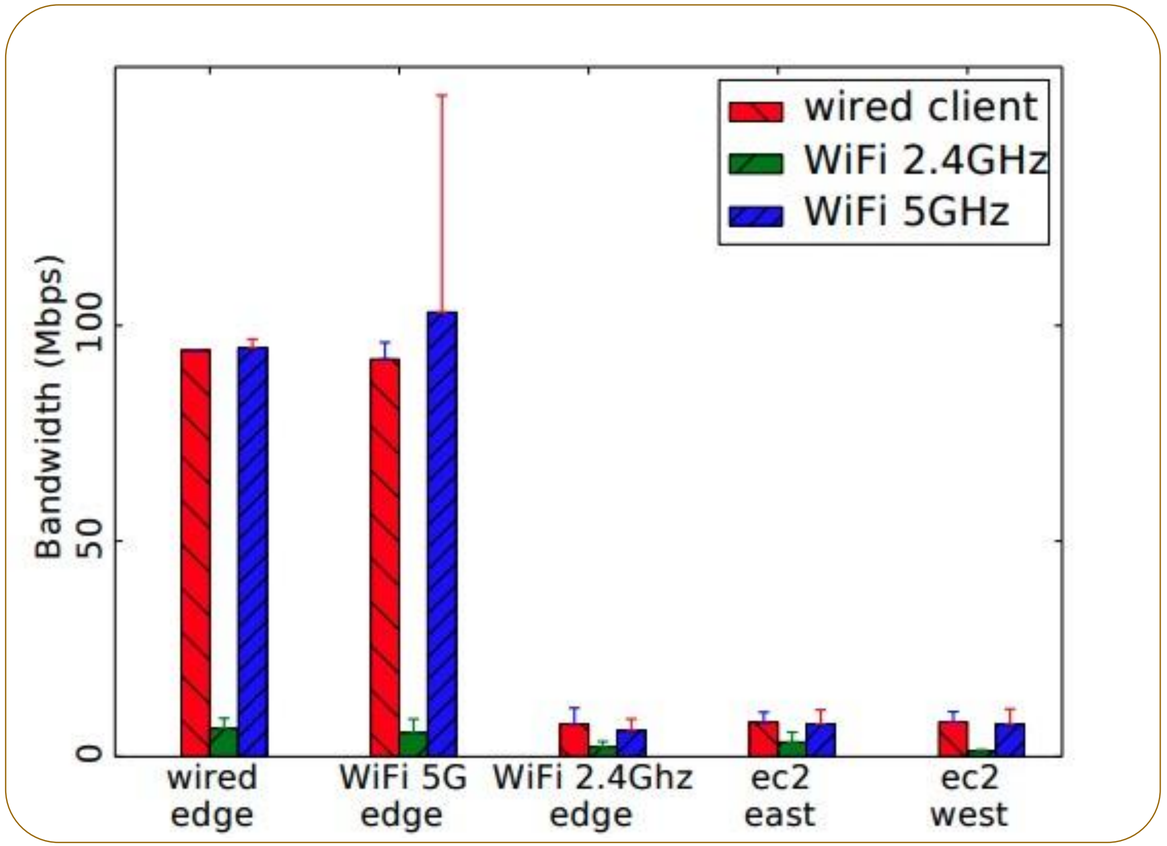
➢ Edge Computing

- **Lower latency**
  - Reduced data travel.
  - Benefit fully autonomous vehicles, augmented reality, etc.
- **Reduced cost**
  - LAN: higher bandwidth and storage, lower cost
  - Less data needs to travel to the cloud
- **Model accuracy**
  - Lower the data size to fed into a cloud model when bandwidth is low
  - Data feedback loops can improve edge AI model accuracy
- **Wider reach**
  - No need for internet access. Extends the range of computing to previously inaccessible or remote locations
- **Data sovereignty**
  - Reduced exposure to cybersecurity attacks, more compliance with data laws

Edge Computing vs Cloud Computing: Latency & Bandwidth



Round-trip time between client and edge/cloud



Bandwidth between client and edge/cloud

Insights: with good connections, edge nodes beat cloud on latency & bandwidth; the cloud remains a backup for saturation & long jobs.

Shanhe Yi et al. “LAVEA: Latency-aware video analytics on edge computing platform”. In: Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing. 2017, pp. 1–13.

Edge Challenges

The 2017 CCC "Grand Challenges in Edge Computing" report groups the open problems into four areas.

Application

Real-time processing & communication, security & privacy, incentives & profitability, adaptive app development, and testing tools.

Architecture

Cage-level security, embracing approximation, trade-off theory, data provenance, QoS at the edge, and testbeds.

Capabilities & Services

Resource naming & discovery, standardized APIs, intelligent edge services, security & trust, and the edge service ecosystem.

Edge Computing Theory

Refining the theoretical foundation & framework so edge computing can better support IoE data processing.

Summary

- Fundamental **concepts and models** of edge computing.
- **Bidirectional computation flow** between edge and cloud.
- **Computation offloading** for efficient task execution.
- Key technologies: **offloading, 5G/6G, storage, lightweight software, edge programming models.**
- Edge computing **complements and extends cloud computing.**
- Key **advantages and challenges of edge computing.**

Practice

1. What are the key differences between distributed and centralized computing systems?
2. Describe the main functionalities of Hadoop’s HDFS and how it supports distributed data storage.
3. Explain the basic concept of edge computing and its key characteristics.
4. Discuss the complementary relationship between edge computing and cloud computing.
5. Identify and explain two main challenges of implementing edge computing systems and propose potential solutions.

Project

Explore the impact of latency on application performance. Set up a cloud-based application and measure its performance, then replicate the application in an edge computing environment and compare the results. Utilize tools like https://prometheus.io/ for monitoring and https://grafana.com/ for visualization to measure and compare latency.

Cloud vs **Edge**

Same application, two locations. Measure how long each takes to respond — and how steady it is.

Experiment Setup

Build one small app

A tiny service — e.g. send an image, get a label back. Package it in a container so it runs the same everywhere.

Deploy it twice

Once to a cloud region far away (AWS / Azure / GCP), once on a nearby edge device (a laptop or Raspberry Pi).

Measure & compare

Send the same requests to both. Use Prometheus to record response times and Grafana to chart them side by side.

Project

Explore the impact of latency on application performance. Set up a cloud-based application and measure its performance, then replicate the application in an edge computing environment and compare the results. Utilize tools like https://prometheus.io/ for monitoring and https://grafana.com/ for visualization to measure and compare latency.

Deliver a short report answering one question: how much does proximity matter for your app?

➢ Task for completing the project

- **Build & containerize a small app**
- **Deploy to cloud + edge**
- **Measure latency with Prometheus**
- **Chart it in Grafana**
- **Report median, p95/p99 & jitter**
- **Explain the result & the trade-off**

References

- Tom White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.
- Matei Zaharia et al. “Spark: Cluster computing with working sets”. In: 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10). 2010.
- Apache Software Foundation. Apache Storm. https://storm.apache.org/. Accessed: 2024-07-31.
- Shanhe Yi et al. “LAVEA: Latency-aware video analytics on edge computing platform”. In: Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing. 2017, pp. 1–13.