# Practical Course: Cloud-Based Data Processing

Michalis Georgoulakis

Chair for Database Systems
School of Computation, Information and Technology
Technical University of Munich

14.04.2026

# Course Organization

# Who am I?

**Michalis Georgoulakis**
Email: geom@in.tum.de
Office Number: 02.11.060
Webpage: https://db.in.tum.de/people/sites/georgoulakis/

PhD candidate at the Chair for Database Systems

MEng National Technical University of Athens

Teaching assistant

Research interests: Cloud-based analytics, Query optimization, Scheduling

**Office Hours**: Thursday, 14:00-15:30

# What this course is about

**Build a distributed data processing system from scratch**

- Split work across machines, move data efficiently, coordinate results

**Deploy and operate it in the cloud**

- Experience what changes when you move from your local machine to cloud infrastructure
- Work across the cloud stack: storage, networking, compute
- Use modern cloud constructs: containers, object storage, serverless functions

**Reason about performance and cost**

- Measure everything: latency, throughput, scalability, monetary cost
- Understand the tradeoffs behind real system design decisions

**Bridge theory and practice**

- Implement ideas from research works on cloud infrastructure and validate them with your own measurements

# Expectations

- Interest in *databases* and *distributed systems*.
- Basic understanding of *modern computers*
- Programming fluency in one *systems language* (**C/C++**).
- *Linux* programming and access

# Prior Experience

Before we dive in, let's see where everyone stands.

https://tweedback.de/2sbh/quiz

# Motivation

# Why distributed systems?

How long would it take one machine to process 1 TB of log data?

- Assume data lies in an SSD with read throughput at ~500 MB/s

# Why distributed systems?

How long would it take one machine to process 1 TB of log data?

- Assume data lies in an SSD with read throughput at ~500 MB/s
  → ~30 minutes
- If we have 10 machines with the data?

# Why distributed systems?

How long would it take one machine to process 1 TB of log data?

- Assume data lies in an SSD with read throughput at ~500 MB/s
  → ~30 minutes
- If we have 10 machines with the data?
  →~3 minutes
- 100?

# Why distributed systems?

How long would it take one machine to process 1 TB of log data?

- Assume data lies in an SSD with read throughput at ~500 MB/s
  → ~30 minutes
- If we have 10 machines with the data?
  →~3 minutes
- 100?
  →~18 seconds.

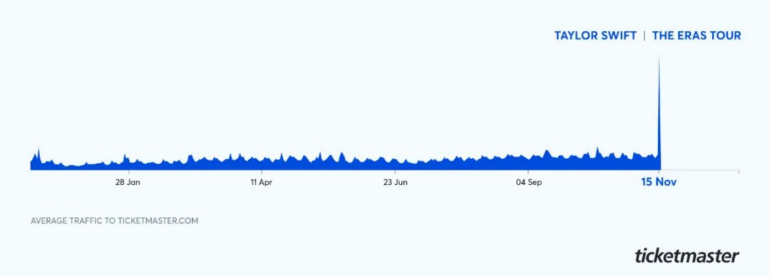
This sounds great, but doesn't come for free.

- Would require access to 100 machines.
- Would require 1 TB of storage distributed across all machines
  - ...but then how does each machine know which part to read?
- Someone would have to pay.
- The chance of a machine failing increases.

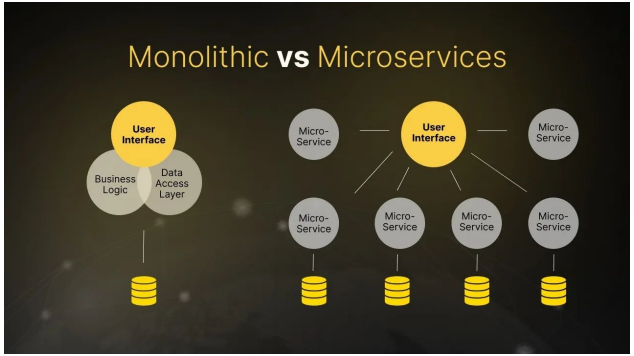
# It's not just scale

Three dimensions that change when you distribute:

- **Performance:** we want to parallelize work across machines
- **Load:** we want to have enough resources to support our workload
- **Fault tolerance:** machines fail - our system shouldn't
- **Cost:** we want to pay for what you use, not what you own




Src: Taylor Swift | The Eras Tour Onsale Explained




Src: Monolith vs Microservice Scalability

The cloud makes all the above possible but introduces new challenges.

# Data Engineer Job Outlook 2025

An article sampled 1,000 Data Engineer job postings.

Here's what employers want today:

- **Common Frameworks:** Apache Spark (38.7%), Snowflake (29.2%), Amazon Redshift (21.8%), Databricks (16.8%).
- Other commonly referred systems: Cassandra, MongoDB, DynamoDB

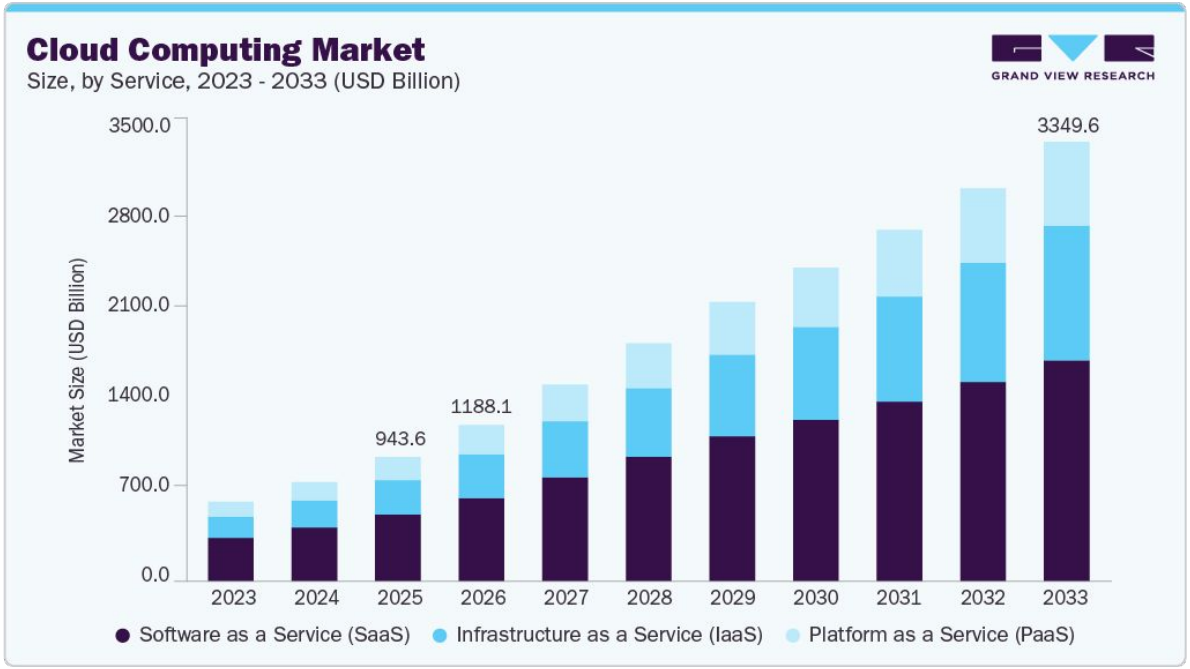
What do these commonly referred frameworks and systems have in common?

➢ *They are all distributed data processing systems.*

If one wants to work with data at scale, it is essential to understand distributed computing.

Source: https://365datascience.com/career-advice/data-engineer-job-outlook-2025/

# Why the cloud?




Src: Cloud Computing Industry

- Internet has over 5 billion users today, and the number is still growing
- Digitalization of society and the Cloud transform whole industries
- US Cloud Computing market (USD billion), expected to triple in 7 years.

# How the Cloud impacts technology development

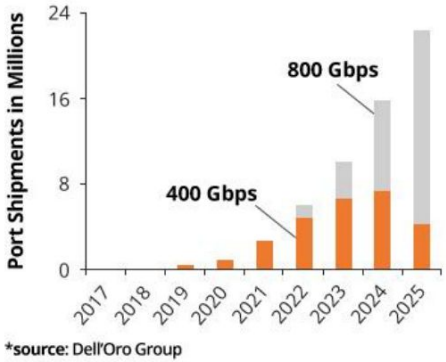
- Cloud makes cutting-edge hardware and infrastructure accessible to everyone.

**AI / LLM Infrastructure**

- GPU clusters on demand: H100, MI300X.
- Training GPT-4 required ~25,000 GPUs, only possible because cloud providers pooled them.
- Custom AI chips: AWS Trainium, Google TPU v5, Azure Maia.

**Fast Network Interconnects**

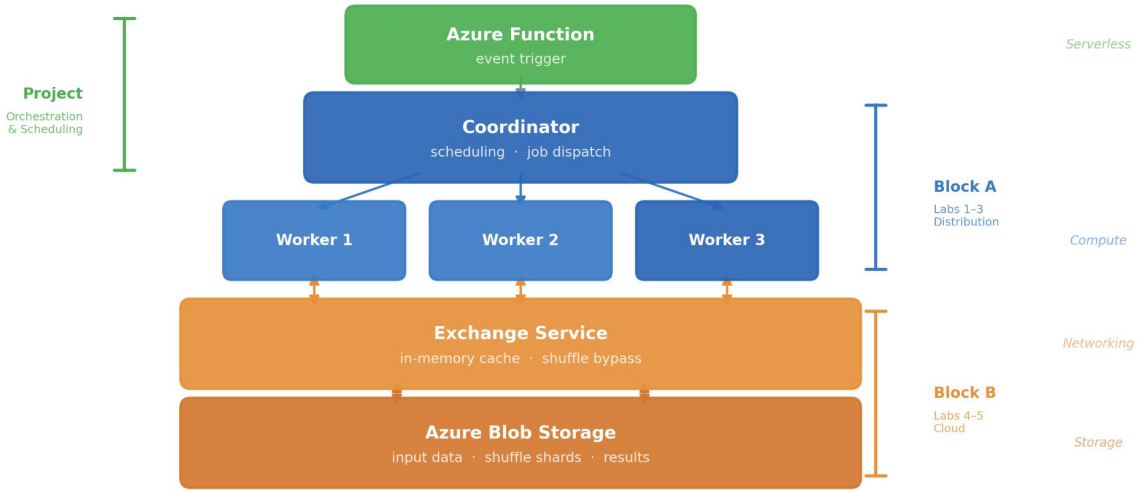
- 400 Gbps already standard. AWS EFA, Azure InfiniBand for GPU-to-GPU.




**Latest Storage Technologies**

- NVMe SSDs with millions of IOPS. Azure Elastic SAN, AWS io2 Block Express.
- Object storage has reached exabyte scale (S3 stores >400 trillion objects).

# What we'll build this semester




By July, you will be able to upload a file and the system processes it automatically, and reason about your decisions in every step.

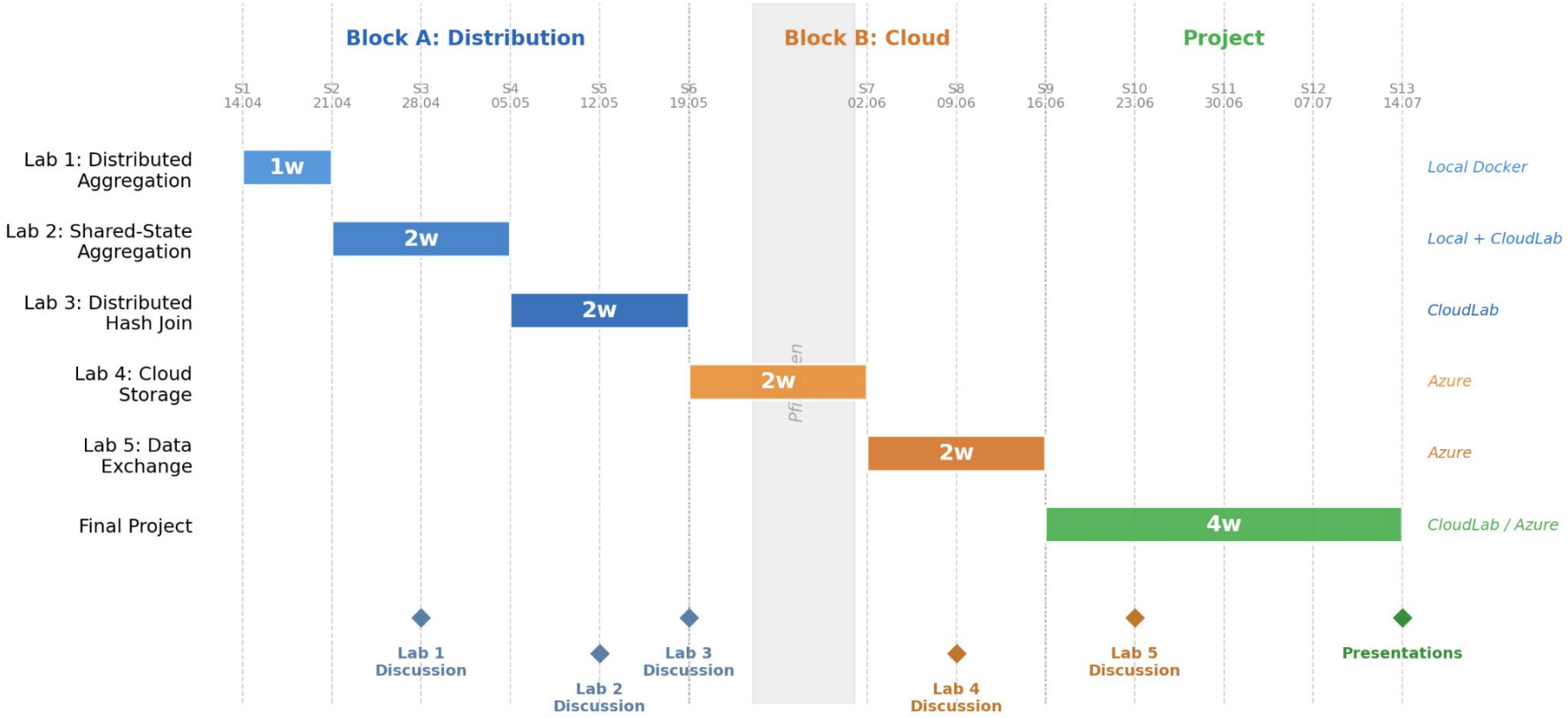
# Course Organization

# Course Resources

- Weekly meetings, Tuesdays at 14:00 (*02.11.018 Seminarraum*).

- Mattermost Channel for offline communication
- GitLab Server for uploading/submitting assignments.


# Schedule




5 progressive assignments + a final project

- Block A: distributed query processing (local + CloudLab)
- Block B: cloud-native architecture (Azure)
- Final project: open-ended

# Grading

| Component | Weight |
| --- | --- |
| Lab assignments | 50% |
| Q&A on Assignments | 20% |
| Final project + presentation | 30% |

# Lab Assignments

- There are 5 lab assignments across the semester.

- Deliverables
  - Code implementing the task
  - Short text file answering a set of design and analysis questions
    - *Optional*: 1-2 plots of your measurements, discuss anomalies and surprises.

- Each lab is graded on:
  - Correctness: Does the system work? Do the tests pass?
  - Performance: Are your results reasonable? Did you address obvious bottlenecks?
  - Report file: Were the questions answered?

# Weekly Discussions

- For each assignment, a few students will briefly discuss their implementation (2–3 minutes) followed by a few questions.
- Every student will be asked to present at least **twice** during the semester.
- No slides or preparation material required.

# Final Project + Presentation

**Individual Project:**

- Try to reproduce a paper, or perform a case study on one aspect (e.g., scheduling algorithms).
- Implement some fancy feature in a distributed system, or
- Work on a topic provided by a **PhD student** of the chair
- More freedom on programming language, might be able to use Chair servers or cloud resources depending on topic and advisor.
- Produce interesting insights and pave the way for future collaborations with us.

# Requirements to pass

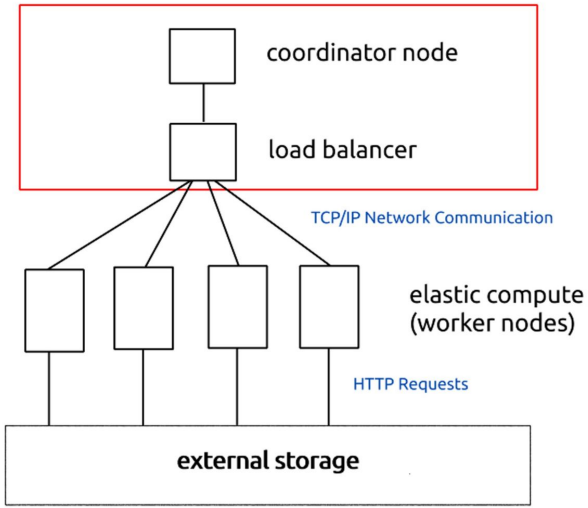
- Score at least 50% in each of the three components individually
  - Submit at least 3 of 5 lab assignments
  - Present at least once in the oral discussions
  - Submit the final project + presentation

# Assignment 1

# Assignment 1: Distributed Aggregation

Task: count domain occurrences across a partitioned web-request log

- Architecture: coordinator + N workers over TCP
- We provide: networking building blocks, protocol, serialization, Docker setup
- You implement: the coordinator's work queue logic and the worker's processing loop



# Execution Environment - Network

Workers are ephemeral system processes.

They communicate with the coordinator through TCP/IP sockets, in a client-server model.

System calls for establishing connection: socket(), bind(), listen(), accept(), connect()
System calls for sending/receiving data: recv(), send(), poll()

We provide you with the communication building blocks so that you can focus more on the coordinator and worker logic.

# Dynamic Work Assignment

**Fault Tolerance**

A worker can fail in numerous ways:

- complete disconnection
- node unresponsive for some time

The system should tolerate node failures.

*Goal:* If a node fails, its work should be handed to other worker/workers, and the result should still be correct.

**Fair Distribution - Load Balancing**

We want to avoid the presence of having underutilized or straggling workers.
*Test*: For the case of 4 workers, each one of them should serve >10% of the workload.

# Deliverables

- Fill the TODOs across coordinator.cpp and worker.cpp
- Run with 1, 2, 4 workers and measure speedup
- Kill a worker mid-run and show that the system recovers

Hints:

- Make sure to read the comments, there are tips that may help you.
- Start the assignment soon, ask your questions on Mattermost or office hours.

Report: speedup plot, load distribution, failure recovery observations
**Due**: next session (21.04 at 14:00)

# Questions

Thank you for your attention!