Advanced Cloud Computing

2ND LAB SESSION

MapReduce

Real-world data comes in many, many forms.

- Streaming, large pieces of information, small database records, etc.

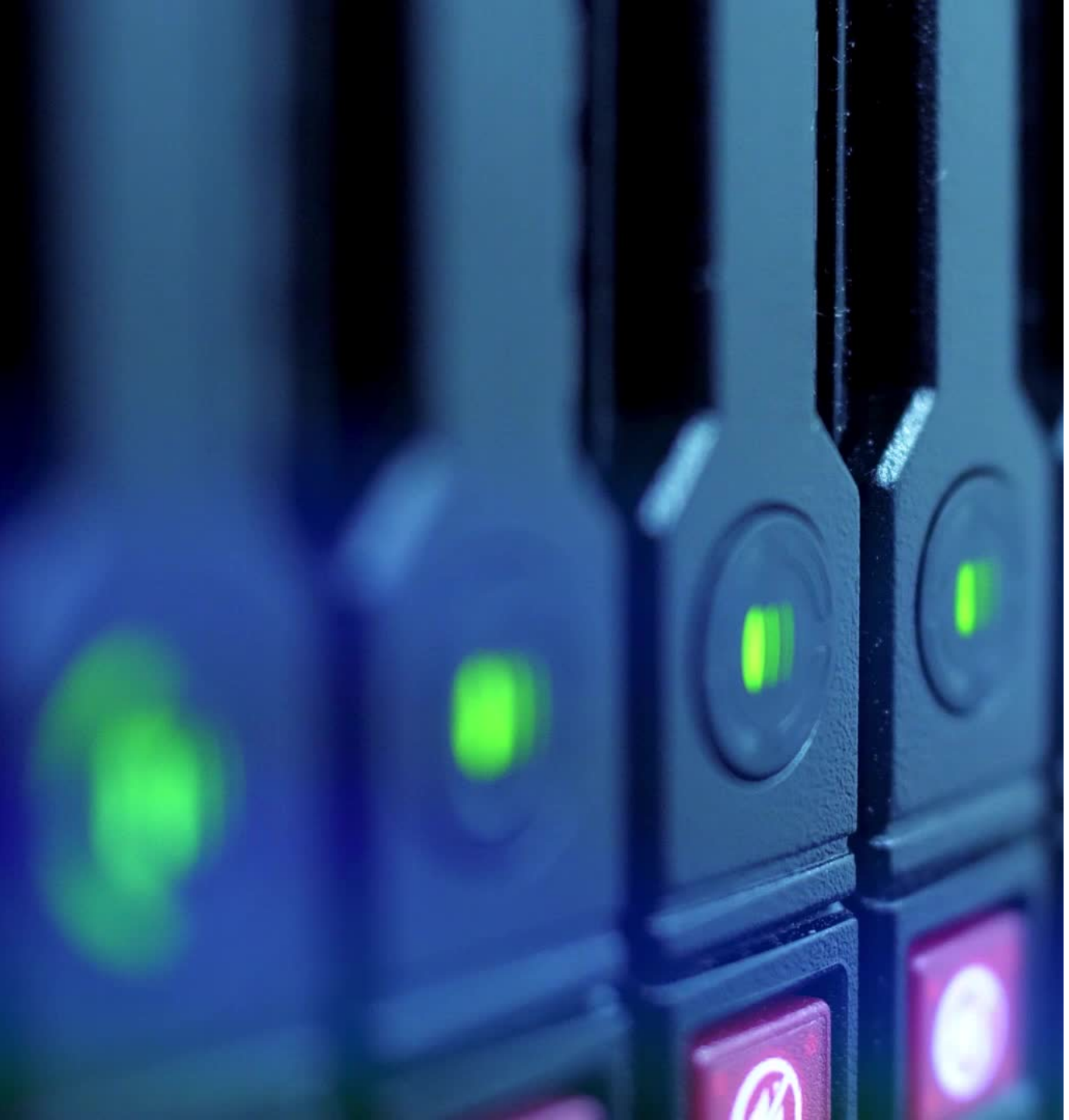
Processing such information at real-time or near real-time is challenging.

- How can we process 100s of GBs of incoming data to make decisions?

Streaming is another problem

- Millions are using the service at the same time and data needs to be transferred correctly

This is where MapReduce comes in



It's about being lazy

- MapReduce advocates for moving processing to the data rather than the inverse.
- This way you can have a cluster of cheap processing and storage in many places instead of having nexuses of super-computers.
- This way:
  - Failures can be handled easily.
  - You don't have to worry about your network performance.
  - You don't need to care about synchronization and hardware idle time.

Divide and Conquer

- MapReduce is comprised of 3 stages
  - Map: Breakdown a large problem into smaller, manageable problems.
  - Reduce: Solve each small problem.
  - Aggregate: Aggregate the results until a solution is found.
- It's more of guideline on how to solve large problems rather than a solution.
- It advocates for using many small jobs and using swarms of cheap, commodity hardware instead of one large job on a handful of powerful ones.

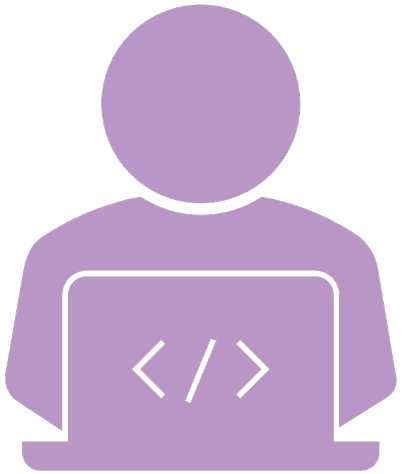
It's all around you

- Google Maps uses MapReduce to give you the best routes.
- Google Search uses MapReduce to index it's search engine.
- Netflix uses MapReduce to make sure you always get the best picture quality as possible.
  - Content Delivery Network (CDN) optimization by calculating the best servers and paths for each cluster of users.
- When you call someone, MapReduce is used to determine which cell towers should be used for connection.
- Facebook and Twitter use it to recommend you friends.
- AWS uses it to make sure your service is always up.

It translates to other domains (roughly)

- How are **large** ML models able to serve millions of users simultaneously?
  - Inference (just running inputs on an ML model) involves many, many operations.
  - Scale that to many users, querying the model at the same time.
  - This is how ChatGPT, Bard, Claude, and etc. work.
- How to allow for millions of users to stream content specifically tailored for them?
  - We have a large database of users.
    - We also have a large DB of their preferences.
  - We have an extremely large database of content.
  - This is essentially how Instagram, TikTok, YouTube, and etc. operate.

Then What are Hadoop and Spark?

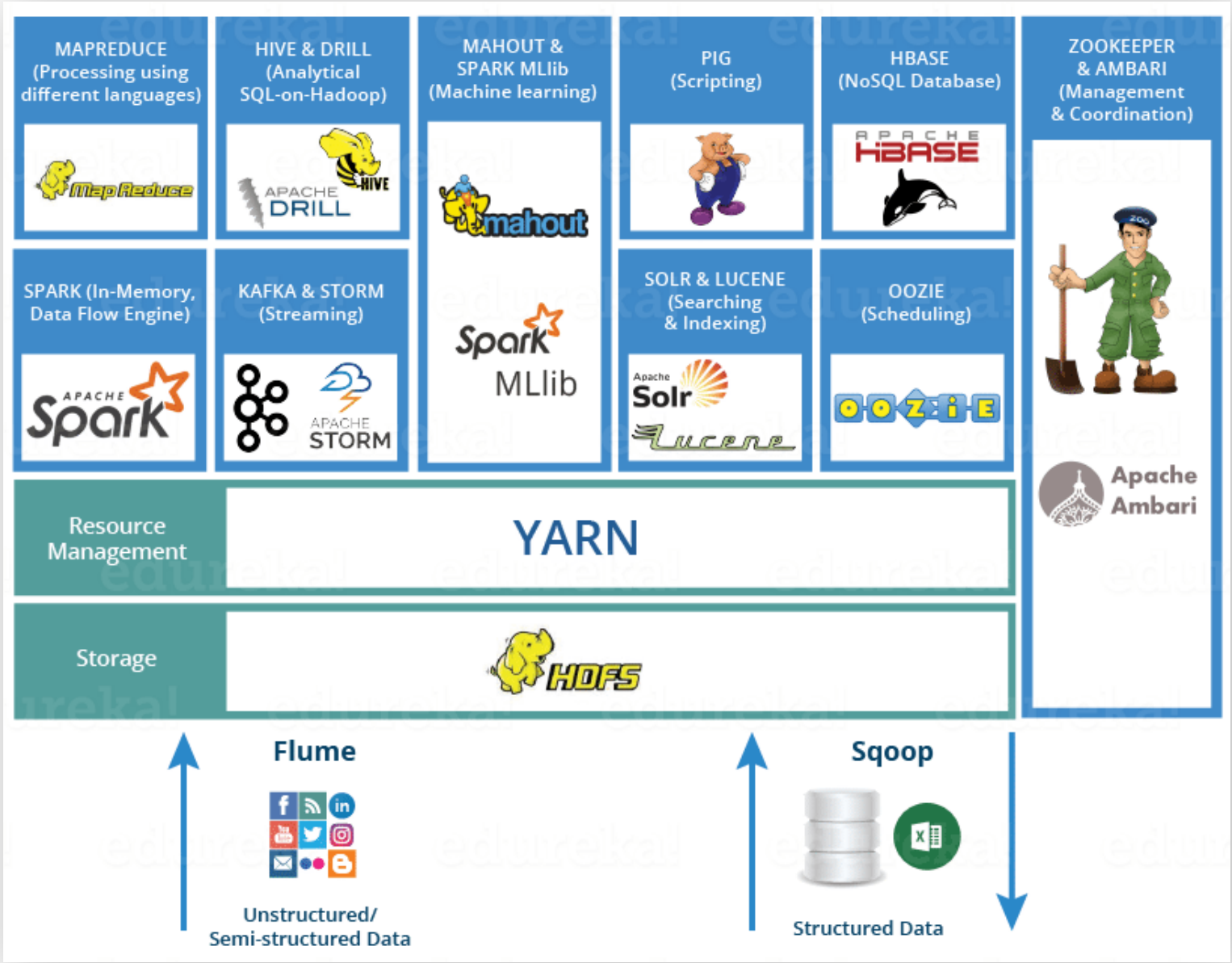


- MapReduce is a set of guidelines.
- Hadoop is a framework for big data management.
  - It is an ecosystem for end-to-end big data processing, analysis, management, and storage.
- Spark is a data processing engine.
  - It allows for processing data in-memory rather reading it over and over from a file system.
  - Spark can be used to do data processing in a standalone fashion.
  - Or it can be a part of the Hadoop ecosystem where it can handle real-time data processing.

Why use Hadoop and Spark?

- Even though MapReduce is simple in concept, it gets very complicated in practice.
- You can define how the task should be broken down and processed.
  - How can you make sure the tasks are broken down correctly?
  - How can you handle failures?
  - How can you address synchronization between your tasks?
  - How can you scale up or down based on your needs?
  - How will you handle idle hardware?

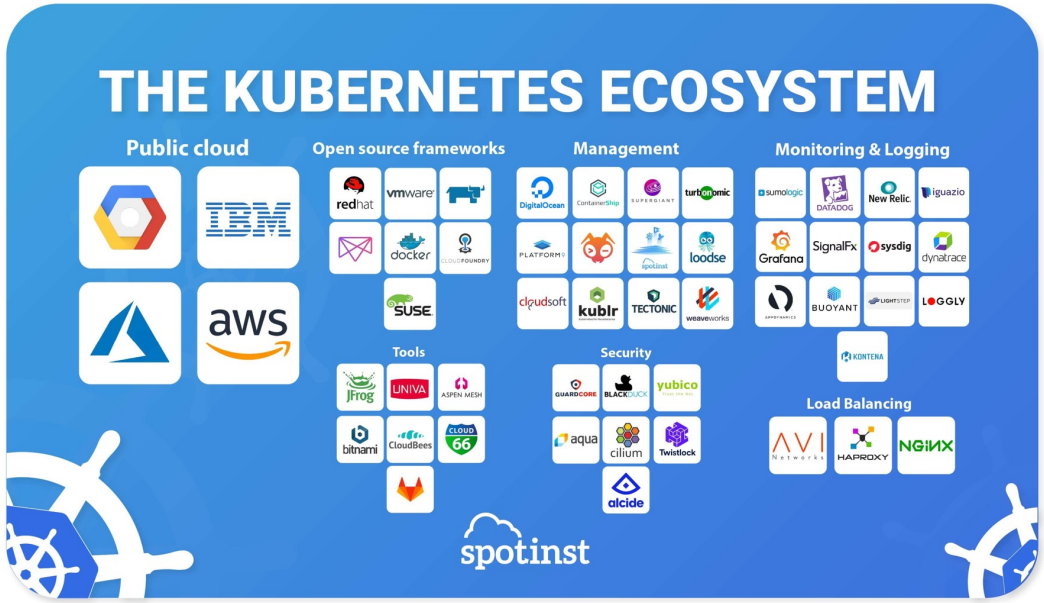
Hadoop Ecosystem



Processing is good, how do I serve?

- Hadoop and Spark are good for analysis, management, and storage.
- Now how can we serve in a distributed way?
- How do we scale?
- CONTAINERS, CONTAINERS, CONTAINERS!
- Think of it as VMs inside VMs.
- Package everything into standalone modules.
  - All the dependencies, scripts, and other goodies get packaged together.
- You already used it for your first assignment.
  - Docker is a containerization solution

Containers



- These packages are spun up as needed.
- A single VM can serve multiple containers.
- In your 1st assignment, you had 1 application per VM.
  - Now, imagine having multiple flask apps running on a single instance.
  - Each accepting requests on different ports.
- This way, instead of serving a single user at a time, we can serve multiple.
- Multiple VMs, multiple containers per VM:
  - Maximize resource allocation.

2nd assignment

Serving ML models on AWS

- You will deploy an ML model on multiple containers on multiple VMs
- Requests are forwarded to VMs, each VM will decide which container will respond

MapReduce on AWS

- You will solve the friend recommendation problem
- You will design the mappers
- You will design the reducers
- You deploy them on a VM