Data Analysis

(continuing model failures and monitoring)

Infrastructure and ML

Announcements

- Clinic 2 grading: Trying to get these by this week
- A (mock-up) Testvision exam is available. You can login and get familiar with the system and the types of questions
- Thursday 1330-1530: Course wrap-up and exam prep
  Submit topics to discuss: https://app.wooclap.com/UMDA

Models failures and monitoring

Refer to L09, L10:

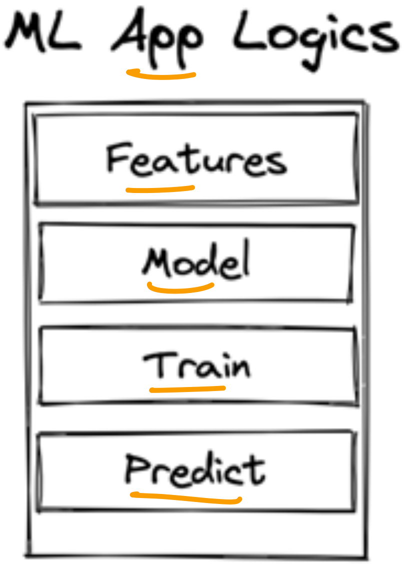
Data distribution shifts

Monitoring and observability

ML systems are complex

Evolving Topic

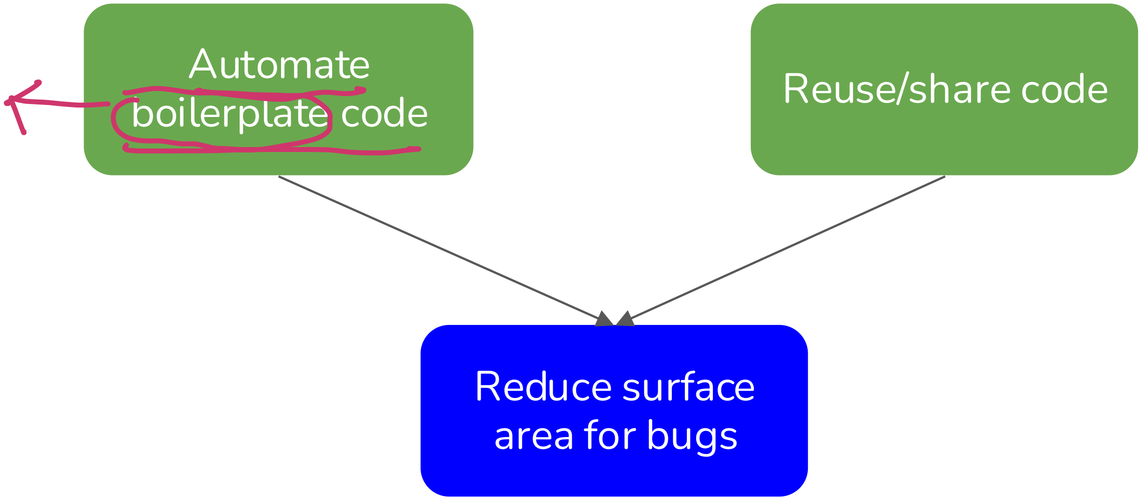
- More components
  - A request might jump 20-30 hops before response
  - A problem occurs, but where?



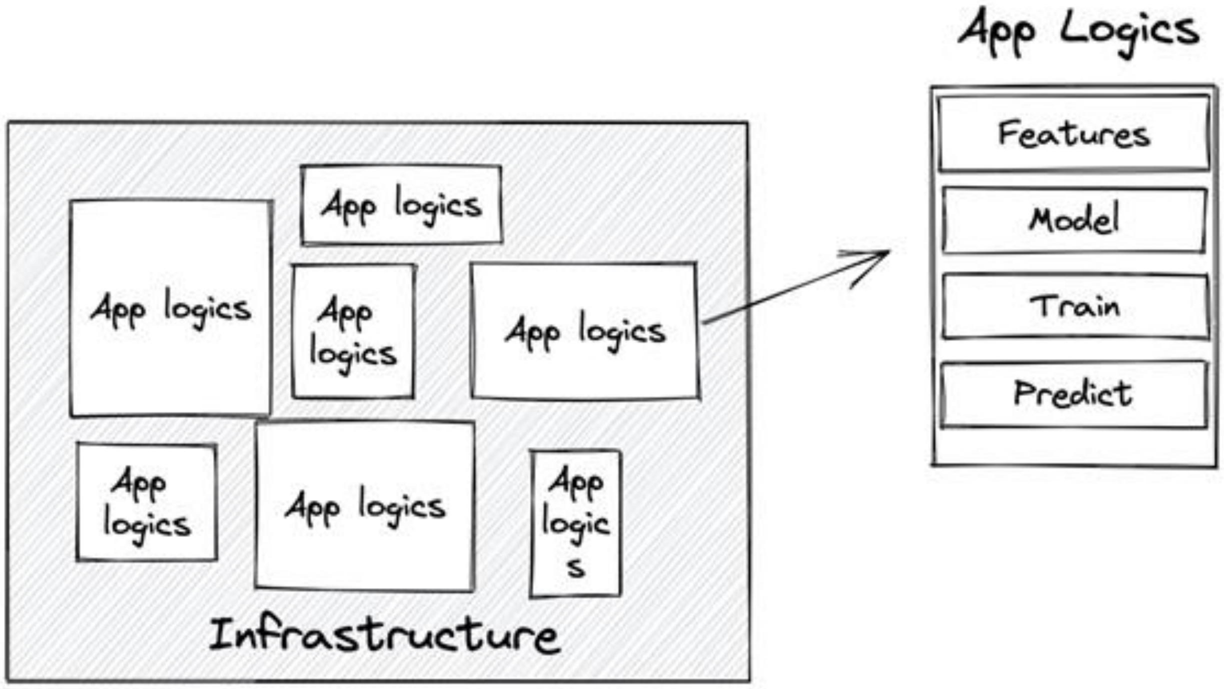
What we are Busy

How to know where an Issue occurs...

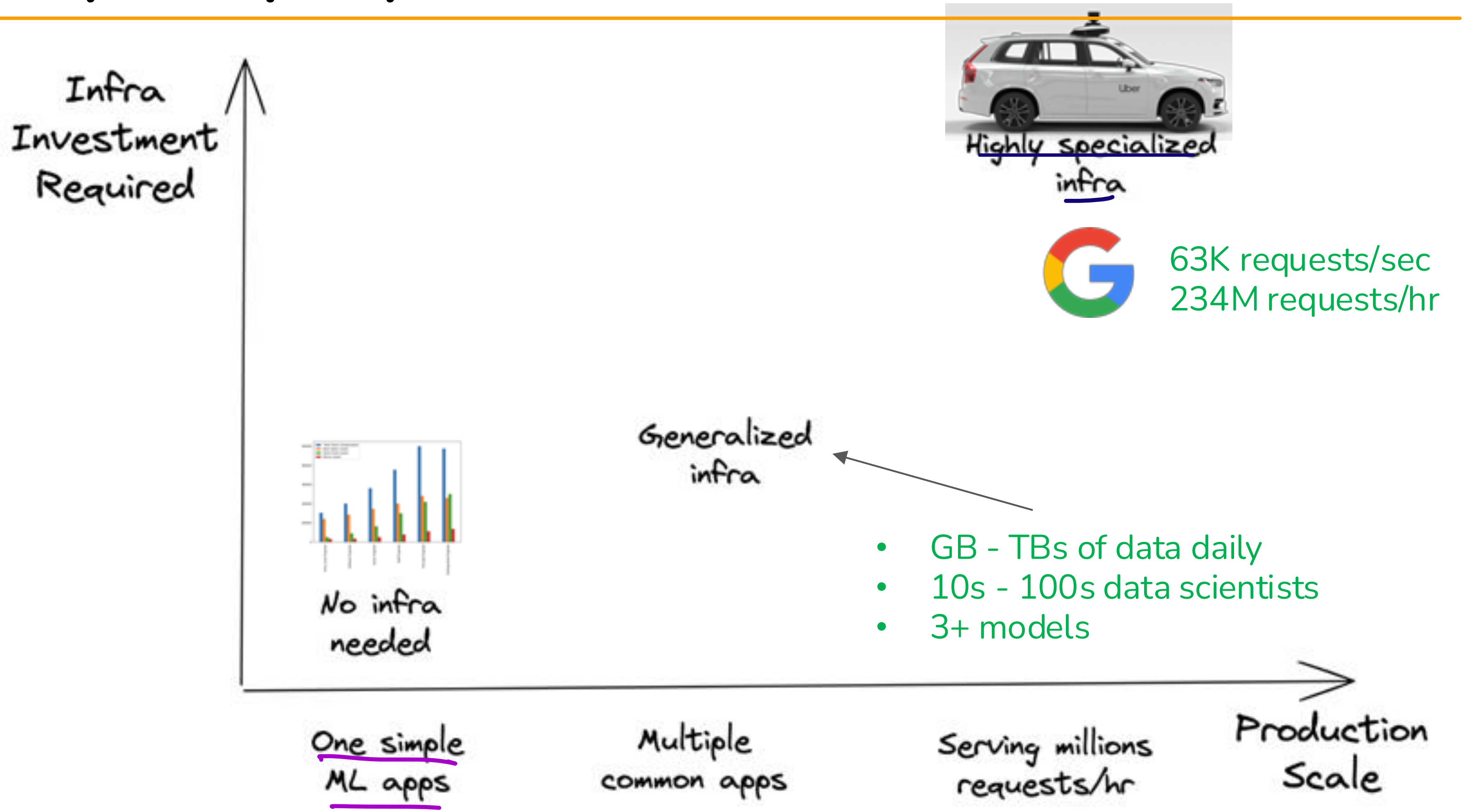
More complex systems, better infrastructure needed



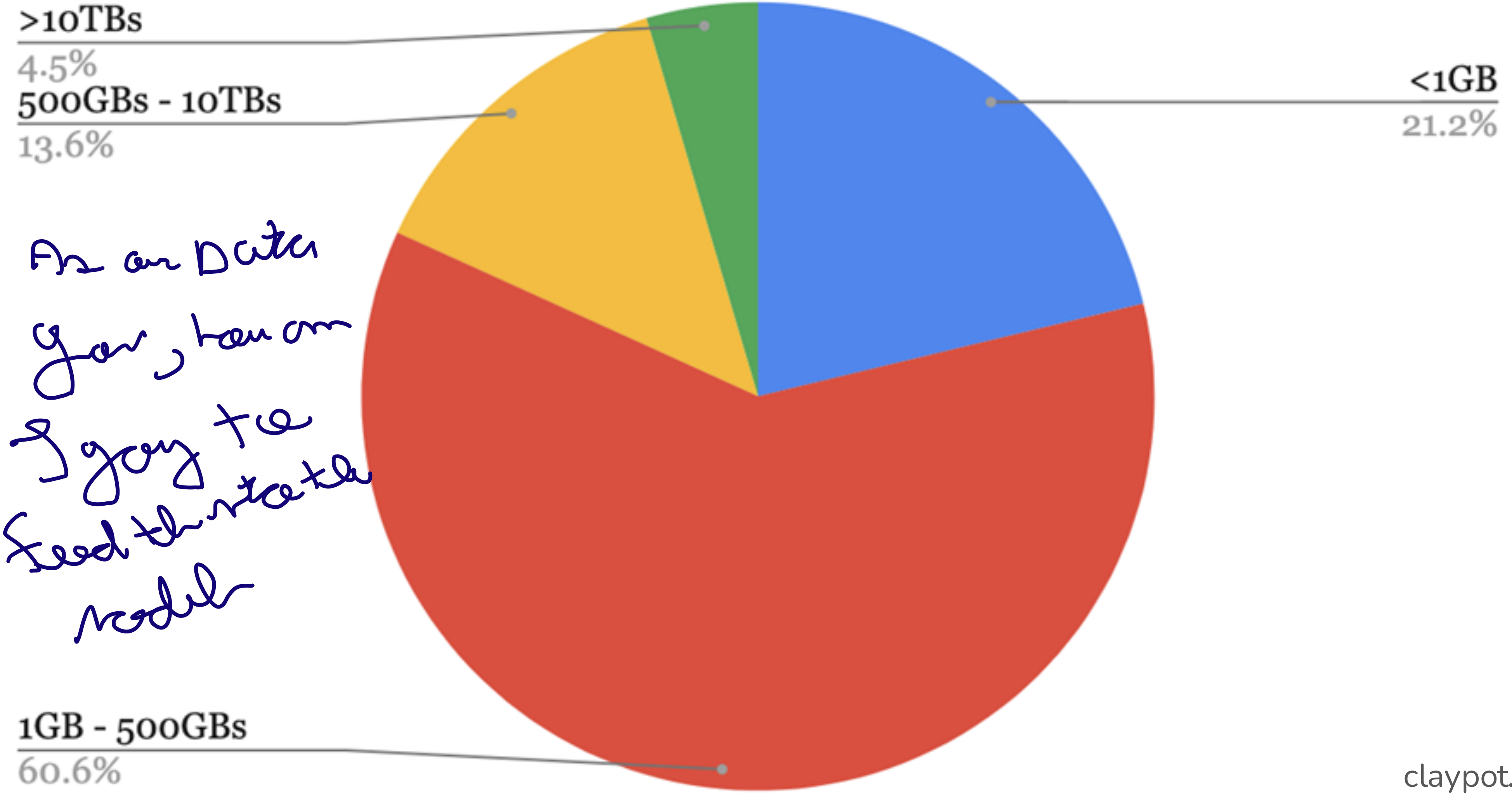
- Infrastructure: the set of fundamental facilities and systems that support the sustainable functionality of households and firms.
- ML infrastructure: the set of fundamental facilities that support the development and maintenance of ML systems.



Every company's infrastructure needs are different

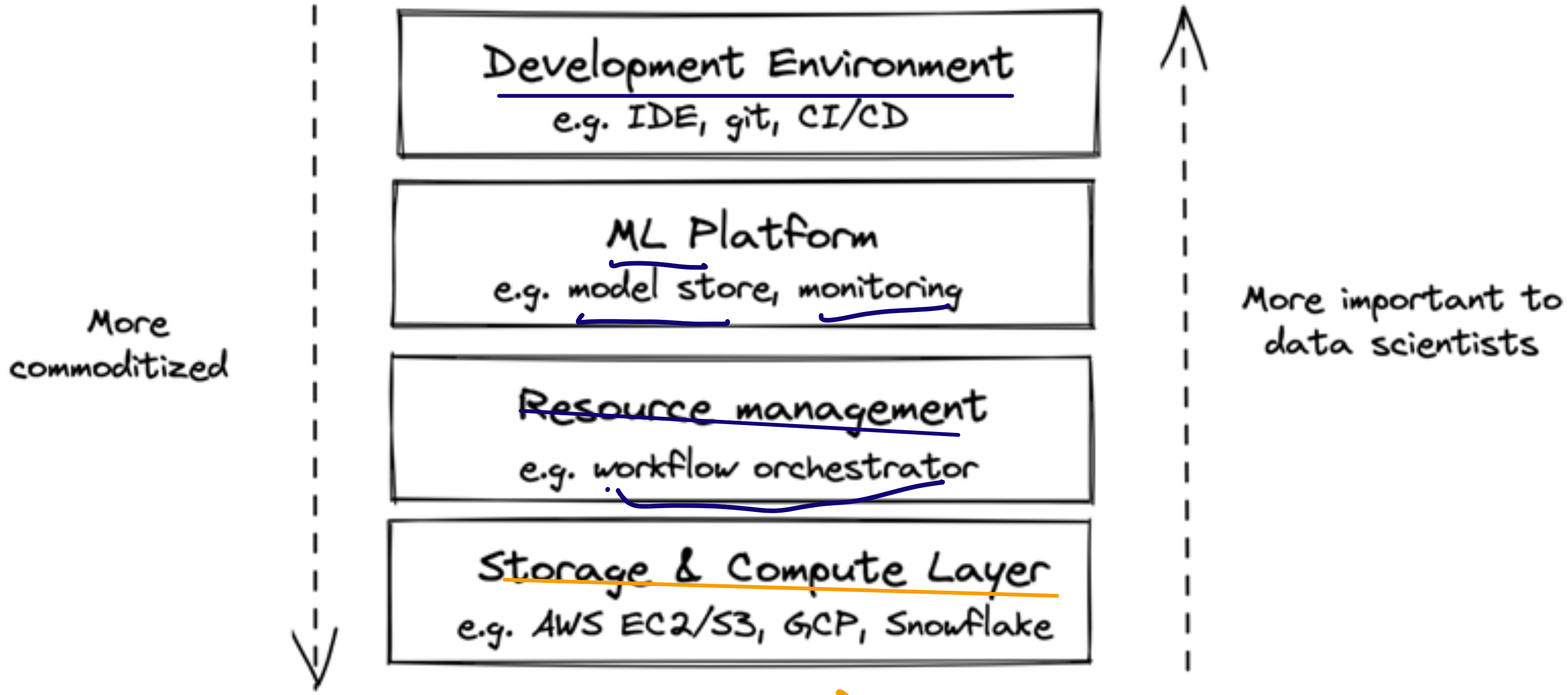


Amount of data the largest ML model handles



As our Data grows, how am I going to feed that into the model

Infrastructure Layers



Different Terms

Development Environment

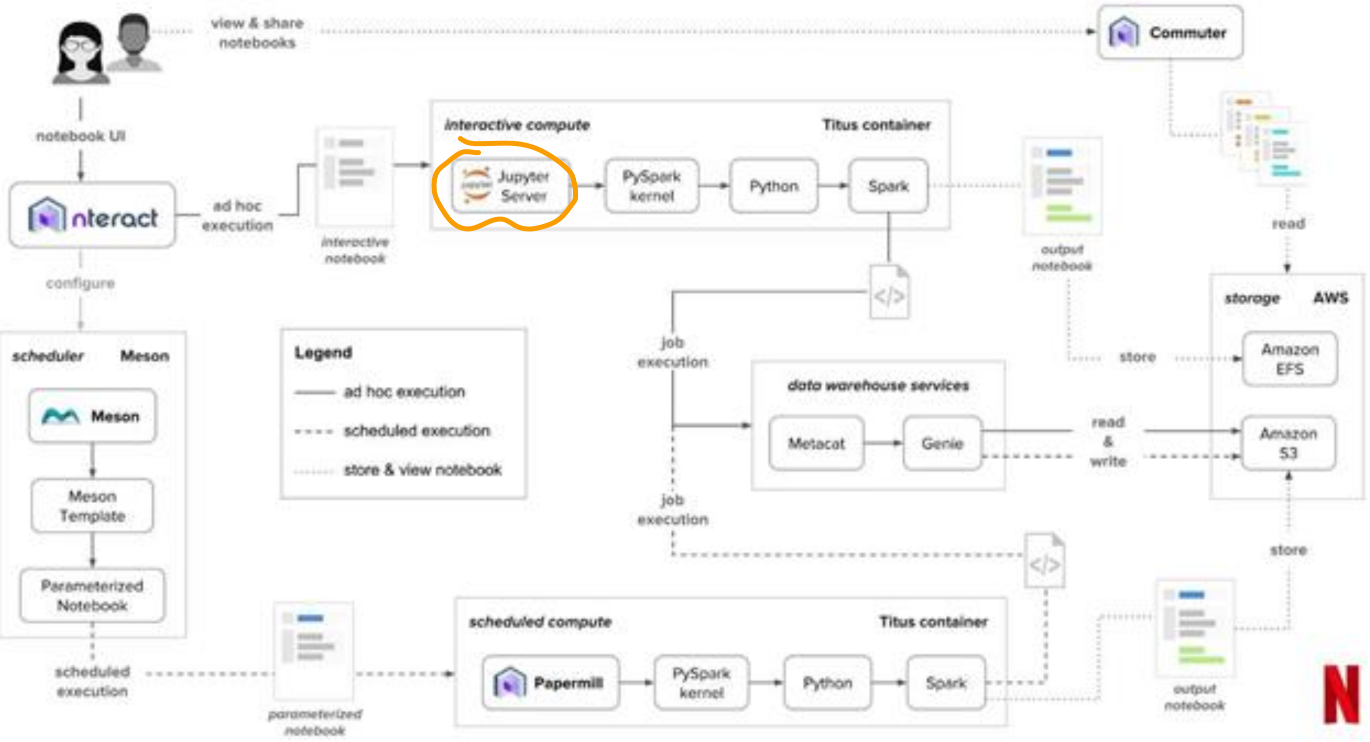
Development Environment

- Text editors & notebooks
  - Where you write code, e.g. VSCode, Vim
- Notebook: Jupyter notebooks, Colab
  - Also works with arbitrary artifacts that aren't code (e.g. images, plots, tabular data)
  - Stateful we do not need to re-run the whole thing
    - Only need to run from the failed step instead from the beginning

```
In [1]: import pandas as pd

In [2]: fname = "large-dataset.csv"

In [3]: df = pd.read_csv(fname)

In [4]: features = df["Timestamp", "Cost"]

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/miniconda3/envs/stove39/lib/python3.9/site-packages/pandas/core/indexes/base.py
ance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
```

Development Environment

- Notebook at Netflix



Development Environment

- Versioning
  - Git: code versioning
  - DVC: data versioning
  - WandB: experiment versioning
- CI/CD test suite: test your code before pushing it to staging/prod
  - Catch bugs early, automate testing, ensure reliability and quality

“if you have time to set up only one piece of infrastructure well, make it the development environment for data scientists.”

Ville Tuulos, *Effective Data Science Infrastructure* (2022)

Dev-to-prod: Container

- Step-by-step instructions on how to recreate an environment in which your model can run:
    - install this package
    - download this pretrained model
    - set environment variables
    - navigate into a folder
    - etc.

Transformers Dockerfile

- CUDA/cuDNN
- bash/git/python3
- Jupyter notebook
- TensorFlow/Pytorch
- transformers

```
FROM nvidia/cuda:10.2-cudnn7-devel-ubuntu18.04
LABEL maintainer="Hugging Face"
LABEL repository="transformers"

RUN apt update && \
    apt install -y bash \
                   build-essential \
                   git \
                   curl \
                   ca-certificates \
                   python3 \
                   python3-pip && \
    rm -rf /var/lib/apt/lists

RUN python3 -m pip install --no-cache-dir --upgrade pip && \
    python3 -m pip install --no-cache-dir \
    jupyter \
    tensorflow \
    torch

RUN git clone https://github.com/NVIDIA/apex
RUN cd apex && \
    python3 setup.py install && \
    pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

WORKDIR /workspace
COPY . transformers/
RUN cd transformers/ && \
    python3 -m pip install --no-cache-dir .

CMD ["/bin/bash"]
```

ML Platform

Model platform: story time

1. Jos started working on recsys at company X
2. To deploy recsys, Jos's team need to build tool like model deployment, model store, feature store, etc.
3. Other teams at X started deploying models and needed to build the same tools
4. X decided to have a centralized platform to serve multiple ML use cases

ML Platform

ML platform: key components

- Data management
- Model deployment
- Model store
- Feature store
- ...

aws SageMaker

Vertex AI

AZURE MACHINE LEARNING

Cloud

mlflow

Kubeflow

METAFLOW

OpenSource

Scaling as well

Resource Management

Resource management

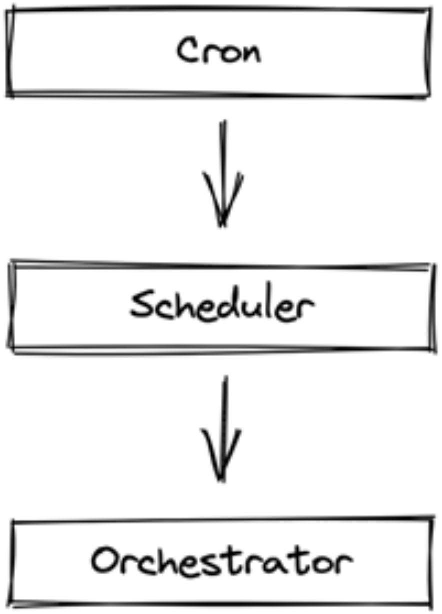
Big Result : Cloud

	Pre-cloud	Cloud
Resources	Finite	Practically infinite
Implication	More resources for an app = less resources for other apps Shared Resource we call	More resources for an app don't have to affect other apps
Goal	Utilization had to be done	Utilization + cost efficiency

ML workloads

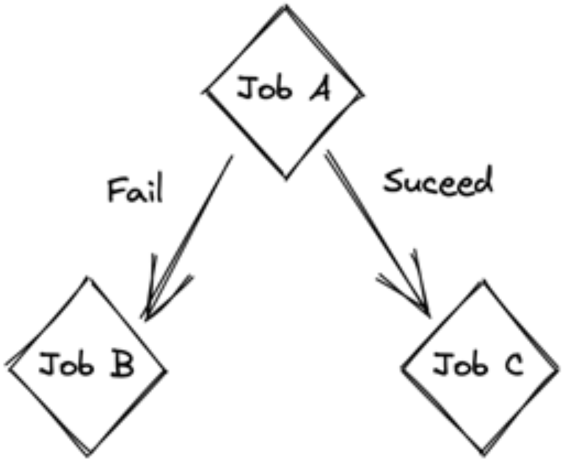
- Repetitive
  - Batch training and/or prediction
  - Periodical retraining
  - Periodical analytics
- Dependencies
  - E.g. train depends on features/pre-processing

Rely on



Cron: extremely simple

- cron is a **time-based job scheduler** in Unix-like operating systems
- Schedule jobs to run at fixed time intervals
- Report the results

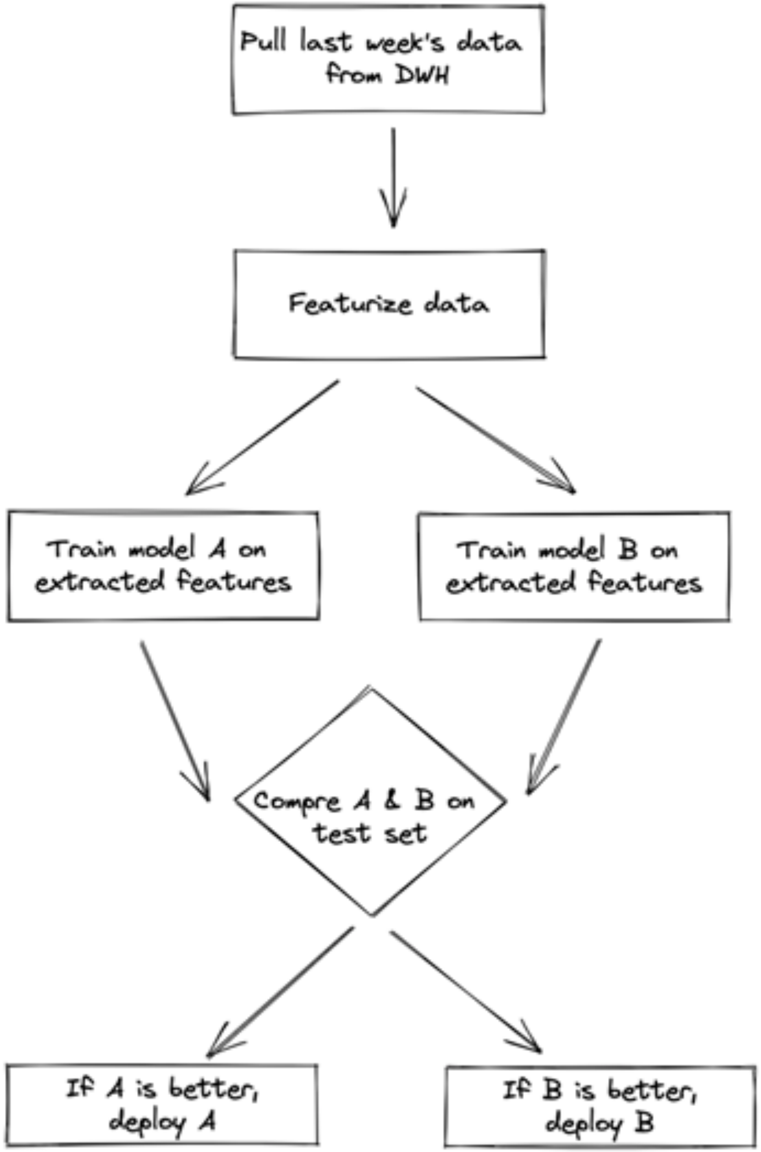


Cron can't
handle this

Dependent cant
Handle

Scheduler

- Most schedulers require you to specify your workloads as DAGs

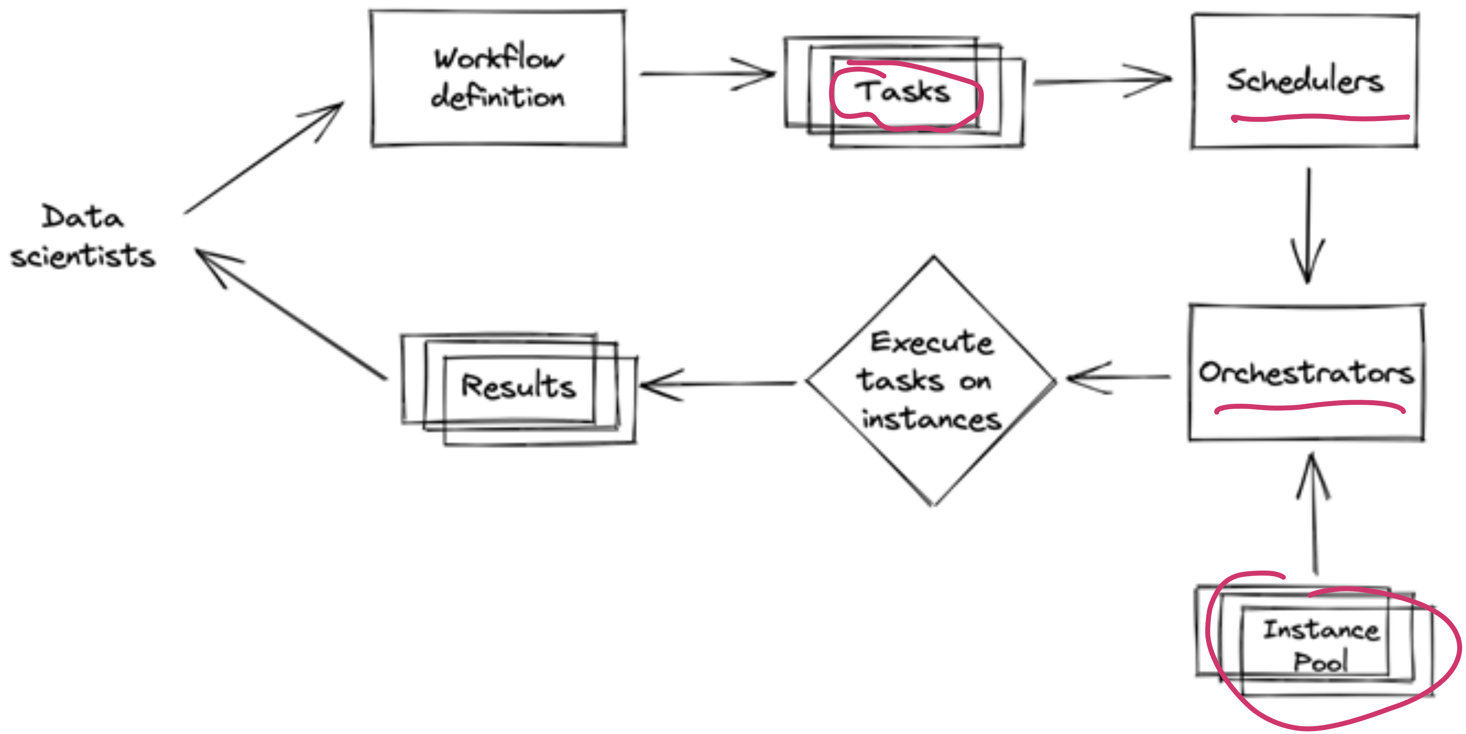


Scheduler to Orchestrator

- Scheduler: **when** to run jobs
  - Handle jobs, queues, user-level quotas, etc.
  - Typically used for periodical jobs like batch jobs
- Orchestrator: **where** to run jobs
  - Handle containers, instances, clusters, replication, etc.
  - Provision: allocate more instances to the instance pool as needed
  - Typically used for long-running jobs like services

Long Run

Data science workflow management



Storage & Compute Layer

Storage

- Where data is collected and stored
- Simplest form: HDD, SSD
- More complex forms: data lake, data warehouse
- Examples: S3, Redshift, Snowflake, BigQuery

- Recall first week lecture on data formats, models and storage!

Storage: heavily commoditized

- Most companies use storage provided by other companies (e.g. cloud)
- Storage has become so cheap that most companies just store everything

- *"Store it, just in case we need it"*

Compute layer: engine to execute your jobs

- Compute resources a company has access to
- Mechanism to determine how these resources can be used

- Simplest form: a single CPU/GPU core
- Most common form: cloud compute

Compute unit

- Compute layer can be sliced into smaller compute units **to be used concurrently**
  - A CPU core might support 2 concurrent threads, **each thread** is used as a compute unit to execute its own **job**
  - Multiple CPUs can be joined to form a **large compute unit** to execute a large **job**

Unit: job

DASK

APACHE Spark

RAY

Unit: pod

Wrapper around container

kubernetes

Compute layer: how to execute jobs

1. Load data into memory Bottleneck
2. Perform operations on that data
    a. Operations: add, subtract, multiply, convolution, etc.

To add arrays A and B

1. Load A & B into memory
2. Perform addition on A and B

If A & B don't fit into memory, it'll be possible to do the ops without out-of-memory algorithms

Important metrics of compute layer:

1. Memory
2. Speed of computing ops

Compute layer: memory

- Amount of memory
    - Straightforward
    - An instance with 8GB of memory is more expensive than an instance with 2GB of memory
- I/O bandwidth: speed at which data can be loaded into memory

Compute layer: speed of ops

- Most common metric: FLOPS
    - **F**loating **P**oint **O**perations **P**er **S**econd

"A Cloud TPU v2 can perform up to 180 teraflops, and the TPU v3 up to 420 teraflops."

- Google, 2021

Compute layer: speed of ops

- Most common metric: FLOPS
- Contentious
  - What exactly is an ops?
    - If 2 ops are fused together, is it 1 or 2 ops?
  - Peak performance at 1 teraFLOPS doesn't mean your app will run at 1 teraFLOPS

Compute layer: utilization

- Utilization = actual FLOPS / peak FLOPS

If peak 1 trillion FLOPS but job runs 300 billion FLOPS
-> utilization = 0.3

- Dependent on how fast data can be loaded into memory

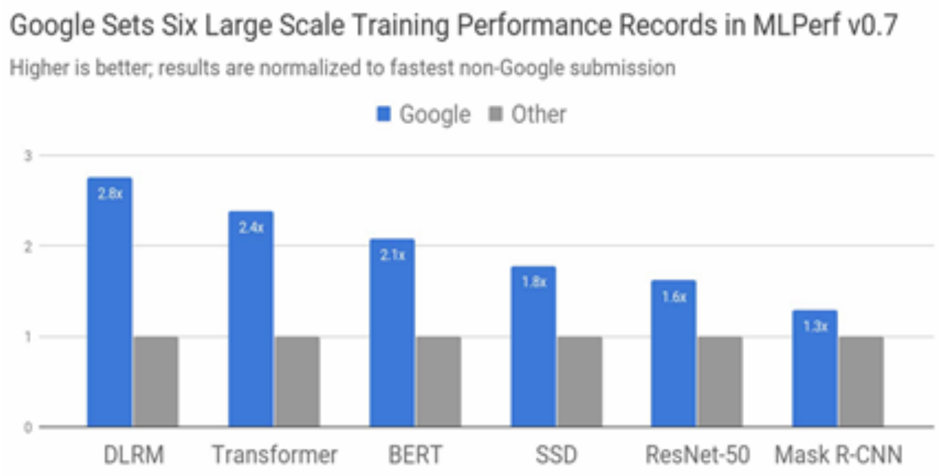
*Tensor Cores are very fast. So fast ... that they are idle most of the time as **they are waiting for memory to arrive from global memory.***

*For example, during BERT Large training, which uses huge matrices — the larger, the better for Tensor Cores — **we have utilization of about 30%**.*

- [Tim Dettmers, 2020]

Compute layer: if not FLOPS, then what?

- How long it will take this compute unit to do common workloads
- MLPerf measure hardware on common ML tasks e.g.
  - Train a ResNet-50 model on the ImageNet dataset
  - Use a BERT-large model to generate predictions for the SQuAD dataset
- Also contentious



Compute layer: evaluation

- Memory
- Cores
- I/O bandwidth
- Cost

Some GPU instances on AWS

Instance	GPUs	vCPU	Mem (GiB)	GPU Mem (GiB)
p3.2xlarge	1	8	61	16
p3.8xlarge	4	32	244	64
p3.16xlarge	8	64	488	128
p3dn.24xlarge	8	96	768	256

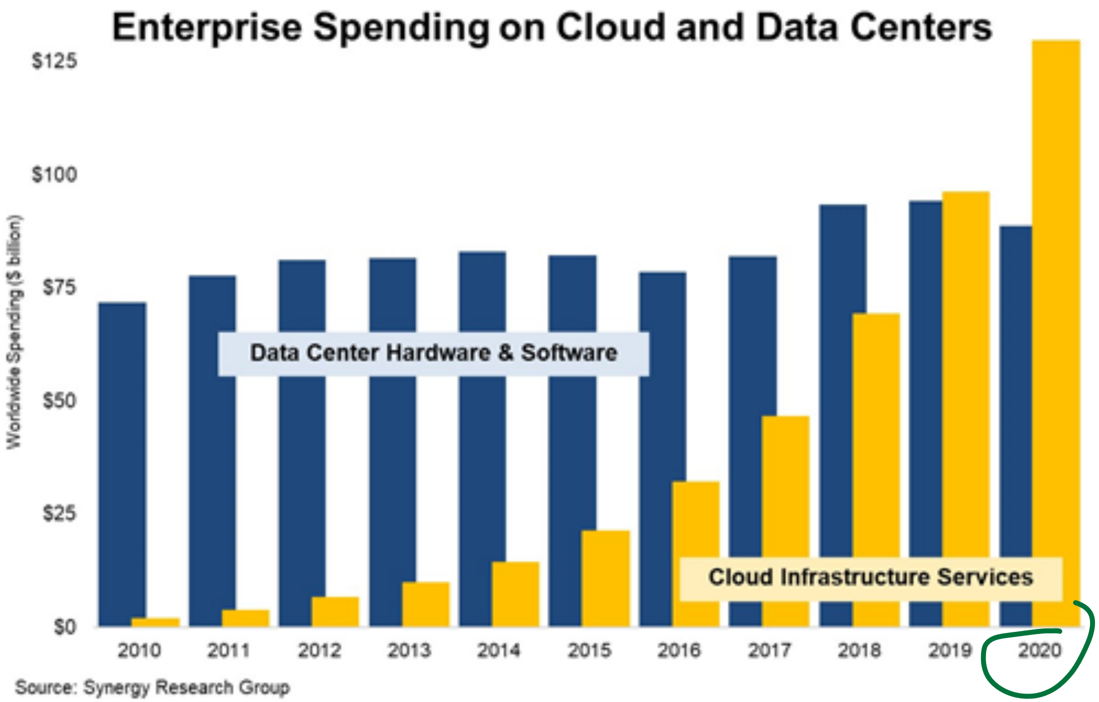
Some TPU instances on GCP

TPU type (v2)	v2 cores	Total memory
v2-8	8	64 GiB

TPU type (v3)	v3 cores	Total memory
v3-8	8	128 GiB

Public Cloud vs. Private Data Centers

- Like storage, compute is largely commoditized

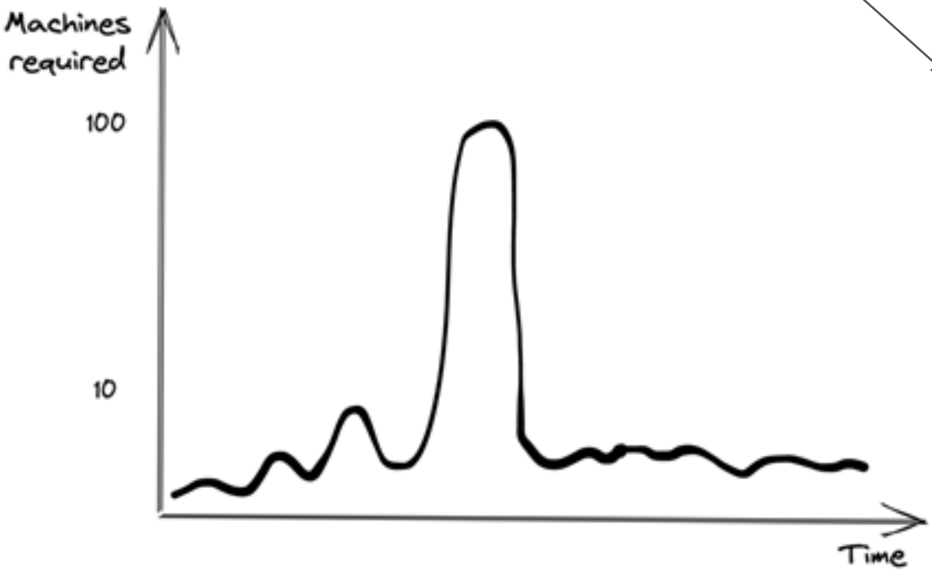


[2020 – The Year that Cloud Service Revenues Finally Dwarfed Enterprise Spending on Data Centers | Synergy Research Group]

Benefits of cloud

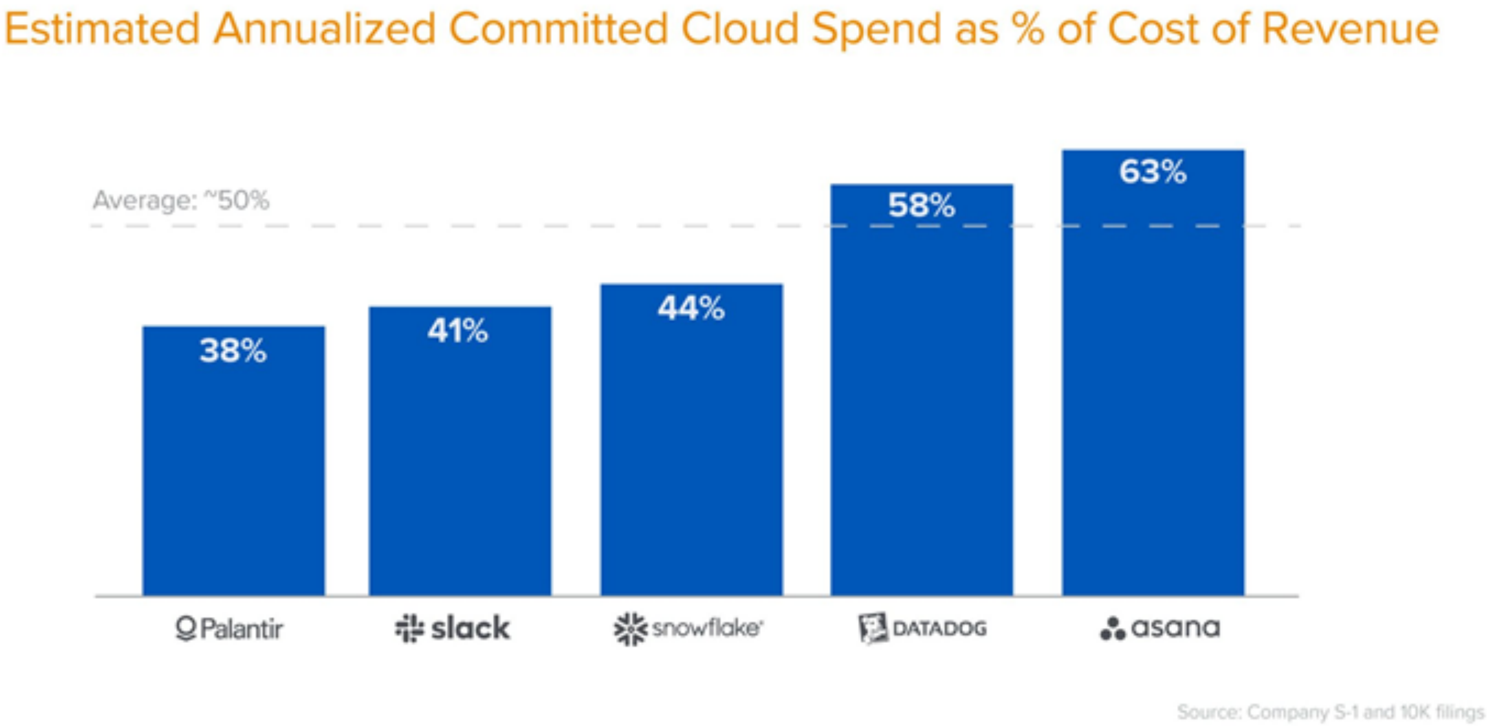
- Easy to get started
- Appealing to variable-sized workloads
    - Private: would need 100 machines upfront, most will be idle most of the time
    - Cloud: pay for 100 machines only when needed

Autoscaling!



Drawbacks of cloud: cost

- Cloud spending: ~50% cost of revenue



The Cost of Cloud, a Trillion Dollar Paradox | Andreessen Horowitz (2021)

Drawbacks of cloud: cost

"Across 50 of the top public software companies currently utilizing cloud infrastructure, an ***estimated $100B of market value is being lost … due to cloud impact on margins*** *— relative to running the infrastructure themselves."*

The Cost of Cloud, a Trillion Dollar Paradox | Andreessen Horowitz (2021)

Drawbacks of cloud: cost

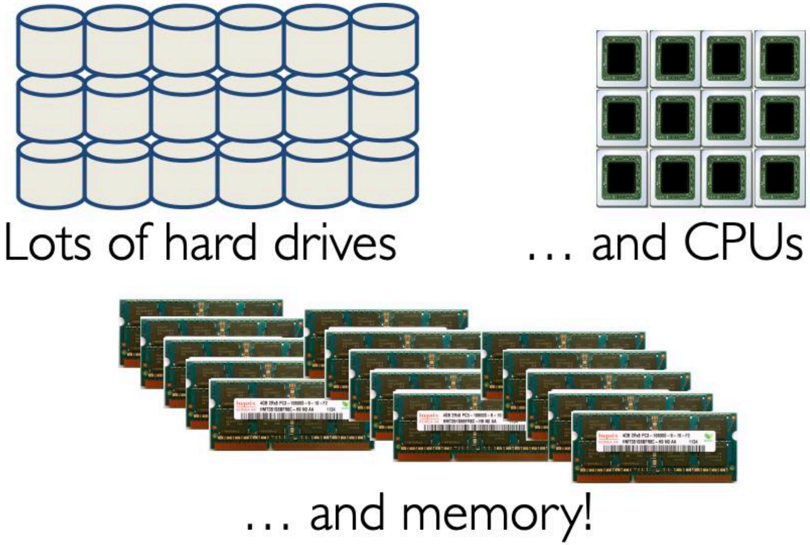
(py)Spark

Spark Facts

- Written in Scala (Highly effective)
- Originally implemented as an in-memory alternative to Hadoop MapReduce
- Started by Matei Zaharia in 2009 during his PhD at UC Berkeley
- Apache Foundation top project since 2014
- It is open-source and free
- Business support offered by Databricks, a company created by Zaharia

The big data problem

- One machine can not process or even store all the data!
- Solution is to distribute data over cluster of machines



The spark computing framework

- Provides programming abstraction and parallel runtime to hide complexities of fault-tolerance and slow machines

- “Here’s an operation, run it on all of the data”
- “I don’t care where it runs (you schedule that)”

Spark is fast

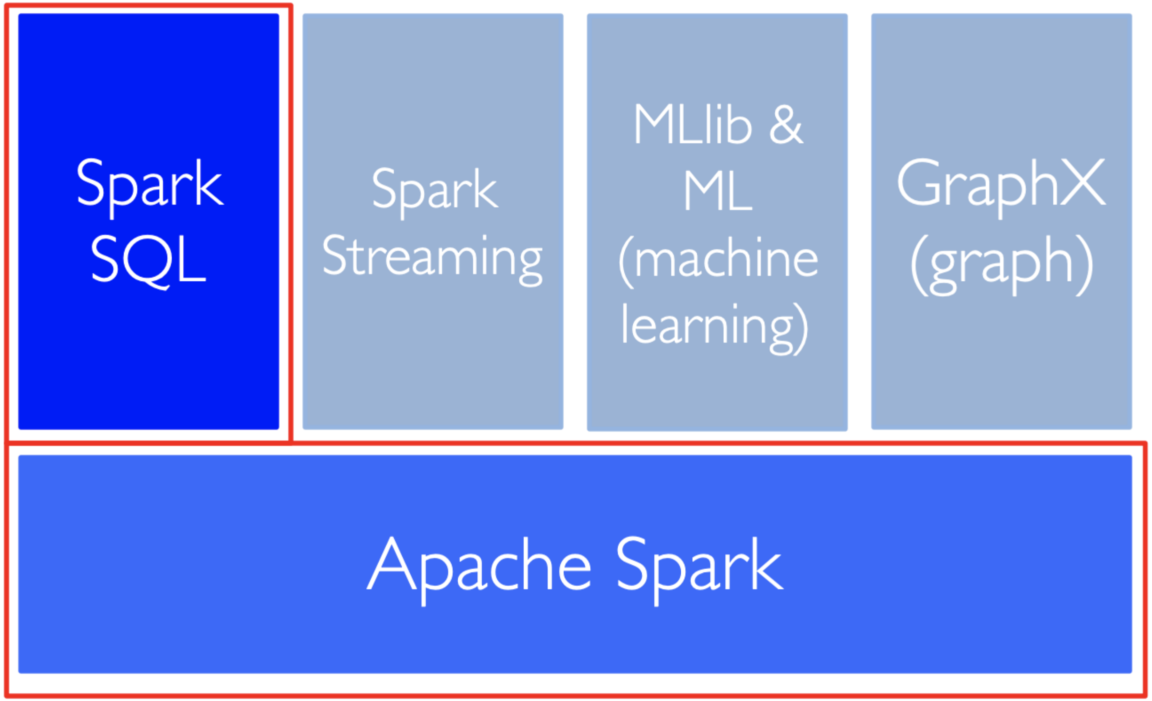
In 2014 Spark beat Hadoop MapReduce by sorting 100TB of data in 23 minutes, and since then it improved a lot

The previous world record was 72 minutes, set by a Hadoop MapReduce cluster of 2100 nodes. This means that Spark sorted the same data 3X faster using 10X fewer machines.

[More info]

Apache Spark Components

- PySpark is the Python API for Apache Spark
- Dataframes are important!

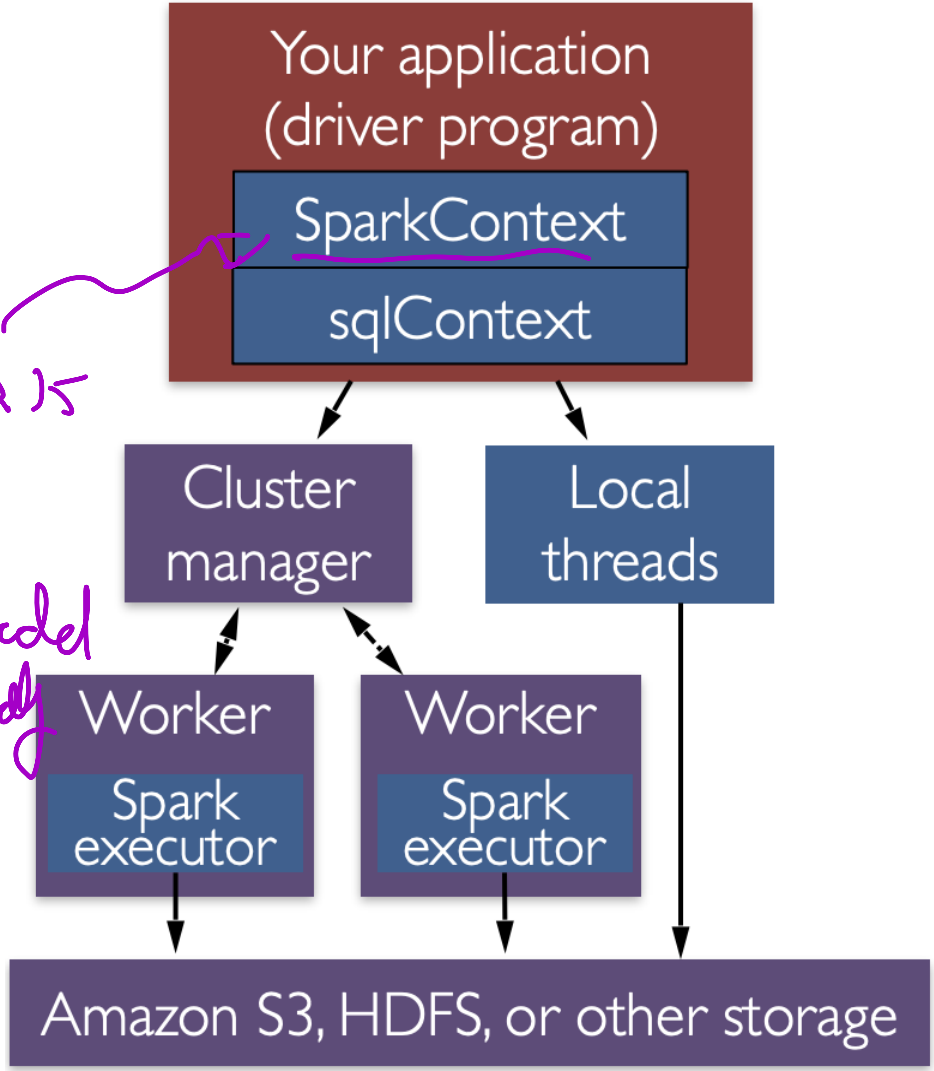


Spark Driver and Workers

- A Spark program is two programs:
  - A driver program Set Up the Context SPARK IS
  - A workers program
- Worker programs run on cluster nodes or in local thread
- DataFrames are distributed across workers

Decided Automatically

" I don't need to know the



Detail

Check notebook demo on spark!