Big-DAMA: Big Data Analytics and Platforms for Network Traffic Monitoring and Analysis

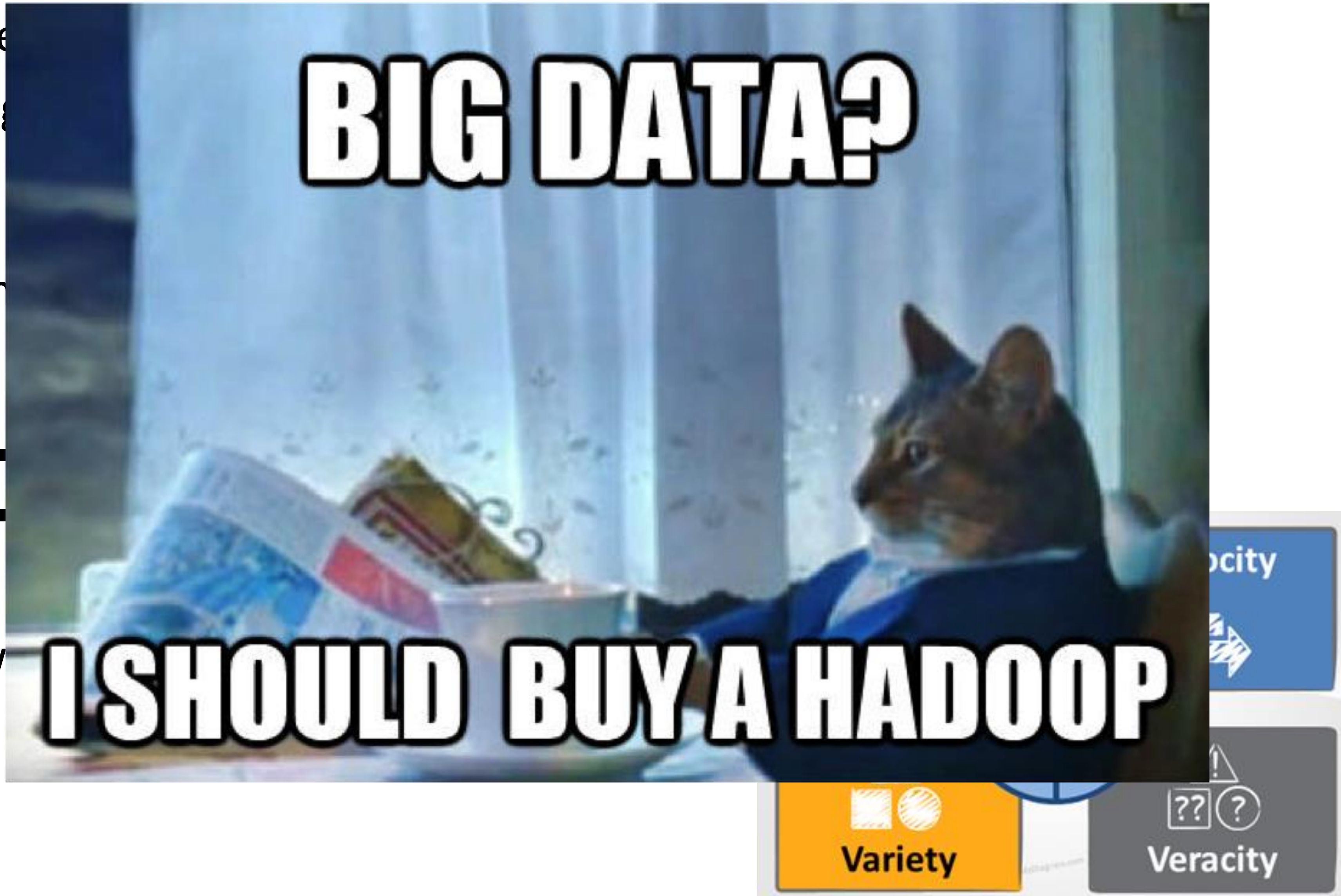
Dr. Pedro Casas (http://pcasas.info)

Senior Scientist – Data Science & Artificial Intelligence

AIT Austrian Institute of Technology

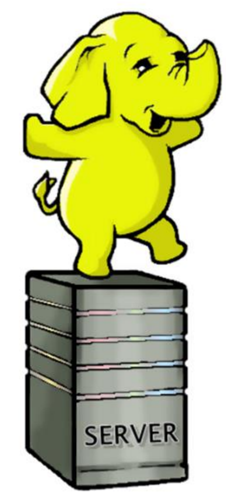
Big data in Network Traffic Monitornig

- Ne
- e.
- Th
- Al
- W



Big Data Frameworks for NTMA

The Apache Hadoop Ecosystem



In a nutshell: what‘s is Hadoop?

- Hadoop is a software **framework** for **storing, processing, and analyzing very large data sets** (big data)
- It’s **distributed** (HDFS + MapReduce)
- It’s **scalable** and runs on commodity hardware
- It’s **fault-tolerant** (it assumes hardware failures are common)
- **It’s open source!**
  - Overseen by the Apache Software Foundation
  - Active committers to core Hadoop from over 20 top-companies (Cloudera, Intel, LinkedIn, Facebook, Yahoo, etc.)
  - **“Hadoop ecosystem”:** hundreds more committers on other Hadoop-related projects and tools
  - Very active community, so system keeps growing and improving FAST!

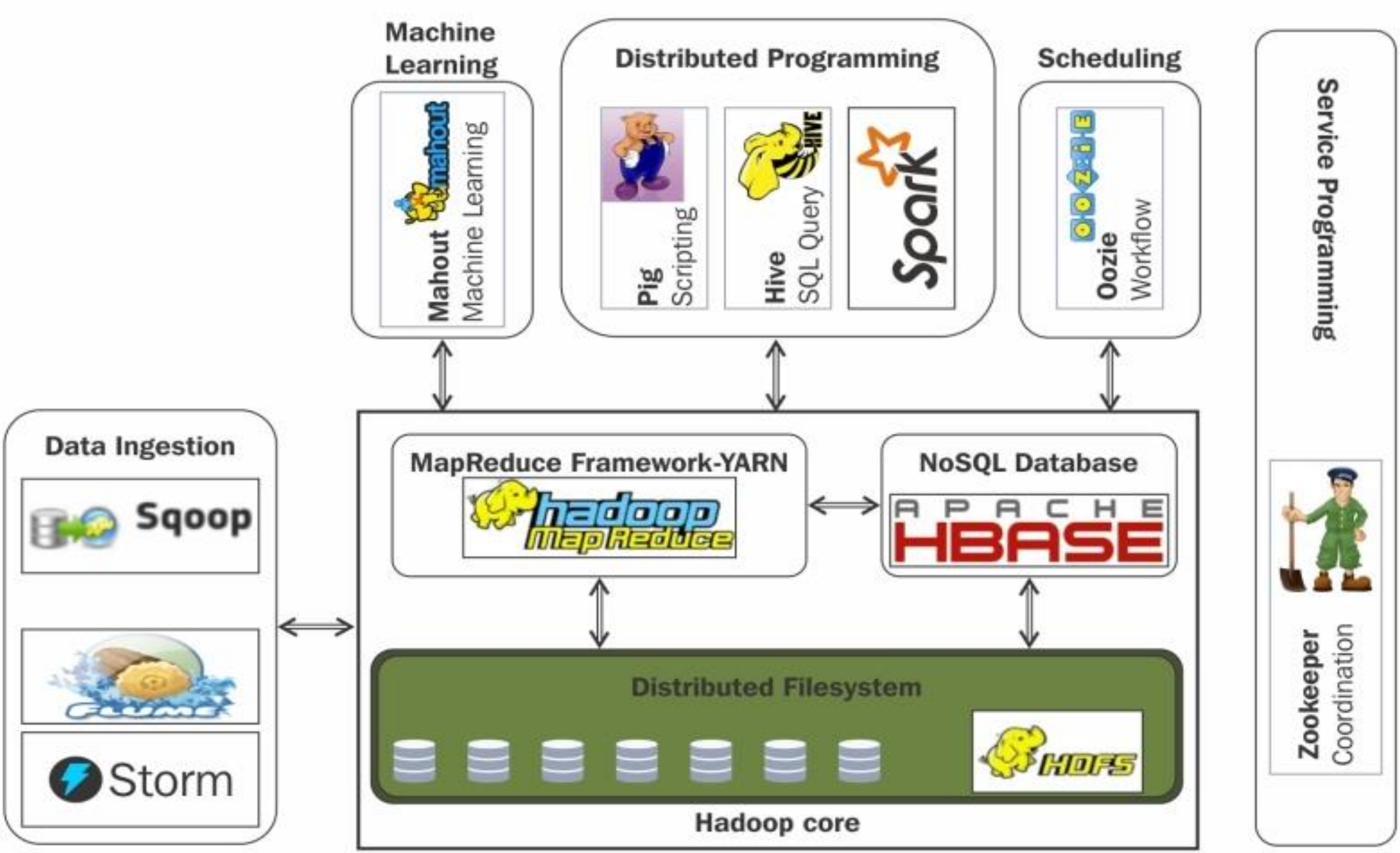
The (growing) Hadoop Ecosystem

see https://hadoopecosystemtable.github.io/

- Distributed Programming
- Hadoop SQL Databases
- NewSQL Databases
- NoSQL Databases
- Applications
- Data Ingestion
- Machine Learning
- Distributed Filesystems
- Security & Deployment
- Service Programming
- Benchmarking
- Scheduling

The (large and growing) Hadoop Ecosystem

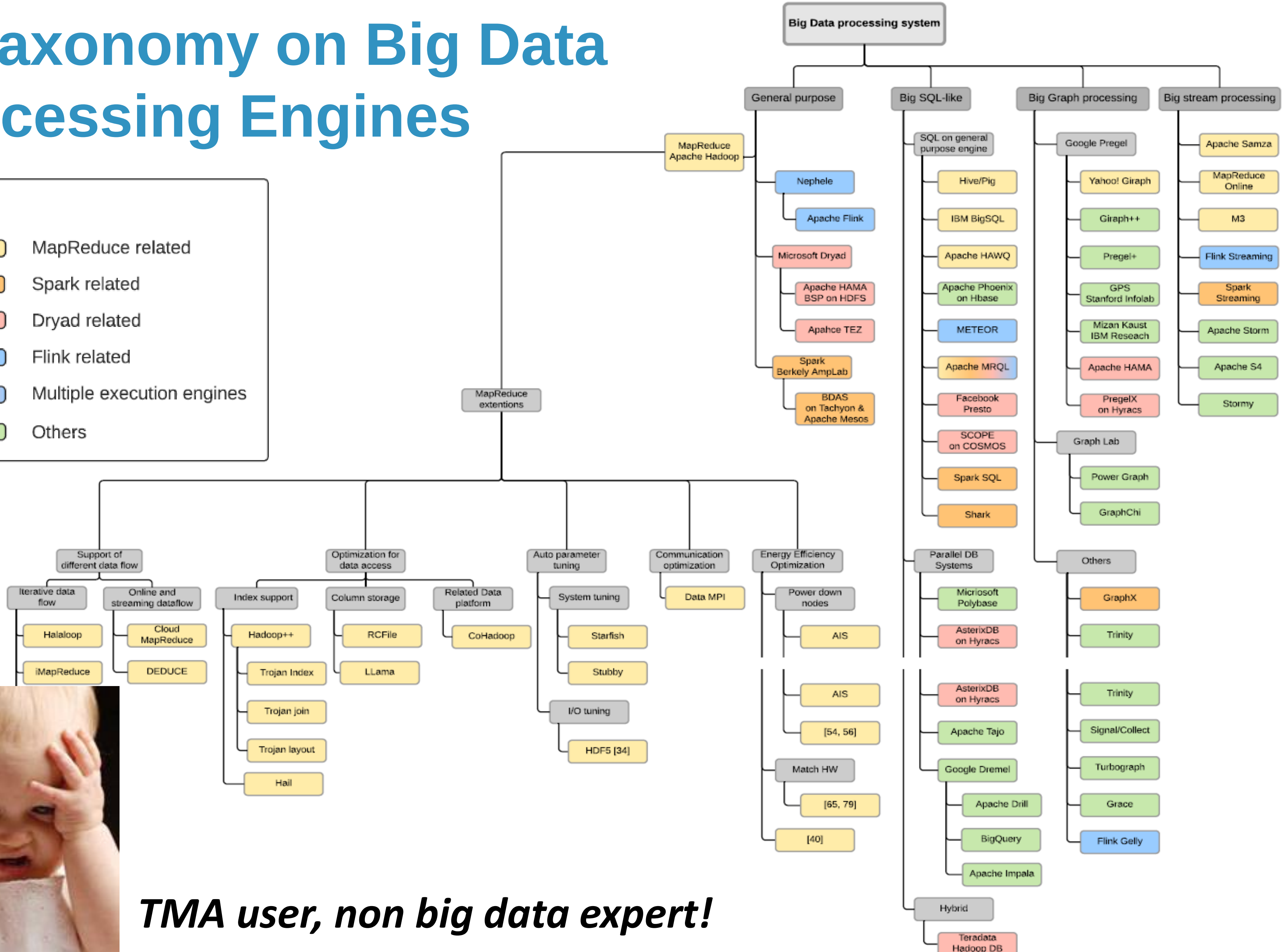
see https://hadoopecosystemtable.github.io/



Some Examples of the Hadoop Ecosystem

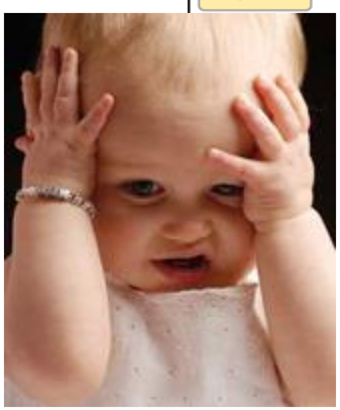
Project	Purpose
Spark	In-memory execution framework
Hbase	NoSQL database built on HDFS
Hive	SQL processing engine designed for batch workloads
Impala	SQL query engine designed for analytic workloads
Parquet	Highly efficient columnar data storage format
Sqoop	Data movement to/from RDBMSs
Flume **Kafka**	Streaming data ingestion
Solr	Highly efficient data-search engine
Hue	Web-based user interface for Hadoop
Oozie	Workflow scheduler used to manage jobs
Sentry	Authorization tool, providing security for Hadoop

A Taxonomy on Big Data Processing Engines



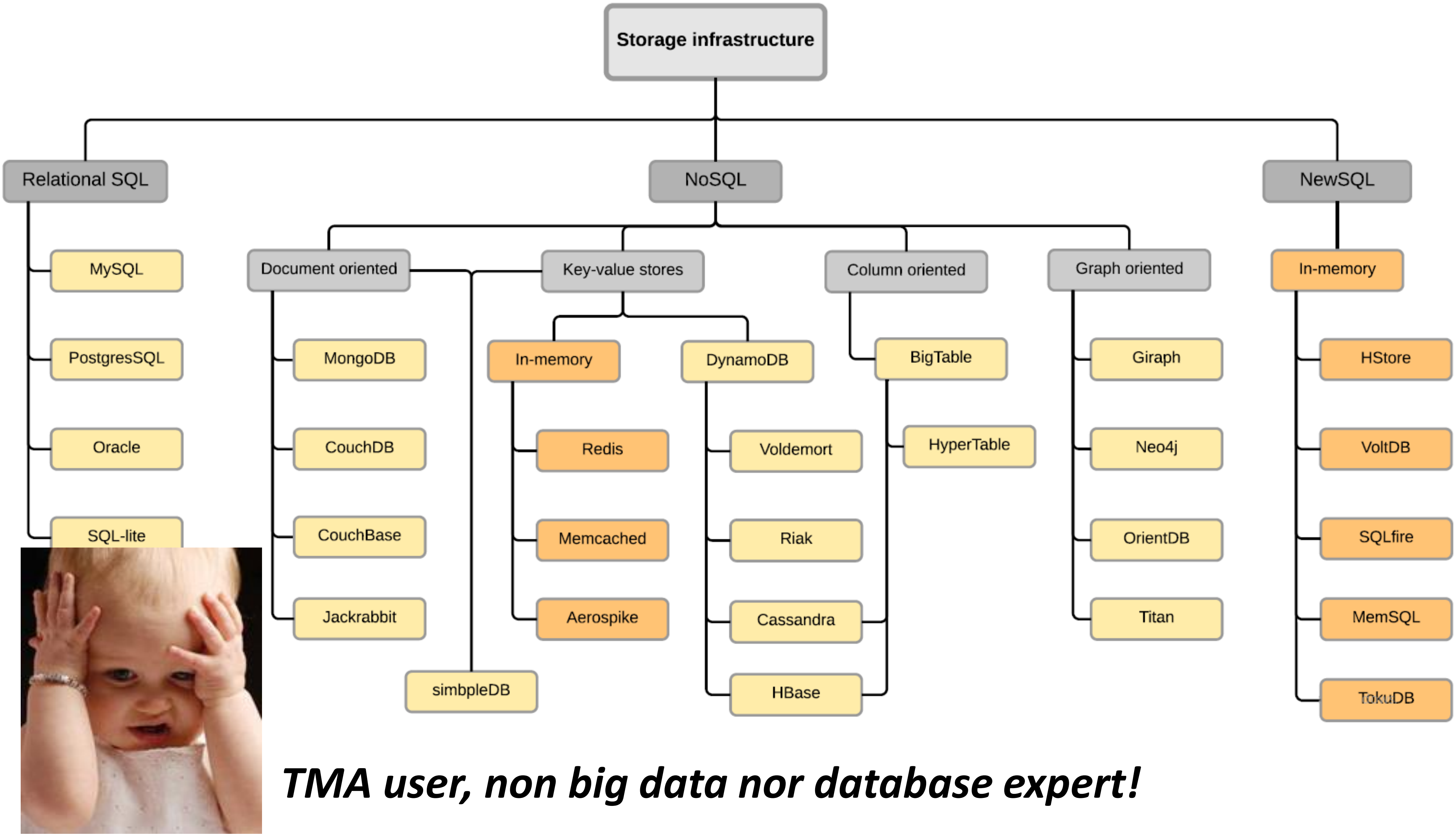
Legend

- MapReduce related
- Spark related
- Dryad related
- Flink related
- Multiple execution engines
- Others



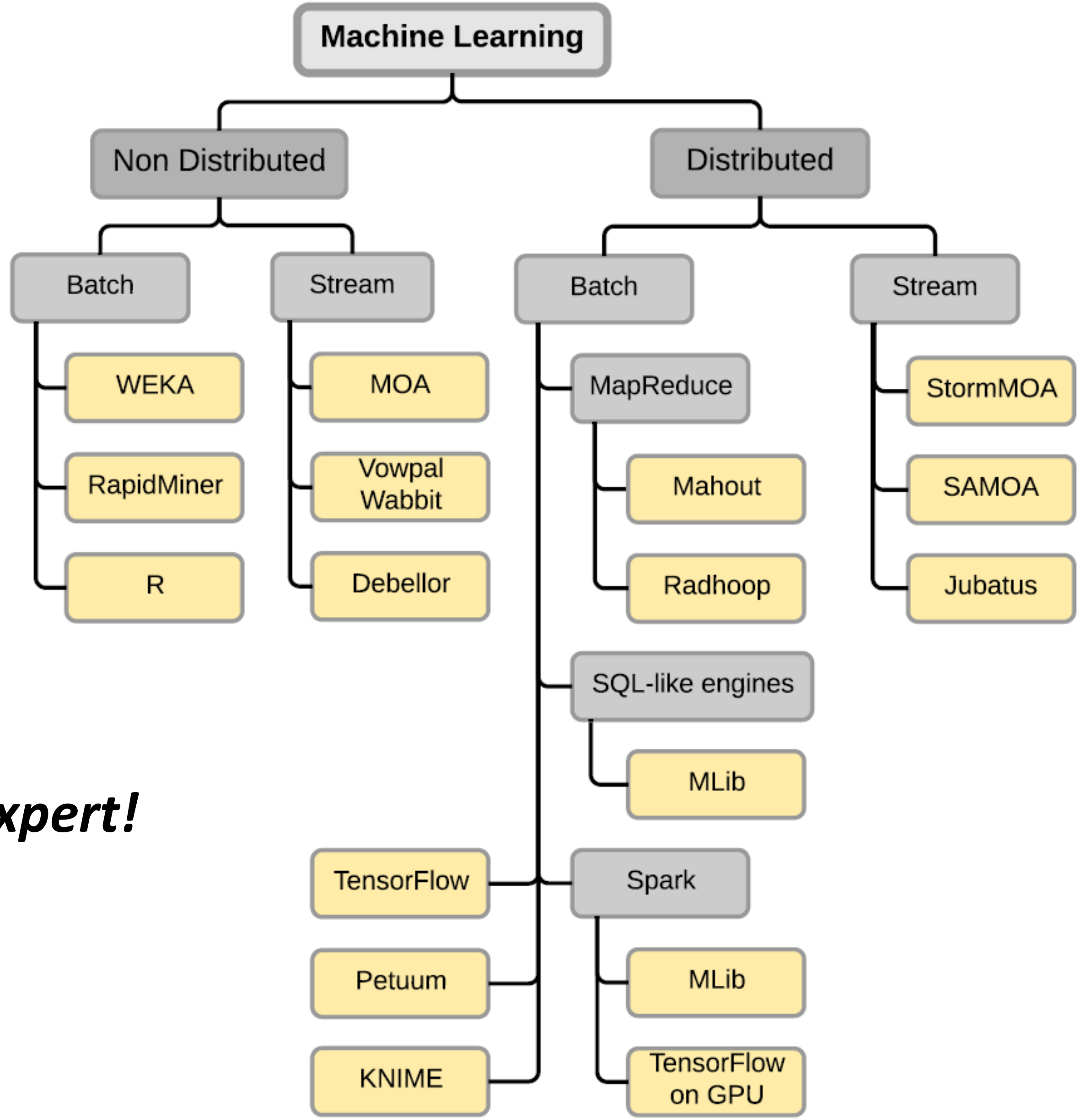
TMA user, non big data expert!

A Taxonomy on Big Data Storage Infrastructure



TMA user, non big data nor database expert!

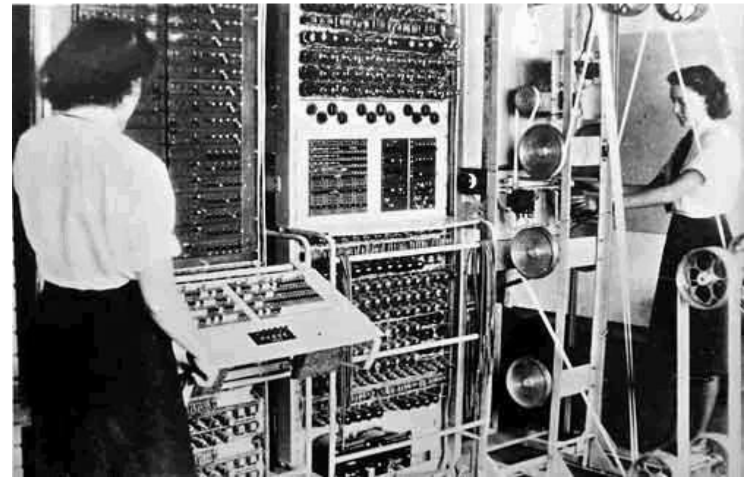
Machine Learning (Big-data) Systems and Libraries



TMA user,
non machine learning expert!

Why Hadoop? A bit of history....

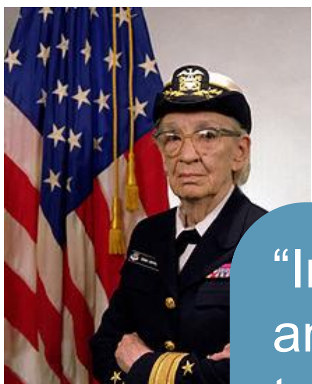
- **Traditionally, computation has been processor-bound:**
  - Relatively small amounts of data
  - Lots of complex processing



- **The early solution: bigger computers:**
  - Faster processor, more memory
  - But even this couldn't keep up

Distributed Systems

- **The solution:**
  - Distributed systems
  - Use a cluster of multiple machines for a single job

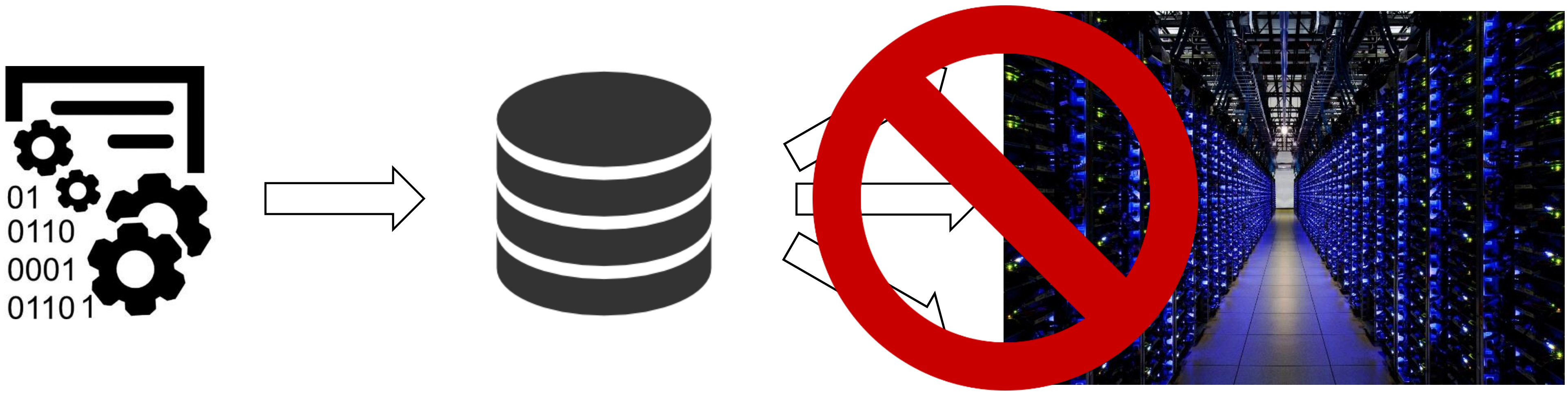


"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, we didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for *more systems* of computers."

– Grace Hopper (mother of compilers)

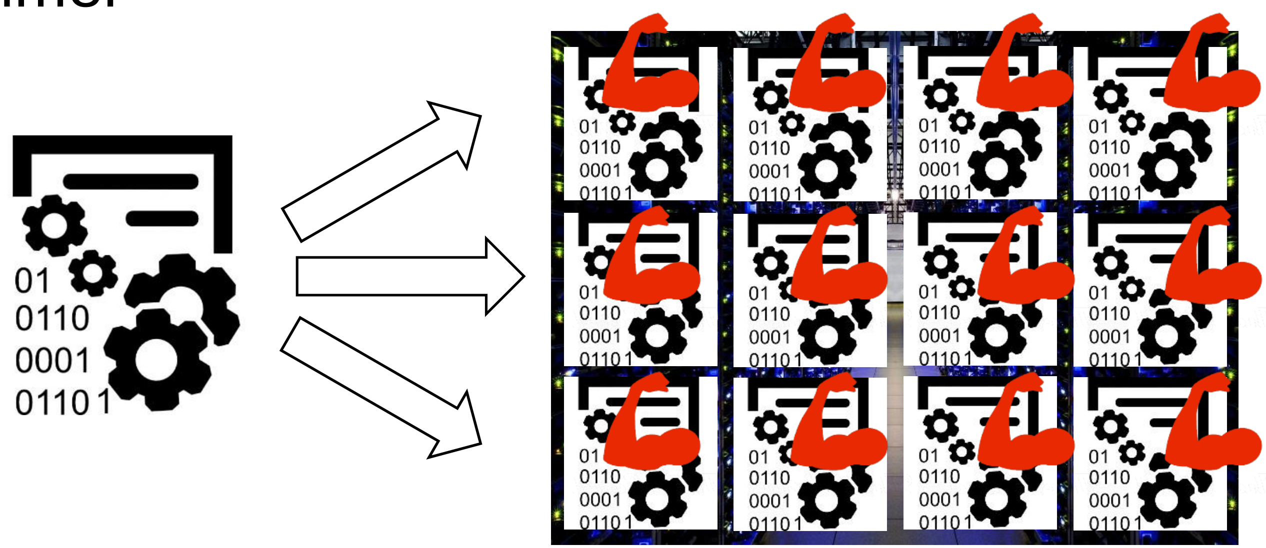
Distributed Systems: centralized storage

- Traditionally, data is stored in a central location
- Data is copied to processors at runtime
- OK for limited amounts of data**…but fails with BIG DATA**



The Hadoop Approach

- Two key concepts used by Hadoop's distributed computing
  - **Distribute data when it is loaded** into the system
  - **Run computation where the data is** stored
- Add capacity by **scaling *out*** (more machines), **not scaling *up*** (to a bigger machine)
- Hadoop makes **distributed computing "*transparent*"** to the programmer



A bit of history....

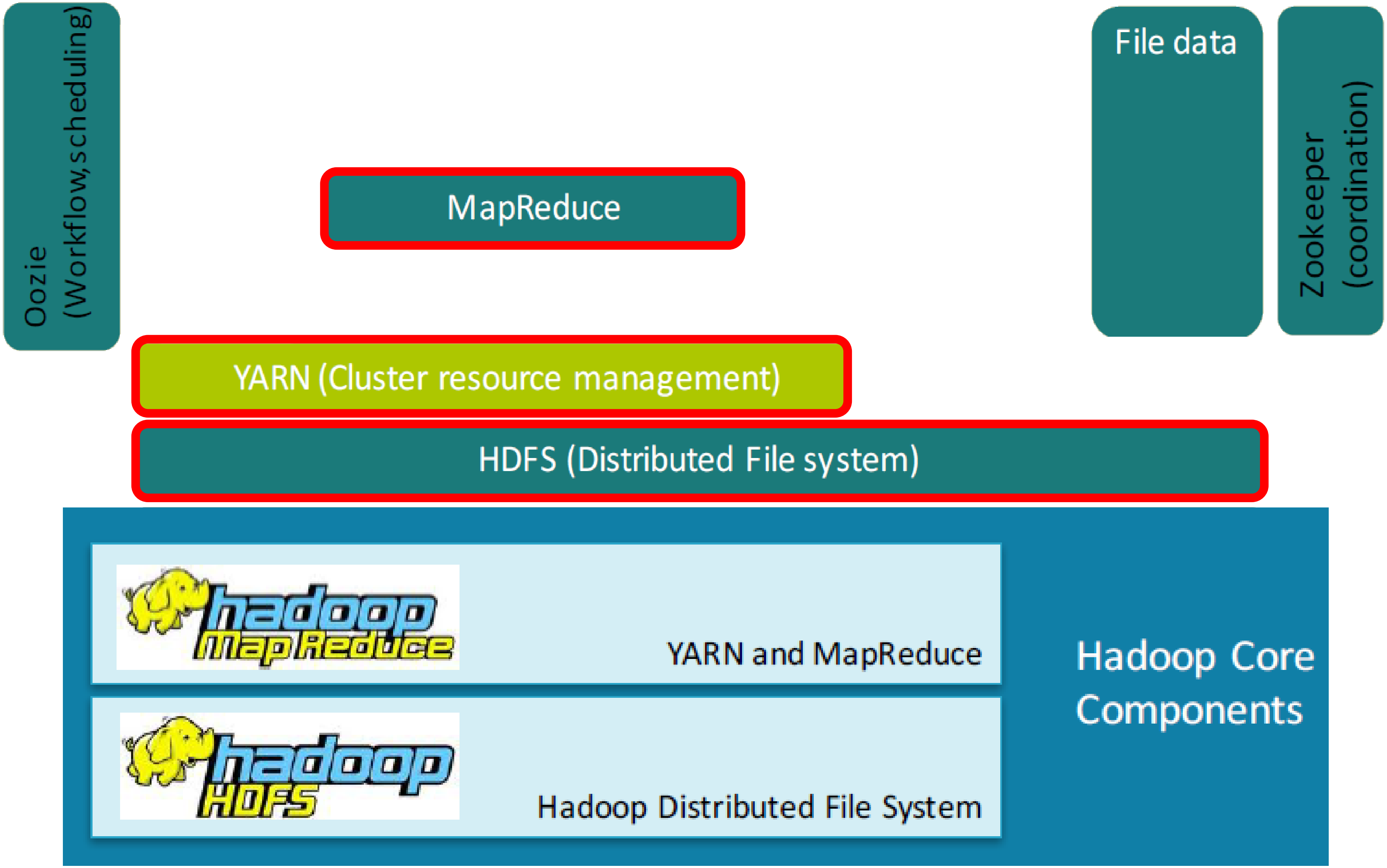
- Hadoop is **based on work done at Google** in the late 1990s/early 2000s

- **Google's problem:**
  - Indexing the entire web requires massive amounts of storage
  - A new approach was required to process such large amounts of data

- **Google's solution:**
  - **GFS, the Google File System**
    - **"The Google File System" @ACM SOSP 2003**
  - **Distributed MapReduce**
    - **"MapReduce: Simplified Data Processing on Large Clusters" @OSDI 2004**

- **Hadoop is based on these former papers, implemented as a similar, open-source solution**

The Hadoop Stack (2.0 and evolutions)



The Core of Hadoop

- **The Hadoop Distributed File System (HDFS)**
  - Any type of file can be stored in HDFS
  - Data is split into chunks and replicated as it is written
    - Provides resiliency and high availability
    - Handled automatically by Hadoop
- **YARN (Yet Another Resource Negotiator)**
  - Manages the processing resources of the Hadoop cluster
  - Schedules jobs
  - Runs processing frameworks
- **MapReduce**
  - A distributed processing framework

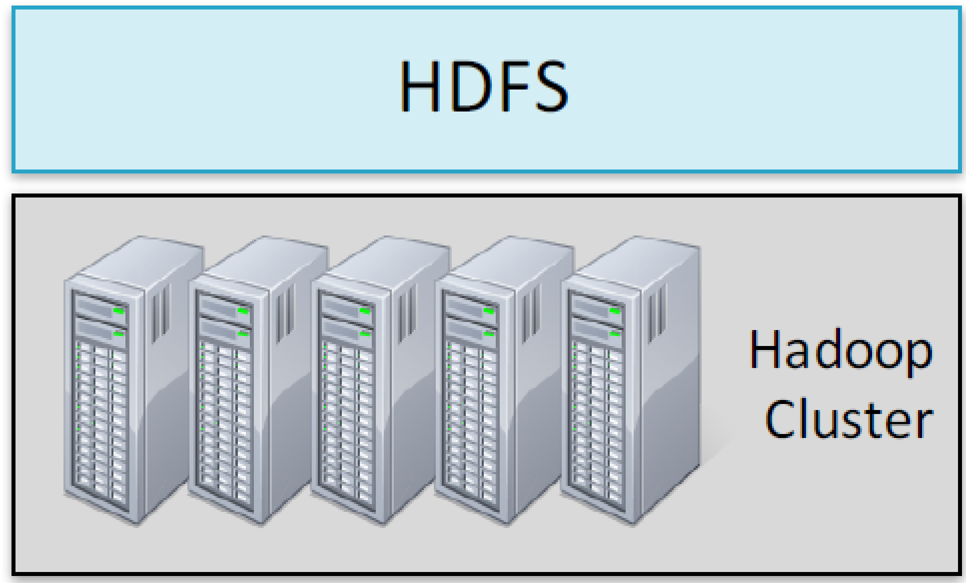
hadoop = hadoop HDFS (storage) + hadoop YARN (scheduling) + hadoop MapReduce (processing)

Hadoop HDFS (1/5)

- HDFS is the **storage layer** for Hadoop
- A file system which can store **any type of data**
- Provides **inexpensive and reliable storage for massive amounts of data**
  - **Data is replicated** across computers
- HDFS performs best with a **“modest” number of large files**
  - Millions, rather than billions, of files
  - Each file typically 100MB or more
- Files in HDFS are “write once”
  - Data is immutable
  - Appends are permitted

Hadoop HDFS (2/5)

- HDFS is a **filesystem** written in **Java**
- Sits on top of a native filesystem
- It's **scalable** (add more nodes to the HDFS)
- **Fault tolerant** (data replication)
- And **supports** efficient data processing with **MapReduce**, **Spark**, and **other frameworks**



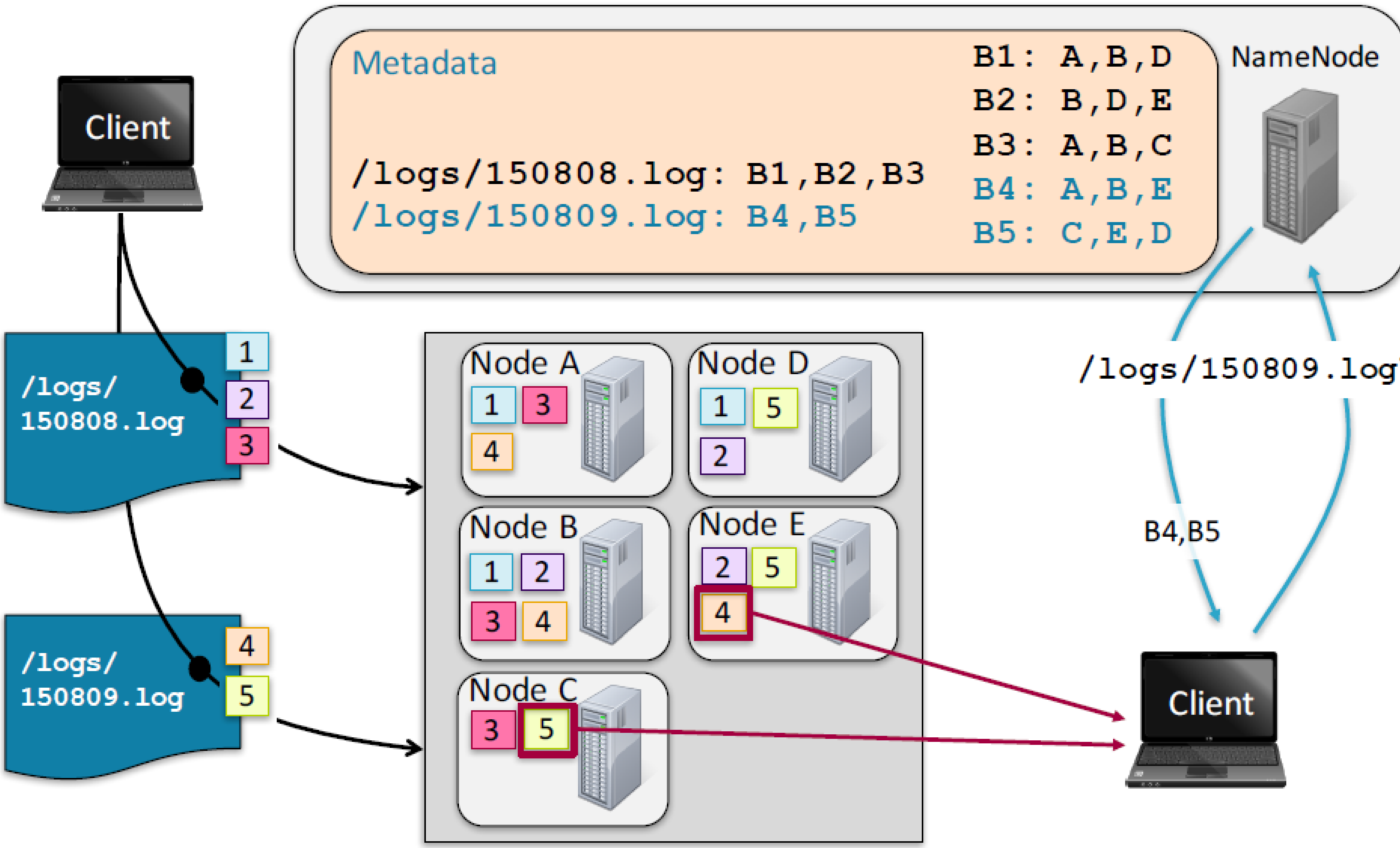
Hadoop HDFS (3/5)

- Data files are split into blocks and distributed to data nodes
- Each block is replicated on multiple nodes (default: 3x replication)
- NameNode stores metadata, useful for locating blocks

Hadoop HDFS (4/5)
Getting data in & out of HDFS

- **Hadoop**
  - Copies data between client (local) and HDFS (cluster)
  - API or command line
- **Ecosystem Projects**
  - Flume
    - Collects data from stream sources
  - Sqoop
    - Transfers data between HDFS and RDBMSs
- **Data Analytics Tools**

Hadoop HDFS (5/5)
Storing & retrieving files



Hadoop Yarn

- **Originally (v1.0), Hadoop only supported MapReduce as processing framework**
  - MapReduce used all of the cluster's processing resources
- **Using YARN, a single cluster may run multiple processing frameworks, such as MapReduce and Spark**
  - Each framework competes for the nodes' resources
- **YARN helps to manage this contention**
- **It allocates resources to different frameworks based on demand, and on system administrator settings**

Hadoop MapReduce

- **MapReduce is a programming model**
  - Facilitates task distribution across multiple nodes
  - Record-oriented data processing (key and value)
  - Requires the programmer to write operations as a sequence of map and reduce operations (additional burden, linear dataflow)
  - MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk.
- **Ma… …ble on**
  - … …ny

MapReduce example counting the appearance of each word in a set of documents:

```
function map(String name, String document):
        // name: document name
        // document: document contents
        for each word w in document:
        emit (w, 1)

function reduce(String word, Iterator partialCounts):
        // word: a word
        // partialCounts: a list of aggregated partial counts
        sum = 0
        for each pc in partialCounts:
        sum += pc
        emit (word, sum)
```

The Hadoop Ecosystem: some basics

- Data **Integration**: Flume, Kafka, and Sqoop
- Data **Processing**: Spark
- Data **Analysis**: Hive and Impala
- User **Interface**: Hue
- Data **Storage**: HBase
- Data **Security**: Sentry
- **Deep Learning in Spark**

Flume and Kafka (1/2)

- **Flume and Kafka are tools for ingesting event data into Hadoop as that data is being generated**
  - Network traffic
  - Log files
  - Streaming data

- **Flume is typically easier to configure, but Kafka provides more functionality**
  - Flume generally provides a path from a data source to HDFS or to a streaming framework such as Spark
  - Kafka uses a "Publish/Subscribe" model, allowing data to be consumed by many different systems, including writing to HDFS

Flume and Kafka (2/2)

- **They are ideal for aggregating event data from many sources into a centralized location (HDFS)**

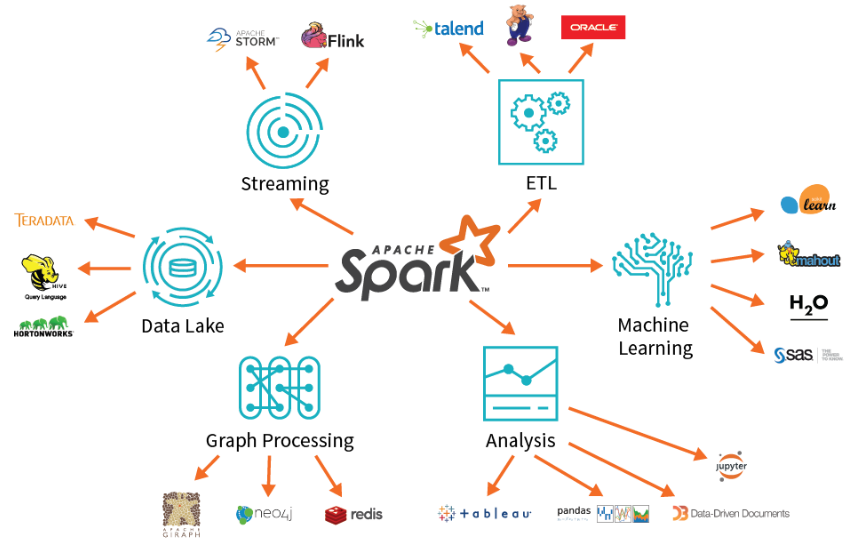
- **Well-suited for event driven data**
  - Network traffic
  - Social-media-generated
  - Digital sensors
  - Log files

- **Allow you to process streaming data as that data is being generated**
  - Anomaly detection, network security, etc.

Sqoop

- **Sqoop moves large amounts of data between relational database management systems (RDBMSs) and HDFS**
  - Import tables (or partial tables) from an RDBMS into HDFS
  - Export data from HDFS to a database table
- **Uses JDBC to connect to the database**
  - Works with virtually all standard RDBMSs
- **Custom “connectors” for some RDBMSs provide much higher throughput**
  - Available for certain databases, such as Teradata and Oracle

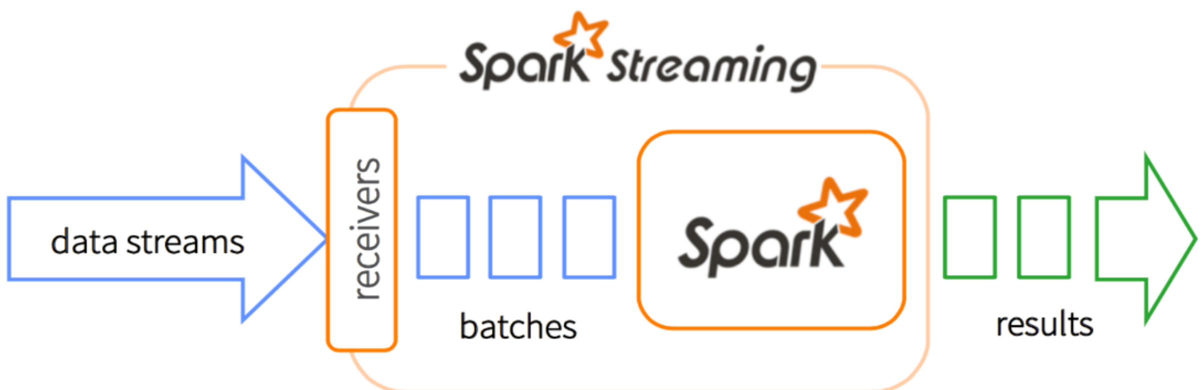
Spark (1/3)



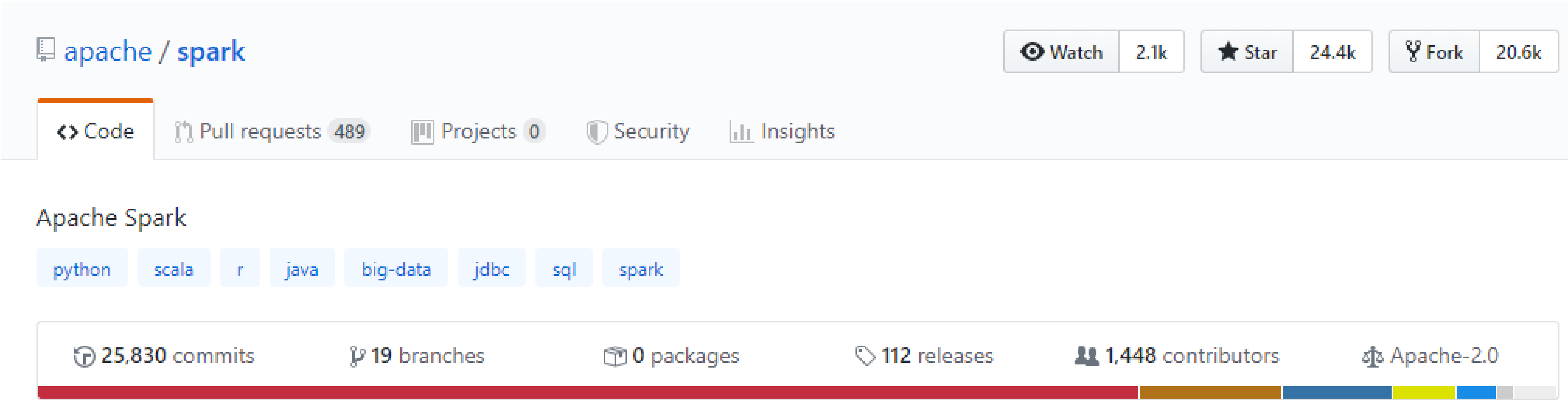
- **Apache Spark is a large-scale data processing engine**
- **The default system for large-scale distributed data analysis**
- **Supports a wide range of workloads and interfaces with most commercial systems**
  - Machine learning (Mllib, Mahout, spark.ml, databricks)
  - Batch applications
  - Iterative algorithms
- **Spark is well-suited to iterative processing algorithms**
- **Runs in memory → It is WAY FASTER than MapReduce**

Spark (2/3)

- **Spark code can be written in Python, Scala, or Java**
  - Easier to develop for than MapReduce
  - If you're new to Hadoop, better to start with Spark and never write MapReduce code
- **Spark Streaming provides real-time data processing features, processing data as that data is being generated**
  - Typically in conjunction with Flume or Kafka



Spark (3/3)



- **Very active open project under Apache**

- **Databricks as the leading industry-framework pushing Spark into the AI/ML domain → Deep Learning Pipelines for Apache Spark**

Hive and Impala (1/2)

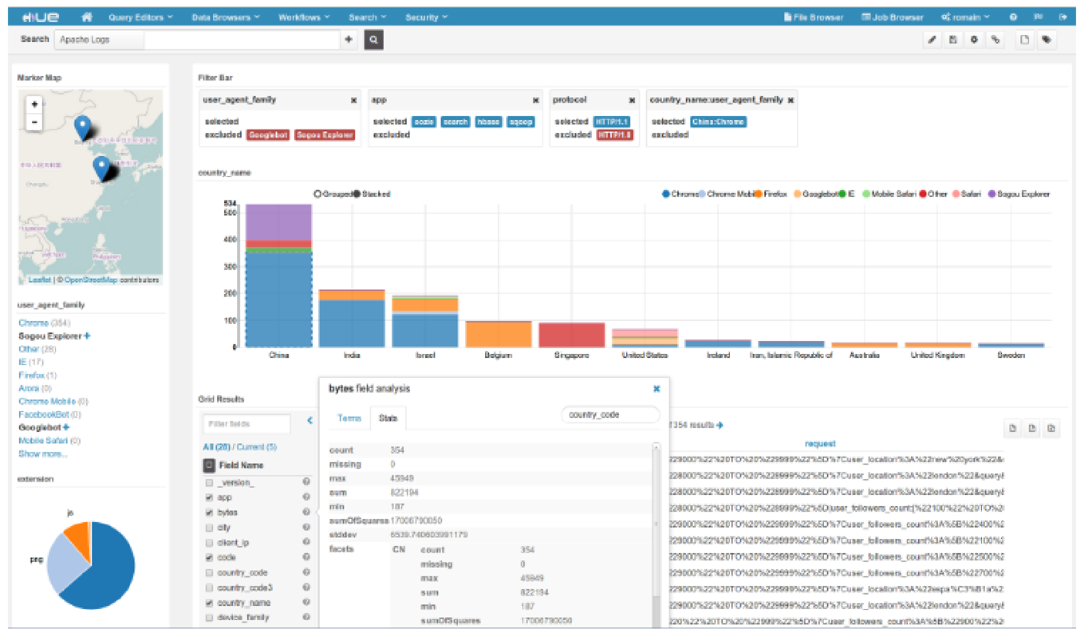
- **SQL engines on top of a Hadoop cluster**
- **Hive is an abstraction layer on top of Hadoop**
  - uses a SQL-like language called HiveQL
  - No need to program in Java, Python, Scala, etc.
- **The Hive interpreter uses MapReduce or Spark to actually process the data**
- **Well suited for structured data**

Hive and Impala (2/2)

- **Apache Impala is a high-performance SQL engine**
  - Runs on Hadoop clusters
  - Does not rely on MapReduce
  - Very low latency—typically measured in milliseconds
- **Impala supports a dialect of SQL very similar to Hive's**
- **Impala is much faster than Hive**
- **Deals with multiple simultaneous queries much better**

HUE

- **Hue provides a Web front-end to a Hadoop cluster**
  - Upload data
  - Browse data
  - Query tables in Impala and Hive
  - Search (using Apache Solr)
  - Etc…
- **Provides access control for the cluster**
- **Makes Hadoop easier to use** (but not particularly attractive for production)



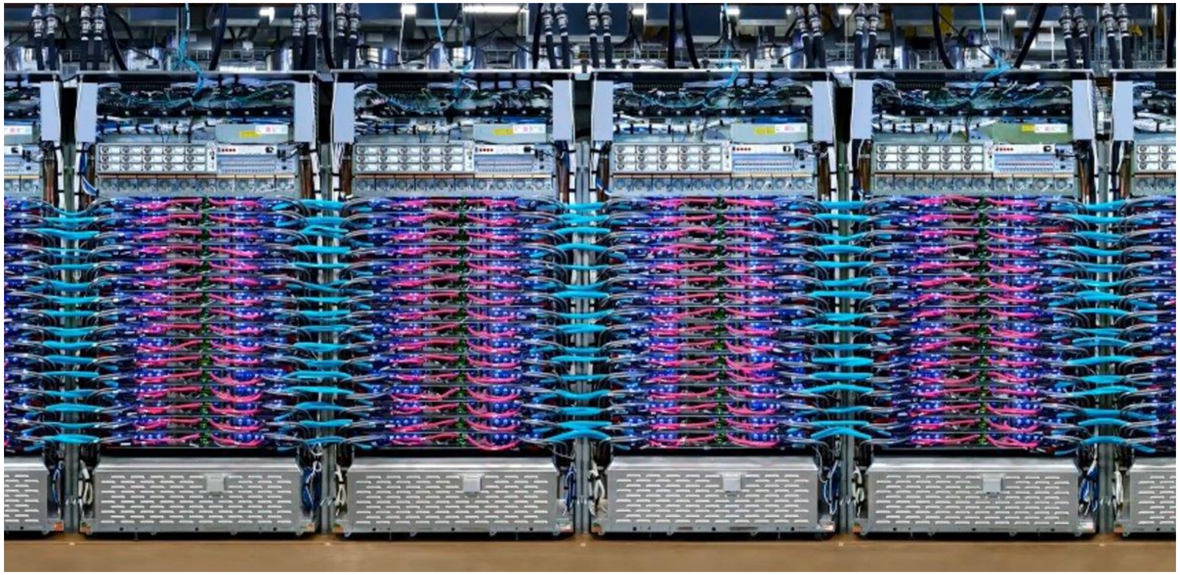
HBase

- **HBase is a NoSQL distributed database**
- **Stores data in HDFS**
- **Scales to support very high throughput for both reads and writes**
  - Millions of inserts or updates per second
- **A table can have many thousands of columns**
  - Handles sparse data well
- **Designed to store very large amounts of data (Petabytes+)**
- ***Avoid killing mosquitos with a hammer***

Sentry

- **Sentry provides fine-grained access control (authorization) to various Hadoop ecosystem components**
  - Impala
  - Hive
  - Cloudera Search
  - The HDFS command-line shell
  - E.g., permission to view only certain columns in a given Hive table
- **Might be worth checking if installing a cluster** (Hadoop has long supported Kerberos for authentication, but can be tricky to set up)

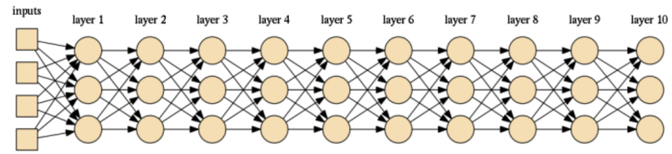
Computational Hardware



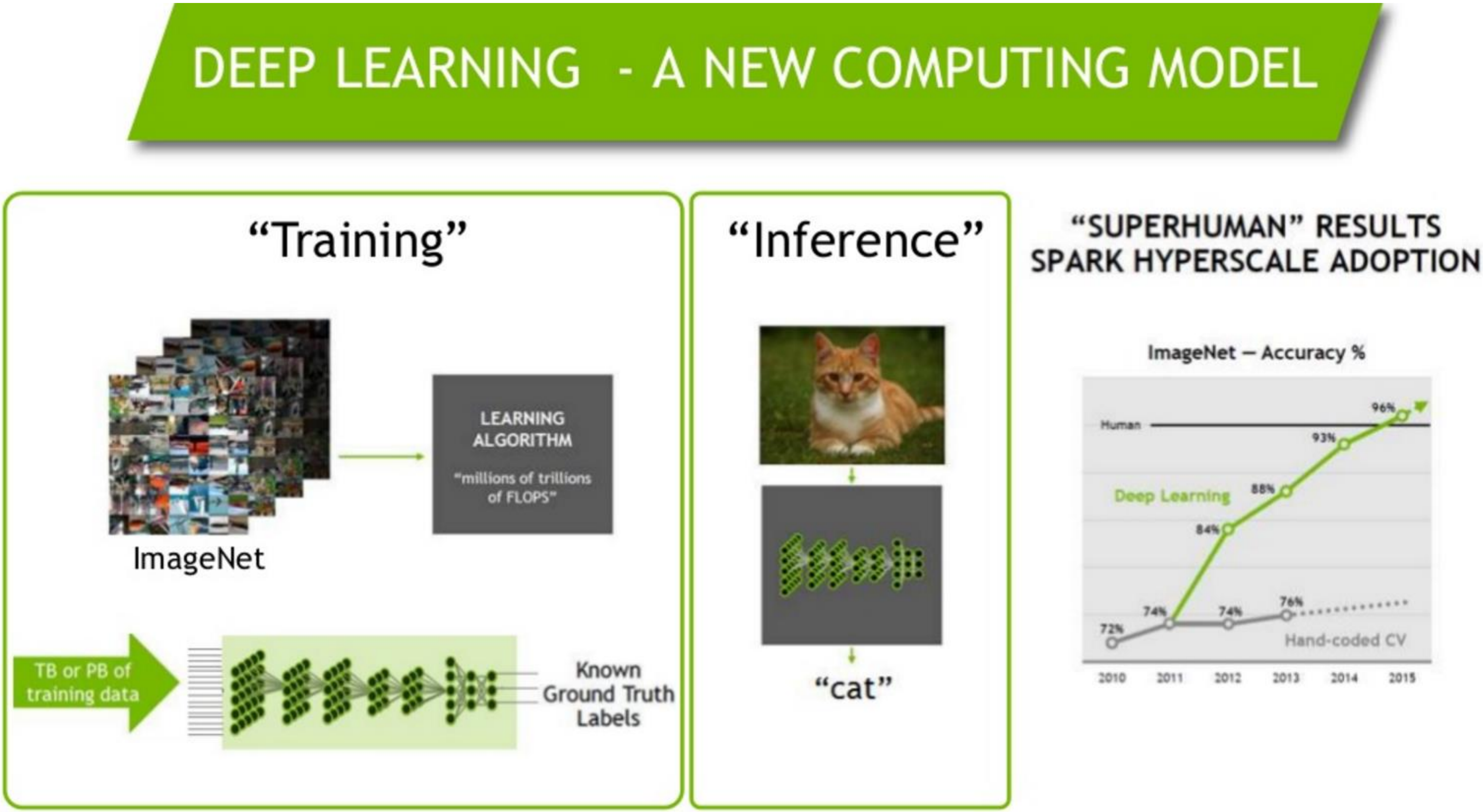
- **CPU** – serial, general purpose, everyone has one
- **GPU** – parallelizable, still general purpose
- **TPU** – custom ASIC (Application-Specific Integrated Circuit) by Google, specialized for machine learning, low precision
- **Ascend** – custom ASIC by Huawei, also specialized for ML

- **Ascend 910** outperforms both GPU Tesla V100 and TPU 2.0 by **100%/50%**, with the **greatest computing density achieved so far**

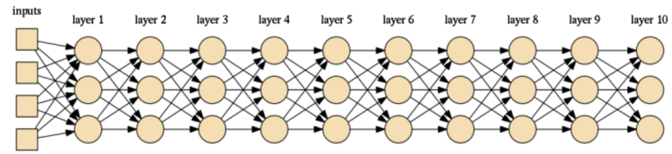
Deep Learning in Spark (1/2)



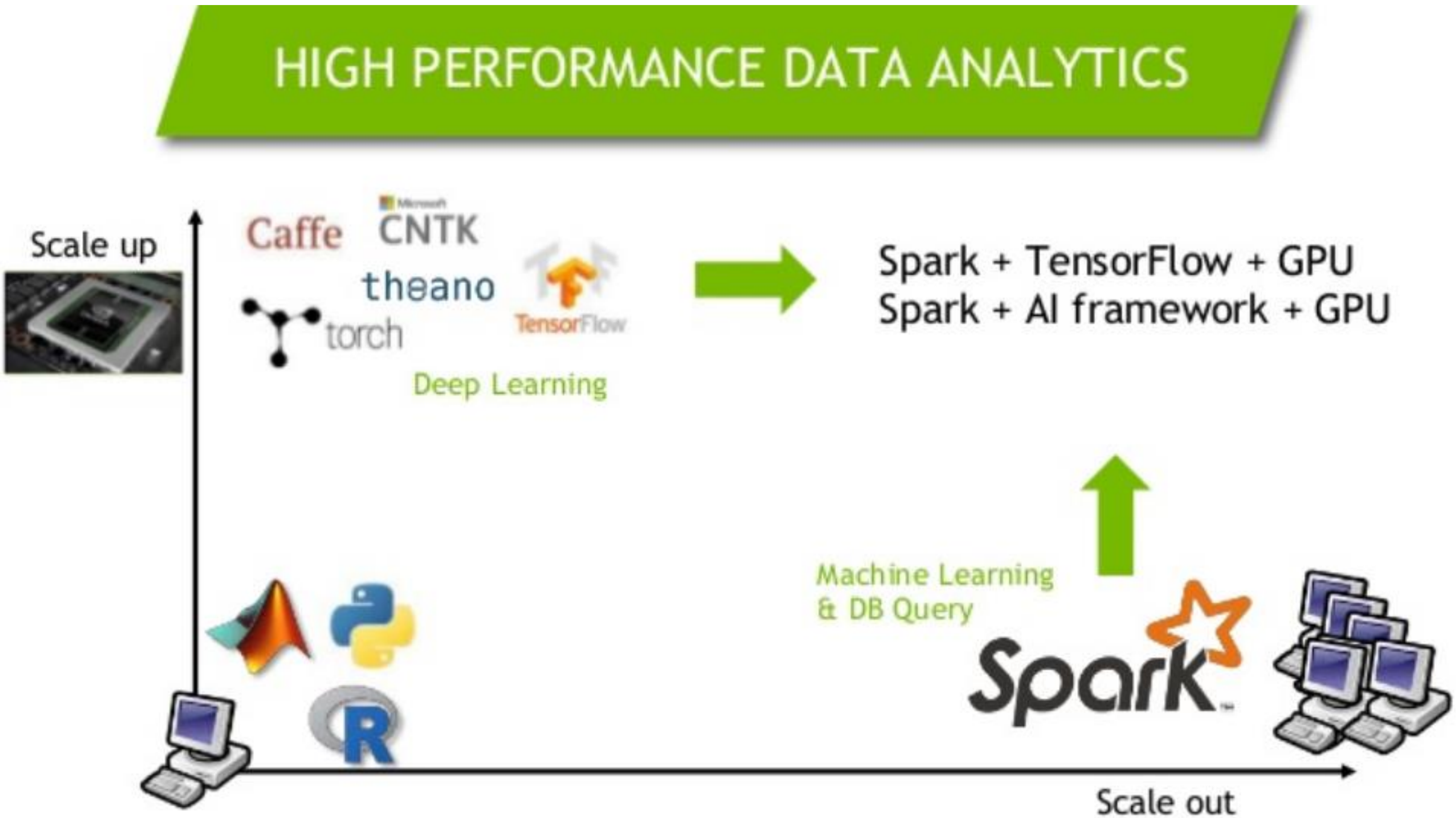
- Neural Networks re-loaded, high benefits out of big-data and distributed



Deep Learning in Spark (2/2)



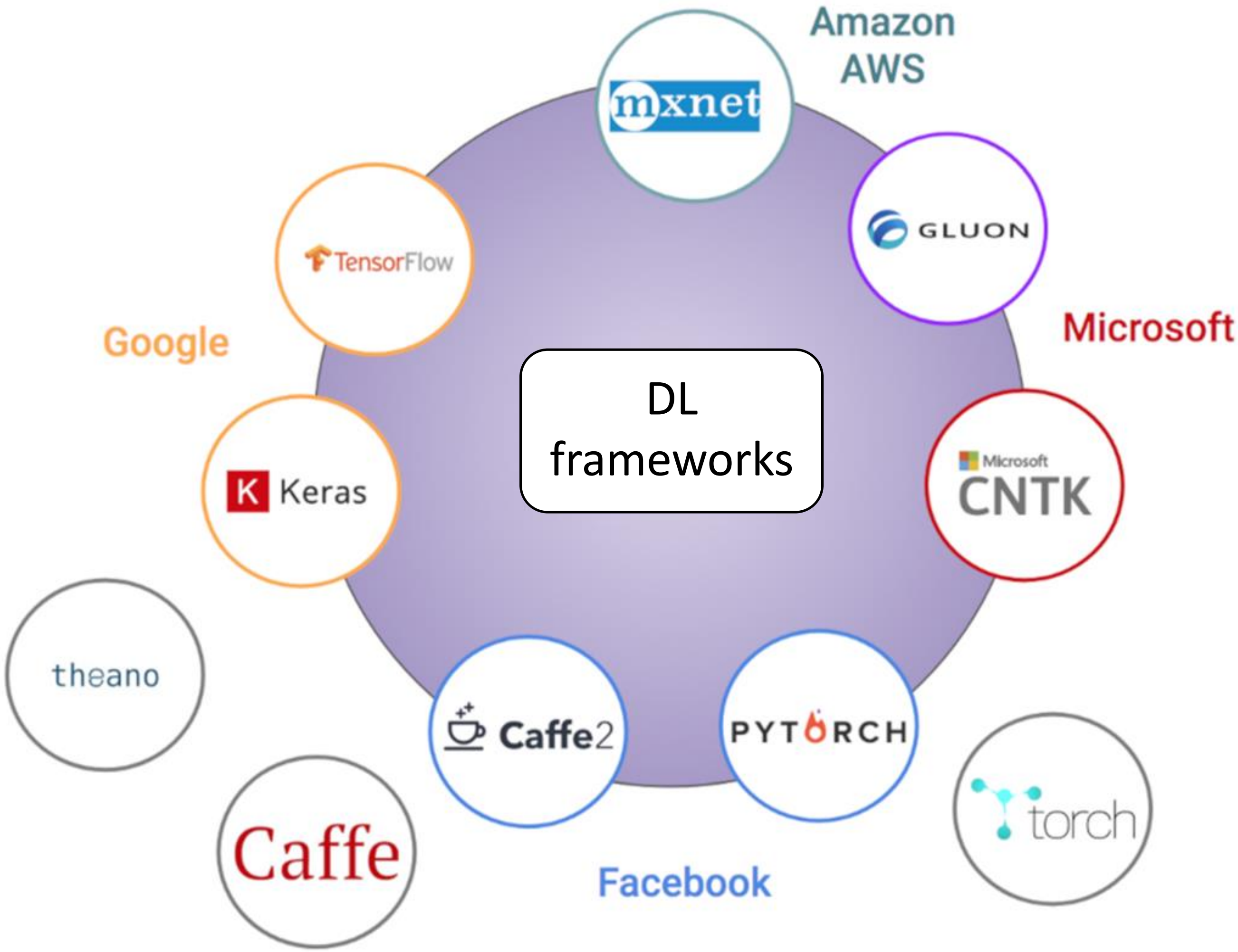
- **GPU-Driven High Performance Data Analytics in…….SPARK ☺** (by NVIDIA)



Big Data Frameworks for NTMA
and What About Deep Learning?

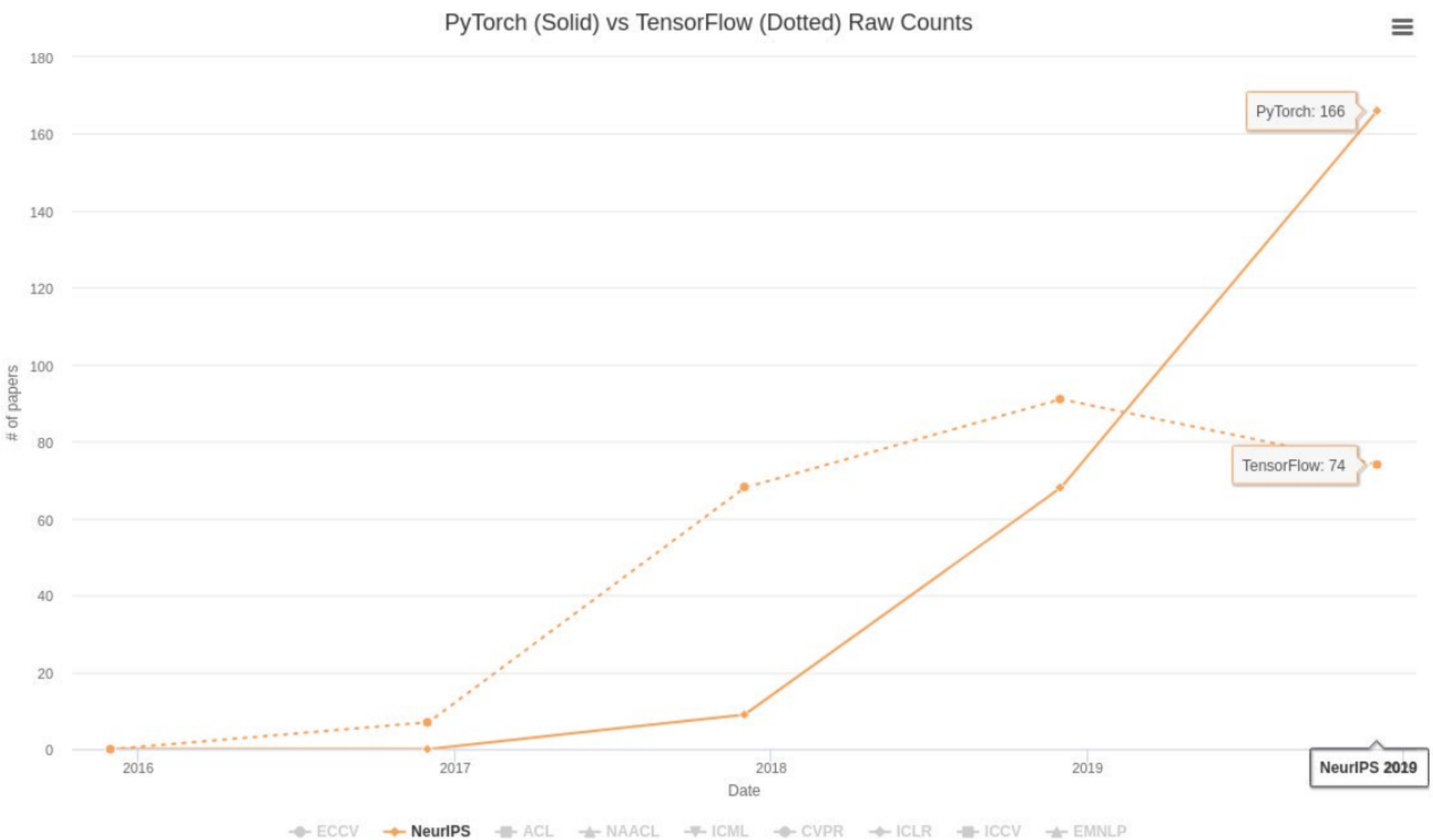
And What About Deep Learning?

- Lots of frameworks for Deep Learning



Most Popular Frameworks

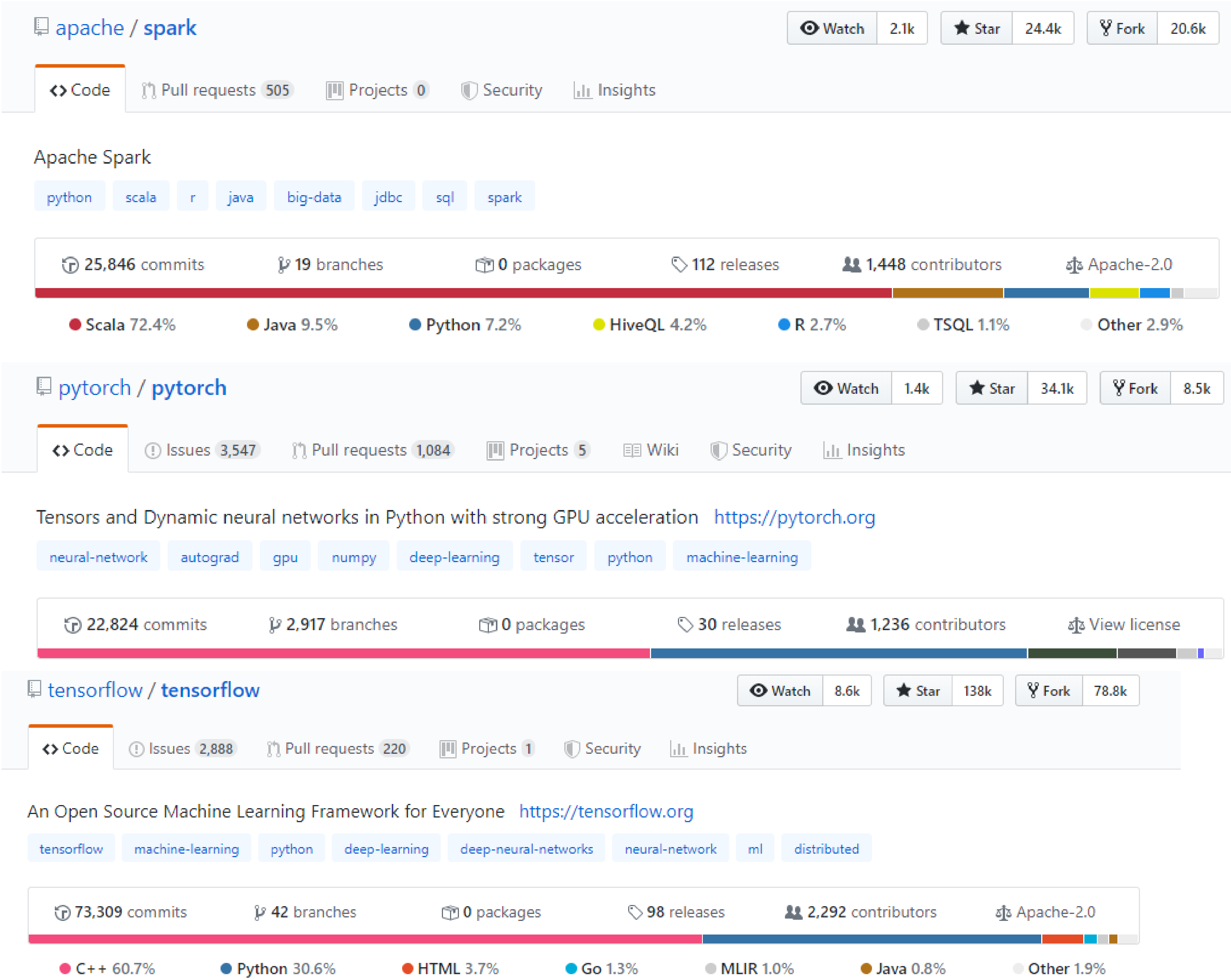
- Papers at top ML conferences mentioning a specific framework



TORCH

Keras

Most Contributed Frameworks



TensorFlow vs. Pytorch

- Using TensorFlow without Keras is quite complex
- Debugging in TensorFlow is complex
- **TensorFlow 2.0 integrates Keras directly**, as well as a **TF Eager API**, which improves debugging and adds **similar-to-pytorch** features
- TF is a very **powerful and mature deep learning library**
- It has **production-ready deployment options** and support for mobile platforms

- Pytorch is more **intuitive and direct**
- PyTorch, on the other hand, is still a **young framework** with **stronger community movement** and it's more Python friendly
- If you want to make things faster and **build AI-related products**, **TensorFlow** + Keras is a good choice
- **PyTorch is mostly recommended for research-oriented developers**

Thanks You for Your Attention!

Pedro Casas

pedro.casas@ait.ac.at