

# Data Parallel Frameworks

Ashvin Goel

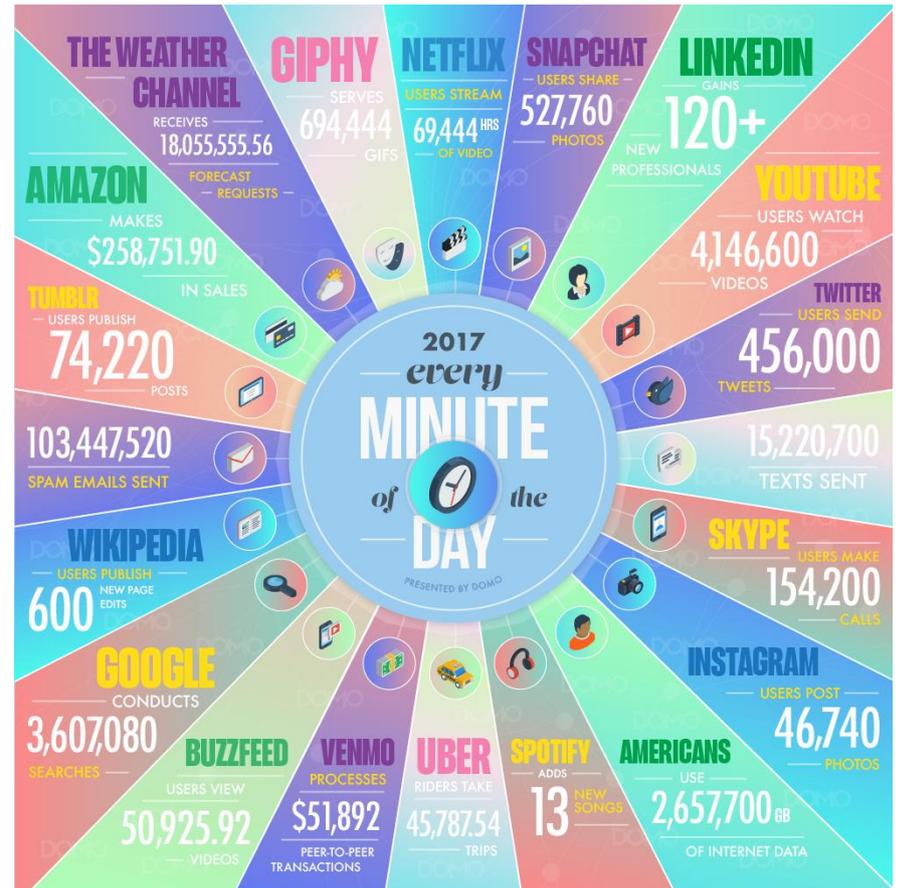
Electrical and Computer Engineering  
University of Toronto

ECE1724

Some of these slides are heavily modified slides from  
Prof. Ken Birman's course on Cloud Computing

# What are Web-Scale Apps?

- Applications that are hosted in massive-scale computing infrastructures such as data centers
- Used by millions of geographically distributed users
  - Via web browsers, mobile clients, etc.
- Produce, store, consume massive amounts of data
  - Scale is hard to comprehend



# What Kind of Data is Stored?

- Companies store data based on their business model ...
  - Google, e.g., daily snapshot of all the web pages in the world
  - Amazon, e.g., current product data & price for every product
  - Facebook, e.g., social networking graph
  - ...
- We have seen various types of storage systems for storing this data
  - Data is typically sharded across many machines
  - Sharded data is replicated for fault tolerance, fast read access
  - Much focus on scalability, data availability, consistency, etc.

# How is the Data Used? i.e, What is “Big Data”?

- Web search (e.g., Google) needs to analyze billions of web pages to determine the most relevant pages
- Product search (e.g., Amazon) needs to analyze millions of products, who bought them, their reviews, etc.
- Recommendation systems (e.g., LinkedIn, Facebooks) need to analyze massive social graphs
- All the above can be used to generate revenue streams, e.g., smart ad placement, recommendations
  - These are restaurants you might like based on your tastes ...
  - These stores have huge Christmas sales for things you like ...

# A Simple Example: Web Search

- Data collection and storage
  - Collect web pages, store them
- Data analytics
  - Grep, sort, word count, e.g., extract words (or phrases) from web pages
  - Index pages, e.g., associate each word with a ranked list of web pages that contain these words
  - Log analysis
- Data serving
  - When user searches for word, serve associated list of pages

# Data Analytics Requirements

- Data analytics
  - Extract words (or phrases) from web pages
  - Associate each word with a ranked list of web pages that contain these words
- Massive computation needs
  - Parse all pages
  - Rank all pages, similar to sorting a very large data set
    - E.g., find the “most authoritative pages” by organizing web pages in a graph, then finding the graph nodes with highest weight (rank)

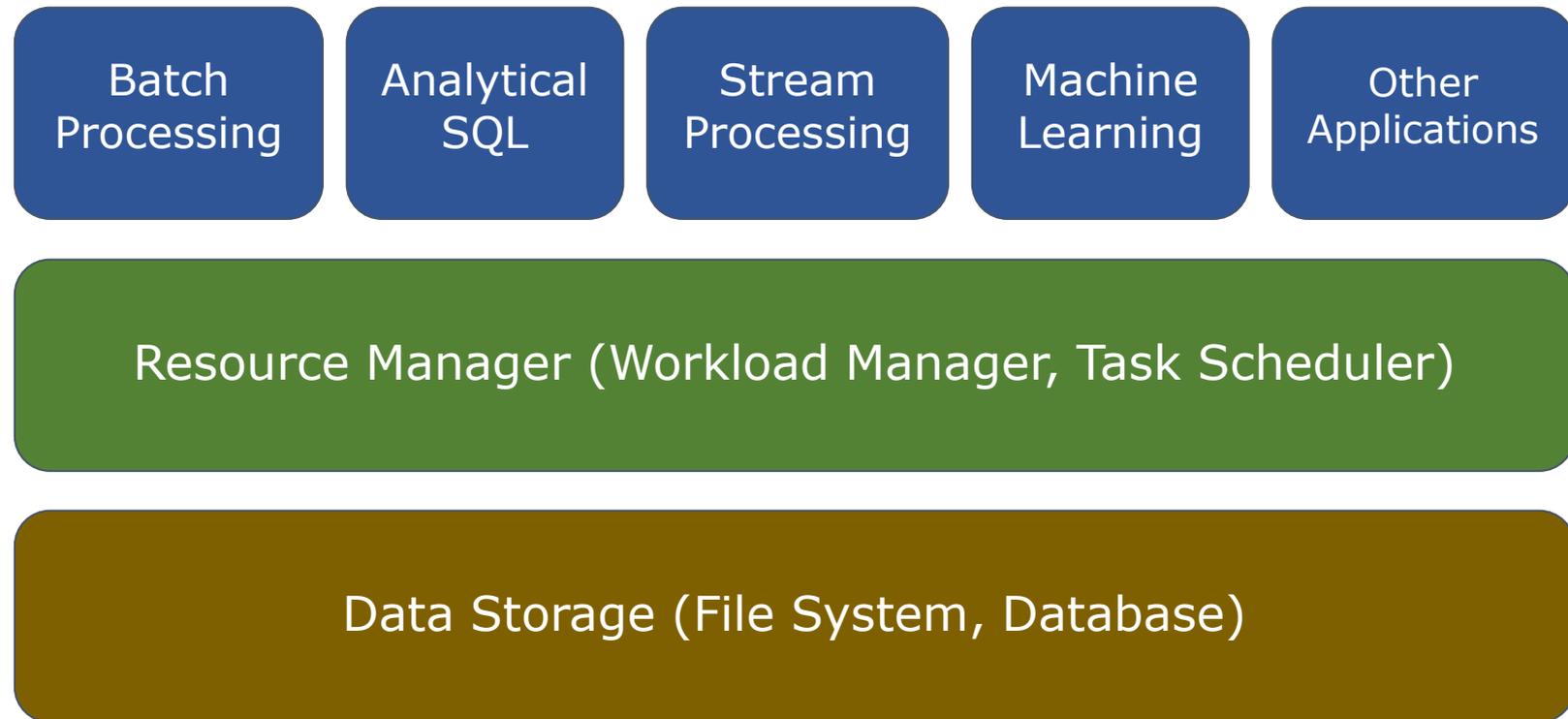
# Data Analytics Challenges

- Data is massive
  - Need sharding across large numbers of machines
  - Need storage on disk
  - Need to handle storage failures
  - Need to handle updates (later)
- Computation is massive
  - Need scalable, parallel computation models
  - Need to handle massive intermediate, final results
  - Need to handle compute failures

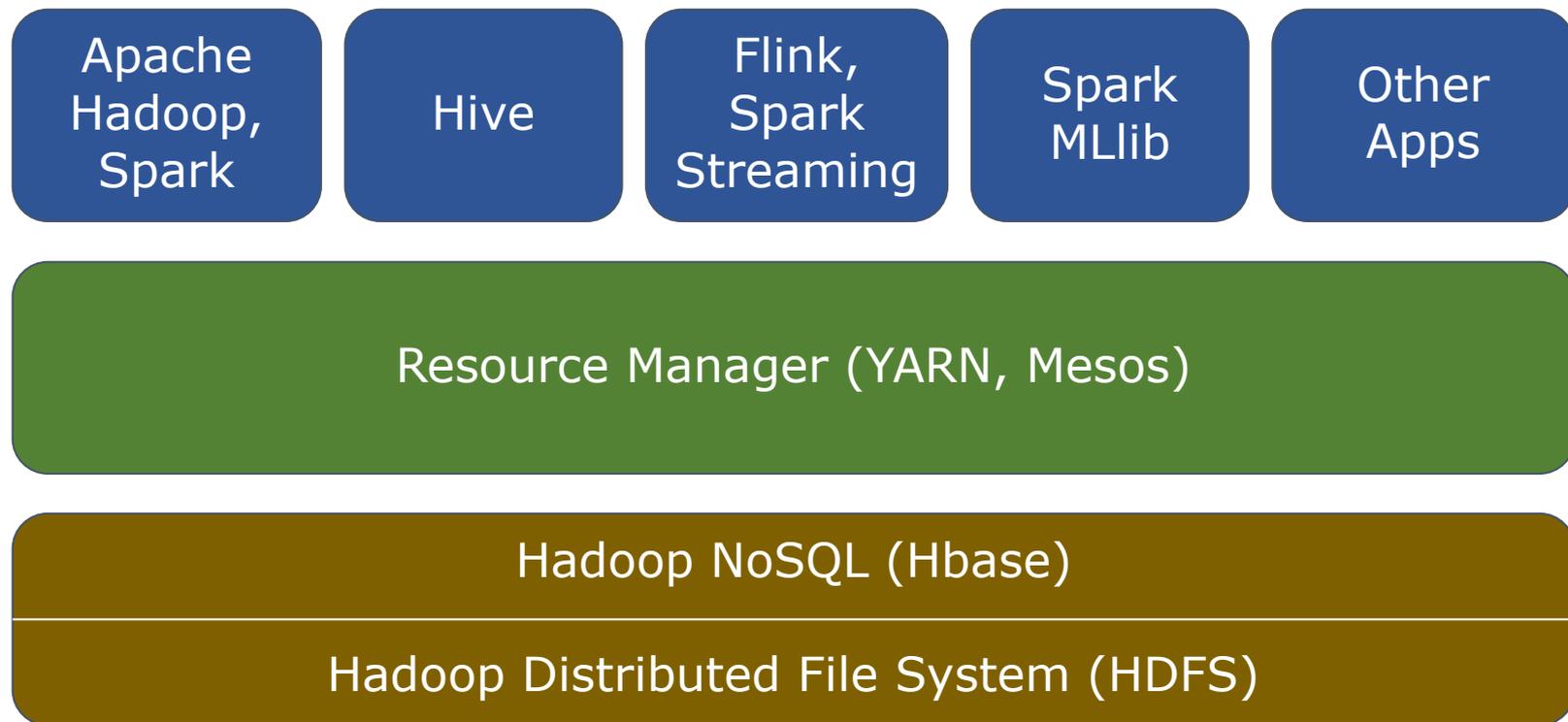
# Data Analytics Frameworks

- These frameworks perform massively parallel (“always sharded”) computing efficiently
- The data starts out sharded
- Often the intermediary states and results are sharded
- Results are typically human-useful output, e.g., charts

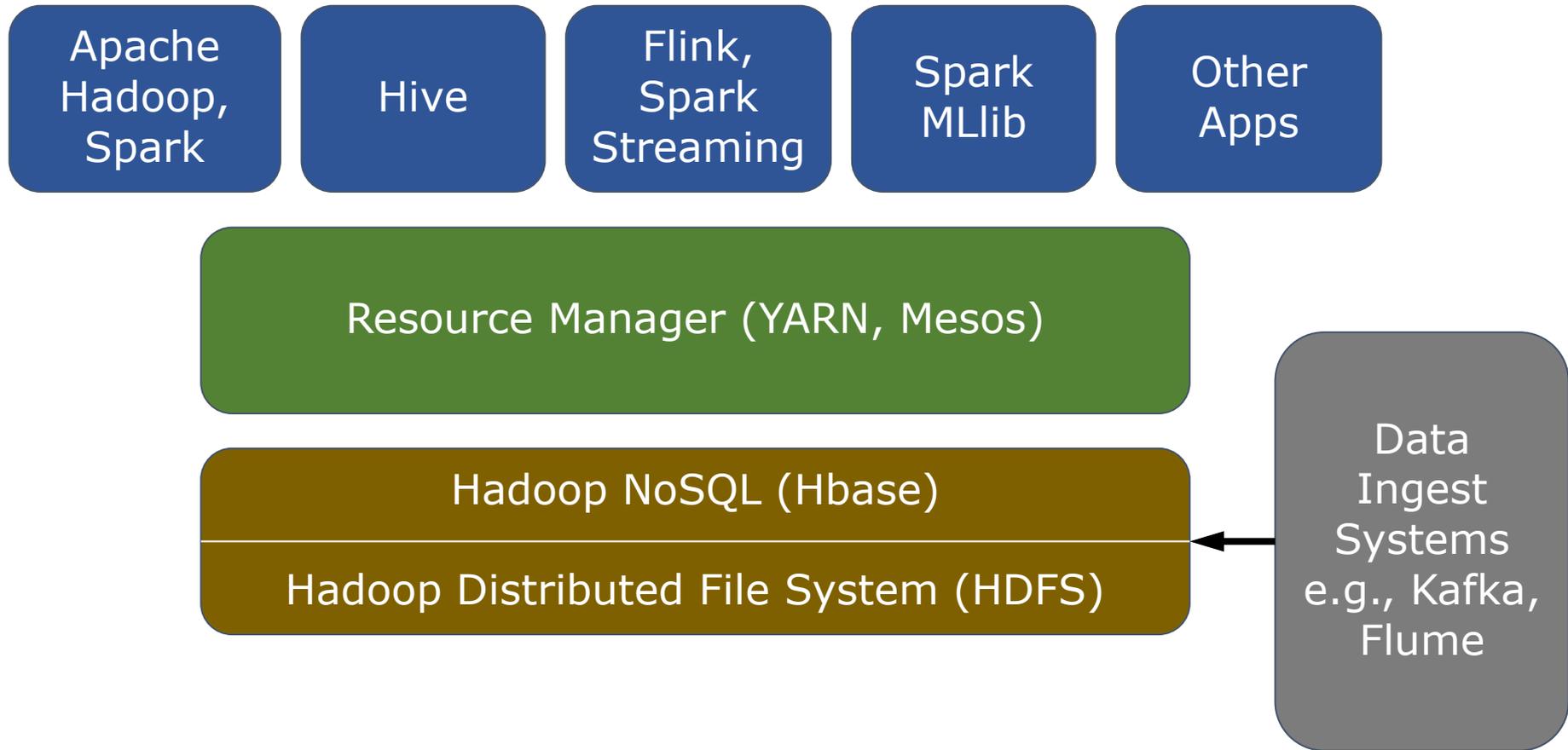
# A Typical Big Data System



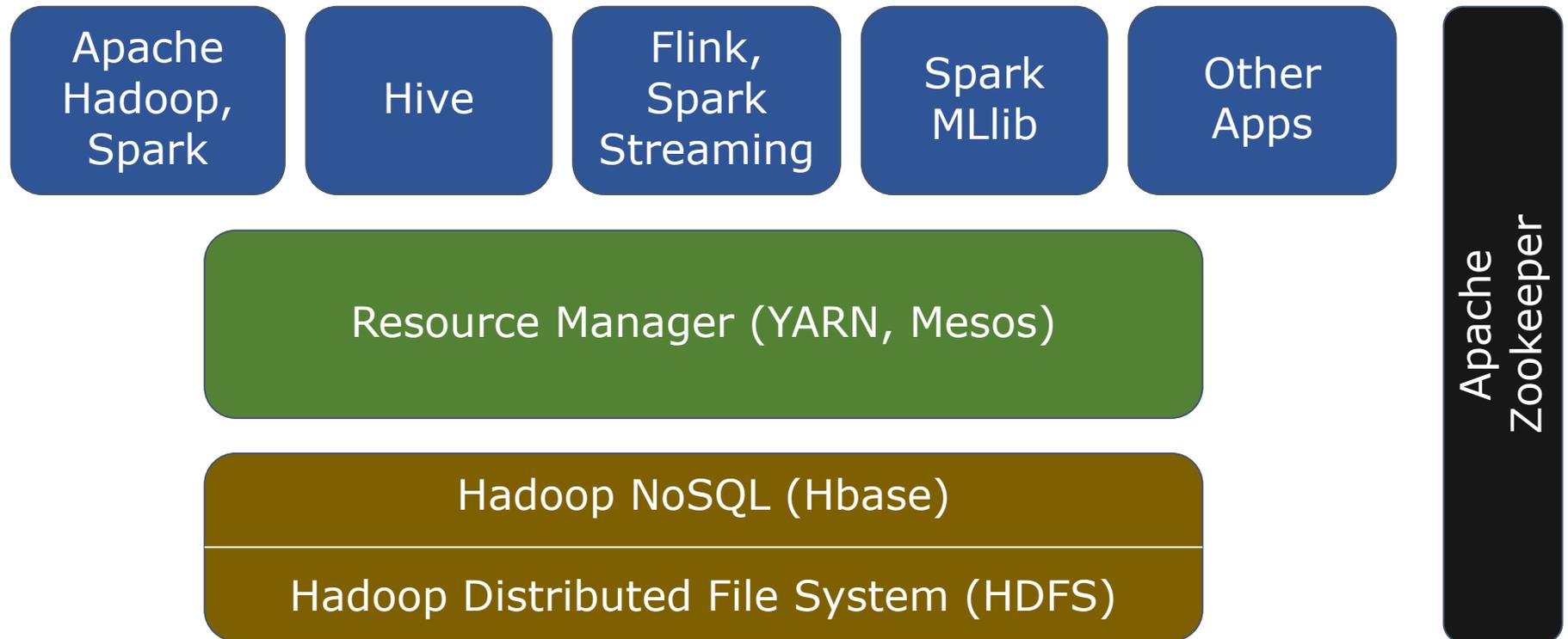
# Open-Source Apache Ecosystem



# Data Ingestion



# Coordination

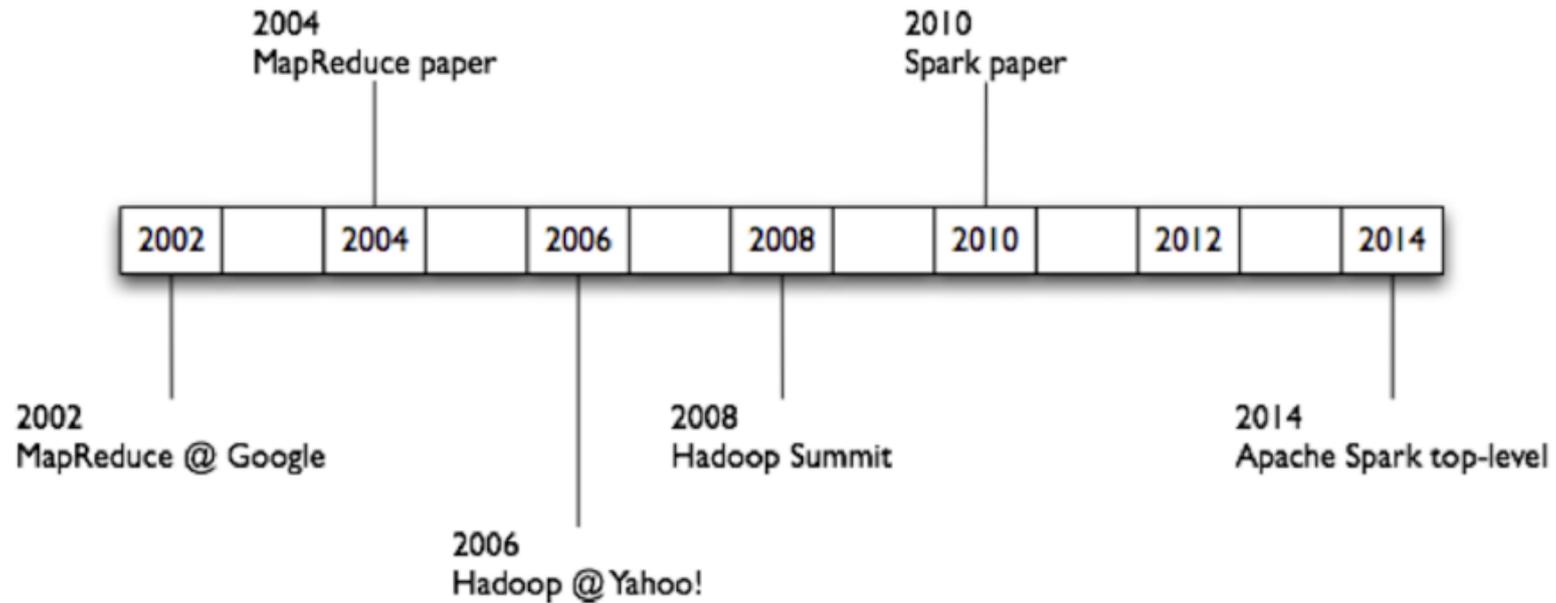


# Batch Processing Frameworks

- Focus on simplifying the complexity of distributed programming
  - Developer focuses on logic for processing data
  - Framework takes care of **parallelization**, **fault tolerance**, scheduling, caching, ...
- Hadoop (MapReduce)
  - Suited for individual batch (long running) jobs
- Spark
  - Also suited for iterative and interactive batch jobs

# History of Hadoop and Spark

---



# Map Reduce

- MapReduce enables distributing (parallelizing) a job across multiple nodes of a cluster
- Allows programmers to describe processing in terms of simple **map** and **reduce** functions on items
- Framework takes care of scaling, scheduling, hardware and software failures

# Spark

- Same goal as Map Reduce, i.e., enable distributing (parallelizing) a job across multiple nodes of a cluster
- Allows programmers to describe processing in terms of **transformations** on fault-tolerant, distributed datasets
  - Datasets are nodes, transformations are edges in a graph
  - Transformations are evaluated only when needed (lazily)
- Framework takes care of **caching, data locality**, scaling, scheduling, hardware and software failures
- **Key idea is to cache intermediate data that is reused**

# Challenges

- Parallelization
- Fault tolerance

# Parallelization

- Key intuition
  - Often same processing is required for all items
  - Processing is independent for each item
- E.g., update count of the # of accesses to each website
  - Same operation is performed for each website
- Operation (e.g., map) can be performed in parallel
  - Like a SIMD instruction
  - However, operation works on shards on different machines
  - Produces intermediate data that is also sharded across machines

# Why Batching?

- Shards are typically large, e.g., 16-64MB
  - Recall GFS chunk size
- Batch processing, i.e., processing all the data in a shard amortizes processing costs
  - Cost of processing each item is typically low, e.g., `count++`
  - Cost of accessing each item from storage is high
  - Batching reduces the latter cost
- However, batching requires enough data (or updates) to be available, so trades **latency** for **efficiency**

# Fault Tolerance

- Why is it vital for large computations?
- Aim is to hide failures from applications
  - Provide behavior equivalent to fail-free operation
  - You will hear terms like exactly-once operation
- Both Map Reduce and Spark provide strong consistency and fault tolerance guarantees
  - Their behavior is equivalent to running a sequential computation, even in the presence of failures
  - Next, we will discuss these ideas in detail