Big Data

Session 3: Column Stores and Coordination

Last week

- Distributed Infrastructures
- Google File System
- HDFS
- Apache Spark RDDs

Intended Learning Outcomes

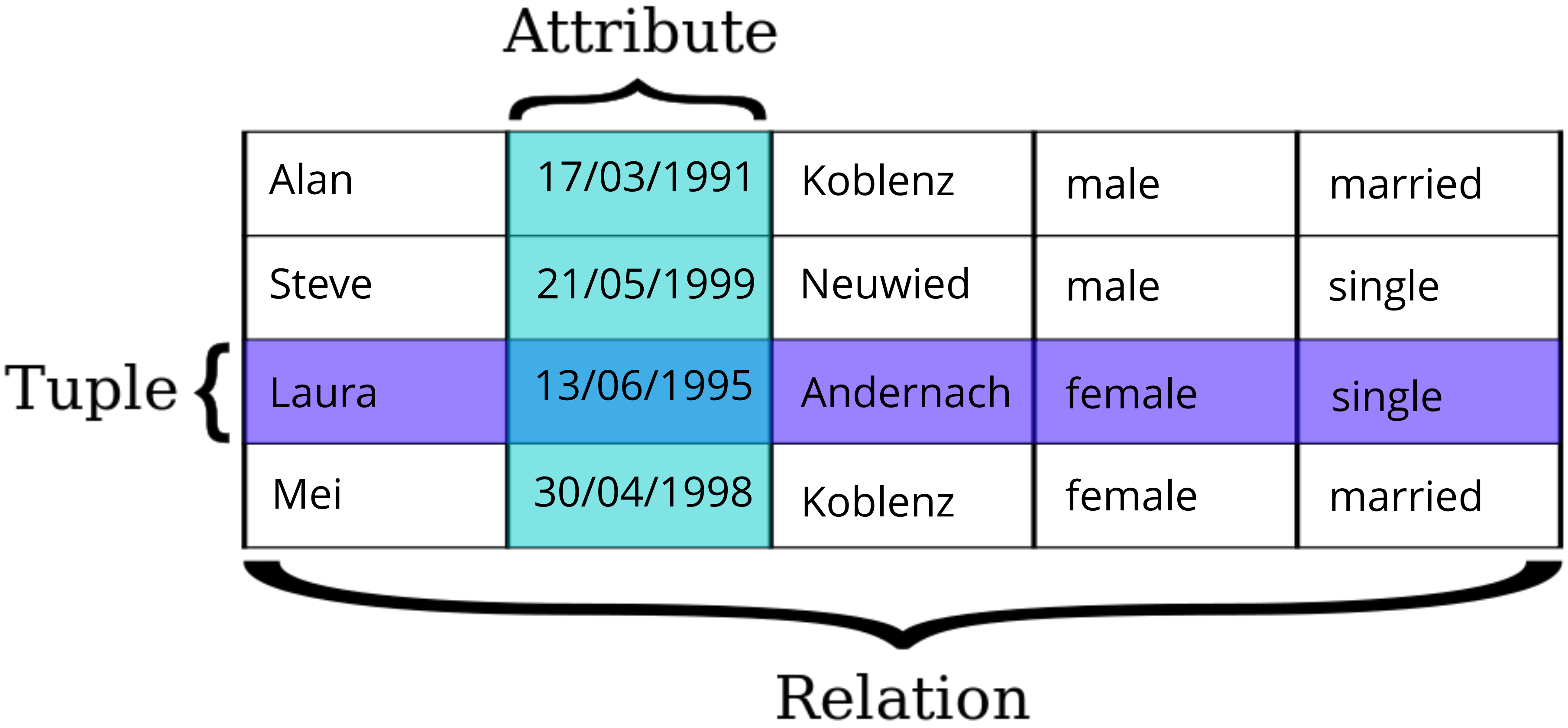
At the end of this lecture, you will be able to:

- Explain the benefits of column stores
- Describe approaches to schedule jobs in distributed infrastructures
- Outline how to manage distributed infrastructures

Outline

- Part 1:
  - Column stores (Introduction)
  - Big Table
  - HBase and Hive
- Part 2:
  - Job Scheduling
  - Coordination

Relational Database



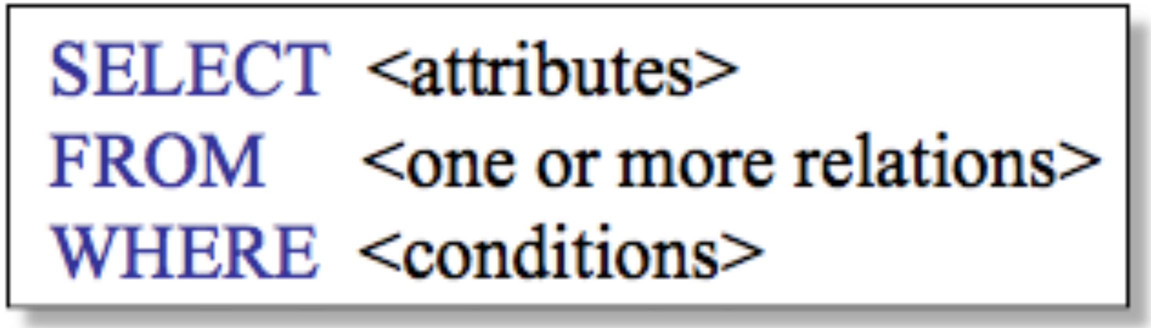
Alan	17/03/1991	Koblenz	male	married
Steve	21/05/1999	Neuwied	male	single
Laura	13/06/1995	Andernach	female	single
Mei	30/04/1998	Koblenz	female	married

Source (Image): https://en.wikipedia.org/wiki/Relational_database#/media/File:Relational_database_terms.svg

Structured Query Language: SQL

- Declarative query language

- Multiple aspects of the language
  - Data definition language
    - Statements to create, modify tables and views
  - Data manipulation language
    - Statements to issue queries, insert, delete data



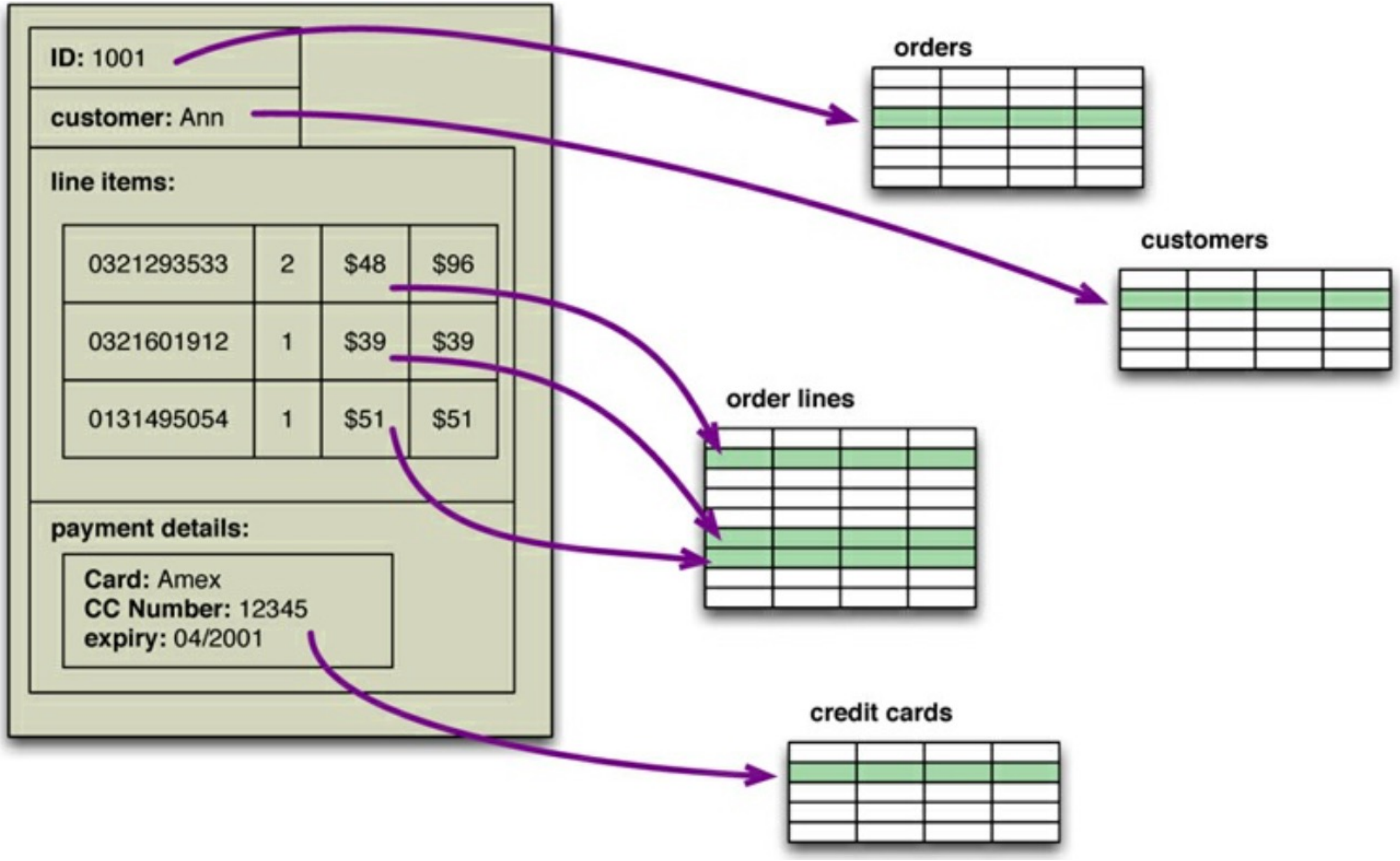
```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

Example
Building an e-commerce website

- Sell items directly to customers over the web

- Store information:
  - About users
  - Product catalog
  - Orders
  - Shipping addresses
  - Billing addresses
  - Payment data

Relational model



ID: 1001

customer: Ann

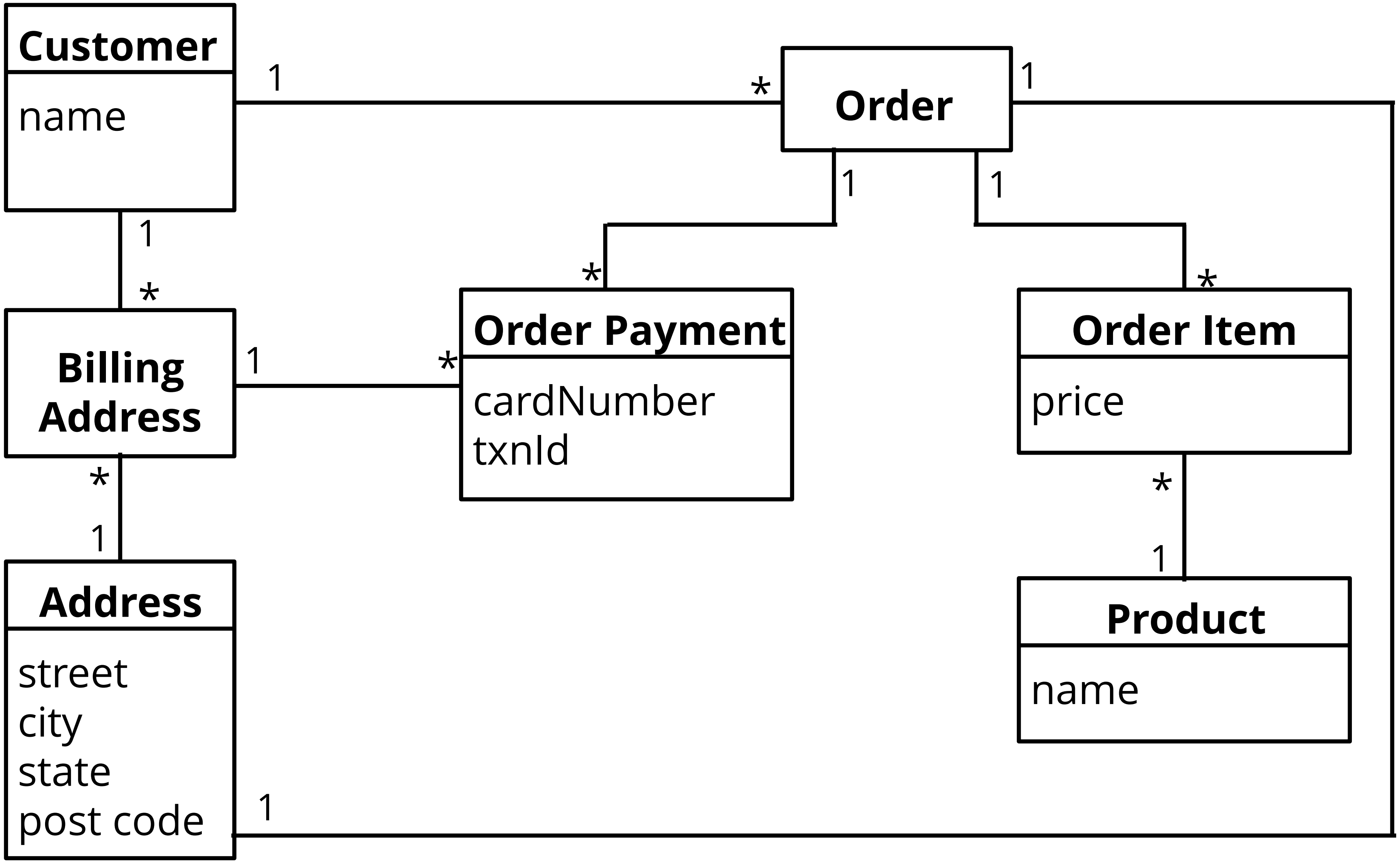
line items:

0321293533	2	$48	$96
0321601912	1	$39	$39
0131495054	1	$51	$51

payment details:

Card: Amex
CC Number: 12345
expiry: 04/2001

Relational Data Model



Typical data using RDBMS data model

Customer

Id	Name
1	Frank

Order

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

Address

Id	City
77	Chicago

OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

Example: How to query these?

- What is the price of all orders in 2022?
- How many orders were payed by credit card in 2022?
- In how many orders did we sell products for less than 10 Euro?
- Which customers have more than one billing address?

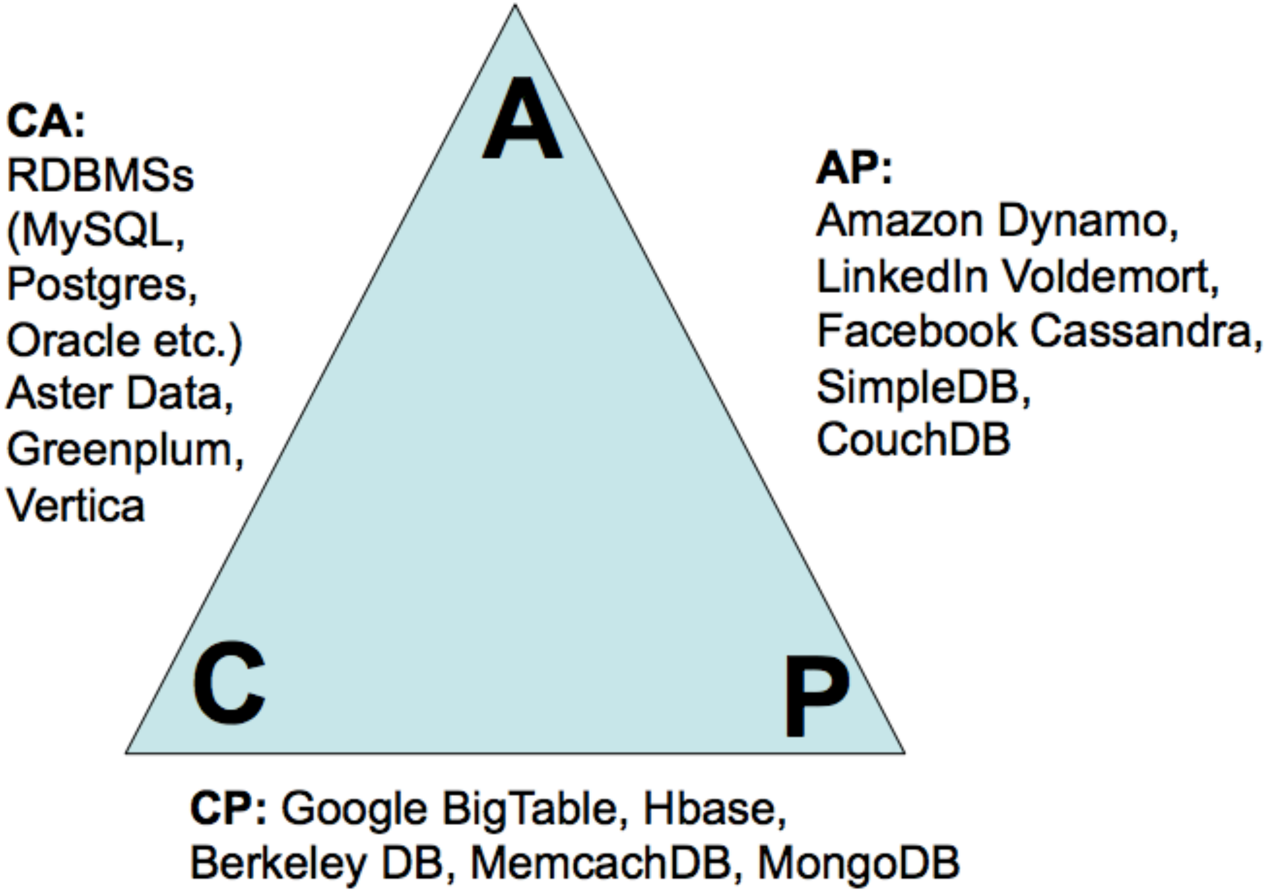
Solution – Non-relational (noSQL) databases (Column stores)

- Not using the relational model
- Running well on clusters
- Open-source
- Built for the 21st century web states
- Schemaless

Influential players

- Google and Amazon particularly influential in the rise of NoSQL databases
  - Google BigTable: powers Gmail, Google Maps, and other services
  - Amazon Dynamo: used e.g., by Amazon, Netflix
- Both capture huge amounts of data
- Both are running huge clusters of data
- Also popular: Open source solutions, e.g., Cassandra, HBase

Some recent systems



CA: RDBMSs (MySQL, Postgres, Oracle etc.) Aster Data, Greenplum, Vertica

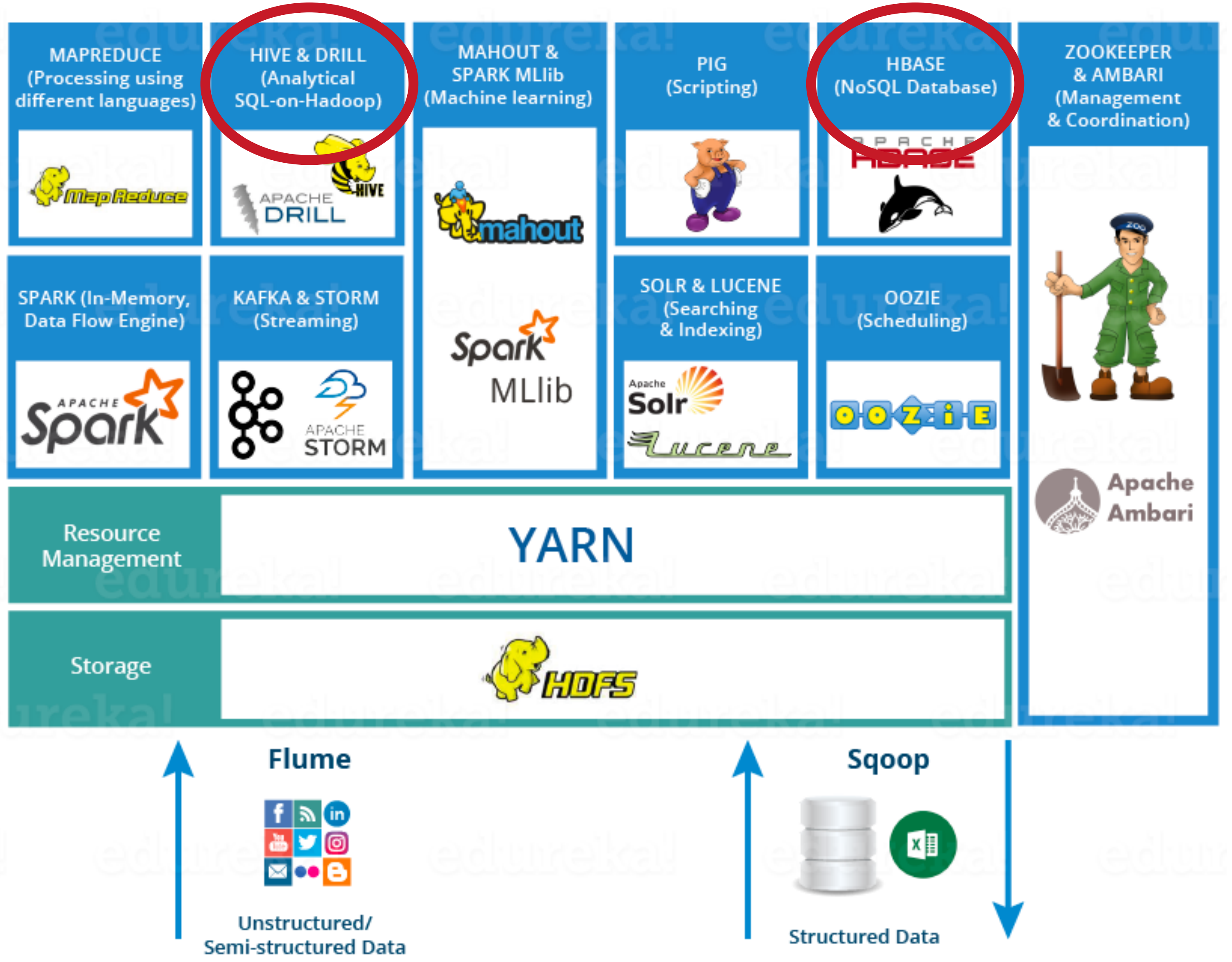
AP: Amazon Dynamo, LinkedIn Voldemort, Facebook Cassandra, SimpleDB, CouchDB

CP: Google BigTable, Hbase, Berkeley DB, MemcachDB, MongoDB

Ali Davoudian, Liu Chen, and Mengchi Liu. 2018. A Survey on NoSQL Stores. *ACM Comput. Surv.* 51, 2, Article 40 (March 2019), 43 pages.

http://blog.nahurst.com/visual-guide-to-nosql-systems

Hadoop ecosystem


MAPREDUCE (Processing using different languages)
MapReduce
HIVE & DRILL (Analytical SQL-on-Hadoop)
HIVE
APACHE DRILL
MAHOUT & SPARK MLlib (Machine learning)
mahout
Spark MLlib
PIG (Scripting)
HBASE (NoSQL Database)
APACHE HBASE
ZOOKEEPER & AMBARI (Management & Coordination)
ZOO
Apache Ambari
SPARK (In-Memory, Data Flow Engine)
APACHE Spark
KAFKA & STORM (Streaming)
APACHE STORM
SOLR & LUCENE (Searching & Indexing)
Apache Solr
Lucene
OOZIE (Scheduling)
OOZIE
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

Differences

- Relational databases (**Row** store):
  - Rows are stored consecutively
  - Optimal for row-wise access (e.g. SELECT *)

- Non-relational databases (**Column** store):
  - Columns are stored consecutively
  - Optimal for attribute focused access (e.g. SUM, GROUP BY)

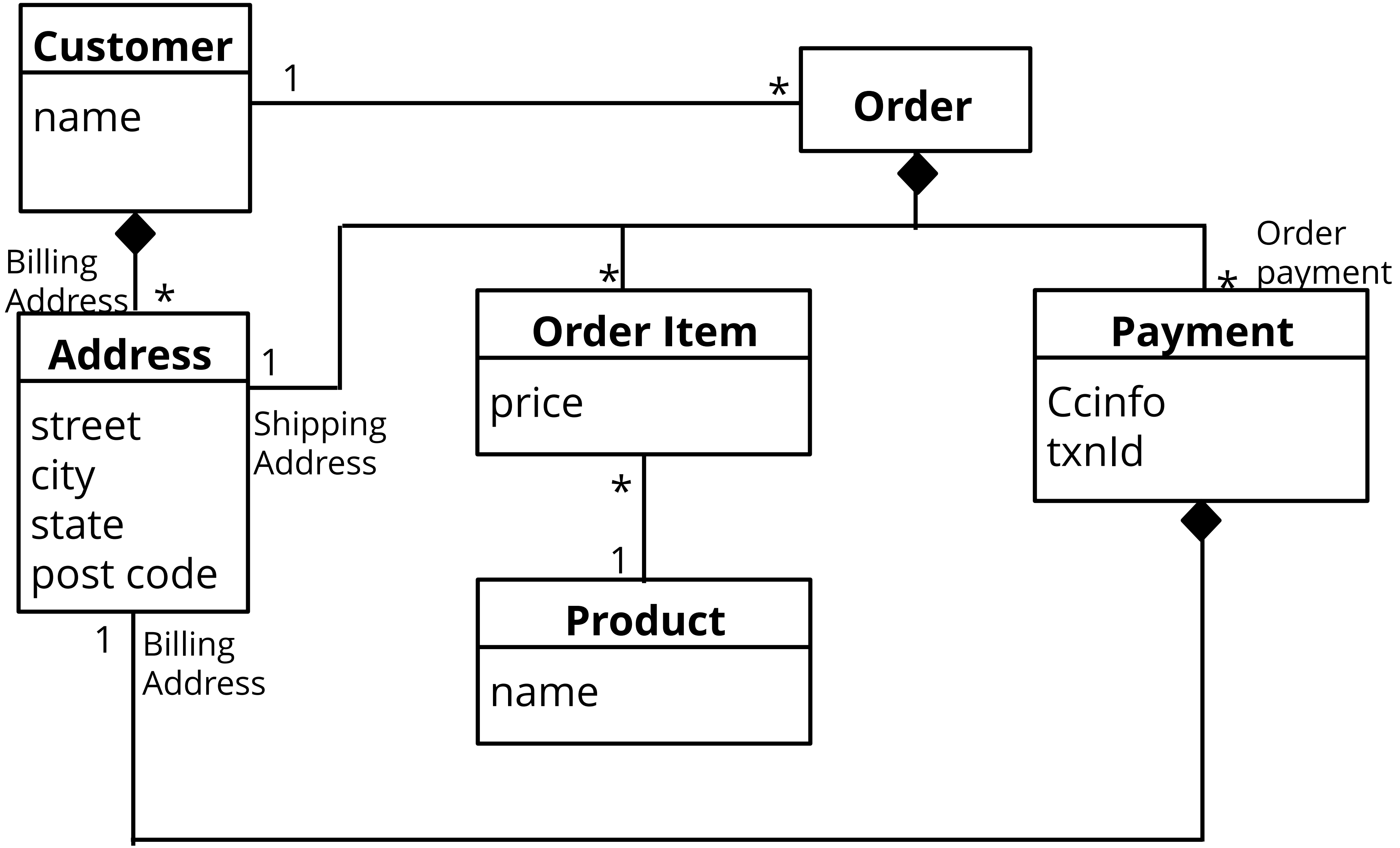
Paolo Atzeni, Francesca Bugiotti, Luca Cabibbo, Riccardo Torlone. Data Modeling in the NoSQL World. *Computer Standards and Interfaces*, 2020, 67, pp.103149. 10.1016/j.csi.2016.10.003 . hal- 01611628

Approach: Aggregates

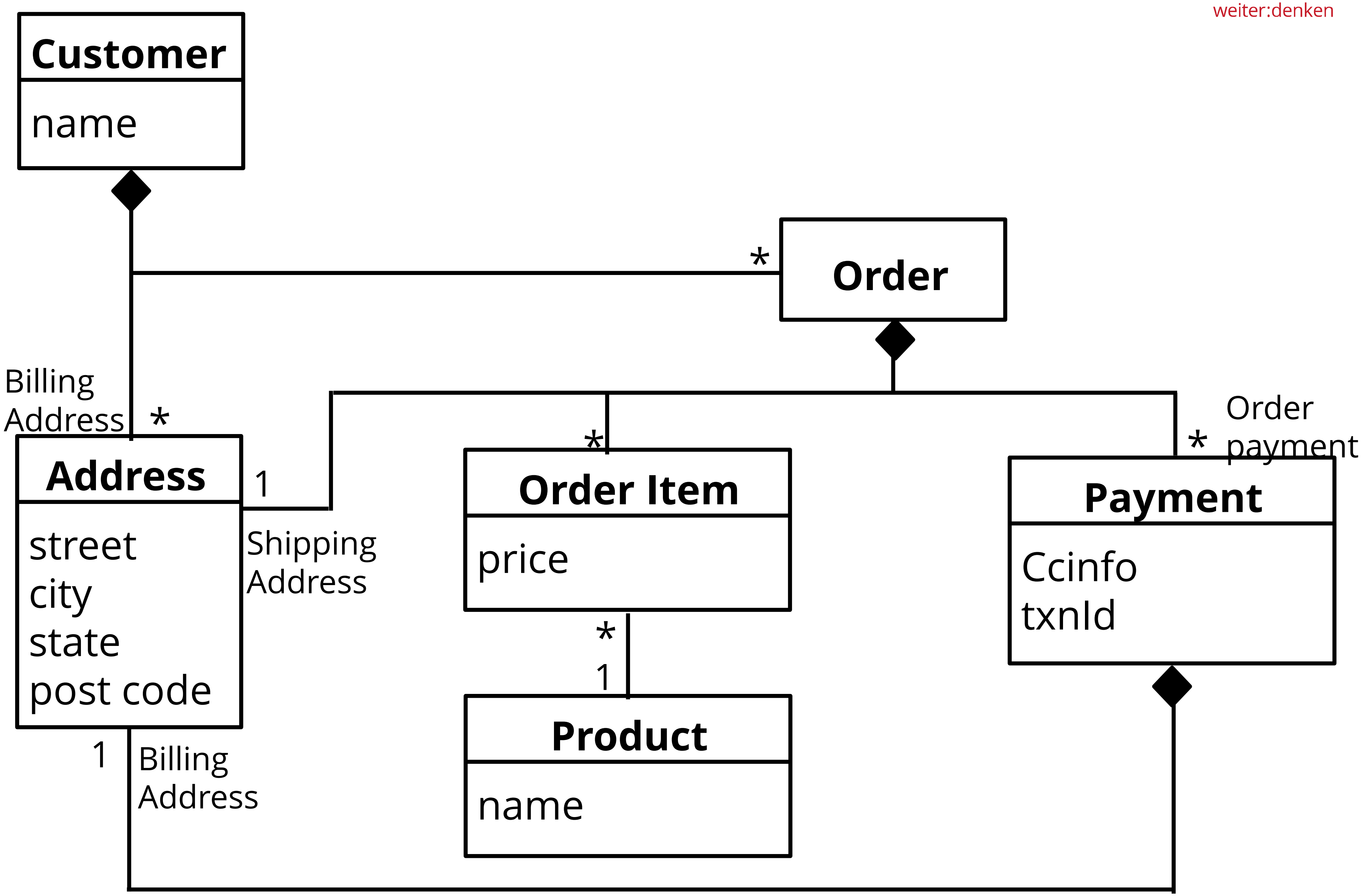
- Data is **aggregated** in units (records) that have more complex design than tuples

- *How* to aggregate depends on the application
  - Sometimes internal validation mechanisms that decide whether aggregate is acceptable

Example: Aggregate data model



Example: Aggregate data model (Alternative)



Consequences

- Aggregate-oriented databases work best when most data interaction is done with the same aggregate.

- An aggregate structure may help with some data interactions but be an obstacle for others.

- Aggregates make it easier for the database to manage data storage over clusters.

Consistency

- **Relational Databases:**
  - Allow you to manipulate any combination of rows from any table in a single transaction.
  - ACID transactions: atomic, consistent Isolated, and Durable

- **Aggregate-oriented databases:**
  - Support atomic manipulation of a single aggregate at a time
  - No ACID transactions that span multiple aggregates
  - We have to manage the atomic manipulation of multiple aggregates in our application code!

Outline

- Part 1:
  - Column stores (Introduction)
  - **Big Table**
  - HBase and Hive
- Part 2:
  - Job Scheduling
  - Coordination

Motivation

- Lots of (semi-)structured data at Google
  - URLs:
    - Contents, crawl metadata, links, anchors, pagerank, …
  - Per-user data:
    - User preference settings, recent queries/search results, …
  - Geographic locations:
    - Physical entities (shops, restaurants, etc.), roads, satellite image data, user annotations, …

- Scale is large
  - Billions of URLs, many versions/page (~20K/version)
  - Hundreds of millions of users, thousands or q/sec
  - 100TB+ of satellite image data

BigTable

- Distributed multi-level map
- Fault-tolerant, persistent
- Scalable
  - Thousands of servers
  - Terabytes of in-memory data
  - Petabyte of disk-based data
  - Millions of reads/writes per second, efficient scans
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2008. Bigtable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26, 2, Article 4 (June 2008), 26 pages.

Building Blocks

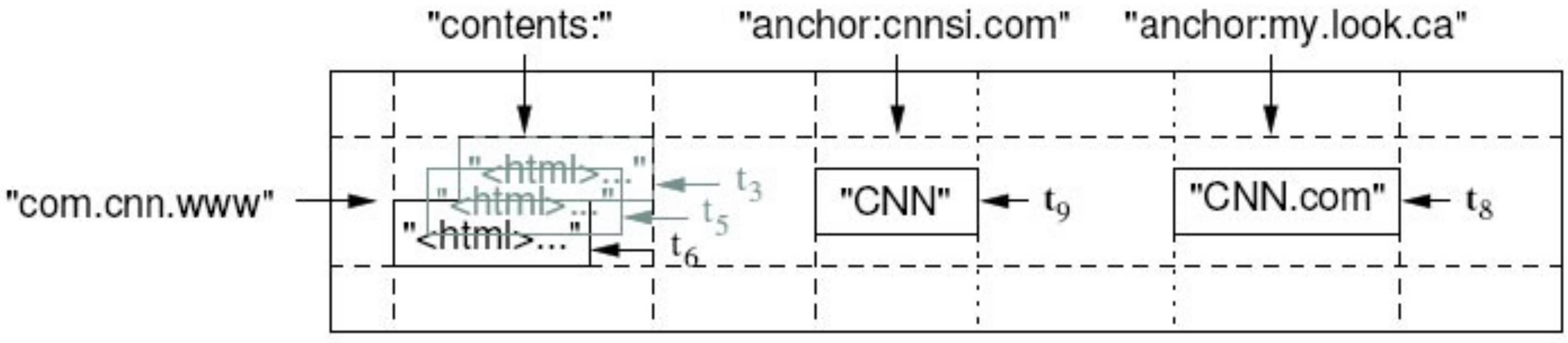
- Building blocks:
  - Google File System (GFS): Raw storage
  - Scheduler: schedules jobs onto machines
  - Lock service: distributed lock manager
  - MapReduce: simplified large-scale data processing

- BigTable's use of building blocks:
  - GFS: stores persistent data (SSTable file format for storage of data)
  - Scheduler: schedules jobs involved in BigTable serving
  - Lock service: master election, location bootstrapping
  - Map Reduce: often used to read/write BigTable data

Basic Data Model

- A BigTable is a sparse, distributed persistent multi-dimensional sorted map

(row, column, timestamp) -> cell contents



- Good match for most Google applications

Application of BigTable

https://www.youtube.com/watch?v=S7VTAoP0bu4

Outline

- Part 1:
  - Column stores (Introduction)
  - Big Table
  - **HBase and Hive**
- Part 2:
  - Job Scheduling
  - Coordination

History of HBase

- Started at the end of **2006**
- Modelled after **Google's Bigtable paper** (2006)
- **January 2008:** Hadoop becomes Apache top level project, HBase becomes subproject
- **May 2010:** HBase becomes an Apache top level project
- Contributors from Cloudera, Facebook, Intel, Hortonworks, etc.

Aiyer, Amitanand S., Mikhail Bautin, Guoqiang Jerry Chen, Pritam Damania, Prakash Khemani, Kannan Muthukkaruppan, Karthika Ranganathan, Nicolas Spiegelberg, Liyin Tang and Madhuwanti Vaidya. "Storage Infrastructure Behind Facebook Messages: Using HBase at Scale." *IEEE Data Eng. Bull.* 35 (2012): 4-13.

Apache HBase

- “Apache HBase is the Hadoop database, a distributed, scalable, big data store.”
- On top of HDFS
- Billions of rows * millions of columns
- Non-relational DB / NoSQL
- Column store: columns instead of tables
- Key-value cells

How HBase works

https://www.youtube.com/watch?v=lSrNUyMR_Ek

Demands of HBase

- **Structured data**, scaling to petabytes

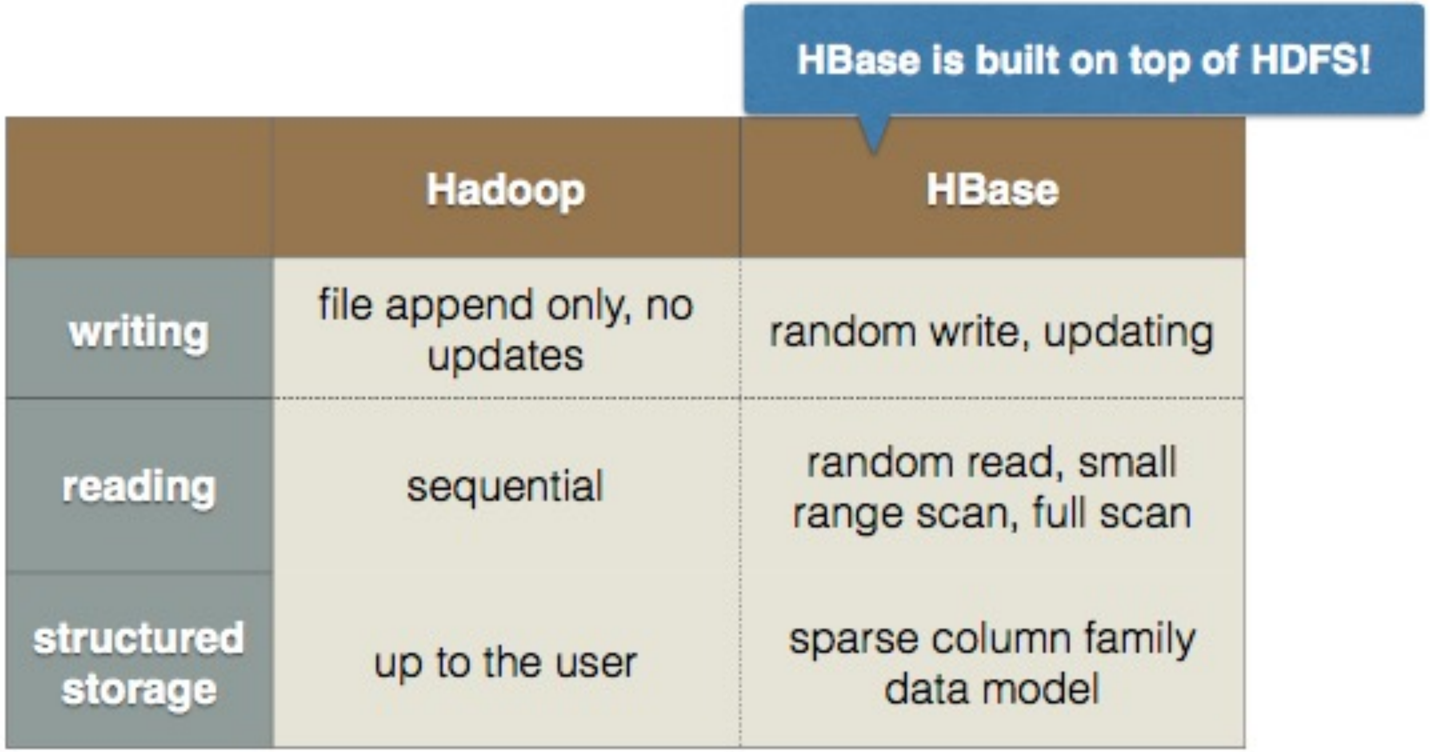
- **Efficient** handling of **diverse** data
  - wrt data size (URLs, web pages, satellite images)
  - wrt latency (backend bulk processing vs. real-time data serving)

- **Efficient** read and write of **individual records**

HBase vs. Hadoop

- Hadoop’s use case is **batch processing**
  - Not suitable for a single record lookup
  - Not suitable for adding **small amounts** of data at all times
  - Not suitable for making **updates** to existing records
- HBase addresses Hadoop’s weaknesses
  - Provides fast lookup of **individual** records
  - Supports **insertion** of **single** records
  - Supports record **updating**
  - Not all columns are of interest to everyone; each client only wants a particular subset of columns (**column-based storage**)

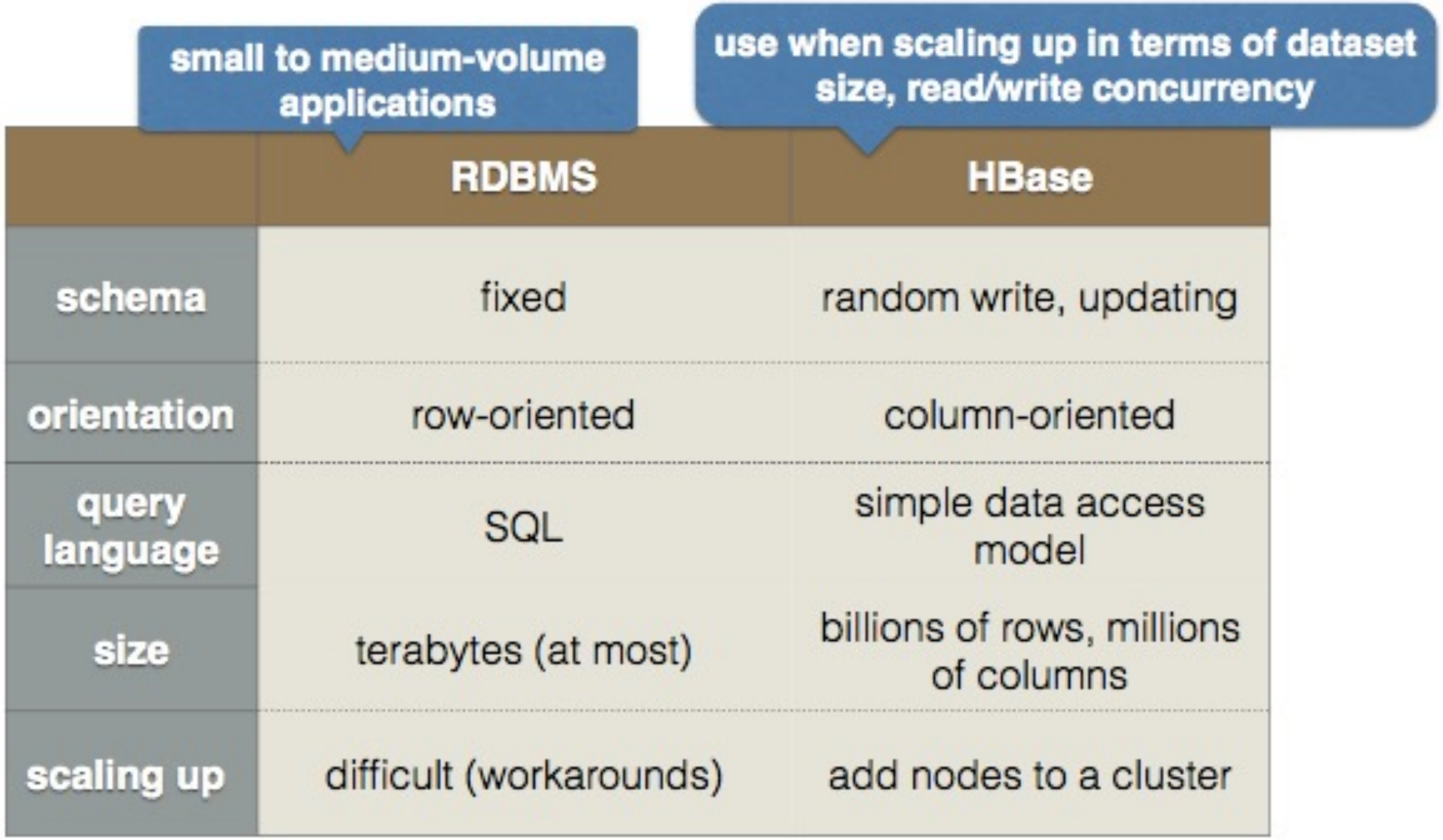
HBase vs. Hadoop



HBase is built on top of HDFS!

	Hadoop	HBase
writing	file append only, no updates	random write, updating
reading	sequential	random read, small range scan, full scan
structured storage	up to the user	sparse column family data model

HBase vs. Hadoop



small to medium-volume applications

use when scaling up in terms of dataset size, read/write concurrency

	RDBMS	HBase
schema	fixed	random write, updating
orientation	row-oriented	column-oriented
query language	SQL	simple data access model
size	terabytes (at most)	billions of rows, millions of columns
scaling up	difficult (workarounds)	add nodes to a cluster

Hive

- Developed at Facebook
- Used for majority of Facebook jobs
- “Relational database” built on Hadoop
  - Maintains list of table schemas
  - SQL-like query language (HiveQL)
  - Can call Hadoop Streaming scripts from HiveQL
  - Supports table partitioning, clustering, complex data types, some optimizations

Outline

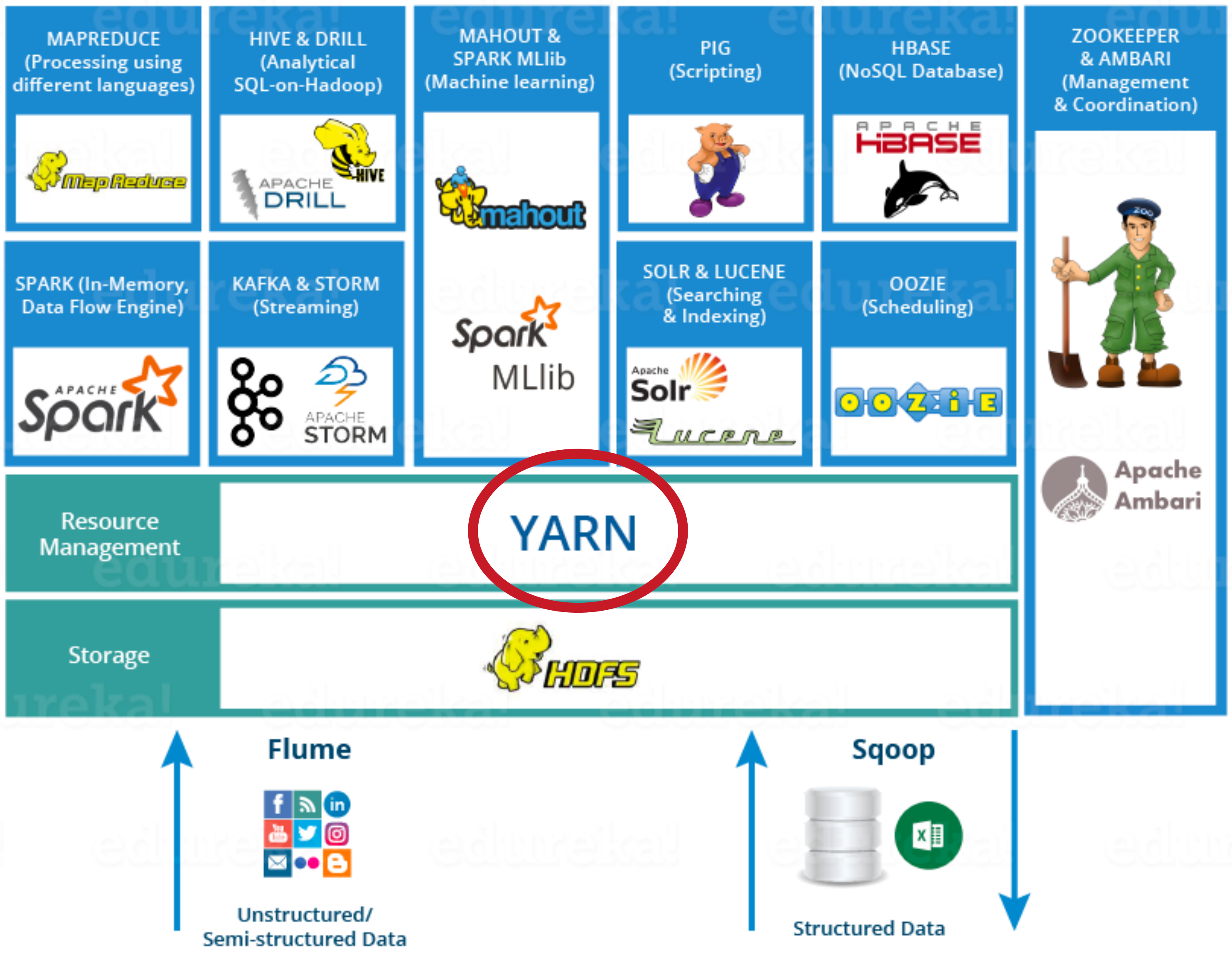
- Part 1:
  - Column stores (Introduction)
  - Big Table
  - HBase and Hive
- **Part 2:**
  - **Job Scheduling**
  - Coordination

Hadoop Scheduling

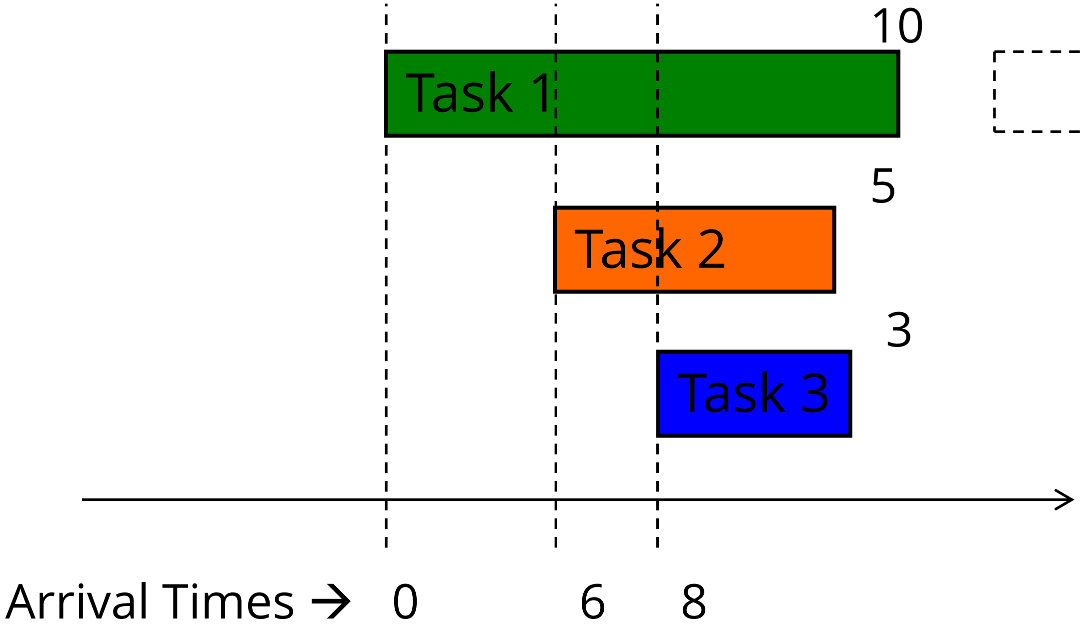
universität koblenz
weiter:denken

- A Hadoop job consists of Map tasks and Reduce tasks
- Multiple customers with multiple jobs
  - Users/jobs = “tenants”
  - Multi-tenant system
- Need a way to schedule all these jobs (and their constituent tasks)
- Need to be *fair* across the different tenants

Hadoop ecosystem



Scheduling

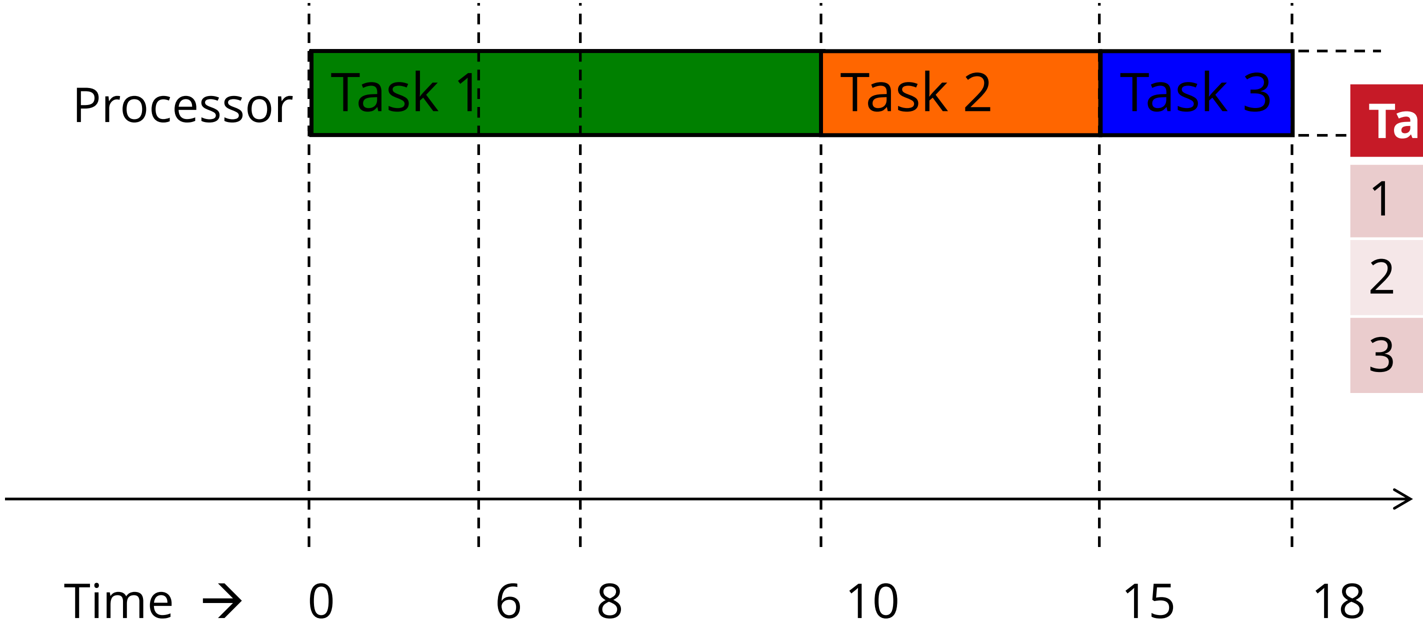


Which tasks run when?

Processor

Task	Length	Arrival
1	10	0
2	5	6
3	3	8

FIFO Scheduling (First-In First-Out) / FCFS



Task	Length	Arrival
1	10	0
2	5	6
3	3	8

- *Maintain tasks in a queue in order of arrival*
- *When processor free, dequeue head and schedule it*

FIFO/FCFS Performance

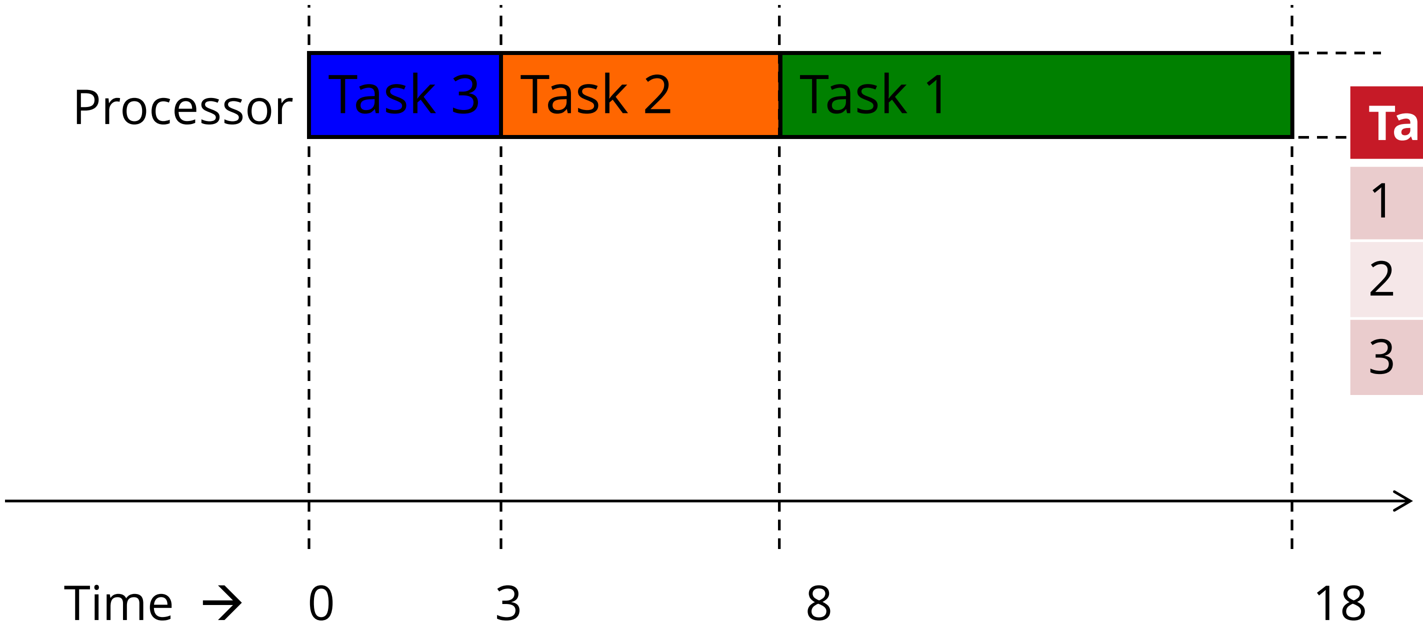
- Average completion time may be high
- For our example on previous slides,
  - average completion time of FIFO/FCFS = (Task 1 + Task 2 + Task 3)/3

    = (10+15+18)/3

    = 43/3

    = 14.33

STF Scheduling (Shortest Task First)



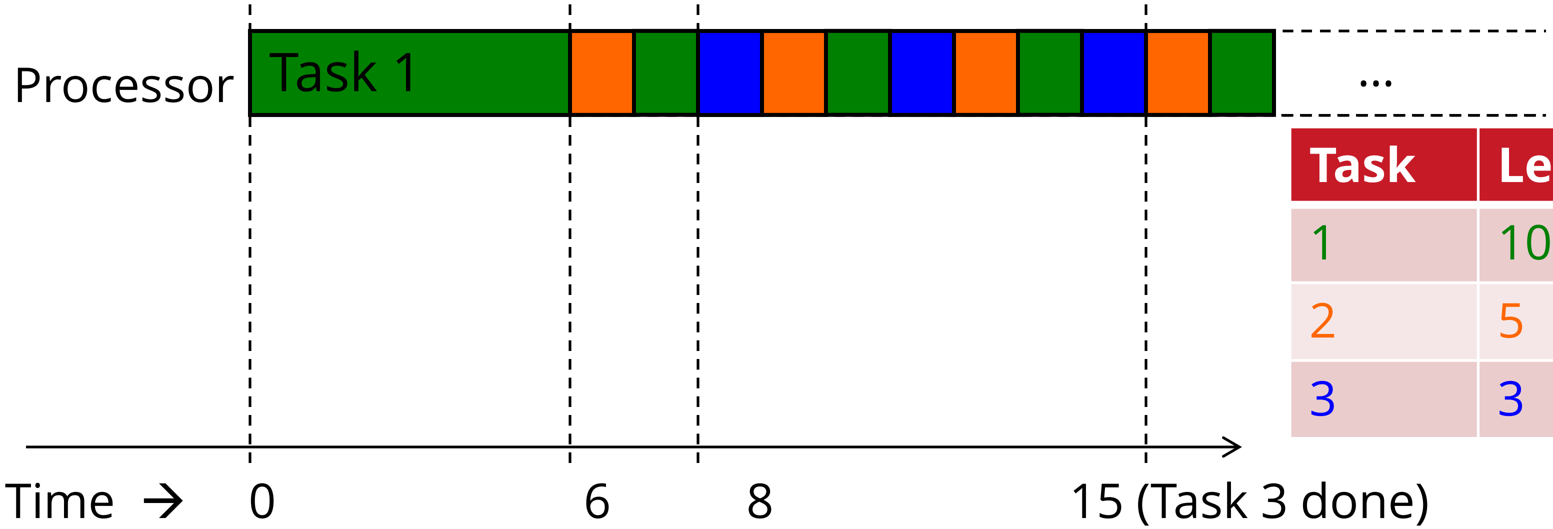
Task	Length	Arrival
1	10	0
2	5	0
3	3	0

- *Maintain all tasks in a queue, in increasing order of running time*
- *When processor free, dequeue head and schedule*

STF is Optimal!

- Average completion of STF is the shortest among *all* scheduling approaches!
- For our example on previous slides,
  - Average completion time of STF =
    - (Task 1 + Task 2 + Task 3)/3
    - = (18+8+3)/3
    - = 29/3
    - = 9.66
    - (versus 14.33 for FIFO/FCFS)
- In general, STF is a special case of *priority scheduling*
  - Instead of using time as priority, scheduler could use user-provided priority

Round-Robin Scheduling



Task	Length	Arrival
1	10	0
2	5	6
3	3	8

- *Use a quantum (say 1 time unit) to run portion of task at queue head*
- *Pre-empts processes by saving their state, and resuming later*
- *After pre-empting, add to end of queue*

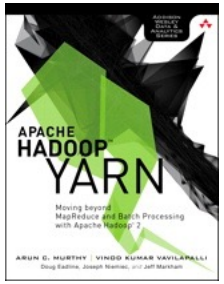
Round-Robin vs. STF/FIFO

- Round-Robin preferable for
  - Interactive applications
  - User needs quick responses from system
- FIFO/STF preferable for Batch applications
  - User submits jobs, goes away, comes back to get result

Apache Hadoop Yarn

- Provides resource management to support processing petabytes of data.

- Hadoop YARN has two popular schedulers
  - *Hadoop Capacity Scheduler*
  - *Hadoop Fair Scheduler*


APACHE
HADOOP
YARN
Moving beyond MapReduce and Batch Processing with Apache Hadoop 2
ARUN C. MURTHY | VINOD KUMAR VAVILAPALLI
Doug Eadline, Joseph Niemiec, and Jeff Markham

Murthy et al. Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2, Addison-Wesley, 2014

Hadoop Capacity Scheduler

- Contains multiple queues
- Each queue contains multiple jobs
- Each queue guaranteed some portion of the cluster capacity, e.g.,
  - Queue 1 is given 80% of cluster
  - Queue 2 is given 20% of cluster
  - Higher-priority jobs go to Queue 1
- For jobs within same queue, FIFO typically used
- Administrators can configure queues

Source: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html

Elasticity in HCS

- Administrators can configure each queue with limits
  - Soft limit: how much % of cluster is the queue guaranteed to occupy
  - (Optional) Hard limit: max % of cluster given to the queue
- Elasticity
  - A queue allowed to occupy more of cluster if resources free
  - But if other queues below their capacity limit, now get full, need to give these other queues resources
- Pre-emption not allowed!
  - Cannot stop a task part-way through
  - When reducing % cluster to a queue, wait until some tasks of that queue have finished

Other HCS Features

- Queues can be hierarchical
  - May contain child sub-queues, which may contain child sub-queues, and so on
  - Child sub-queues can share resources equally
- Scheduling can take memory requirements into account (memory specified by user)

Hadoop Fair Scheduler

- Goal: all jobs get equal share of resources
- When only one job present, occupies entire cluster
- As other jobs arrive, each job given equal % of cluster
  - E.g., each job might be given equal number of cluster-wide YARN containers
  - Each container == 1 task of job

Source: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/FairScheduler.html

Hadoop Fair Scheduler (2)

- Divides cluster into pools
  - Typically one pool per user
- Resources divided equally among pools
  - Gives each user fair share of cluster
- Within each pool, can use either
  - Fair share scheduling, or
  - FIFO/FCFS
  - (Configurable)

Pre-emption in HFS

- Some pools may have *minimum shares*
  - Minimum % of cluster that pool is guaranteed
- When minimum share not met in a pool, for a while
  - Take resources away from other pools
  - By pre-empting jobs in those other pools
  - By *killing* the currently-running tasks of those jobs
    - Tasks can be re-started later
    - Ok since tasks are idempotent!
  - To kill, scheduler picks most-recently-started tasks
    - Minimizes wasted work

Other HFS Features

- Can also set limits on
  - Number of concurrent jobs per user
  - Number of concurrent jobs per pool
  - Number of concurrent tasks per pool

- Prevents cluster from being hogged by one user/job

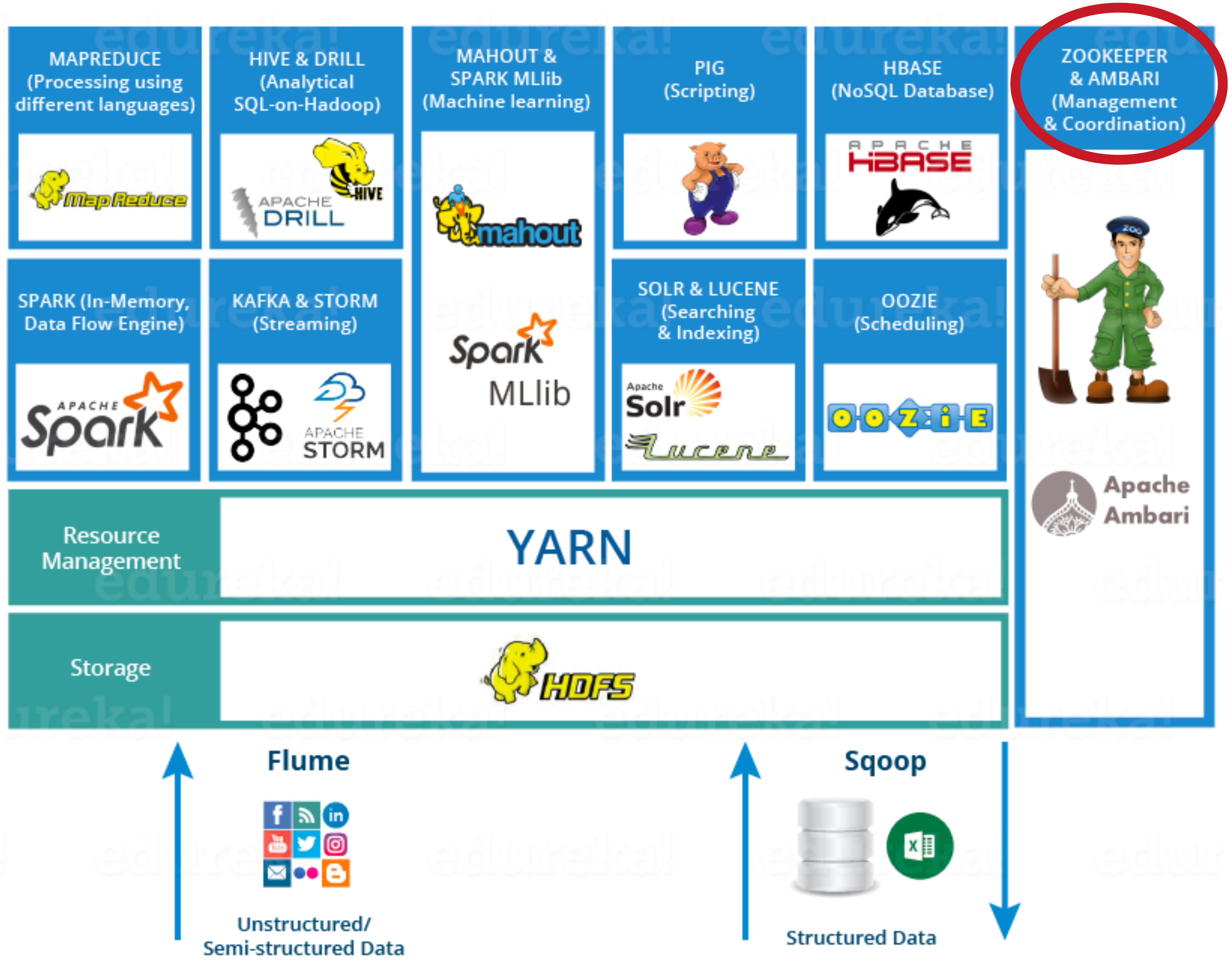
Outline

- Part 1:
  - Column stores (Introduction)
  - Big Table
  - HBase and Hive
- **Part 2:**
  - Job Scheduling
  - **Coordination**

Challenge

- In the past: a **single** program running on a **single** computer with a **single** CPU
- Today: applications consist of **independent** programs running on a **changing** set of computers
- Difficulty: **coordination** of those independent programs
- Developers have to deal with **coordination logic** and **application logic** at the same time

Hadoop ecosystem



MAPREDUCE (Processing using different languages)

HIVE & DRILL (Analytical SQL-on-Hadoop)

MAHOUT & SPARK MLlib (Machine learning)

PIG (Scripting)

HBASE (NoSQL Database)

ZOOKEEPER & AMBARI (Management & Coordination)

SPARK (In-Memory, Data Flow Engine)

KAFKA & STORM (Streaming)

SOLR & LUCENE (Searching & Indexing)

OOZIE (Scheduling)

Resource Management

YARN

Storage

Flume

Unstructured/ Semi-structured Data

Sqoop

Structured Data

Apache Zookeeper

- ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
- https://zookeeper.apache.org/

What is Zookeeper?

IBM Big Data

What is Zookeeper?

with Rafael Coss,
Manager Big Data Enablement

August 2012

Source: https://www.youtube.com/watch?v=Kgf9EjTNucM

Zookeeper design

- ZooKeeper service is an ensemble of servers that use replication (high availability)

- (meta)data is cached on the client side
  - ID of datanodes, instead of probing ZooKeeper every time.
  - What if data changes?
    - Polling?
    - Watch mechanism: clients can watch for an update of a given object

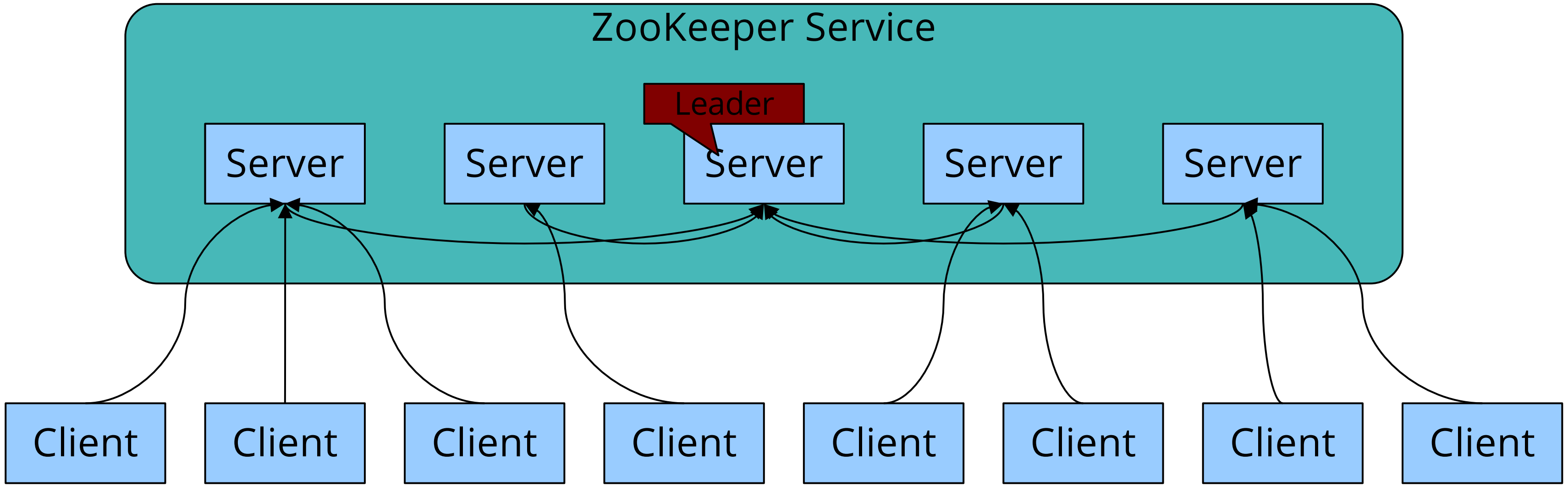
Zookeeper design

- ZooKeeper data is replicated on each server that composes the service
  - Recovery from master failures

- Clients connect to exactly one server to submit requests
  - **read** requests served from the local replica
  - **write** requests are processed by an agreement protocol (an elected server leader initiates processing of the write request)

“ZooKeeper: Wait-free coordination for Internet-scale systems”, Hunt et al., USENIX 2010

Zookeeper Servers



- All servers store a copy of the data (in memory)
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

Summary

- Big Table
- HBase and Hive
- Job Scheduling
- Coordination

What’s next – Distributed Data Processing

- Map/Reduce framework

- Querying
  - Spark Core API
  - Pig, Pig Latin

- Machine Learning at Scale