Lecture @DHBW: Data Warehouse

02 Tools

Andreas Buckenhofer

Daimler TSS
Ein Unternehmen der Daimler AG

Andreas Buckenhofer

Senior DB Professional

Since 2009 at Daimler TSS
Department: Machine Learning Solutions
Business Unit: Analytics

vcard

Contact/Connect

- Over 20 years experience with database technologies
- Over 20 years experience with Data Warehousing
- International project experience

- Oracle ACE Associate
- DOAG responsible for InMemory DB
- Lecturer at DHBW
- Certified Data Vault Practitioner 2.0
- Certified Oracle Professional
- Certified IBM Big Data Architect

Change Log

Date	Changes
10.10.2019	Initial version
17.10.2019	Solutions published on slides 40-42 and 62-64

What you will learn today

- Get a basic understanding for
  - Ingesting data
  - Storing data
  - Visualizing + Analyzing data
  - Cataloging data
- Differentiate between tools suitable for DWH and Big Data
- Describe the characteristics of different storage systems
  - RDBMS
  - NoSQL, Hadoop, Spark, Messaging
- Understand the differences between schema-on-read and schema-on-write

Tools

Data Integration (ETL) Tools

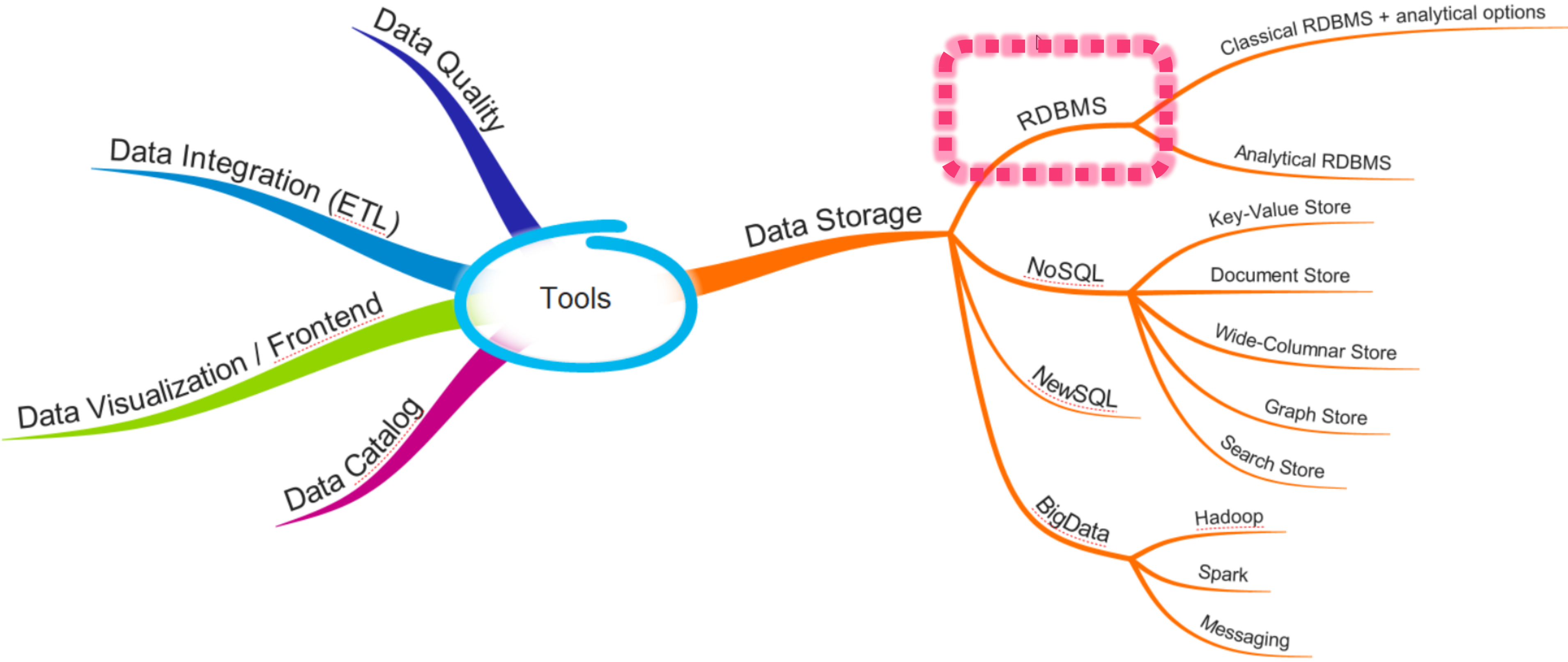
Data Quality Tools

Data Storage Tools
- RDBMS
- NoSQL
- Hadoop

Data Catalog Tools

Data Visualization Tools

Tools



Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Relational Database Management Systems

ORACLE DATABASE

EXASOL

Microsoft SQL Server

VERTICA

DB2

snowflake

SAP HANA

amazon REDSHIFT

PostgreSQL

MySQL

Google BigQuery

SQL

ACID

Tables

Indexes

Backup

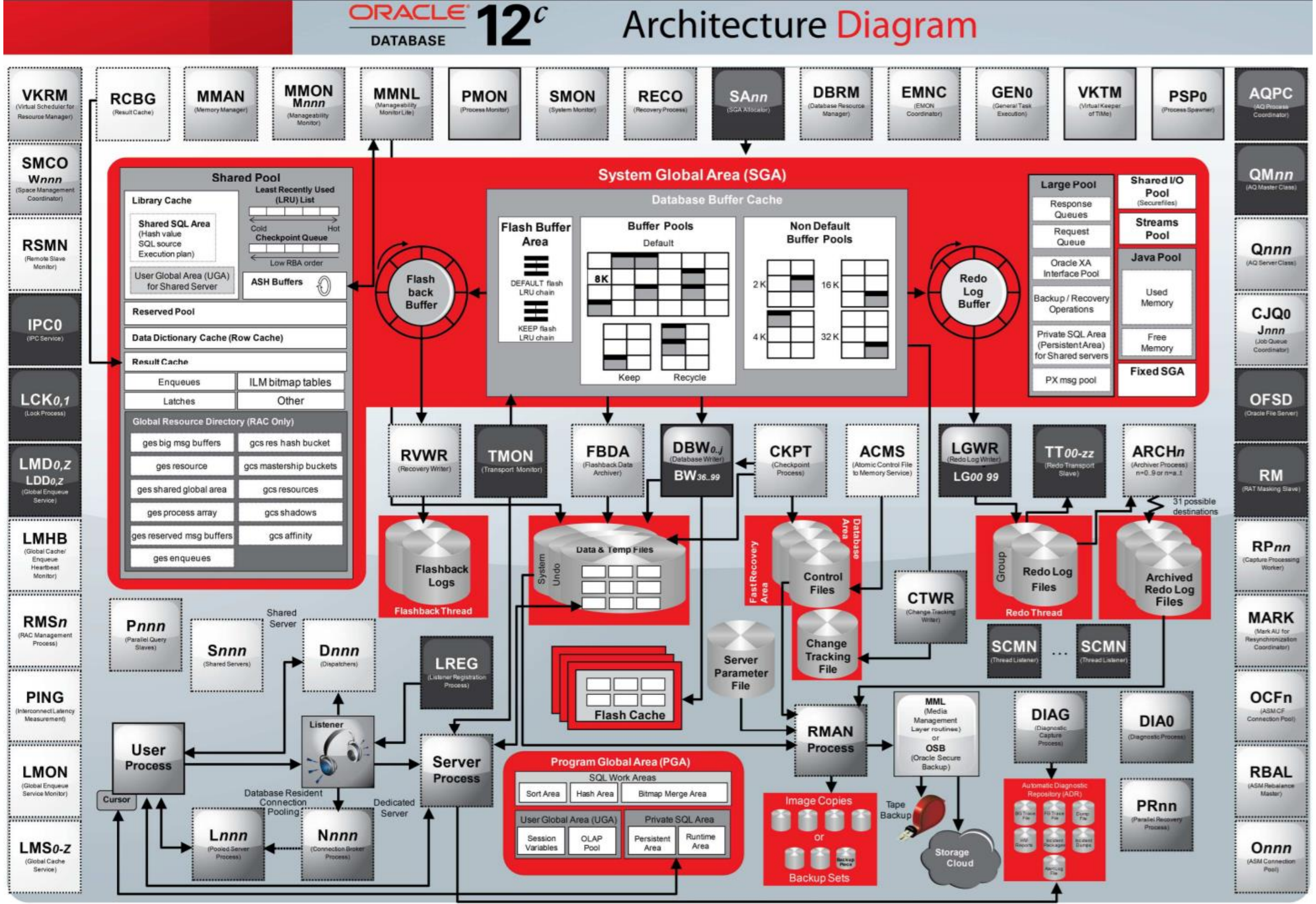
Recovery

Caches

Tablespace

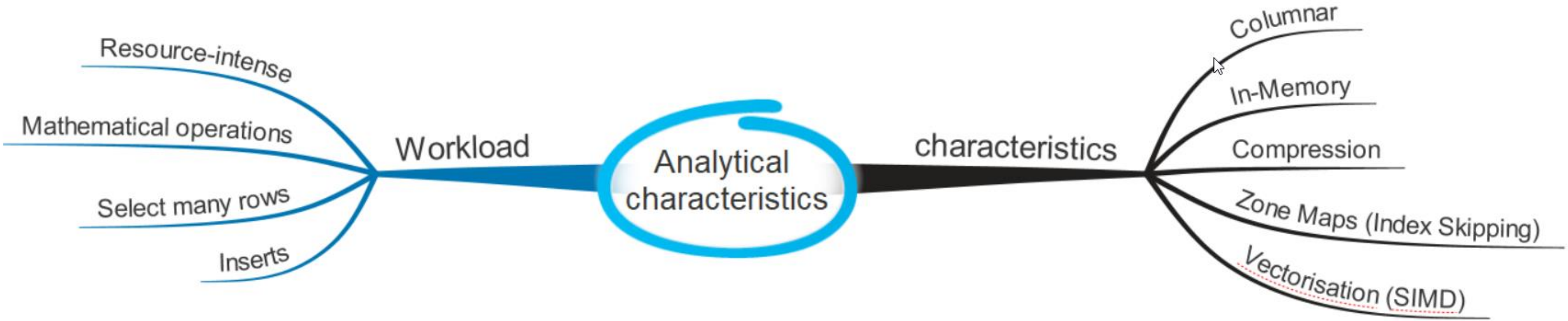
Tuning

Partitions



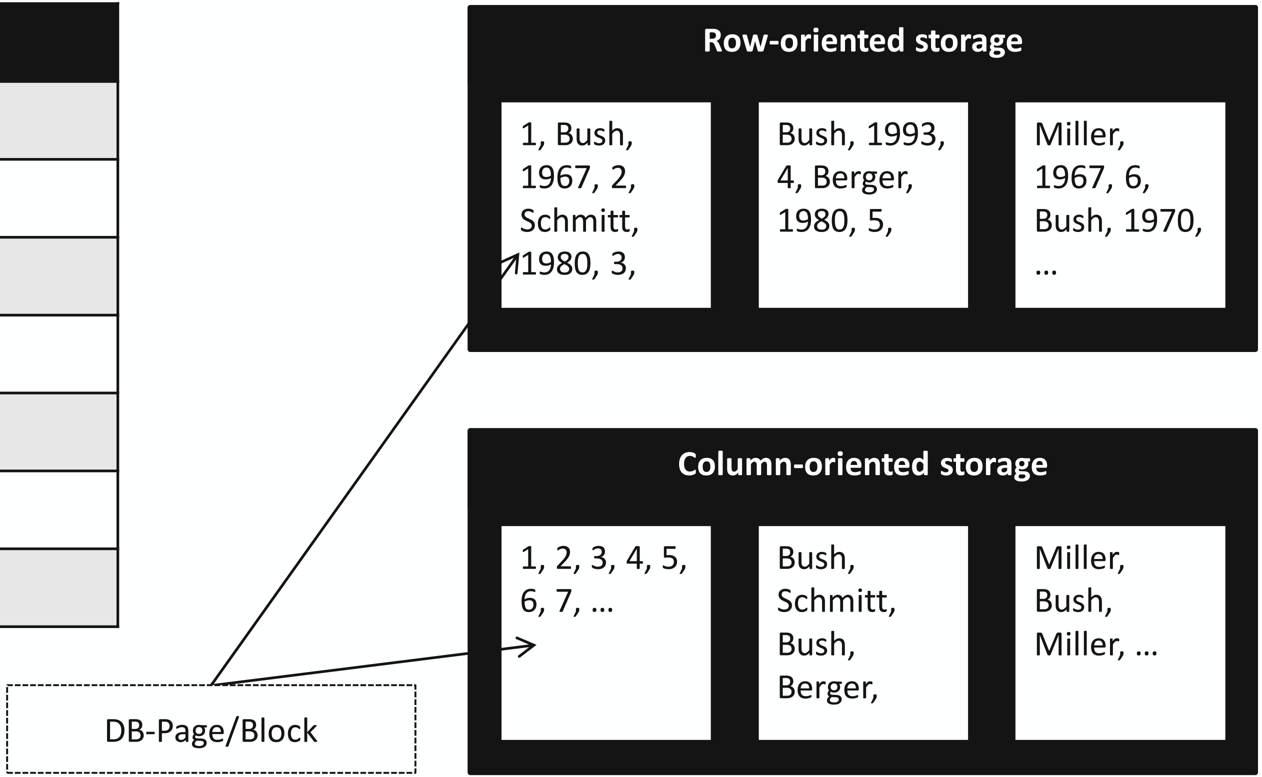
Source:
https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/poster/OUTPUT_poster/poster.html
https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/poster/OUTPUT_poster/pdf/Database%20Architecture.pdf

Characteristics of analytical databases



Row and column oriented db block storage

Id	Name	Birthdate
1	Bush	1967
2	Schmitt	1980
3	Bush	1993
4	Berger	1980
5	Miller	1967
6	Bush	1970
7	Miller	1980



Row vs column-oriented storage: row-oriented storage

Row-oriented storage

- Data of one row is grouped on disk and can be retrieved through one read operation
- Single values can be retrieved through efficient index and off-set computations
- Good Insert, update and delete operations performance

→ **Suited for OLTP systems**

Row vs column-oriented storage: column-oriented storage

Column-oriented storage

- Data-of one column is grouped on disk and can be retrieved with far fewer read operations than for row-oriented storage
- This makes computation of aggregations much faster for tables with a lot of columns
- In general better suited for queries involving partial table scans
- Bad Insert, update and delete operations performance
- Normally excellent compression as identical data types are stored in same blocks

→ **Suited for OLAP systems**

Row format vs columnar format

Source: https://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.pdf

In-memory cache latency

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 μs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

„In-Memory“

Hard disk

volatile

non-volatile

Source: Brendan Gregg: “Systems Performance: Enterprise and the Cloud”

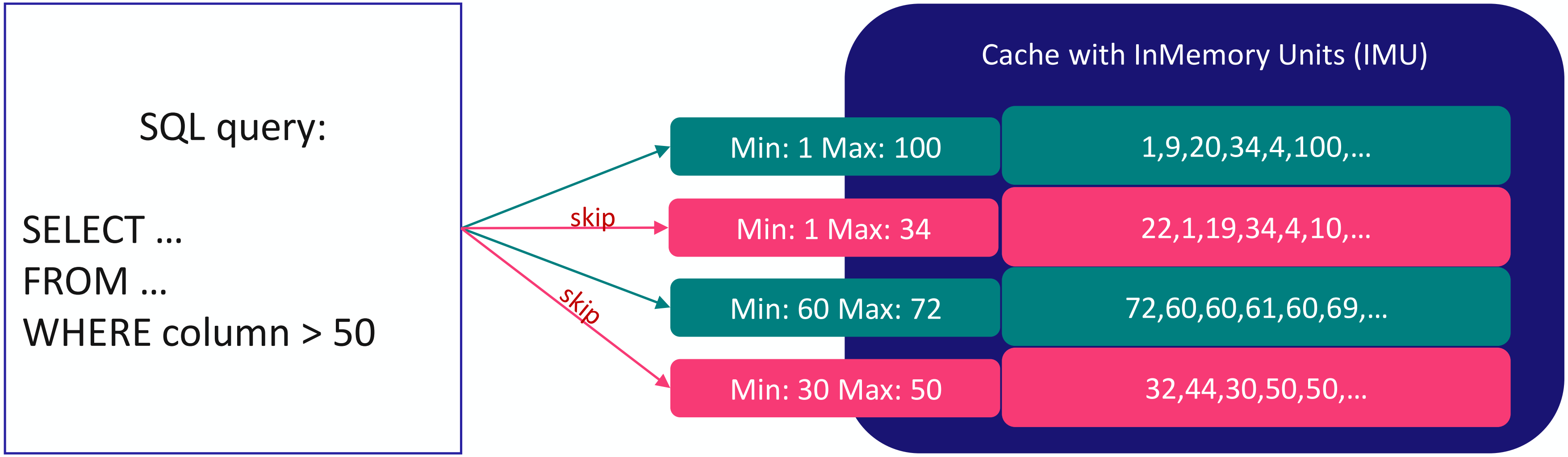
Compression

High chance of repetitive data in a block, e.g. dates like shipping date or transaction date, status field or other fields with known reference data

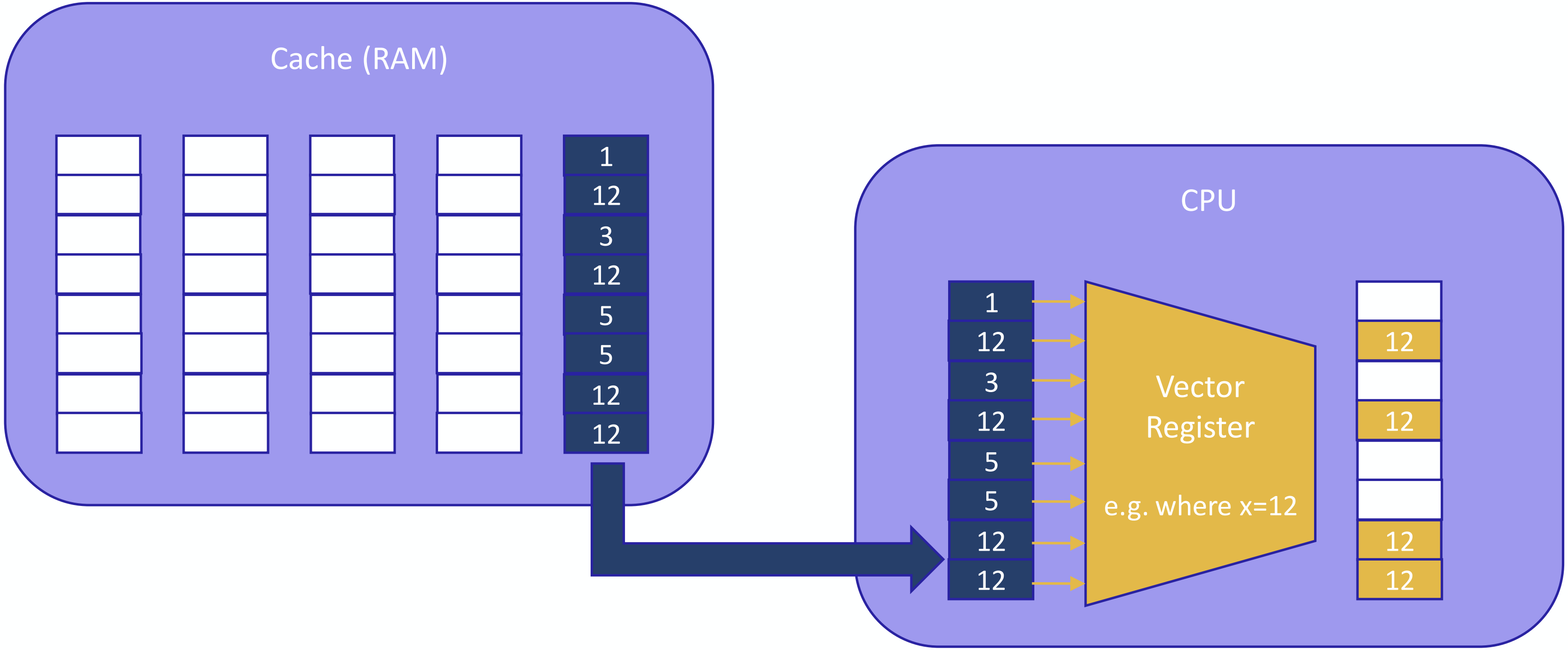
- Repetitive data can be omitted
- Additionally data can be compressed (high vs low compression)

Column-oriented storage		
01.01.2015, 01.01.2015, 02.01.2015, 02.10.2015	Bush, Schmitt, Bush, Berger,	Male, Male Female, Male

Zone Maps (Index skipping)



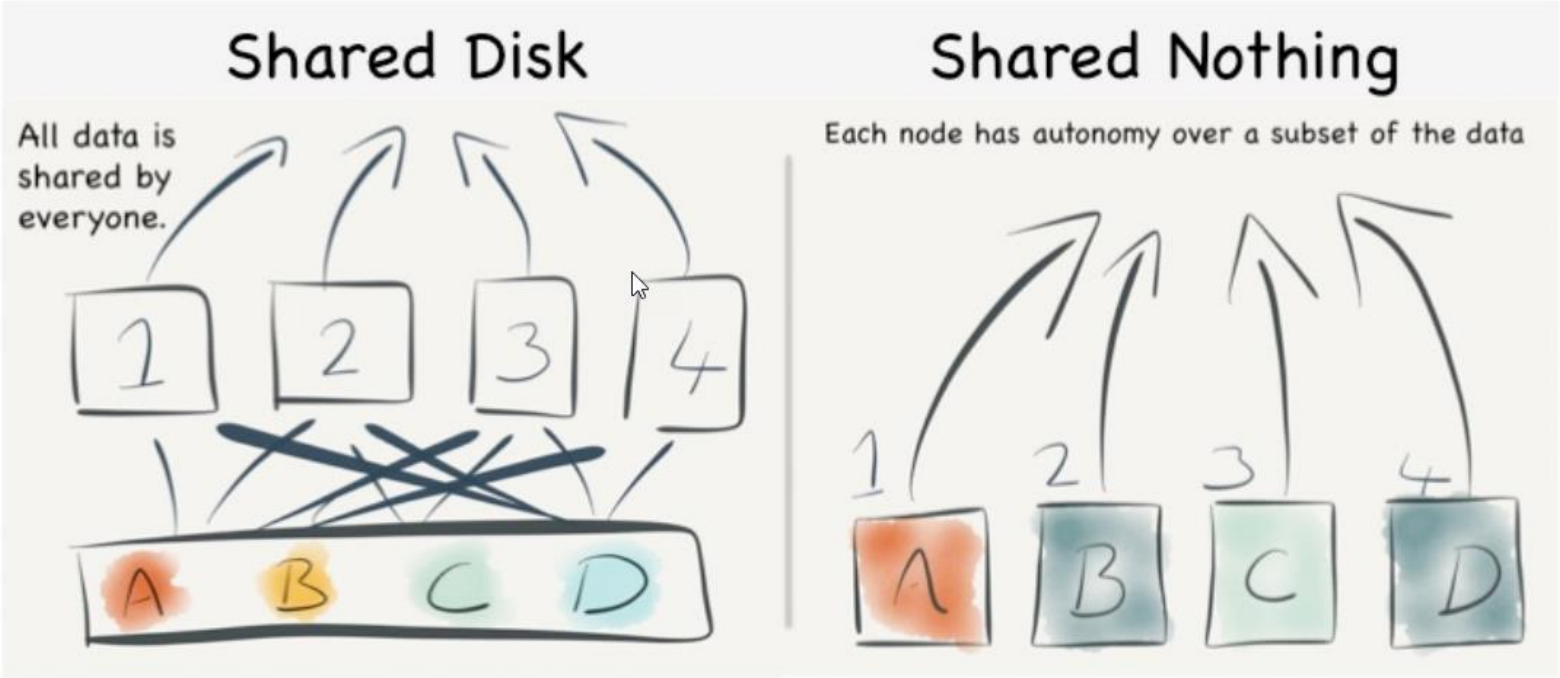
SIMD (single instruction, multiple data)



Analytical RDBMS

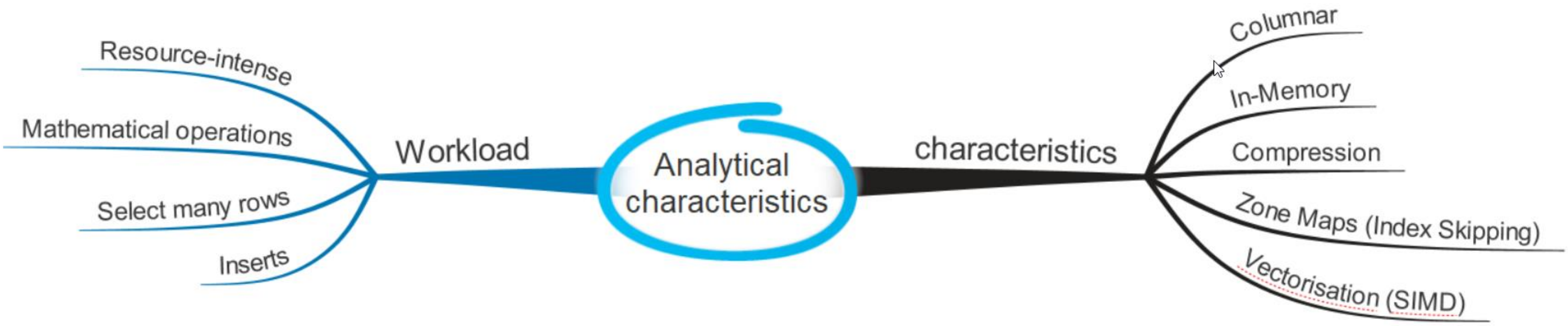
- Classical RDBMS with additional columnar format
  - Oracle DBIM, IBM BLU, SQL Server Columnstore Index, SAP HANA
  - Shared disk architecture
  - Adjusted configuration compared to OLTP applications, e.g. larger memory area for hash joins
- Dedicated analytical RDBMS with columnar format only (true analytical database)
  - Exasol, Vertica, (Teradata: -> row-oriented storage)
  - Cloud-only: Amazon Redshift, Snowflake
  - Shared Nothing architecture

Shared disk vs shared nothing architecture



Source: http://www.benstopford.com/2009/11/24/understanding-the-shared-nothing-architecture/

Characteristics of analytical databases



One size does not fit all

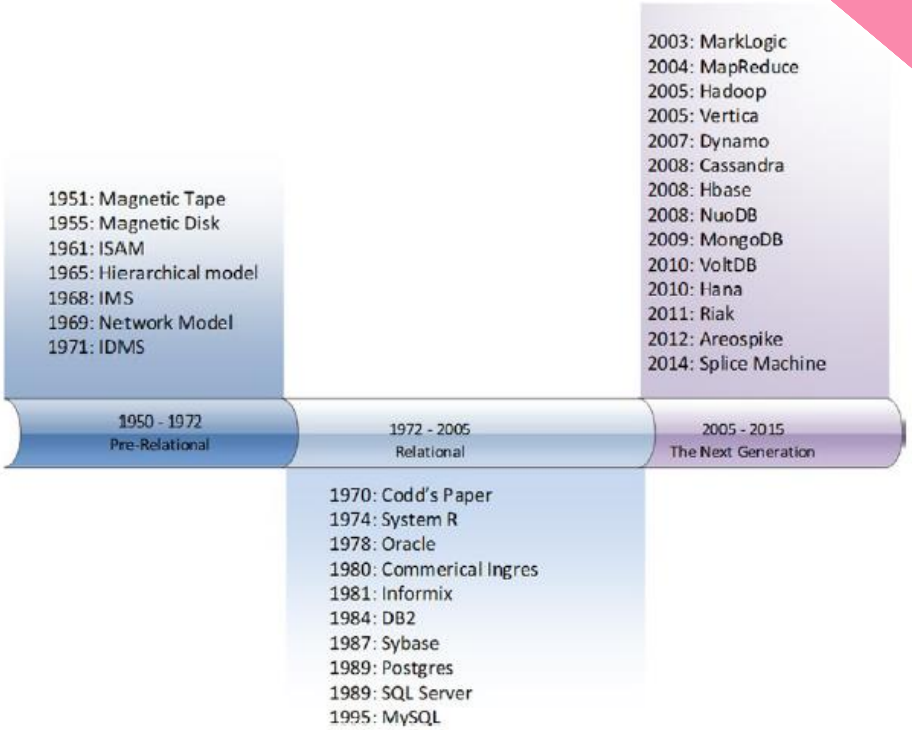


Figure 1-1. Timeline of major database releases and innovations

Stonebraker:
"One Size Fits All":
An Idea Whose Time Has Come and Gone

https://cs.brown.edu/~ugur/fits_all.pdf

Source: Harrison, Next Generation Databases, Apress 2016, p. 4

Two major developments in the 2000ies lead to new databases

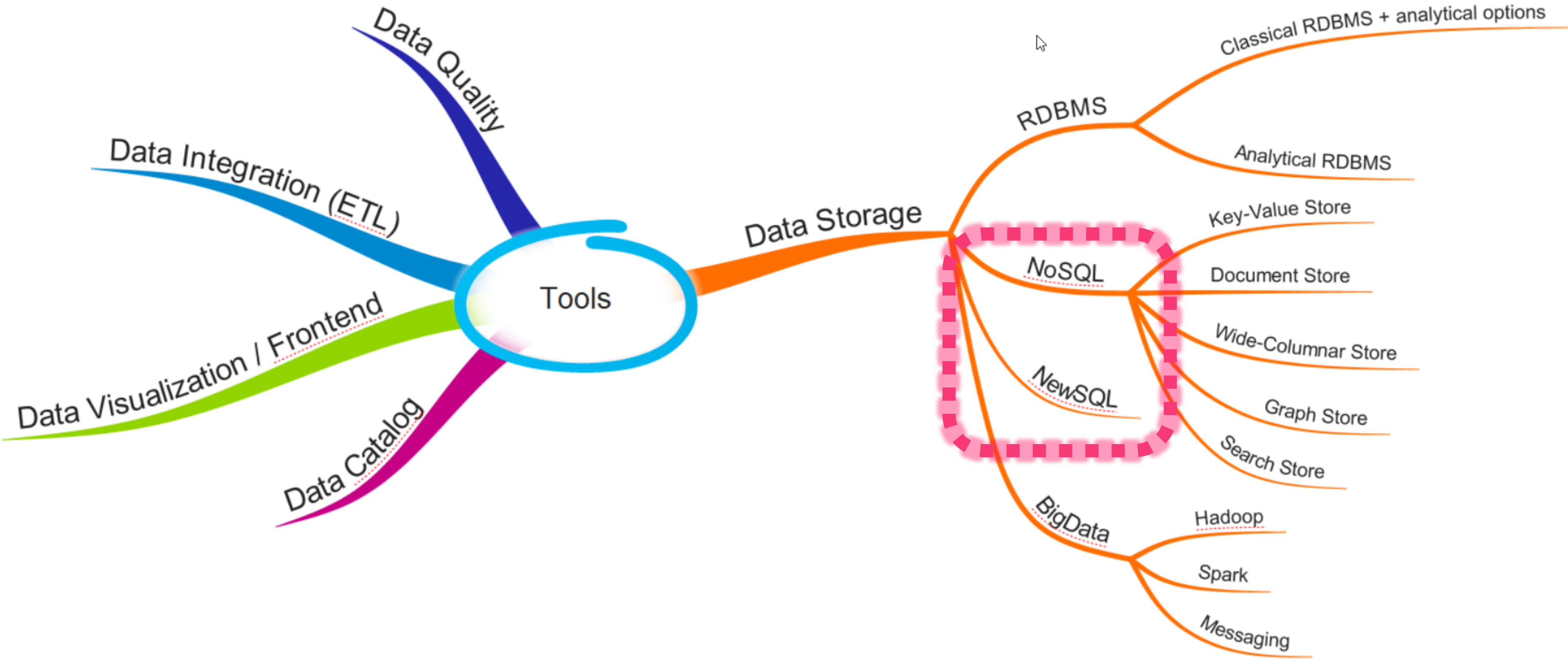
Google

- New hardware and software architectures to **store and process** the **exponentially growing quantity** of websites it needed to index

Amazon

- Webscale: **transactional processing** capability that could operate **at massive scale**

Tools



Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

NoSQL – Key-Value stores

- Simple physical model with pairs of
  - Unique key
  - Values (atomic or complex)
- Access only possible via Key
- Main use case is for Caching
- Examples: Redis, Aerospike, Oracle NoSQL.

Key	Value
userID1	ISBN1
userID2	ISBN2, ISBN8, ISBN9
userID3	

NoSQL – Document stores

- Structures like XML, JSON, BSON are stored in the DB
- Flexible schema
- Data and metadata are mixed
- Access via key or index
- DB does not interpret model
- Examples: MongoDB, CouchDB

ID: 12345
Name: Mustermann
Born: 04.02.1992

ID: 637
Name: Berger
Adress:

From 01.01.2005
zip: 89004

from 01.07.2010
zip: 80990
city: München

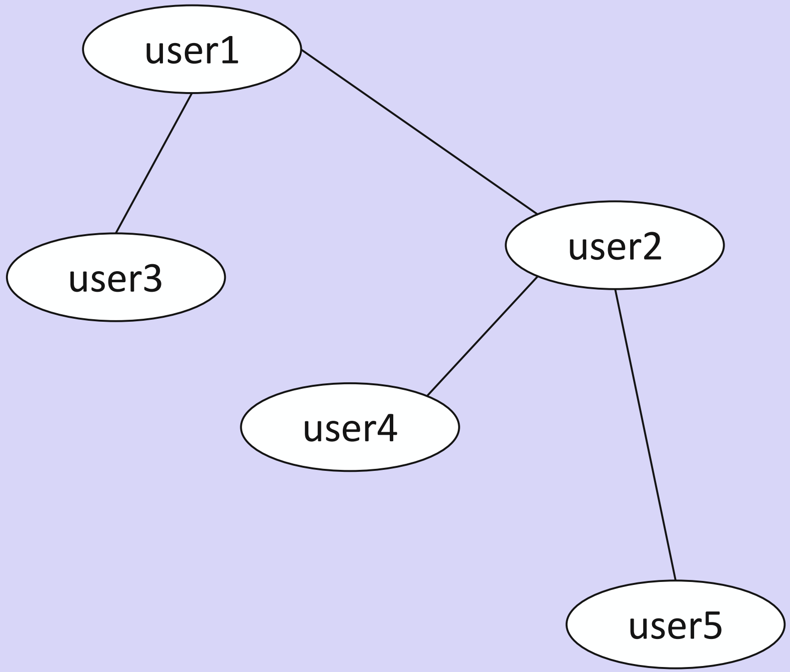
NoSQL – Wide-column stores

- Data are organized by keys and flexible number of columns
- Data is physically stored in rows
- Column families separate data into different lists of columns
- Access via name (key)
- High scalability for write and selective reads
- Very limited query capabilities
- Slow for scans, esp. long scans
- Examples: HBase, Cassandra

Row Key	Address: street	Adress: city	Order: date	Order: quantity	…
Buckenh	Parkstr	Ulm	15.03.15	5	
Jsmith	Hide Park	London	15.03.15	8	
Kmajor			16.03.15	1	
…					

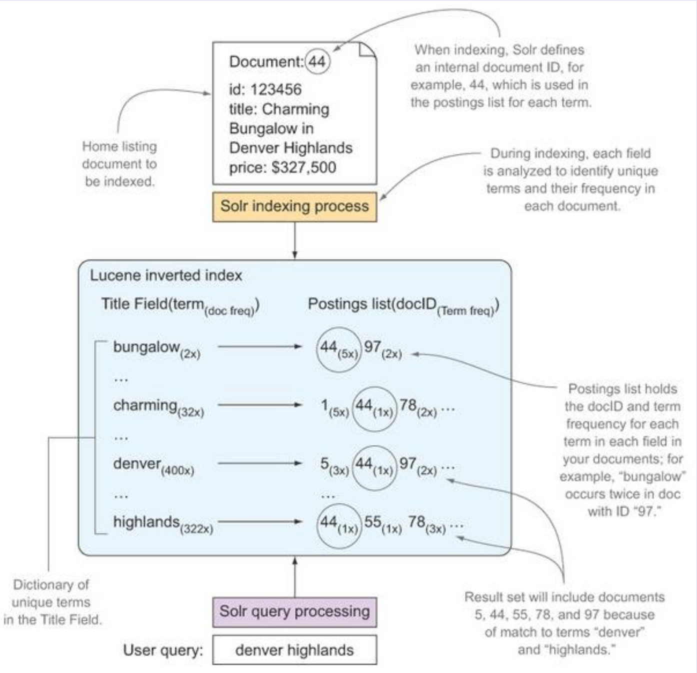
NoSQL – Graph stores

- The physical model contains
  - Edges
  - Vertices
  - Characteristics
- Relationships are of main interest
- Optimized for graph queries (graph traversal), e.g. Social network analysis, Fraud, routes
- Example: Neo4j



NoSQL – Search stores

- The physical model is an inverted index
- Similar to a book index
- Optimized for document/text searches
- Example: Lucene (Java API), Solr, Elastic

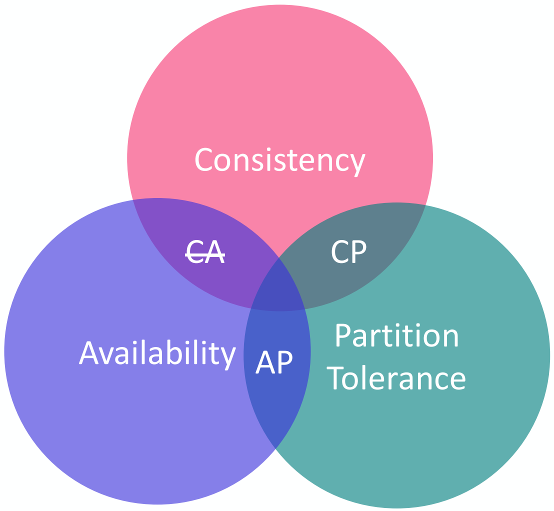


Source: Grainger/Potter: Solr in Action, Manning Publications, 2014, ch. 1

Multi-model databases

- Combine 2 or more physical NoSQL models into one database
- Examples:
  - OrientDB: Key-value store + Document store + Graph store
  - MarkLogic: Document store + Graph store
  - Microsoft CosmosDB: Key-value store + Document store + Graph store

CAP theorem and BASE (Basically Available, Soft state, Eventual consistency)



- With Partition Tolerance set, choose
  - CP: strong consistency
  - AP: eventual consistency
- Actually, the CAP theorem says that it is impossible for a system that guarantees consistency to guarantee 100% availability in the presence of a network partition. So if you can only choose one, it makes sense to choose availability.
- IF AP is chosen / BASE (can often be noticed on twitter with followers):
  - If X is 4 and one node in the cluster is updated with X = 10
  - Now compute: Y = X + 8
    What is the value of Y?
  - Y = 12 or 18 BASE
    (Y = 18 ACID)

Source: http://dbmsmusings.blogspot.com/2018/09/newsql-database-systems-are-failing-to.html and https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html

NewSQL

"***In all such systems, we find developers spend a significant fraction of their time building extremely complex and error-prone mechanisms to cope with eventual consistency and handle data that may be out of date***" (Google white paper)

- Importance of SQL
- Importance of Consistency / ACID
- Examples: VoltDB, MemSQL

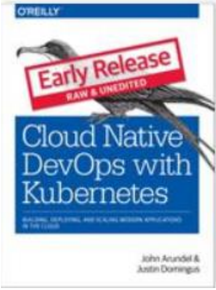
Source: http://db.disi.unitn.eu/pages/VLDBProgram/pdf/industry/p769-shute.pdf

Cloud-native databases

Are **built** to run in the cloud

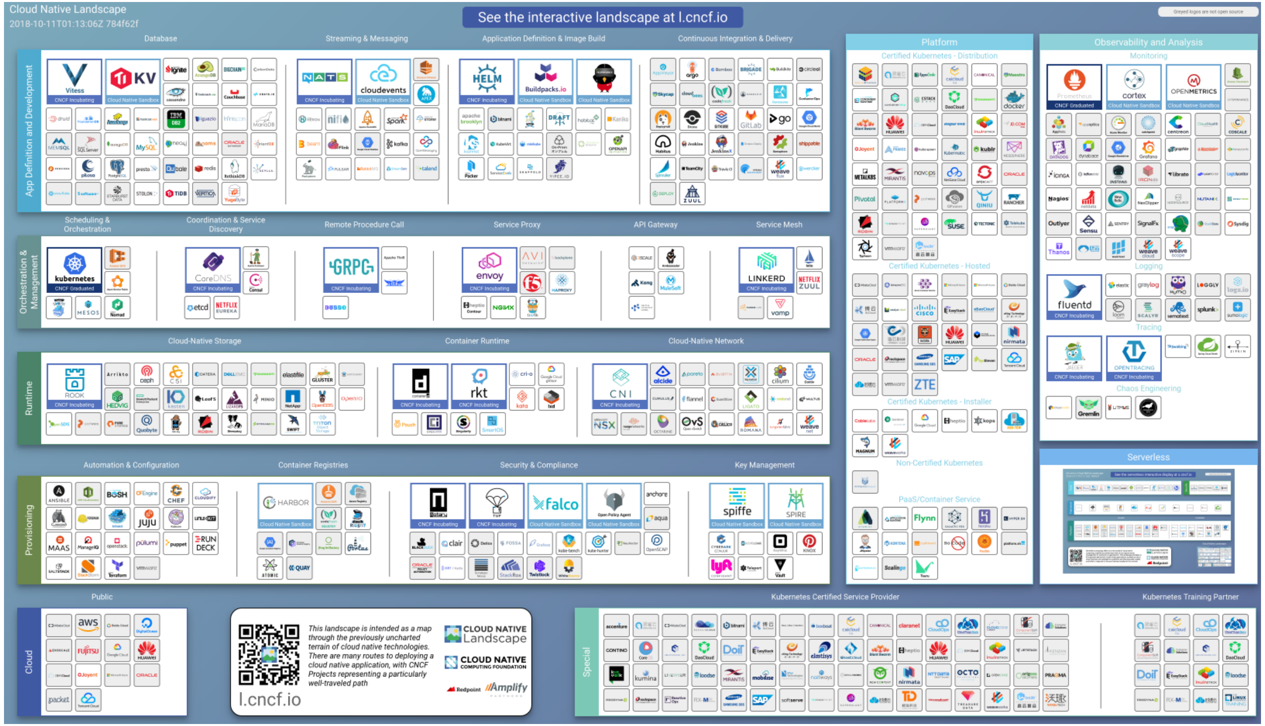
- **Ubiquitous and flexible**: standard container run in any cloud
- **Resilient and scalable**: highly available, redundancy, graceful degradation
- **Dynamic**: rolling upgrades, autonomous
- **Automatable**: everything is code, eg infrastructure
- **Observable**: logging, tracing, metrics
  (Netflix: https://medium.com/netflix-techblog/lessons-from-building-observability-tools-at-netflix-7cfafed6ab17)
- **Distributed**: take advantage of distributed cloud

Source:



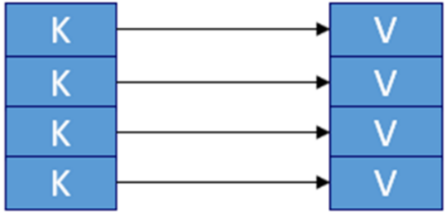
Examples: Google Spanner with the open source variant CockroachDB, YugaByte DB, and FaunaDB

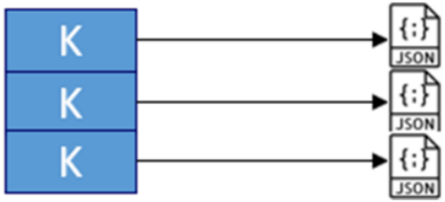
CNCF (cloud native computing foundation) cloud native landscape

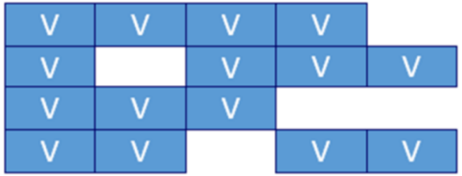


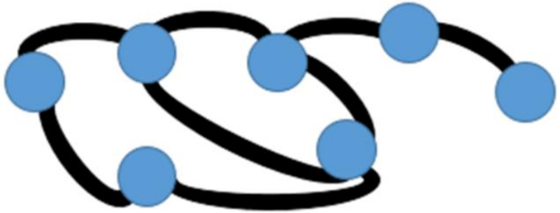
Source: https://github.com/cncf/landscape

Summary: NoSQL, NewSQL, cloud-native Databases









- Graph stores and search stores have relevance for analytical applications
    - Creating knowledge graphs from several data sources
    - Recommender systems, fraud detection
- Other NoSQL / NewSQL stores have their use cases in OLTP applications
    - Row-oriented storage
    - Fast, small reads
    - Many inserts

Exercise

Check the website from some NoSQL or NewSQL vendors

- Which (reference) customers do they have?
- What is the customer’s use case?

Source: http://mattturck.com/wp-content/uploads/2018/07/Matt_Turck_FirstMark_Big_Data_Landscape_2018_Final.png

JSON – schemaless future for Database design?

```
{
  "customer": {
    "firstName": "Jim",
    "lastName": "Bush",
    "interests": [
      "Travel",
      "Sports",
      "Photography"
    ]
  }
}
```

{objects}

Key: value pair

[array]

- Standard in many NoSQL databases, e.g. MongoDB
- But also: ISO SQL Standard 2017, Part 6 with SQL Query-Functions:
  - JSON_EXISTS
  - JSON_VALUE
  - JSON_QUERY
  - JSON_TABLE

JSON (Java Script Object Notation)

- JSON was developed by Douglas Crockford in 2001 to exchange data
- JSON is simpler and more compact compared to XML
- JSON is replacing XML more and more to exchange data
- JSON is used
    - for data exchange (often via RESTful APIs)
    - as a configuration file (for example, Node.js stores metadata in package.json)
    - as a primary storage format in databases like MongoDB
- Standardized by IETF (Internet Engineering Task Force) and ECMA (European Computer Manufacturers Association)

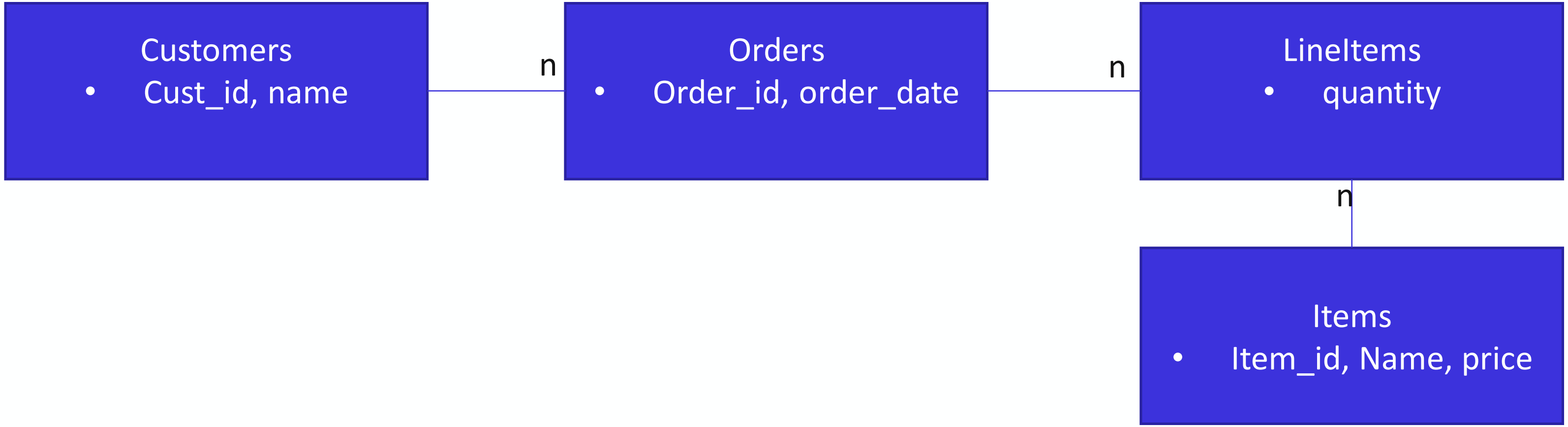
Exercise JSON

Design a physical data model from the JSON document on the next slide

```
{
  "customer_orders": [{
    "id": 1,
    "name": "Eric Cartman",
    "num_orders": 2,
    "orders": [{
      "order_id": 1,
      "date": "2019-10-09T12:35:19",
      "items": [{
        "id": 845,
        "name": "Meteor Impact Survival Kit",
        "quantity": 1,
        "single_item_price": 299,
        "total_price": 299
      }]
    }, {
      "order_id": 2,
      "date": "2019-10-09T12:35:19",
      "items": [{
        "id": 232,
        "name": "Rubber Christmas Tree",
        "quantity": 1,
        "single_item_price": 65,
        "total_price": 65
      }, {
        "id": 429,
        "name": "Air Guitar",
        "quantity": 4,
        "single_item_price": 9.99,
        "total_price": 39.96
      }]
    }]
  }, {
```

```
  }, {
    "id": 2,
    "name": "Kenny McCormick",
    "num_orders": 1,
    "orders": [{
      "order_id": 4,
      "date": "2019-10-09T12:35:19",
      "items": [{
        "id": 345,
        "name": "Border Patrol Costume",
        "quantity": 1,
        "single_item_price": 19.99,
        "total_price": 19.99
      }]
    }]
  }, {
    "id": 3,
    "name": "Kyle Brofloski",
    "num_orders": 1,
    "orders": [{
      "order_id": 3,
      "date": "2019-10-09T12:35:19",
      "items": [{
        "id": 122,
        "name": "Potato Gun",
        "quantity": 1,
        "single_item_price": 29.99,
        "total_price": 29.99
      }]
    }]
  }, {
    "id": 4,
    "name": "Stan Marsh",
    "num_orders": 0
  }]
}
```

Exercise JSON – physical data model



Exercise JSON – physical data model

```
select c.cust_id, c.first || c.last as name
     , o.order_id, o.order_date
     , i.item_id, i.name
     , l.quantity, i.price, i.price * l.quantity as total_price
from   customers c
left join orders o on c.cust_id = o.cust_id
left join lineitems l on l.order_id = o.order_id
left join items i on i.item_id = l.item_id
order by c.cust_id, o.order_id, i.item_id
;
```

Abfrageergebnis ×

SQL | Alle Zeilen abgerufen:6 in 0,009 Sekunden

	CUST_ID	NAME	ORDER_ID	ORDER_DATE	ITEM_ID	NAME_1	QUANTITY	PRICE	TOTAL_PRICE
1	1	EricCartman	1	09.10.19 14:17:00,000000000	845	Meteor Impact Survival Kit	1	299	299
2	1	EricCartman	2	09.10.19 14:17:00,000000000	232	Rubber Christmas Tree	1	65	65
3	1	EricCartman	2	09.10.19 14:17:00,000000000	429	Air Guitar	4	9,99	39,96
4	2	KennyMcCormick	4	09.10.19 14:17:00,000000000	345	Border Patrol Costume	1	19,99	19,99
5	3	KyleBrofloski	3	09.10.19 14:17:00,000000000	122	Potato Gun	1	29,99	29,99
6	4	StanMarsh	(null)	(null)	(null)	(null)	(null)	(null)	(null)

Exercise JSON – physical data model and JSON functions

```sql
SELECT JSON_OBJECT(
          'id' VALUE c.cust_id,
          'name' VALUE (c.first || ' ' || c.last),
          'num_orders' VALUE (
               SELECT COUNT(*)
               FROM orders o
               WHERE o.cust_id = c.cust_id),
          'orders' VALUE (
               SELECT JSON_ARRAYAGG(
                    JSON_OBJECT(
                         'order_id' VALUE o.order_id,
                         'date' VALUE o.order_date,
                         'items' VALUE (
                              SELECT JSON_ARRAYAGG (
                                   JSON_OBJECT(
                                        'id' VALUE l.item_id,
                                        'name' VALUE i.name,
                                        'quantity' VALUE l.quantity,
                                        'single_item_price' VALUE i.price,
                                        'total_price' VALUE (i.price * l.quantity)))
                               FROM lineitems l, items i
                               WHERE l.order_id = o.order_id
                               AND i.item_id = l.item_id)))
          FROM orders o
          WHERE o.cust_id = c.cust_id) ABSENT ON NULL)
FROM customers c;
```

bfrageergebnis ×

SQL | Alle Zeilen abgerufen:4 in 0,012 Sekunden

JSON_OBJECT('ID'VALUEC.CUST_ID,'NAME'VALUE(C.FIRST||''||C.LAST),'NUM_ORDERS'VALUE(SELECTCOUNT(*)FROMORDERSOWHEREO.CUST_ID=C.CUST_ID),'ORDERS'VALUE(SELECTJSON_ARRAYAGG(JSON_OBJECT('ORDER_ID'VALUEO.ORDER_ID,'DATE'VALUEO.ORDER_DATE,'ITEMS'VALUE(SELE

1 {"id":1,"name":"Eric Cartman","num_orders":2,"orders":[{"order_id":1,"date":"2019-10-09T14:17:00","items":[{"id":845,"name":"Meteor Impact Survival Kit","quantity":1,"single_item_price":299,"total_price":299}]
2 {"id":2,"name":"Kenny McCormick","num_orders":1,"orders":[{"order_id":4,"date":"2019-10-09T14:17:00","items":[{"id":345,"name":"Border Patrol Costume","quantity":1,"single_item_price":19.99,"total_price":19.99
3 {"id":3,"name":"Kyle Brofloski","num_orders":1,"orders":[{"order_id":3,"date":"2019-10-09T14:17:00","items":[{"id":122,"name":"Potato Gun","quantity":1,"single_item_price":29.99,"total_price":29.99}]}]}
4 {"id":4,"name":"Stan Marsh","num_orders":0}

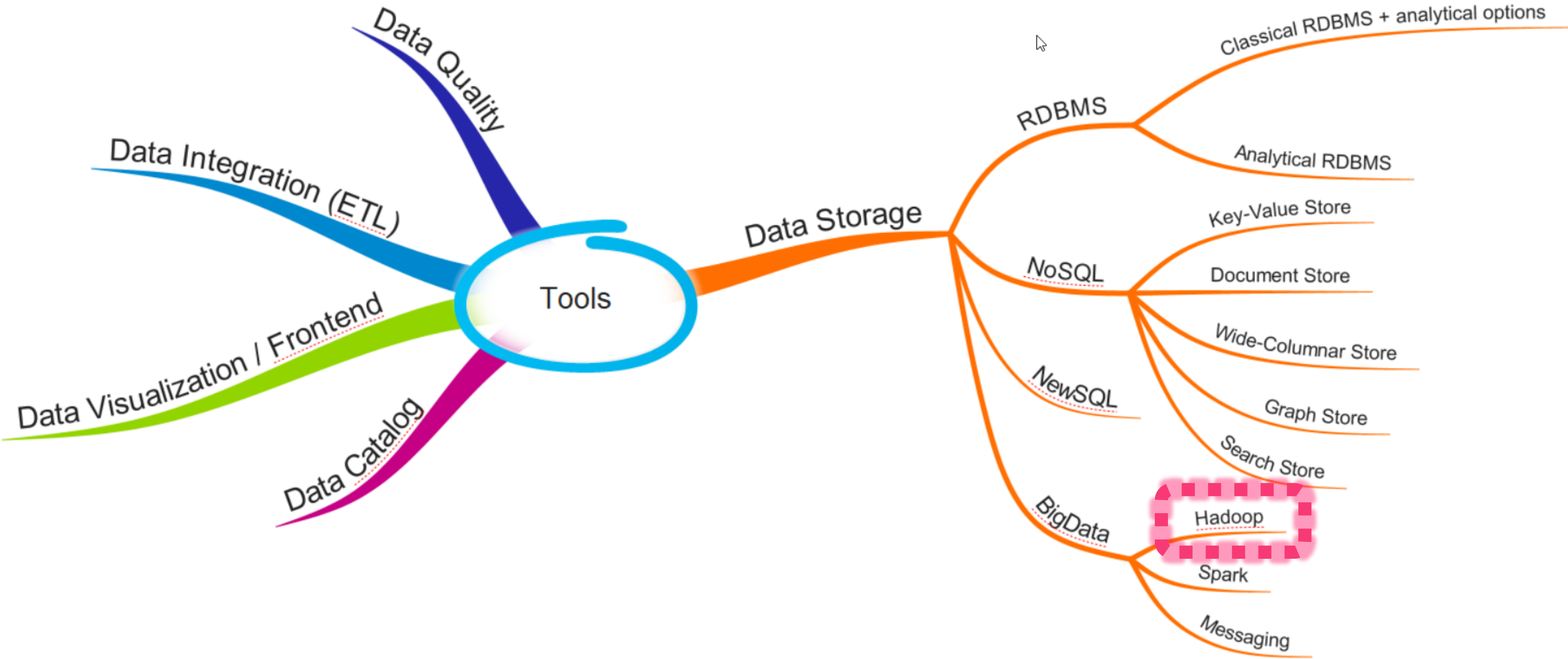
Schema-on-read vs schema-on-write

Schema-on-read	Schema-on-write
No data structure in DB necessary	Table in DB is created first
Schema is applied while reading the data	Schema is applied while writing the data
Fast for writes, slower for reads	Fast for reads, slower for writes
Flexible for programmer as he just copies data in	Flexible for user as he just reads data
High effort for user as he tries to ensure data quality	High effort for programmer as he has to ensure data quality
Wrong data can be stored	Wrong data can not be stored
Security is critical as data is not known	Security can be managed as data is known

Summary: Schema-on-read vs schema-on-write

- Many argue that schema-on-read is great and flexible
  - Data just needs to be copied
  - Data structure can be applied later (schemaless is misnomer)
  - Sensor data has often different formats (new vs old versions in the field) where the format makes sense
- But, there are enormous disadvantages
  - Data quality
  - Same work is done many times
  - Data security is at high risk

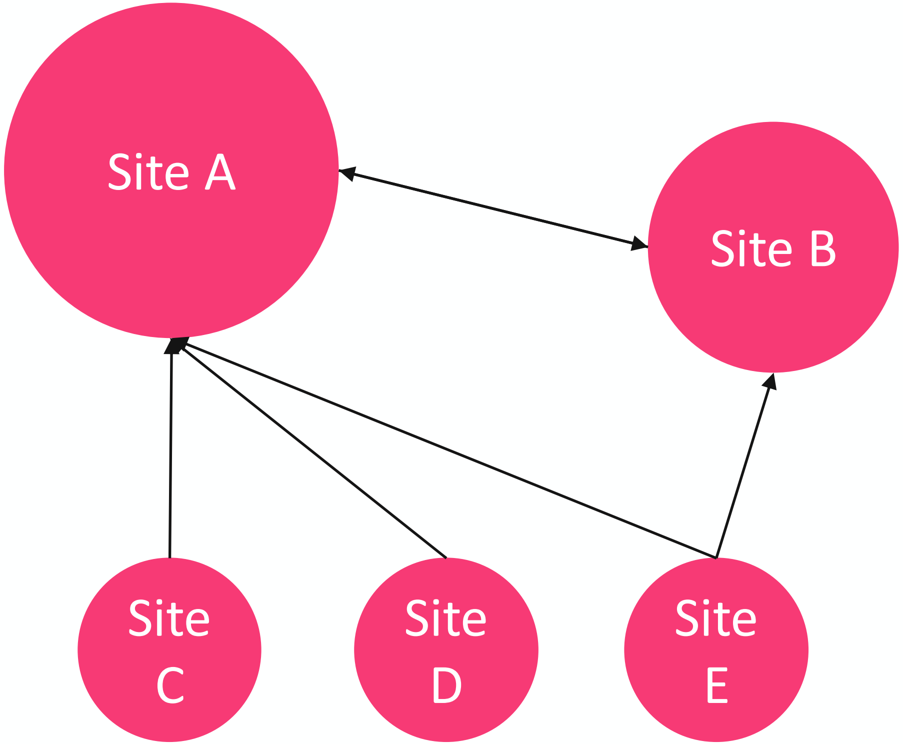
Tools


Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Website popularity: page rank

Webpage

Keywords: DWH, BigData, Microservice, NoSQL, RDBMS, Lambda, Cloud, Python, Spark, Kafka, Streaming, Agile, and many more popular buzzwords



Origin of Hadoop

Pre-Google search engines (Google was founded in 1996):

- Existing search engines simply **indexed on keywords** within webpages
- **Inadequate**, given the sheer number of possible matches for any search term
- The results were primarily weighted by the number of occurrences of the search term within a page, with **no account for usefulness or popularity**

PageRank

- Relevance of a page to be weighted based on the number of links to that page
- Provide a better search outcome than its competitors
- PageRank is a great example of a data-driven algorithm that leverages the “wisdom of the crowd” (collective intelligence)
- High parallel computing power required → Hadoop

Which main components are part of the original Google SW stack?

- **Google File System (GFS)**: a distributed cluster file system that allows all of the disks within the Google data center to be accessed as one massive, distributed, redundant file system.
  http://research.google.com/archive/gfs.html
- **MapReduce**: a distributed processing framework for parallelizing algorithms across large numbers of potentially unreliable servers and being capable of dealing with massive datasets.
  http://research.google.com/archive/MapReduce.html
- **BigTable**: a nonrelational database system that uses the GFS for storage.
  http://research.google.com/archive/bigtable.html

What are the main components in Hadoop?

Hadoop = Open source framework for distributed computations

- Mainly written in Java
- Apache Top-Level project

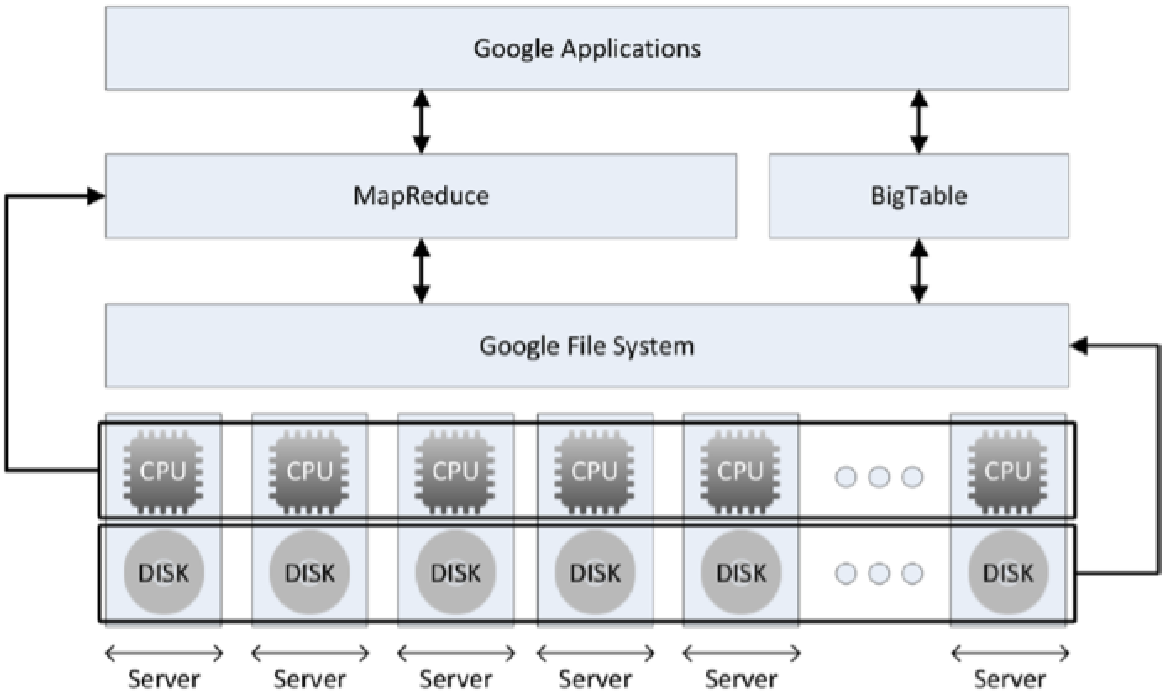
Components:

- **HDFS** (Google: GFS) → clustered filesystem (Hadoop distributed file system)
- **MapReduce** → parallel processing framework
- **HBase** (Google: BigTable) → wide-columnar NoSQL database

Hadoop Pagerank - How did Google use the components?

- GFS / HDFS
  - store webpages
- MapReduce
  - process webpages to identify and weigh incoming links
- BigTable /HBase
  - store results (e.g. from MapReduce) for fast access

Google software architecture



SAN / NAS was rising in the 2000ies but Goggle chose local, directly attached disks

Source: Harrison: Next Generation Databases, Apress 2016

Hadoop timeline

2003: Paper „**Google‘s File System**“ http://research.google.com/archive/gfs.html

2004: Paper „**Google‘s MapReduce**“ http://research.google.com/archive/MapReduce.html

2006: Paper „**Google‘s BigTable**“ http://research.google.com/archive/bigtable.html

2006: Doug Cutting implements Hadoop 0.1. after reading above papers

2008: Yahoo! Uses Hadoop as it solves their search engine scalability issues

2010: Facebook, LinkedIn, eBay use Hadoop

2012: Hadoop 1.0 released

2013: Hadoop 2.2 („aka Hadoop 2.0“) released

2017: Hadoop 3.0 released

HDFS architecture

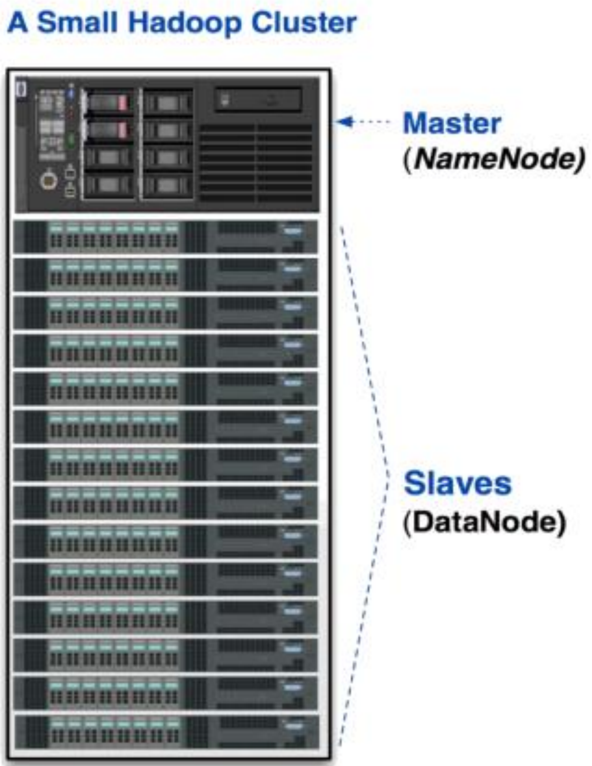
Hadoop has a master/slave architecture

HDFS master daemon: Name Node

- Manages namespace (file to block mappings) and metadata (block to machine mappings)
- Monitors slave nodes

HDFS slave daemon: Data Node

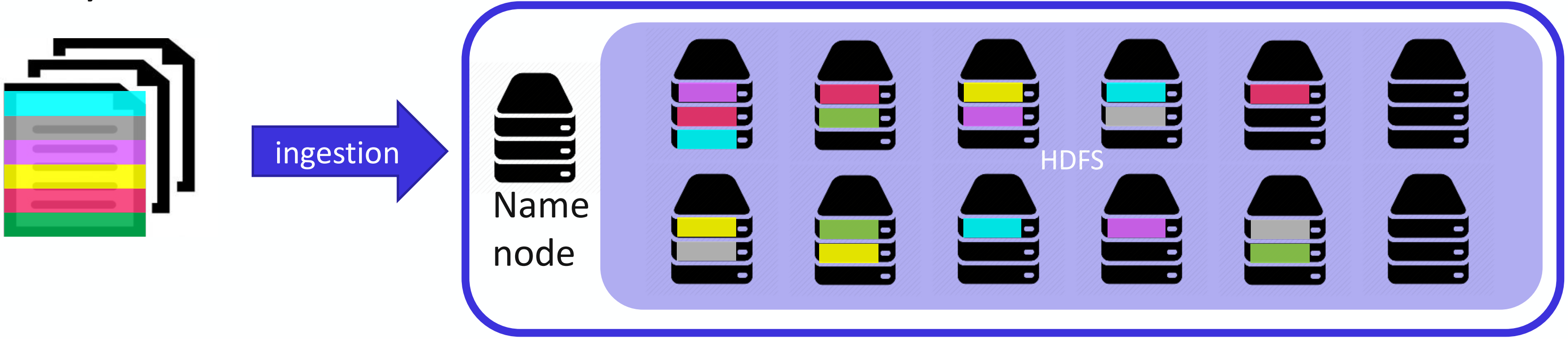
- Reads and writes the actual data



Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

How HDFS works

- Input file is split into blocks (> 64MB)
  - → HDFS is suitable for large files only
  - Splittable compression preferable: LZO, bzip2, ~~gzip~~, ~~snappy~~
- Each block is stored on 3 different disks (default) for fault-tolerance
- Many servers with local disks instead of SAN



Transfering data into HDFS and back

Remember that HDFS is separated from your local filesystem

- Use `hadoop fs -put` to copy local files to HDFS
- Use `hadoop fs -get` to copy HDFS files to local files

Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Some more HDFS commands

Copy file input.txt from the local disk to the user's home directory in HDFS

```
hadoop fs -put input.txt input.txt
```

- This will copy the file to /user/username/input.txt

Get a directory listing of the HDFS root directory

```
hadoop fs -ls /
```

Delete the file /reports/sales.txt

```
hadoop fs -rm /reports/sales.txt
```

Other command options emulating Posix commands are available

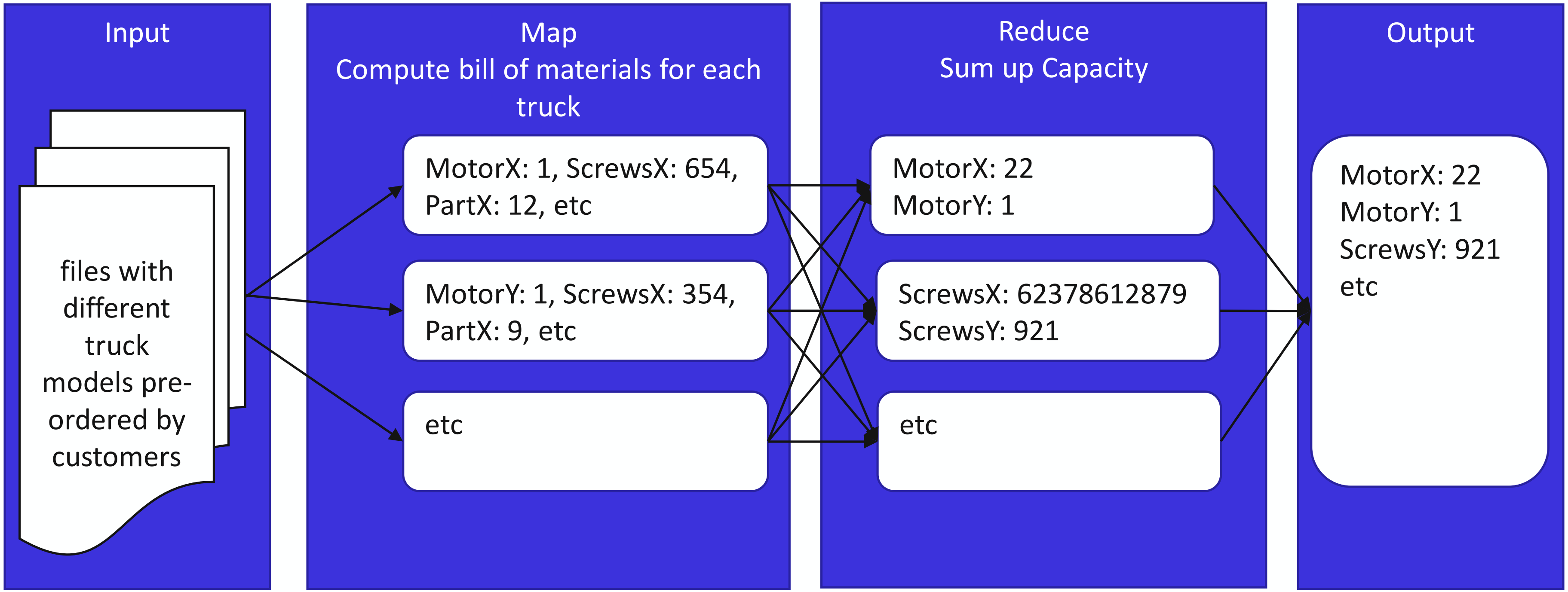
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

HDFS challenges

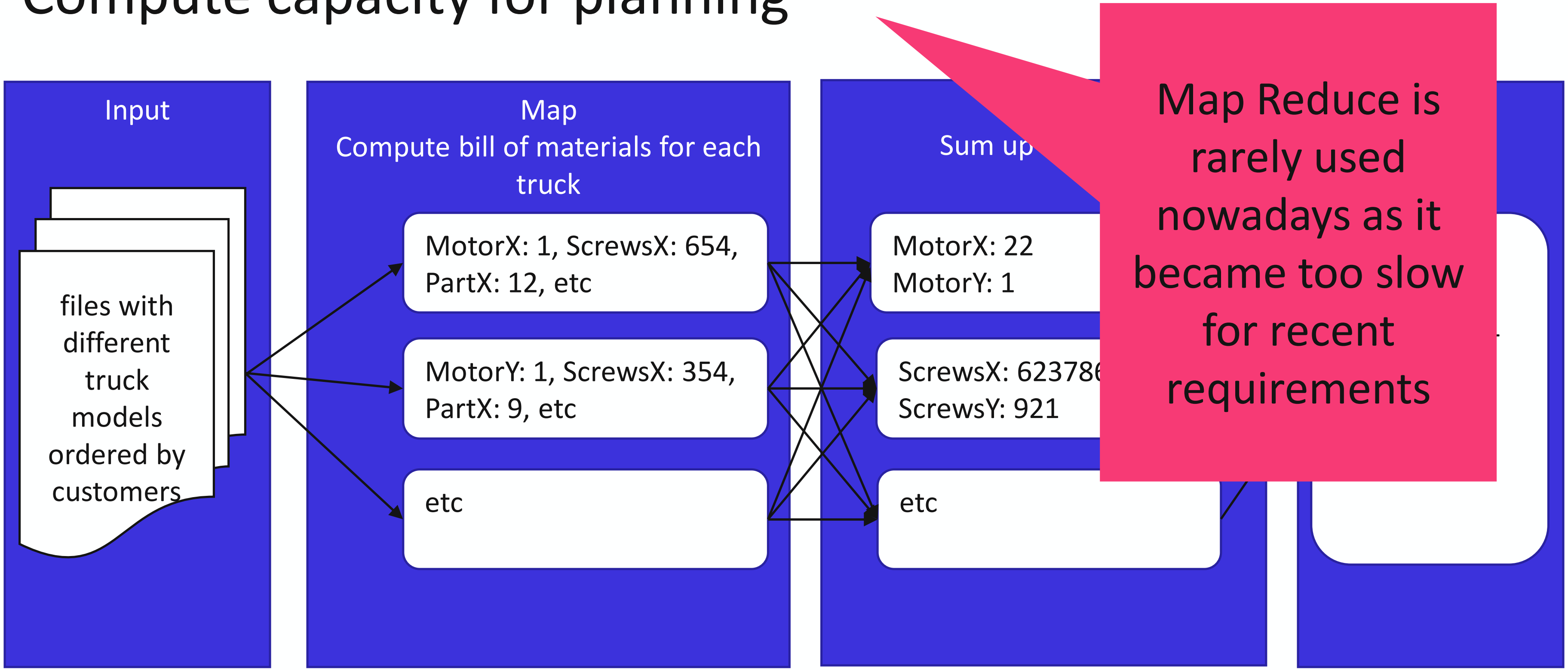
- Optimal for handling millions of large files, rather than billions of small files, because:
  - In pursuit of responsiveness, the NameNode stores all of its file/block information
  - Too many files will cause the NameNode to run out of storage space
  - Too many blocks (if the blocks are small) will also cause the NameNode to run out of space
  - Processing each block requires its own Java Virtual Machine (JVM) and (if you have too many blocks) you begin to see the limits of HDFS scalability
- Not really suited for sensor data: small files
  - Merge of small files into large file or store data in NoSQL stores

Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

MapReduce parallel processing framework
Compute capacity for planning



MapReduce parallel processing framework
Compute capacity for planning



HBase – wide columnar NoSQL database relational data model vs wide columnar model

Name	Site	Visits
Dick	Ebay	507,018
Dick	Google	690,414
Jane	Google	716,426
Dick	Facebook	723,649
Jane	Facebook	643,261
Jane	ILoveLarry.com	856,767
Dick	MadBillFans.com	675,230

Id	Name	Ebay	Google	Facebook	(other columns)	MadBillFans.com
1	Dick	507,018	690,414	723,649	....	675,230

Id	Name	Google	Facebook	(other columns)	ILoveLarry.com
2	Jane	716,426	643,261	....	856,767

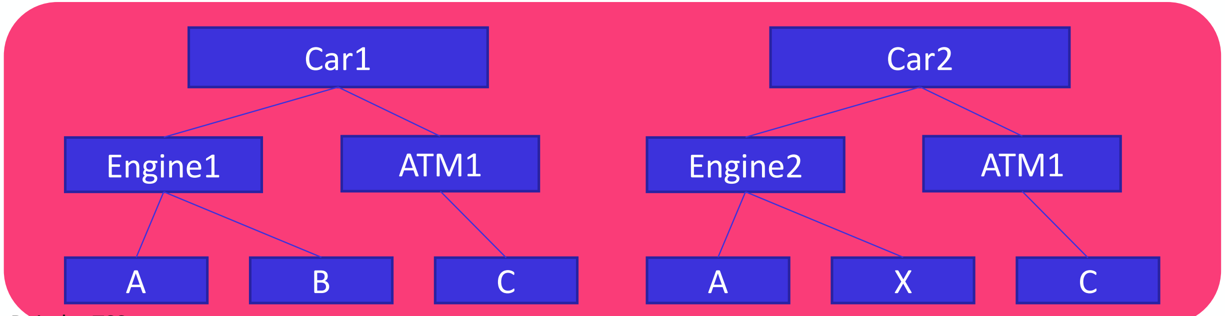
Source: Guy Harrison: Next generation databases, Apress 2015, p.33

Exercise: HBase data model

Sensor1, 17.01.2012 18:00:00, temperature: 15.1°, speed: 3.1km/h

Sensor1, 17.01.2012 18:00:01, temperature: 15.1°

Sensor2, 17.01.2012 18:00:01, temperature: 85.1F, speed: 10.5km/h



Daimler TSS

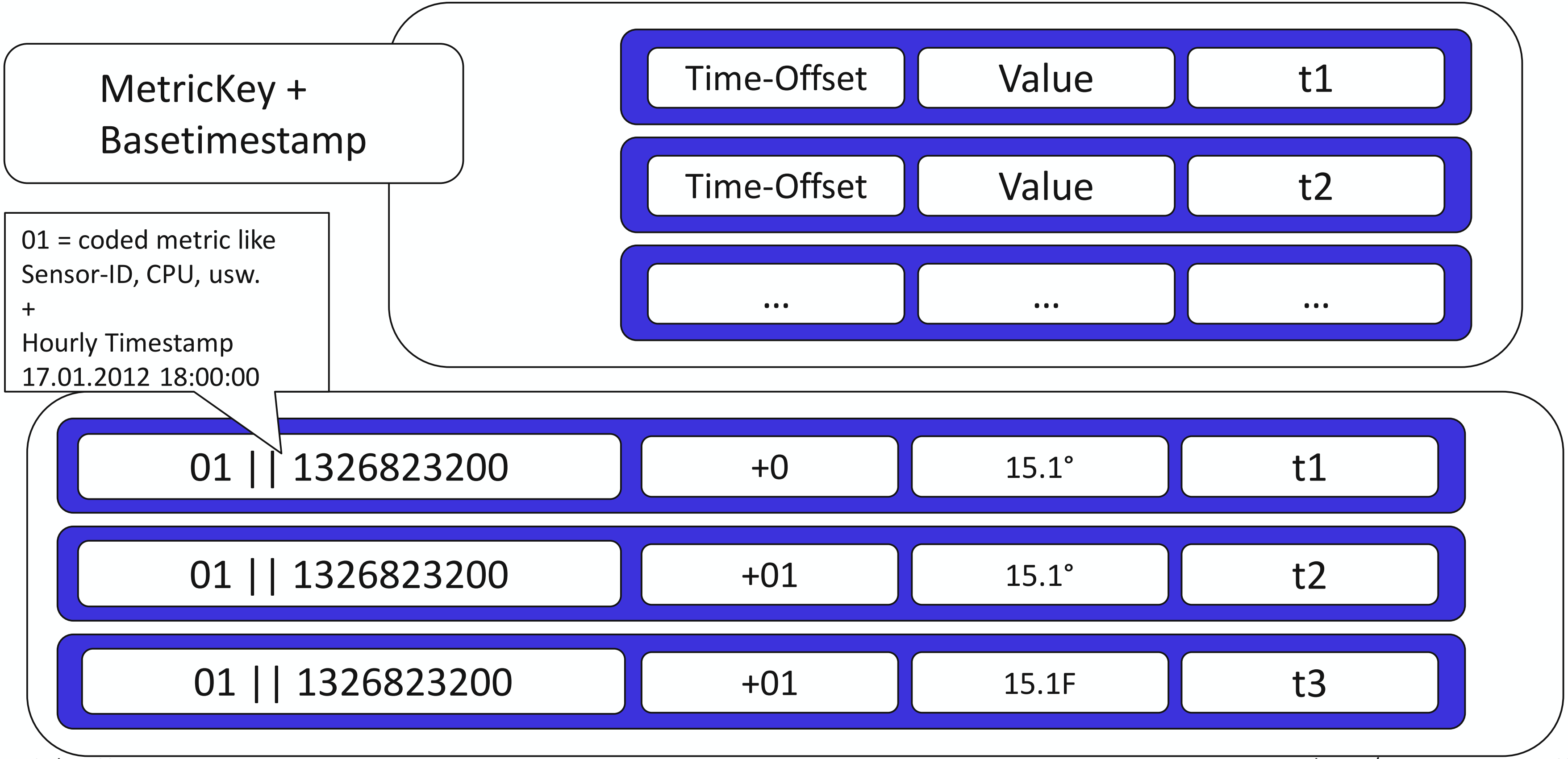
HBase – time series data, e.g. sensor data

Rowkey	Timestamp	Temperature	Speed
Sensor1	17.01.2012 18:00:00	15.1°	3.1km/h
Sensor1	17.01.2012 18:00:01	15.1°	
Sensor2	17.01.2012 18:00:01	85.1F	10.5km/h

Can become slow: Should be searchable

Or better: split measurement and unit into separate fields

HBase – time series data, performance optimized



HBase – bill of materials

Rowkey	Engine1	Engine2	ATM1	A	B	C	X
Car1	Car1		Car1	Engine1	Engine1	ATM1	
Car2		Car2	Car2	Engine2		ATM1	Engine2

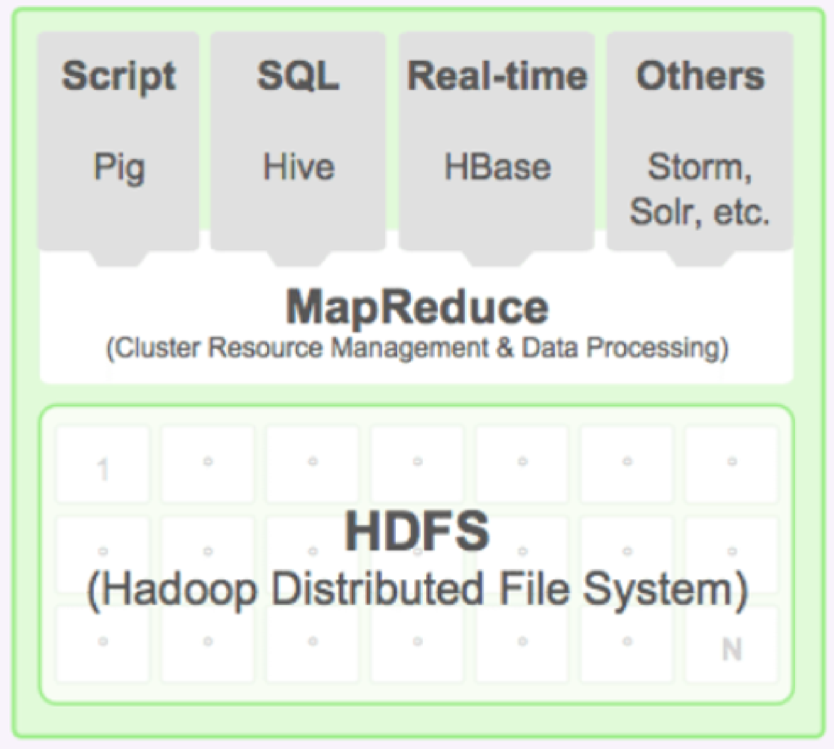
HBase vs HDFS

HDFS / MapReduce (Hadoop)	HBase based on HDFS
Batch	Interactive (ms)
Sequential reads and writes	Random reads and writes
Optimized for full scans	Optimized for selective queries or short scans
append-only	Insert, updates and deletes

How can all these features be possible on HDFS???

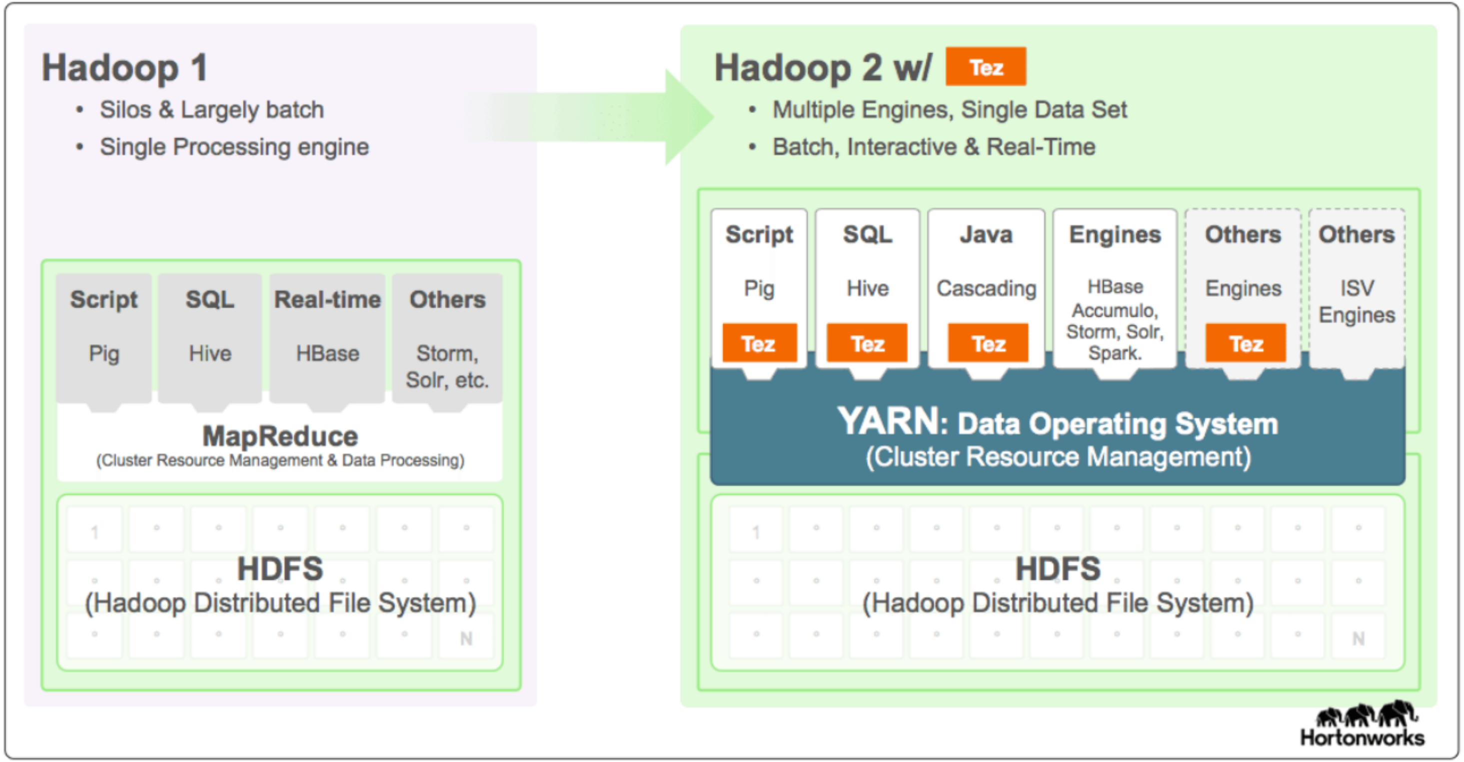
HBase uses e.g. WAL (write-ahead log)

Hadoop V1



Source: https://de.hortonworks.com/apache/Tez/

Hadoop V1 vs Hadoop V2



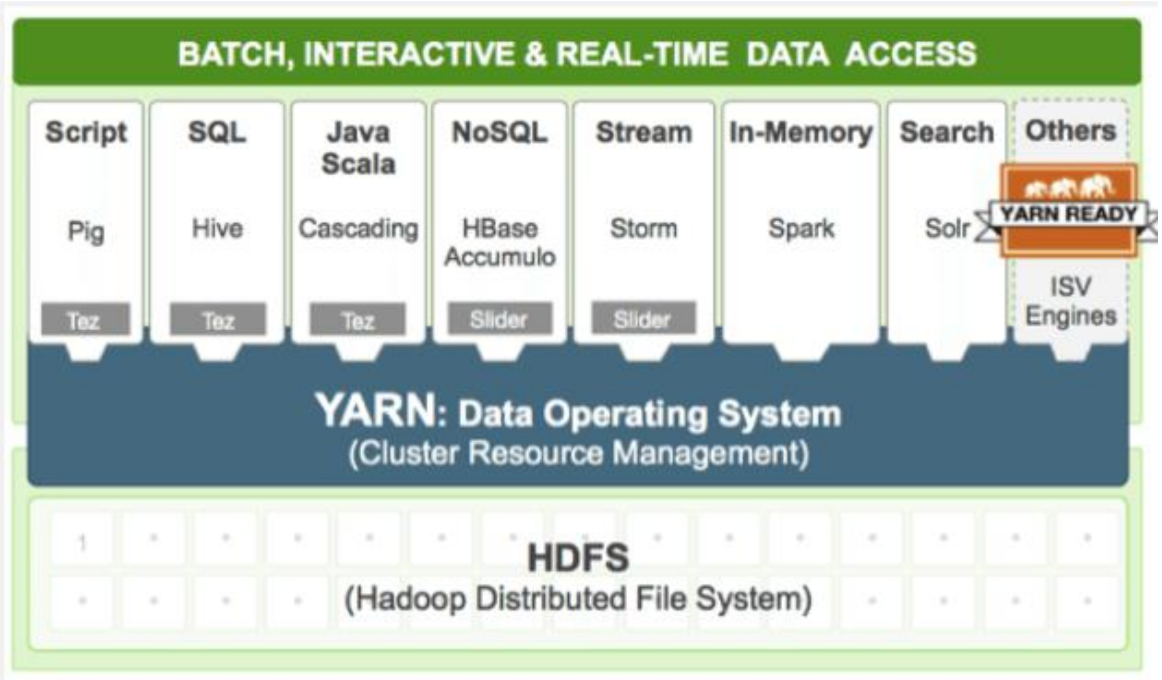
Source: https://de.hortonworks.com/apache/Tez/

Hadoop 1 vs Hadoop2

- Name Node is not single point of failure anymore
  - Manual switch-over
- YARN (Yet Another Resource Negotiator) improves scalability and flexibility by splitting the roles of the Task Tracker into two processes:
  - Much more flexible and scalable compared to MapReduce
  - Resource Manager controls access to the clusters resources (memory, CPU, etc.)
  - Application Manager (one per job) controls task execution within containers
  - **YARN allows to use other engines, not just MapReduce**

YARN replaces Map Reduce and introduces a layer to serve different engines

YARN (yet another resource negotiator)



Configure Hive execution engine

Set hive.execution.engine=

- mr (default)
- Tez
- Spark

Source: https://de.hortonworks.com/apache/YARN/

Which tools exist in the Hadoop ecosystem and what are their function?

Workflow Scheduler

Streaming

Database management systems

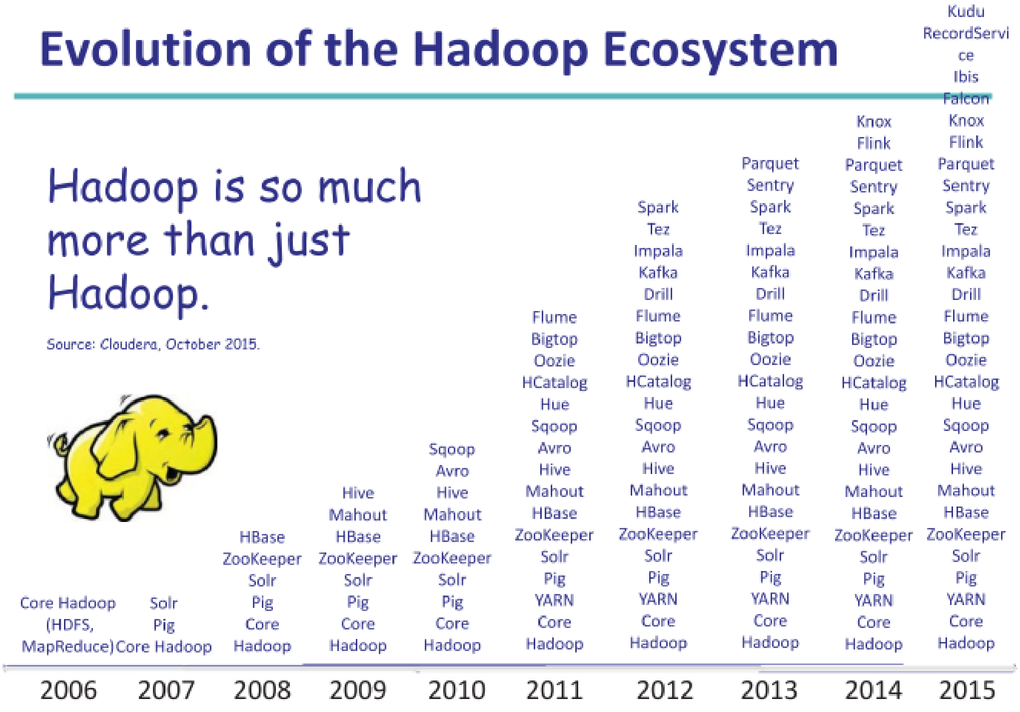
Data Ingestion

Machine Learning

Monitoring

Security

Evolution of the Hadoop ecosystem



Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

Which commercial distributions exist?

Hadoop Apache Project Commercial Support Tracker December 2017	Amazon EMR 5.11	Cloudera CDH 5.13	Google	Hortonworks HDP 2.6.2	MapR MEP 3.0.1
Total supported projects					
Apache HDFS	2.7.3	2.6.0	**2.8.1**	2.7.3	API
Apache Mapreduce	2.7.3	2.6.0	**2.8.1**	2.7.3	2.7.0+
Apache YARN	2.7.3	2.6.0	**2.8.1**	2.7.3	2.7.0+
Apache Hive	**2.3.2**	1.2	2.1.1	2.1.0	2.1.1
Apache Pig	**0.17**	0.12.0	0.16.0	0.16.0	0.16
Apache Spark	**2.2.1**	**2.2**	**2.2.0**	2.1.1	2.1.0
Apache Avro	X	1.7.6	**1.8.2**	1.7.5	1.7.4
Apache Flume	X	1.7.0	1.7.0	1.5.2	1.7
Apache HBase	1.3.1 +S3	1.2	**1.3.1**	1.1.2	1.1.8
Apache Kafka	X	0.11	0.11.0.1	0.10.1.2	0.9 (Streams)
Apache Oozie	4.3.0	4.1.0	4.3.0	4.2.0	4.3.0
Apache Parquet	X	1.51	**1.9.0**	1.8.1	1.8.1
Apache Sqoop	1.4.6	1.4.8	1.99.4	1.4.6	1.4.6
Apache Zookeeper	3.4.10	3.4.5	3.4.6	3.4.6	3.4.5
Hue	4.0.1	4.00	3.11.0	2.6.1	3.12

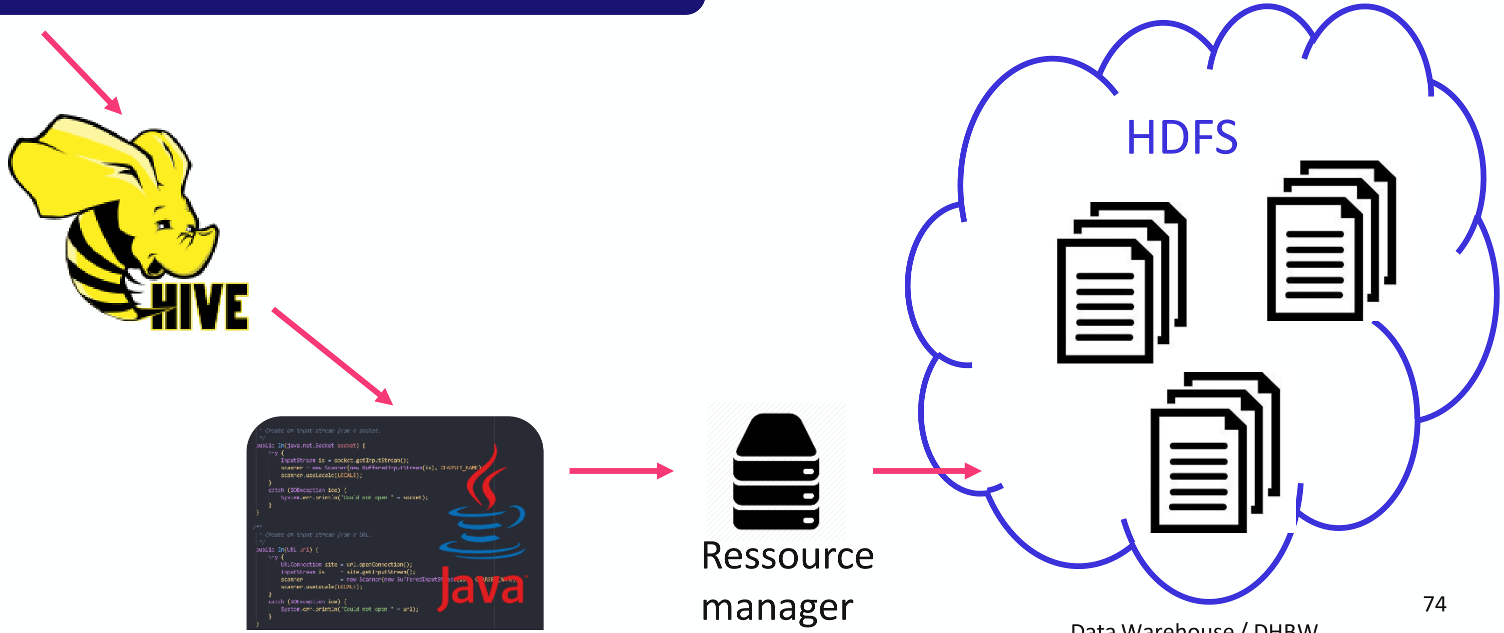
Source: https://blogs.gartner.com/merv-adrian/2017/12/29/december-2017-tracker-wheres-Hadoop/

Other tools

- **SQOOP**, a utility for exchanging data with relational databases, either by importing relational tables into HDFS files or by exporting HDFS files to relational databases.
- **Oozie**, a workflow scheduler that allows complex workflows to be constructed from lower level jobs (for instance, running a Sqoop job prior to a MapReduce application).
- **Hue** / **Ambari**, graphical user interfaces that simplifies Hadoop administrative and development tasks.
- **Knox** / **Ranger / Sentry**, tools for secure data access, identity control, security monitoring, etc.

Hive: SQL-like access on files stored on HDFS initially developed by facebook (2007/2008)

SQL > SELECT sum(income) from calculation group by location



Hive sample with JSON data - View file

```
[root@sandbox ~]# cat Sample-Json-simple.json
{"username":"abc","tweet":"Sun shine is bright.","timestamp": 1366150681 }
{"username":"xyz","tweet":"Moon light is mild .","timestamp": 1366154481 }
[root@sandbox ~]#
```

Hive sample with JSON data - load file into HDFS

```
[root@sandbox ~]# hadoop fs -mkdir  /user/hive-simple-data/
[root@sandbox ~]# hadoop fs -put Sample-Json-simple.json /user/hive-simple-
data/
```

Hive sample with JSON data - create hive table

```
hive>  CREATE EXTERNAL TABLE simple_json_table (
username string,
tweet string,
time1 string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION '/user/hive-simple-data/';
OK
Time taken: 0.433 seconds
```

Hive sample with JSON data - select data from hive table

```
hive> select * from simple_json_table ;
OK
abc      Sun shine is bright.    1366150681
xyz      Moon light is mild .    1366154481
Time taken: 0.146 seconds, Fetched: 2 row(s)
hive>
```

Hive – create table examples
csv, json, avro, parquet, ORC, etc.

```
CREATE EXTERNAL TABLE IF NOT EXISTS Cars (
        Name STRING,
         …
        Origin CHAR(1))
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
location '/user/myDirectory';
```

```
CREATE EXTERNAL TABLE external_parquet
(c1 INT, c2 STRING, c3 TIMESTAMP)
STORED AS PARQUET LOCATION '/user/myDirectory';
```

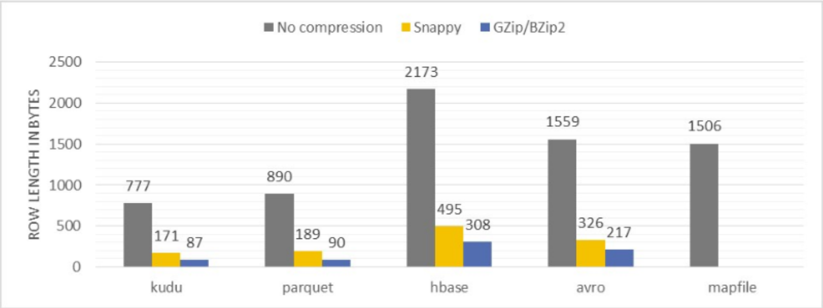
```
CREATE EXTERNAL TABLE my_table STORED AS AVRO LOCATION
'/user/…/my_table_avro/'
TBLPROPERTIES ('avro.schema.url'='HDFS:///user/…/my_table.avsc');
```

Serde – serialization and deserialization

- Different storage formats „schemas“
- **Schema-on-read**:
  - JSON, CSV, HTML, …
  - Text-based formats
- **Schema-on-write**: AVRO, PARQUET, ORC, THRIFT, PROTOCOL BUFFER, …
  - + structural integrity
  - + guarantees on what can and can‘t be stored
  - + prevent corruption
  - Column-oriented data serialization for efficient data analytics: PARQUET, ORC

Storage optimization – performance tests by CERN

SPACE UTILIZATION PER FORMAT



The figure reports on the average row length in bytes for each tested format and compression type

INGESTION RATE PER FORMAT

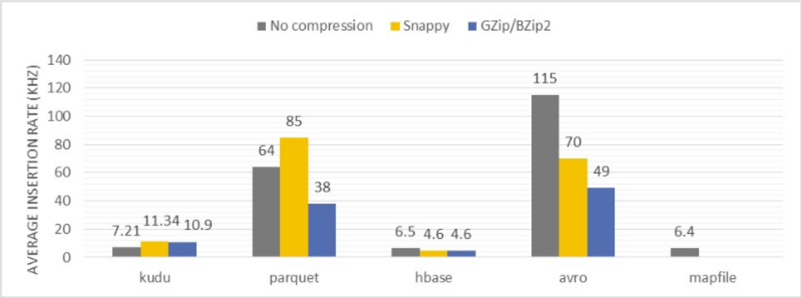


Figure reports on the average ingestion speed (10^3 records/s) per data partition for each tested format and compression type

RANDOM DATA LOOKUP LATENCY PER FORMAT

Figure reports on the average random record lookup latency [in seconds] for each tested format and compression type

DATA SCAN RATE PER FORMAT

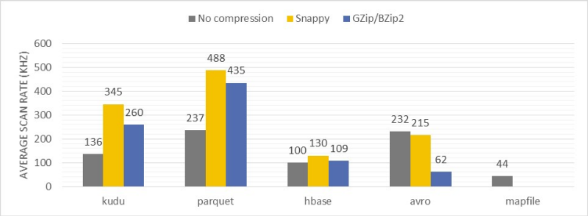


Figure reports on the average scans speed with the same predicate per core [in k records/s] for each tested format and compression type

Source: https://db-blog.web.cern.ch/blog/zbigniew-baranowski/2017-01-performance-comparison-different-file-formats-and-storage-engines

Advantages and disadvantages of hive

- ☺ Higher level query language
- ☺ SQL is widely known
- ☺ Simplifies working with data
- ☺ Better learning curve compared to Map Reduce or other tools like Pig
- ☹ High latency / no real time capability
  - use HBase instead, but HBase is only for very selective queries
- ☹ Updates and deletes are slow (but available since latest releases)

Some known Hadoop cluster

1 PB/s (short-time)

42000 Nodes

300PB (1100 Nodes)

5,3PB (532 Nodes)

Hadoop is

- A distributed file storage
- A mainly batch-oriented processing framework for parallelization
- Flexible and scalable
- Suitable for highly diverse data with low information density
- Fault tolerant and robust
- A long-term storage

Hadoop is not

- A relational database
- A self-service BI tool
- Suitable for transactional data
- Suitable for small data (files)
- Easy for development and operations
- Yet mature
- Suitable for low latency

Gartner on Hadoop deployments

Nick Heudecker
@nheudecker

Folge ich

Thru 2018, 70% of Hadoop deployments will not meet cost savings & revenue generation objectives due to skills & integration challenges. #SPA

Tweet übersetzen

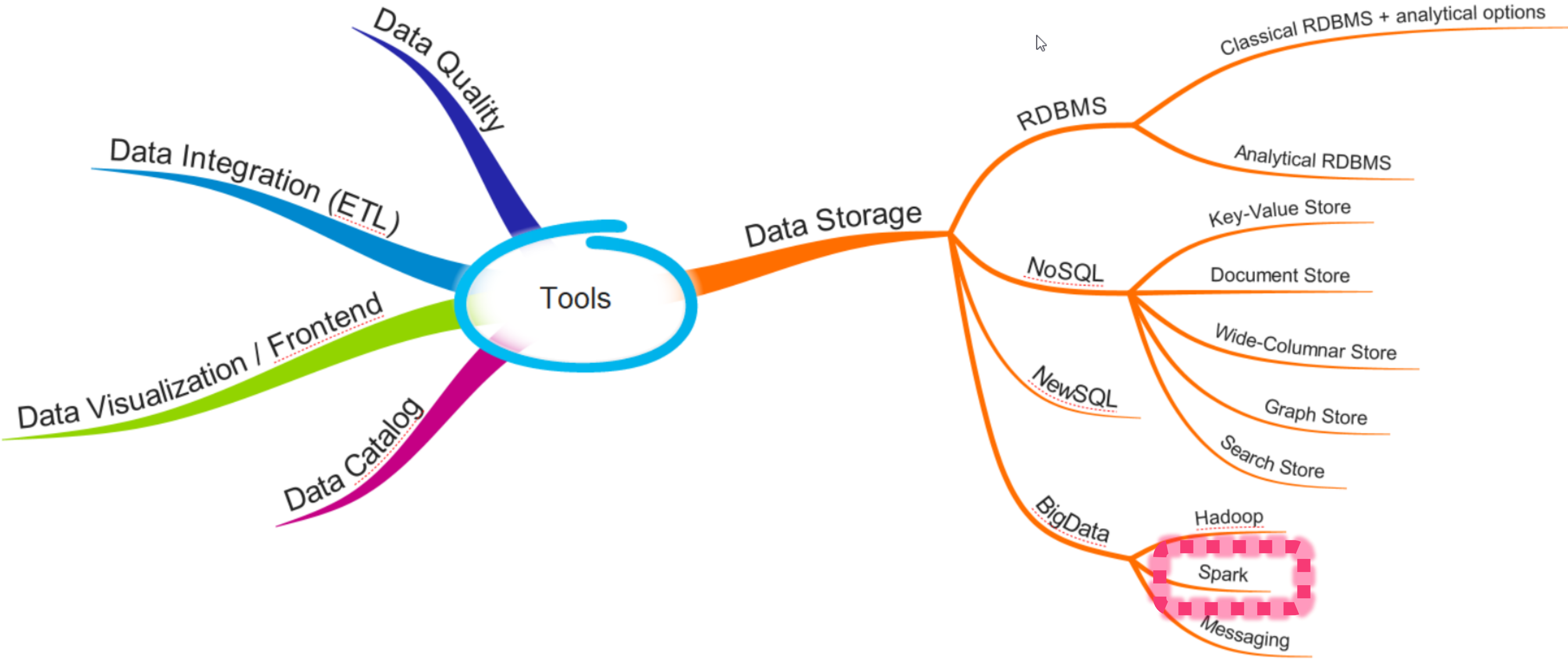
03:48 - 27. Feb. 2015

16 Retweets **11** „Gefällt mir"-Angaben

3 16 11

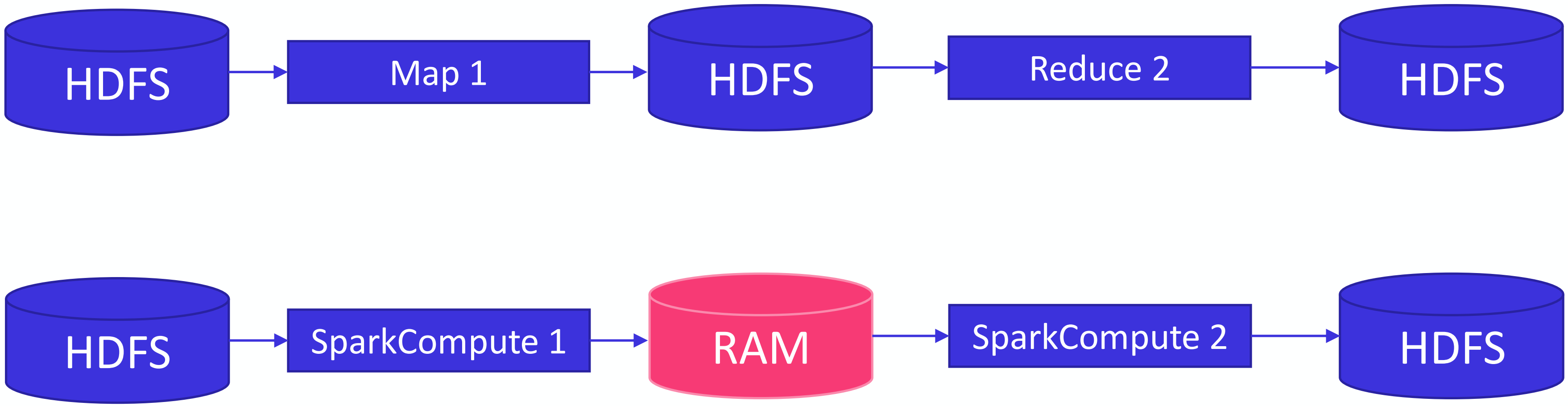
Source: https://twitter.com/nheudecker/status/571139810879893504

Tools



Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Hadoop MapReduce vs Spark



Apache Spark

- Open-source distributed general-purpose cluster-computing framework
- Originally developed at the University of California, Berkeley's AMPLab in 2009
- Stores intermediate results in-memory
- According to Apache: Spark compares to Hadoop to be 100x faster in memory and 10x faster when running on disk
- Databricks is a company founded by the creators of Apache Spark and offers currently the most well-known distribution
- Written in Scala, but APIs available in Scala, Java, Python, R

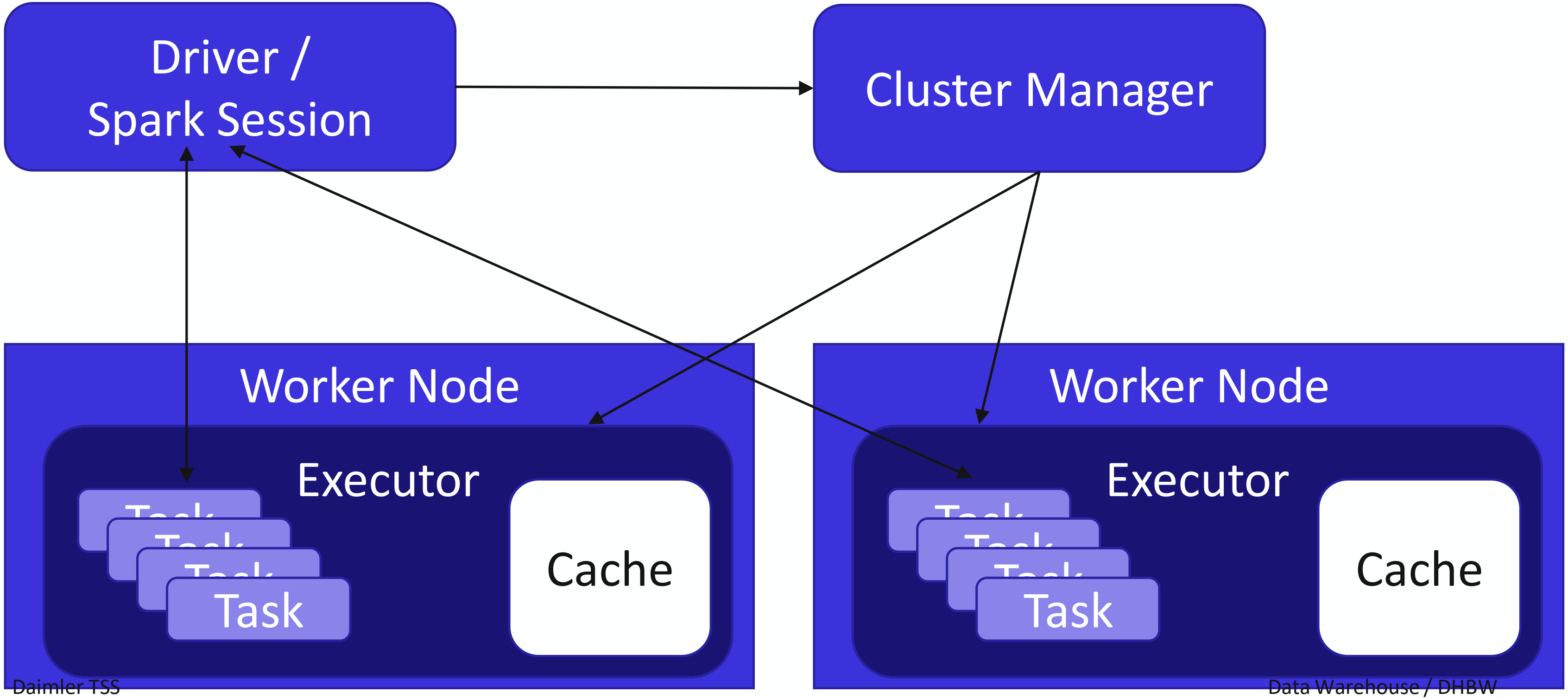
Apache Spark history

2009	Start der Entwicklung am AMPLab der Universität Berkeley durch Matei Zaharia
2013	Gründung der Firma Databricks
Juni 2013	Apache Incubation
Feb. 2014	Apache Top-Level
Mai 2014	Spark 1.0: SparkSQL, MLlib, GraphX, Streaming
März 2015	Spark 1.3: DataFrame API
Jan. 2016	Spark 1.6: Dataset API
Juli 2016	Spark 2.0: überarbeite API für DataFrames und Datasets, Performance Optimierungen, SparkSQL erweitert
Dez. 2016	Spark 2.1: Verbesserungen bei Streaming und Machine Learning
Mai 2017	Spark 2.1.1

Source: https://www.slideshare.net/JensAlbrecht2/einfuehrung-in-apache-Spark

Spark architecture

Spark cluster



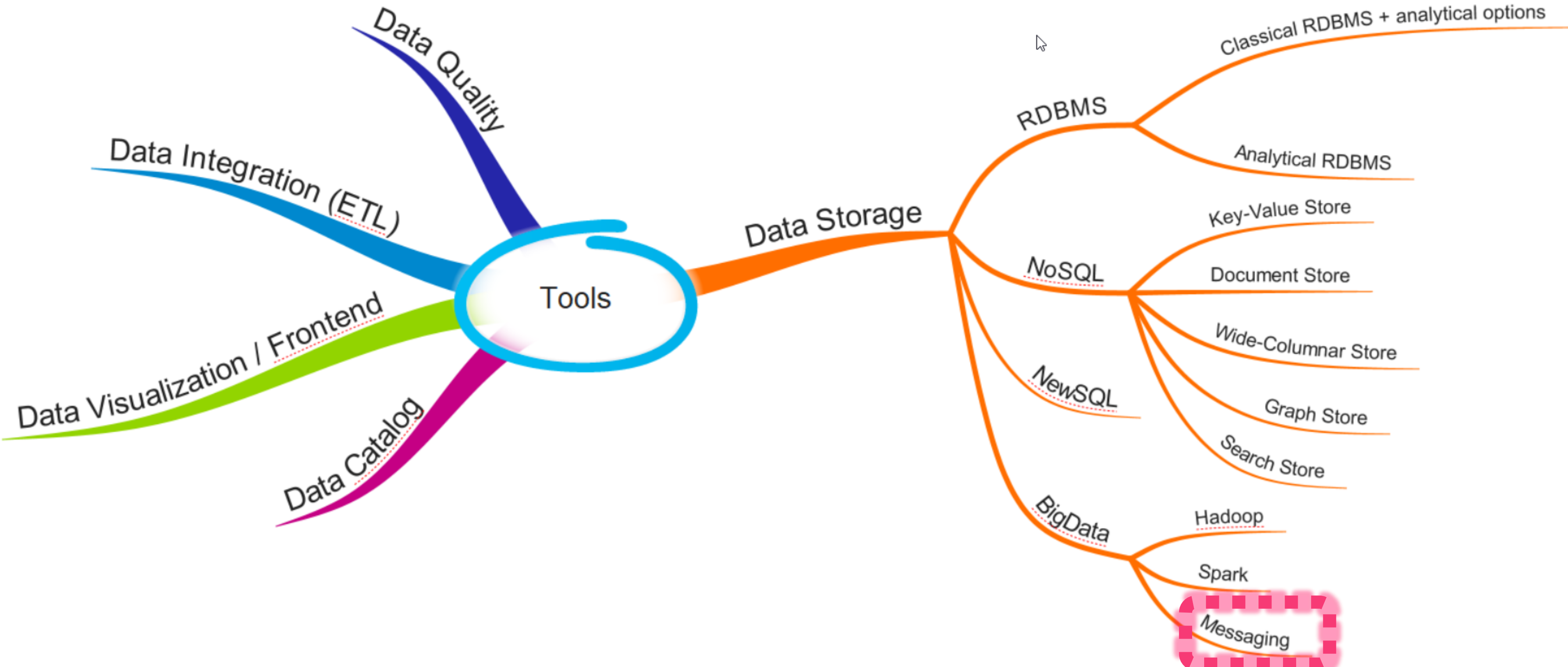
Spark Use cases

- Batch processing
- High performance computing in the cluster
- Microstreaming / Near-real time processing
- Interactive analysis and data discovery
- Machine Learning and Data Science

Spark pros and cons

advantages	disadvantages
• Distributed framework for batch and micro-streaming high-performance in-memory processing • APIs get more stable • Databricks as reliable distributer	• Interactive performance not as good as in-memory databases • Streaming with high latency due to micro batches • Requires high resources in the cluster

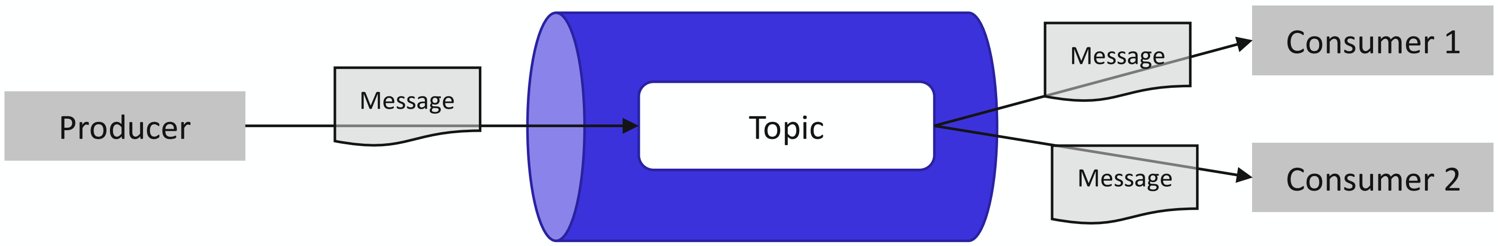
Tools



Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Messaging

- **Producers** publish messages
- **Consumers** subscribe to messages
- **Messaging system** decouples producers and consumers and captures messages (logs, events, etc.) in topics



What is a message? A log!

```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 8
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 268
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 4141
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```

Figure 1-1. An excerpt from an Apache log

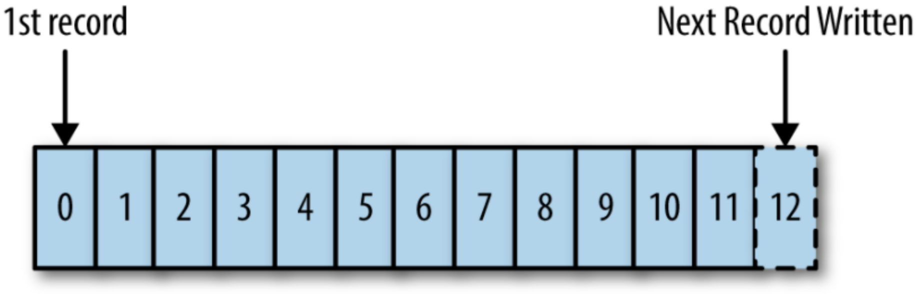


Figure 1-2. A structured log (records are numbered beginning with 0 based on the order in which they are written)

Source: Jay Kreps: I heart logs, O'Reilly 2014

What is a log?

- A bank account's current balance can be built from a complete list of its debits and credits, but the inverse is not true.
- In this way, the log of transactions is the more "fundamental" data structure than the database records storing the results of those transactions.

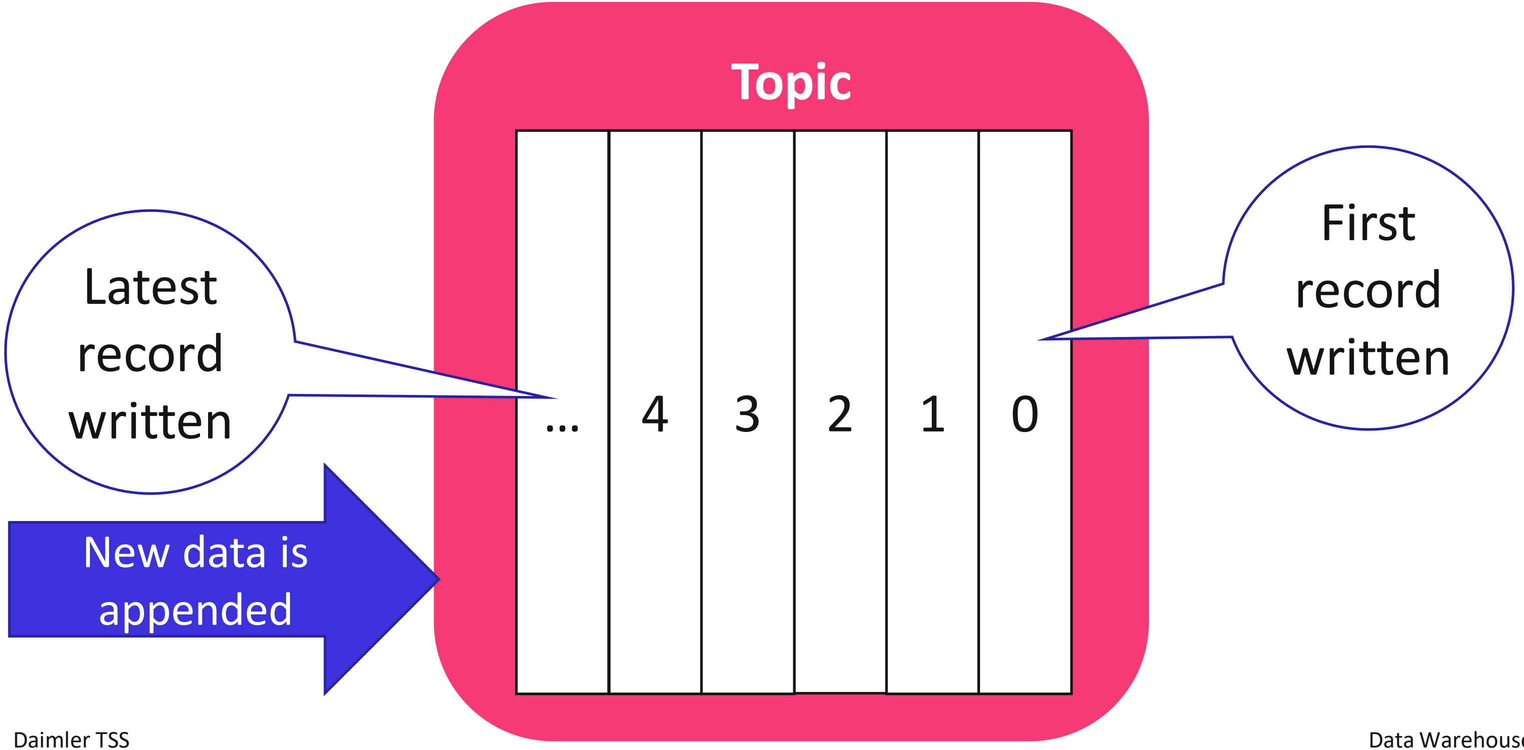
A software application's database is better thought of as a series of time-ordered immutable facts collected since that system was born, instead of as a current snapshot of all data records as of right now.

Source: https://blog.parse.ly/post/1550/kreps-logs/

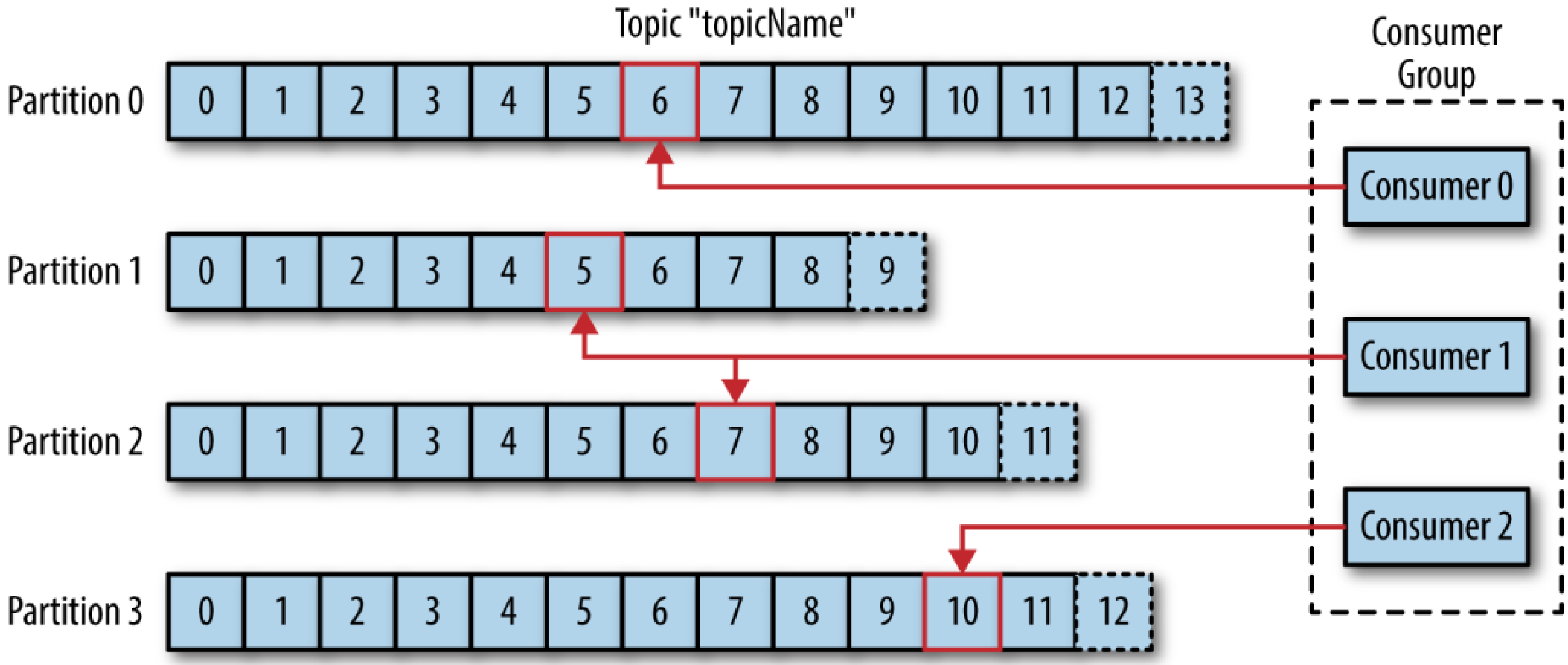
Types of data – all data can be regarded as a series of time-ordered immutable facts (logs)

- Database transactions/data
  - User, products, etc.
- Events
  - Tweets, clicks, impressions, pageviews, etc.
- Application metrics
  - CPU usage, requests, etc.
- Application logs
  - Service calls, errors, etc.

Keep it simple - Immutable Logs

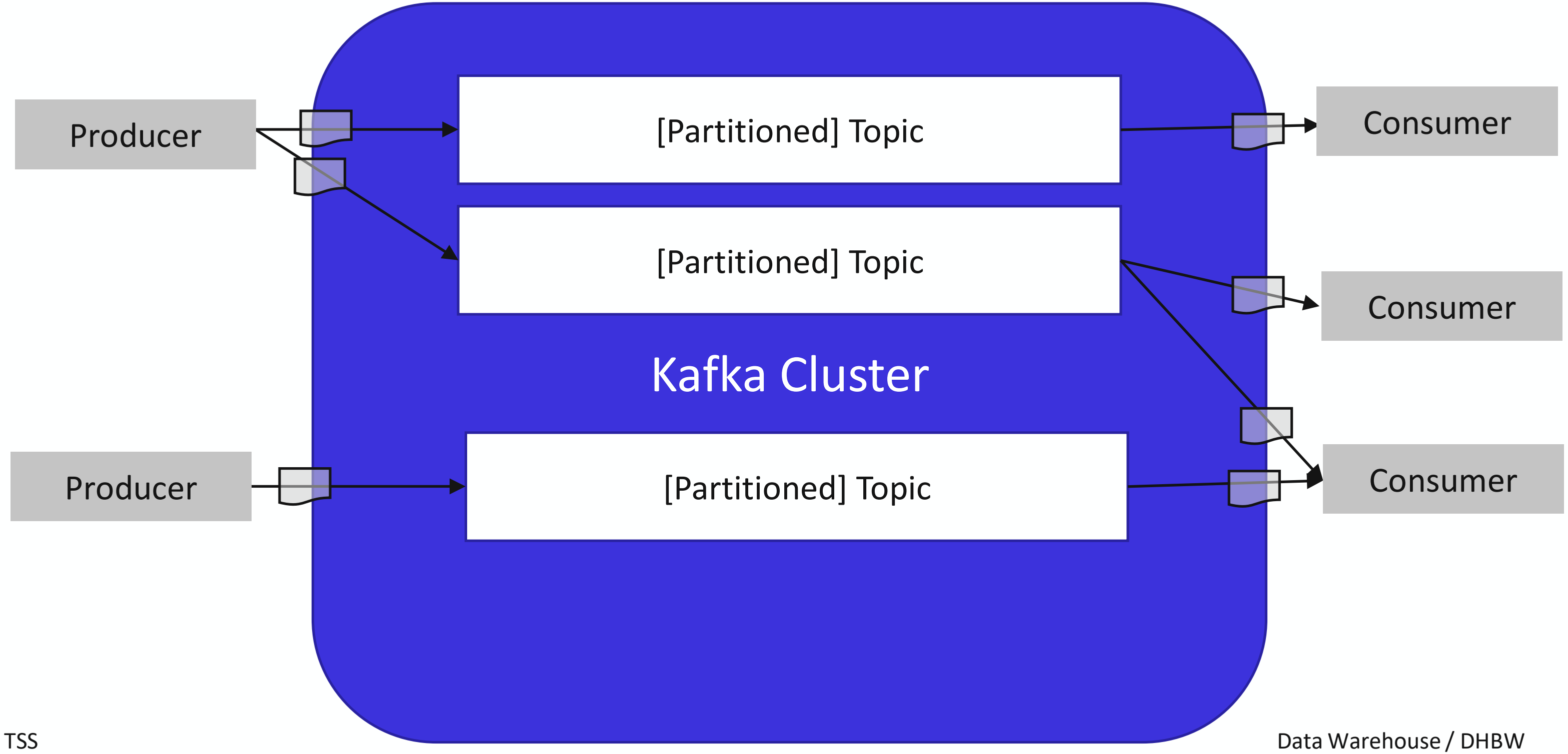


Consumer group reading from a topic

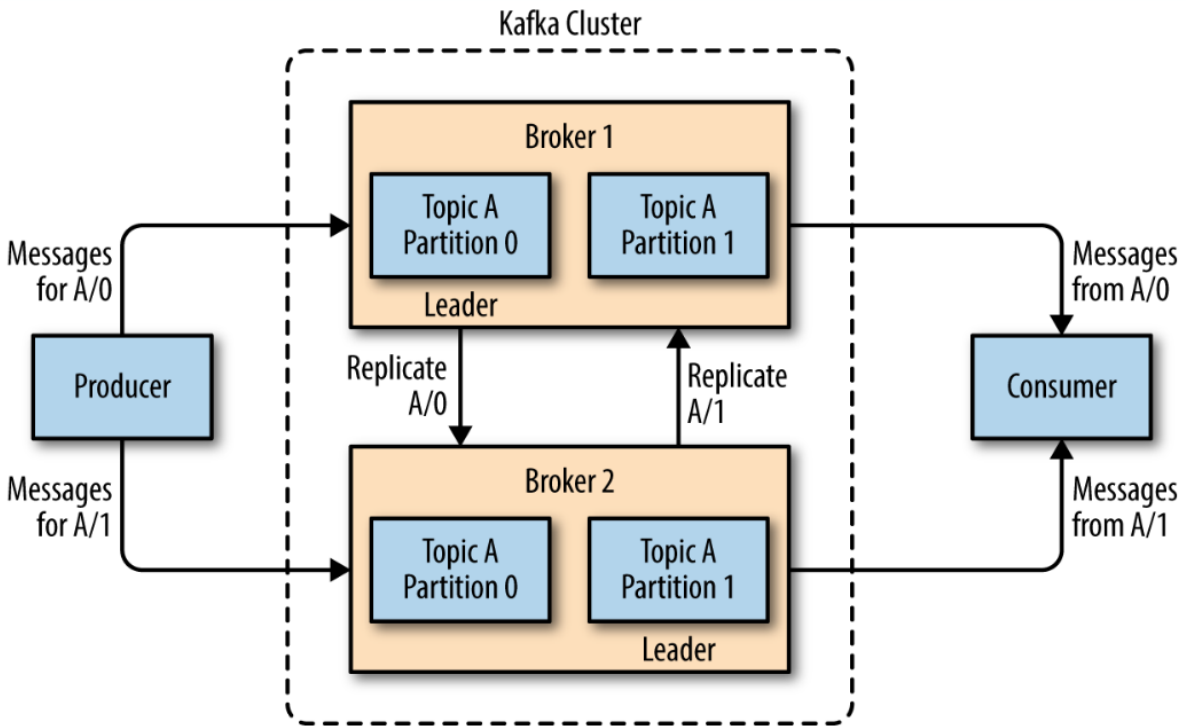


Source: Gwen Shapira, Neha Narkhede, Todd Palino - Kafka: The Definitive Guide, O'Reilly 2017

Kafka architecture

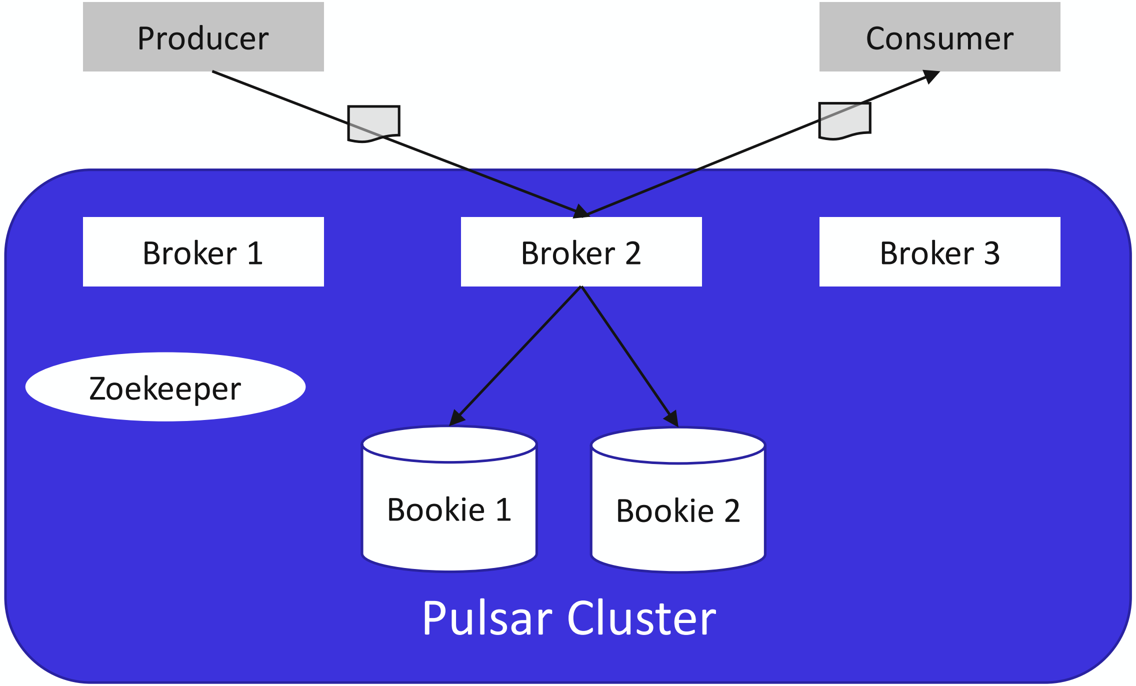


Kafka cluster: Producer – broker - consumer



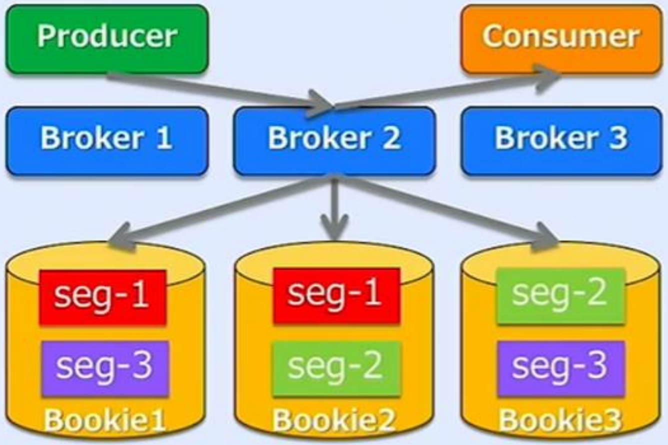
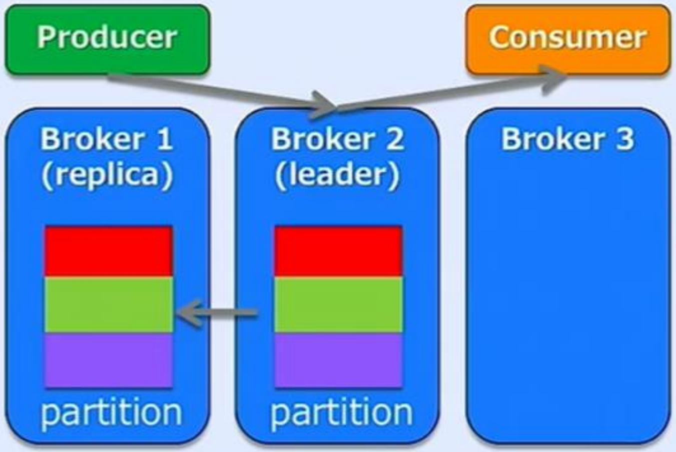
Source: Gwen Shapira, Neha Narkhede, Todd Palino - Kafka: The Definitive Guide, O'Reilly 2017

Apache Pulsar architecture



- **Broker**: stateless serving node
- **Bookie** (BookKeeper): storage node (distributed Write-ahead log)
- **Zoekeeper**: metadata + configuration

Apache Pulsar vs Apache Kafka

	Apache Pulsar	Apache Kafka
Architecture image: • # of partition=1 • # of storage=3 • # of replication=2		

	Apache Pulsar	Apache Kafka
Replication unit	Segment (more granular than Partition)	Partition
Data distribution	Distributed and balanced across all Bookies	Kept in only leader and replica Brokers
Max capacity	Not limited by any single node	Limited by the smallest broker's capacity
Data Rebalancing when scale	Not required	Required
Geo-replication	Built-in	Need to start an extra process: MirrorMaker

Source: Nozomi Kurihara: Apache Pulsar at Yahoo! Japan, O'Reilly 2019 https://learning.oreilly.com/case-studies/scaling/apache-pulsar-at-yahoo-japan/9781491991336-video325421/

Summary: Enterprise-wide data distribution

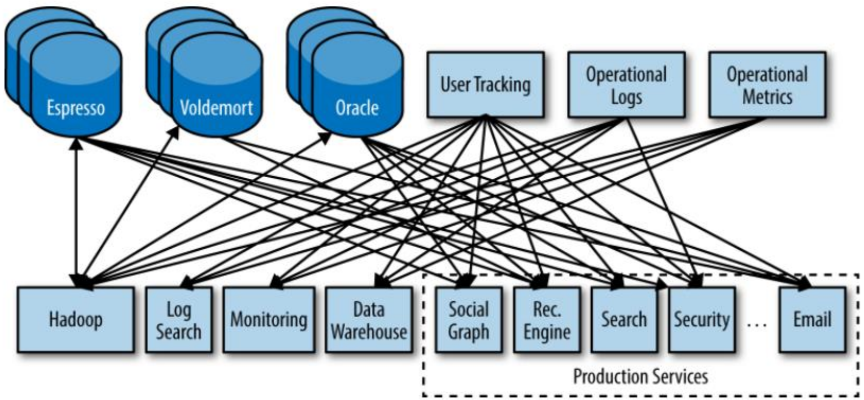


Figure 2-5. A fully connected architecture that has a separate pipeline between each system

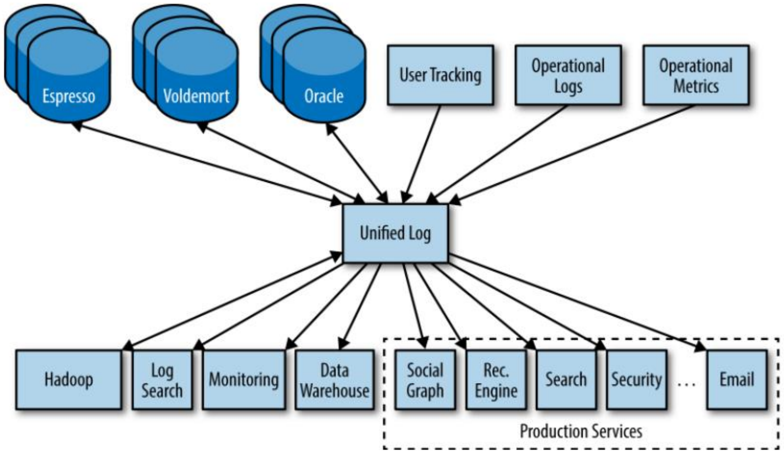
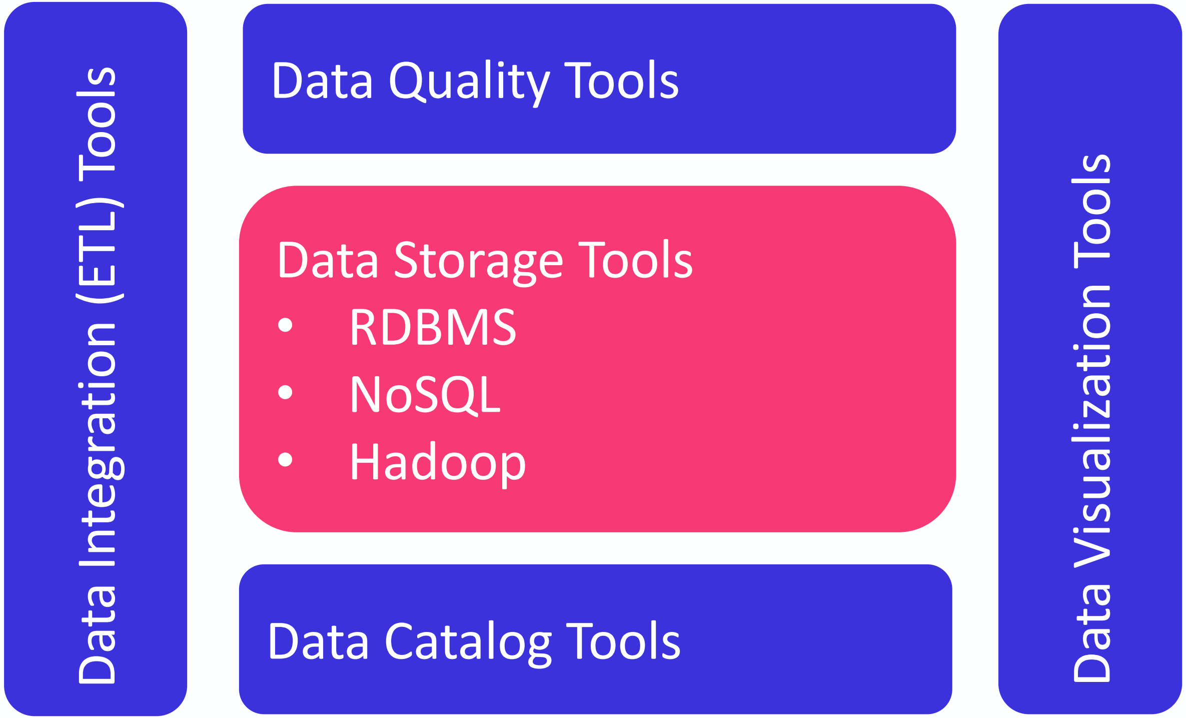


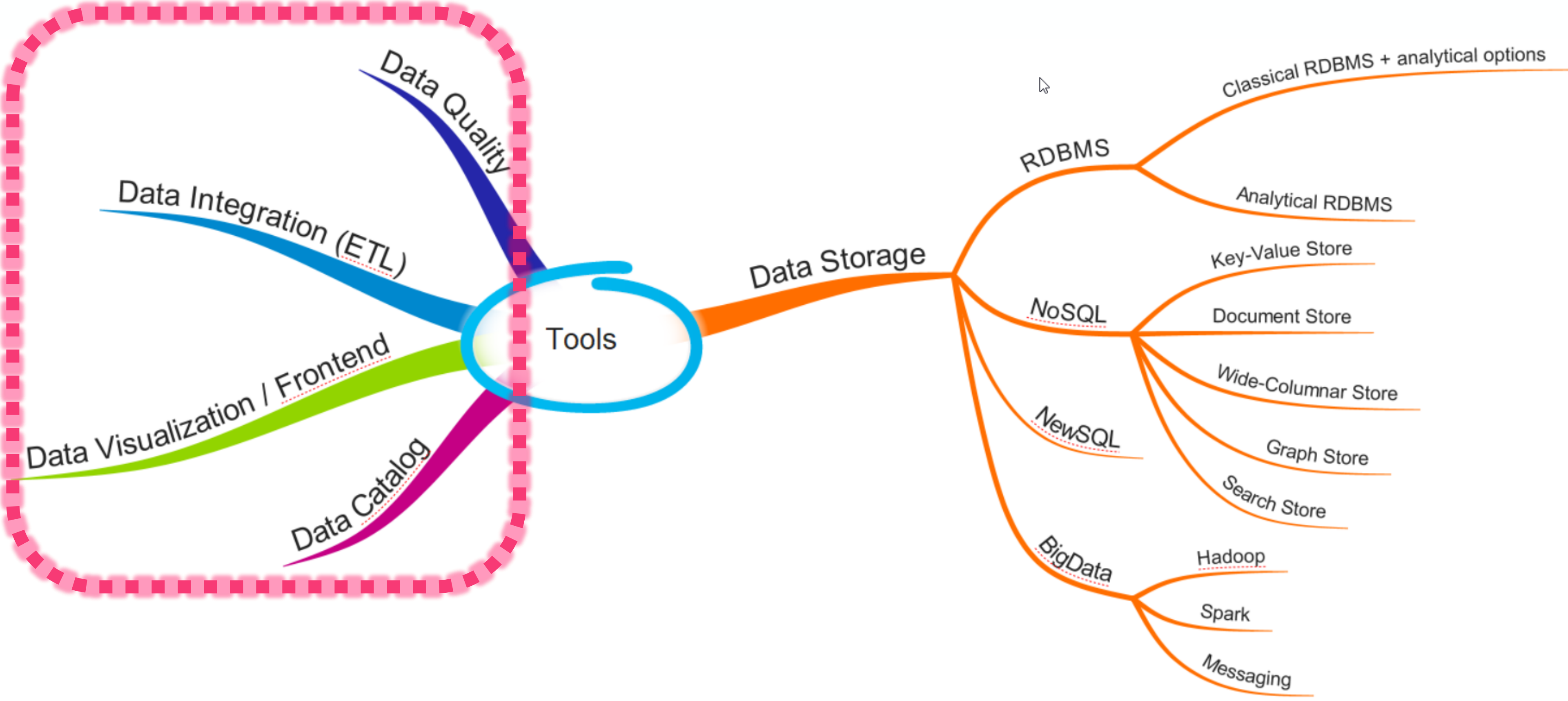
Figure 2-6. An architecture built around a central hub

Source: Jay Kreps: I heart logs, O'Reilly 2014

Tools


Data Integration (ETL) Tools
Data Quality Tools
Data Storage Tools
• RDBMS
• NoSQL
• Hadoop
Data Catalog Tools
Data Visualization Tools

Tools

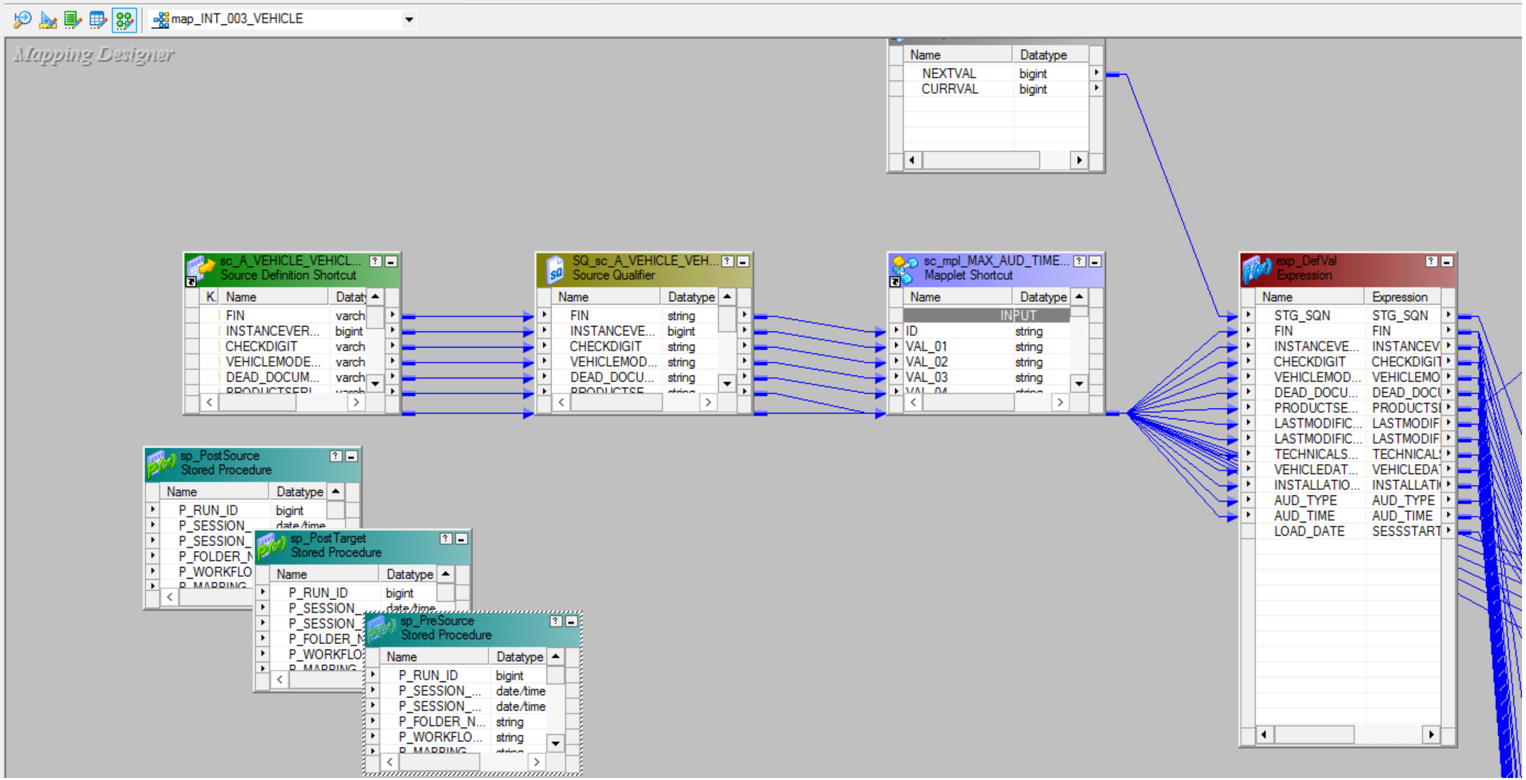


Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Data integration (ETL) and data quality tools

- Many tools available in the market, e.g.
  - Informatica, IBM DataStage, Talend, Pentaho, Wherescape, Oracle Data Integrator, …
- Data quality also handled with Data integration (ETL – Extract, transform, load), but also specialized tools available, e.g.
  - Tolerant Post+Match (address validation and deduplication), Tolerant Bank (bank account validation)
- Data integration (and data quality) will be covered in a separate lecture

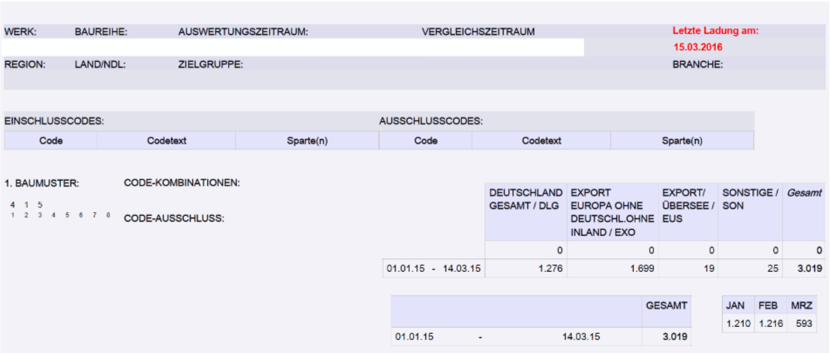
Visual data integration tool



Data visualization tools

- Many tools available in the market, e.g.
  - Tableau, Microsoft PowerBI, Cognos BI, Microstrategy, SAS, Airbnb Superset, ThoughtSpot, Looker, …
  - Excel ☹
- Frontend will be covered in a separate lecture

Textual and visual Reports



WERK:	BAUREIHE:	AUSWERTUNGSZEITRAUM:	VERGLEICHSZEITRAUM	Letzte Ladung am: 15.03.2016
REGION:	LAND/NDL:	ZIELGRUPPE:		BRANCHE:

EINSCHLUSSCODES:			AUSSCHLUSSCODES:		
Code	Codetext	Sparte(n)	Code	Codetext	Sparte(n)

1. BAUMUSTER: CODE-KOMBINATIONEN:

4 1 5
1 2 3 4 5 6 7 8

CODE-AUSSCHLUSS:

	DEUTSCHLAND GESAMT / DLG	EXPORT EUROPA OHNE DEUTSCHL.OHNE INLAND / EXO	EXPORT/ ÜBERSEE / EUS	SONSTIGE / SON	*Gesamt*
	0	0	0	0	0
01.01.15 - 14.03.15	1.276	1.699	19	25	3.019

			GESAMT
01.01.15	-	14.03.15	3.019

JAN	FEB	MRZ
1.210	1.216	593

Data catalog tools

- Recent trend that replaces metadata management, e.g.
  - Alation, Collibra, Waterline, Apache Atlas, …
- Data catalog/metadata management will be covered in a separate lecture

Catalogs are everywhere … Google, Amazon

INVENTORY

Similar idea for data

Cataloging @GOOGLE

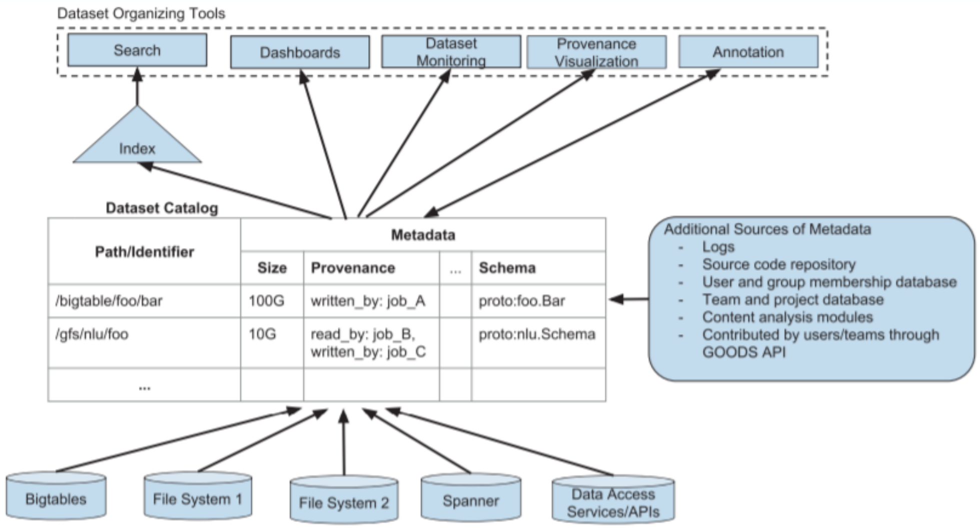


Figure 1: Overview of Google Dataset Search (Goods). The figure shows the Goods dataset catalog that collects metadata about datasets from various storage systems as well as other sources. We also infer metadata by processing additional sources such as logs and information about dataset owners and their projects, by analyzing content of the datasets, and by collecting input from the Goods users. We use the information in the catalog to build tools for search, monitoring, and visualizing flow of data.

Heavy usage of
Automation
and
Machine Learning

Source: https://ai.google/research/pubs/pub45390

Tools:
On-premises or Cloud?

- On-premises
- Cloud
- HW appliances (Engineered Systems)

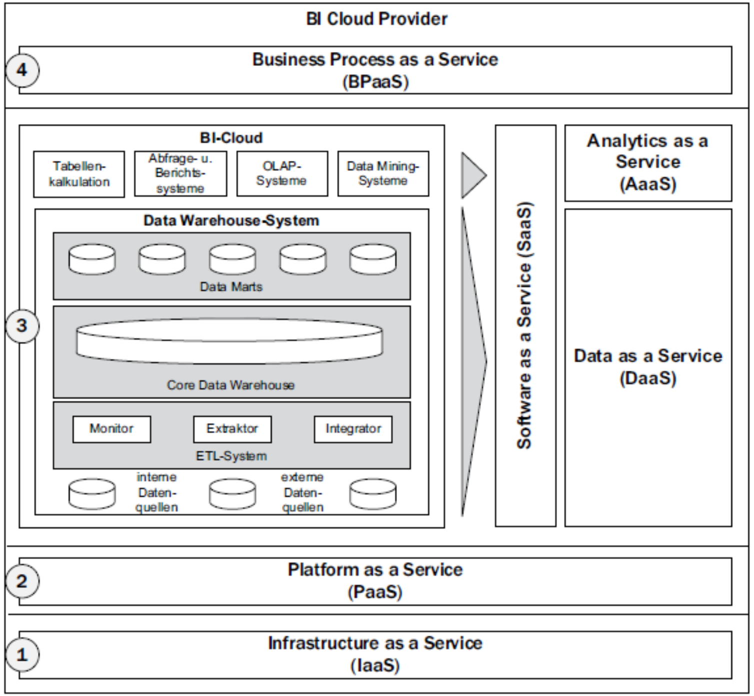
On-Premises vs Cloud

	On-premises	Public Cloud
Deployment	Resources are deployed within an in-house IT infrastructure	Resources are deployed within the infrastructure of a service provider
Control	Full control and ownership	Control is dependant from service provider
Cost	Infrastructure, space, people	Pay for the used resources only
Security	Full control	Security concerns remain through service providers provide many features

Cloud BI

- BI applications (database, ETL tools, Frontend) are hosted in a public cloud, e.g.
    - AWS (Amazon Web Services)
    - Microsoft Azure
    - …
- Many tools nowadays are available in the cloud first
    - Vendors try to force customers to use clouds
- Or even available in the cloud only
    - E.g. Microsoft Power BI

Cloud BI architecture



Source: Lang: Business Intelligence erfolgreich umsetzen, 5.Aufl., p. 185

Cloud BI architecture

- Analytics as a service
  - Provide complete BI (Analytics) SW stack including
    - data storage
    - data integration (ETL)
    - data visualization and/or data modeling (Frontend)
    - Data catalog (meta data management)
- Data as a service
  - Provide quality data for further usage
    - Data marketplace
    - Data monetization

Data Warehouse and Big Data Appliances

Setting up and configuring a data warehouse system is a complex task

- Hardware
    - Servers + Storage + Network
    - Connectivity to source systems
- Software
    - Database management system
    - ETL software
    - Reporting and analytics software
    - …

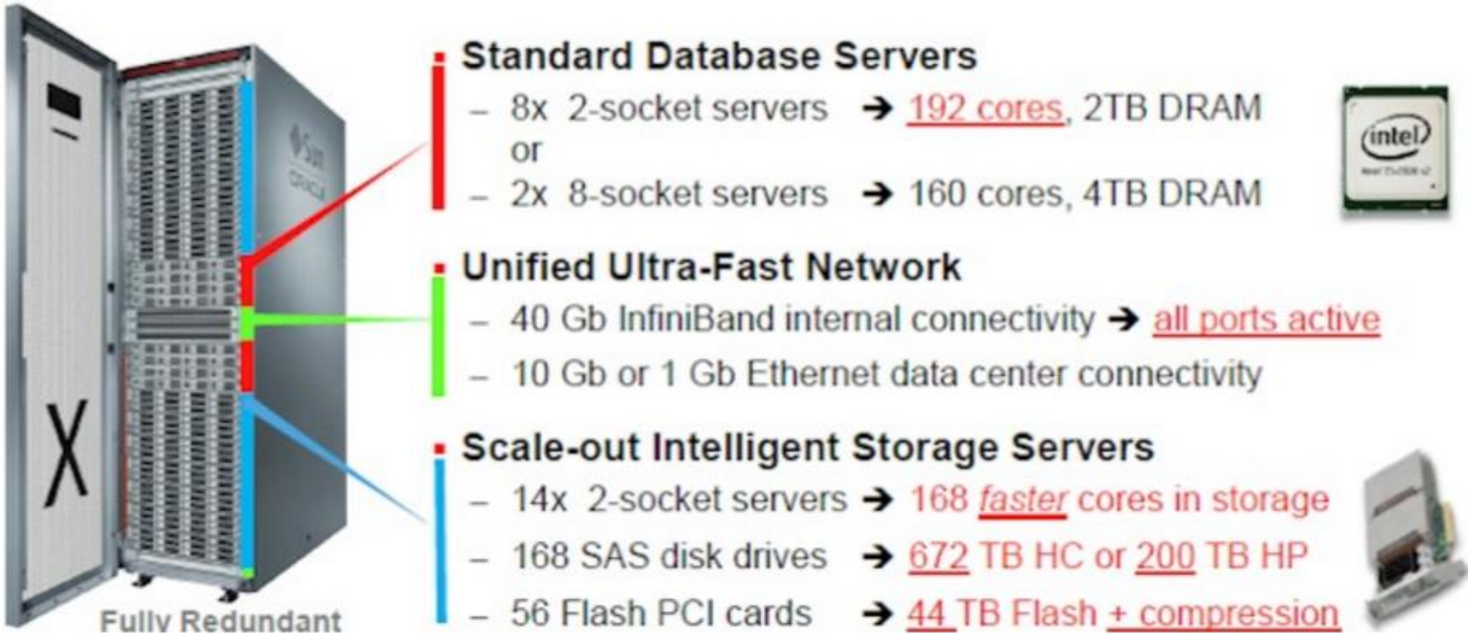
An optimal performance of the whole system is difficult to achieve

Data Warehouse and Big Data Appliances

Data Warehouse Appliances are

- Pre-configured and pre-tested hard- and software configurations developed for running a data warehouse
- **Optimized for data warehousing workload / Only suited for running OLAP**
  In contrast one size fits all: RDBMS are suited for OLTP, OLAP and mixed workloads
- Ready to be used after they are delivered to the customer
- Products, e.g. Teradata, HP Vertica, Exasol, Oracle Exadata, IBM Netezza (IBM PureData System for Analytics), MS Analytic Platform System

Appliance Hardware (e.g. Oracle Exadata)



Oracle Exadata X7 – some key figures

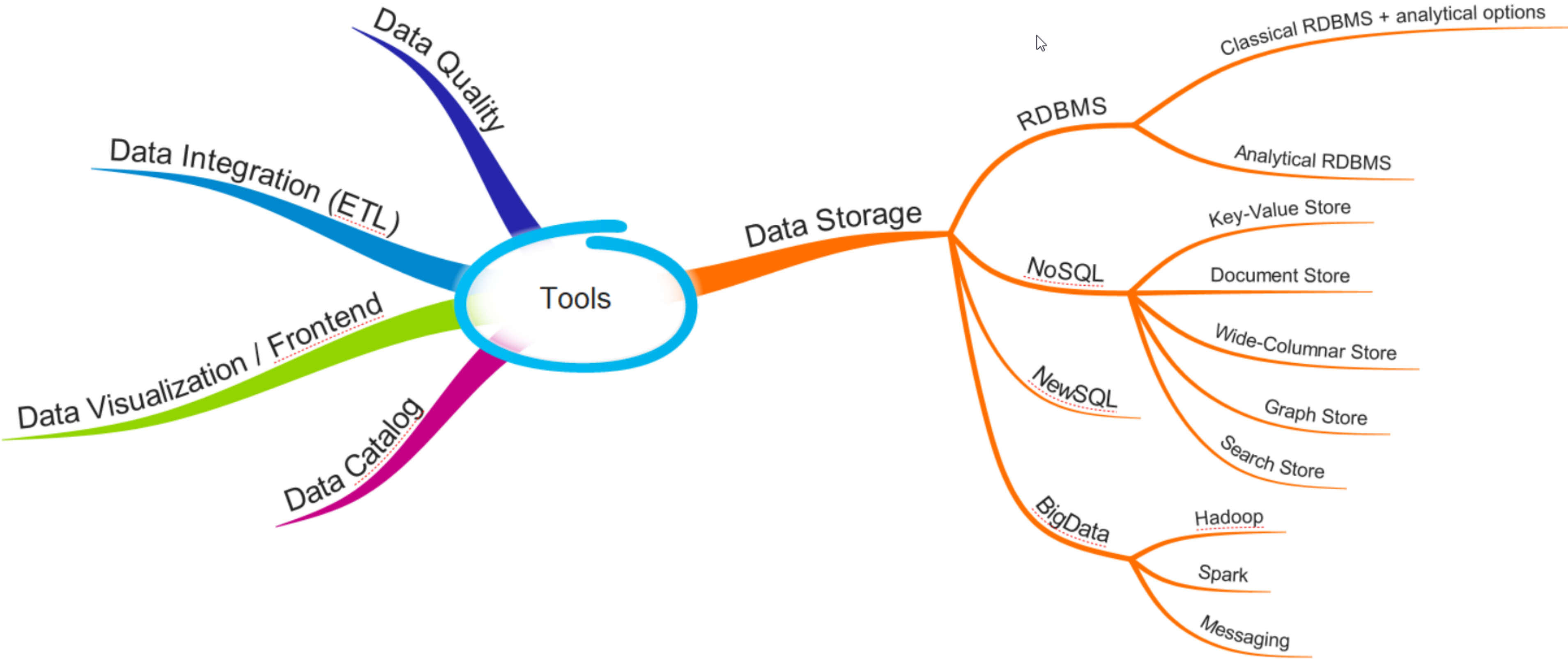
- Up to 912 CPU core and 28.5TB memory per rack
- 2 to 19 DB servers per rack
- 3 to 18 Storage servers per rack
- Maximum of 920TB flash capacity
- 2.1PB of disk capacity
- 10TB size disk (10TB x 12 = 120TB RAW per storage server)
- About 4.8 million reads and about 4.3 million writes per second

Source: http://jaffardba.blogspot.com/2017/10/whats-new-in-exadata-x7.html

Appliances - Typical enhancements

- Move as many **operations** as possible **to storage cell** instead of moving data to the DB server
  - E.g. filter data already at storage cell and not at DB server
    - Avoid transferring unnecessary data
- **Column-oriented In-memory** storage with high compression
- Many appliances are based on **shared nothing architecture**
  - Each node is independent
  - Each node has its own storage or memory
  - Parallel processing simpler and faster as no overhead due to contention

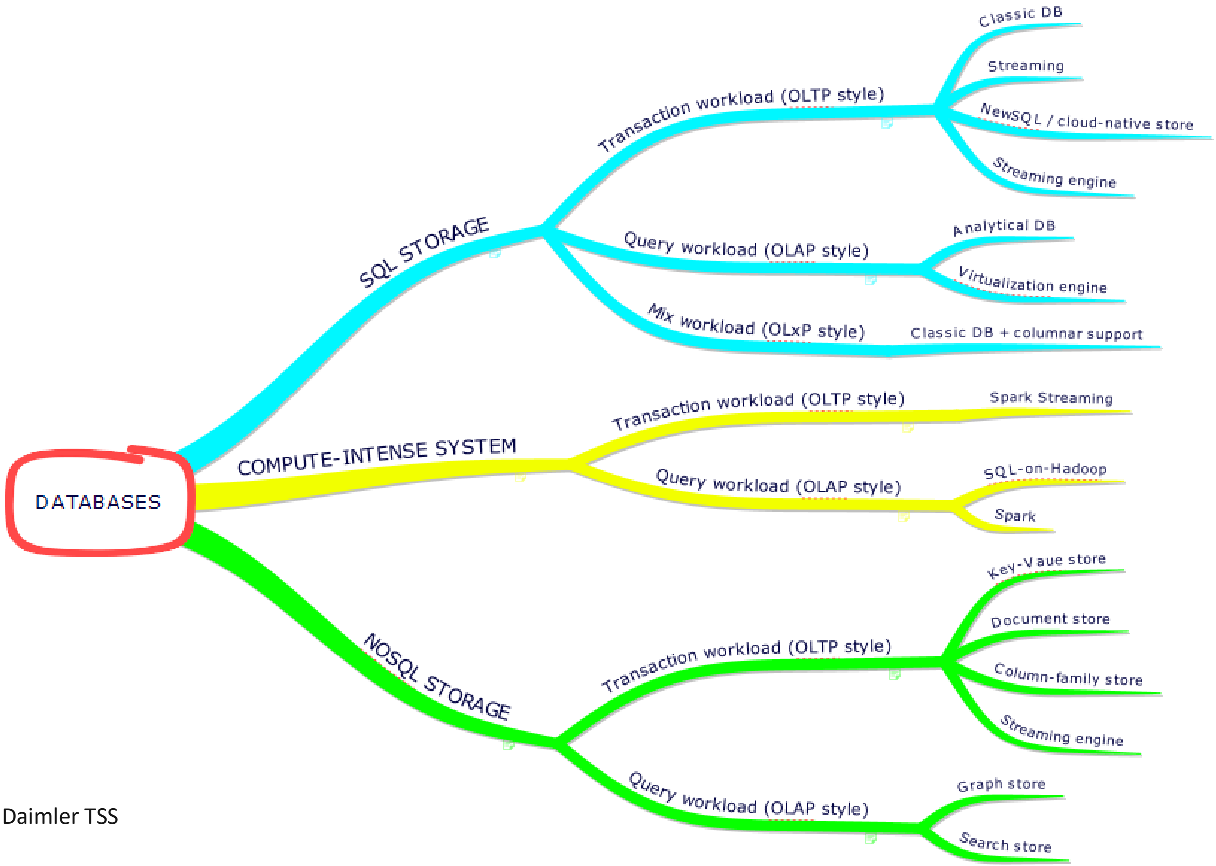
Tools



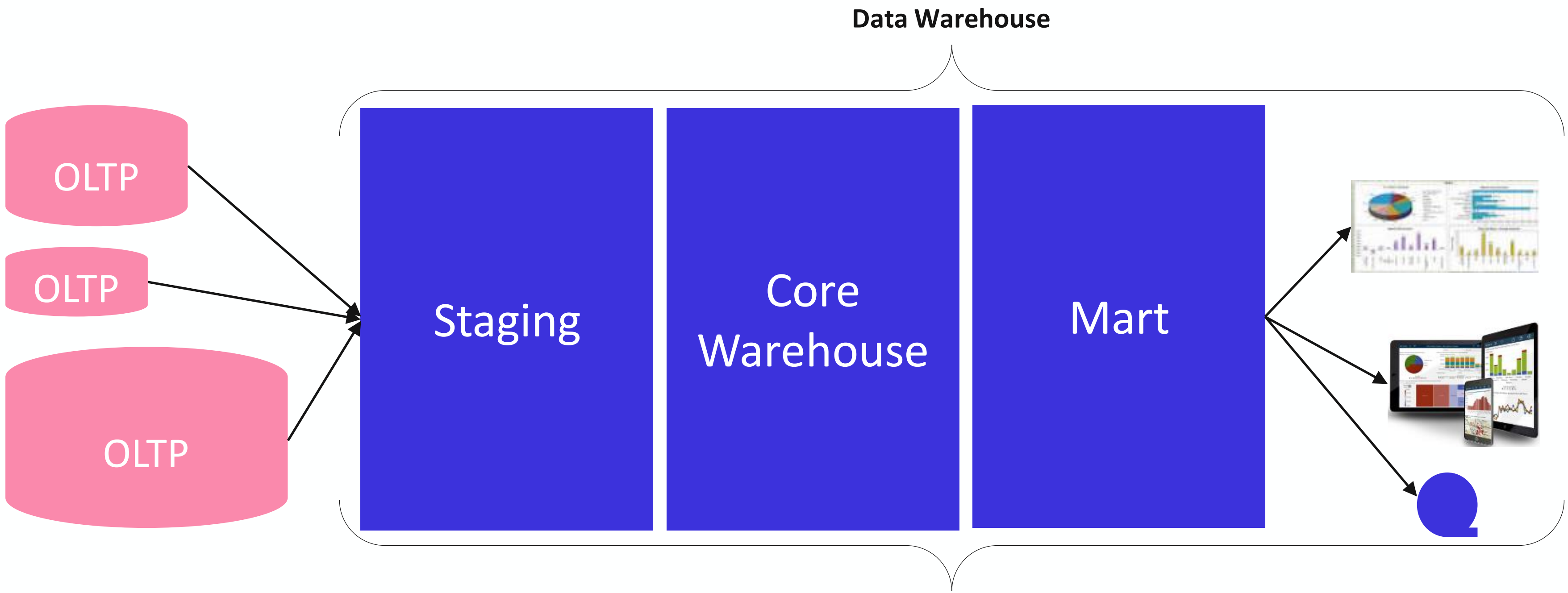
Tools
Data Quality
Data Integration (ETL)
Data Visualization / Frontend
Data Catalog
Data Storage
RDBMS
Classical RDBMS + analytical options
Analytical RDBMS
NoSQL
Key-Value Store
Document Store
Wide-Columnar Store
Graph Store
Search Store
NewSQL
BigData
Hadoop
Spark
Messaging

Daimler TSS GmbH
Wilhelm-Runge-Straße 11, 89081 Ulm / Telefon +49 731 505-06 / Fax +49 731 505-65 99
tss@daimler.com / Internet: www.daimler-tss.com
Sitz und Registergericht: Ulm / HRB-Nr.: 3844 / Geschäftsführung: Martin Haselbach (Vorsitzender), Steffen Bäuerle

Simplified Database landscape

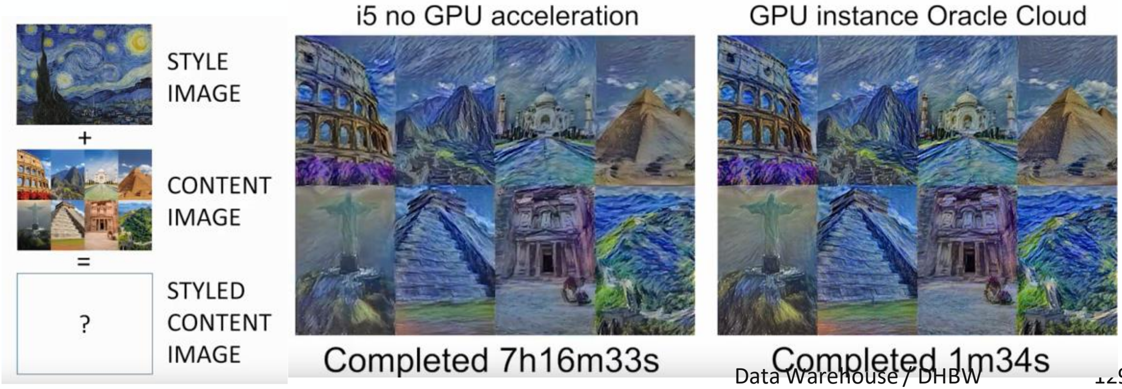


High-Level Data Warehouse Architecture



High compute with GPUs

- High compute with Hadoop / Spark cluster not enough for e.g. machine learning or deep learning (model training)
- GPUs required
  - On-demand availability in the Cloud
  - Can become outdated soon on-premises



Source: https://www.youtube.com/watch?v=R5fKuw36OyA

The rise of data capital avoiding data bankruptcy

Source: https://mydoag.doag.org/formes/pubfiles/10578521/Keynote:%20Data%20capital%20and%20how%20to%20avoid%20data%20bankruptcy/Jean-Pierre%20Dijcks,%20Oracle%20Corporation/

SUMMARY

Which challenges could not be solved by OLTP? Why is a DWH necessary?

- Integrated view, distributed data, historic data, technological challenges, system workload, different data structures

Name two “fathers” of the DWH

- Bill Inmon and Ralph Kimball

Which characteristics does a DWH have according to Bill Inmon?

- Subject-oriented, integrated, non-volatile, time-variant

- https://databricks.com/try-databricks

Spark exercise

Launch cloud-optimized Apache Spark™ clusters in minutes

DATABRICKS PLATFORM – FREE TRIAL

For businesses looking for a zero-management cloud platform built around Apache Spark

- Unlimited clusters that can scale to any size
- Job scheduler to execute jobs for production pipelines
- Fully interactive notebook with collaboration, dashboards, REST APIs
- Advanced security, role-based access controls, and audit logs
- Single Sign On support
- Integration with BI tools such as Tableau, Qlik, and Looker
- 14-day full feature trial (excludes cloud charges)

GET STARTED

COMMUNITY EDITION

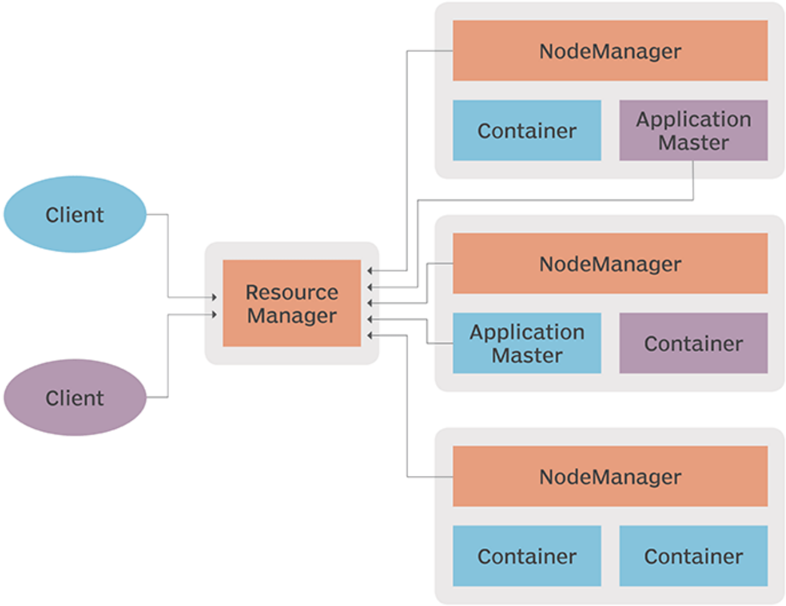
For students and educational institutions just getting started with Apache Spark

- Single cluster limited to 6GB and no worker nodes
- Basic notebook without collaboration
- Limited to 3 max users
- Public environment to share your work

GET STARTED

- Create Spark cluster
- Run and understand existing default notebook
- Create new notebook and run SQL + Python code

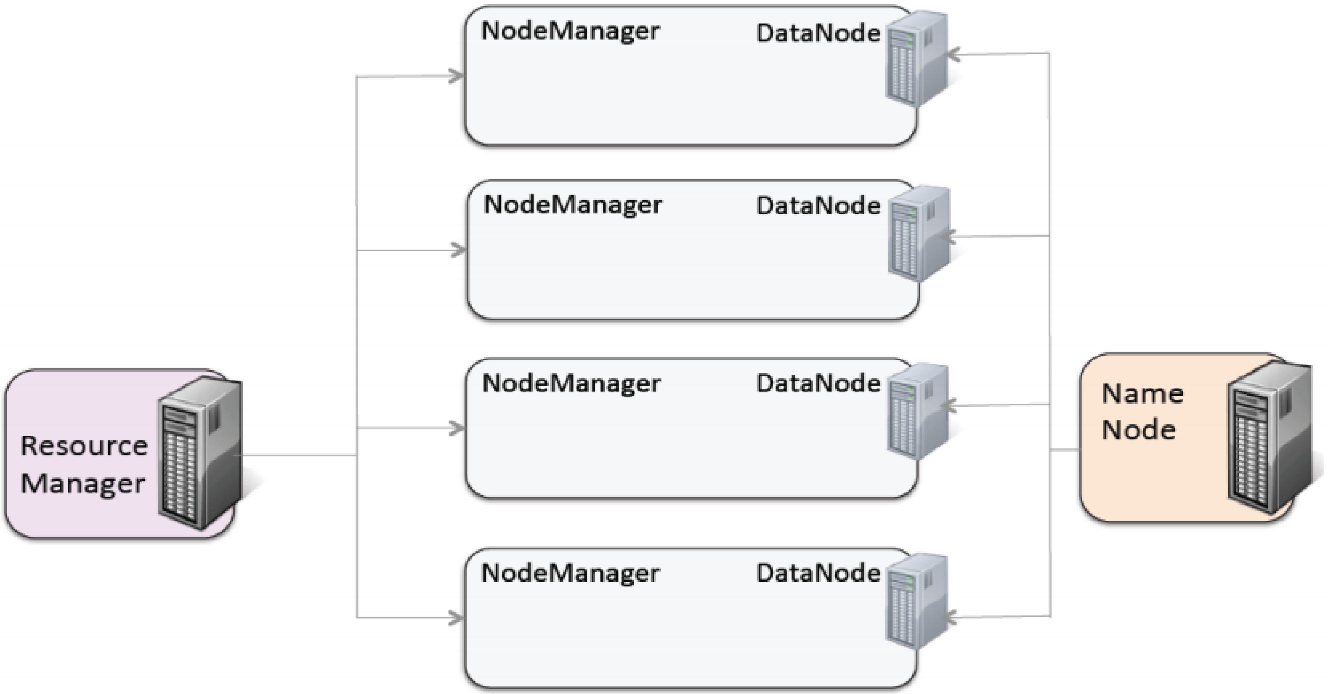
YARN architecture



- **Resource Manager**: accepts job submissions, allocates resources
- **Node Manager**: is a monitoring and reporting agent of the Resource Manager
- **Application Master**: created for each application to negotiate for resources and work with the NodeManager to execute and monitor tasks
- **Container**: controlled by NodeManagers and assigned the system resources

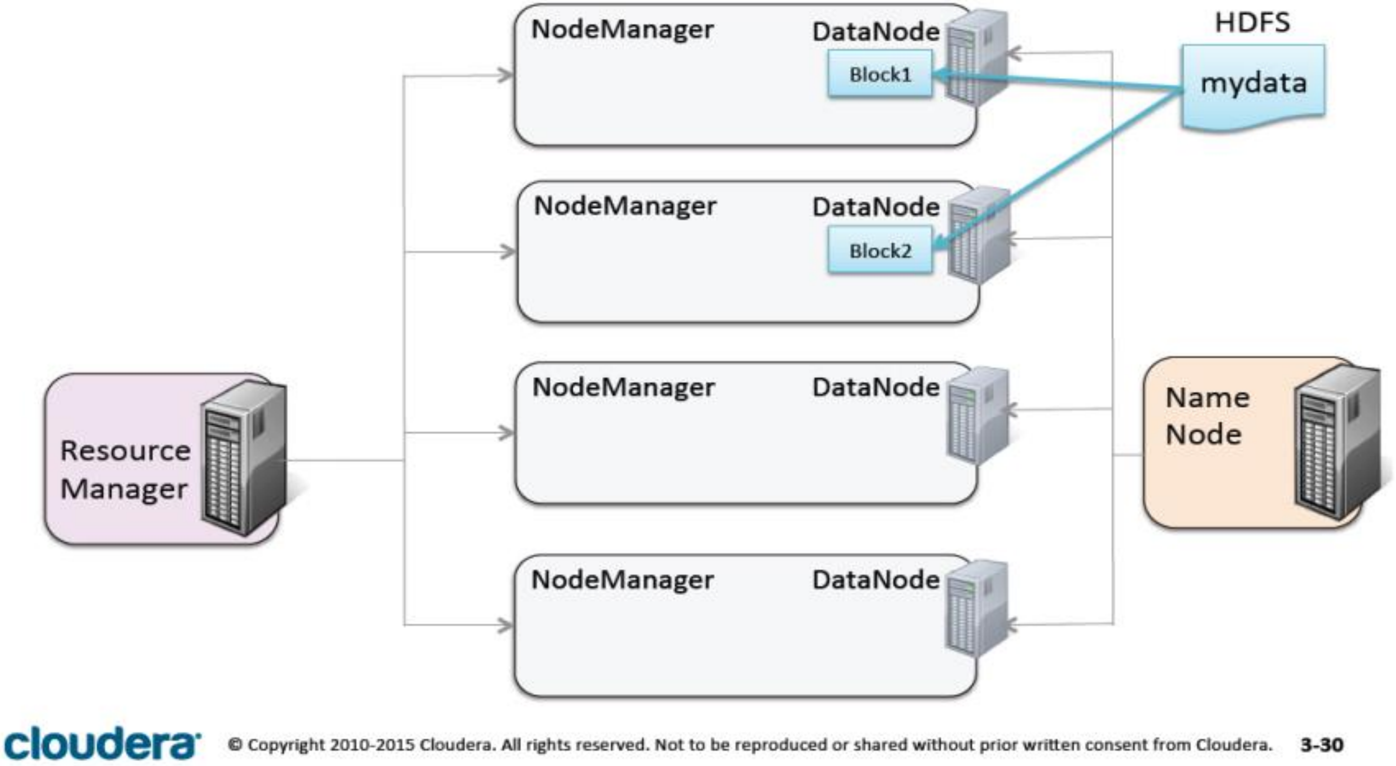
Source: https://searchdatamanagement.techtarget.com/definition/Apache-Hadoop-YARN-Yet-Another-Resource-Negotiator

Running an application on YARN (1)



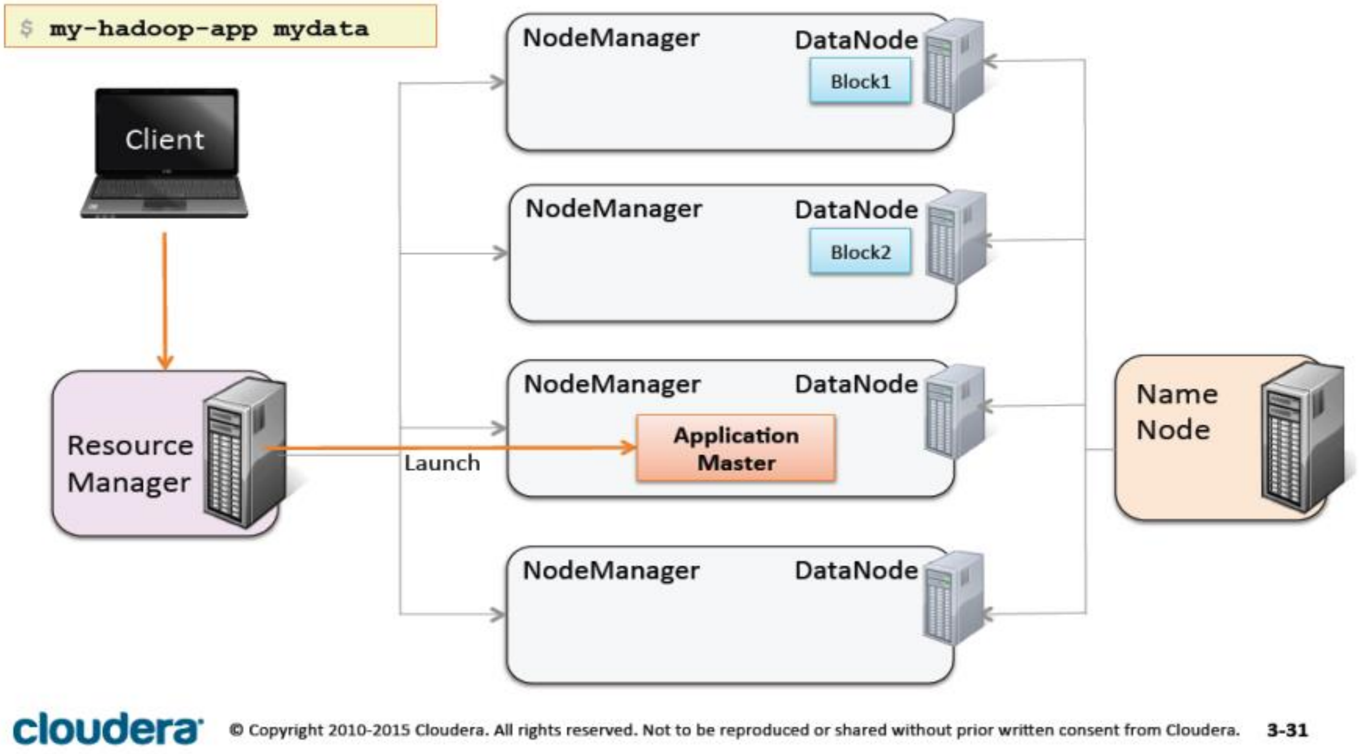
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (2)



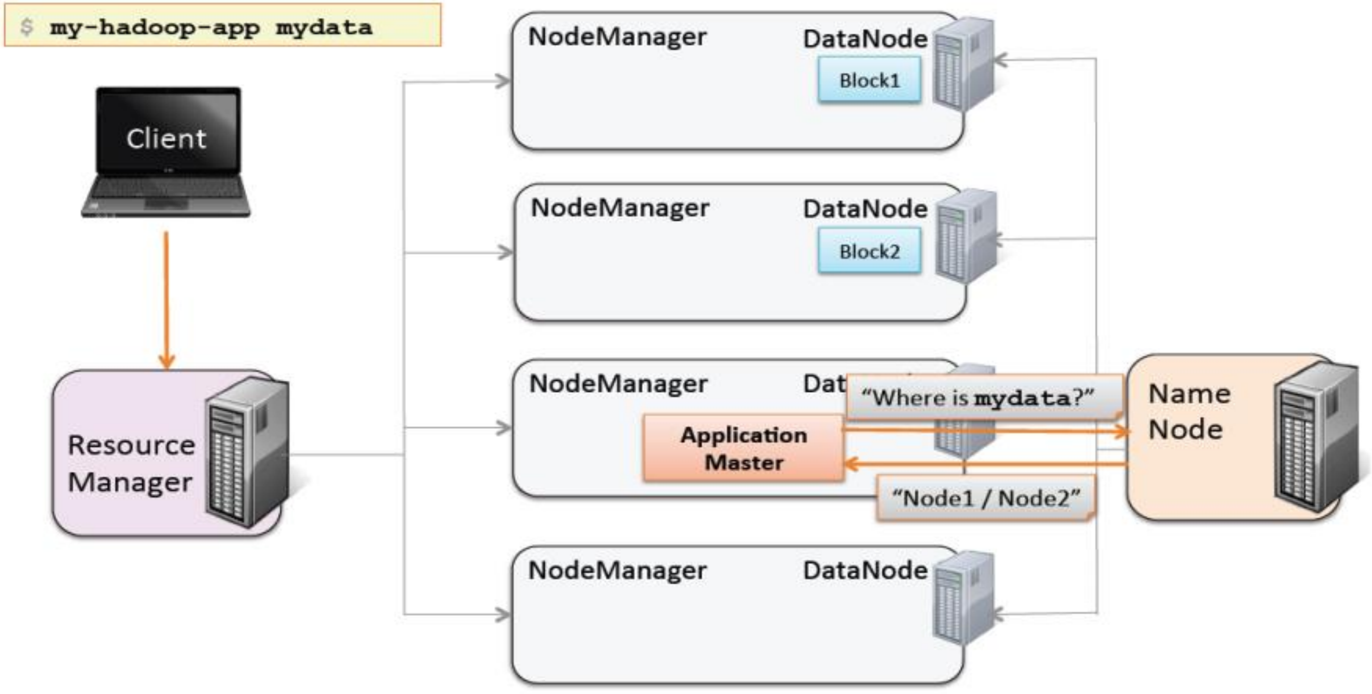
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (3)



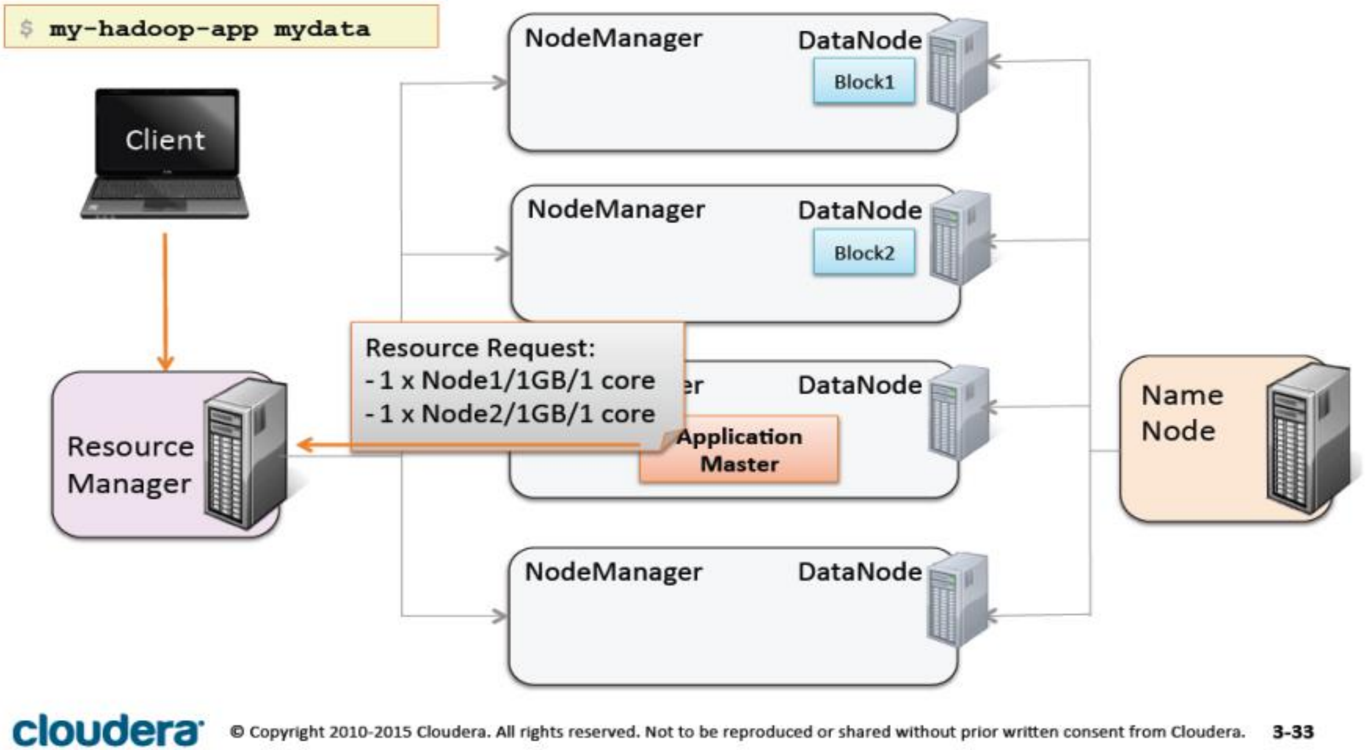
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (4)



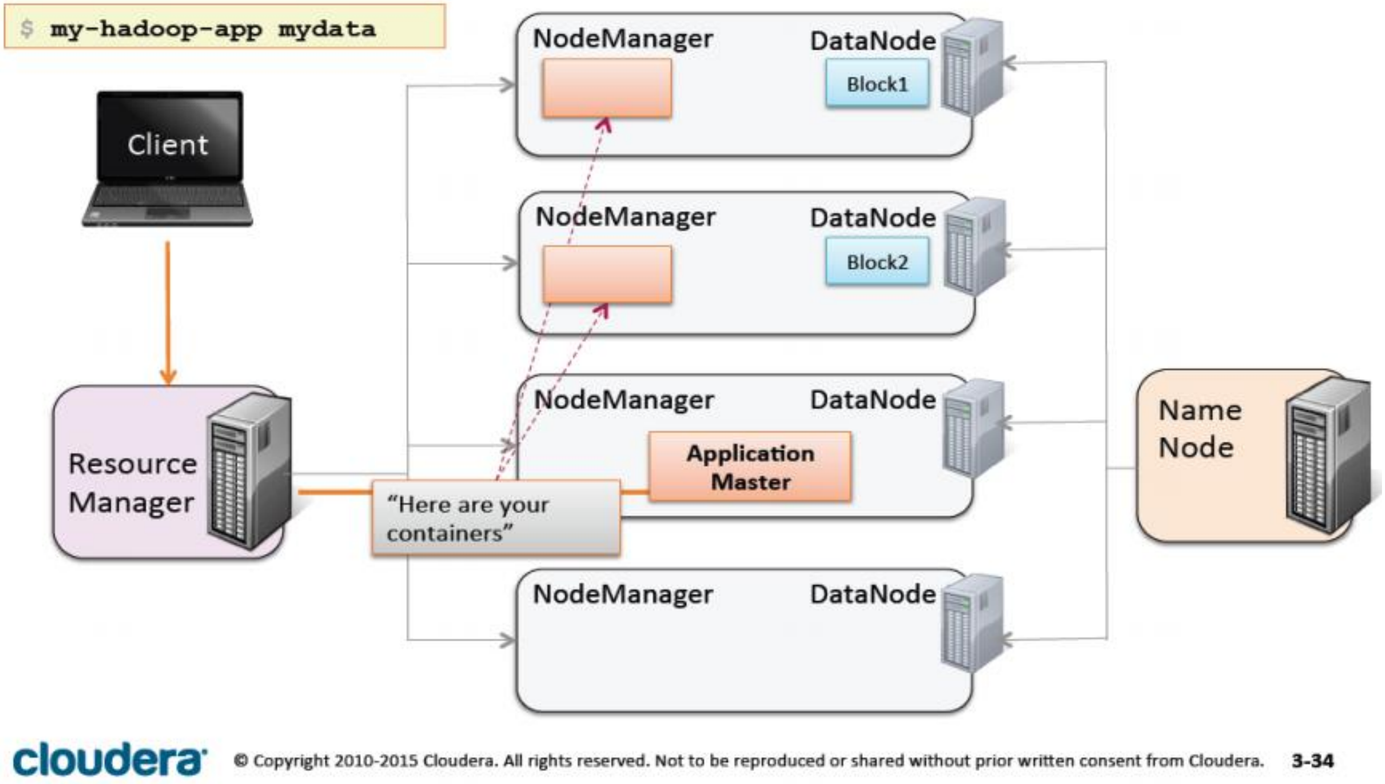
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (5)



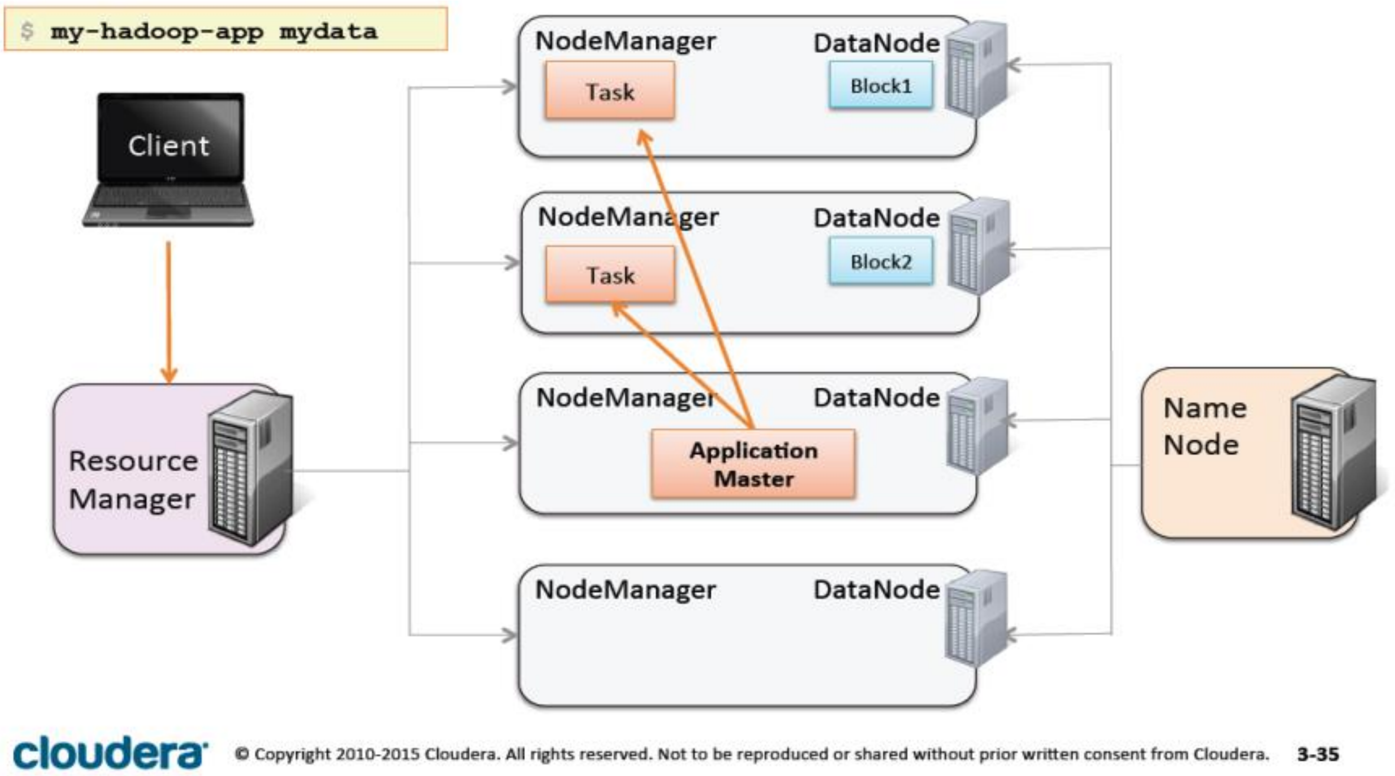
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (6)



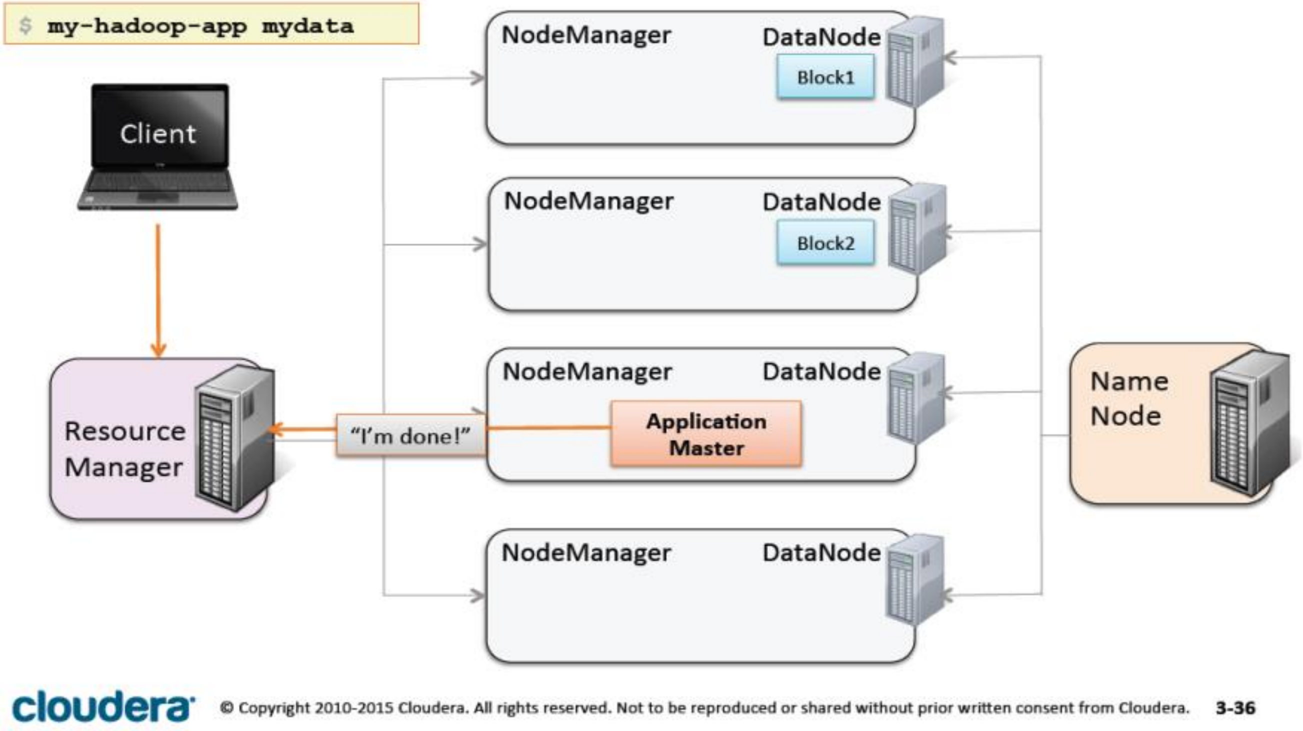
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (7)



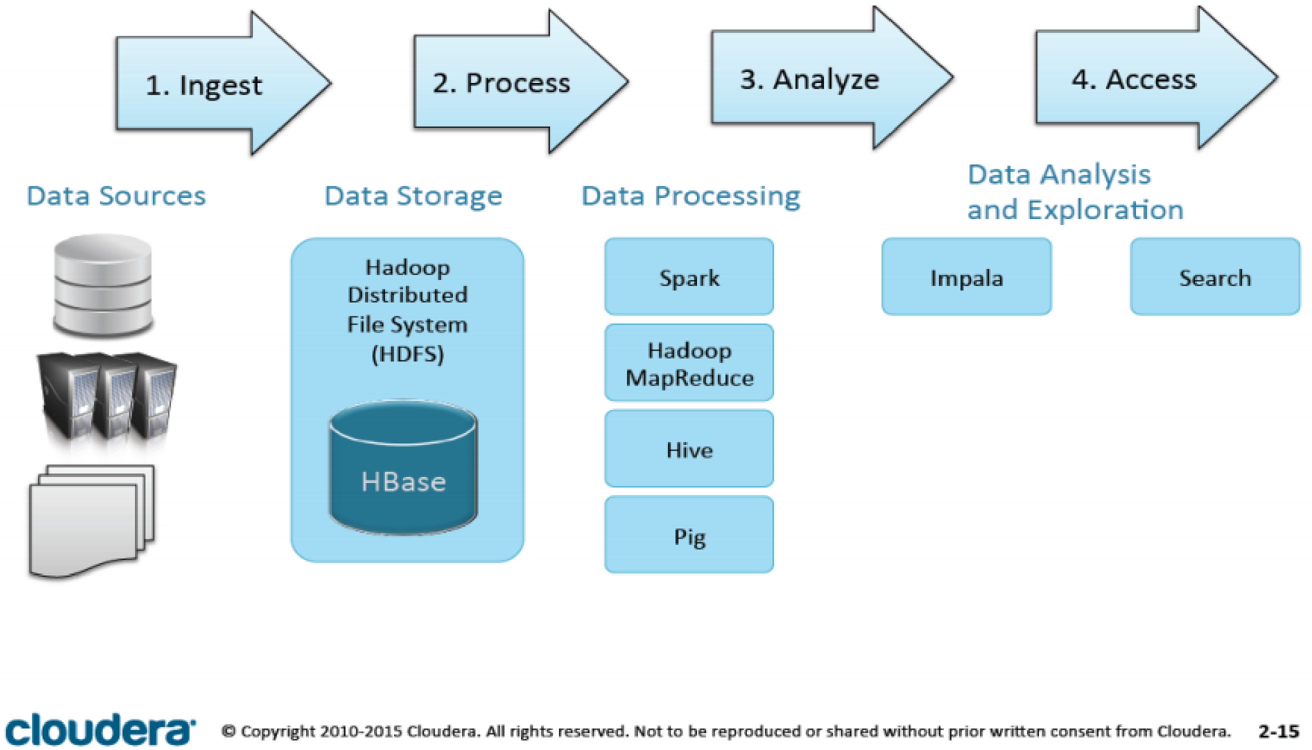
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Running an application on YARN (8)



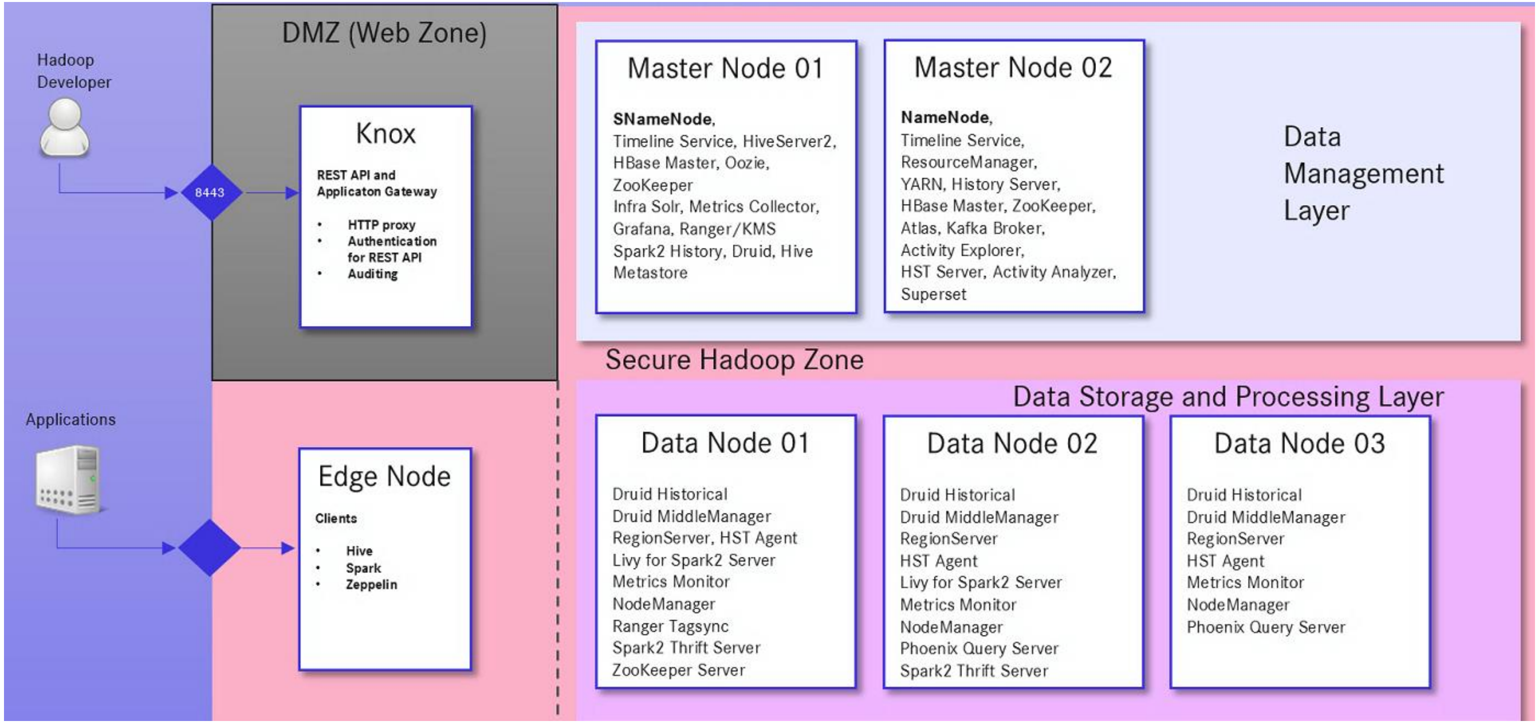
Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

Hadoop ecosystem

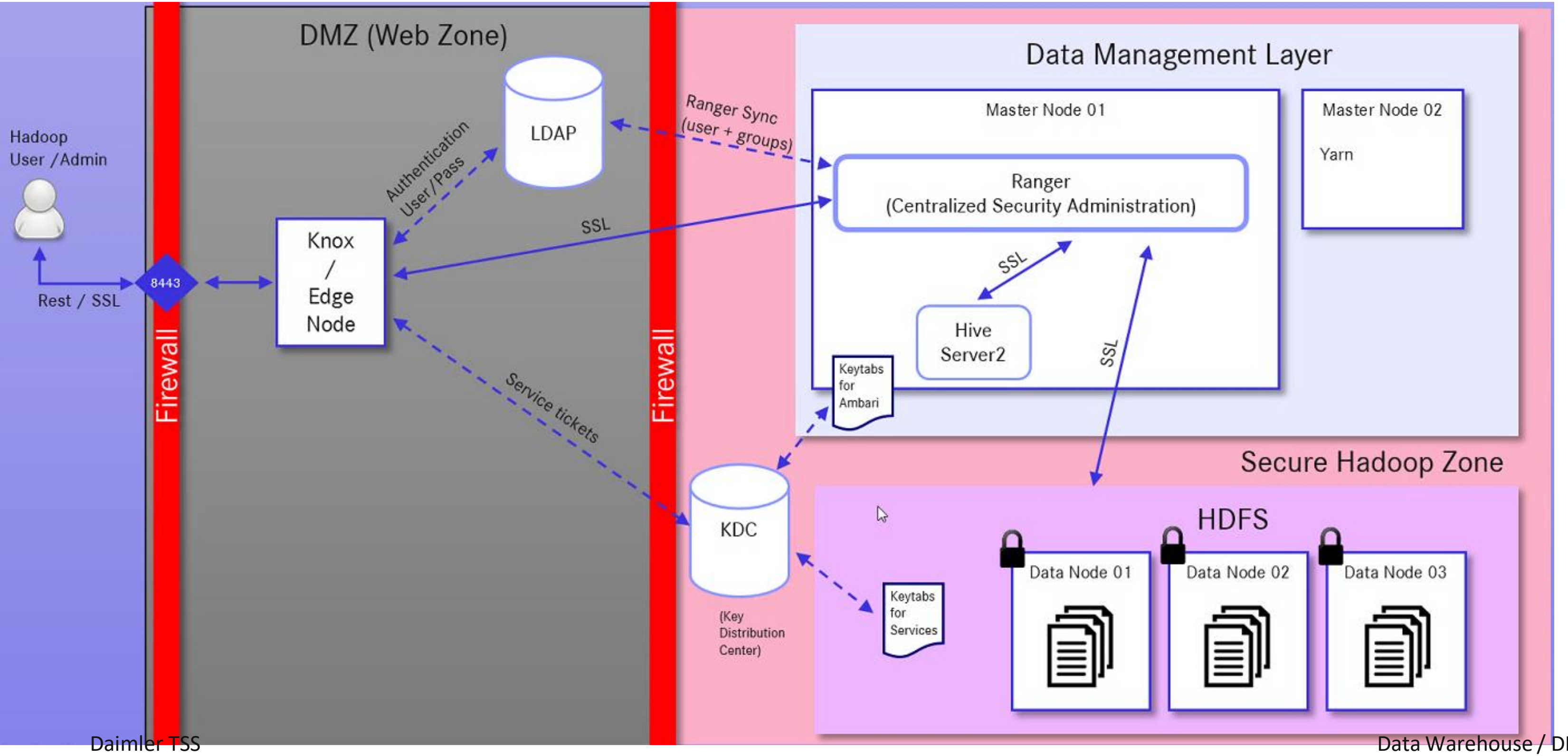


Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

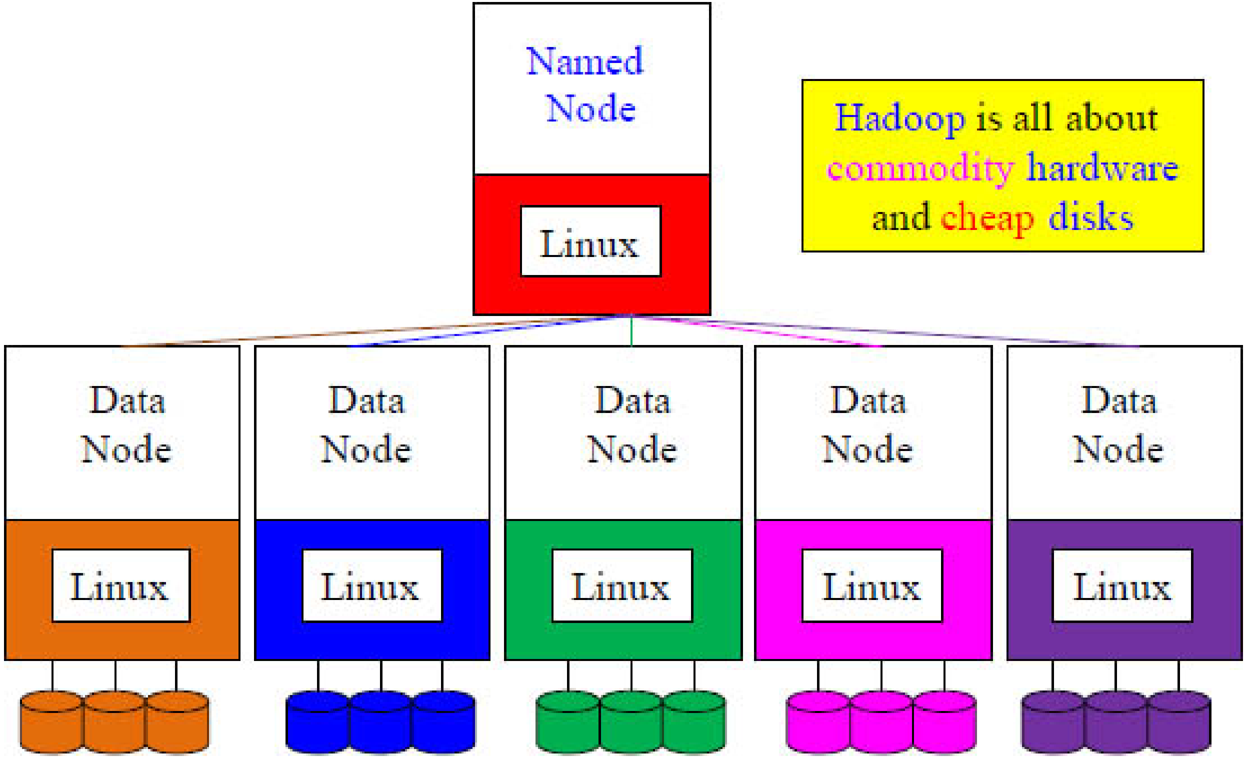
Hadoop Cluster



Hadoop Cluster - security

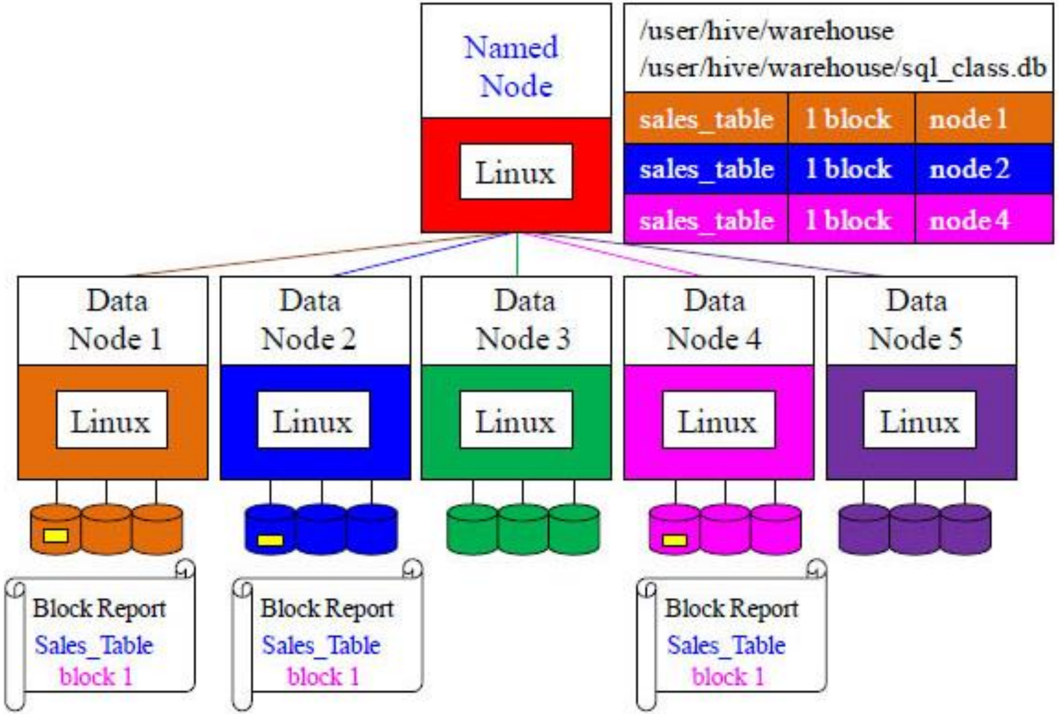


What is Hadoop all about?



Source: Jason Nolander, Tom Coffing: Tera-Tom Genius Series - Hadoop Architecture and SQL, Coffing Publishing 2016

Data layout

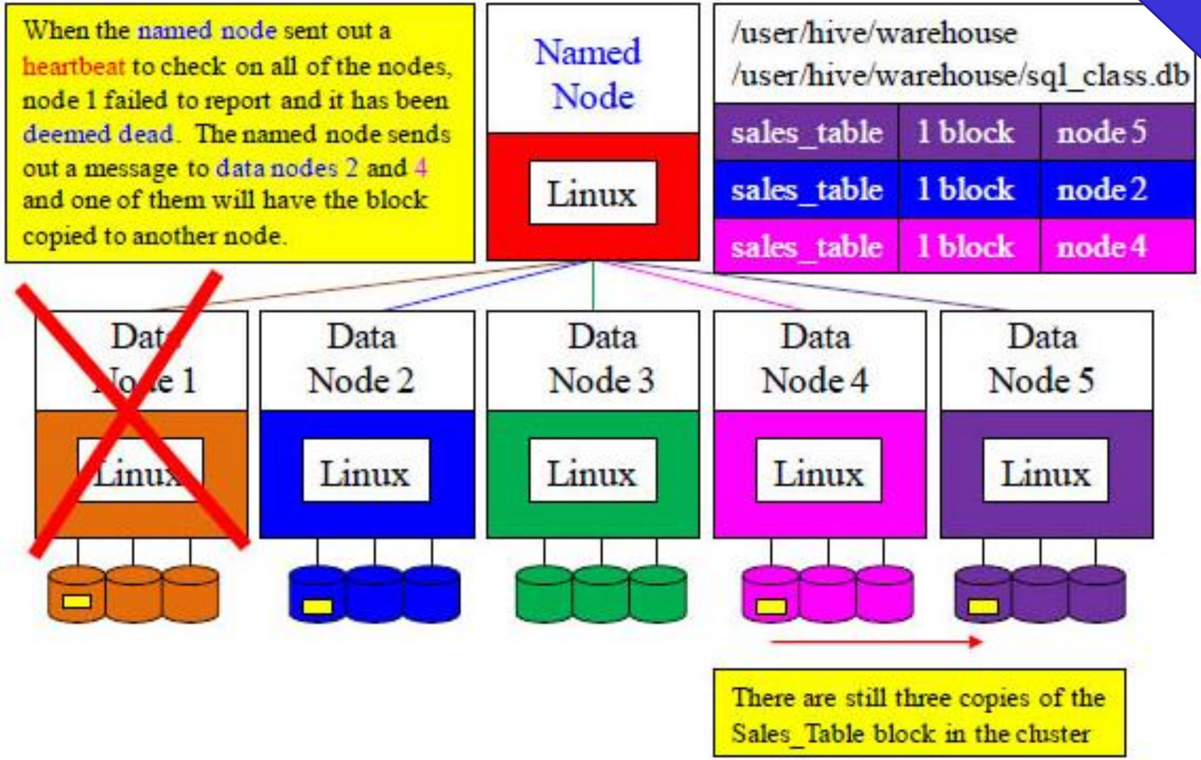


Name Node is single point of failure and can become bottleneck ☹

Algorithms come to the data and not vice versa

Source: Jason Nolander, Tom Coffing: Tera-Tom Genius Series - Hadoop Architecture and SQL, Coffing Publishing 2016

Data layout and protection



Is replication a substitute for backups? What about the Name Node?

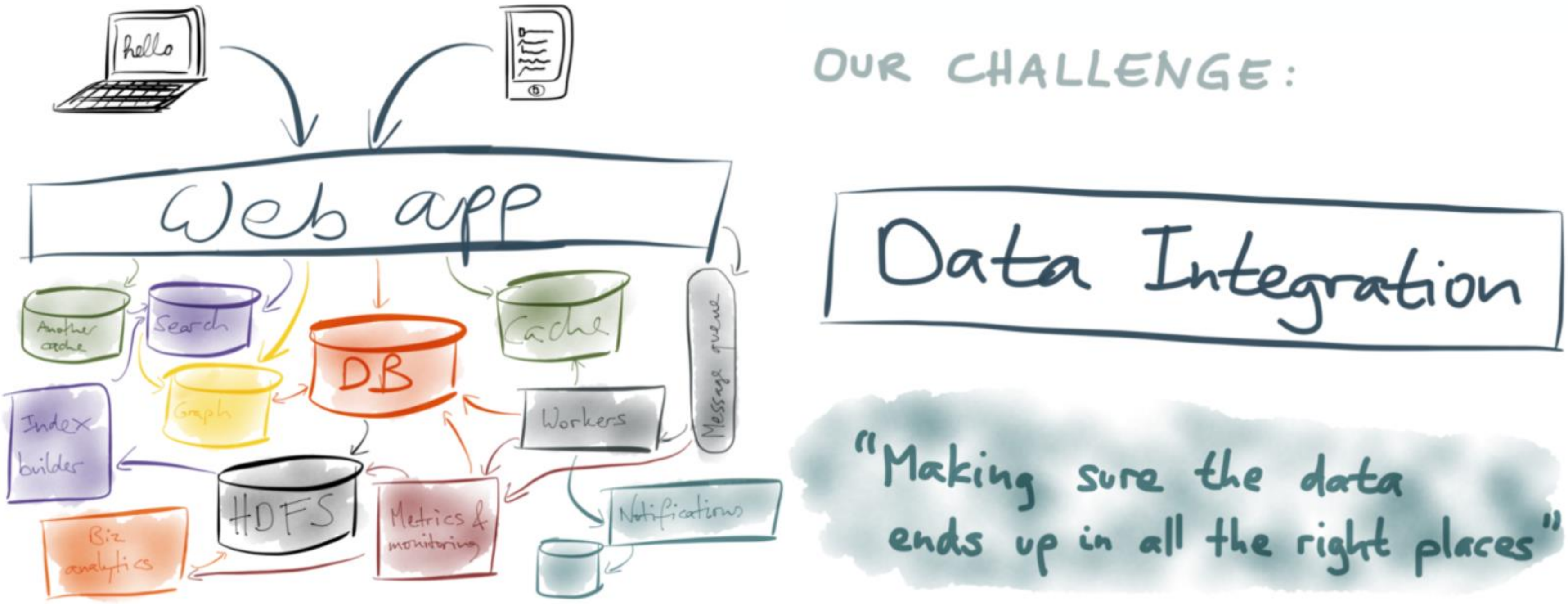
Source: Jason Nolander, Tom Coffing: Tera-Tom Genius Series - Hadoop Architecture and SQL, Coffing Publishing 2016

Rdd = resilient distributed dataset

- An RDD is an immutable fault-tolerant, distributed collection of objects that are operated on in parallel
- An RDD can contain any type of object and is created by loading an external dataset
- Lazy transformations by Spark
  - Results are not computed right away
- Spark loads datasets in memory
  - When datasets are too large to fit into memory, they are spilled to disk automatically
- Actions on RDDs instruct worker nodes either to save the data in the RDD or to send data to main program (running in the driver node)

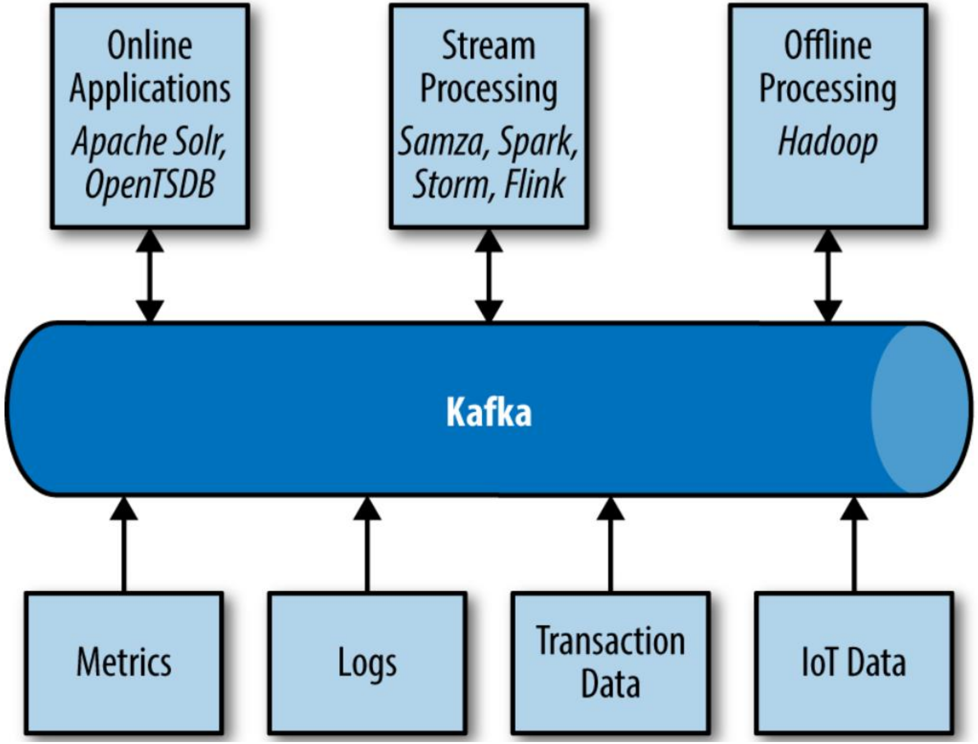
Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

Logs to build a solid intrastructure



Source: https://www.confluent.io/blog/using-logs-to-build-a-solid-data-infrastructure-or-why-dual-writes-are-a-bad-idea/

Ecosystem around kafka



Source: Gwen Shapira, Neha Narkhede, Todd Palino - Kafka: The Definitive Guide, O'Reilly 2017

Kafka – distributed streaming platform

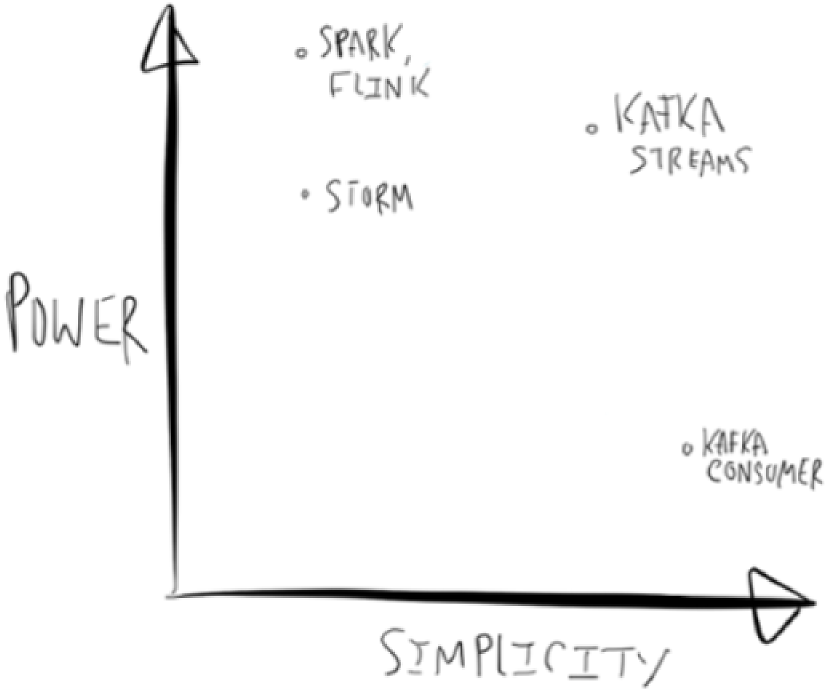
- Open-source distributed streaming platform
- Originally developed at LinkedIn
  - Old system was a mix of pull and push (large intervals)
    - Monitoring of application metrics for user requests like CPU
    - User activity tracking, e.g. page views
  - Vision: Combine systems into one backend
  - Vision was not possible with architecture they had
- Confluent is most well-know distribution
- Kafka is a system optimized for writing: Jay Kreps searched for a writer‘s name: [Franz] Kafka sounded cool

Kafka – original goals @linkedIn

- Required: publish/subscribe messaging system with storage layer similar to a log
- Goals
  - Decouple producers and consumers by using push – pull
  - Provide persistence for message data within messaging system to allow for multiple consumers
  - Optimized for high throughput of messages (performance tests with ActiveMQ failed)
  - Allow for horizontal scaling

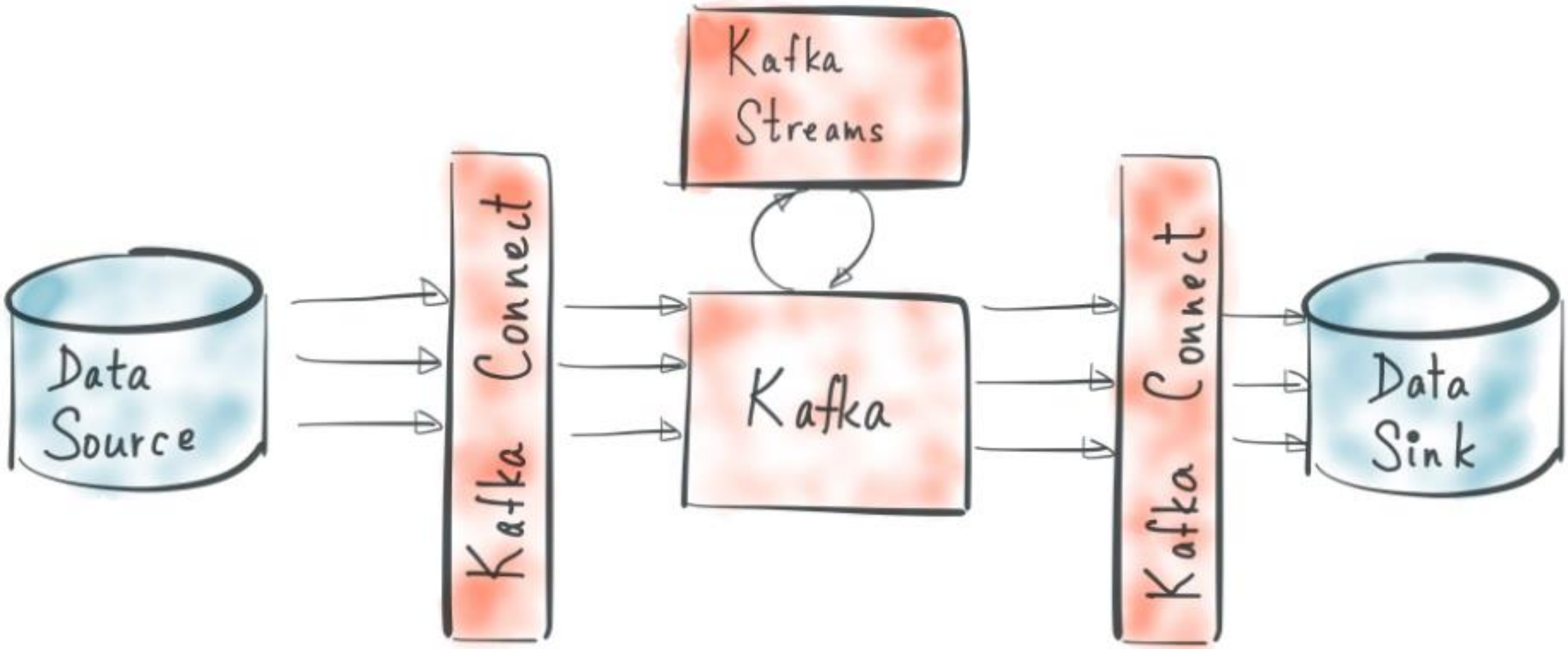
Source: Gwen Shapira, Neha Narkhede, Todd Palino - Kafka: The Definitive Guide, O'Reilly 2017

Streaming tools



Source: https://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple/

Kafka streams for creating stream processing applications



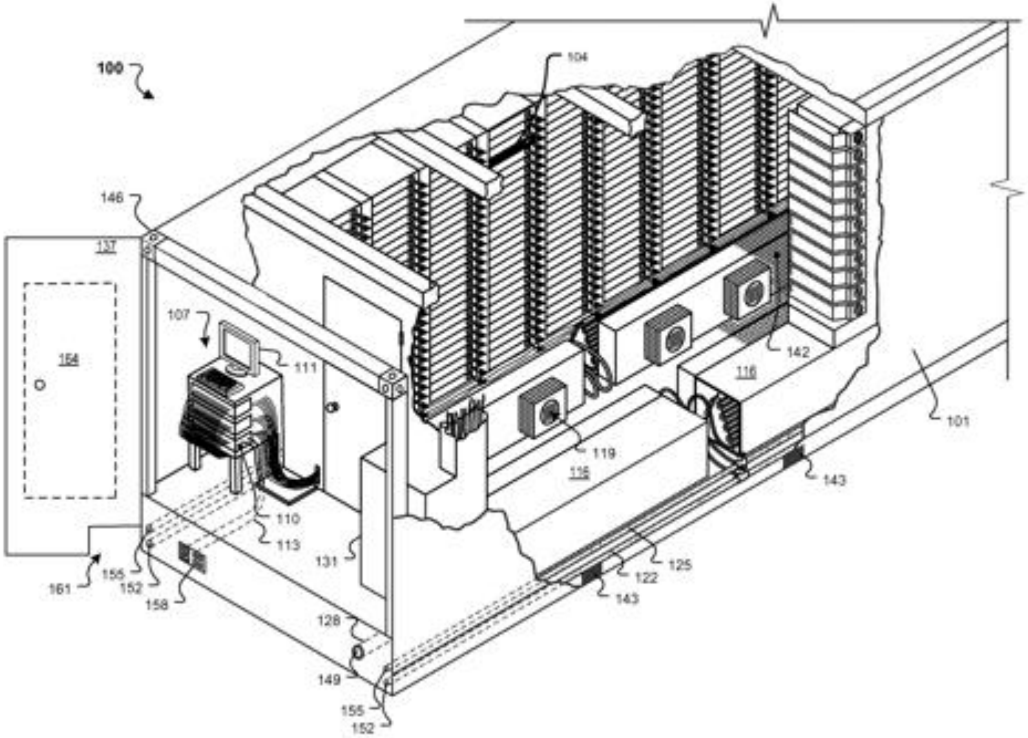
Source: https://www.confluent.io/blog/hello-world-kafka- connect-kafka-streams/

KSQL – streaming SQL

- SQL engine for Kafka (on top of Kafka Streams)
- SQL for analyzing data streams in real-time / near-real-time
- Stream is an unbounded set of data

```
$ ksql-server-start etc/ksql/ksql-server.properties
...
                  ===========================================
                  =        _  __ _____  ____  _             =
                  =       | |/ // ____|/ __ \| |            =
                  =       | ' /| (___ | |  | | |            =
                  =       |  <  \___ \| |  | | |            =
                  =       | . \ ____) | |__| | |____        =
                  =       |_|\_\_____/ \___\_\______|       =
                  =                                         =
                  =  Streaming SQL Engine for Apache Kafka® =
                  ===========================================

Copyright 2017-2018 Confluent Inc.

Server 5.0.0 listening on http://localhost:8088

To access the KSQL CLI, run:
ksql http://localhost:8088
```

Google modular data center

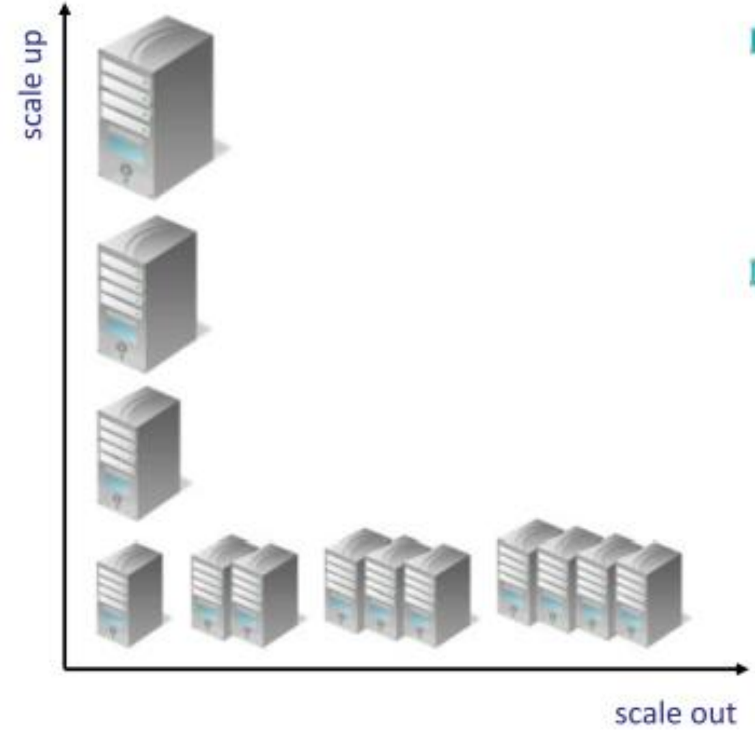


Increase data center capacity by adding 1000 new servers modules at once

Data center: https://www.youtube.com/watch?v=zRwPSFpLX8I

Source: https://patents.google.com/patent/US20100251629

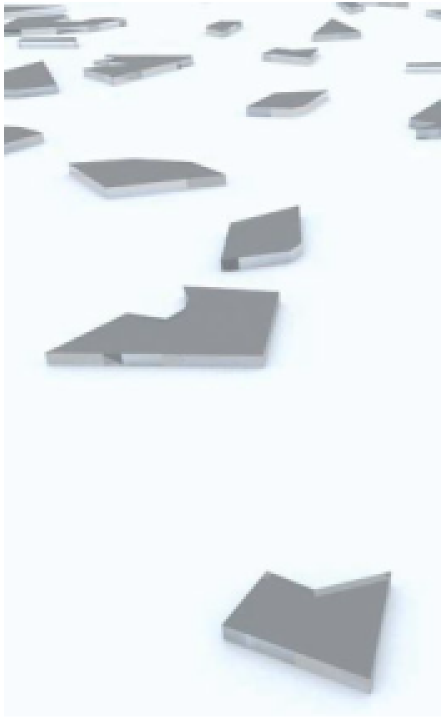
Scale up vs scale out



- Scale up (vertical scaling) means adding more resources to one node in a system
- Scale out (horizontal scaling) means adding more nodes to a system
  - Continuous availability/redundancy
  - Cost/performance flexibility
  - Contiguous upgrades
  - Geographical distribution

Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

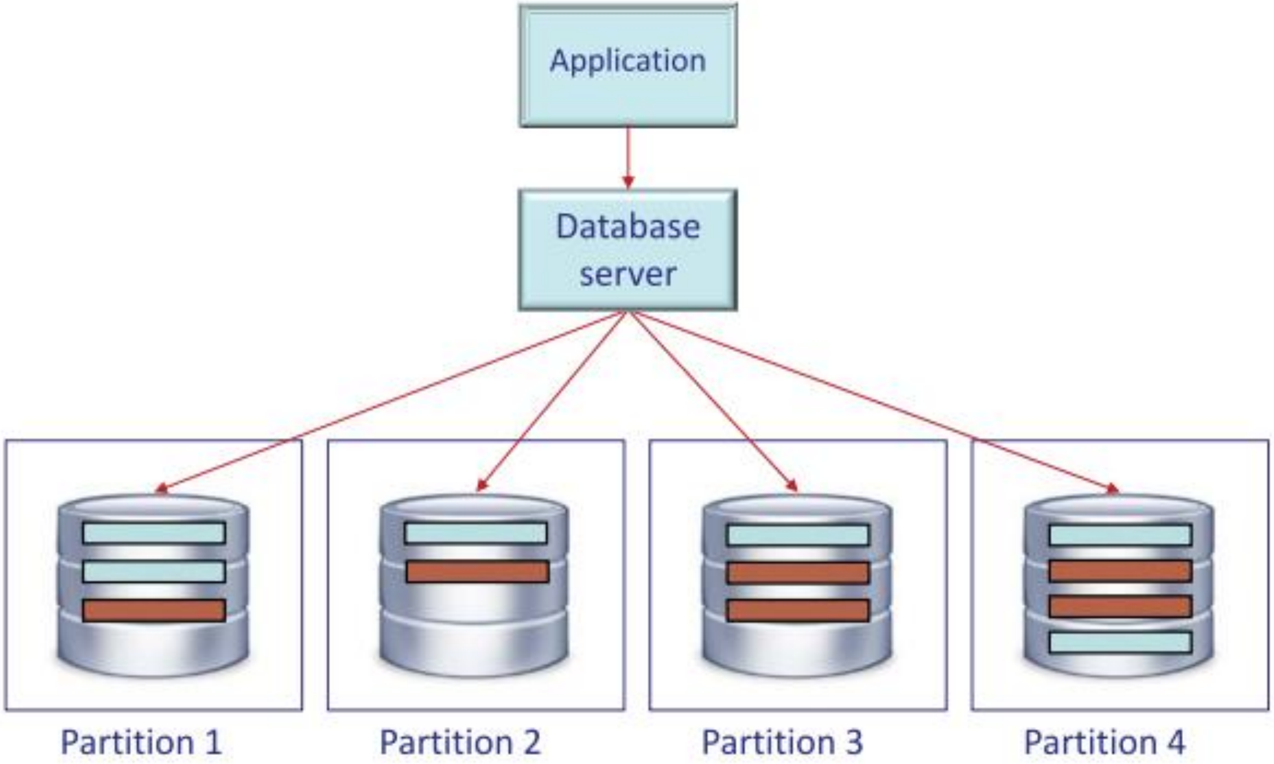
Partitioning versus sharding



- Partitioning =
  - Table partitioning means breaking a table horizontally or vertically
- Sharding =
  - Sharding is a “shared-nothing” horizontal partitioning scheme across a number of servers each with their own CPU, memory and disk
  - A lookup table keeps track of which data is stored in which shard

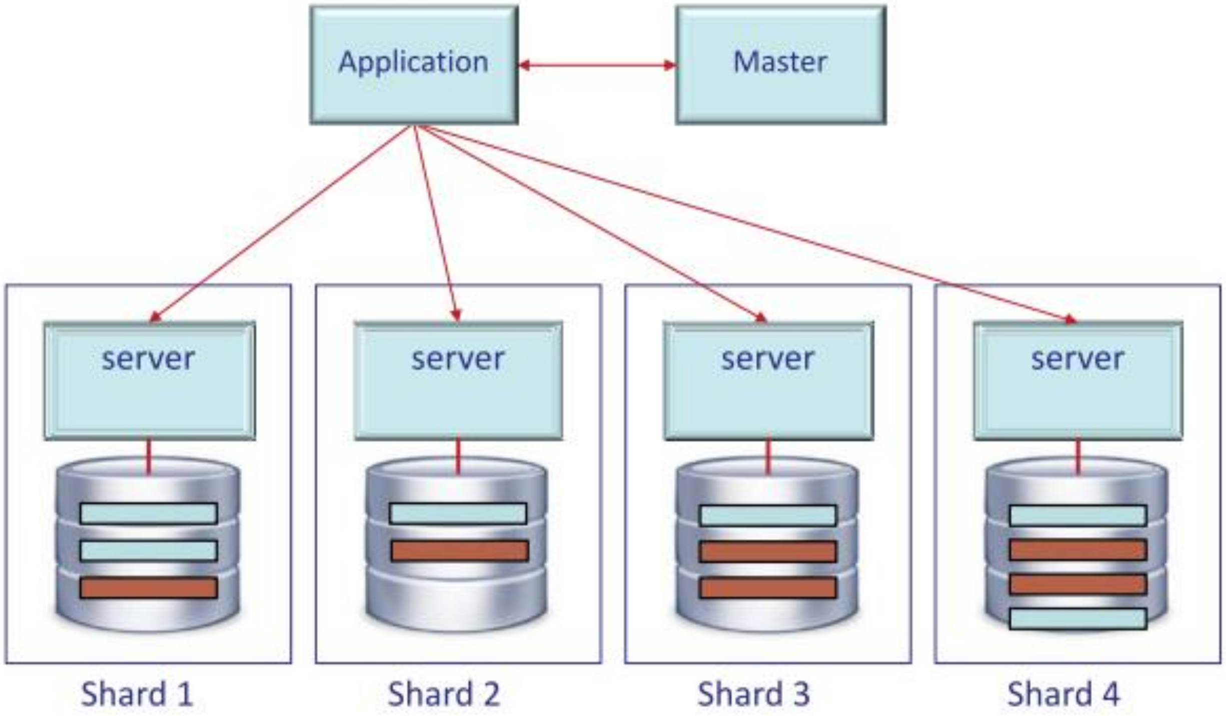
Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

Partitioning of table data



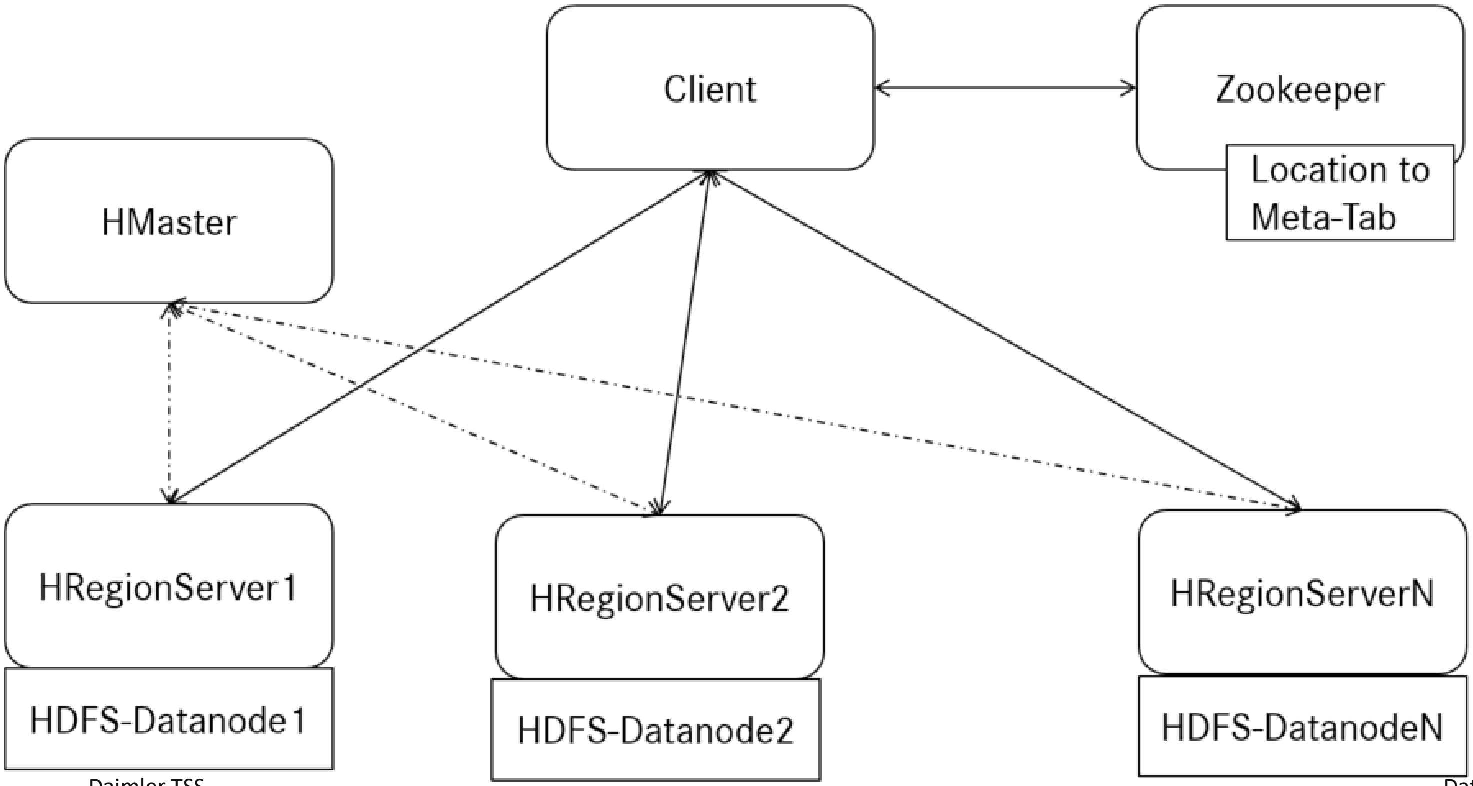
Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

sharding of table data

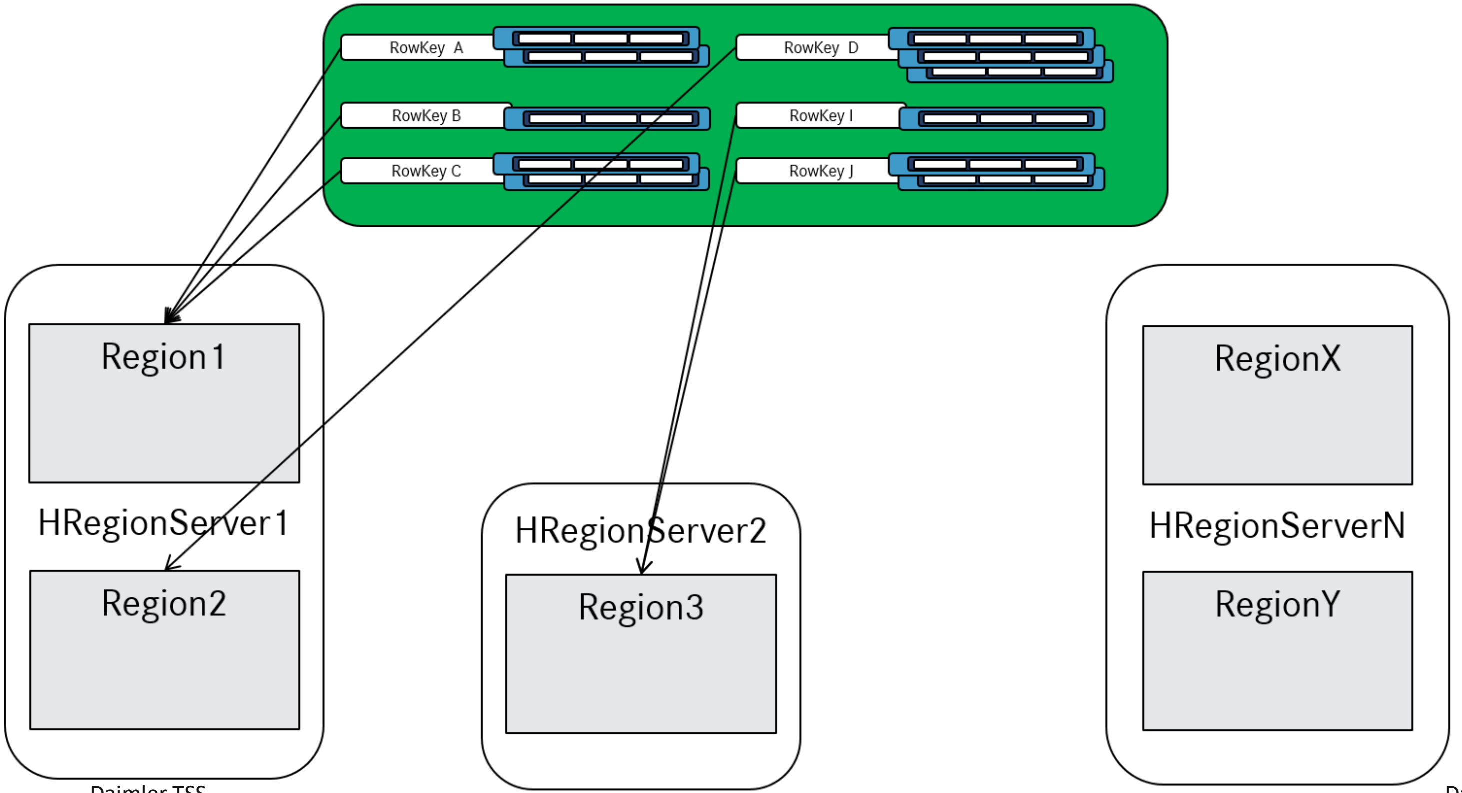


Source: Rick F. van der Lans: New Data Storage Technologies, TDWI Munich 2018

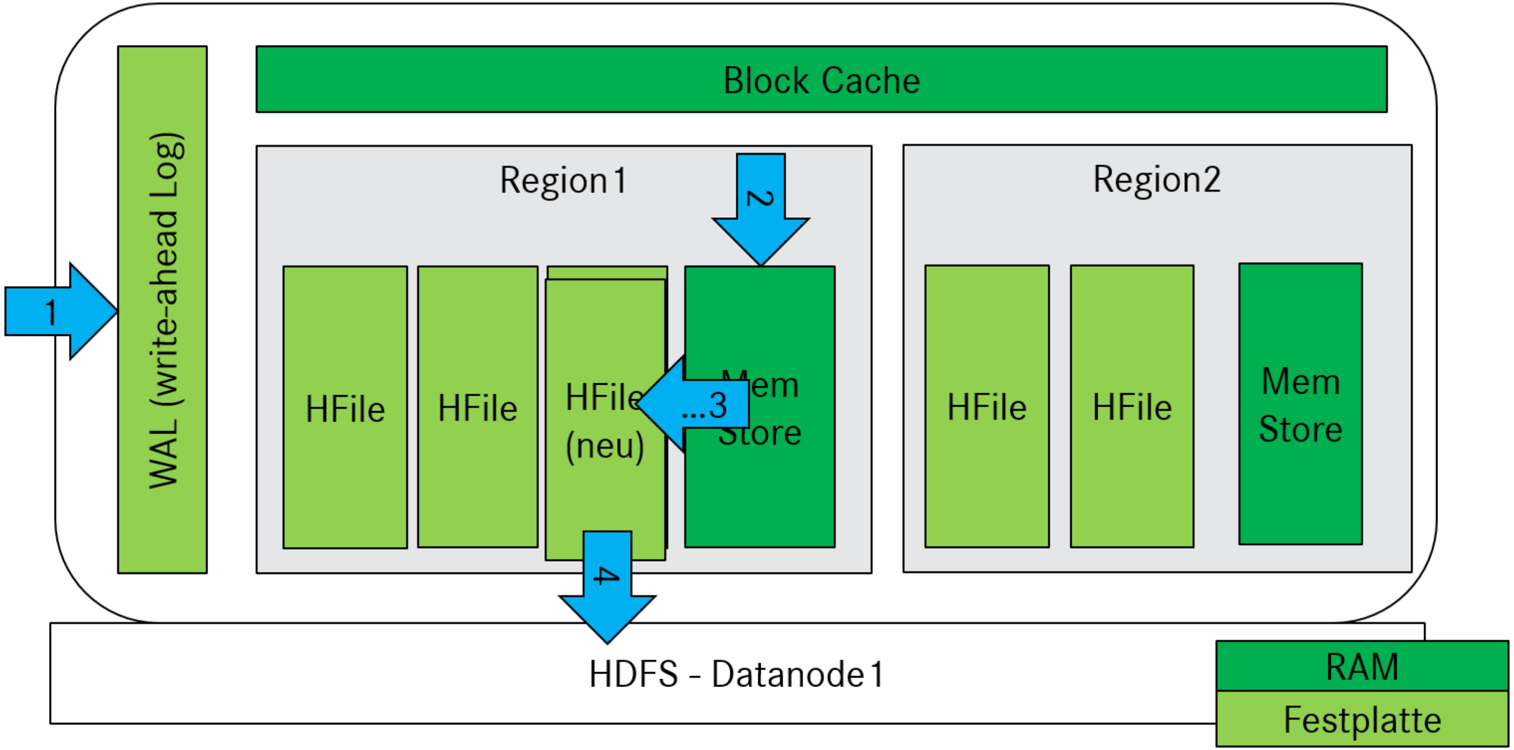
HBase architecture



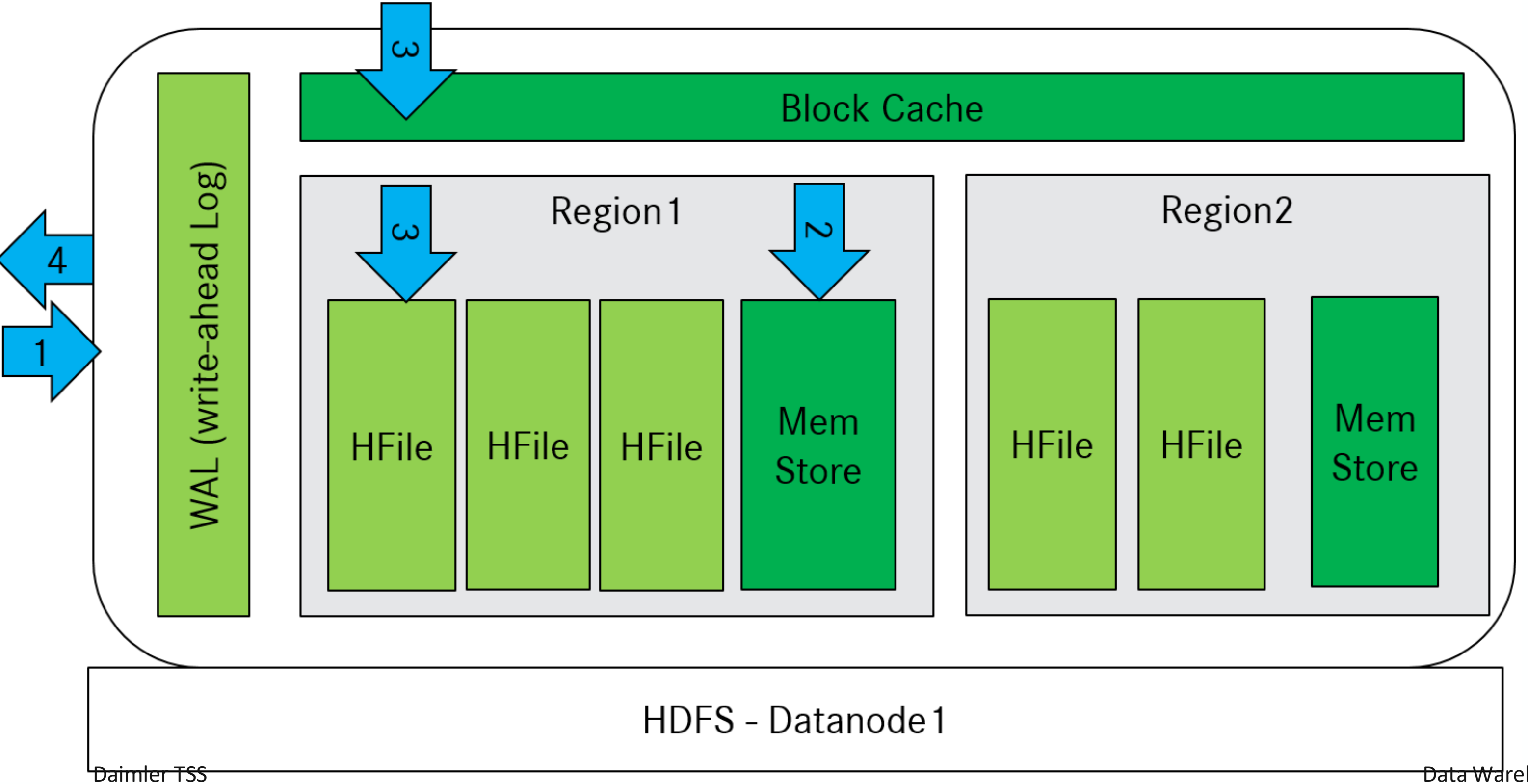
HBase architecture – data distribution



HBase architecture – regionserver writes



HBase architecture – regionserver reads

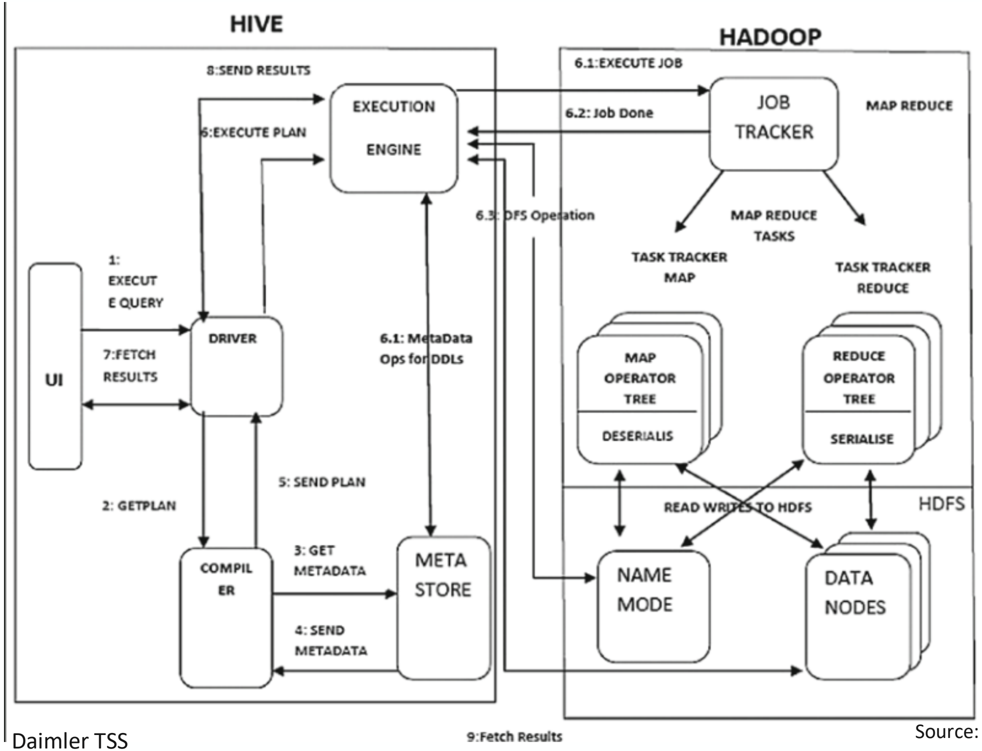


HBase compactions

Merge data files and sort row keys (server stays online)

- Minor
  - Merge HFiles (>= 2) into a new HFile
- Major
  - additionally: Delete data from delete-operations
  - additionally: Delete expired cells

Hive architecture



Source: https://www.researchgate.net/figure/Apache-hive-architecture-9_fig1_319193375

Storage optimization - compression

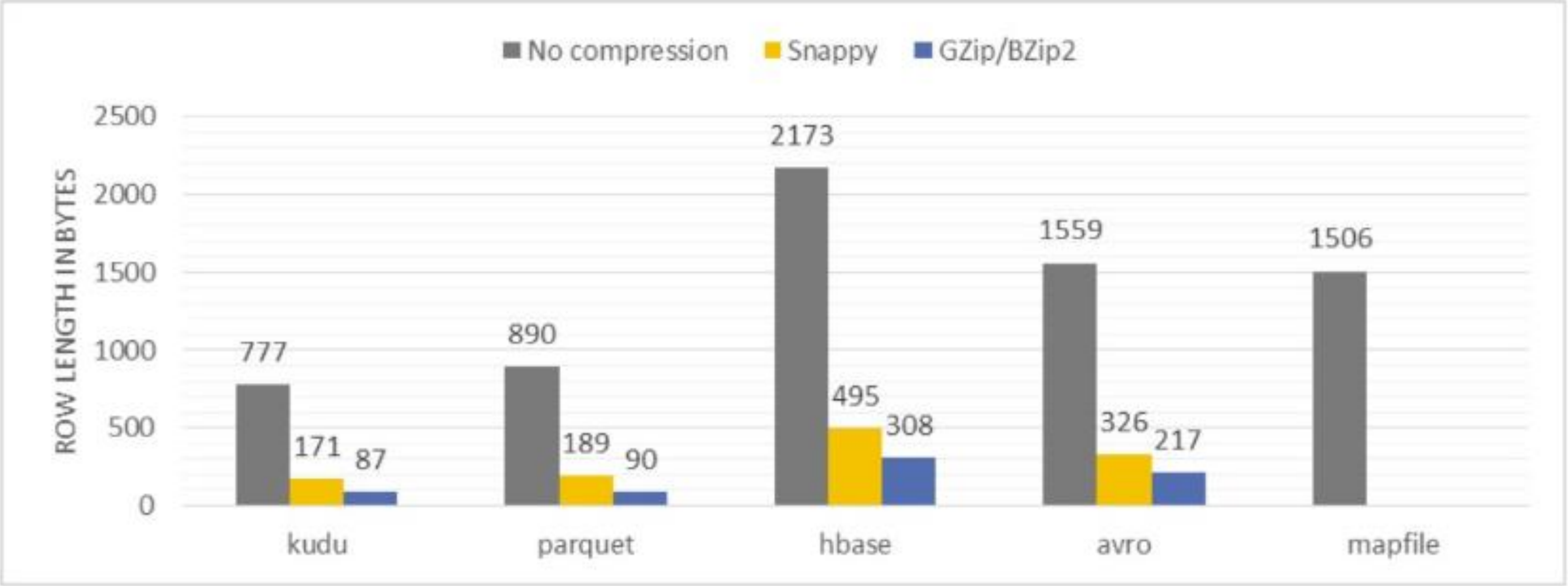
Compression format	Tool	Algorithm	Filename extension	Splittable
DEFLATE[a]	N/A	DEFLATE	*.deflate*	No
gzip	*gzip*	DEFLATE	*.gz*	No
bzip2	*bzip2*	bzip2	*.bz2*	Yes
LZO	*lzop*	LZO	*.lzo*	No[b]
Snappy	N/A	Snappy	*.snappy*	No

[a] DEFLATE is a compression algorithm whose standard implementation is zlib. There is no commonly available command-line tool for producing files in DEFLATE format, as gzip is normally used. (Note that the gzip file format is DEFLATE with extra headers and a footer.) The *.deflate* filename extension is a Hadoop convention.

[b] However, LZO files are splittable if they have been indexed in a preprocessing step. See page 91.

Source: White, Tom - Hadoop The Definitive Guide 3rd Edition - OReilly 2012

Space utilization



The figure reports on the average row length in bytes for each tested format and compression type

Source: https://blog.cloudera.com/blog/2017/02/performance-comparing-of-different-file-formats-and-storage-engines-in-Hadoop-file-system/

Ingestion rate

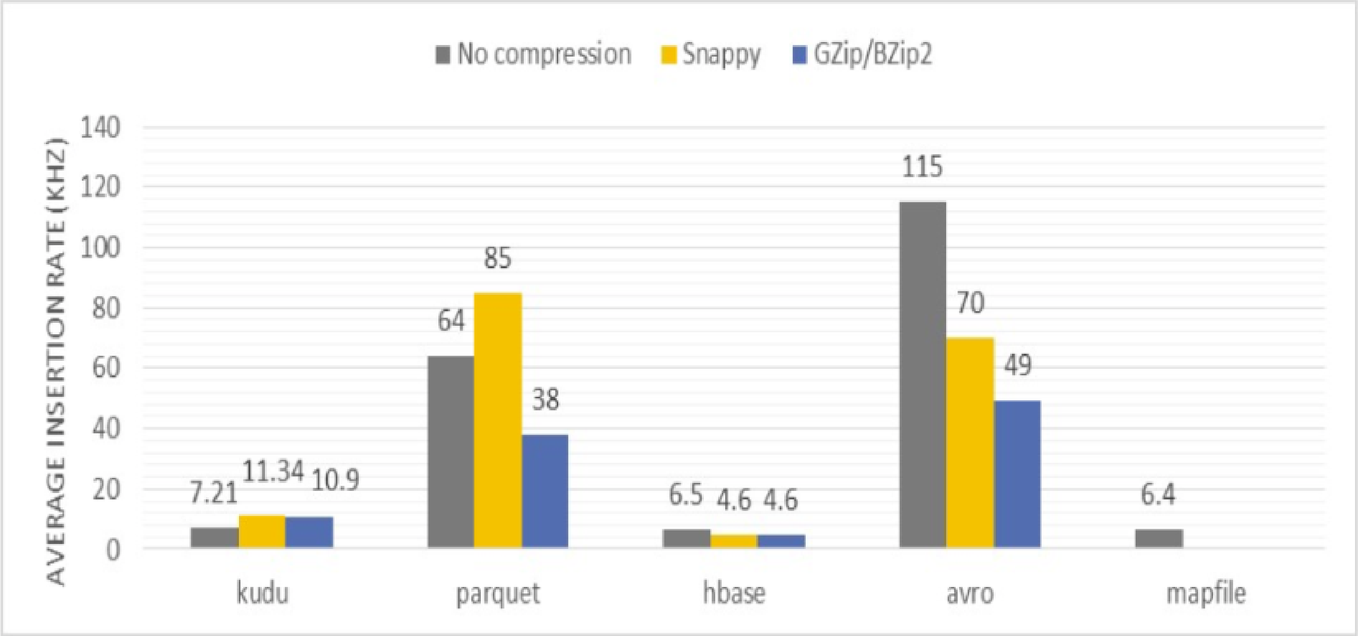


Figure reports on the average ingestion speed (10^3 records/s) per data partition for each tested format and compression type

Source: https://blog.cloudera.com/blog/2017/02/performance-comparing-of-different-file-formats-and-storage-engines-in-Hadoop-file-system/

Random data lookup latency

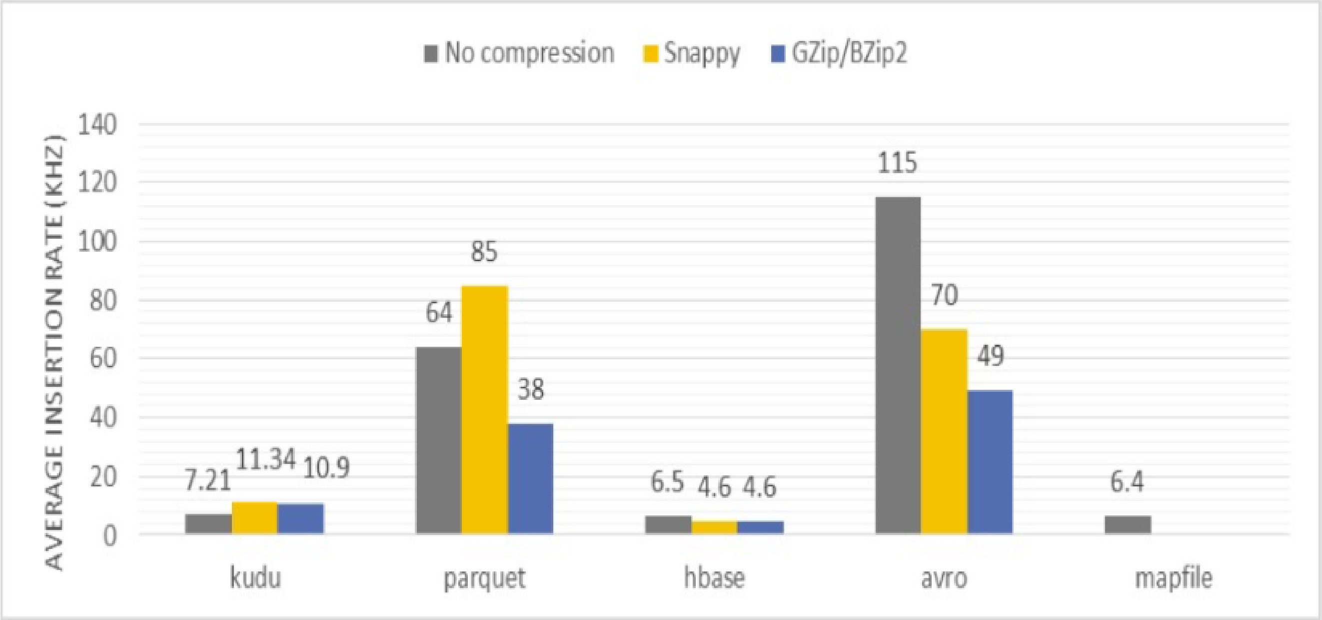


Figure reports on the average ingestion speed (10^3 records/s) per data partition for each tested format and compression type

Source: https://blog.cloudera.com/blog/2017/02/performance-comparing-of-different-file-formats-and-storage-engines-in-Hadoop-file-system/

Data scan rate

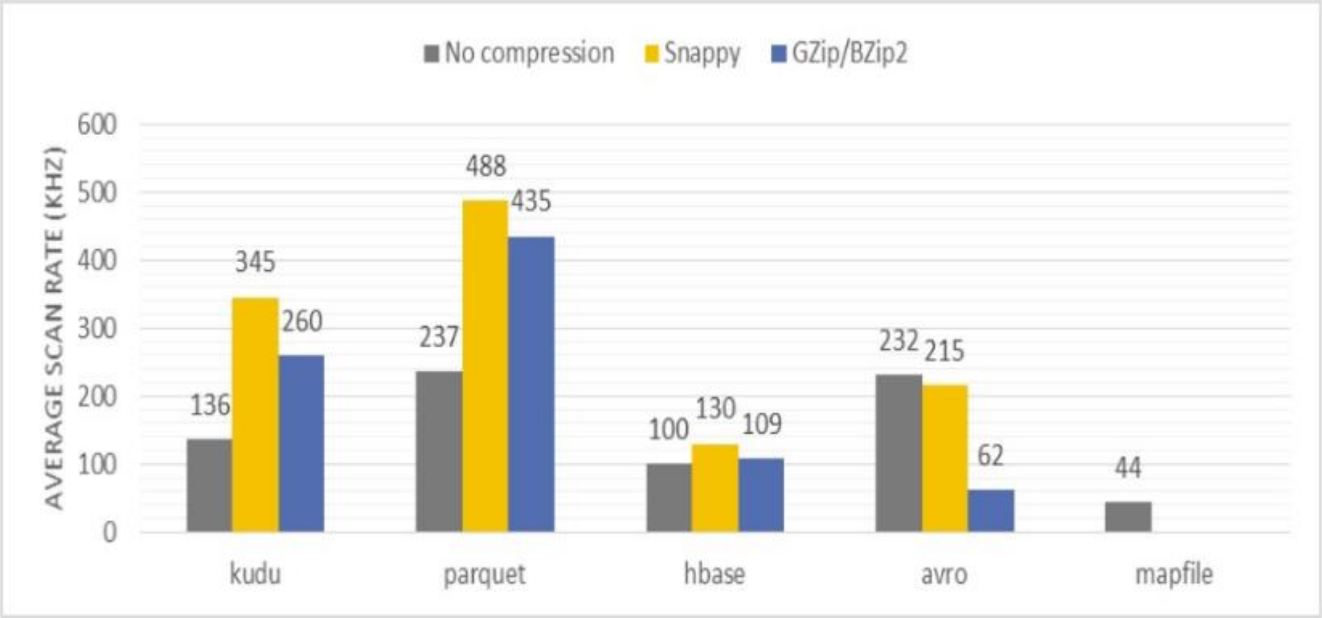
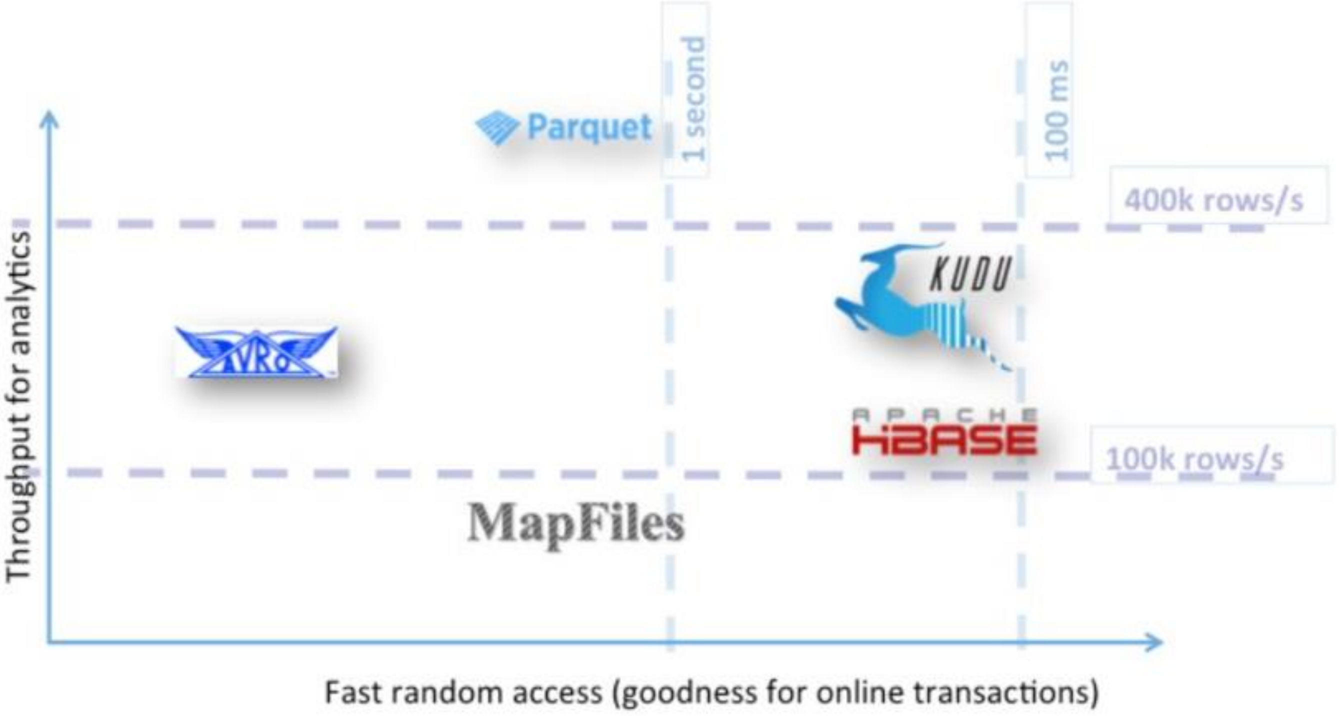


Figure reports on the average scans speed with the same predicate per core [in k records/s] for each tested format and compression type

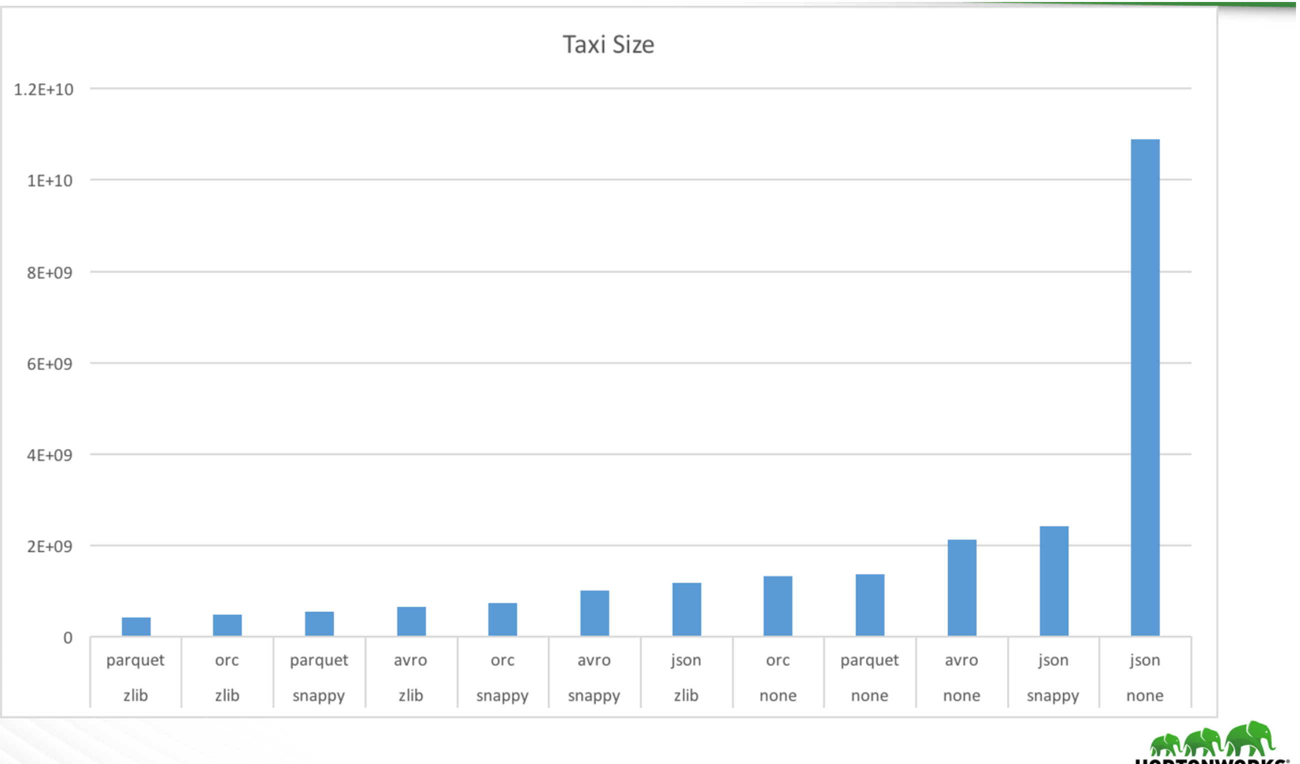
Source: https://blog.cloudera.com/blog/2017/02/performance-comparing-of-different-file-formats-and-storage-engines-in-Hadoop-file-system/

Throughput and latency



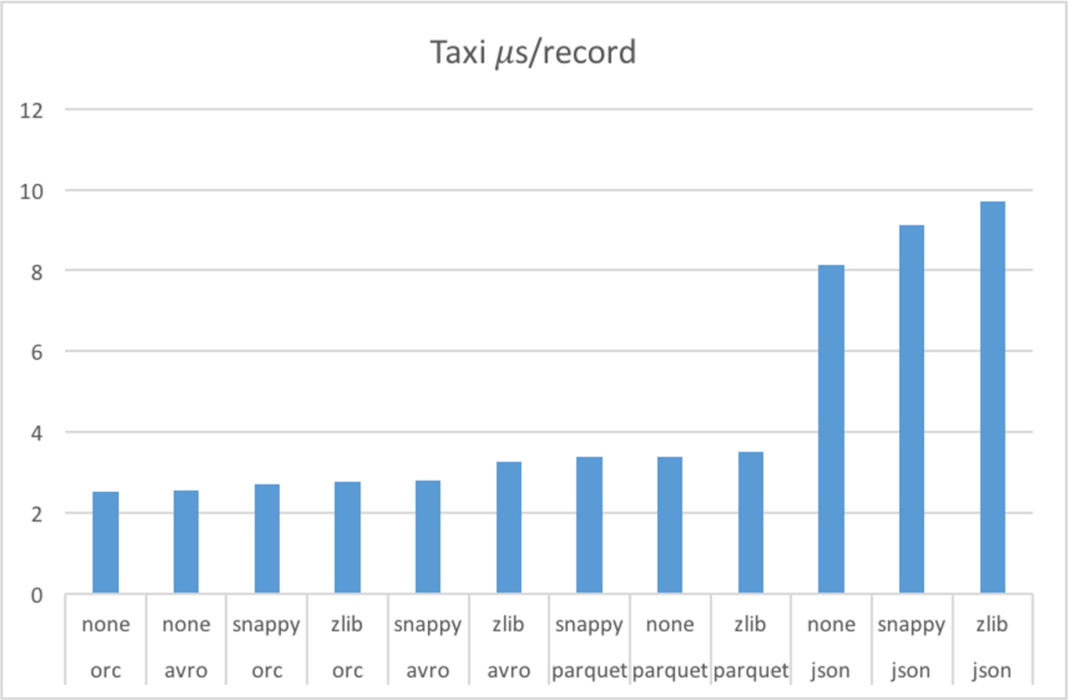
Source: https://blog.cloudera.com/blog/2017/02/performance-comparing-of-different-file-formats-and-storage-engines-in-Hadoop-file-system/

Serde – comparison file size



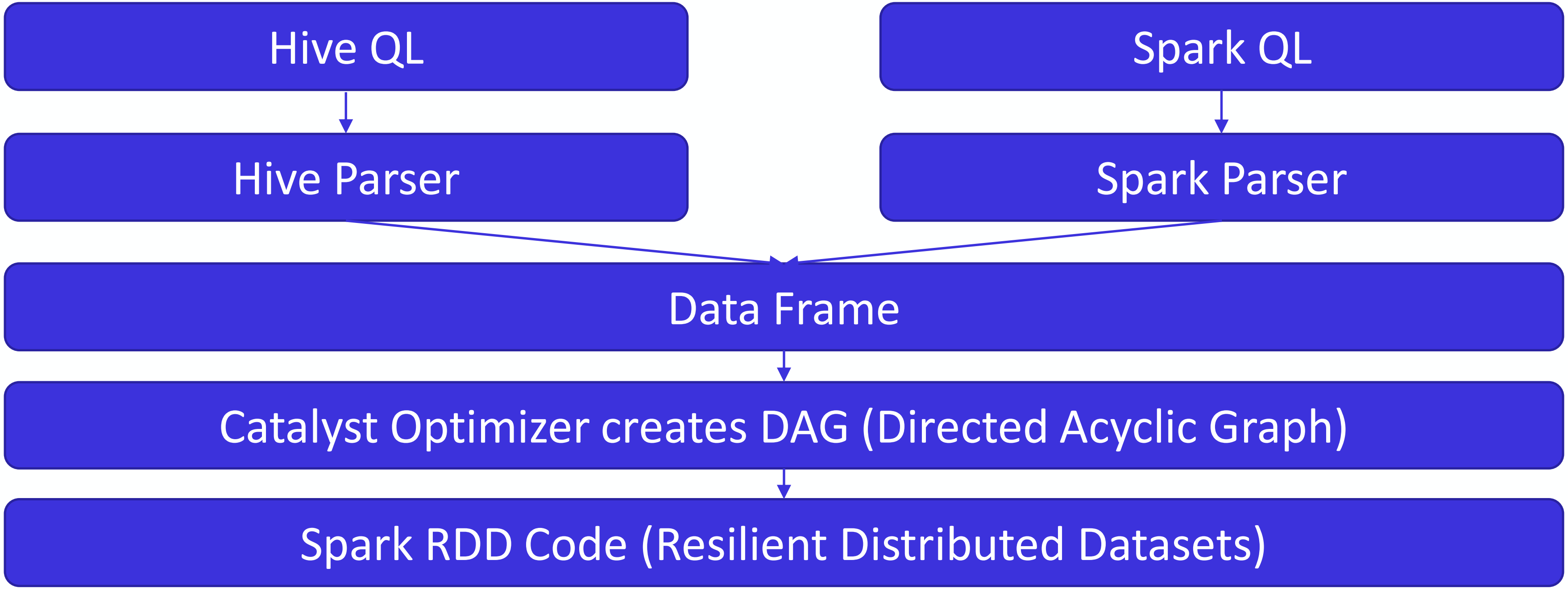
Owen O'Malley: File format benchmark: Avro, JSON, ORC, and Parquet https://conferences.oreilly.com/strata/strata-ny-2016/public/schedule/detail/51952

Serde – comparison read performance



Owen O'Malley: File format benchmark: Avro, JSON, ORC, and Parquet https://conferences.oreilly.com/strata/strata-ny-2016/public/schedule/detail/51952

Data Frames, RDDs and DAGs



RDD and datasets/dataframes

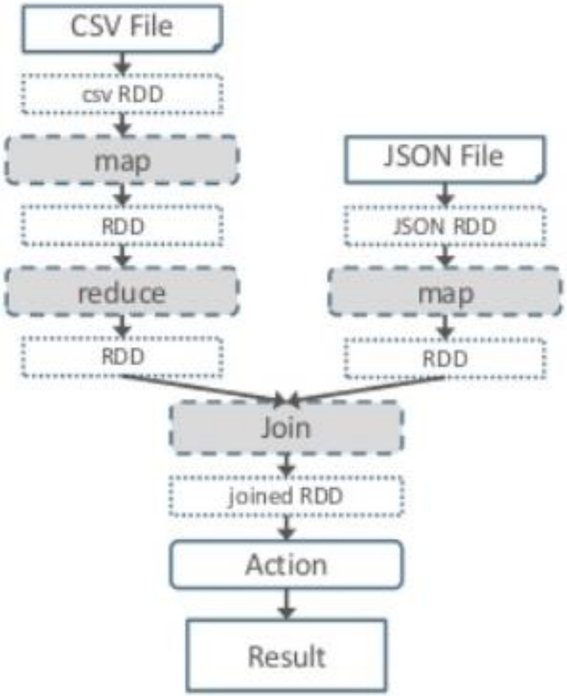
RDD	Dataframe
schemafree	Schema (datasets additionally with data types at compile time)
No Query optimization	Query optimization
Immutable	Immutable

[403,	15.6,	"OK"]
[3668,	16.6,	"OK"]
[3379,	14.8,	"OK"]
[3379,	-999.0,	"ERROR"]
[3668,	13.9,	"OK"]

ID: Integer	Temp: Double	Status: String
403	15.6	OK
3668	16.6	OK
3379	14.6	OK
3379	-999.0	ERROR
3668	13.9	OK

Source: https://www.slideshare.net/JensAlbrecht2/einfuehrung-in-apache-Spark

DAG (Directed acyclic graph)



Source: https://www.slideshare.net/JensAlbrecht2/einfuehrung-in-apache-Spark

Spark operations

Transformation

- A new RDD (or data frame) is produced from the existing one without actual execution e.g. filter, distinct, union, leftOuterJoin
- Lazy evaluation

Action

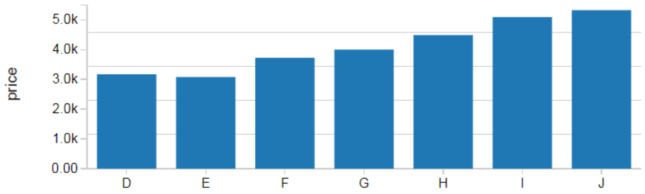
- Triggers execution, e.g. reduce, count, collect

Quickstart Notebook (SQL)

Detached File View: Code Permissions Run All Clear Publish Comments

```
SELECT color, avg(price) AS price FROM diamonds GROUP BY color ORDER BY color
```

(2) Spark Jobs



Plot Options...

Command took 3.82 seconds -- by andreas.buckenhofer@gmail.com at 20.2.2019, 10:11:59 on abu2

Cmd 12

Convert the table to a chart

Under the table, click the bar chart icon.

Cmd 13

Repeat the same operations using Python DataFrame API.

This is a SQL notebook; by default command statements are passed to a SQL interpreter. To pass command statements to a Python interpreter, include the `%python` magic command.

Cmd 14

The next command creates a DataFrame from a Databricks dataset

Cmd 15

```
%python
diamonds = spark.read.csv("/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header="true", inferSchema="true")
```

(2) Spark Jobs

diamonds: pyspark.sql.dataframe.DataFrame = [_c0: integer, carat: double ... 9 more fields]

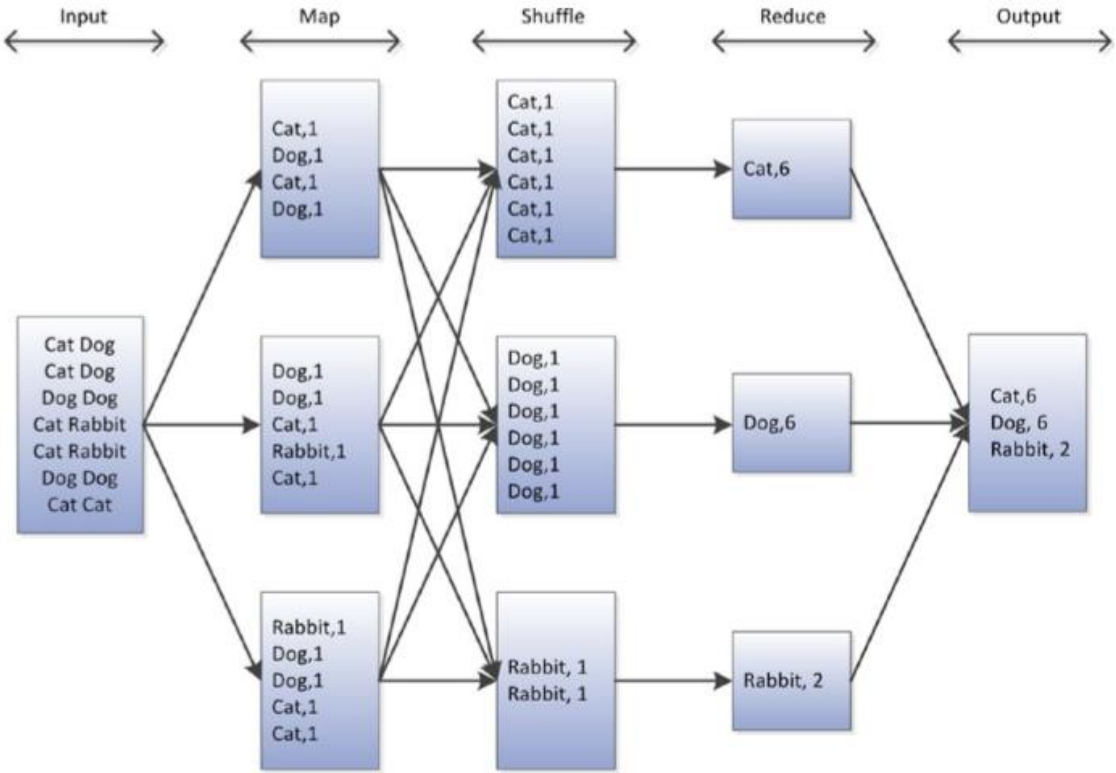
Source: Databricks

Rise of the notebooks for data-driven analytics

Web-based interactive IDE (integrated development environment) well-suited for data-driven tasks

- Popular notebooks: Jupyter, Apache Zeppelin
- Notebooks became very popular for data-driven data analytics and easy sharing
- Interpreters typically available for
    - Embed code in SQL, Python, Java, Scala and inspect data

Map reduce parallel processing framework



Source: Harrison: Next Generation Databases, Apress 2016

Map reduce sample code

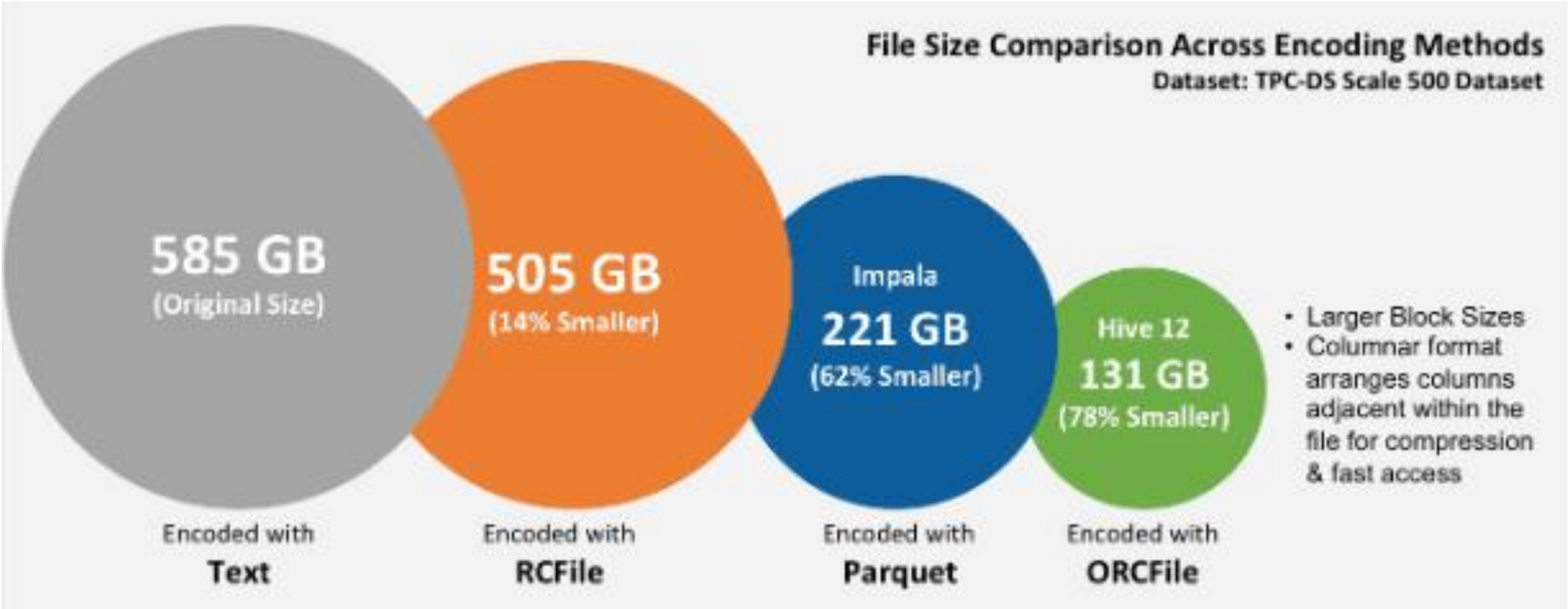
```
public void map(LongWritable key, Text val,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  if (pattern.matcher(val.toString()).matches()) {
    output.collect(val, new IntWritable(1));
  }
}

public void reduce(Text key, Iterator<IntWritable> vals,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  int sum = 0;
  while (vals.hasNext()) {
    sum += vals.next().get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Map Reduce is rarely used nowadays as it became too slow for recent requirements

Serde – serialization and deserialization

File format	Description	Code generation	Schema evolution	Splittable Compression	Apache Hive support
AVRO	row storage format	optional	Yes	Yes	Yes
PARQUET	columnar storage format	No	Yes	Yes	Yes
ORCFILE	columnar storage format	No	Yes	Yes	Yes
PROTOCOL BUFFER	originally designed by Google with interface description language to generate code	Optional	Yes	No	No
THRIFT	data serialization format designed at Facebook similar to PROTOCOL BUFFER	mandatory	Yes	No	No

File format does matter



Source: http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/

MapReduce & Tez engine comparison

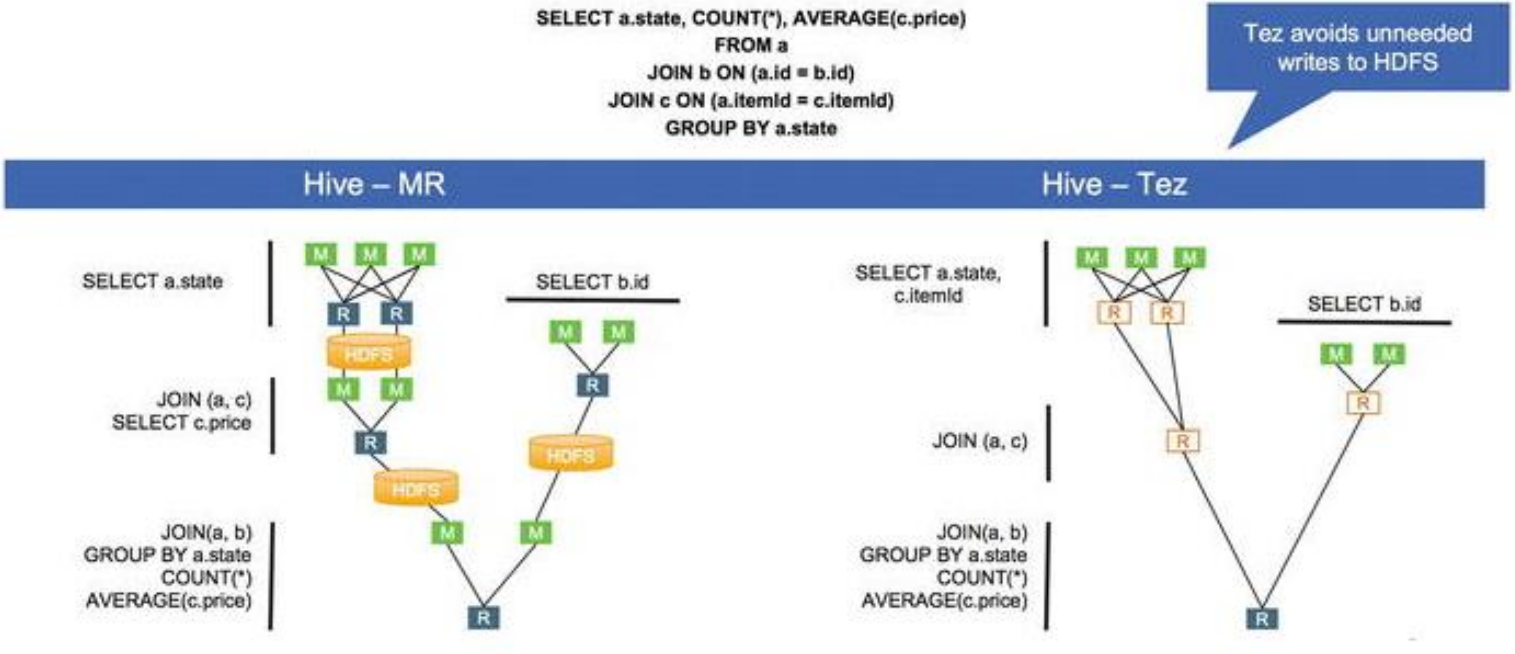
```
SELECT  origin, COUNT(*) as cnt
  FROM flights f JOIN airports a ON (f.origin = a.code)
                JOIN weather w ON (a.station = w.station AND w.year = f.
             year AND w.month = f.month and w.day=f.day)
  WHERE f.depdelay>15 and w.metric = 'AWND' and w.value>10
  GROUP by origin SORT BY cnt DESC LIMIT 5;
```

```
MapReduce Jobs Launched:
Stage-Stage-11: Map: 6   Cumulative CPU: 233.33 sec   HDFS Read: 164317688 HDFS Write: 71108
Stage-Stage-2: Map: 13  Reduce: 50   Cumulative CPU: 1438.11 sec   HDFS Read: 3278981109 HDF
Stage-Stage-3: Map: 4  Reduce: 1   Cumulative CPU: 15.57 sec   HDFS Read: 292269 HDFS Write:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 3.89 sec   HDFS Read: 10052 HDFS Write: 2
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 4.05 sec   HDFS Read: 4787 HDFS Write: 57
Total MapReduce CPU Time Spent: 28 minutes 14 seconds 950 msec
OK
ORD     1297377
ATL     1112511
DFW     933903
LAX     626875
PHX     584062
Time taken: 475.732 seconds, Fetched: 5 row(s)
```

```
ution.engine=tez;
set hive.prewarm.enabled=true;
set hive.prewarm.numcontainers=10;
Total jobs = 1
Launching Job 1 out of 1

Status: Running (Executing on YARN cluster with App id application_1457719973622_0118)

--------------------------------------------------------------------------------
VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..... SUCCEEDED     22         22        0        0       0       0
Map 5 ..... SUCCEEDED      1          1        0        0       0       0
Map 6 ..... SUCCEEDED     29         29        0        0       0       0
Reducer 2 ..SUCCEEDED     28         28        0        0       0       0
Reducer 3 ..SUCCEEDED     14         14        0        0       0       0
Reducer 4 ..SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 06/06  [==========================>>] 100%  ELAPSED TIME: 141.23 s
--------------------------------------------------------------------------------
OK
ORD     1297377
ATL     1112511
DFW     933903
LAX     626875
PHX     584062
Time taken: 166.448 seconds, Fetched: 5 row(s)
```

Source: Vermeulen, Gupta, et al: Practical Hive: A Guide to Hadoop's Data Warehouse System, Apress, 2016, chapter 9

MapReduce & Tez engine comparison
Execution plans



Source: Vermeulen, Gupta, et al: Practical Hive: A Guide to Hadoop's Data Warehouse System, Apress, 2016, chapter 9

Storage optimization – serialization and deserialization formats

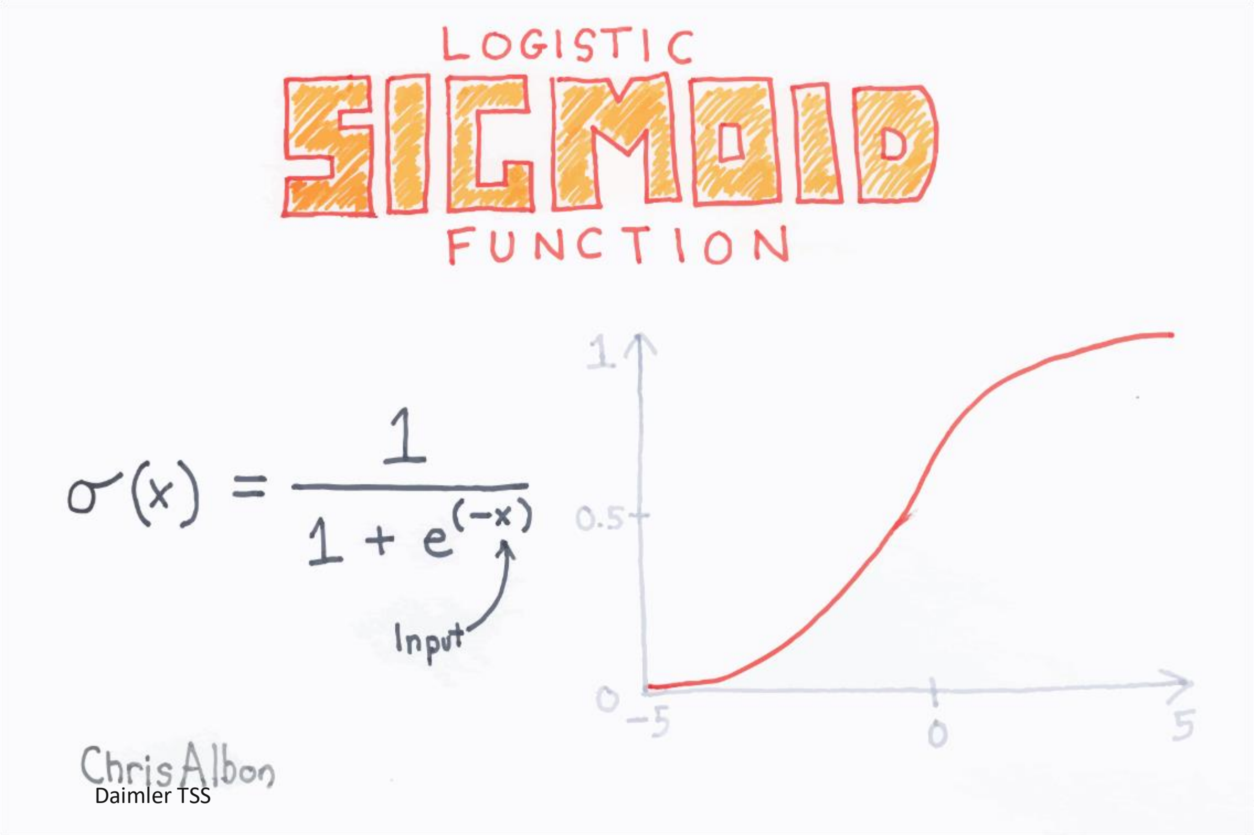
- CSV / JSON / XML
  - text-based formats
- Avro
  - Stores data definition (as JSON) and data in one file; widely used
- Parquet
  - column oriented data serialization standard for efficient data analytics
  - Stores data definition and data in one file
- RCFile, ORCFile, Protocol Buffers (invented by Google), Sequence Files, etc

HDFS interfaces

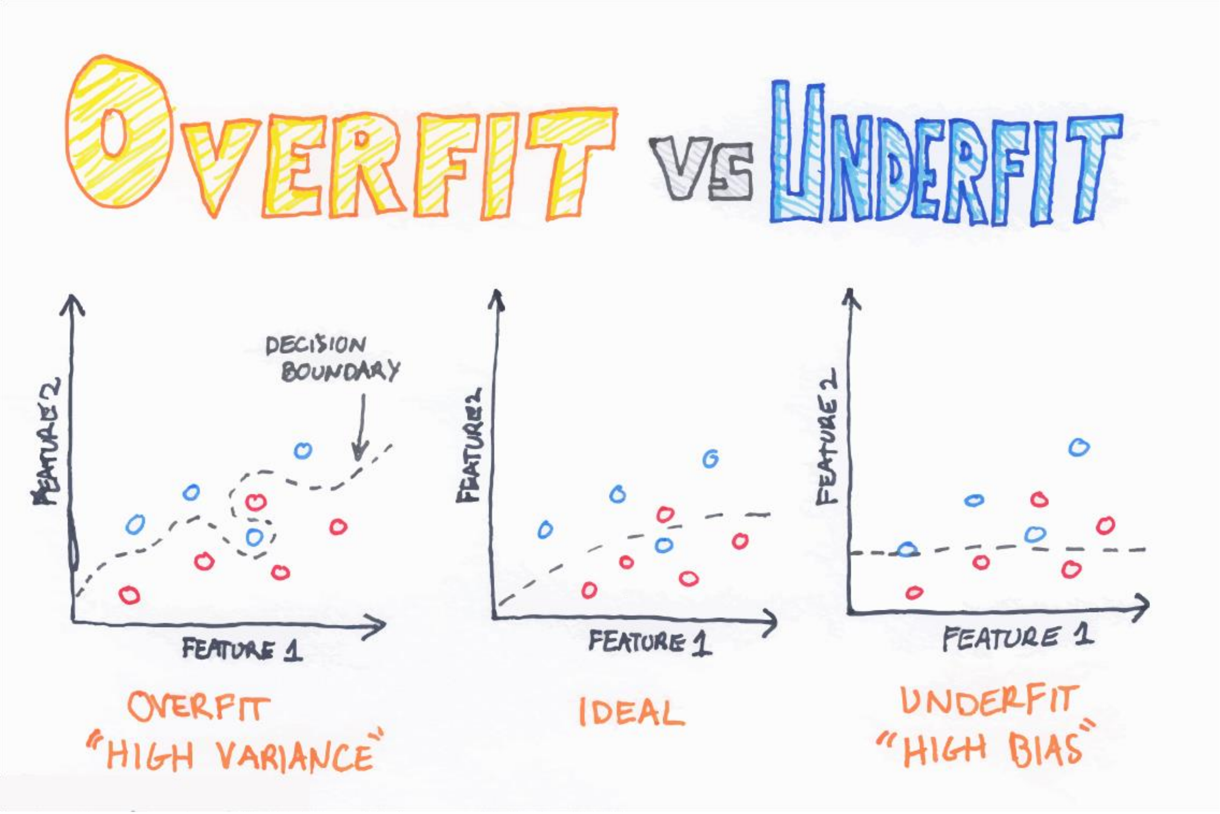
- Command line
- Java API
- Web Interface

Source: https://pdfs.semanticscholar.org/presentation/e67d/6df768eb171e1750b8a613884b193bf486e2.pdf

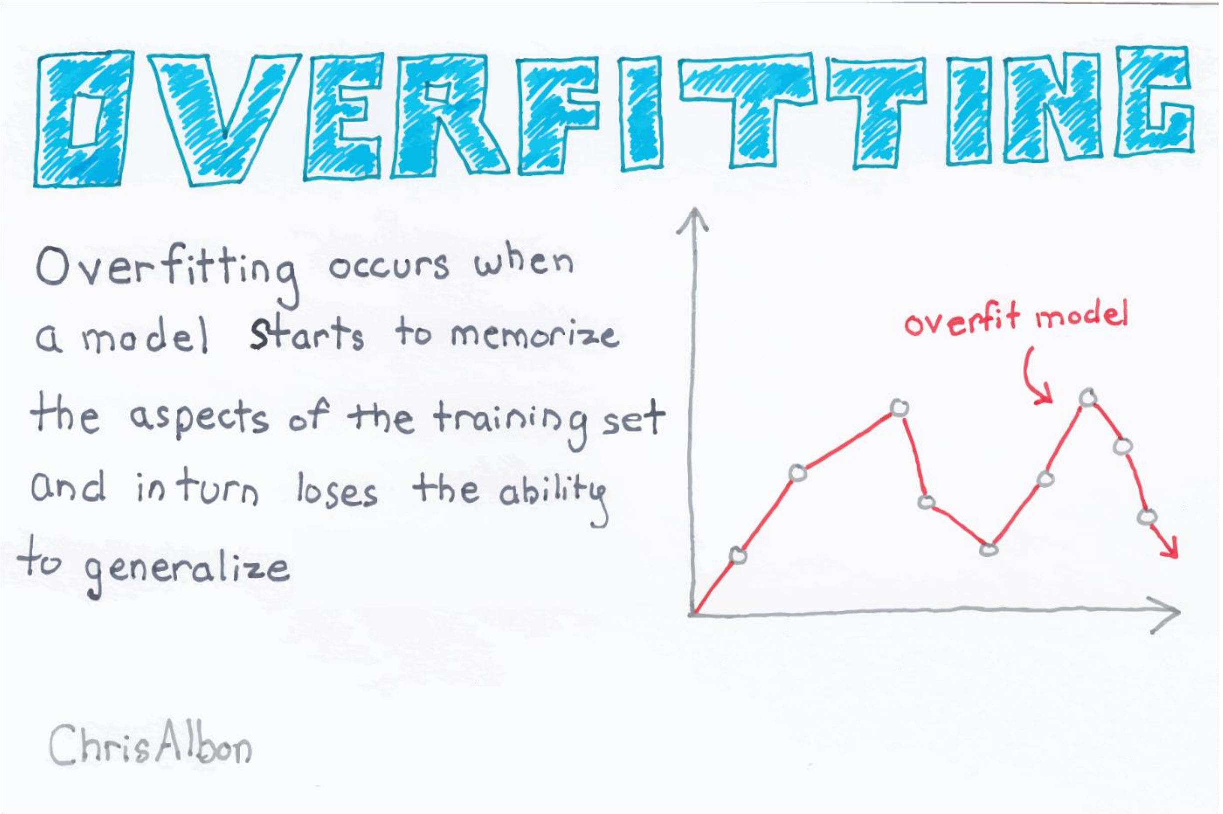
Flashcards-1



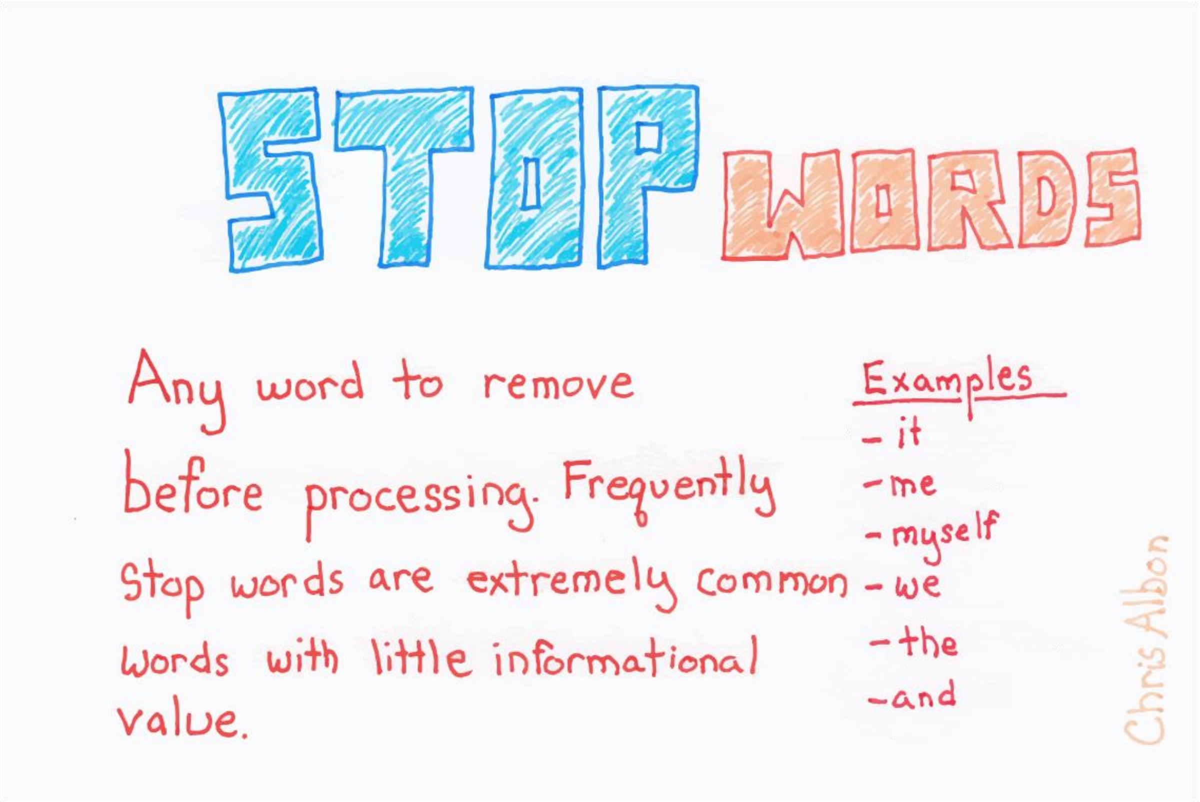
Flashcards-2



Flashcards-3



Flashcards-3



Data Model

<u>3 levels of representation</u>

- Conceptual business modeling
- Logical database design
- Physical implementation

Data model, Formal definition [Codd]

- Data structures
- Data integrity
- Data manipulation

E.g. Relational Data Model (RDM)

- Relations on domains
- Constraints
- Relational algebra

Beware of

- Conceptual-logical conflation (CLC)
- Logical-physical confusion (LPC)

Conceptual modelers can't use a data model for end-user requirements as it is abstract
Database designers derive a logical design (data model) from facts (conceptual model)

What is a DWH [Inmon]?

Subject-oriented
Data is organised along the lines of
the major entities

Time-variant
Any record in the DWH is
accurate relative to some moment
in time

Integrated
Physical unification and cohe-
siveness of the data

Nonvolatile
Changes are captured in the form
of a time-variant snapshot:
no deletes, no updates

Comprised both of detailed and summary data

JSON and SQL

{object}

Key: value pair

[array]

```
{
  "customer": {
    "firstName": "Jim",
    "lastName": "Bush",
    "interests": [
      "Travel",
      "Sports",
      "Photography"
    ]
  }
}
```

ISO SQL Standard 2017, Part 6

SQL Query-Functions:

- JSON_EXISTS
- JSON_VALUE
- JSON_QUERY
- JSON_TABLE