The Data Lake and OLAP

Rahul Dave(@rahuldave), Univ.Ai

Data Engineering Lifecycle

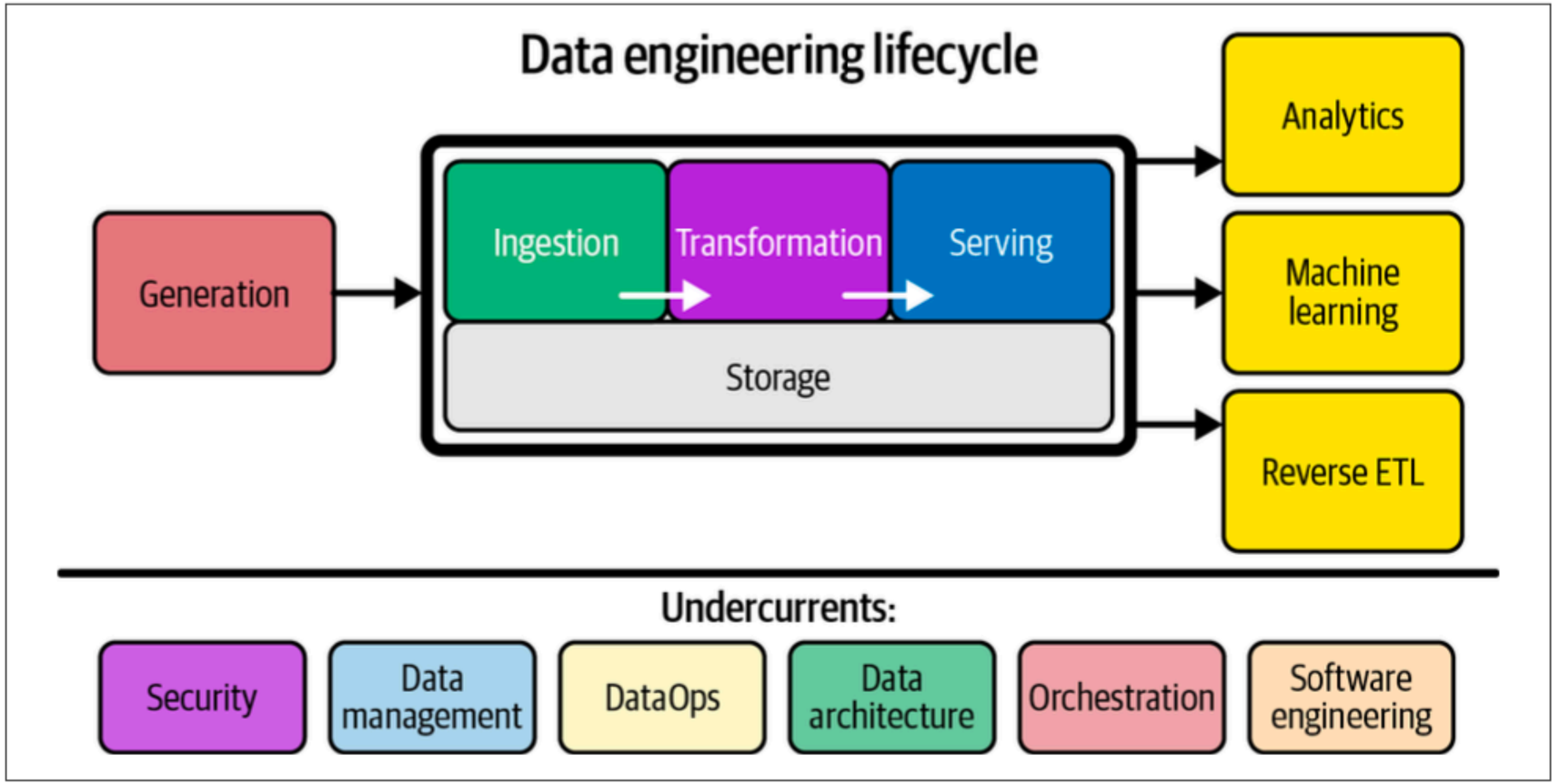
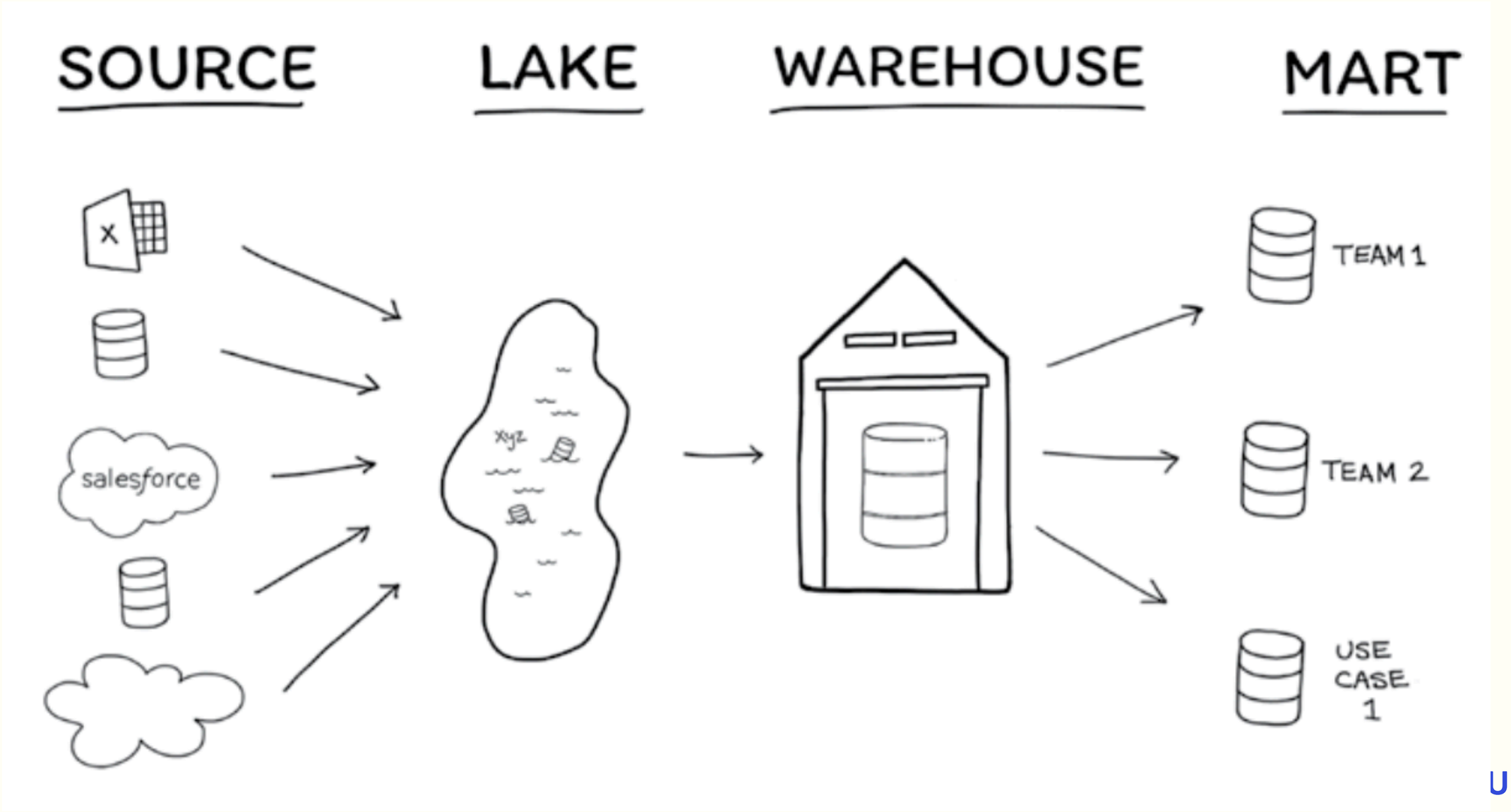


Figure 1-1. The data engineering lifecycle

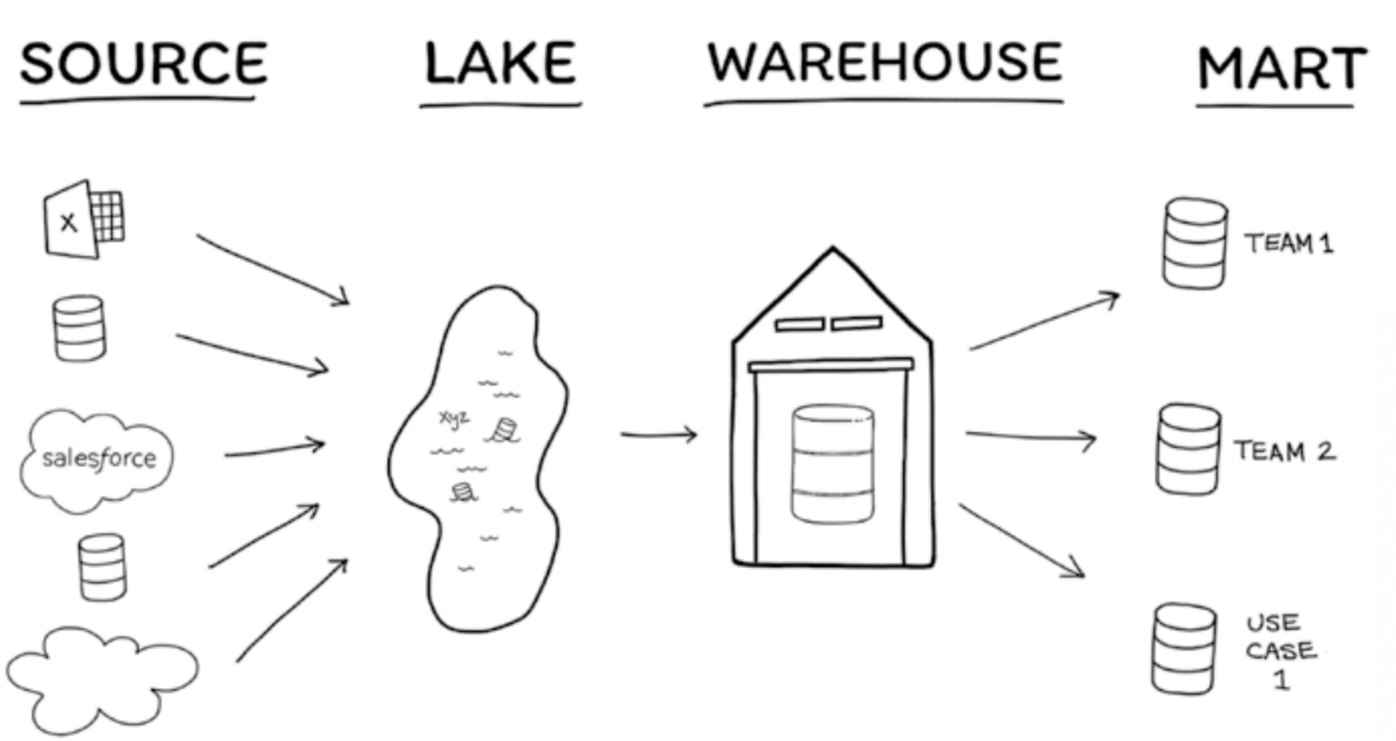
The Arrival of Hadoop

- In 2003, Google published the GFS paper, and in 2004 the Map-Reduce Paper
- Hadoop was contributed by Yahoo in 2006
- The big difference with Hadoop was that one brought the compute to the data and spread the storage out: this was the genesis of HDFS
- The disadvantage was that people now needed to maintain Hadoop/HDFS clusters and this was a lot of work in itself
- Until the cloud: now you can spin up on-demand HDFS clusters for a computation on AWS EMR with the original data on AWS S3.

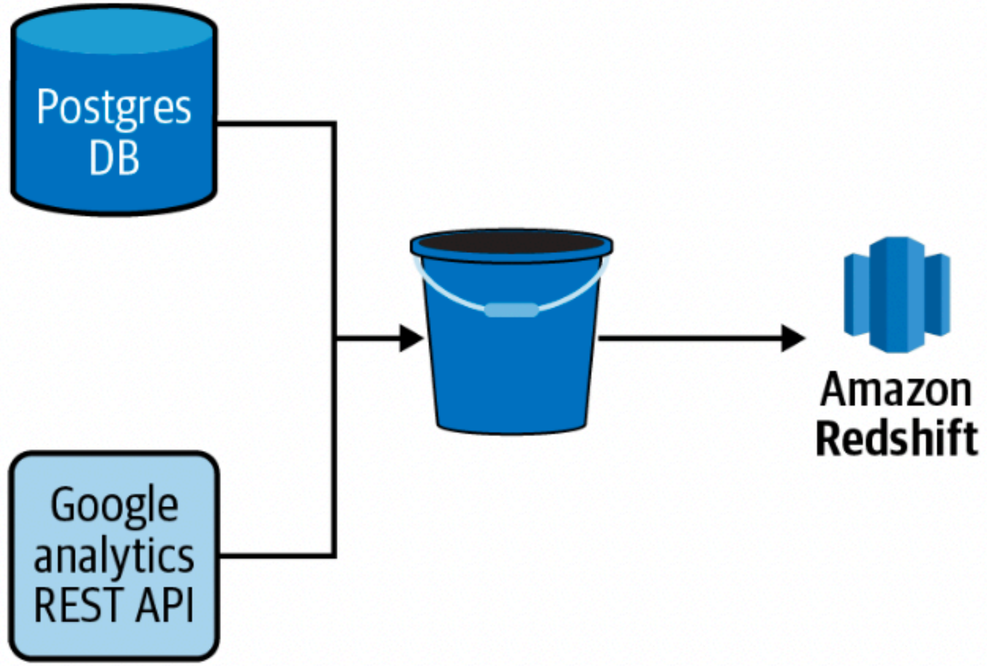
Introducing the Lake



The Lake is usually built on Object Storage.



A lake is often a useful intermediary. All kinds of data: tabular/image/text can be dumped there. It can also be used for post warehousing data outputs for downstream pursuits such as machine learning.



Analysis using non-SQL code such as using SPARK can also be carried out in the data lake.

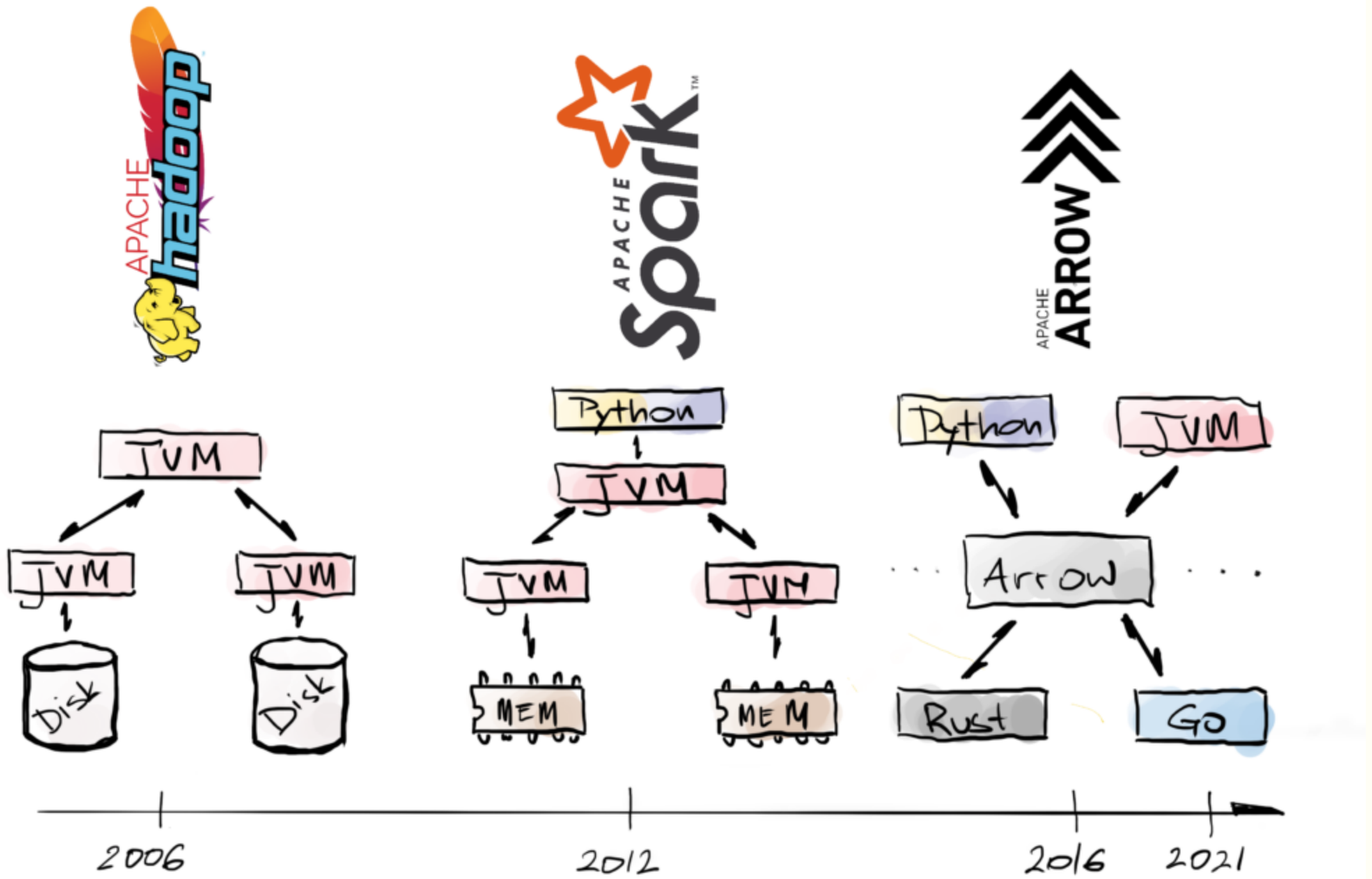
SQL on the lake

- The trend was started by facebook with Hive, and then Presto. SparkSQL implemented SQL in the spark ecosystem
- Now you have many choices including duckdb, Presto, trino, sparksql, dremio. SQL capability is a MUST for any new engine.
- Meanwhile Redshift can now query parquet files in S3 and Snowflake is built on a micro-partioned architecture. Duckdb has its own database format but will query parquet and CSV, just like SPARK does.
- Thus there is a great convergence towards SQL everywhere

To SQL or Not to SQL(Spark).
Thats the question...

- With object storage and SQL-on-file as base layer it is not entirely clear what is in the database and what is not
- There are two kinds of ELT: into warehouse, and into lake and (possibly) then with some T into warehouse (a kind of ETL).
- Dont focus on ETL/ELT, focus on the needs of the data
- We should think about SQL workflows vs non-SQL workflows instead. What transformations would be hard to express with SQL? There you want to use Spark and PySpark.

Data Processing Ecosystem Evolution



Hadoop

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Transactions	ACID	None

The idea

- provide a system for batch processing, using “map-reduce” semantics. This is a parallel system, and as long as the map step does a fair bit of computation, “mapped” to each node, we can scale linearly.

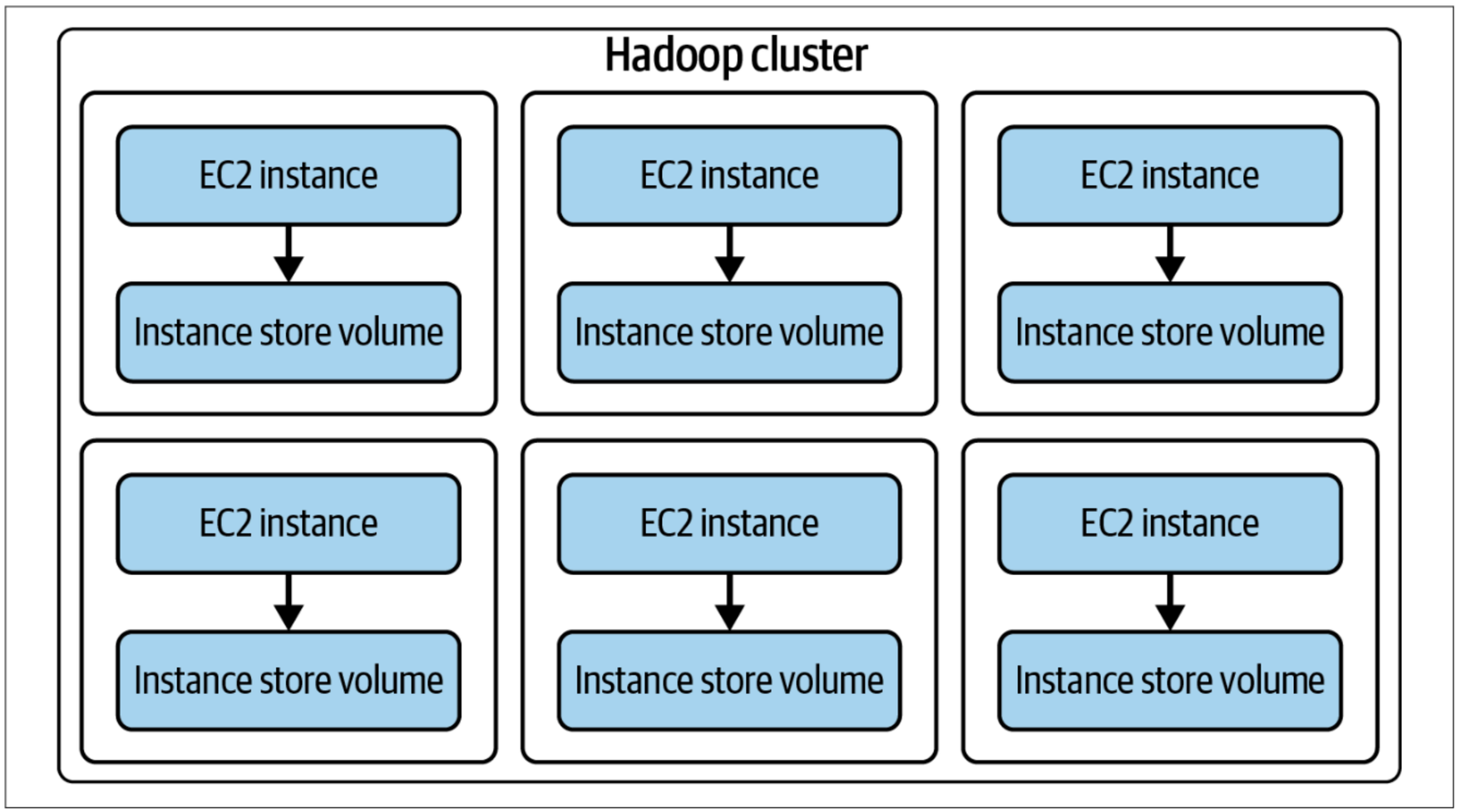
- the underlying storage is distributed over a cluster and called hdfs. The idea is to colocate compute and storage, to make things fast.

- Applications are built on this, like hbase, a distributed key-value store. This is an “online” system, providing online read-write access of individual rows and batch access for bulk reading and writing.

- YARN is the cluster “manager” which decides where jobs can be run depending upon the resources of individual nodes

- The implementation detects failures and schedules replacements on healthy machines. This works because the architecture of map-reduce is shared-nothing.

Distributed Computing



Keys are character line offsets

The map function acts on each key, here, and extracts years and temps

```
0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
0043012650999991949032418004...0500001N9+00781+99999999999...
```

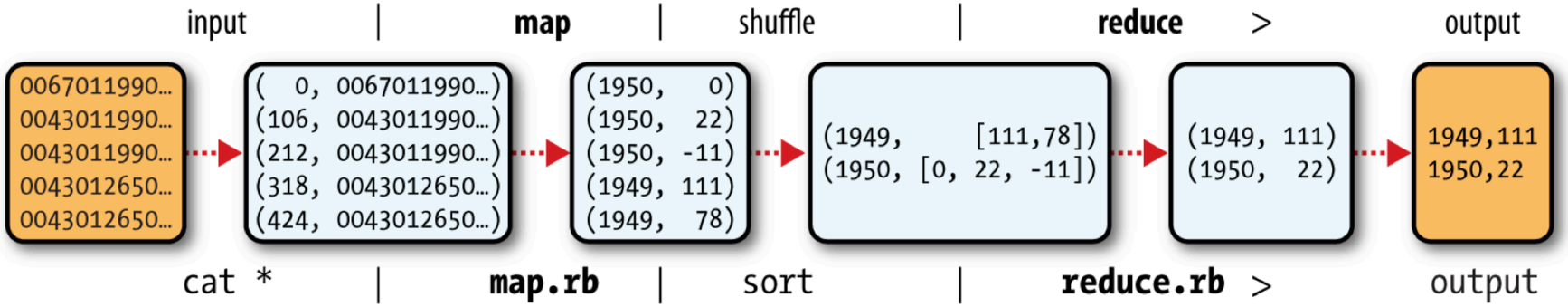
These lines are presented to the map function as the key-value pairs:

```
(0, 0067011990999991950051507004...9999999N9+00001+99999999999...)
(106, 0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-00111+99999999999...)
(318, 0043012650999991949032412004...0500001N9+01111+99999999999...)
(424, 0043012650999991949032418004...0500001N9+00781+99999999999...)
```

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```

The mapreduce framework collates the temps by year into a list and sorts them.

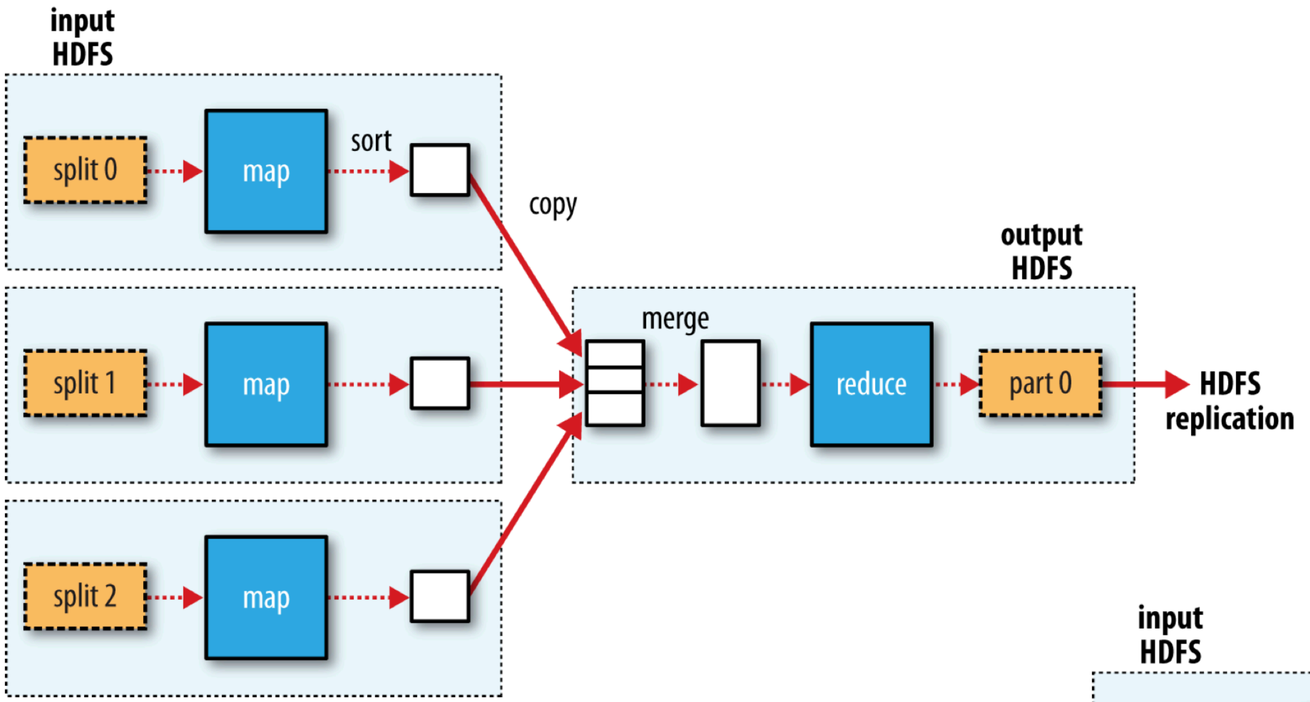
```
(1949, [111, 78])
(1950, [0, 22, -11])
```



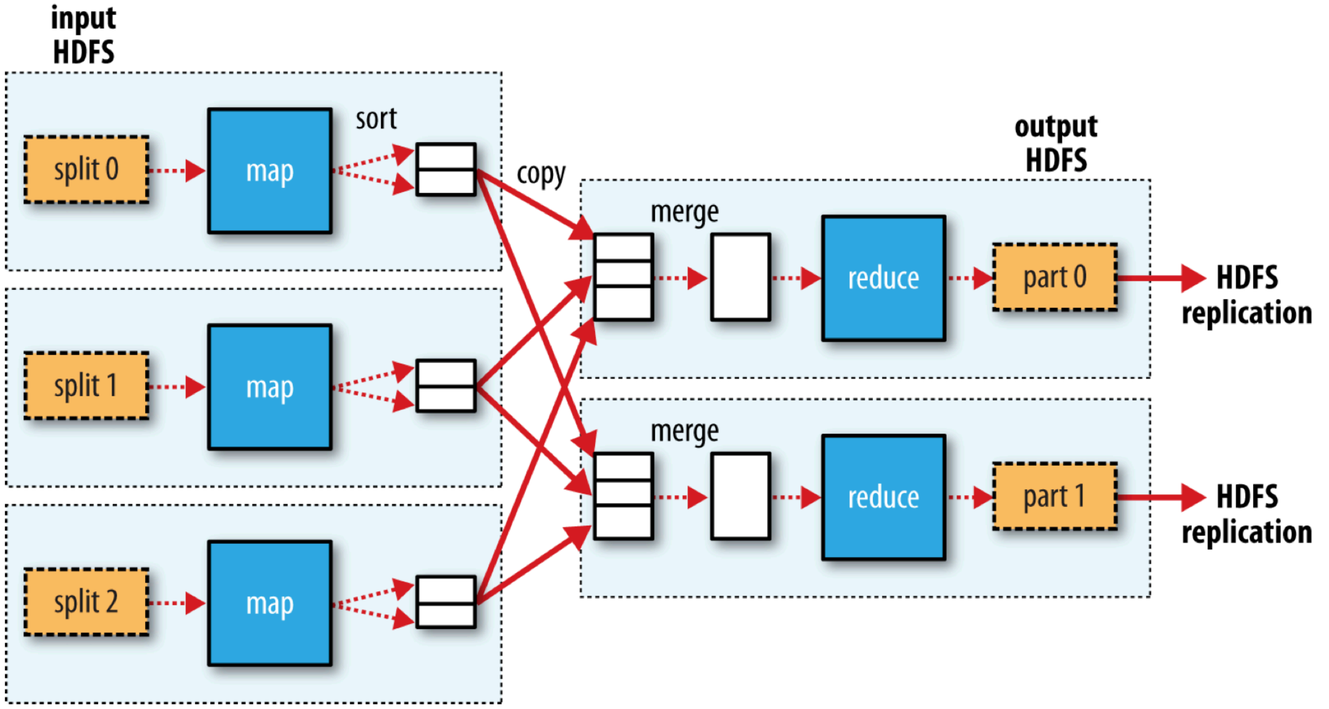
The reduce function gets the max over the list.

```
(1949, 111)
(1950, 22)
```

v.AI



Map-reduce flow with a single reduce function.



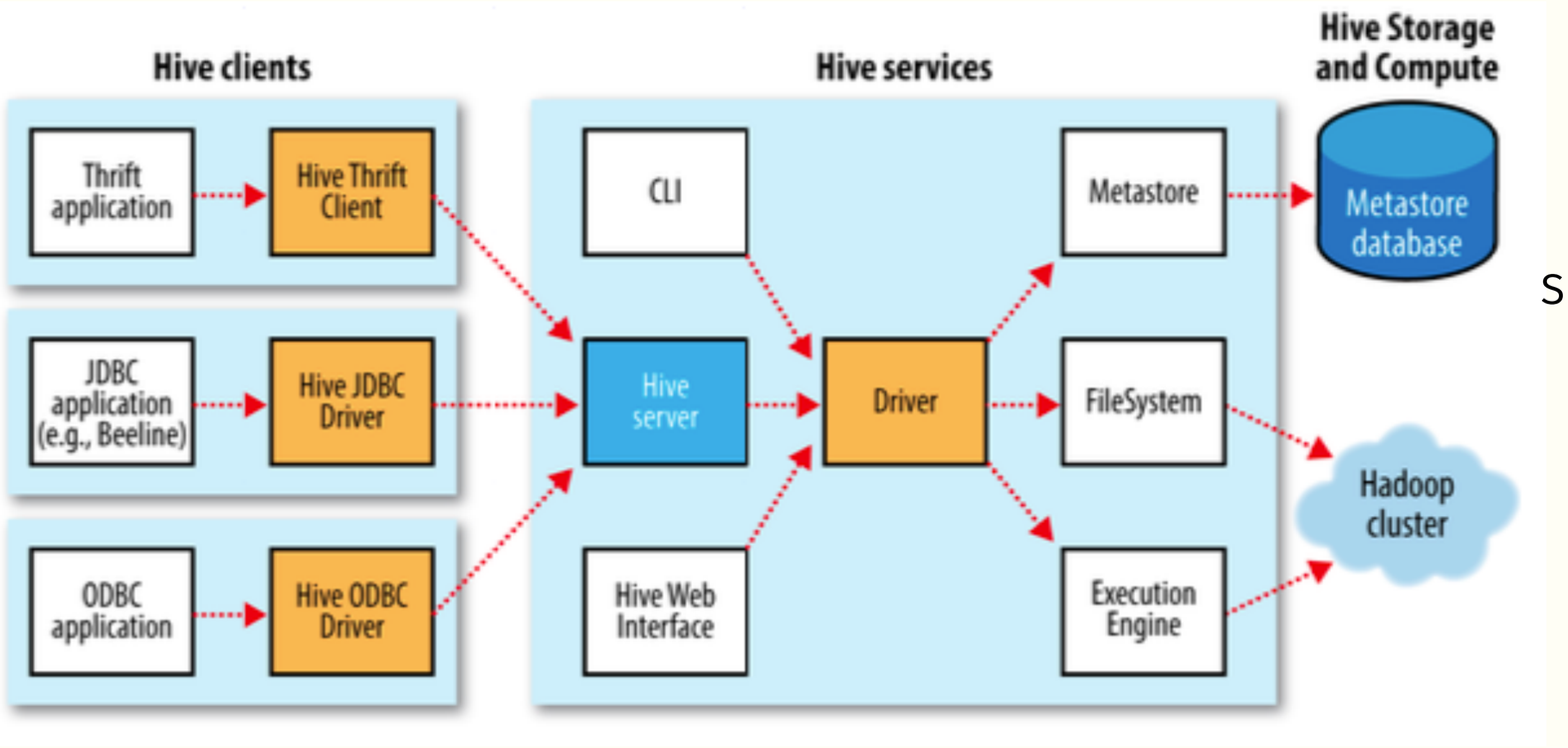
Map-reduce flow with multiple reduce functions.

Hive

- Hive was created by Jeff Hammerbacher et. al at Facebook
- Hive was created so that analysts with good SQL skills but poor to non-existent java skills could make queries on the huge amount of data that facebook had
- Hive consists of a local client that converts your sql queries to mapreduce jobs that are run on a Hadoop cluster. It organizes data in HDFS into tables
- A key component, one that we will see later is the metastore which stores metadata such as table schemas
- Hive can also use Apache Tez and Apache Spark.

```
CREATE TABLE records (year STRING, temperature INT, quality INT)
    ROW FORMAT DELIMITED
      FIELDS TERMINATED BY '\t';
```

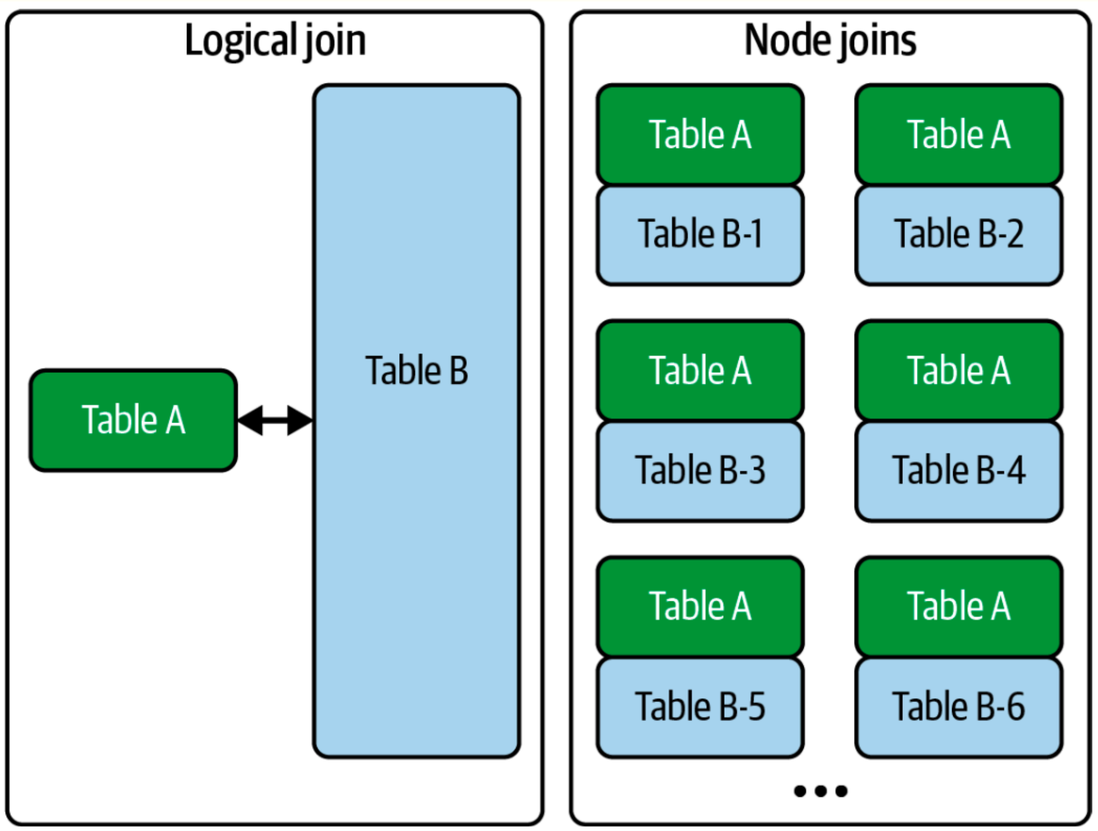
```
LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample.txt'
    OVERWRITE INTO TABLE records;
```



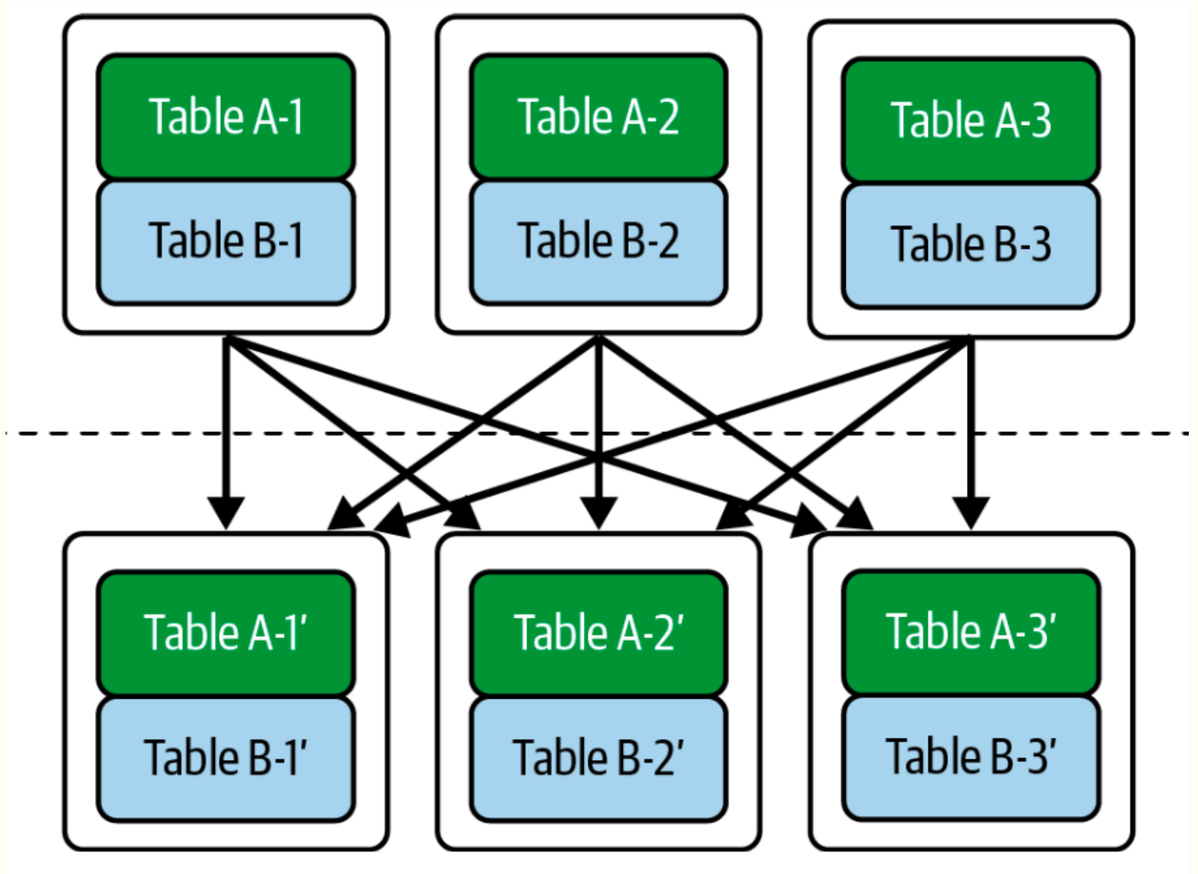

```
SELECT station, year, MAX(temperature)
    FROM valid_records
    GROUP BY station, year;
```

Cluster joins

Distributed joins break a logical join into node joins.



Broadcast join: when one table is much smaller



Shuffle-hash-join: one strategy when table A is too large

More from Hive

```
/user/hive/warehouse/logs
├── dt=2001-01-01/
│   ├── country=GB/
│   │   ├── file1
│   │   └── file2
│   └── country=US/
│       └── file3
└── dt=2001-01-02/
    ├── country=GB/
    │   └── file4
    └── country=US/
        ├── file5
        └── file6
```

```
CREATE TABLE logs (ts BIGINT, line STRING)
PARTITIONED BY (dt STRING, country STRING);
LOAD DATA LOCAL INPATH 'input/hive/partitions/file1'
INTO TABLE logs
PARTITION (dt='2001-01-01', country='GB');
```

```
CREATE TABLE bucketed_users
(id INT, name STRING)
CLUSTERED BY (id) SORTED BY
(id ASC) INTO 4 BUCKETS;
```

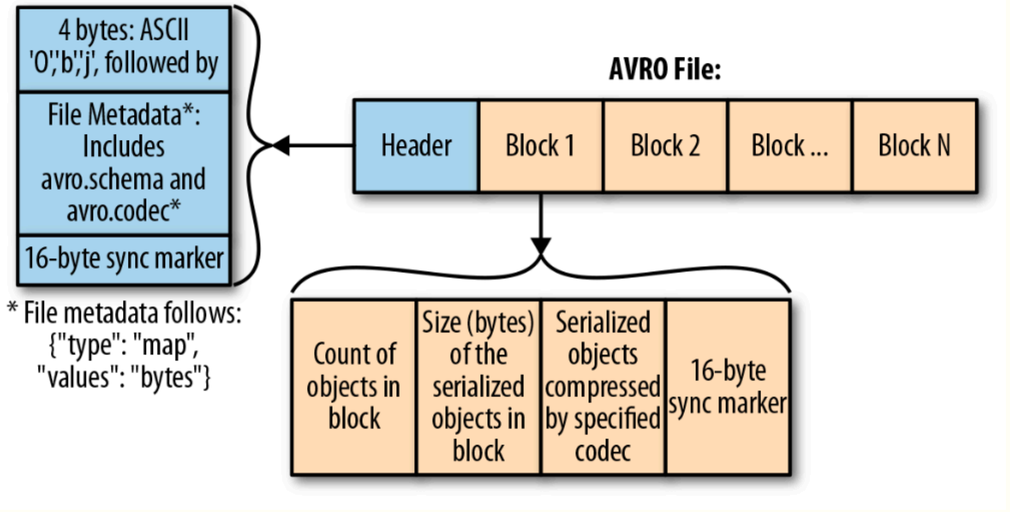
- Another key contribution from Hive is the notion of partitions, where the data is split according to some attribute.
- Partitions are made at table creation time by using the PARTITIONED BY clause. At load time the partitions are specified explicitly.
- you can also bucket a table (eg after partitioning it)
- The hive metastore was the basis of metadata catalog systems present in every data engineering system and in modern lake/lakehouse formats such as delta-lake and iceberg.

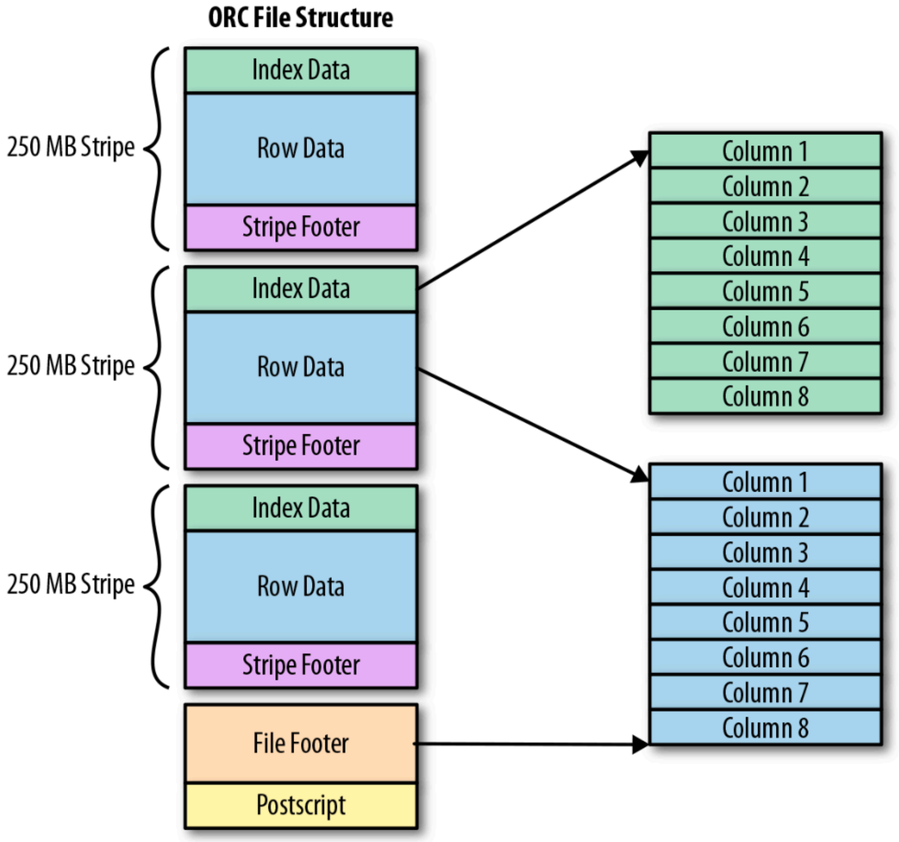
File Formats

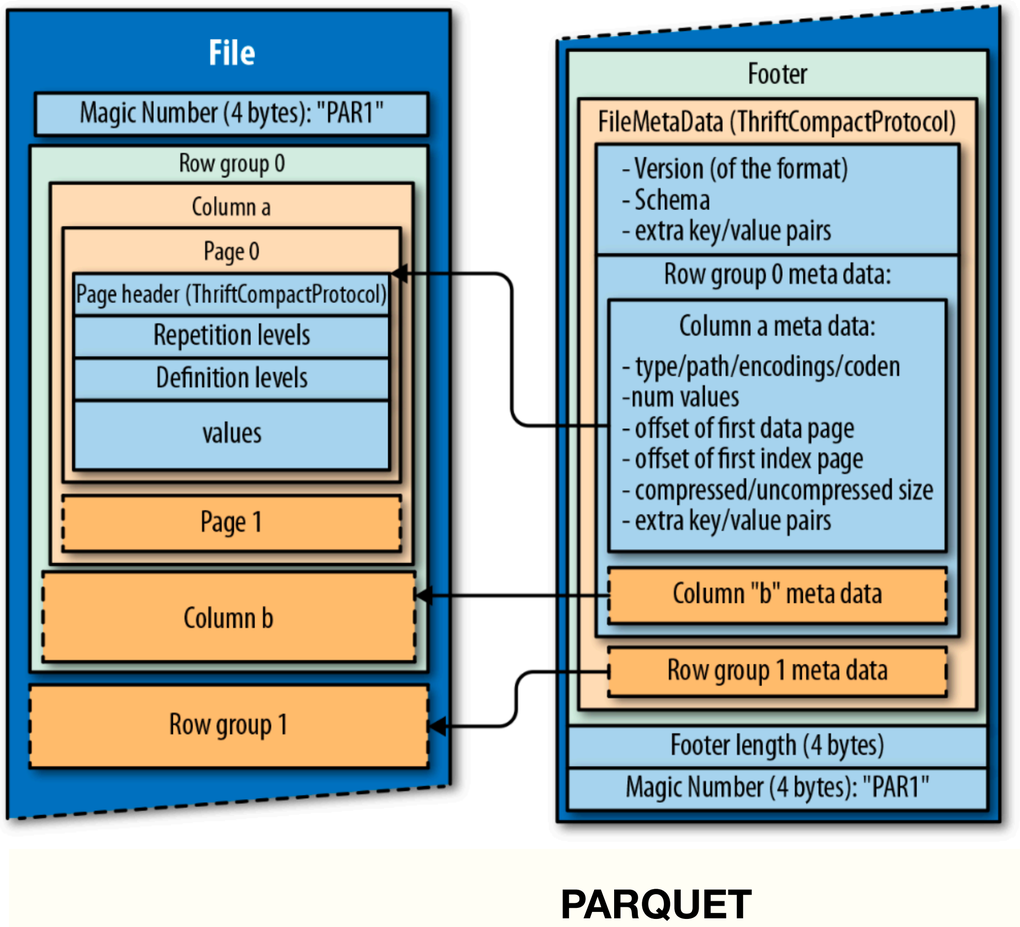
FORMAT	COLUMNAR	COMPRESSION	SUPPORT
AVRO	✗	GOOD	HADOOP SPARK ATHENA PRESTO
PARQUET	✓	GREAT	HADOOP SPARK ATHENA PRESTO
ORC	✓	EXCELLENT	HADOOP SPARK ATHENA PRESTO
CARBONDATA	✓	GOOD	HADOOP SPARK

	ORC	Parquet	Avro
Row or column	Column	Column	Row
Compression	Great	Great	Good
Speedup (compared to text file)	10–100x	10–100x	10x
Schema evolution	Good	Better	Best
Platforms	Hive, Spark, Presto	Hive, Spark, Presto	Hive, Spark
Splittability	Best	Best	Better
File statistics	Yes	Yes	No
Indexes	Yes	Yes	No
Bloom filters	Yes	No	No

	Parquet	AVRO
Format Type	Column-based	Row-based
Built For/with	Optimized column-wise compression and querying in a splittable file format designed for efficient Map-Reduce processing	Compact binary storage and exchange of records, with schema evolution and support for many different programming languages
Schema Storage	Column metadata stored at the end of the file (allows for fast, one-pass writing)	Stored in human-readable JSON format at the beginning of each message or file
Use Case	Quickly query all the values from a particular column across a very large dataset. For example, compute the average price of purchases over millions of purchase records	Share entire records between applications. For example, event data of in-app purchases for use by many downstream applications such as logging, auditing, and business analytics







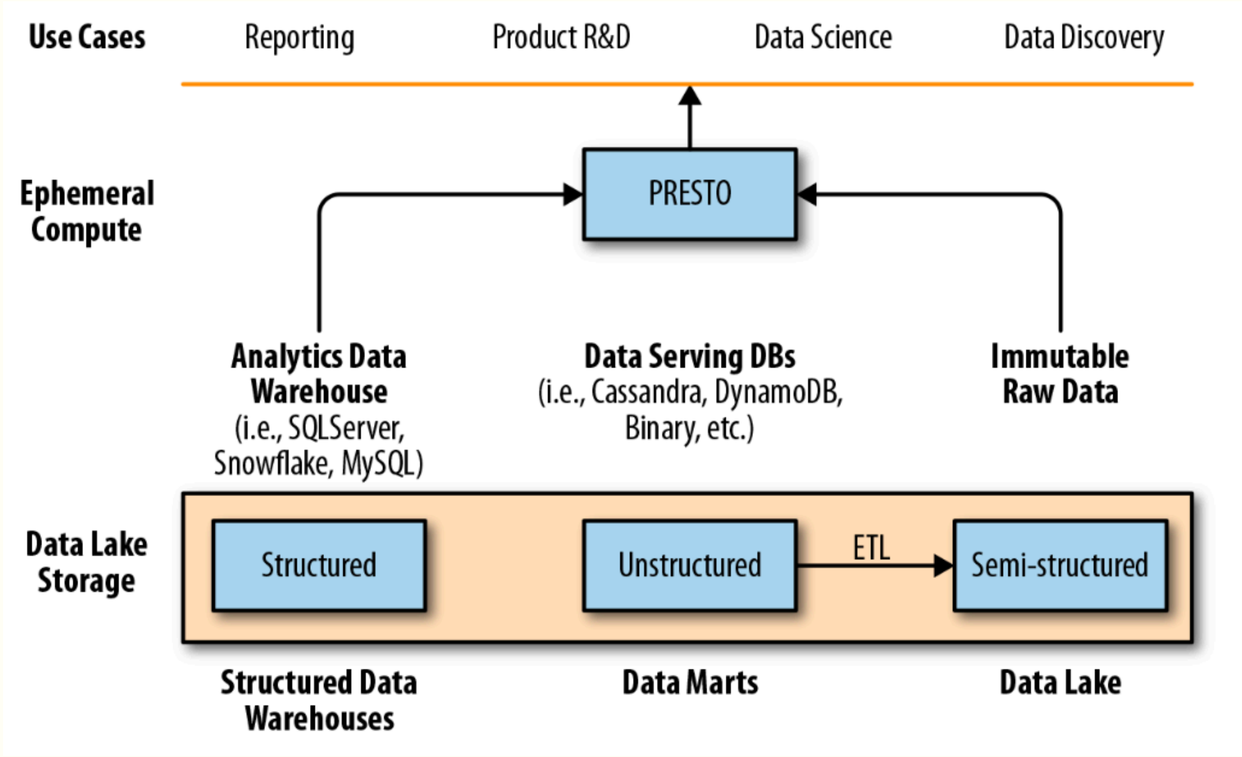
PARQUET

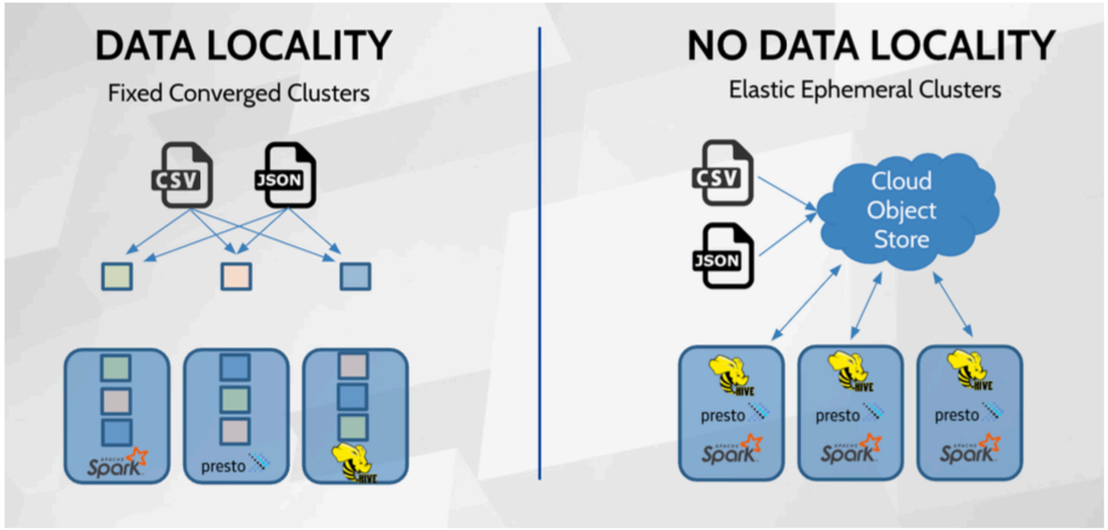
Univ.AI

SQL on Hadoop Alternatives

- Cloudera Impala ran a dedicated daemon on each datanode in the Hadoop cluster. Uses Hive metastore and supports HiveQL constructs
- Presto from facebook in 2012, conbstructed to speed up slow queries on a 300PB data lake and its successor Trino have a similar architecture
- SparkSQL
- Dremio, built on top of Apache Arrow (more on that later)

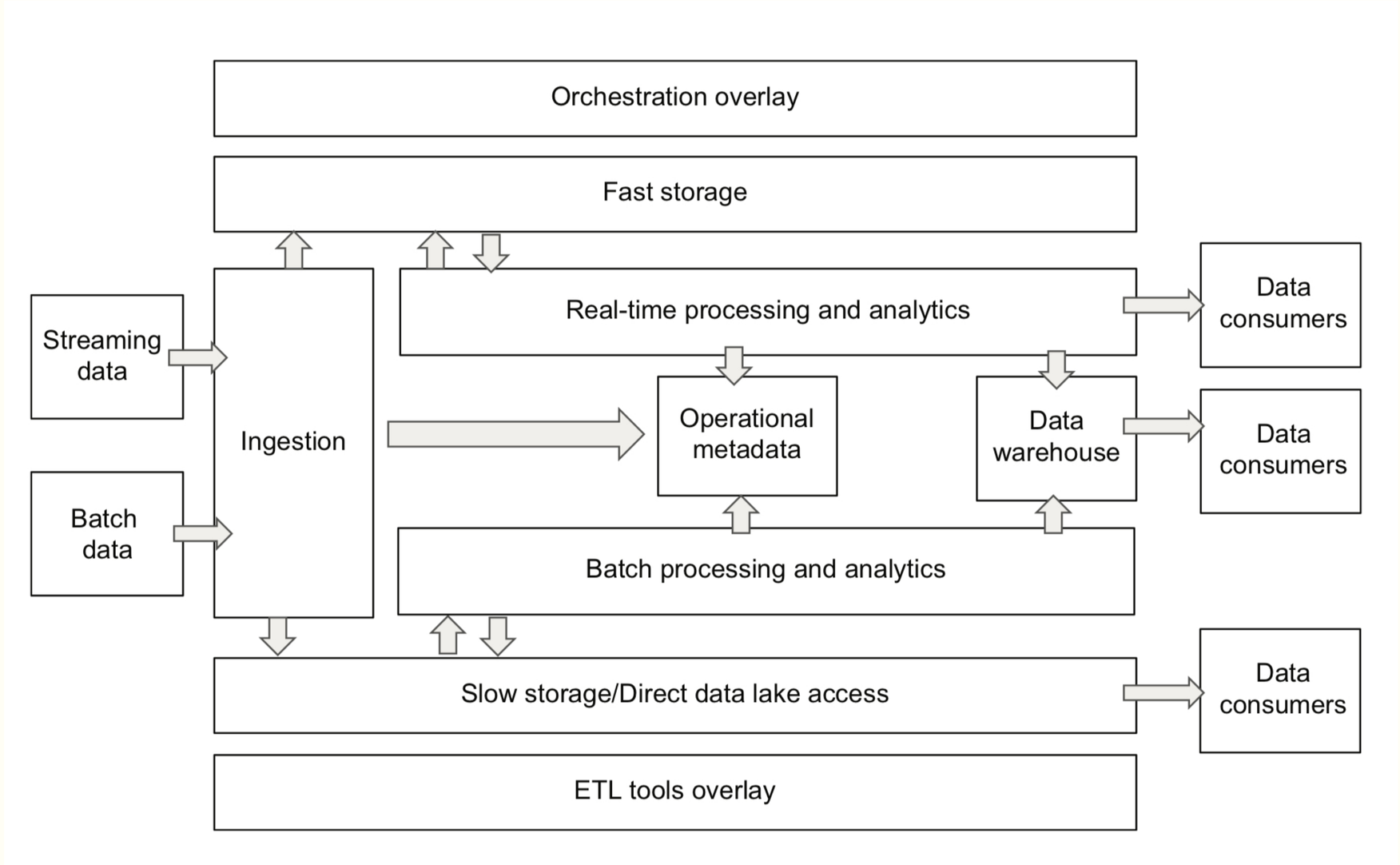
The Holy Grail





Univ.AI

Infra for the holy grail



Cloud Infra for the holy grail