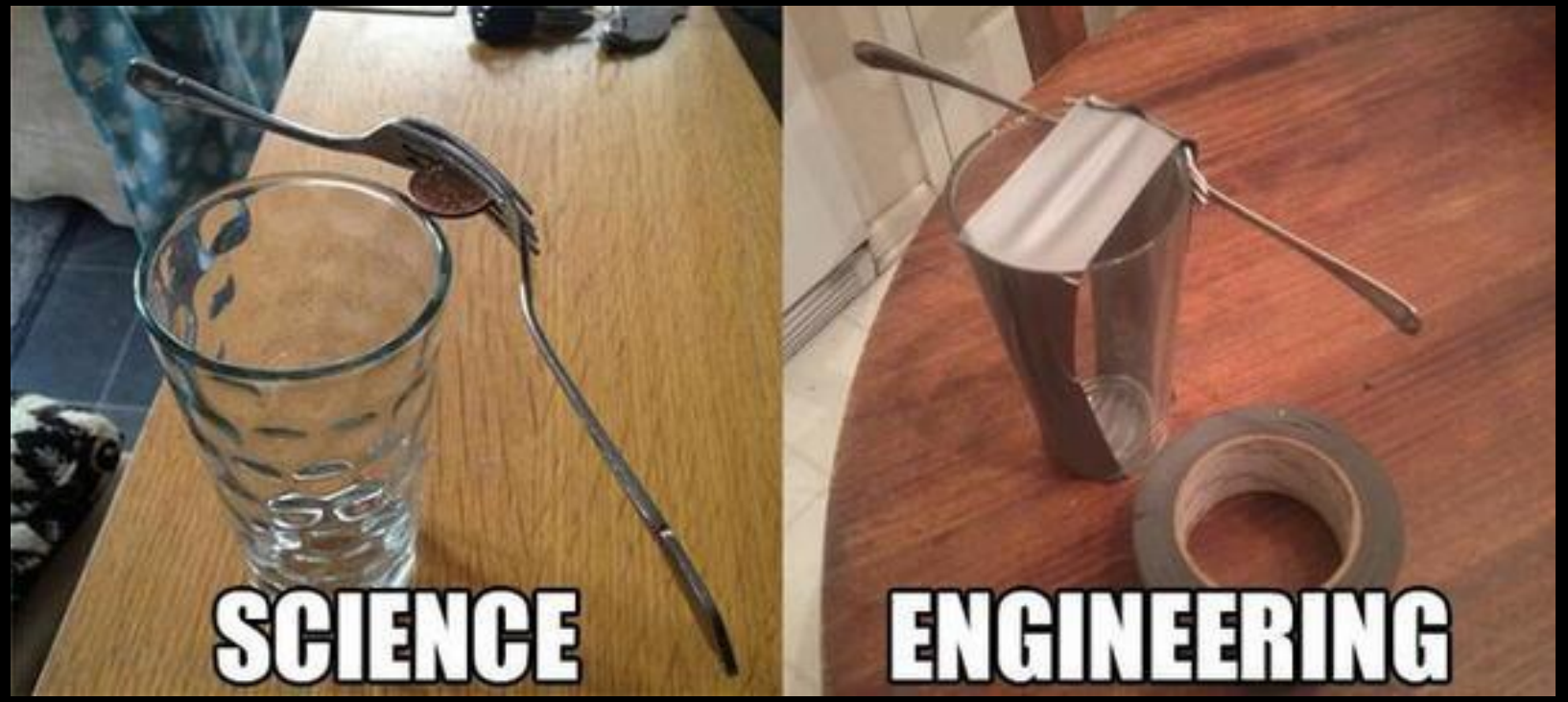
Memento

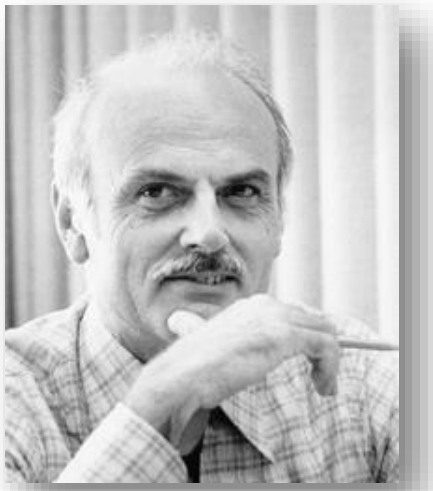
How data storage hardware constraints impact database software architectures

Michael Hausenblas, Chief Data Engineer MapR

NoSQL Matters, Dublin, 2014-09-03


SCIENCE
ENGINEERING

Warm-up

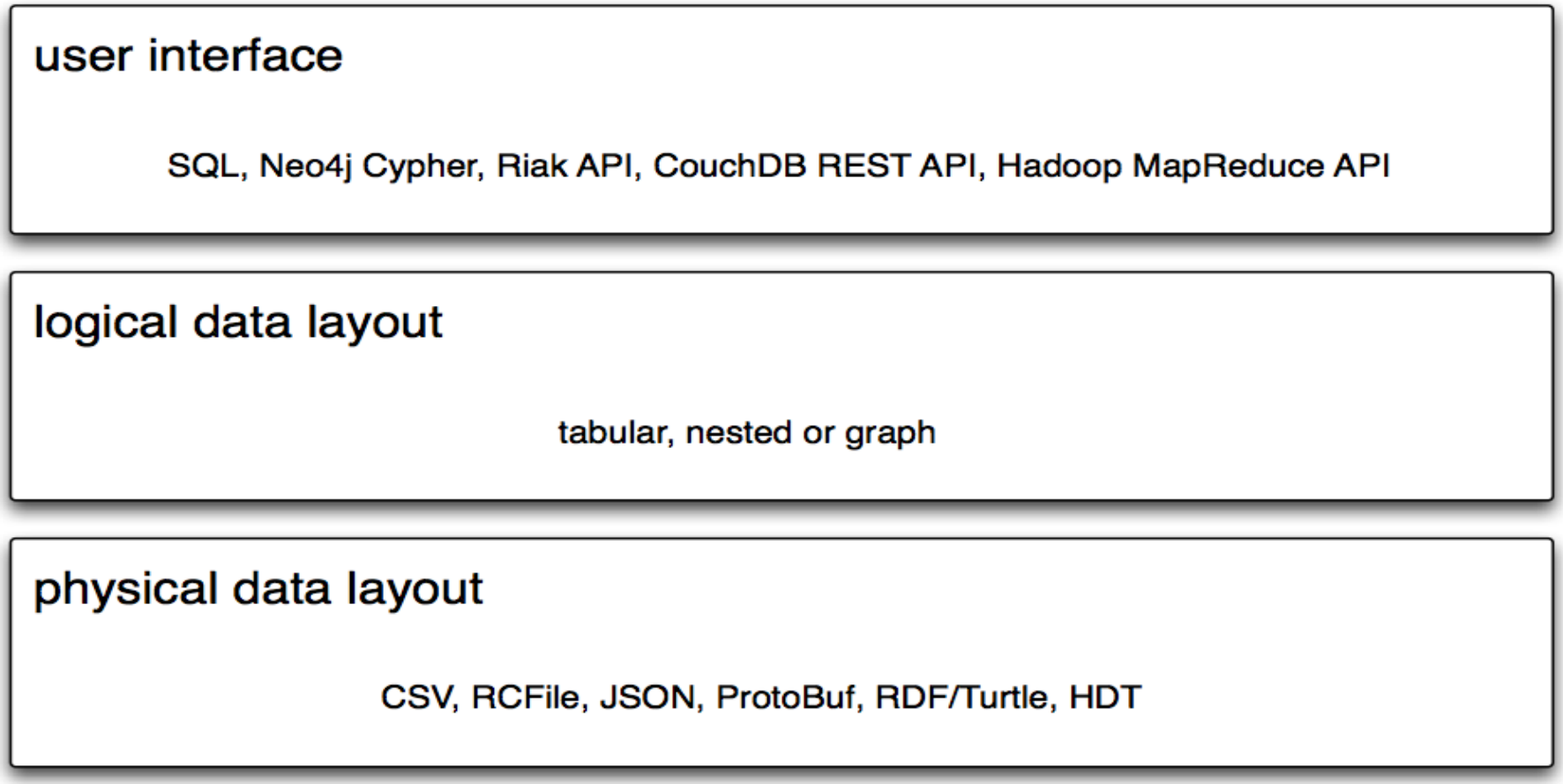


Edgar Frank Codd

Jeffrey Adgate Dean

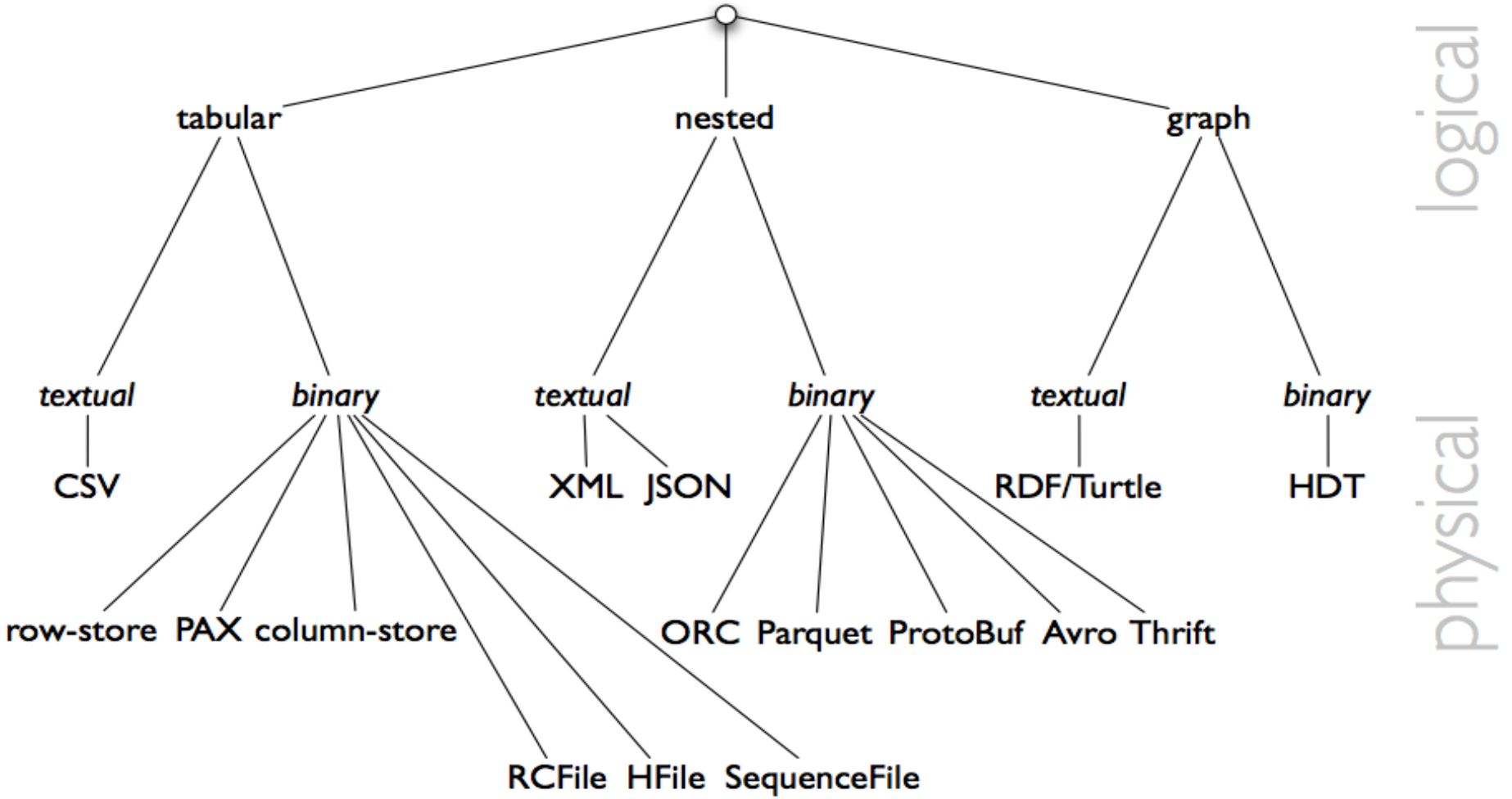
Matei Zaharia

Data layouts



From ‘Notes on Physical & Logical Data Layouts’ via http://arxiv.org/abs/1305.6506

Data layouts



From ‘Notes on Physical & Logical Data Layouts’ via http://arxiv.org/abs/1305.6506

Row-oriented vs. columnar-oriented storage

- Row-oriented
  - entities together
  - traditionally in RDBMS
  - operational workloads preferred

- Columnar
  - attributes together
  - more and more popular in NoSQL solutions (Parquet, ORC, etc.)
  - analytical workloads preferred

Assumptions in RDBMS design

- Hard disk space is limited → up-front decide what to ‘allow in’
- I/O is the major bottleneck → heavily use index
- Separate physical & logical layout → declarative interface (SQL)
- Sharding?

Assumptions in NoSQL datastore design

- Commodity hardware → scale out (but also: CAP)
- Seek times expensive → avoid joins, de-normalize data
- Physical layout may be exposed → imperative interface
- ‘natural-born sharders’

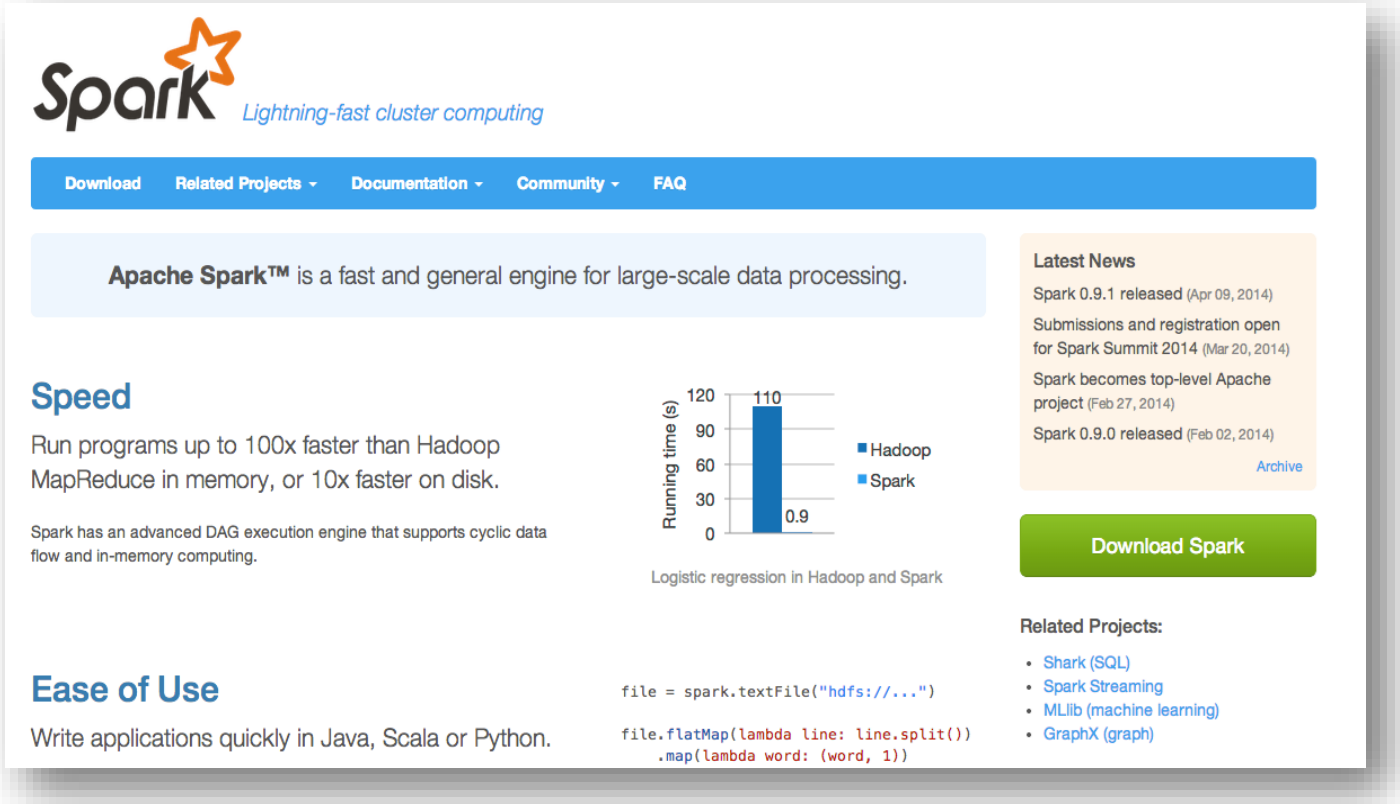
Assumptions in distributed query engine design

- data volume >> code volume → data locality (in-situ processing)
- network ‘good enough’ → RPCs, streaming operators
- de-couple physical data layout from interface (SQL over JSON)

Case study: Apache Spark

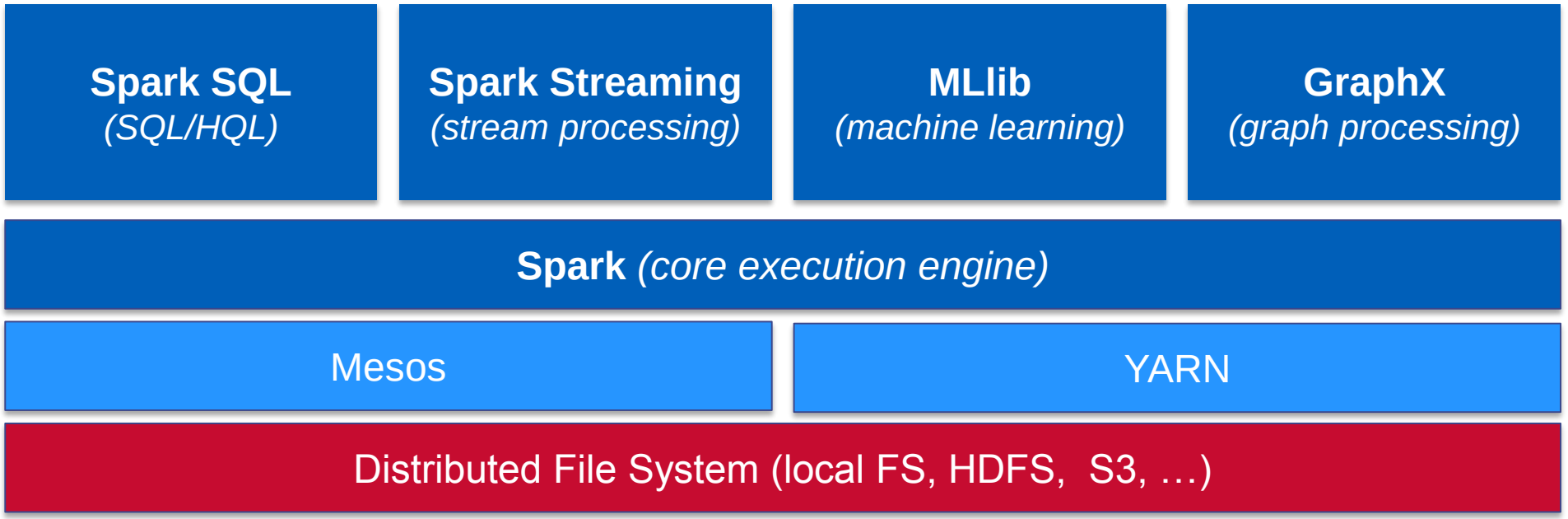
Apache Spark

- Originally developed in 2009 in UC Berkeley's AMP Lab
- A top-level Apache project as of 2014
- Databricks are commercial shephards
- Enterprise support from Hadoop distributions



https://spark.apache.org/

Spark—a unified platform …



Continued innovation bringing new functionality, such as:

- **Tachyon** (Shared RDDs, off-heap solution)
- **BlinkDB** (approximate queries)
- **SparkR** (R wrapper for Spark)

… for complex workloads …

- Iterative Algorithms
  - machine learning
  - graph processing beyond DAG
- Interactive Data Mining
- Streaming Applications

… expressive API …

map

reduce

… expressive API …

```
map
filter
groupBy
sort
union
join
leftOuterJoin
rightOuterJoin
```

```
reduce
count
fold
reduceByKey
groupByKey
cogroup
cross
zip
```

```
sample
take
first
partitionBy
mapWith
pipe
save    ...
```

Resilient Distributed Datasets (RDD)

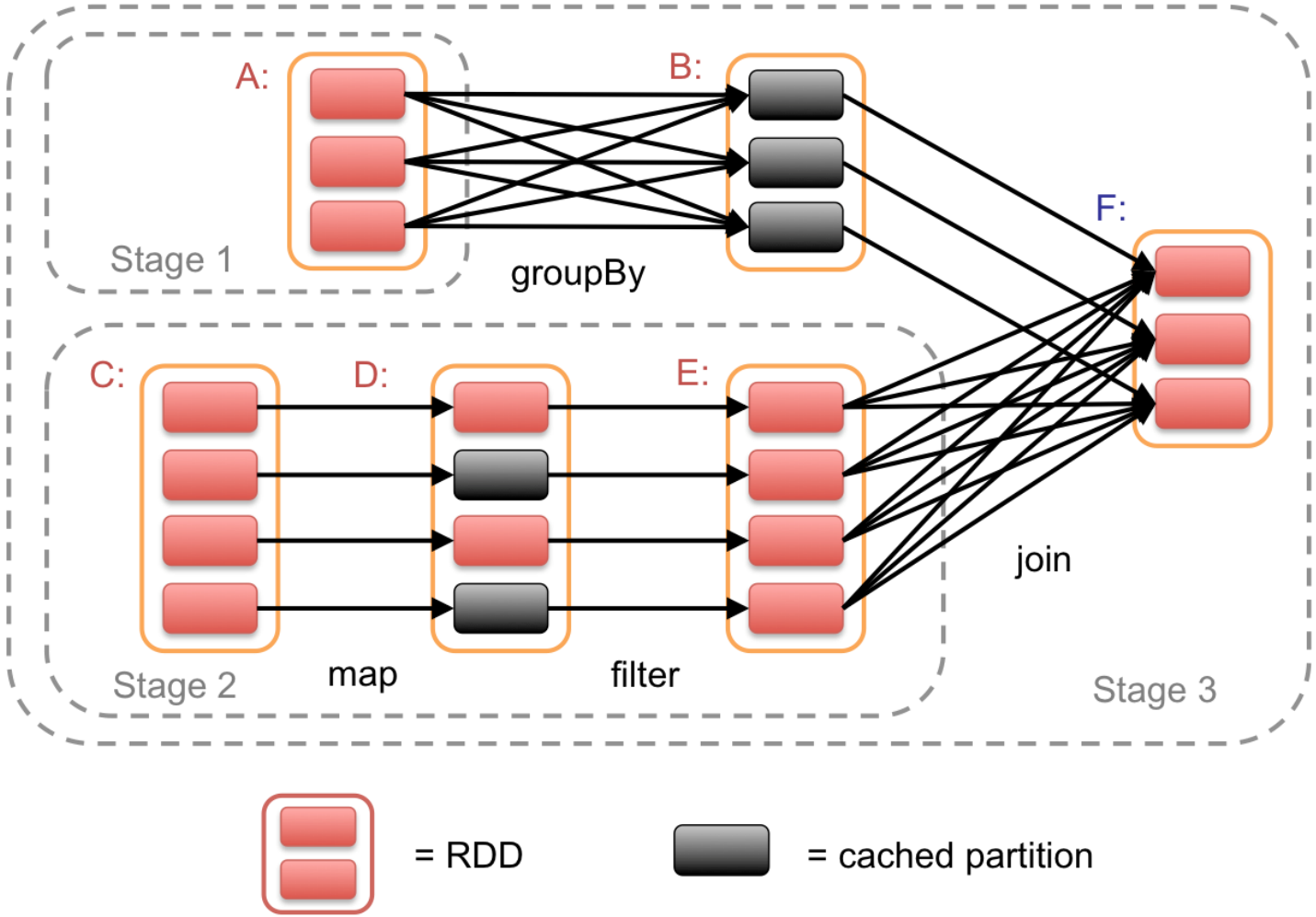
- RDDs are the core of the Spark execution engine
- Collections of elements that can be operated on in parallel
- Persistent in memory between operations
- Fault tolerance through linage (DAG of ops)

http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

RDD operations

- Lazy evaluation is key to Spark
- Transformations
  - Creation of a new dataset from an existing:
    map, filter, distinct, union, sample, groupByKey, join, etc.
- Actions
  - Return a value after running a computation:
    collect, count, first, takeSample, foreach, etc.

RDD fault tolerance



Conclusion

What matters?

- Scalability +
- Performance +
- Business continuity +
- Cost-effectiveness

Q&A

Engage with us!

@mhausenblas   maprtech

mapr-technologies   MapR

mhausenblas@mapr.com   maprtech

MAPR