

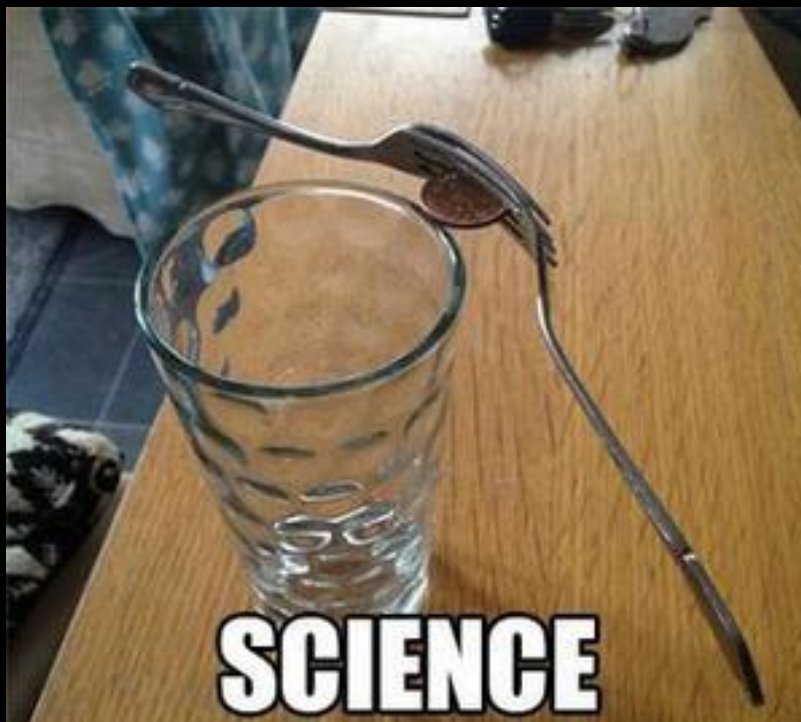


Memento

How data storage hardware constraints impact database software architectures

Michael Hausenblas, Chief Data Engineer MapR

NoSQL Matters, Dublin, 2014-09-03

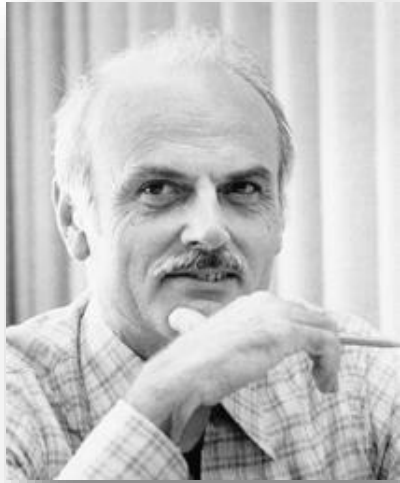


SCIENCE



ENGINEERING

Warm-up



Edgar Frank Codd



Jeffrey Adgate Dean



Matei Zaharia



Data layouts

user interface

SQL, Neo4j Cypher, Riak API, CouchDB REST API, Hadoop MapReduce API

logical data layout

tabular, nested or graph

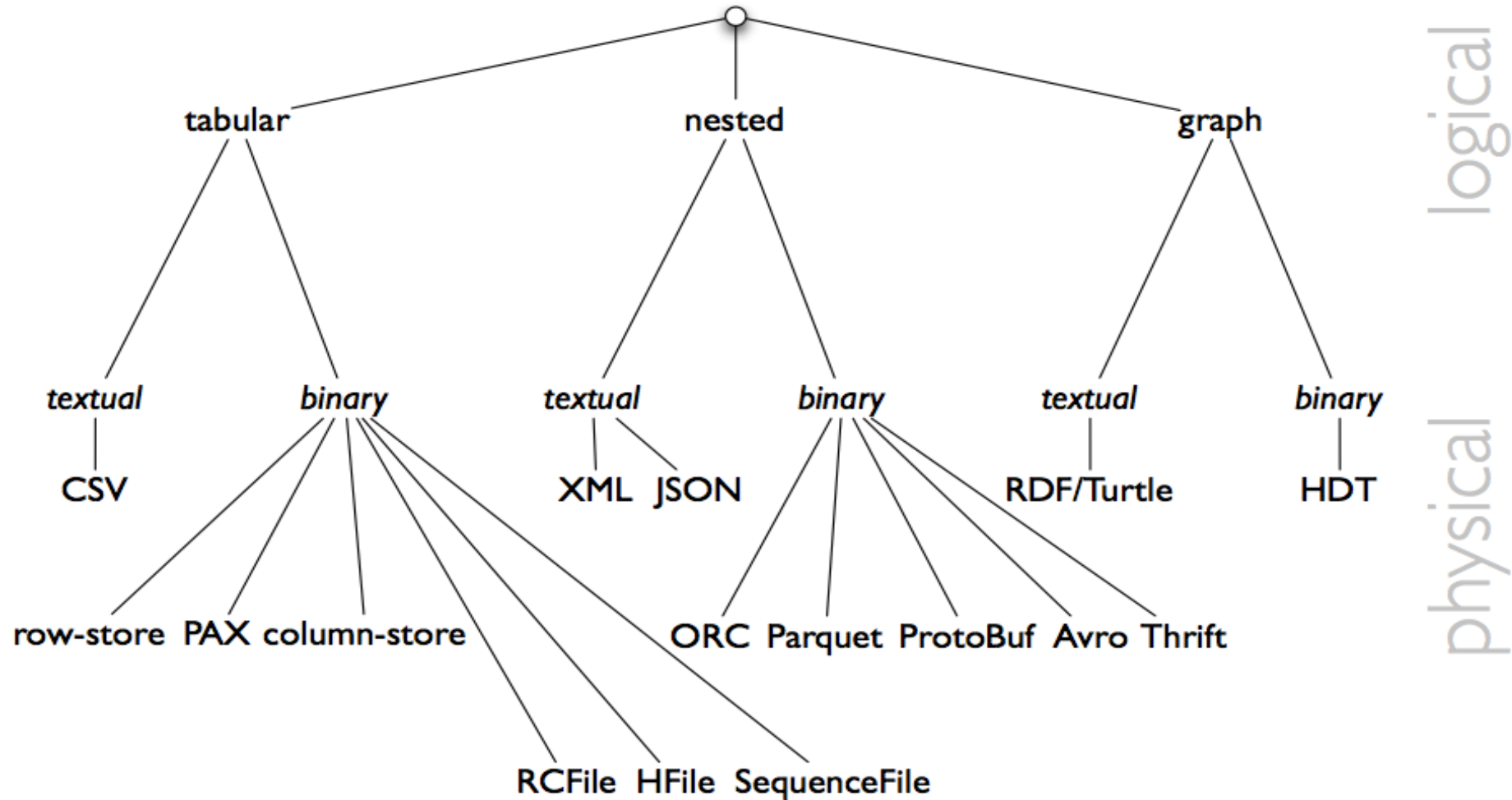
physical data layout

CSV, RCFile, JSON, ProtoBuf, RDF/Turtle, HDT

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

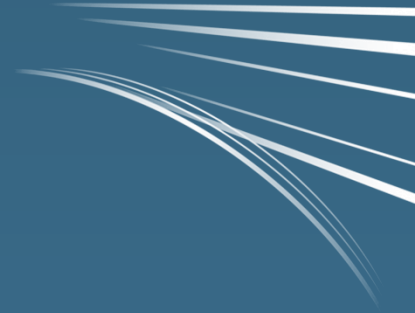
- Commodity hardware → scale out (but also: CAP)
- Seek times expensive → avoid joins, de-normalize data
- Physical layout may be exposed → imperative interface
- ‘natural-born sharders’



Assumptions in distributed query engine design

- data volume \gg code volume \rightarrow data locality (in-situ processing)
- network 'good enough' \rightarrow 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 shepherds
- Enterprise support from Hadoop distributions

The screenshot shows the Apache Spark website homepage. At the top left is the Spark logo with the tagline "Lightning-fast cluster computing". A blue navigation bar contains links for "Download", "Related Projects", "Documentation", "Community", and "FAQ". Below the navigation bar is a light blue banner stating "Apache Spark™ is a fast and general engine for large-scale data processing." To the right of this banner is a "Latest News" section with several announcements and an "Archive" link. Below the banner is a "Speed" section with a bar chart comparing Hadoop and Spark running times for logistic regression. The chart shows Hadoop at 110 seconds and Spark at 0.9 seconds. Below the chart is a "Download Spark" button. To the left of the chart is text describing Spark's advanced DAG execution engine. Below the chart is an "Ease of Use" section with code snippets for reading a file and performing a flatMap operation.

Spark Lightning-fast cluster computing

Download Related Projects Documentation Community FAQ

Apache Spark™ is a fast and general engine for large-scale data processing.

Speed
Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

Ease of Use
Write applications quickly in Java, Scala or Python.

Running time (s)

Framework	Running time (s)
Hadoop	110
Spark	0.9

Logistic regression in Hadoop and Spark

file = spark.textFile("hdfs://...")
file.flatMap(lambda line: line.split())
.map(lambda word: (word, 1))

Latest News
Spark 0.9.1 released (Apr 09, 2014)
Submissions and registration open for Spark Summit 2014 (Mar 20, 2014)
Spark becomes top-level Apache project (Feb 27, 2014)
Spark 0.9.0 released (Feb 02, 2014) [Archive](#)

Download Spark

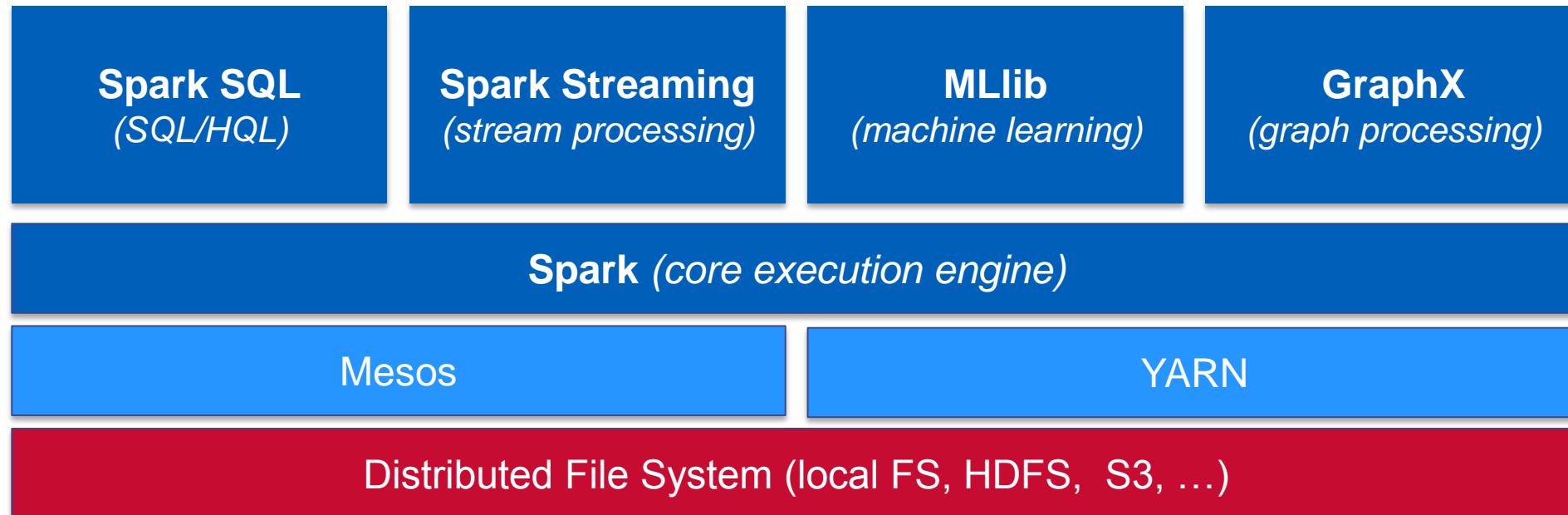
Related Projects:

- Shark (SQL)
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)

<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 lineage (DAG of ops)

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

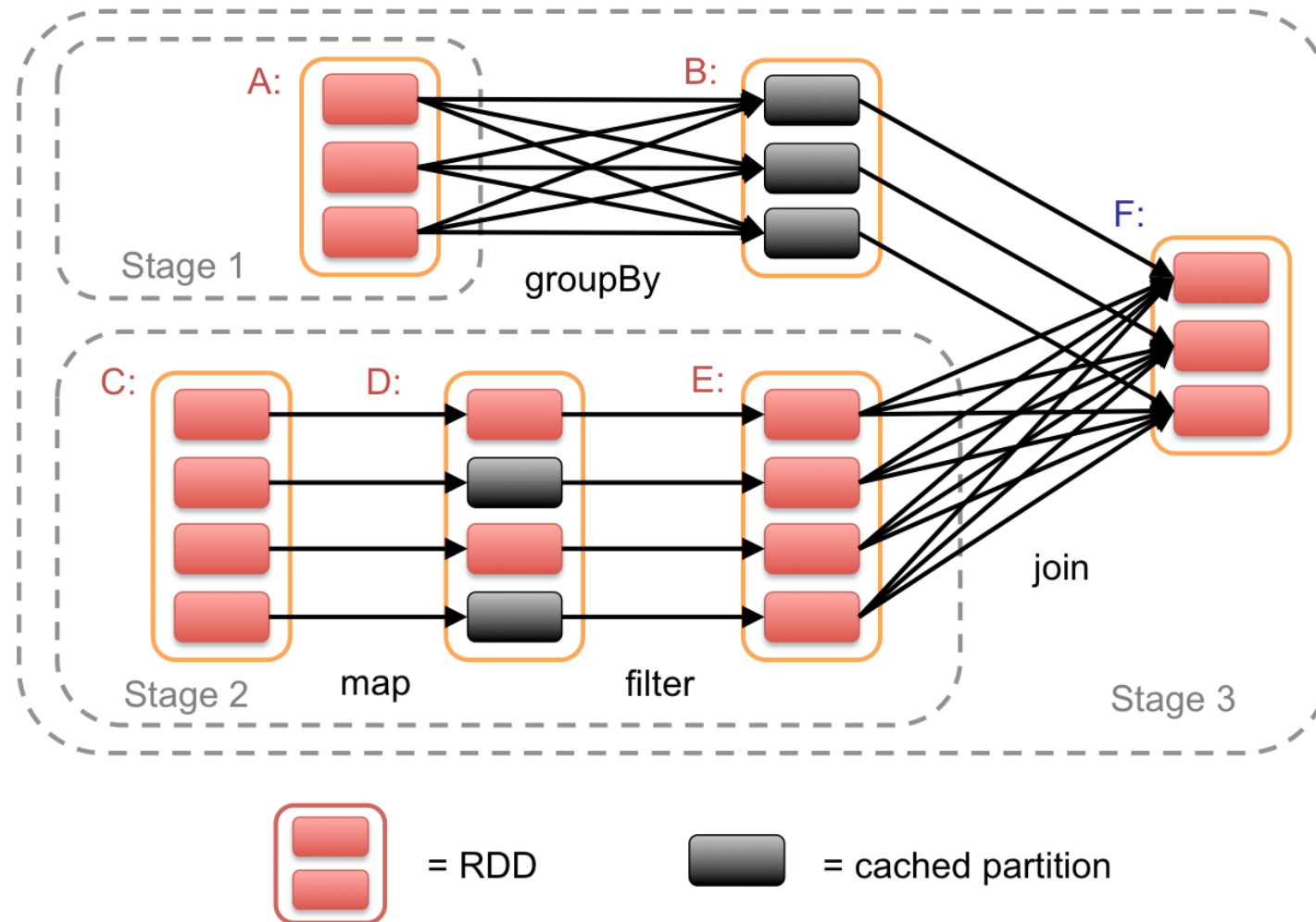


RDD operations

- 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

