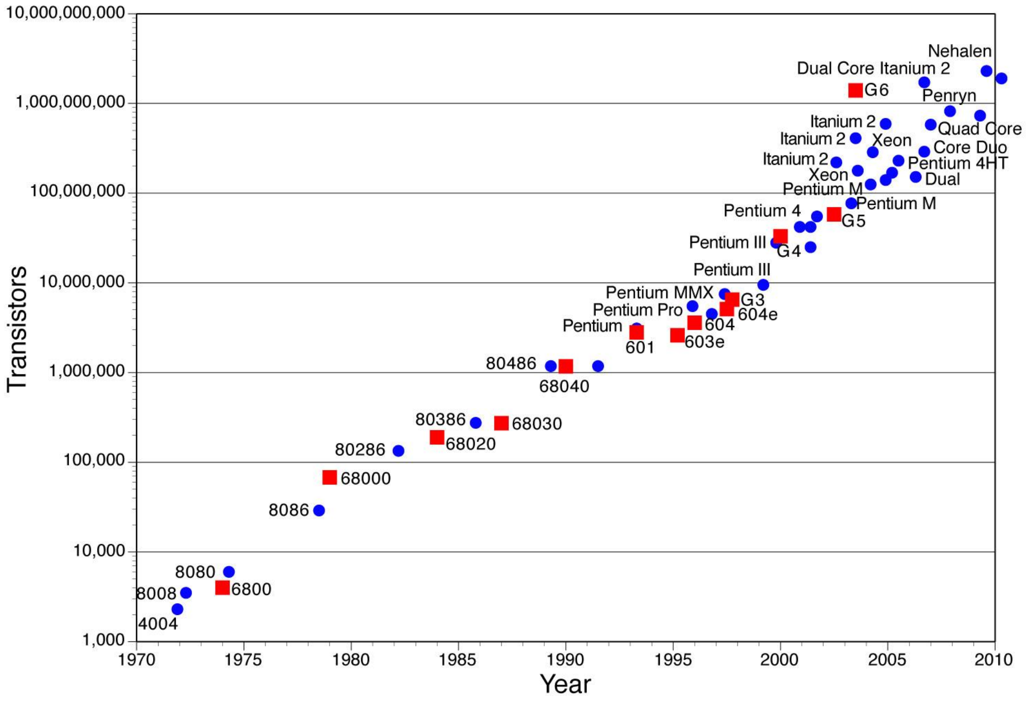
SQL-on-Hadoop

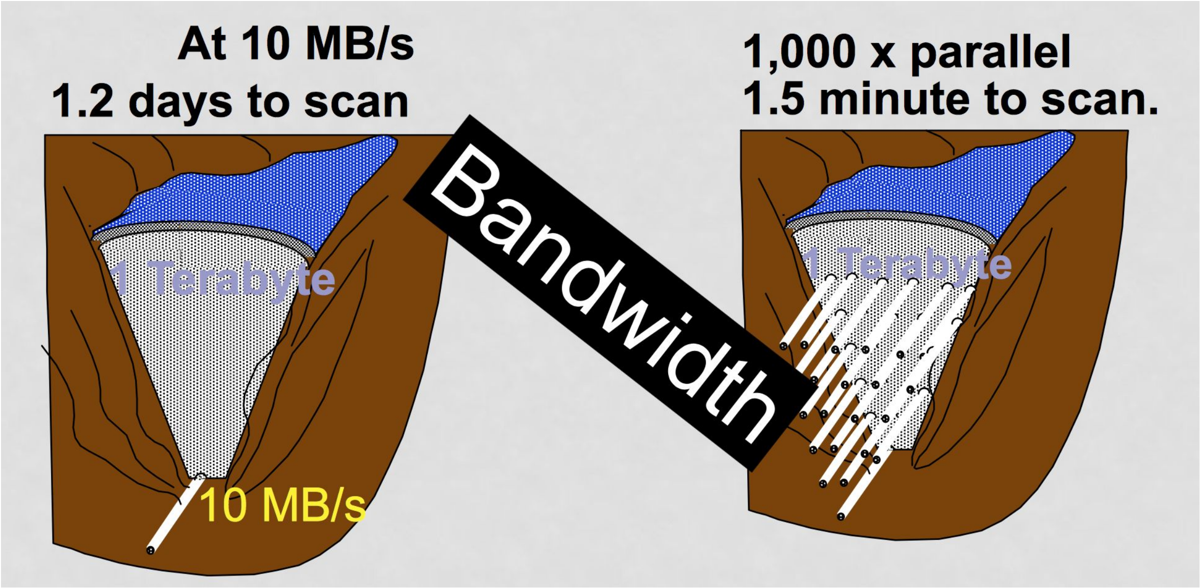
Aron Szanto and Jack Dent

Why do we need to parallelize data analysis?



Source(s): http://www.is.umk.pl/~duch/Wyklady/komput/w03/Moores_Law.jpg

Why do we need to parallelize data analysis?



Source(s): http://web.cs.wpi.edu/~cs561/s12/Lectures/4-5/ParallelDBs.pdf

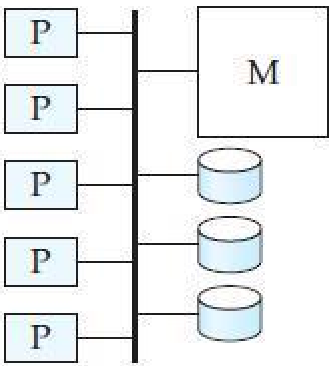
Why do we need to parallelize data analysis?

d = data size (GB)
b = bandwidth of single machine (GB/s)

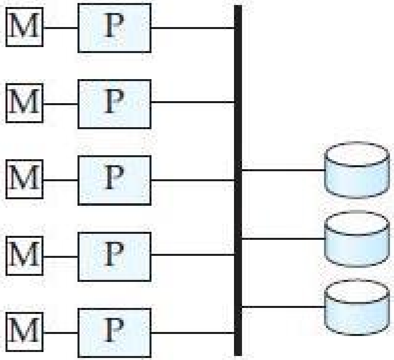
Time on single machine architecture = *d/b*

Time on *n*-machine architecture = *d/nb*
(assumes perfect horizontal scalability)

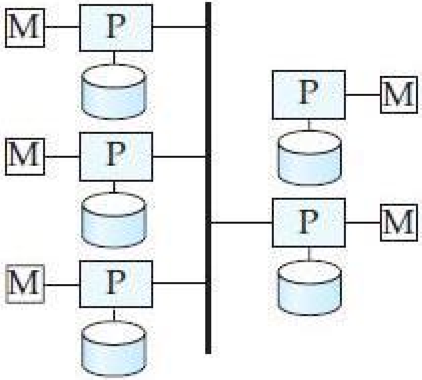
Parallel database architectures



(a) shared memory



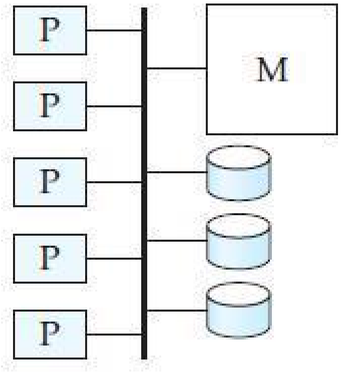
(b) shared disk



(c) shared nothing

Source(s): http://backstopmedia.booktype.pro/big-data-dictionary/parallel-databases/

Shared-memory architectures

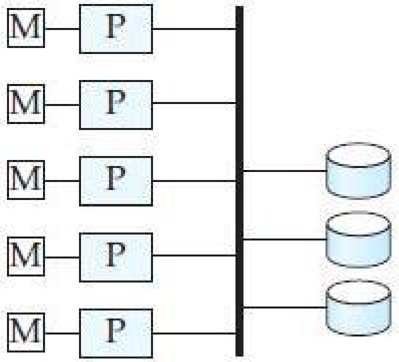


(a) shared memory

Definition: there is a single memory address-space for all processors, but each processor can have its own disk, local memory, and cache

Source(s): adapted from http://web.cs.wpi.edu/~cs561/s12/Lectures/4-5/ParallelDBs.pdf

Shared-disk architectures

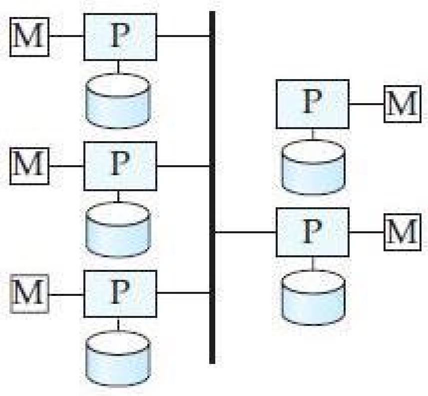


(b) shared disk

Definition: “every processor has its own memory (not accessible by others), and all machines can access all disks in the system”

Source(s): http://web.cs.wpi.edu/~cs561/s12/Lectures/4-5/ParallelDBs.pdf

Shared-nothing architectures

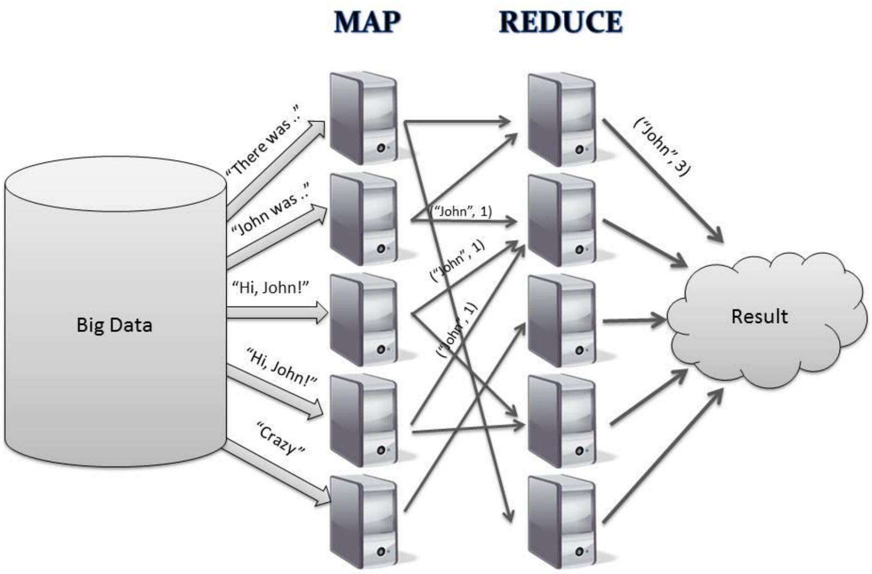


(c) shared nothing

Definition: “a collection of independent, possibly virtual, machines, each with local disk and local main memory, connected together on a high-speed network”

Source(s): “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads”

MapReduce: shared-nothing data analysis



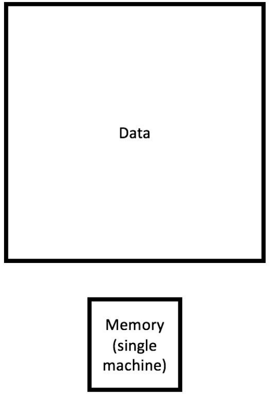
Key paper: "MapReduce: Simplified Data Processing on Large Clusters", Dean and Ghemawat, Google, 2004

Open source implementation in Apache Hadoop suite

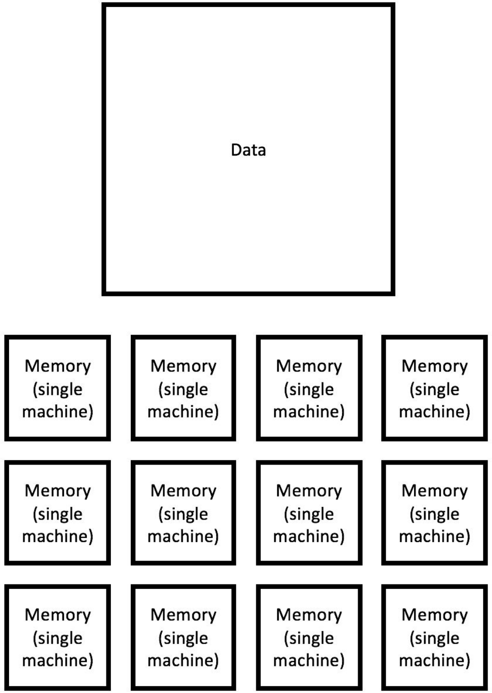
Source(s): https://scr.sad.supinfo.com/articles/resources/207908/2807/1.png

Scaling main memory

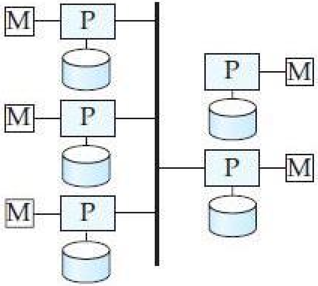
Single machine



Parallel machines



Challenge: SQL queries on shared-nothing architectures?



(c) shared nothing

+

SQL

Scale out to 1000s of machines
Fault tolerant
Support heterogeneous environments

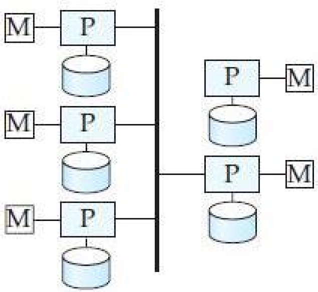
… but difficult to program, and not performant for structured data

Scale up (fast queries over structured data)
Flexible query language

… but do not scale out well

Source(s): http://tinyurl.com/jd3a8ao

Challenge: SQL queries on shared-nothing architectures?



(c) shared nothing

+

Can we combine the positive features (*performance*, *flexible query interface*) of shared-architecture parallel databases with the positive features (*fault tolerance*, *horizontal scalability*) of shared-nothing architectures?

Source(s): http://tinyurl.com/jd3a8ao

HadoopDB (background)

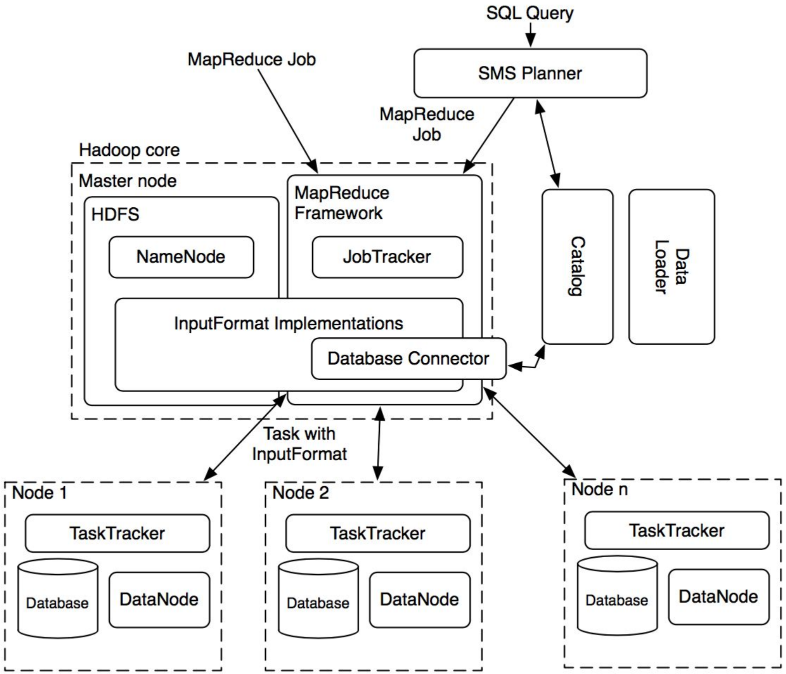
+

HDFS + MapReduce
inter-node

SQL query execution
intra-node

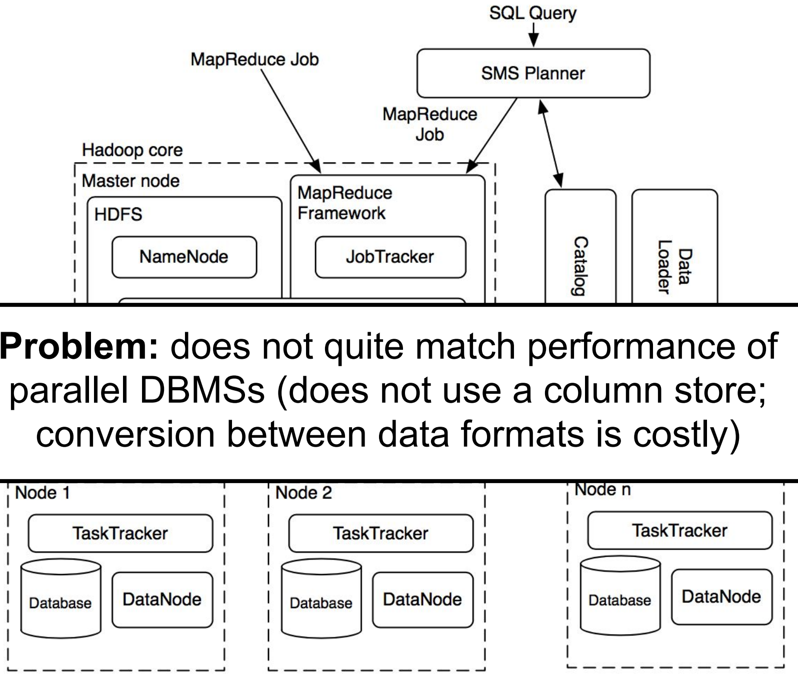
Source(s): http://sites.gsu.edu/skondeti1/files/2015/10/Untitled-1-122jwp8.png; https://www.carnaghan.com/wp-content/uploads/2016/08/postgresql-logo.png

HadoopDB (background)



Source(s): “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads”

HadoopDB (background)



Problem: does not quite match performance of parallel DBMSs (does not use a column store; conversion between data formats is costly)

Source(s): “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads”

SQL with shared-nothing architectures

	File system	File format	Query language	Distributed runtime
Apache Hive	Apache HDFS	Optimized Row Columnar (ORC)	HiveQL	MapReduce or Tez
Cloudera Impala	Apache HDFS	Parquet	Impala SQL	`impalad`

Source(s): “SQL-on-Hadoop: Full Circle Back to Shared-Nothing Database Architectures”

Hive file format: ORC

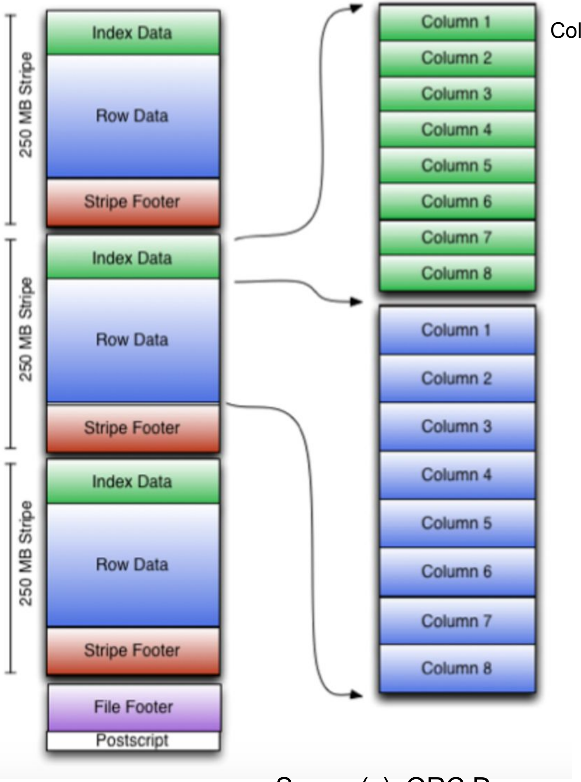
(Bloom Filter) Column 1 (min, max, sum)

Column 2 (min, max, sum) ……

Ids 40k-50k

Ids 50k-60k

Ids 60k-70k



Source(s): ORC Documentation Pages, https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

Hive file format: ORC

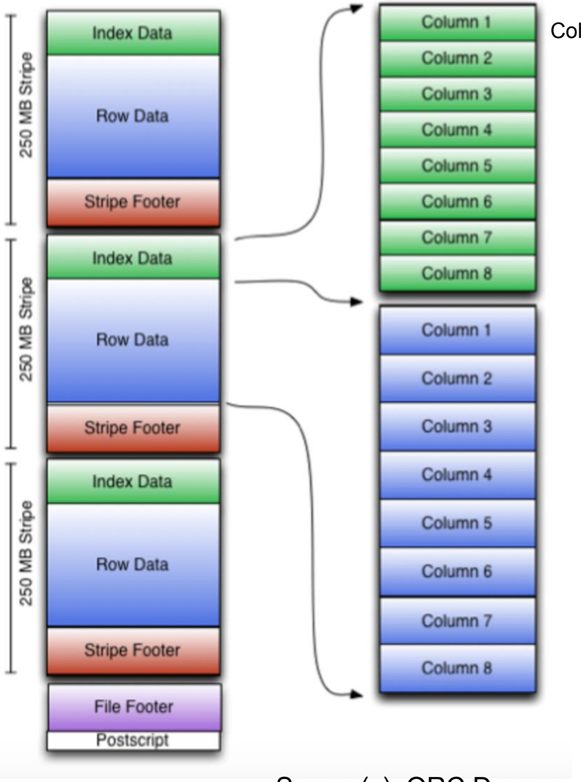
(Bloom Filter) Column 1 (min, max, sum)

Column 2 (min, max, sum) ......

Ids 40k-50k

Ids 50k-60k

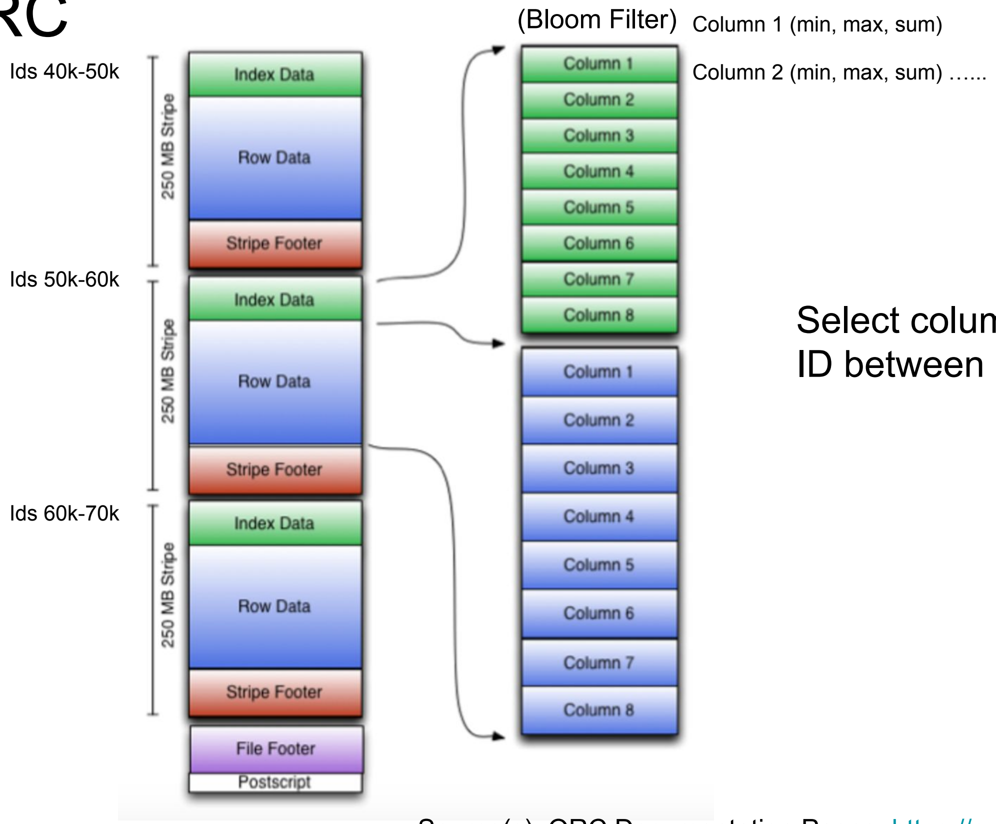
Ids 60k-70k



Select sum(column_2)/sum(column_1) from table where ID between 50k and 60k

Source(s): ORC Documentation Pages, https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

Hive file format: ORC



Select column_2, column_4 from table where ID between 52k and 57k

Source(s): ORC Documentation Pages, https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

Hive file format: ORC

Ids 40k-50k

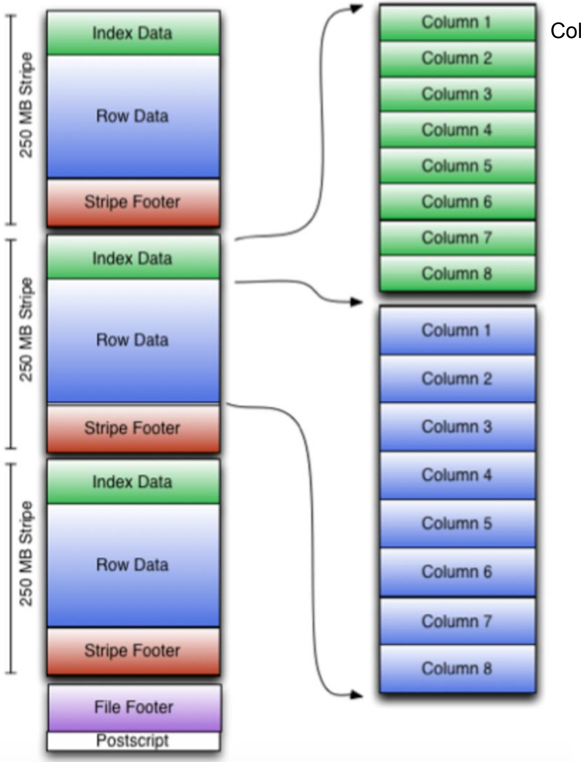
Ids 50k-60k

Ids 60k-70k

(Bloom Filter)

Column 1 (min, max, sum)

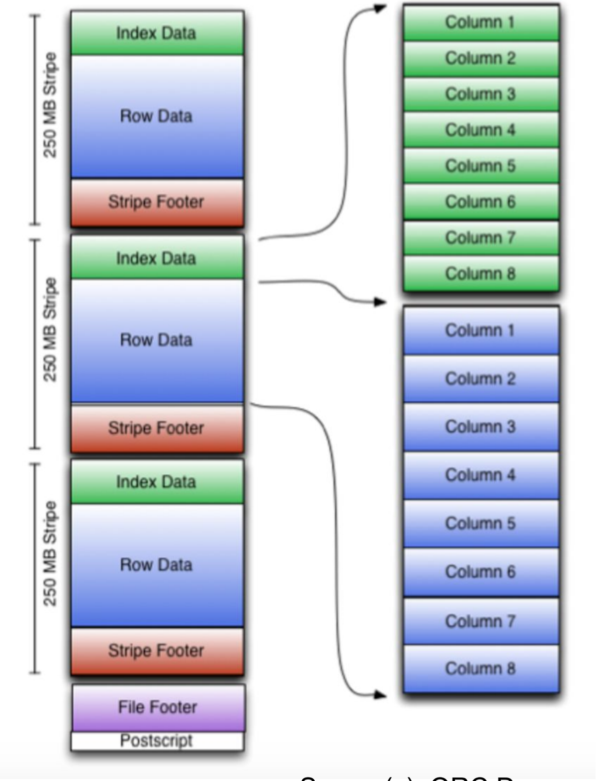
Column 2 (min, max, sum) ……



Select column_2, column_4 from table where ID = 52566 (which doesn't exist!)

Source(s): ORC Documentation Pages, https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

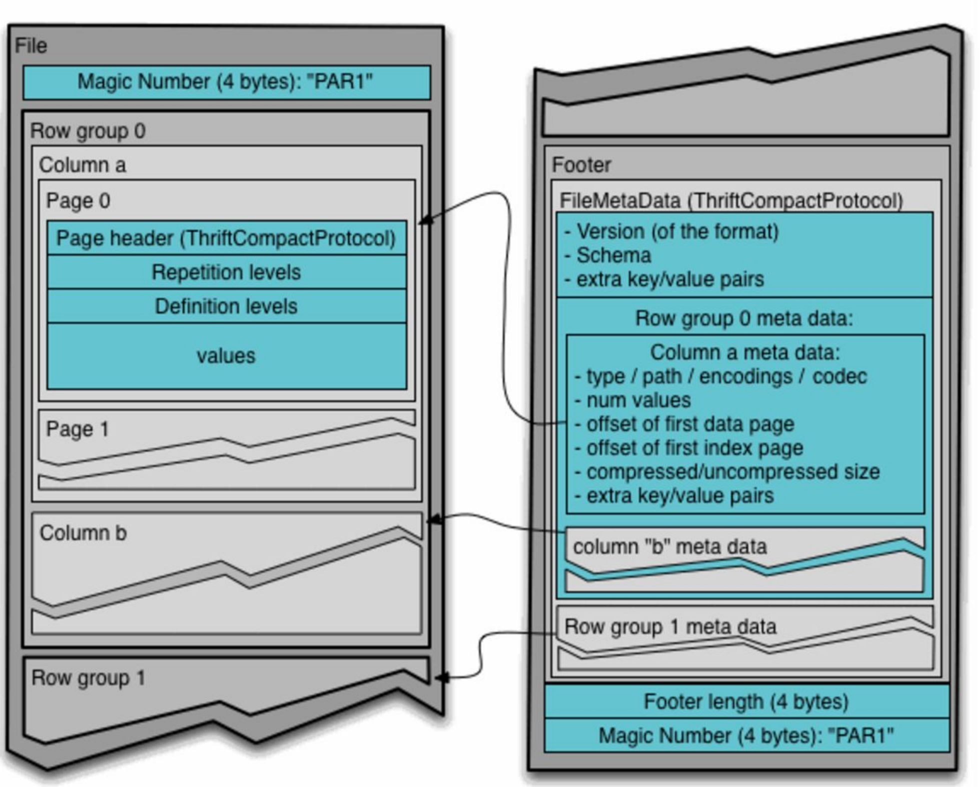
Hive file format: ORC



Is this a “good” architecture

Source(s): ORC Documentation Pages, https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

Impala file format: Parquet

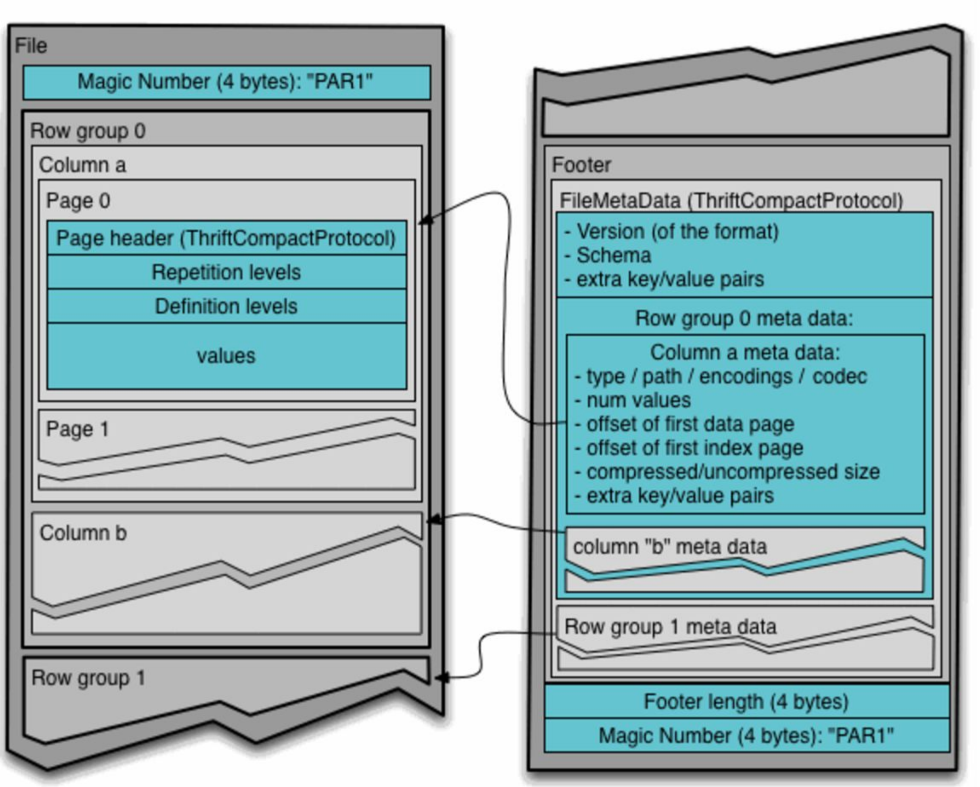


What's the big difference

Why does it matter

Source(s): Parque Documentation Pages, https://www.parquet.apache.org/documentation/latest/

Impala file format: Parquet

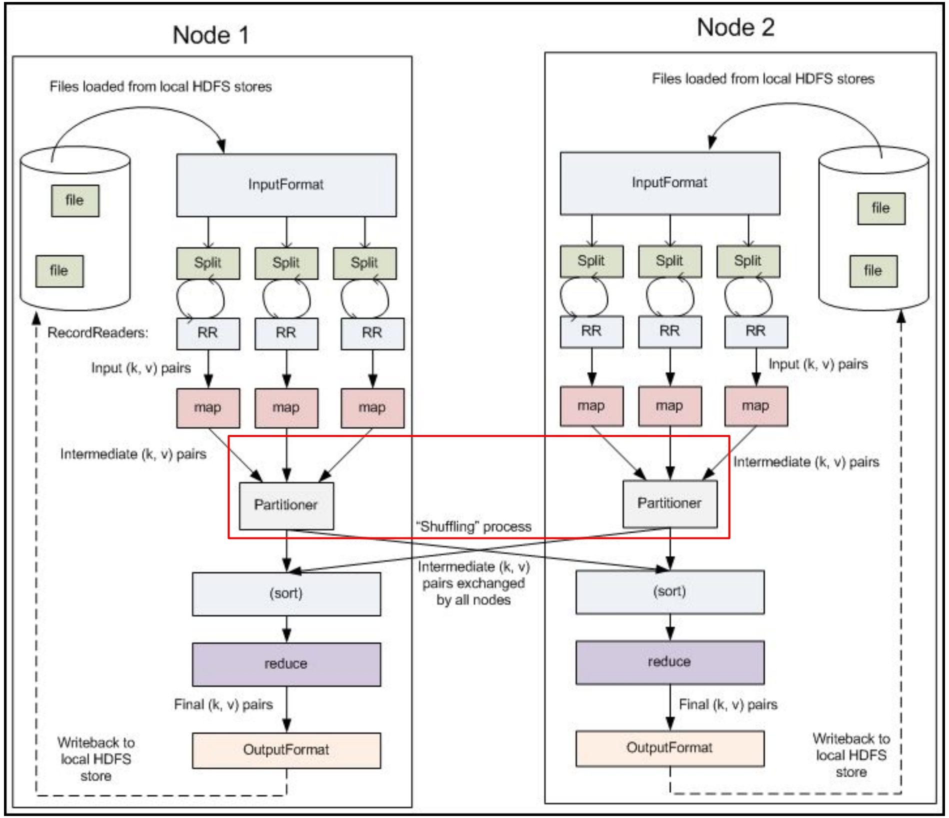


What's the big difference

Why does it matter

Source(s): Parque Documentation Pages, https://www.parquet.apache.org/documentation/latest/

Hive runtime: MapReduce

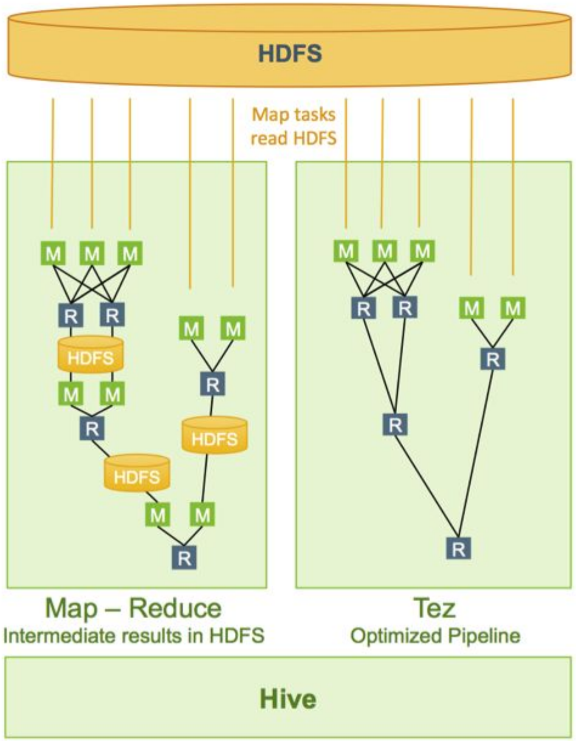


Hive-MapReduce materializes intermediate results and writes to disk

Why is this bad? Why is this good?

Source(s): https://www.hadooptpoint.com

Hive runtime: From MR to Tez

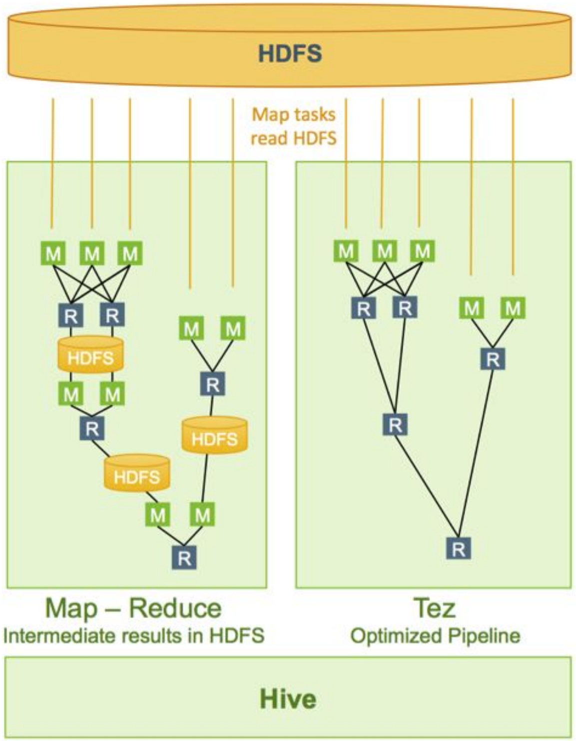


What's the big difference?

Why does it matter?

Source(s): HortonWorks, https://www.docs.hortonworks.com

Hive runtime: From MR to Tez

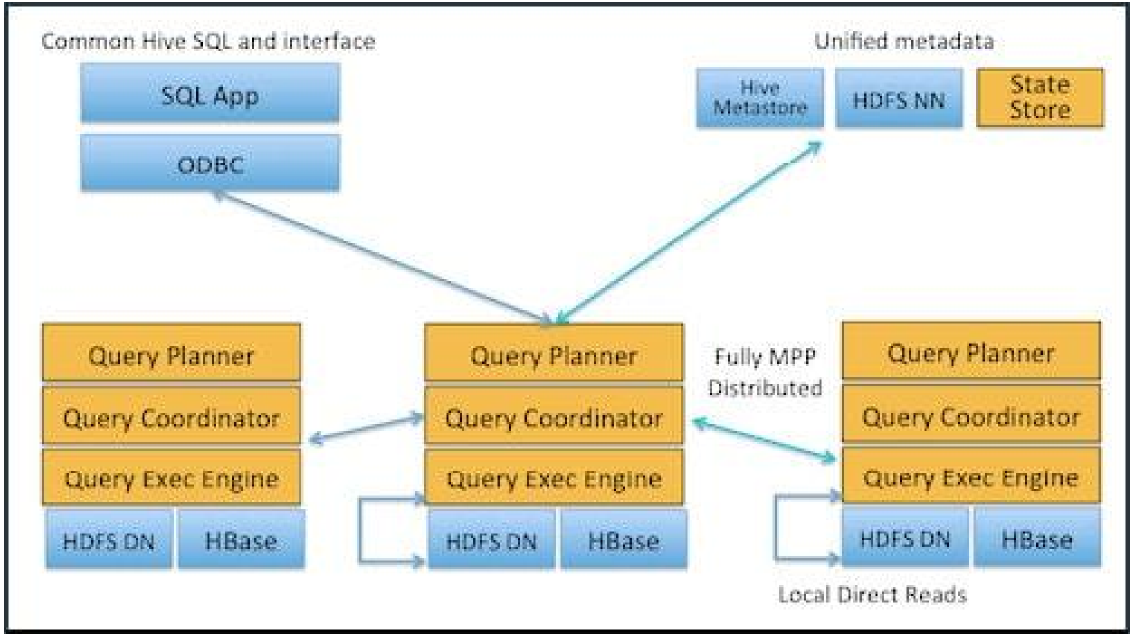


What's the big difference?

Why does it matter?

Source(s): HortonWorks, https://www.docs.hortonworks.com

Impala runtime



Fully shared-nothing architecture with no intermediate materialization

Source(s): Big Data Reviews, https://www.bigdatareviews.org/?p=121

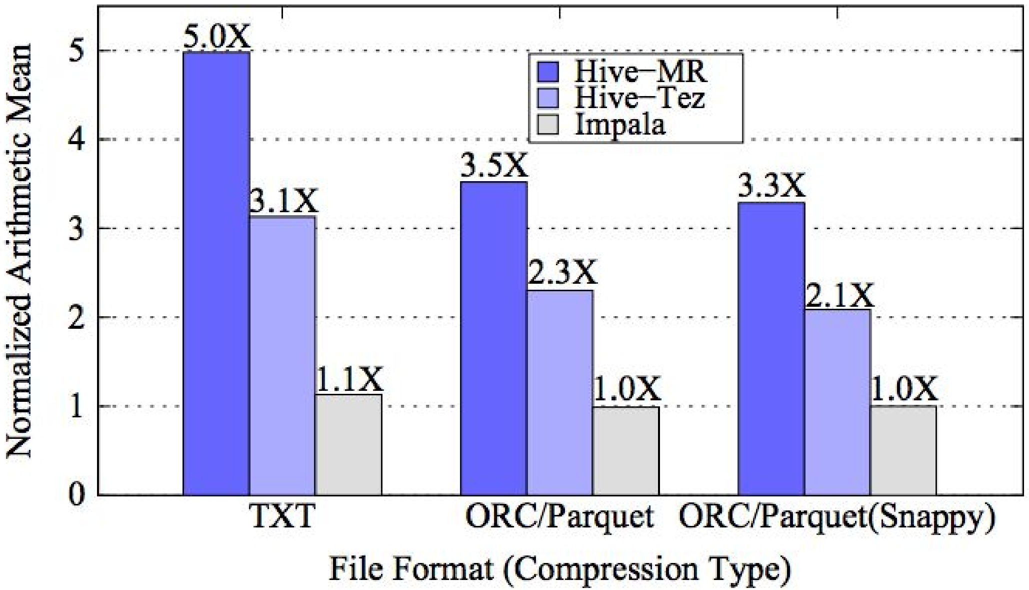

How Fast is Really Fast?
COPYRIGHT LEVON BISS 2011 - WWW.LEVONBISS.COM

Benchmarks: Loading Time

Task: Load 1TB data

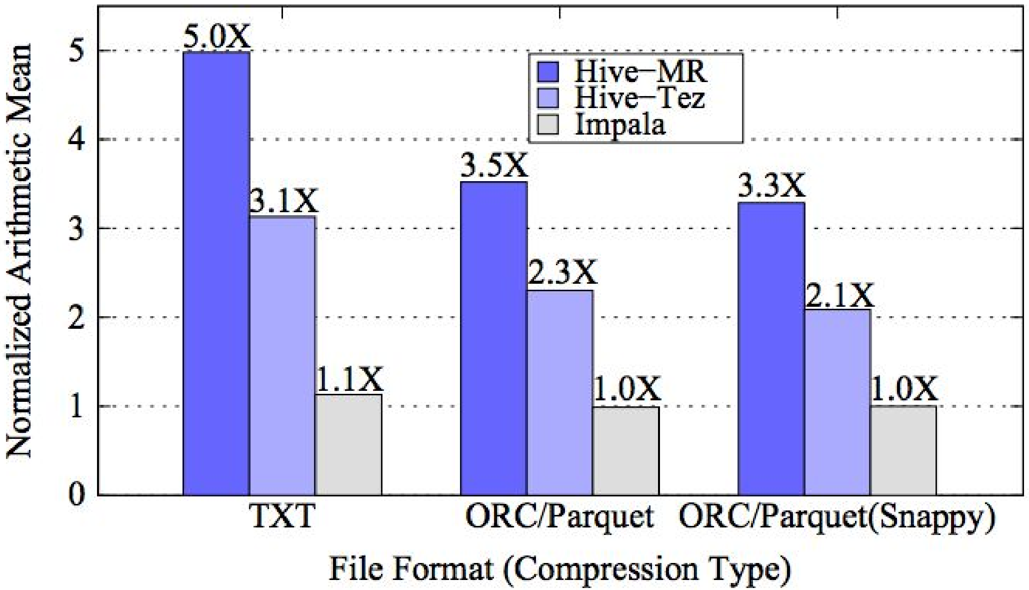
Vary: Compression and data system

Result: 

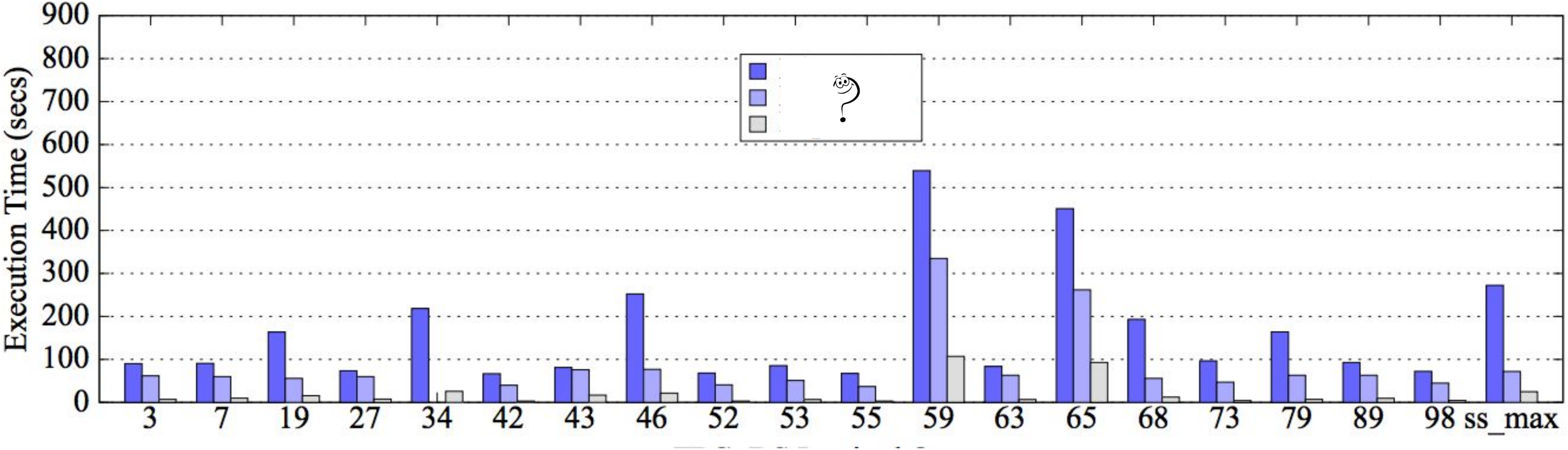


Benchmarks: Loading Time

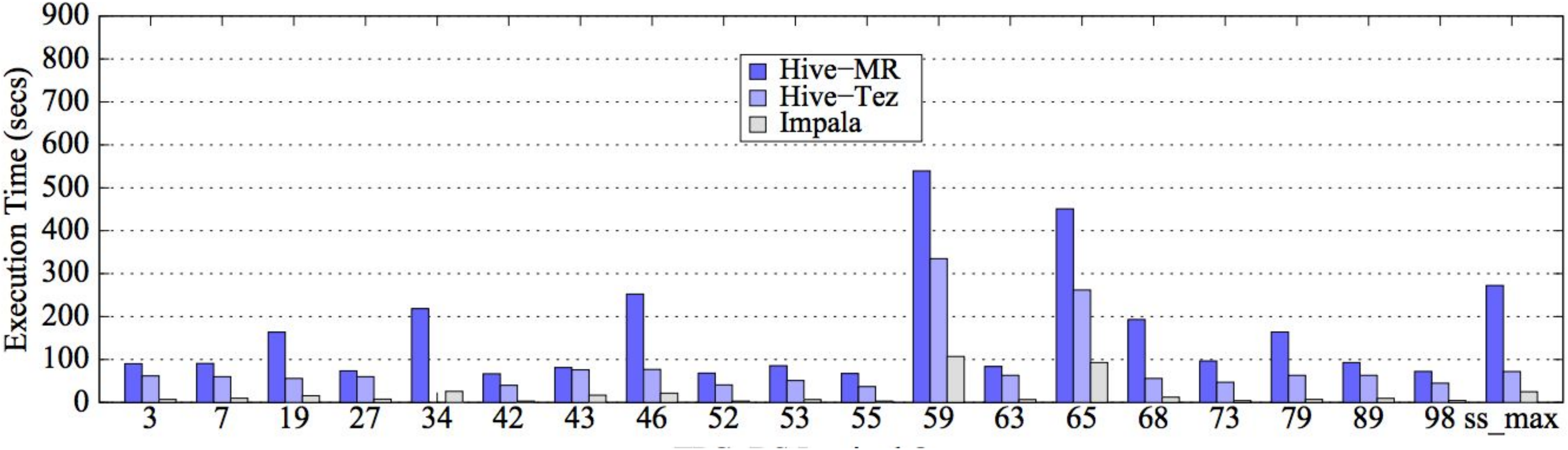
Why the difference?



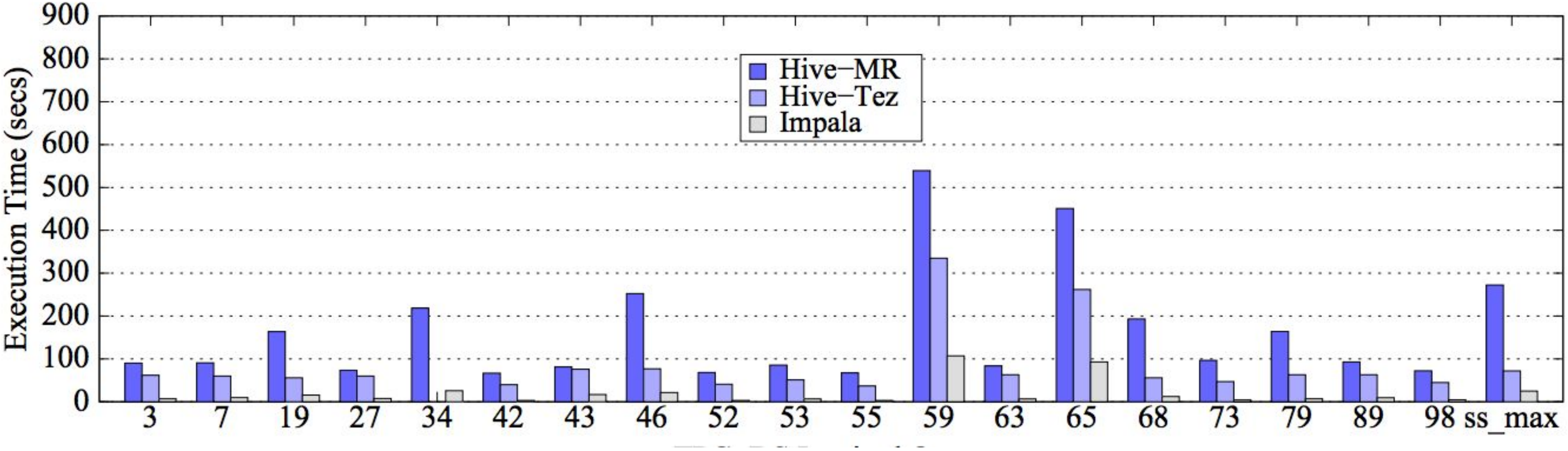
Benchmarks: Query Execution Time



Benchmarks: Query Execution Time

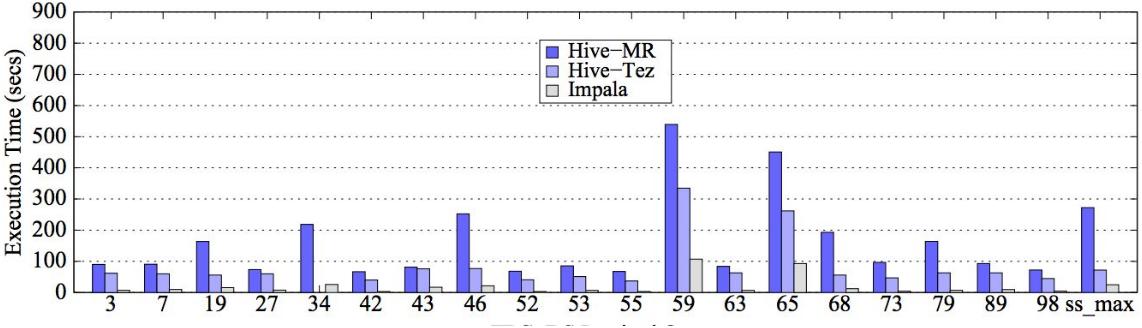


Benchmarks: Query Execution Time



Why is Impala so much faster

Benchmarks: Query Execution Time



Quiz: which of these is responsible?

(a) efficient I/O

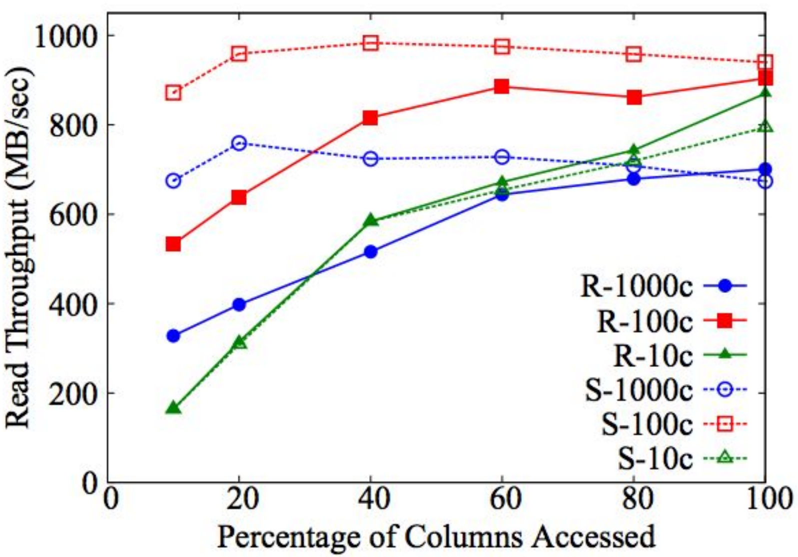
(b) no initialization overhead

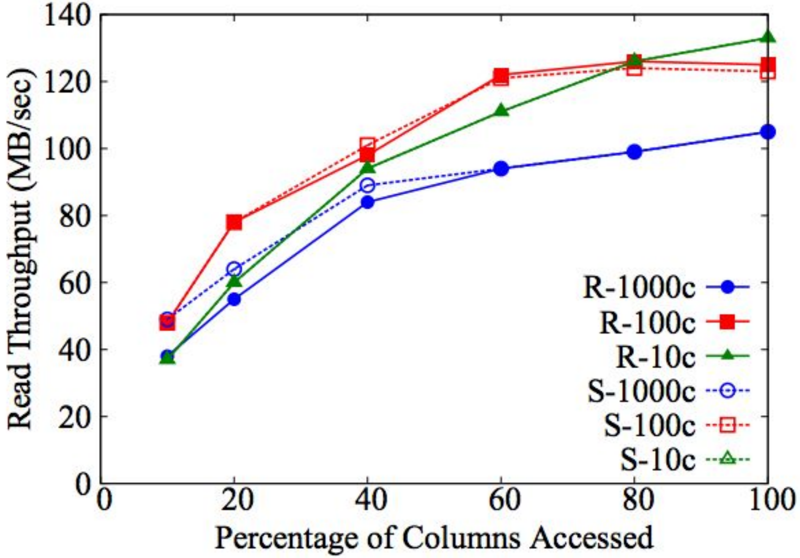
(c) pipelined rather than materialized intermediaries

(d) magic??

Why is Impala so much faster

Benchmarks: Data Access





How similar are these graphs?

Future work

Failure recovery for Impala

Workloads that exceed the size of main memory (e.g. backpressure, or buffer intermediate results to disk)

Caching common sub-DAG query results