cloudera®

Hive Now Sparks

Chao Sun | April 2014

Outline

- Background
- Architecture and Design
- Challenges
- Current Status
- Benchmarks

© Cloudera, Inc. All rights reserved.

Background

- Apache Hive: a popular data processing tool for Hadoop
- Apache Spark: a data computing framework to succeed MapReduce
- Marrying the two can benefit users from both community
- Hive-7292: The most watched JIRA in Hive (160+)

© Cloudera, Inc. All rights reserved.

Community Involvement

- Efforts from both communities (Hive and Spark)
- Contributions from many organizations

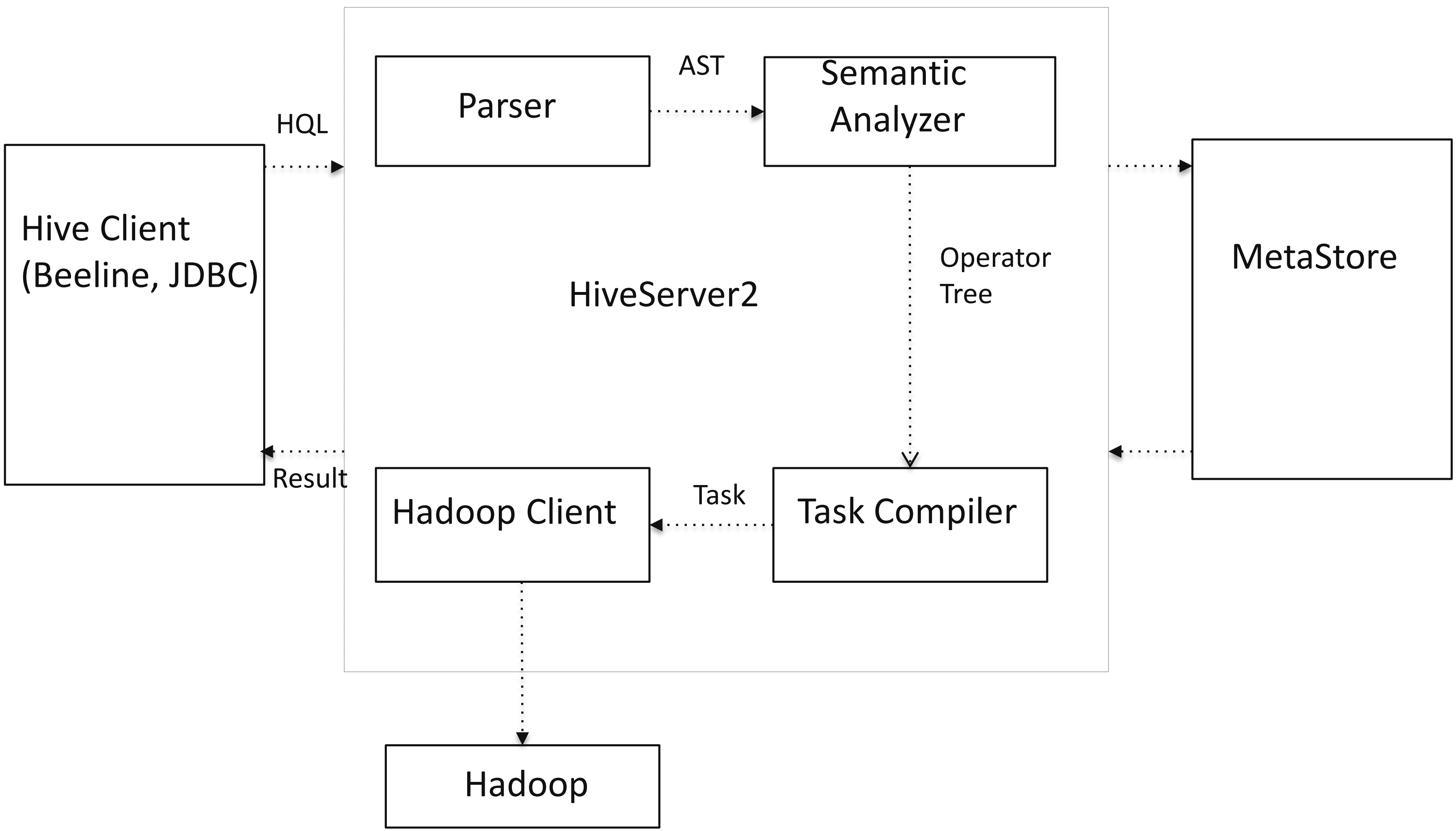
© Cloudera, Inc. All rights reserved.

Design Principles

- No or limited impact on Hive's existing code path
- Maximum code reuse
- Minimum feature customization
- Low future maintenance cost

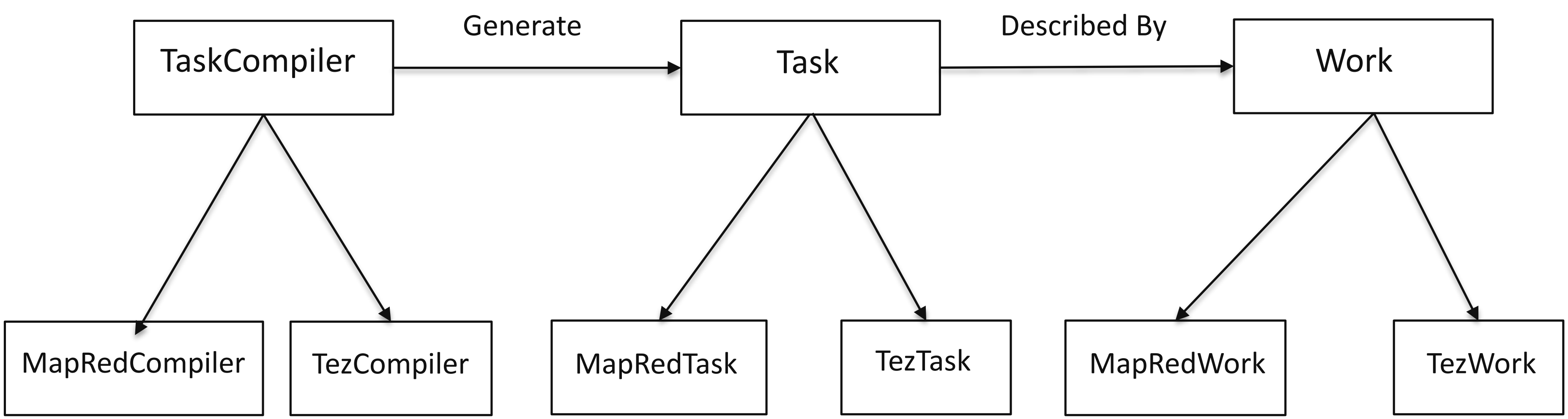
© Cloudera, Inc. All rights reserved.

Hive Internal



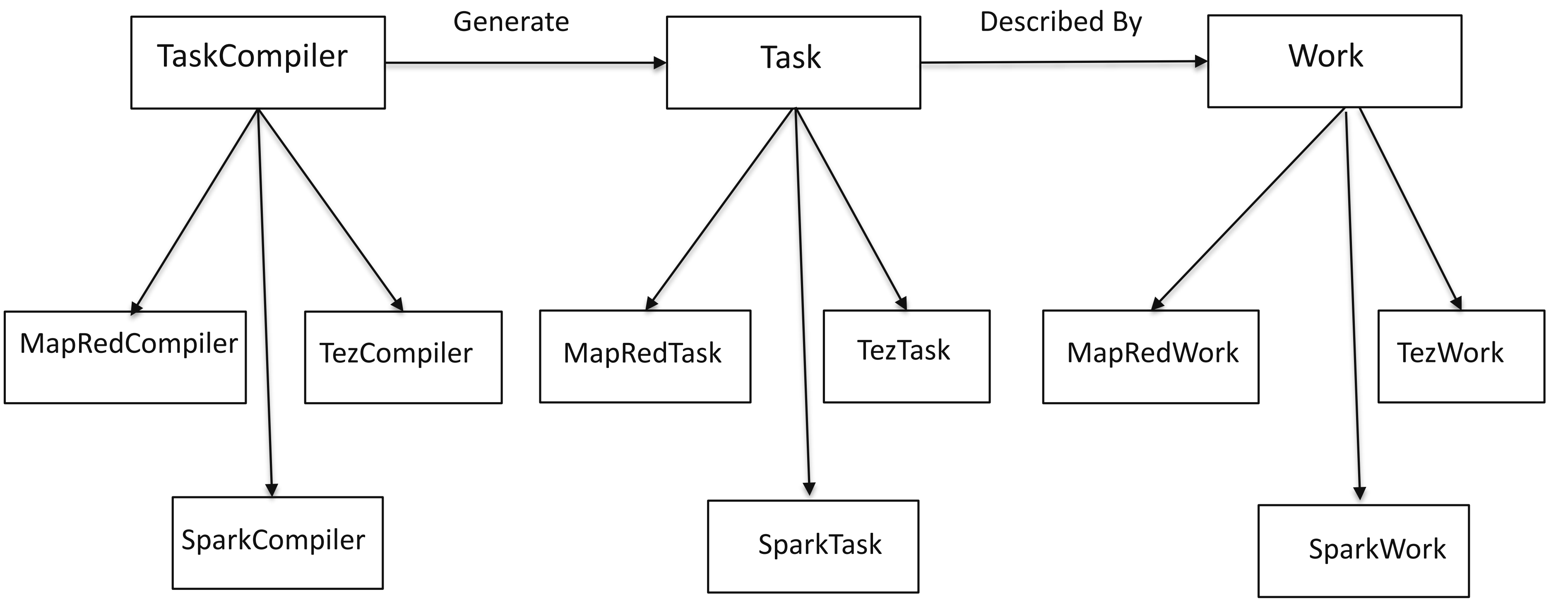
© Cloudera, Inc. All rights reserved.

Class Hierarchy



© Cloudera, Inc. All rights reserved.

Class Hierarchy



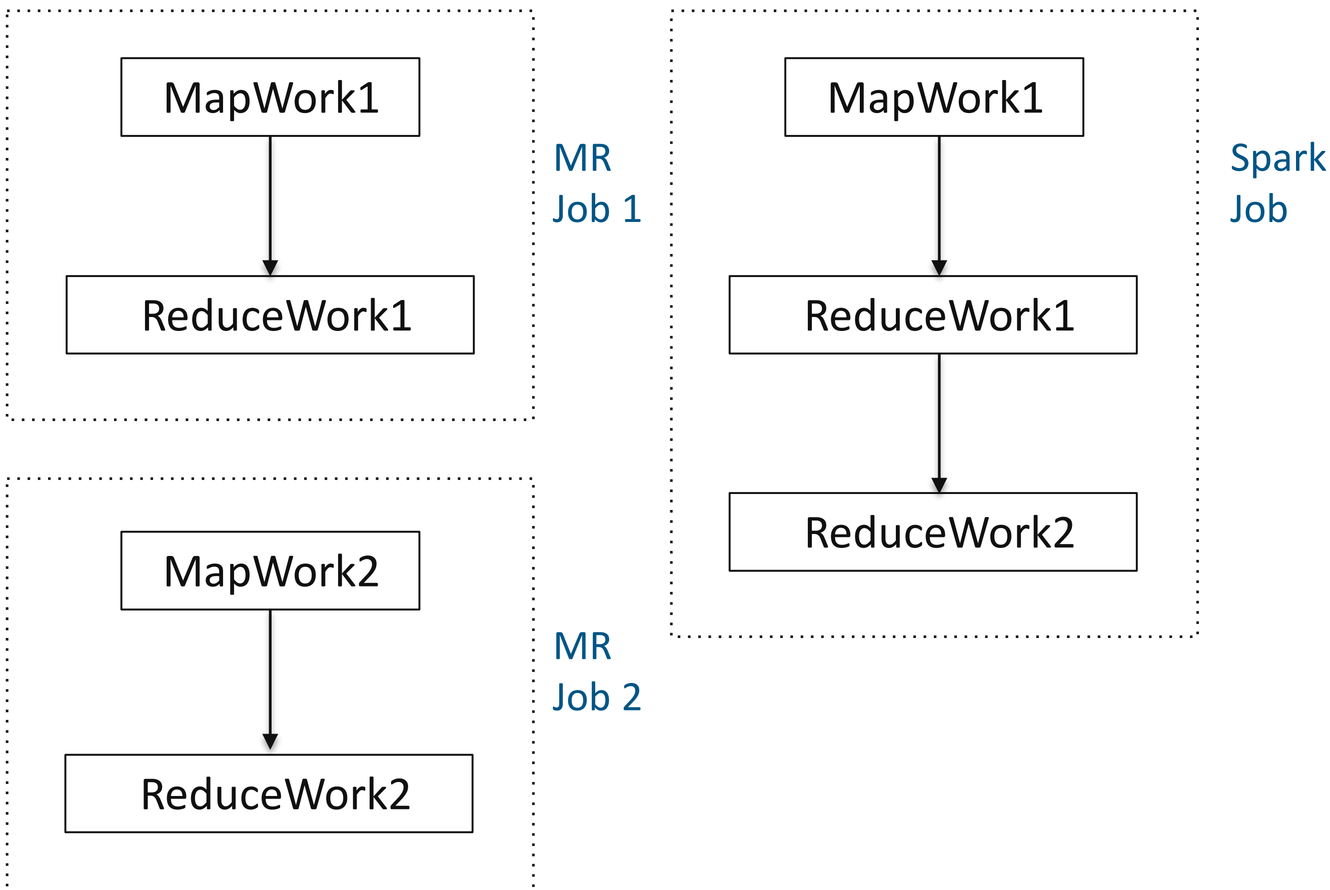
cloudera
© Cloudera, Inc. All rights reserved.

Work – Metadata for Task

- MapRedWork contains a MapWork and a possible ReduceWork
- SparkWork contains a graph of MapWorks and ReduceWorks

Ex Query:

```
SELECT name,
sum(value) AS v
FROM src
GROUP BY name
ORDER BY v;
```



© Cloudera, Inc. All rights reserved.

Spark Client

- Abreast with MR client and Tez Client
- Talk to Spark cluster
- Support local, local-cluster, standalone, yarn-cluster, and yarn-client
- Job submission, monitoring, error reporting, statistics, metrics, and counters.

cloudera

© Cloudera, Inc. All rights reserved.

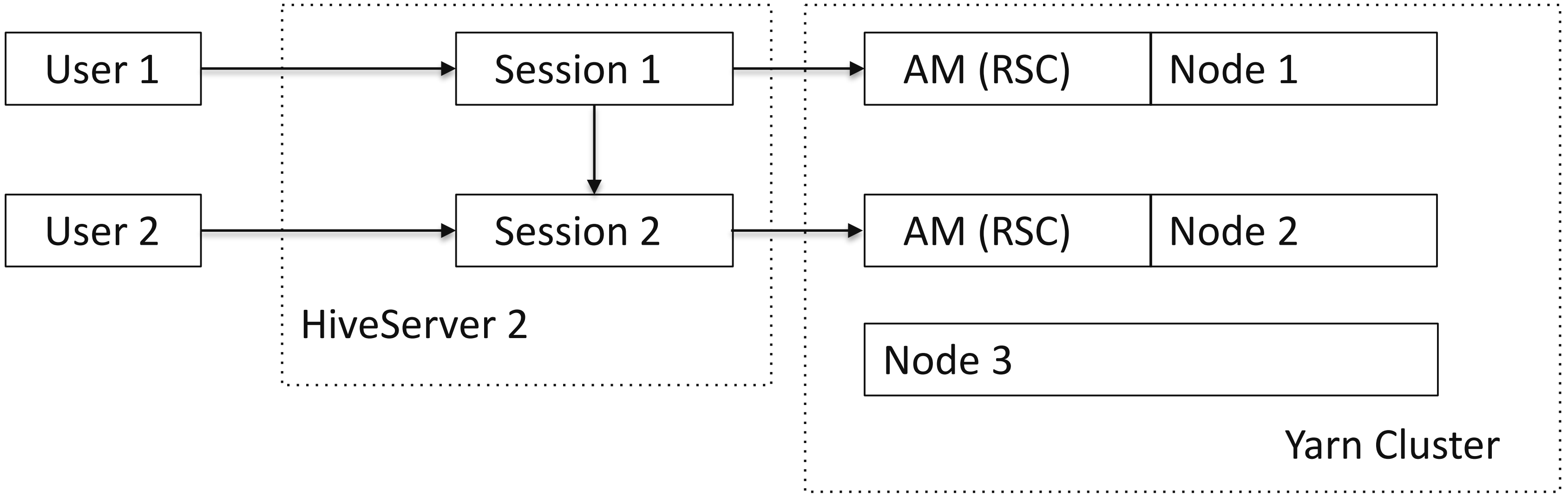
Spark Context

- Core of Spark client
- Heavy-weighted, thread-unsafe
- Designed for a single-user application
- Doesn’t work in multi-session environment
- Doesn’t scale with user sessions

© Cloudera, Inc. All rights reserved.

Remote Spark Context (RSC)

- Being created and living outside HiveServer2
- In yarn-cluster mode, Spark context lives in application master (AM)
- Otherwise, Spark context lives in a separate process (other than HiveServer2)



© Cloudera, Inc. All rights reserved.

Data Processing via MapReduce

- Table as HDFS files and read by MR framework
- Map-side processing
- Map output is shuffled by MR framework
- Reduce-side processing
- Reduce output is written to disk as part of reduce-side processing
- Output may be further processed by next MR job or returned to client

© Cloudera, Inc. All rights reserved.

Data Processing via Spark

- Treat Table as **`HadoopRDD`** (input RDD)
- Apply the function that wraps MR's map-side processing
- Shuffle map output using Spark's transformations (**`groupByKey`**, **`sortByKey`**, etc)
- Apply the function that wraps MR's reduce-side processing
- Output is either written to file or shuffled again

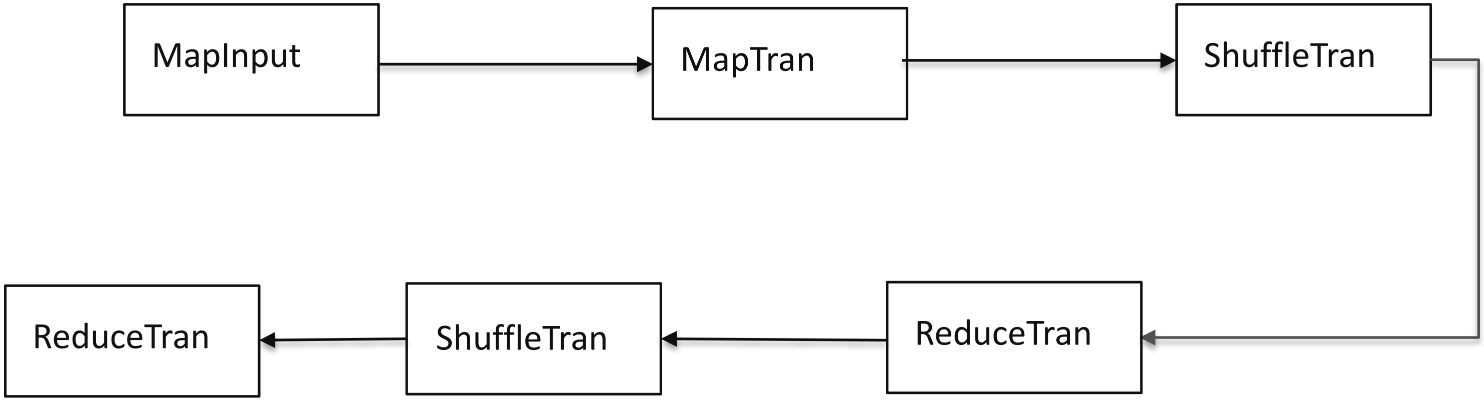
© Cloudera, Inc. All rights reserved.

Spark Plan

- **`MapInput`** – encapsulates a table
- **`MapTran`** – map-side processing
- **`ShuffleTran`** – shuffling
- **`ReduceTran`** – reduce-side processing

Ex Query:

```
SELECT name,
sum(value) AS v
FROM src
GROUP BY name
ORDER BY v;
```



© Cloudera, Inc. All rights reserved.

Advantages

- Reuse existing map-side and reduce-side processing
- Agonistic to Spark’s special transformations or actions
- No need to reinvent wheels
- Adopt existing features: authorization, window functions, UDFs, etc
- Open to future features

© Cloudera, Inc. All rights reserved.

Challenges

- Missing features or functional gaps in Spark
  - Concurrent data pipelines
  - Spark Context issues
  - Scala vs Java API
  - Scheduling issues
- Large code base in Hive, many contributors working in different areas
- Library dependency conflicts among projects

© Cloudera, Inc. All rights reserved.

Dynamic Executor Scaling

- Spark cluster per user session
- Heavy user vs light user
- Big query vs small query
- Solution: executors up and down based on workload

© Cloudera, Inc. All rights reserved.

Current Status

- All functionality is implemented
- First round of optimization is completed
- More optimization and benchmarking coming
- Beta release in CDH5.4
- Released in Apache Hive 1.1.0
- Follow HIVE-7292 for current and future work

© Cloudera, Inc. All rights reserved.

Optimizations

- Map join, bucket map join, SMB, skew join (static and dynamic)
- Split generating and grouping
- CBO, vectorization
- More to come, including table caching, dynamic partition pruning

© Cloudera, Inc. All rights reserved.

Summary

- Community driven project
- Multi-organization support
- Combining merits from multiple projects
- Benefiting a large user base
- Bearing solid foundations
- A solid, evolving project

© Cloudera, Inc. All rights reserved.

Benchmarks – Cluster Setup

- Done by our team members from Intel
- 8 physical nodes
- Each has 32 logical cores and 64GB memory
- 10000Mb/s network between the nodes

Component	Version
Hive	spark-branch
Spark	1.3.0
Hadoop	2.6.0
Tez	0.5.3

© Cloudera, Inc. All rights reserved.

Benchmarks – Test Configuration

- 320GB and 4TB TPC-DS datasets
- Three engines share the most of the configurations
  - Each node is allocated 32 cores and 48GB memory
  - Vectorization enabled
  - CBO enabled
  - **`Hive.auto.convert.join.noconditionaltask.size = 600MB`**

© Cloudera, Inc. All rights reserved.

Benchmarks – Test Configurations

- Hive on Tez
  - `hive.prewarm.numcontainers = 250`
  - `hive.tez.auto.reducer.parallelism = true`
  - `hive.tez.dynamic.partition.pruning = true`

- Hive on Spark
  - `spark.master = yarn-client`
  - `spark.executor.memory = 5120m`
  - `spark.yarn.executor.memoryOverhead = 1024`
  - `spark.executor.cores = 4`
  - `spark.kryo.referenceTracking = false`
  - `spark.io.compression.codec = lzf`

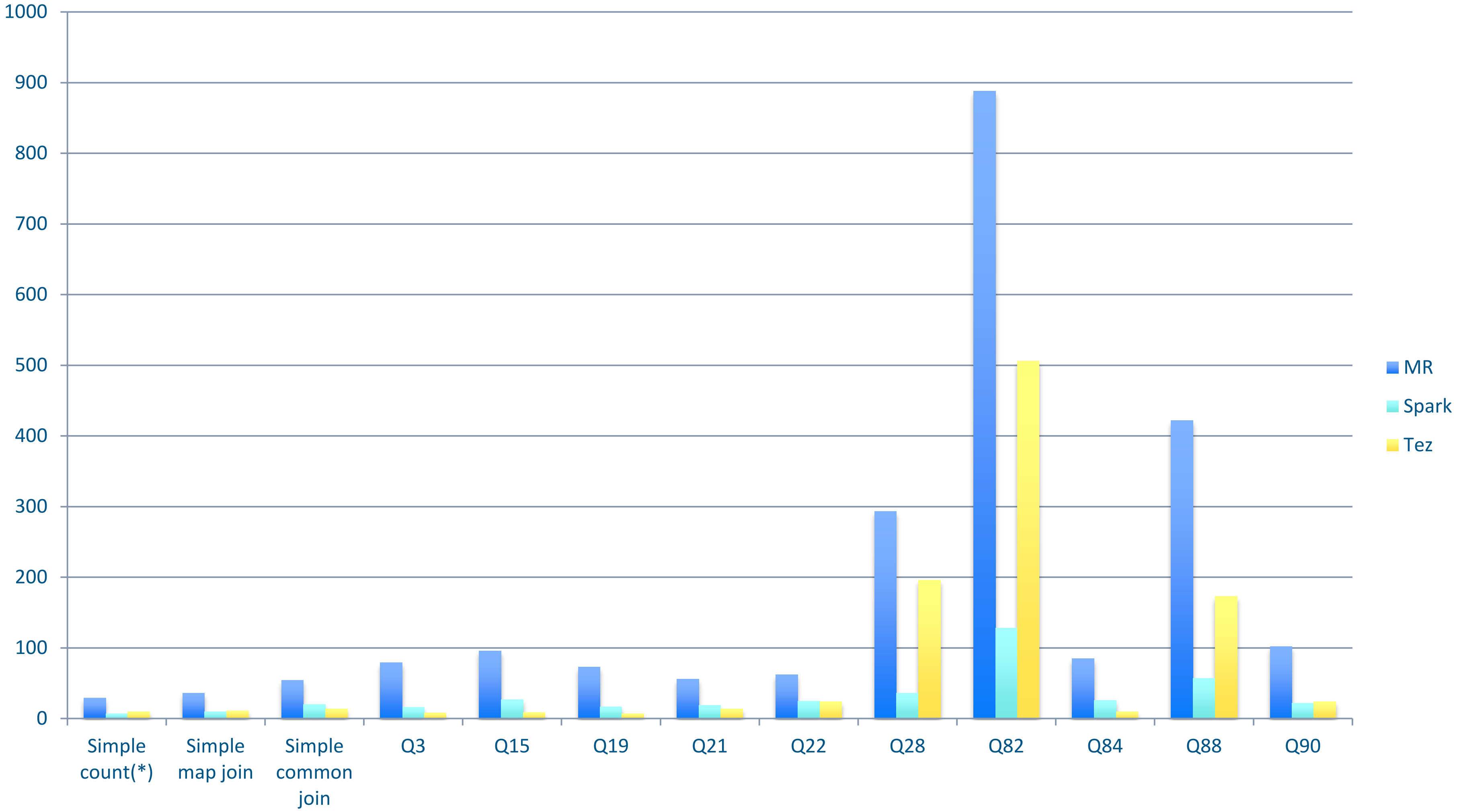
© Cloudera, Inc. All rights reserved.

Benchmarks – Data Collecting

- We run each query 2 times and measure the 2nd run
- Spark on yarn waits for a number of executors to register before scheduling tasks, thus with a bigger start-up overhead
- We measure a few queries for Hive on Tez w/ or w/o dynamic partition pruning for comparison, as this optimization hasn't been implemented in Hive on Spark yet

© Cloudera, Inc. All rights reserved.

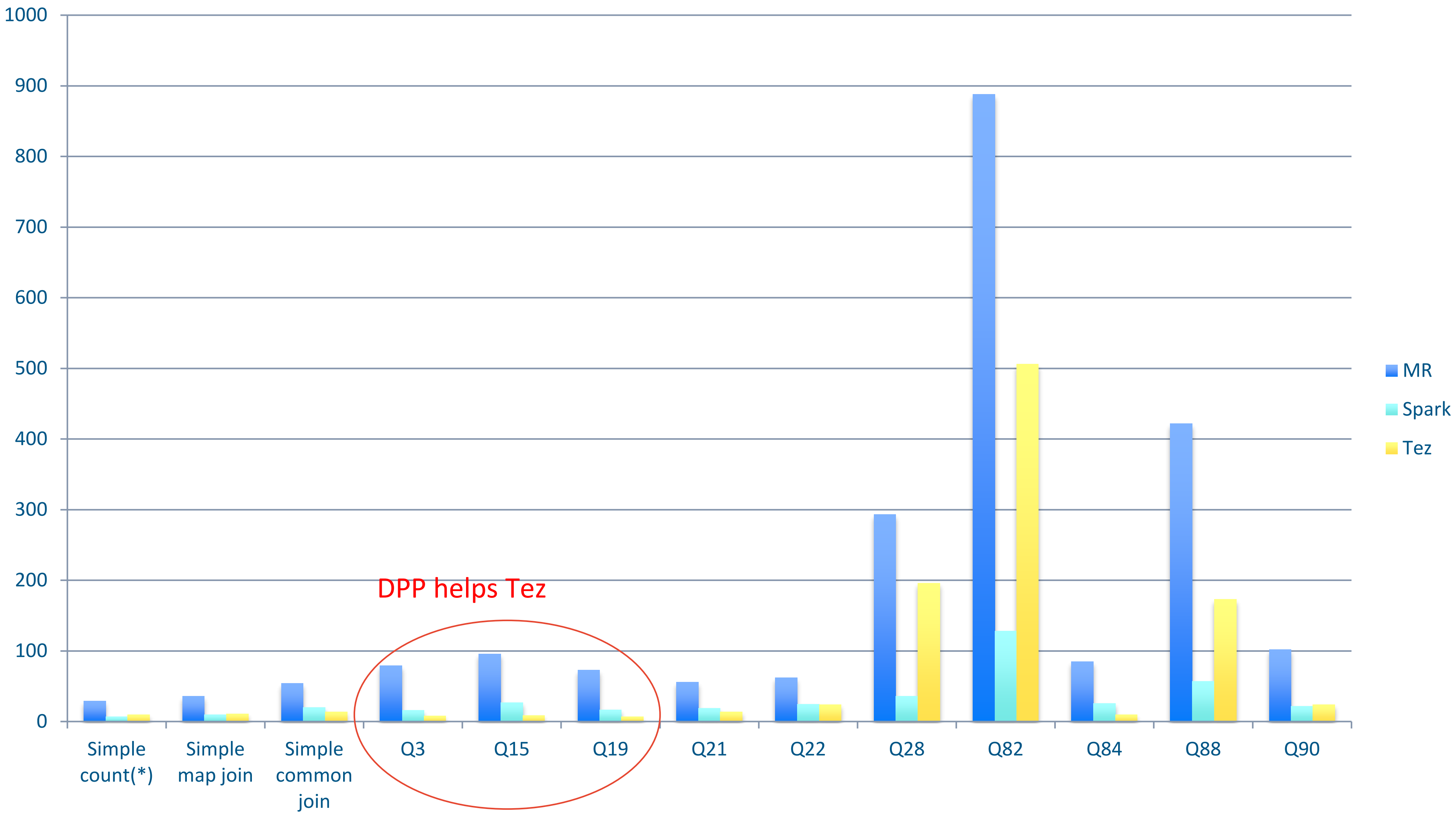
MR vs Spark vs Tez, 320GB



cloudera

© Cloudera, Inc. All rights reserved.

MR vs Spark vs Tez, 320GB



cloudera

© Cloudera, Inc. All rights reserved.

Benchmark -Dynamic Partition Pruning

- Prune partitions at runtime
  - Ex Query:

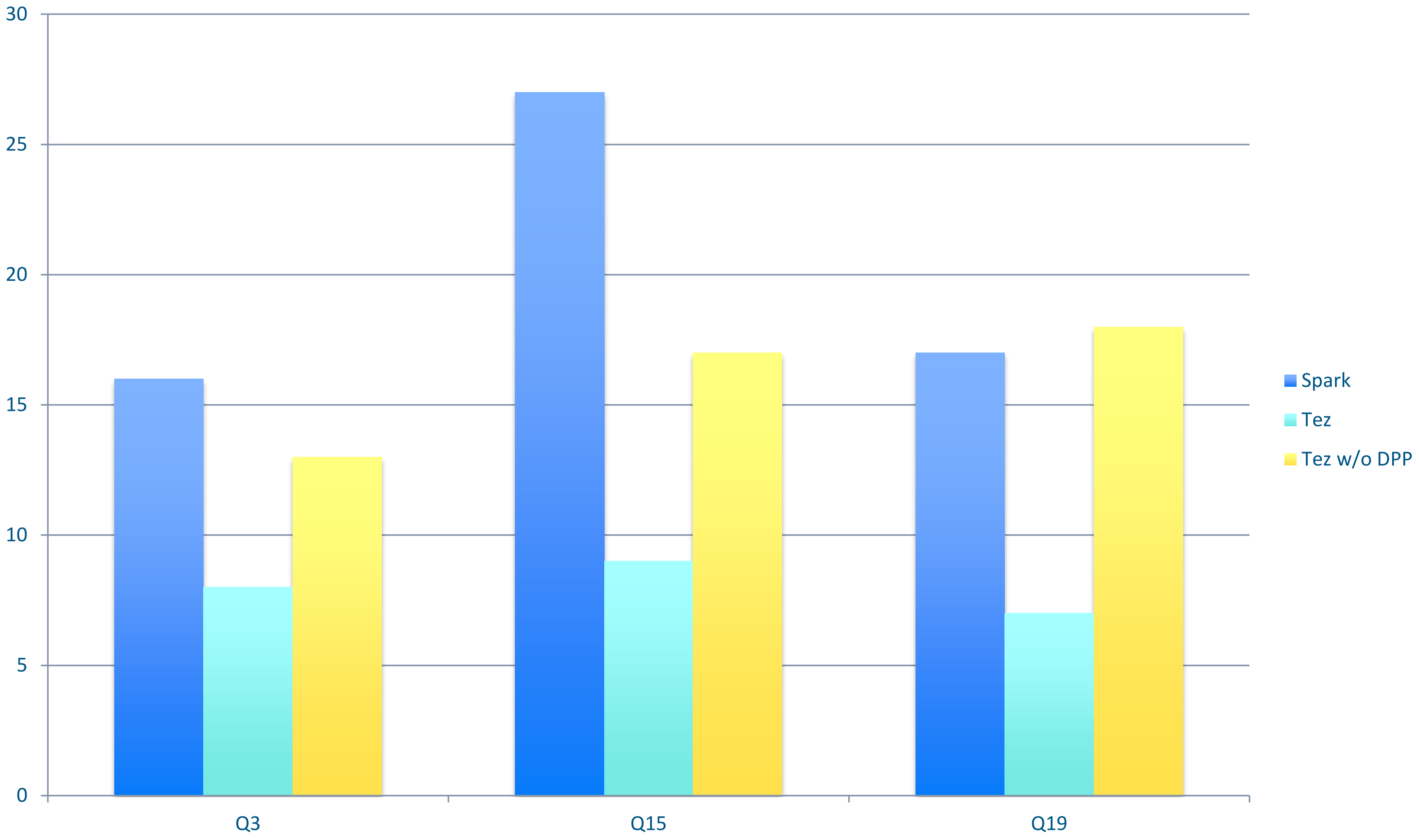
```
SELECT count(*) FROM src JOIN src_date ON (src.ds =
src_date.ds) WHERE src_date.`date` = '2015-04-16'

(src is partitioned on ds, but src_date is NOT
partitioned.)
```

- Can dramatically improve performance when tables are joined on partitioned columns
- To be implemented in Hive on Spark

© Cloudera, Inc. All rights reserved.

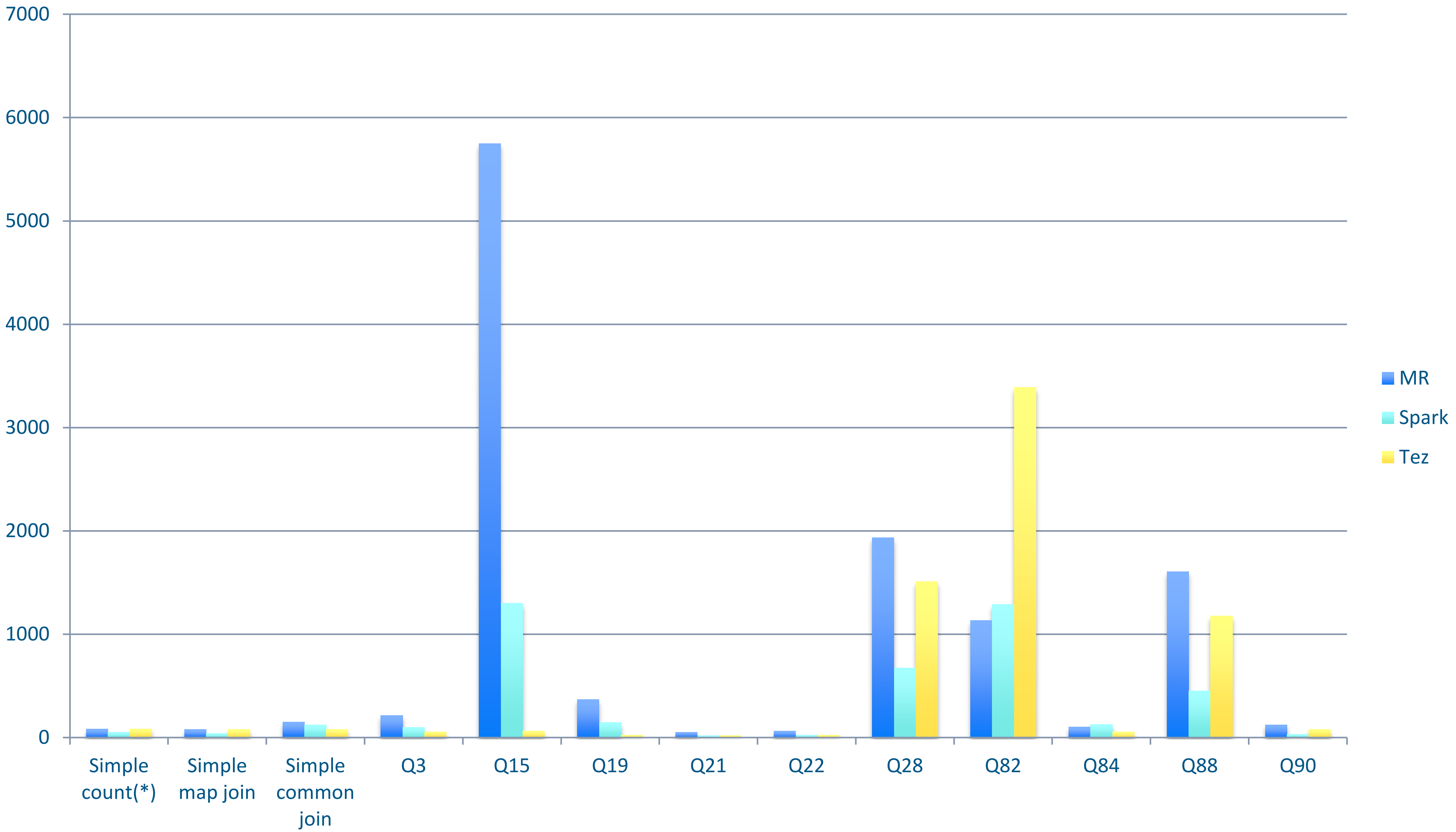
Spark vs Tez vs Tez w/o DPP, 320GB



cloudera

© Cloudera, Inc. All rights reserved.

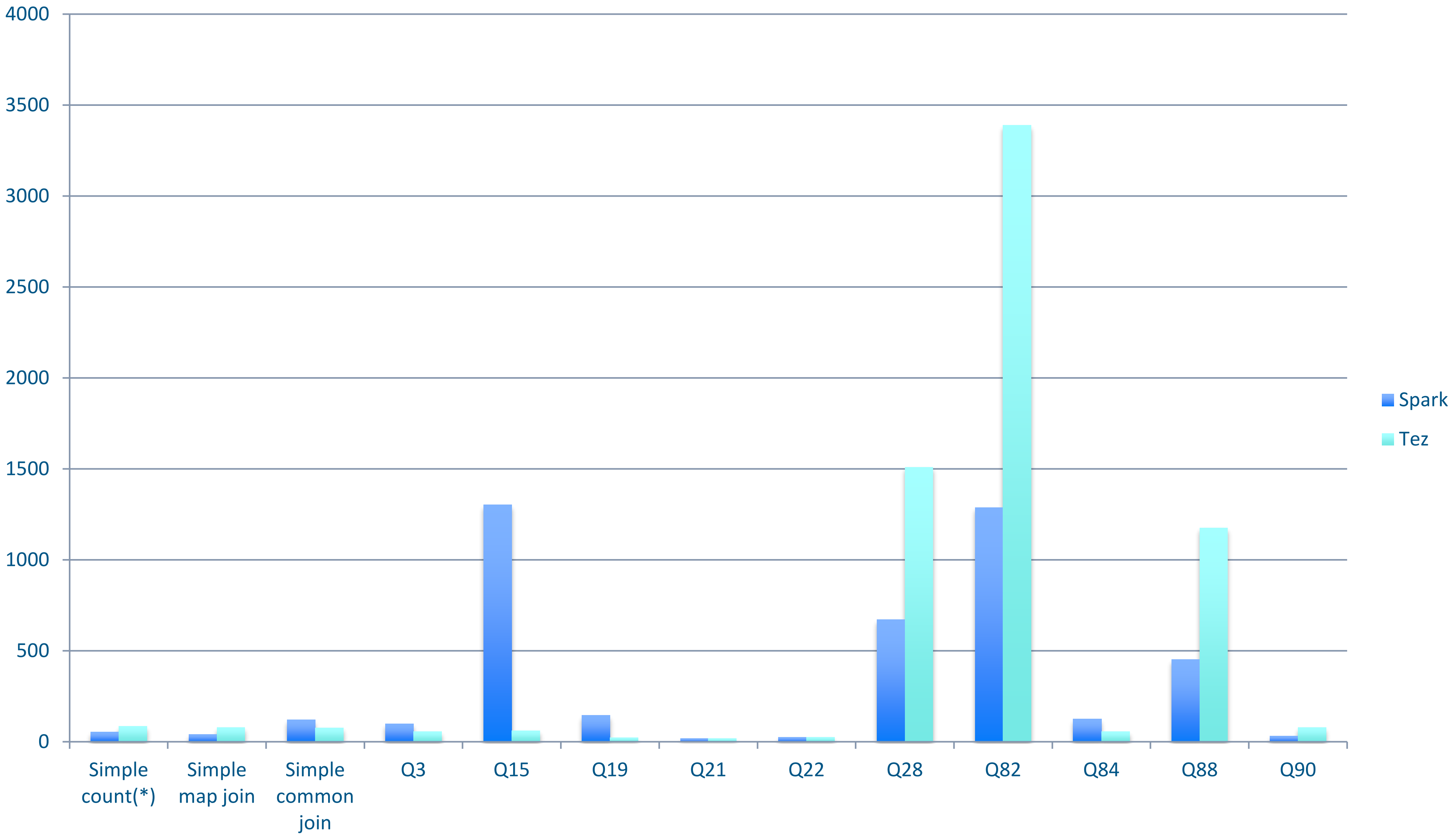
MR vs Spark vs Tez, 4TB



cloudera

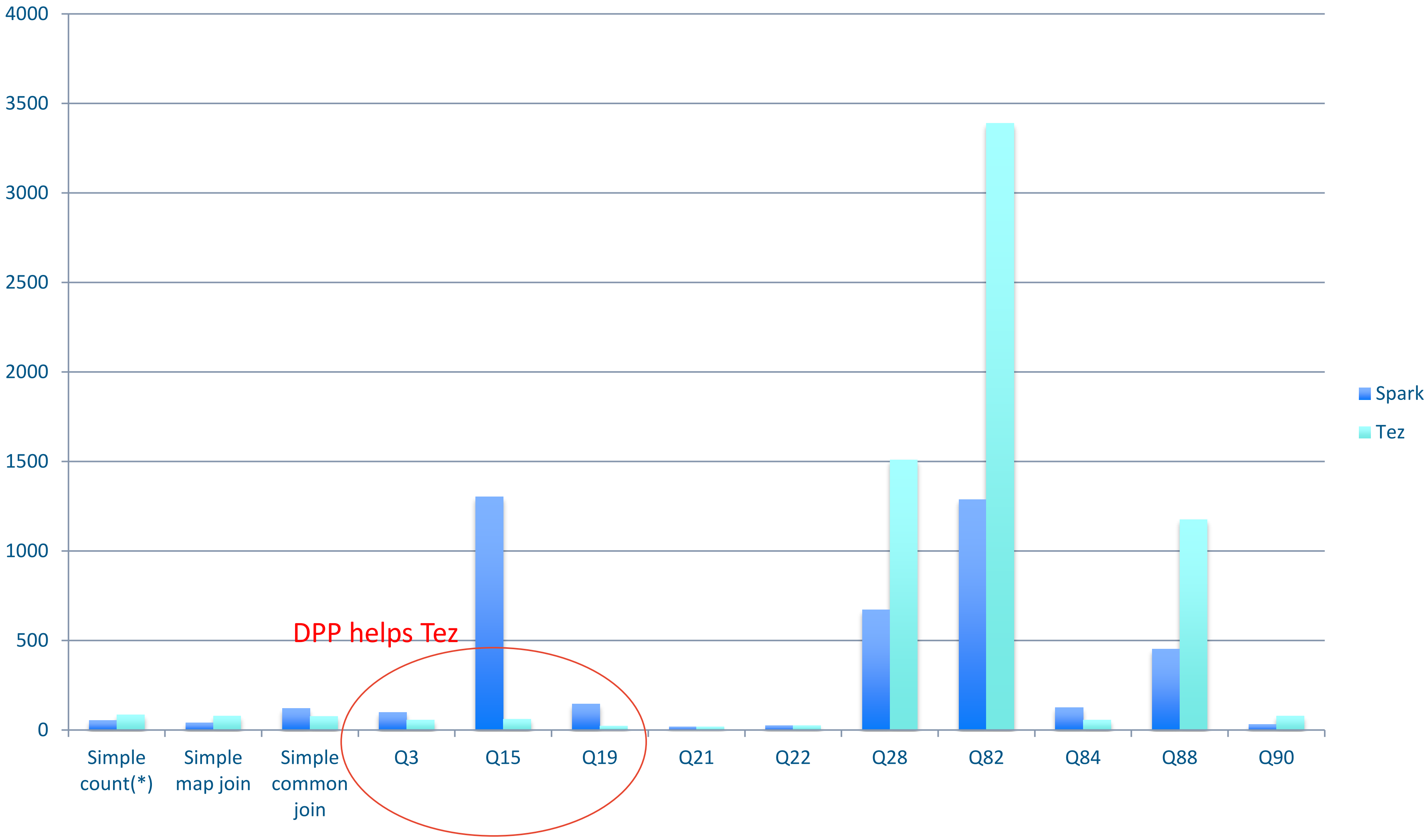
© Cloudera, Inc. All rights reserved.

Spark vs Tez, 4TB



© Cloudera, Inc. All rights reserved.

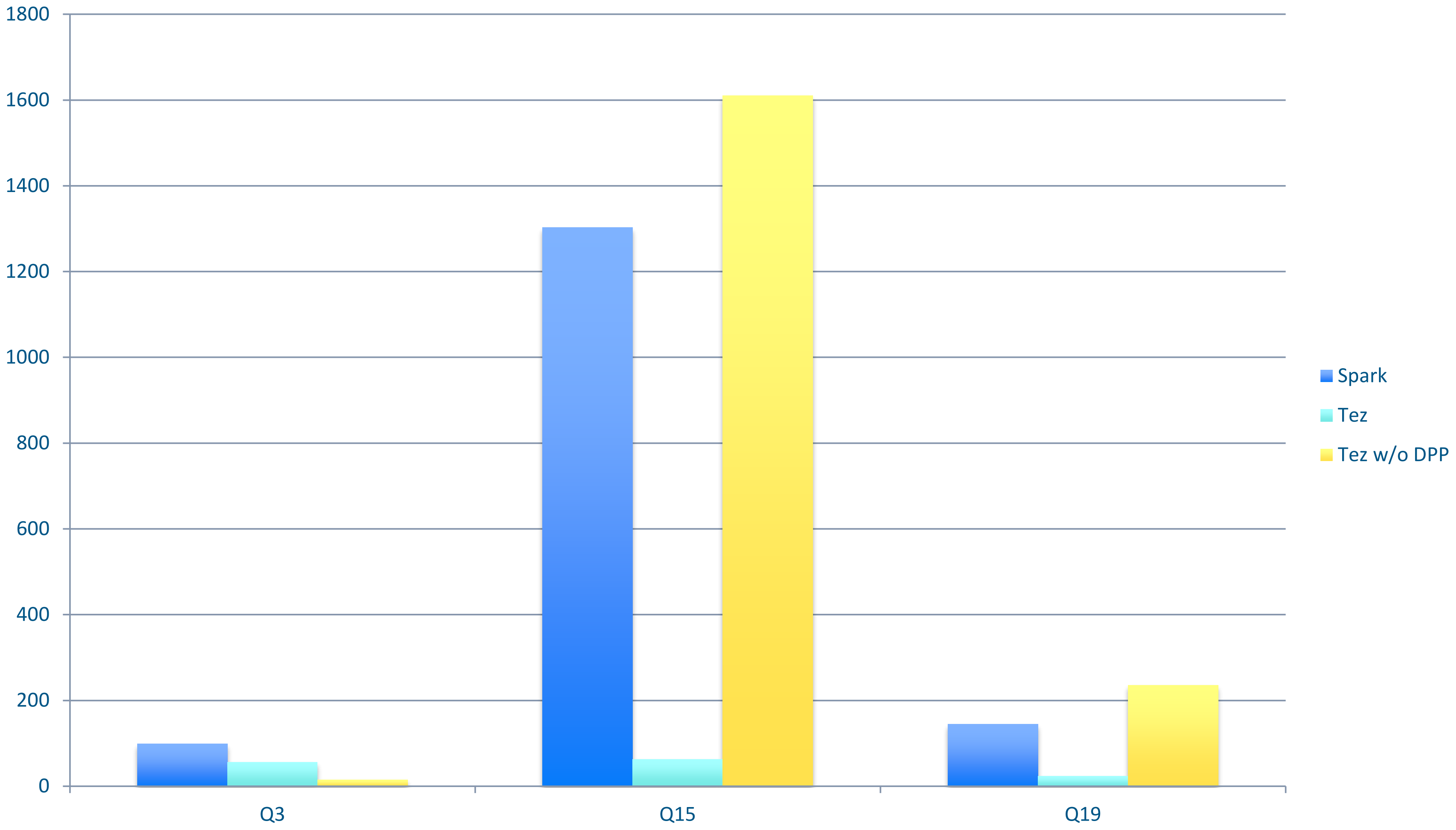
Spark vs Tez, 4TB



cloudera

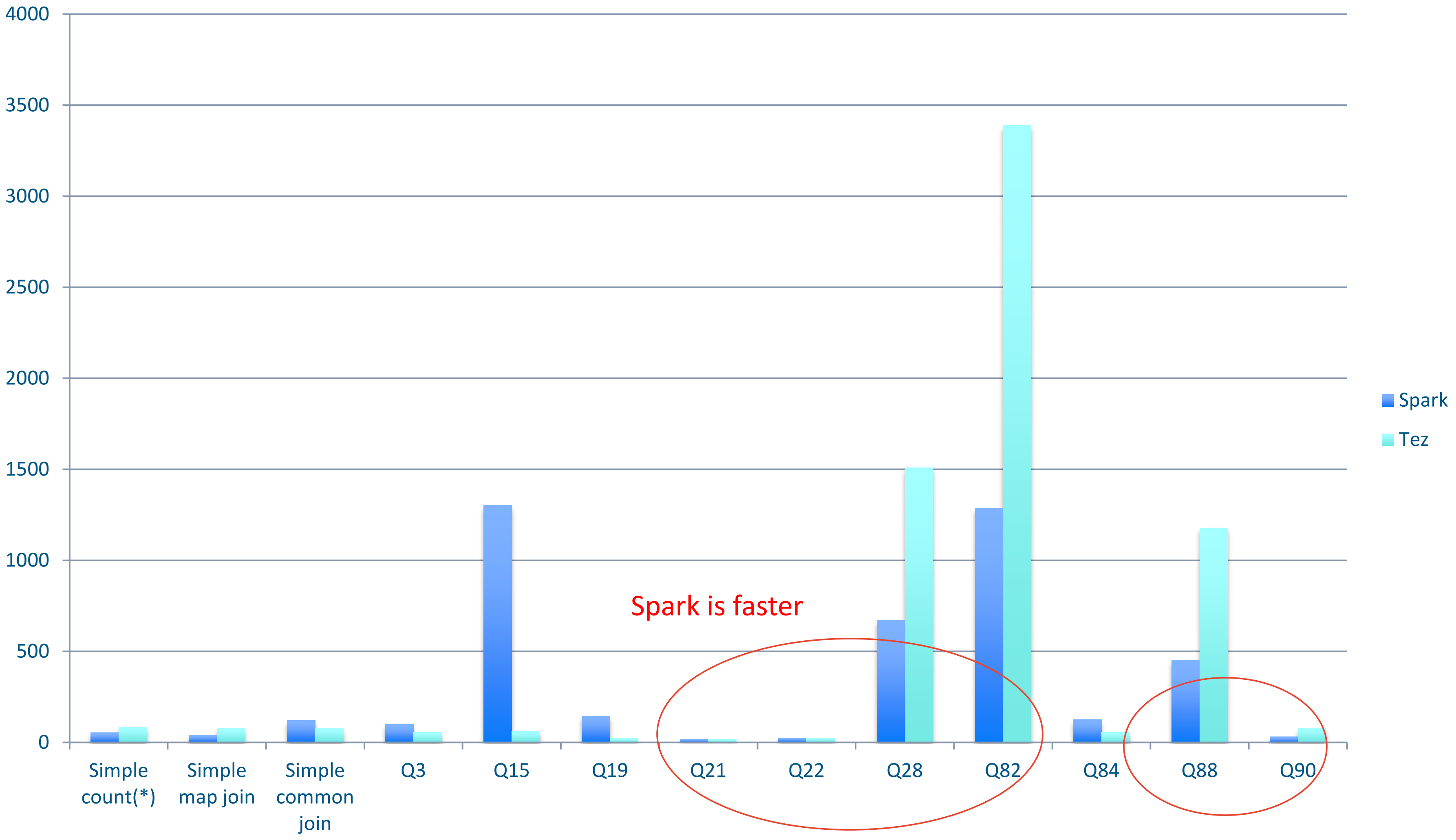
© Cloudera, Inc. All rights reserved.

Spark vs Tez vs Tez w/o DPP, 4TB



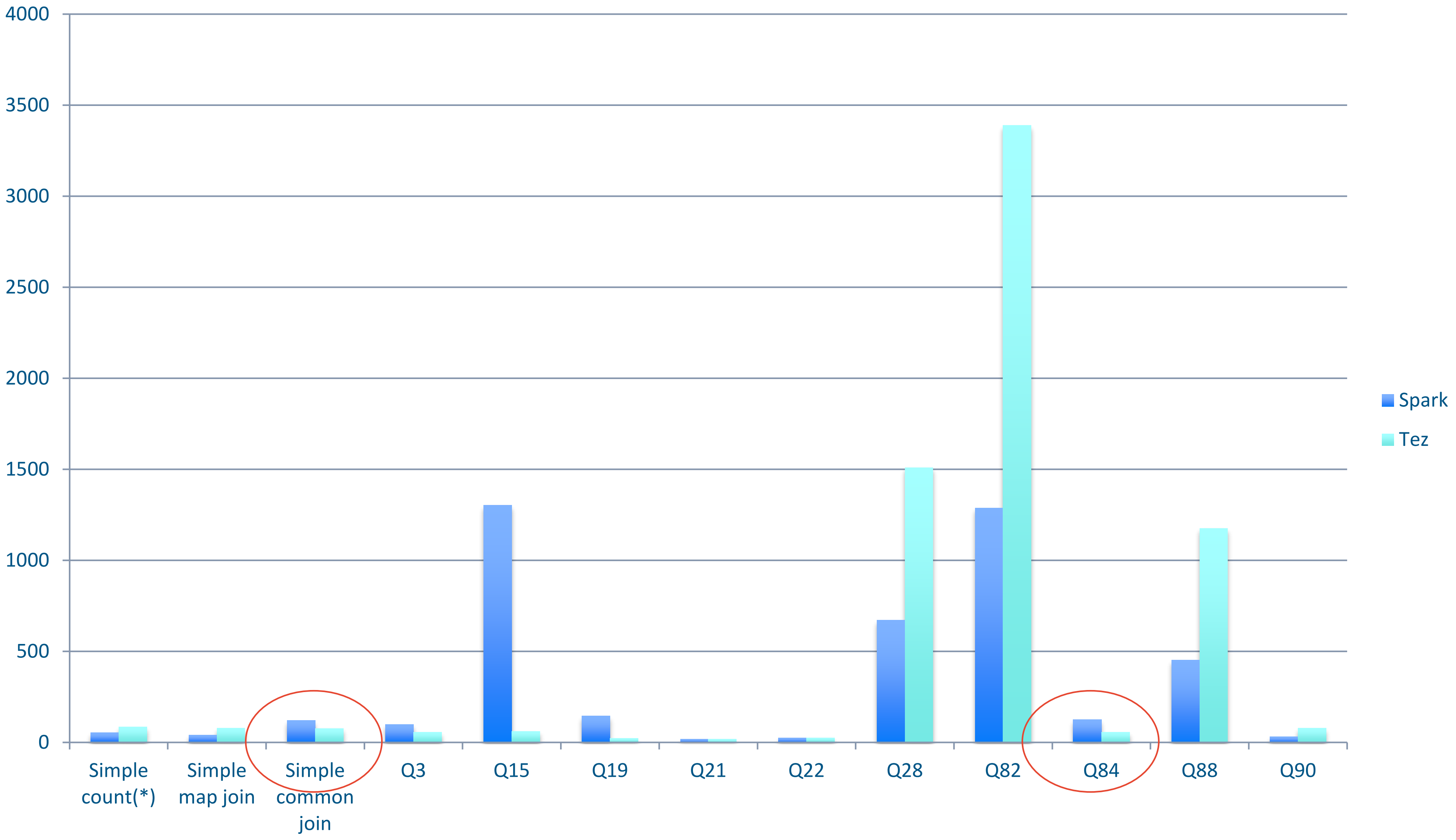
© Cloudera, Inc. All rights reserved.

Spark vs Tez, 4TB



© Cloudera, Inc. All rights reserved.

Spark vs Tez, 4TB



cloudera

© Cloudera, Inc. All rights reserved.

Benchmarks - Summary

- Spark is as fast as or faster than Tez on many queries (Q28, Q88, etc.)
- Dynamic partition pruning helps Tez a lot on certain queries (Q3, Q15, Q19). Without DPP for Tez, Spark is close or even faster
- Spark is slower on certain queries (common join, Q84) than Tez. Investigation and improvement is on the way
- Bigger dataset seems to help Spark more
- Spark will likely be faster after DPP is implemented

© Cloudera, Inc. All rights reserved.

cloudera

Questions?

© Cloudera, Inc. All rights reserved.