Apache Hive & Stinger

Petabyte-scale SQL in Hadoop

Gunther Hagleitner
(*gunther@apache.org*)

Batch AND Interactive SQL-IN-Hadoop

Stinger Initiative

A broad, community-based effort to drive the next generation of HIVE

Goals:

Speed

Improve Hive query performance by 100X to allow for interactive query times (seconds)

Scale

The only SQL interface to Hadoop designed for queries that scale from TB to PB

SQL

Support broadest range of SQL semantics for analytic applications running against Hadoop

…all IN Hadoop

Stinger Project

(announced February 2013)

Hive 0.11, May 2013:

- Base Optimizations
- SQL Analytic Functions
- ORCFile, Modern File Format

Hive 0.12, October 2013:

- VARCHAR, DATE Types
- ORCFile predicate pushdown
- Advanced Optimizations
- Performance Boosts via YARN

Coming Soon:

- Hive on Apache Tez
- Query Service
- Buffer Cache
- Cost Based Optimizer (Optiq)
- Vectorized Processing

Hive 0.12

	Hive 0.12
Release Theme	Speed, Scale and SQL
Specific Features	• 10x faster query launch when using large number (500+) of partitions • ORC File predicate pushdown speeds queries • Evaluate LIMIT on the map side • Parallel ORDER BY • New query optimizer • Introduces VARCHAR and DATE data types • GROUP BY on structs or unions
Included Components	Apache Hive 0.12

SQL: Enhancing SQL Semantics

Hive SQL Datatypes

- 🟩 INT
- 🟩 TINYINT/SMALLINT/BIGINT
- 🟩 BOOLEAN
- 🟩 FLOAT
- 🟩 DOUBLE
- 🟩 STRING
- 🟩 TIMESTAMP
- 🟩 BINARY
- 🟩 DECIMAL
- 🟩 ARRAY, MAP, STRUCT, UNION
- 🟧 DATE
- 🟧 VARCHAR
- 🟦 CHAR

Hive SQL Semantics

- 🟩 SELECT, INSERT
- 🟩 GROUP BY, ORDER BY, SORT BY
- 🟩 JOIN on explicit join key
- 🟩 Inner, outer, cross and semi joins
- 🟩 Sub-queries in FROM clause
- 🟩 ROLLUP and CUBE
- 🟩 UNION
- 🟩 Windowing Functions (OVER, RANK, etc)
- 🟩 Custom Java UDFs
- 🟩 Standard Aggregation (SUM, AVG, etc.)
- 🟩 Advanced UDFs (ngram, Xpath, URL)
- 🟦 Sub-queries in WHERE, HAVING
- 🟦 Expanded JOIN Syntax
- 🟦 SQL Compliant Security (GRANT, etc.)
- 🟦 INSERT/UPDATE/DELETE (ACID)

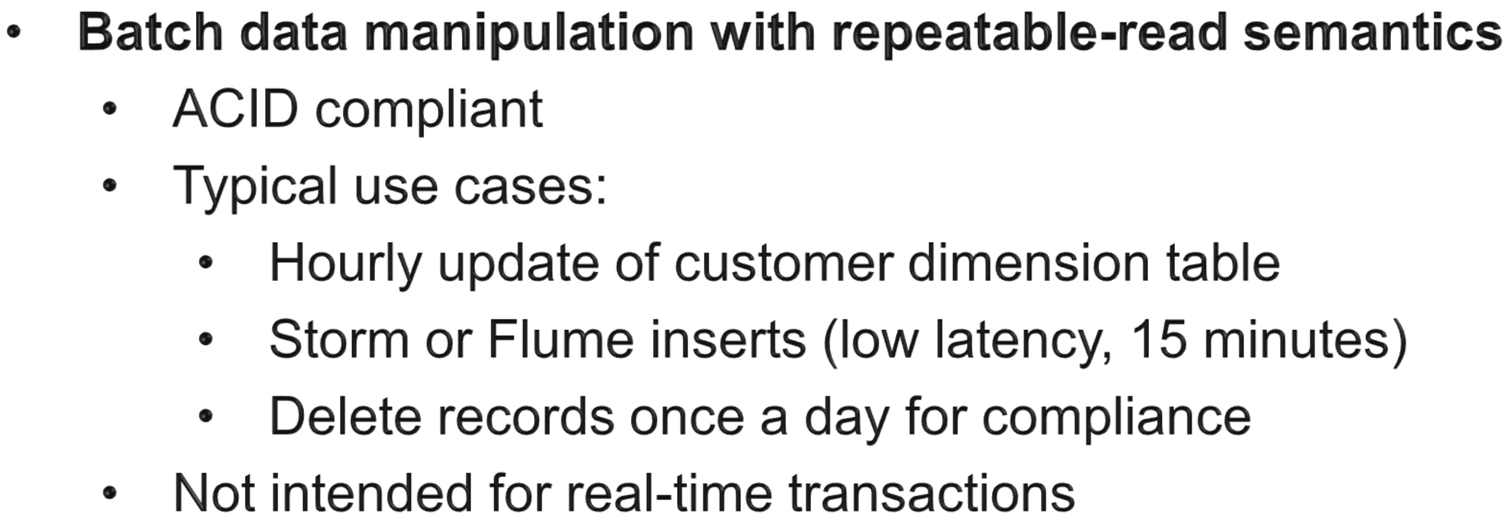
SQL Compliance

Hive 12 provides a wide array of SQL data types and semantics so your existing tools integrate more seamlessly with Hadoop

🟩	Available
🟧	Hive 0.12
🟦	Roadmap

HIVE-5317

Insert, Update and Delete (ACID)

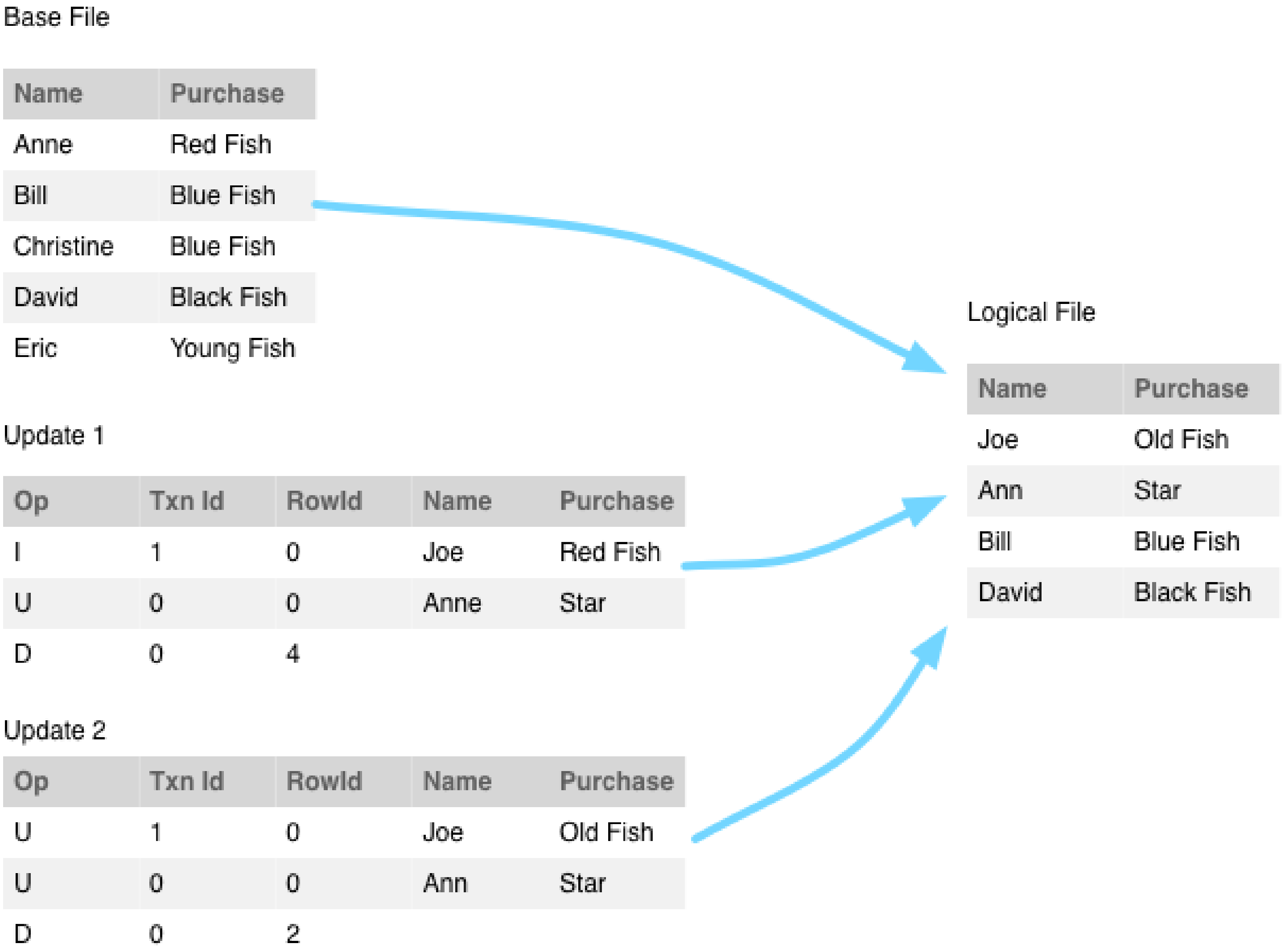


- **Batch data manipulation with repeatable-read semantics**
  - ACID compliant
  - Typical use cases:
    - Hourly update of customer dimension table
    - Storm or Flume inserts (low latency, 15 minutes)
    - Delete records once a day for compliance
  - Not intended for real-time transactions

- **Additional SQL statements:**
  - INSERT INTO table SELECT …
  - INSERT INTO table VALUES …
  - UPDATE table SET … WHERE …
  - DELETE FROM table WHERE …
  - MERGE INTO table …
  - BEGIN/END TRANSACTION

Hortonworks

Insert, Update and Delete



Base File

Name	Purchase
Anne	Red Fish
Bill	Blue Fish
Christine	Blue Fish
David	Black Fish
Eric	Young Fish

Update 1

Op	Txn Id	RowId	Name	Purchase
I	1	0	Joe	Red Fish
U	0	0	Anne	Star
D	0	4		

Update 2

Op	Txn Id	RowId	Name	Purchase
U	1	0	Joe	Old Fish
U	0	0	Ann	Star
D	0	2		

Logical File

Name	Purchase
Joe	Old Fish
Ann	Star
Bill	Blue Fish
David	Black Fish

HIVE-5317

SQL Compliant security

- **Hive user has access to HDFS files**
- **Manages fine grained access via standard:**
  - Users and roles
  - Privileges (insert, select, update, delete, all)
  - Objects (tables, views)
  - Grant/Revoke/Show statements to administrate
- **Special roles**
  - PUBLIC – all users belong to this role
  - SUPERUSER – privilege to create/drop role, grant access, …
- **Trusted UDFs**
- **Pluggable sources for user to role mapping.**
  - E.g.: HDFS groups

HIVE-784

Extended sub-query support

- **Sub-queries in where/having clause**
  - (NOT) IN/EXISTS
- **Transformation at a high level are:**
  - In/Exists => Left outer join
  - Not In/Exists => Left outer join + null check
  - Correlation converted to “group by” in sub-query
- **Will work only if query can be flattened**
  - No “brute force” of sub-query

Example:

```
select o_orderpriority, count(*)
from orders o
where
  o_orderdate  >= ’2013-01-01’
  and exists  (
          select *
          from lineitem
          where
            l_orderkey = o.o_orderkey
            and l_commitdate < l_receiptdate
          )
group by o_orderpriority
order by o_orderpriority;
```

HIVE-784

Alternate join syntax

- **Allows for ‘comma separated’ join syntax**
  - Easier to use
  - Facilitates integration with tools
- **Important aspect is correct push down of predicates**
  - Cross product is very expensive
  - Conditions need to be pushed as close to the table source as possible

Example:

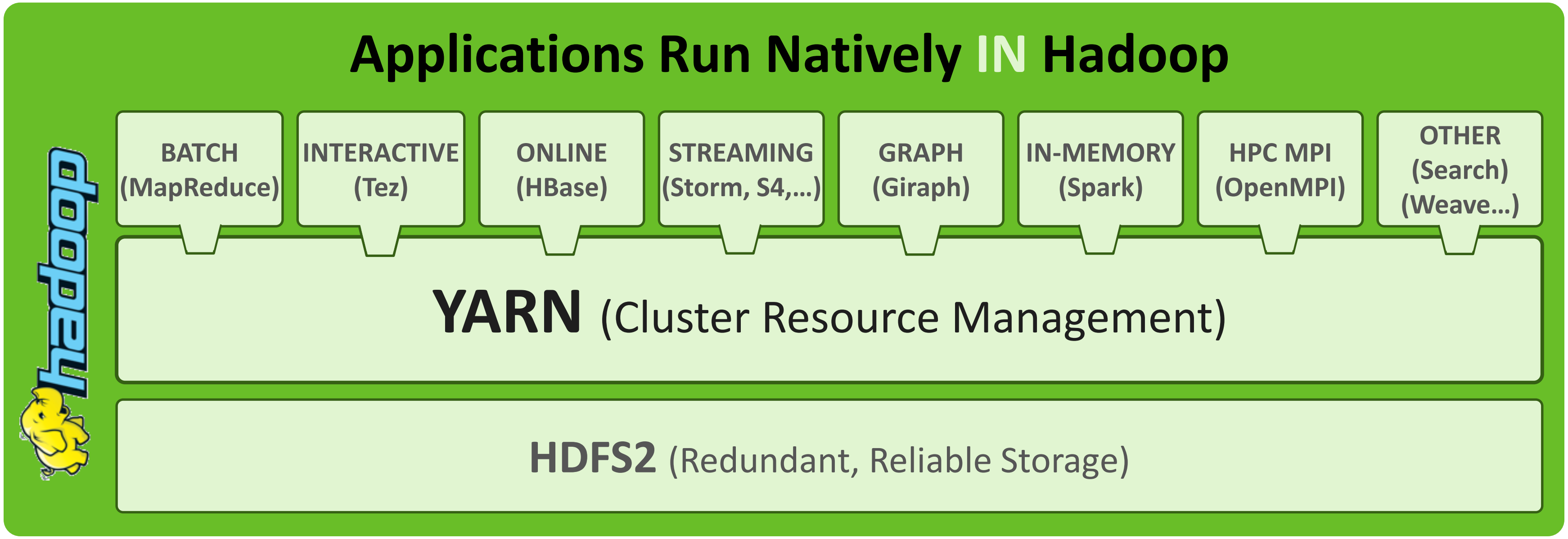
```
select o_orderpriority, count(*)
from orders, lineitem
where
  o_orderdate  >= ’2013-01-01’
  and l_orderkey = o_orderkey
  and l_commitdate < l_receiptdate
group by o_orderpriority
order by o_orderpriority;
```

YARN: Taking Hadoop Beyond Batch

Store ALL DATA in one place…

Interact with that data in MULTIPLE WAYS

with Predictable Performance and Quality of Service



Applications Run Natively IN Hadoop

BATCH (MapReduce)

INTERACTIVE (Tez)

ONLINE (HBase)

STREAMING (Storm, S4,…)

GRAPH (Giraph)

IN-MEMORY (Spark)

HPC MPI (OpenMPI)

OTHER (Search) (Weave…)

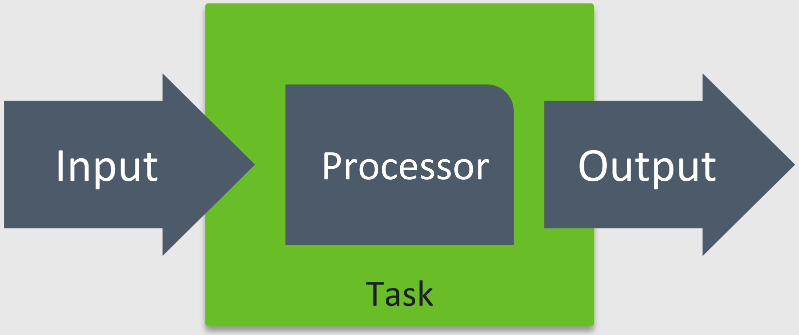
YARN (Cluster Resource Management)

HDFS2 (Redundant, Reliable Storage)

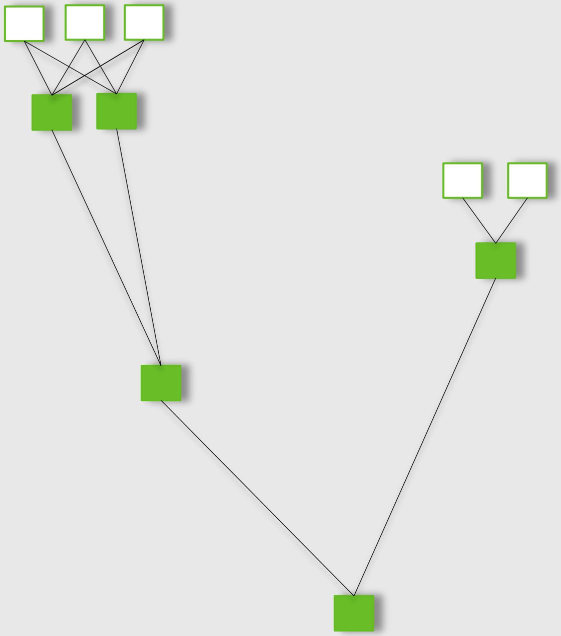
Apache Tez ("Speed")

- **Replaces MapReduce as primitive for Pig, Hive, Cascading etc.**
  - Smaller latency for interactive queries
  - Higher throughput for batch queries
  - 22 contributors: Hortonworks (13), Facebook, Twitter, Yahoo, Microsoft

Task with pluggable *Input, Processor and Output*



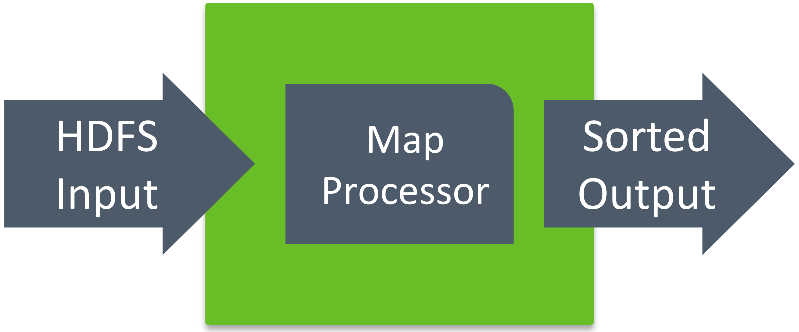
Tez Task - <*Input, Processor, Output*>



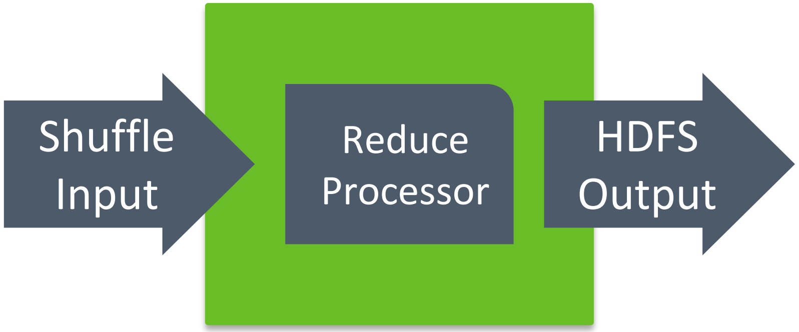
YARN ApplicationMaster to run DAG of Tez Tasks

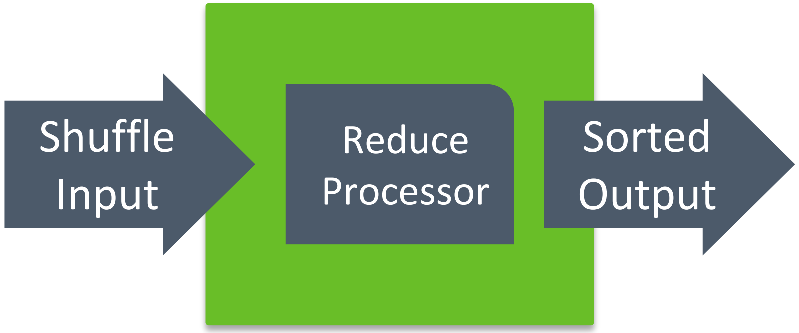
Tez: Building blocks for scalable data processing

Classical ‘Map’



Classical ‘Reduce’





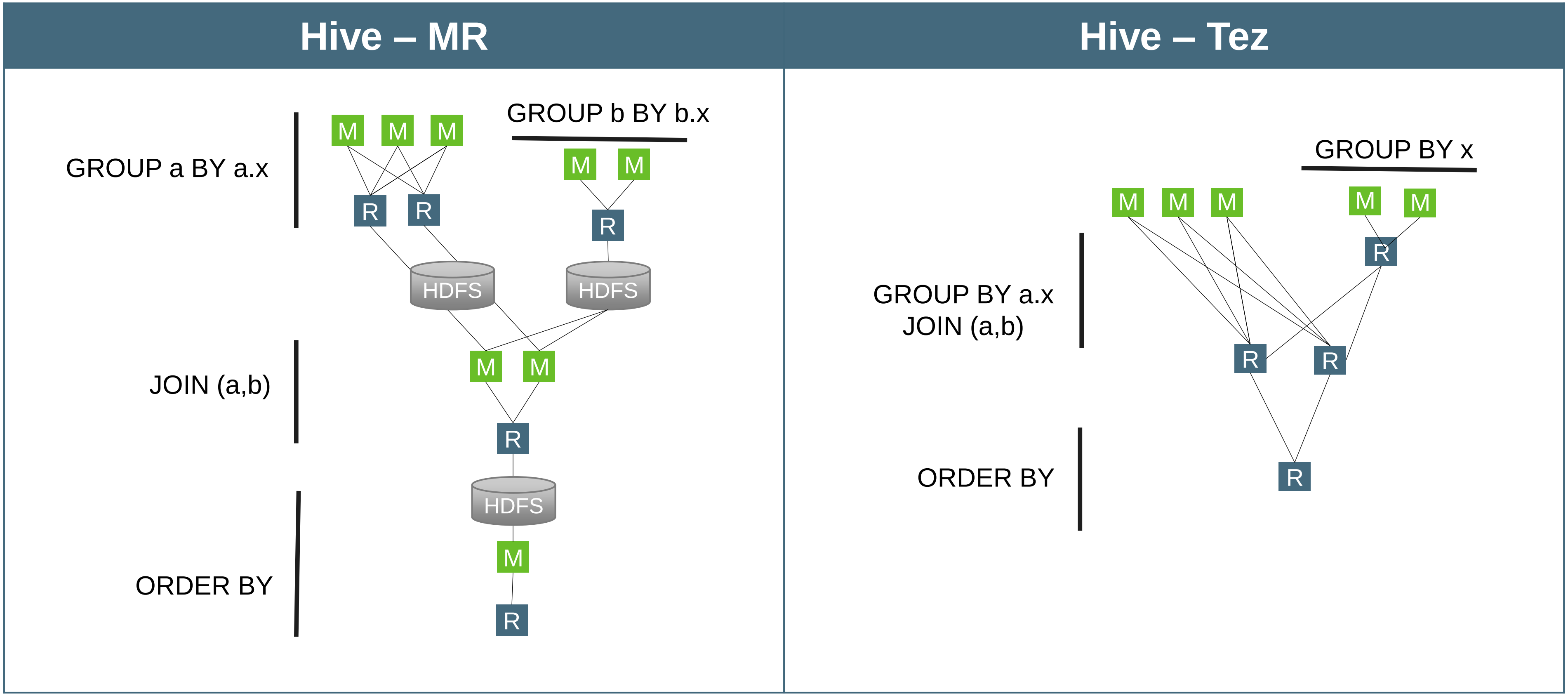
Intermediate ‘Reduce’ for Map-Reduce-Reduce

HIVE-4660

Hive-on-MR vs. Hive-on-Tez

```
SELECT g1.x, g1.avg, g2.cnt
FROM (SELECT a.x, AVERAGE(a.y) AS avg FROM a GROUP BY a.x) g1
JOIN (SELECT b.x, COUNT(b.y) AS avg FROM b GROUP BY b.x) g2
ON (g1.x = g2.x)
ORDER BY avg;
```

Tez avoids unnecessary writes to HDFS



Tez Sessions

… because Map/Reduce query startup is expensive

- **Tez Sessions**
  - Hot containers ready for immediate use
  - Removes task and job launch overhead (~5s – 30s)
- **Hive**
  - Session launch/shutdown in background (seamless, user not aware)
  - Submits query plan directly to Tez Session

Native Hadoop service, not ad-hoc

Tez Delivers Interactive Query - Out of the Box!

Feature	Description	Benefit
Tez Session	Overcomes Map-Reduce job-launch latency by pre-launching Tez AppMaster	Latency
Tez Container Pre-Launch	Overcomes Map-Reduce latency by pre-launching hot containers ready to serve queries.	Latency
Tez Container Re-Use	Finished maps and reduces pick up more work rather than exiting. Reduces latency and eliminates difficult split-size tuning. **Out of box performance!**	Latency
Runtime re-configuration of DAG	Runtime query tuning by picking aggregation parallelism using online query statistics	Throughput
Tez In-Memory Cache	Hot data kept in RAM for fast access.	Latency
Complex DAGs	Tez Broadcast Edge and Map-Reduce-Reduce pattern improve query scale and throughput.	Throughput

HIVE-3874

ORC File Format

- **Columnar format for complex data types**
- **Built into Hive from 0.11**
- **Support for Pig and MapReduce via HCatalog**
- **Two levels of compression**
  - Lightweight type-specific and generic
- **Built in indexes**
  - Every 10,000 rows with position information
  - Min, Max, Sum, Count of each column
  - Supports seek to row number

ORC File Format

- **Hive 0.12**
  - –Predicate Push Down
  - –Improved run length encoding
  - –Adaptive string dictionaries
  - –Padding stripes to HDFS block boundaries
- **Trunk**
  - –Stripe-based Input Splits
  - –Input Split elimination
  - –Vectorized Reader
  - –Customized Pig Load and Store functions

HIVE-4160

Vectorized Query Execution

- **Designed for Modern Processor Architectures**
  - –Avoid branching in the inner loop.
  - –Make the most use of L1 and L2 cache.
- **How It Works**
  - –Process records in batches of 1,000 rows
  - –Generate code from templates to minimize branching.
- **What It Gives**
  - –30x improvement in rows processed per second.
  - –Initial prototype: 100M rows/sec on laptop

HDFS Buffer Cache

- **Use memory mapped buffers for zero copy**
  - –Avoid overhead of going through DataNode
  - –Can mlock the block files into RAM
- **ORC Reader enhanced for *zero-copy* reads**
  - –New compression interfaces in Hadoop
- **Vectorization specific reader**
  - –Read 1000 rows at a time
  - –Read into Hive’s internal representation

HIVE-5775

Cost-based optimization (Optiq)

- **Optiq: Open source, Apache licensed query execution framework in Java**
  - – Used by Apache Drill, Apache Cascade, Lucene DB, …
  - – Based on Volcano paper
  - – 20 man years dev, more than 50 optimization rules

- **Goals for hive**
  - – Ease of Use – no manual tuning for queries, make choices automatically based on cost
  - – View Chaining/Ad hoc queries involving multiple views
  - – Help enable BI Tools front-ending Hive
  - – Emphasis on latency reduction

- **Cost computation will be used for**
  - ➢ Join ordering
  - ➢ Join algorithm selection
  - ➢ Tez vertex boundary selection

How Stinger Phase 3 Delivers Interactive Query

Feature	Description	Benefit
Tez Integration	Tez is significantly better engine than MapReduce	Latency
Vectorized Query	Take advantage of modern hardware by processing thousand-row blocks rather than row-at-a-time.	Throughput
Query Planner	Using extensive statistics now available in Metastore to better plan and optimize query, including predicate pushdown during compilation to eliminate portions of input (beyond partition pruning)	Latency
ORC File	Columnar, type aware format with indices	Latency
Cost Based Optimizer (Optiq)	Join re-ordering and other optimizations based on column statistics including histograms etc. (future)	Latency

SCALE: Interactive Query at Petabyte Scale

Sustained Query Times

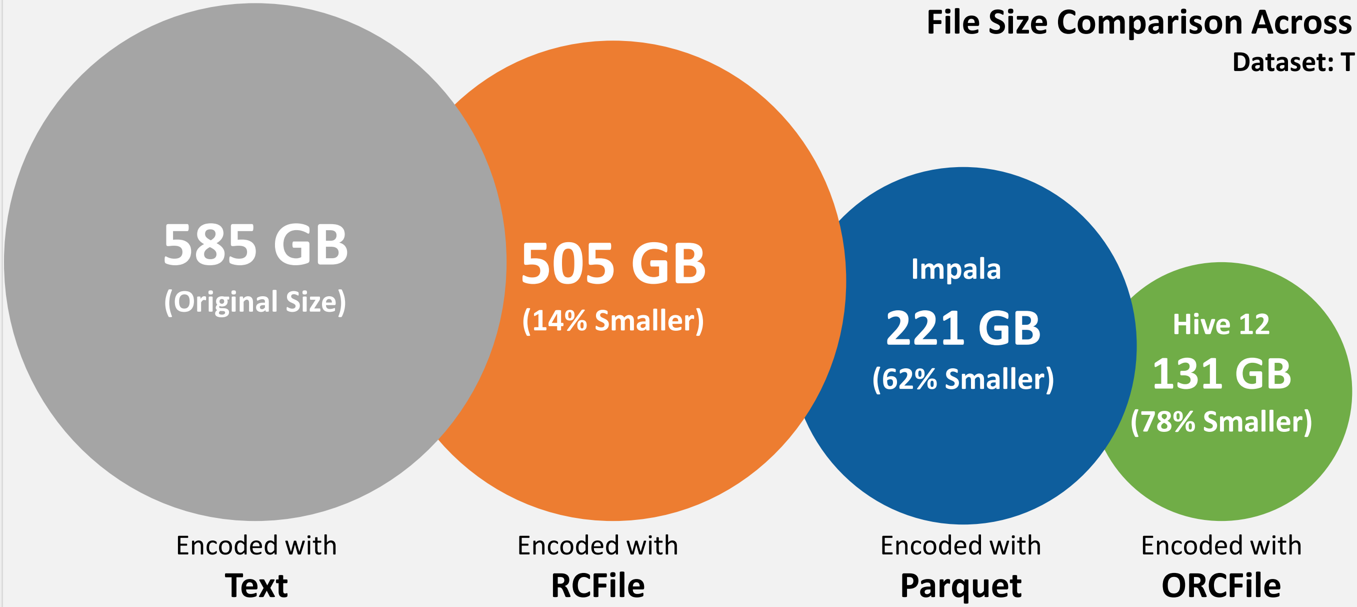
Apache Hive 0.12 provides **sustained** acceptable query times even at petabyte scale

Smaller Footprint

Better encoding with ORC in Apache Hive 0.12 reduces resource requirements for your cluster

File Size Comparison Across Encoding Methods

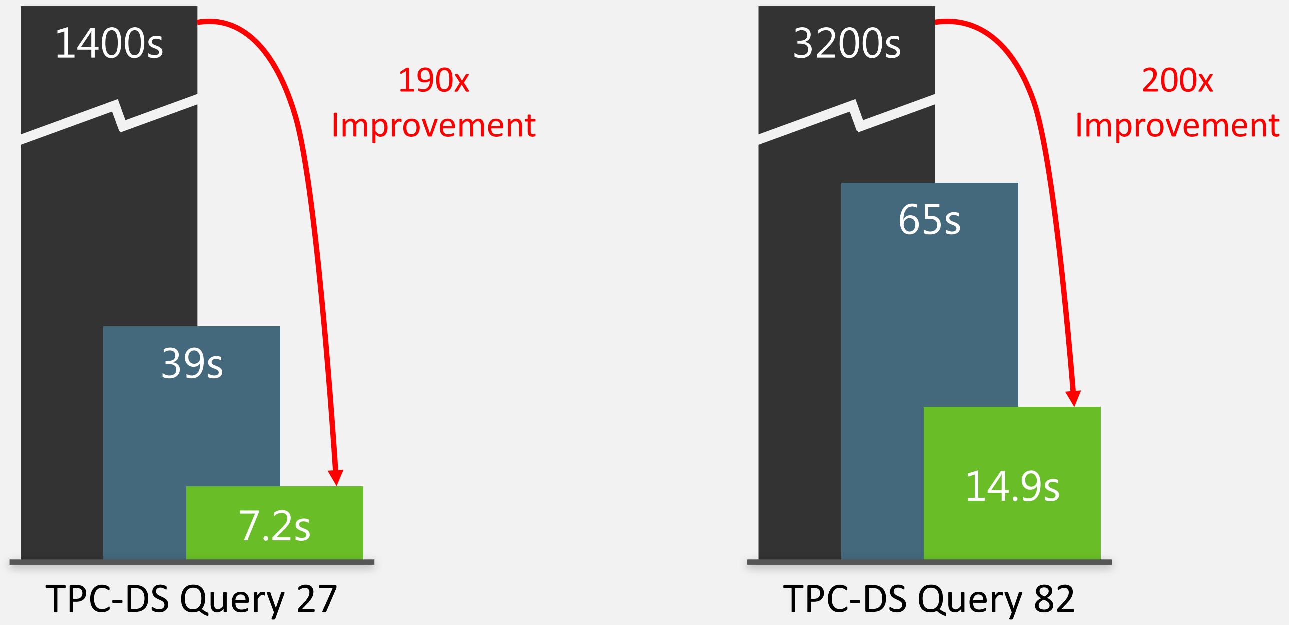
Dataset: TPC-DS Scale 500 Dataset



- Larger Block Sizes
- Columnar format arranges columns adjacent within the file for compression & fast access

Stinger Phase 3: Interactive Query In Hadoop

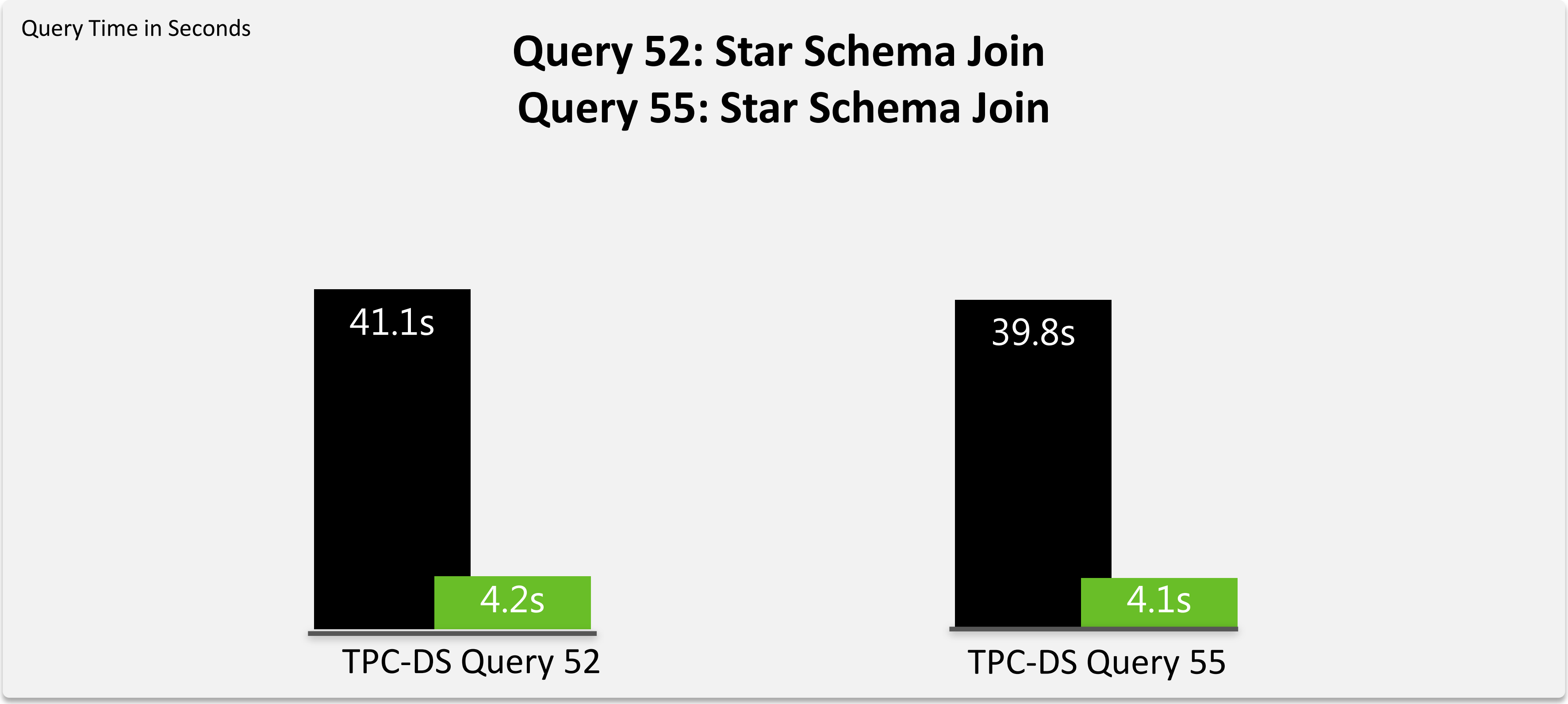
Query 27: Pricing Analytics using Star Schema Join
Query 82: Inventory Analytics Joining 2 Large Fact Tables



⬛ Hive 10 🟦 Hive 0.11 (Phase 1) 🟩 Trunk (Phase 3)

All Results at Scale Factor 200 (Approximately 200GB Data)

Speed: Delivering Interactive Query



Query 52: Star Schema Join
Query 55: Star Schema Join

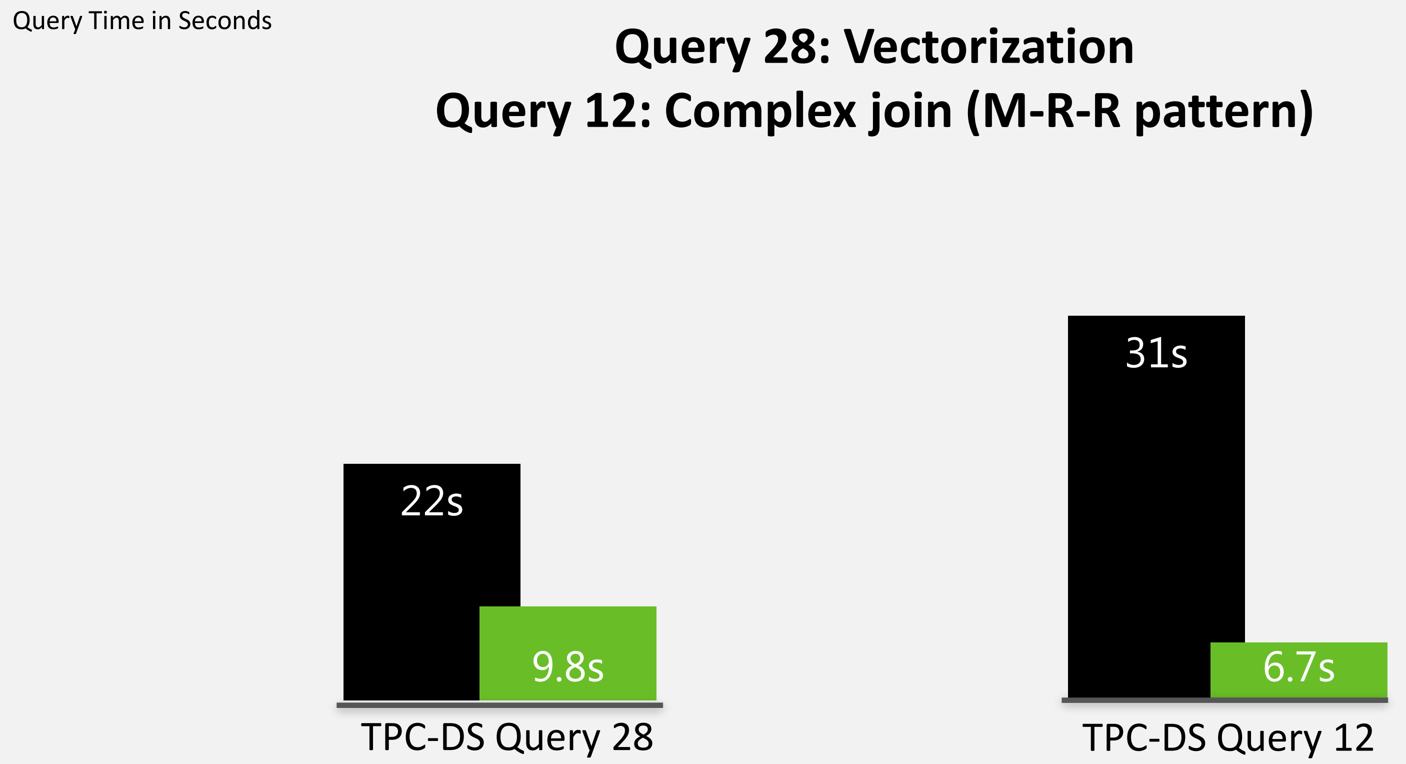
■ Hive 0.12

🟩 Trunk (Phase 3)

Test Cluster:

- 200 GB Data (ORCFile)
- 20 Nodes, 24GB RAM each, 6x disk each

Speed: Delivering Interactive Query



Query 28: Vectorization

Query 12: Complex join (M-R-R pattern)

■ Hive 0.12

🟩 Trunk (Phase 3)

Test Cluster:

- 200 GB Data (ORCFile)
- 20 Nodes, 24GB RAM each, 6x disk each

Next Steps

- **Blog**

  http://hortonworks.com/blog/delivering-on-stinger-a-phase-3-progress-update/

- **Stinger Initiative**

  http://hortonworks.com/labs/stinger/

- **Stinger Beta: HDP-2.1 *Beta*, December, 2013**

Thank You!

gunther@apache.org
@yakrobat
@hortonworks