Cost-based query optimization in Apache Hive 0.14

Julian Hyde

Seattle

September 24th, 2014

Hortonworks

About me

Julian Hyde

Architect at Hortonworks

Open source:

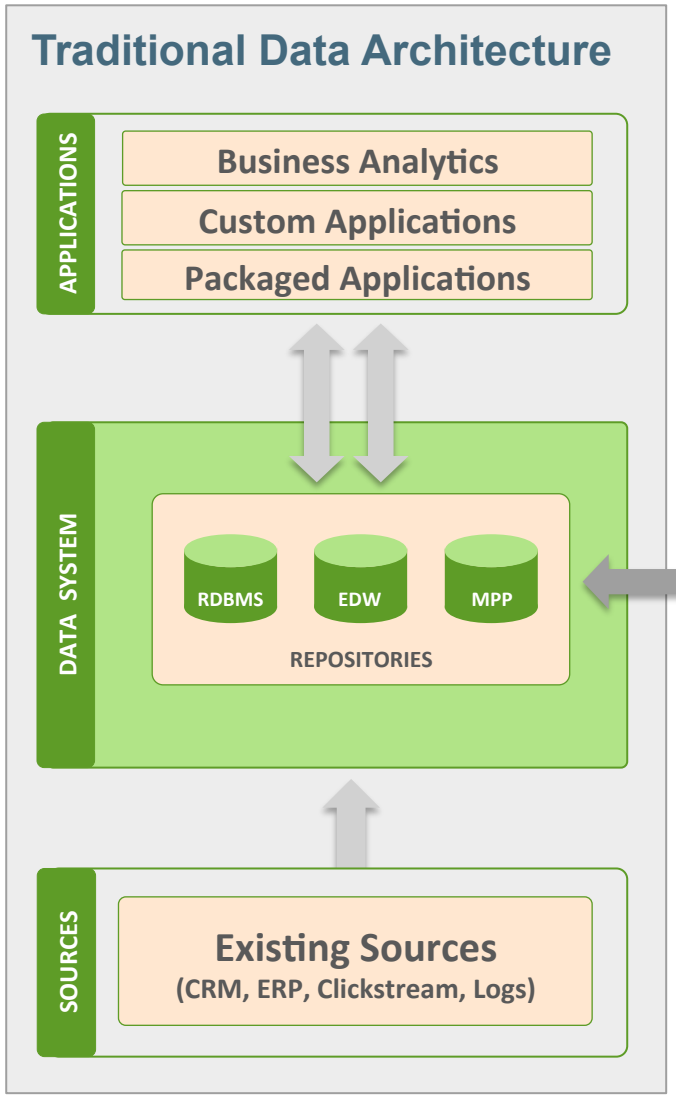
- Founder & lead, Apache Optiq (query optimization framework)
- Founder & lead, Pentaho Mondrian (analysis engine)
- Committer, Apache Drill
- Contributor, Apache Hive
- Contributor, Cascading Lingual (SQL interface to Cascading)

Past:

- SQLstream (streaming SQL)
- Broadbase (data warehouse)
- Oracle (SQL kernel development)

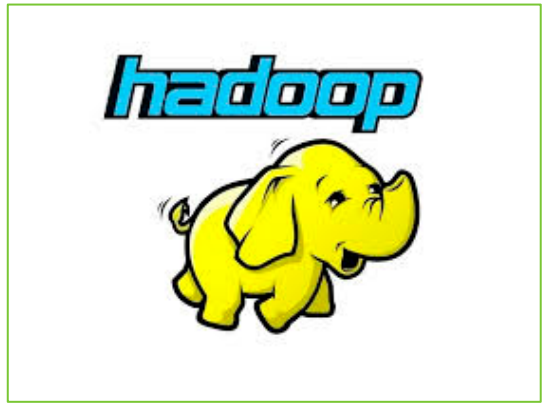
Hortonworks

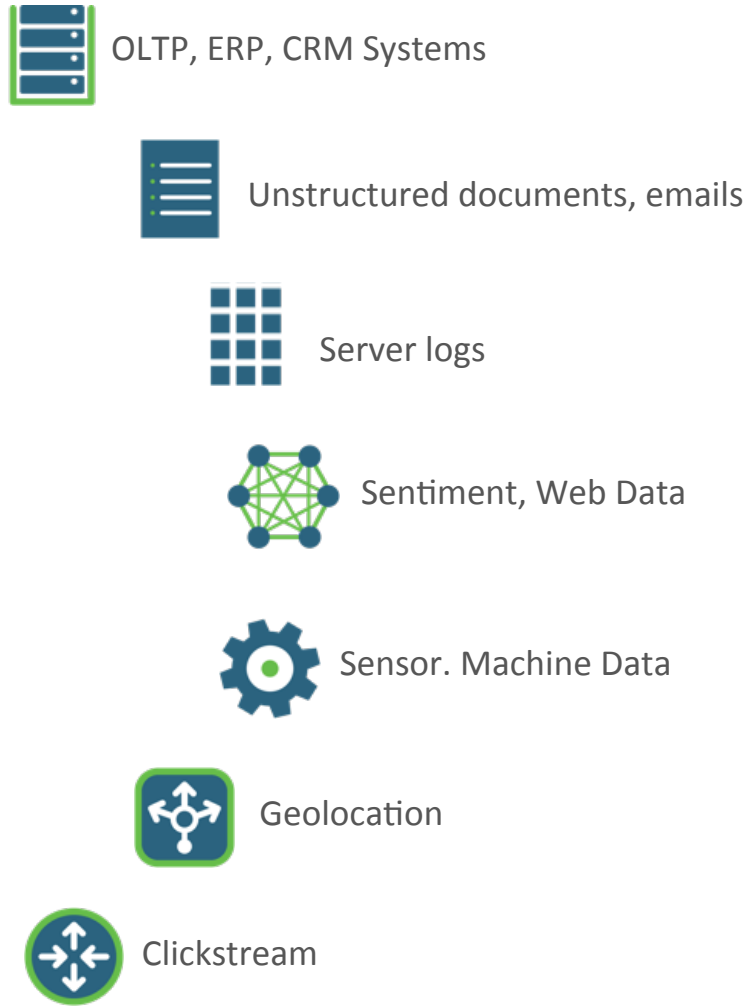
Hadoop - A New Data Architecture for New Data



New Data Requirements:

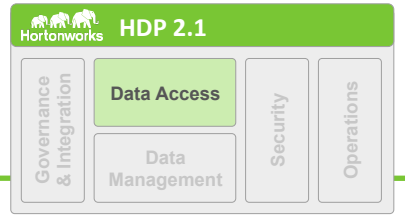
- Scale
- Economics
- Flexibility





Hortonworks

Interactive SQL-IN-Hadoop Delivered



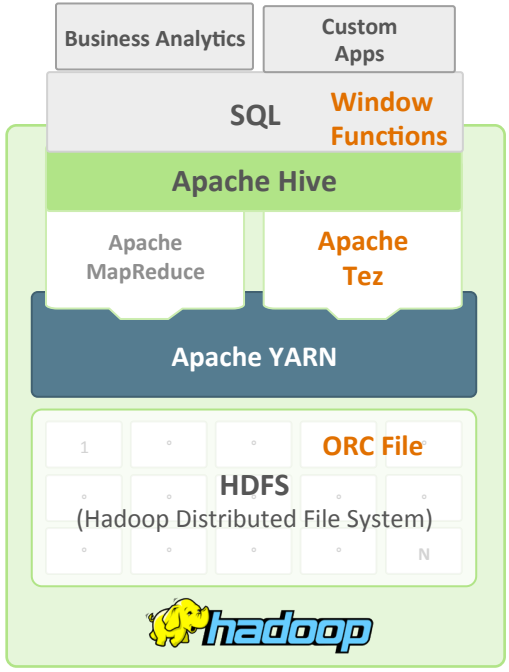
Stinger Initiative – DELIVERED

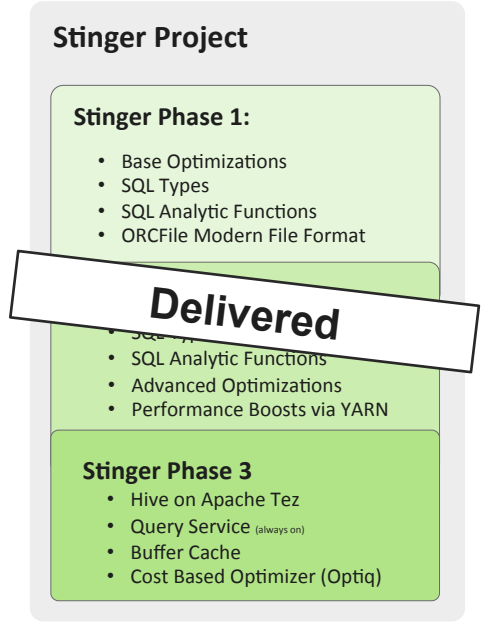
Next generation SQL based interactive query in Hadoop

Speed
Improve Hive query performance has increased 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



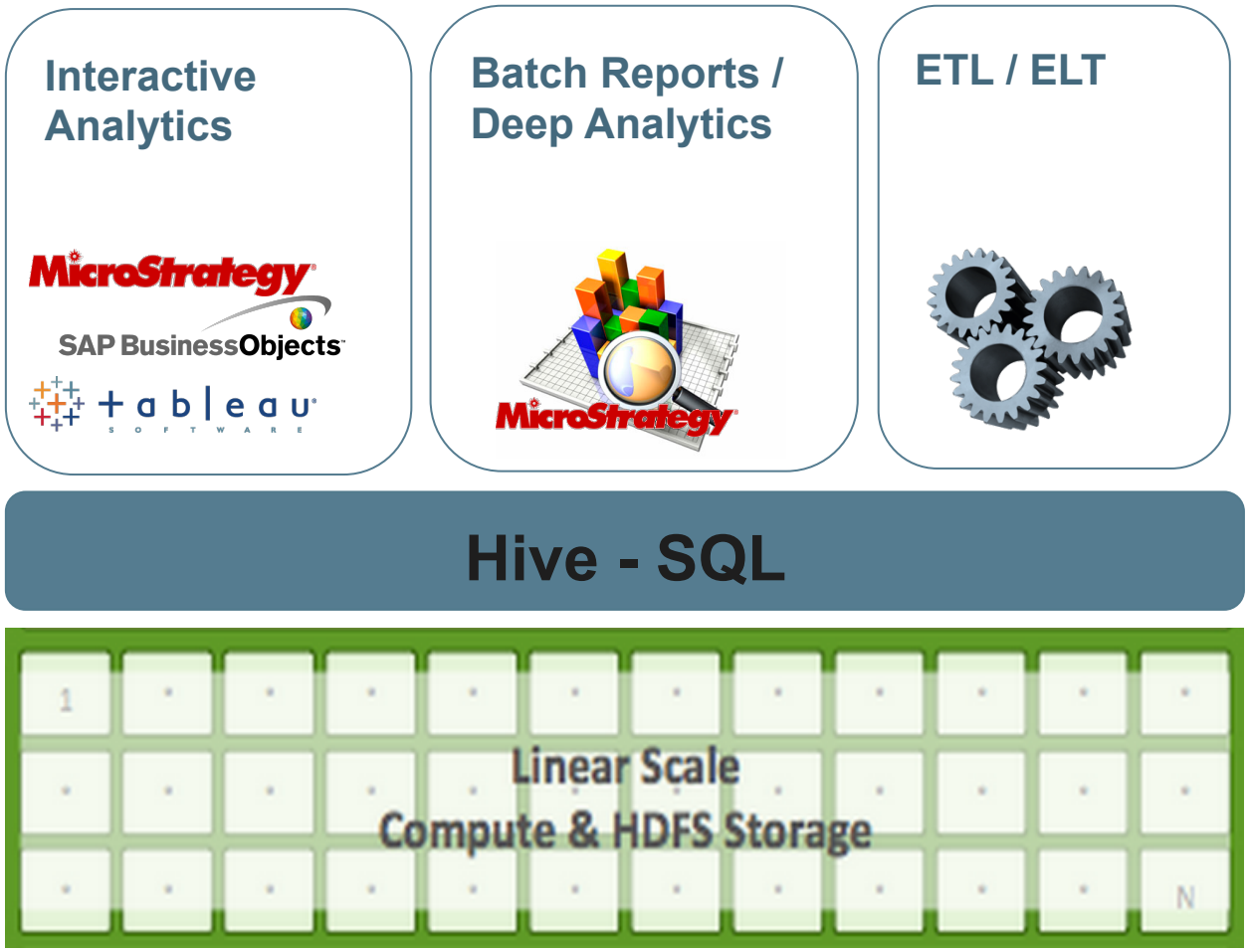


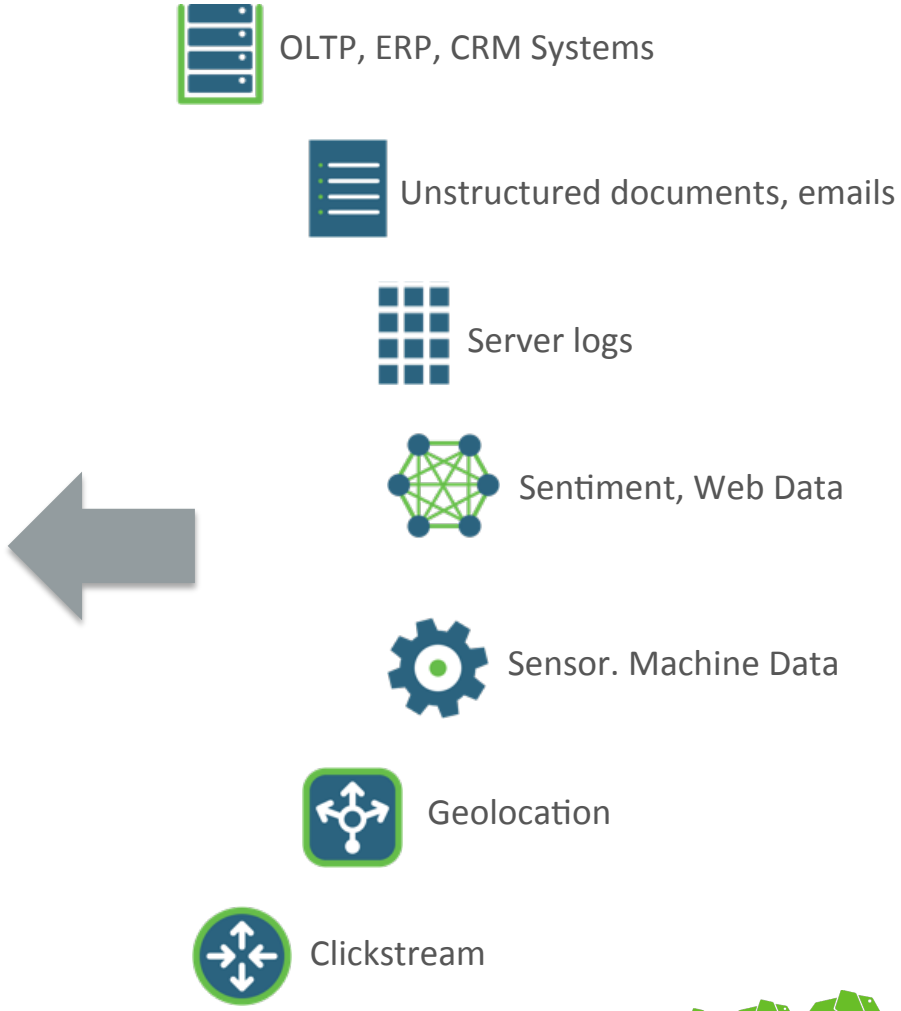
Apache Hive Contribution… an Open Community at its finest

1,672	145	44	~390,000	13
Jira Tickets Closed	Developers	Companies	Lines Of Code Added… (2x)	Months

Hortonworks

Hive – Single tool for all SQL use cases





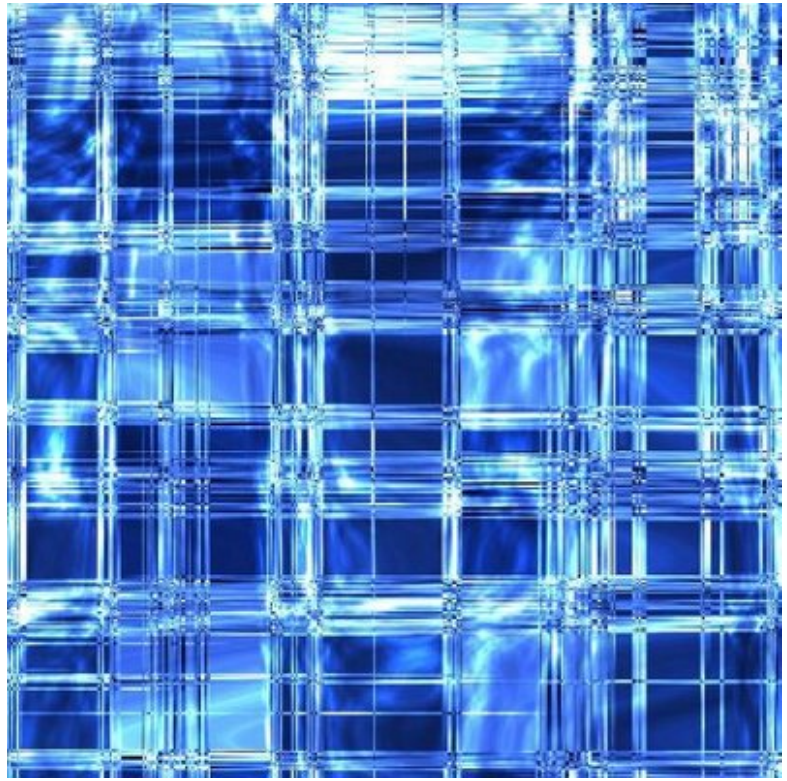
Hortonworks

Incremental cutover to cost-based optimization

Release	Date	Remarks
Apache Hive 0.12	October 2013	• Rule-based Optimizations • No join reordering • Main optimizations: predicate push-down & partition pruning • Semantic info and operator tree tightly coupled
Apache Hive 0.13	April 2014	"Old-style" JOIN & push-down conditions: *… FROM t1, t2 WHERE …*
HDP 2.1	April 2014	Cost-based ordering of joins
Apache Hive 0.14	soon	Bushy joins, large joins, better operator coverage, better statistics, …

Hortonworks

Apache Optiq (incubating)



Hortonworks

Apache Optiq

Apache incubator project since May, 2014

Query planning framework

- Relational algebra, rewrite rules, cost model
- Extensible
- Usable standalone (JDBC) or embedded

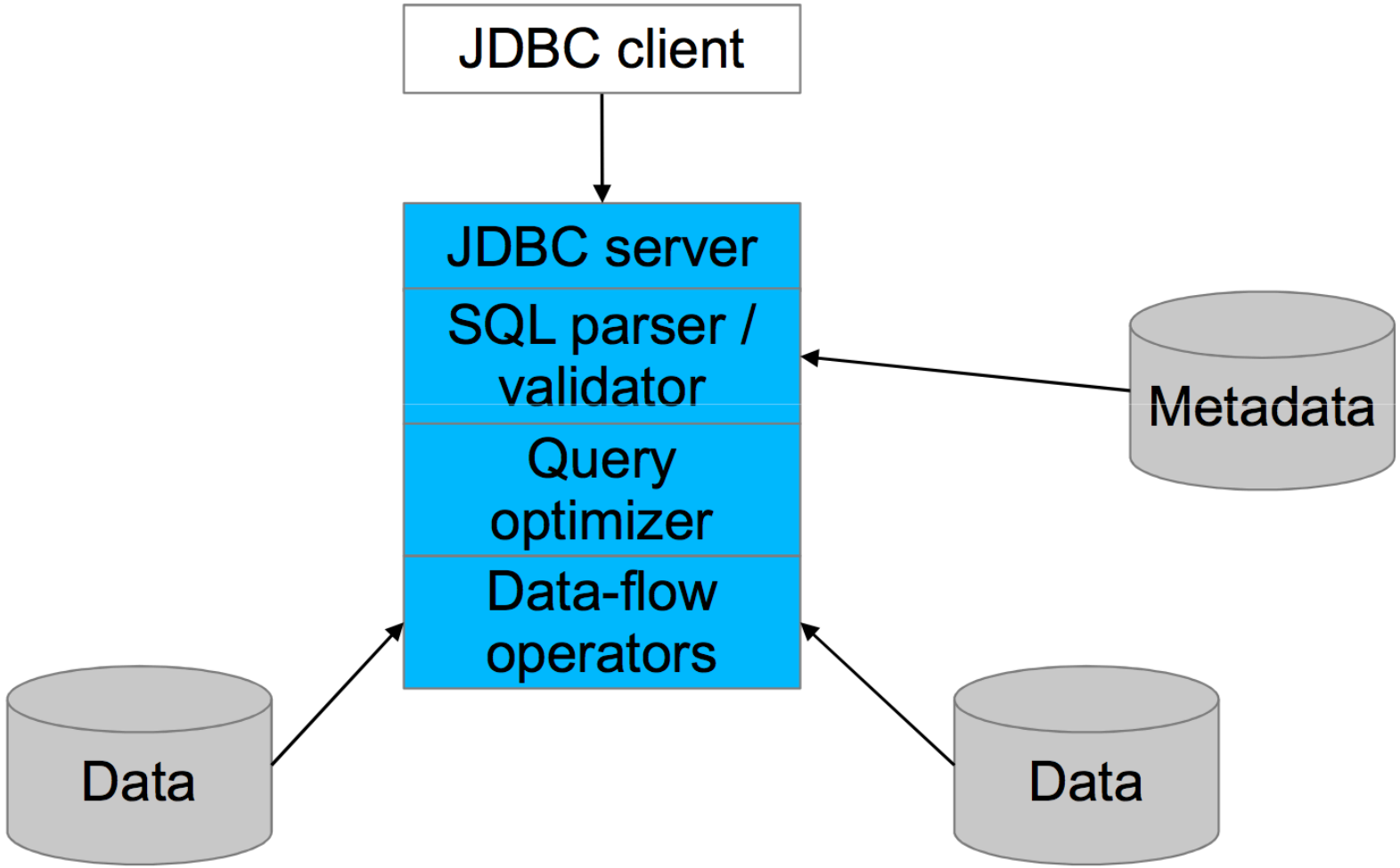
Adoption

- Lingual – SQL interface to Cascading
- Apache Drill
- Apache Hive

Adapters: Splunk, Spark, MongoDB, JDBC, CSV, JSON, Web tables, In-memory data

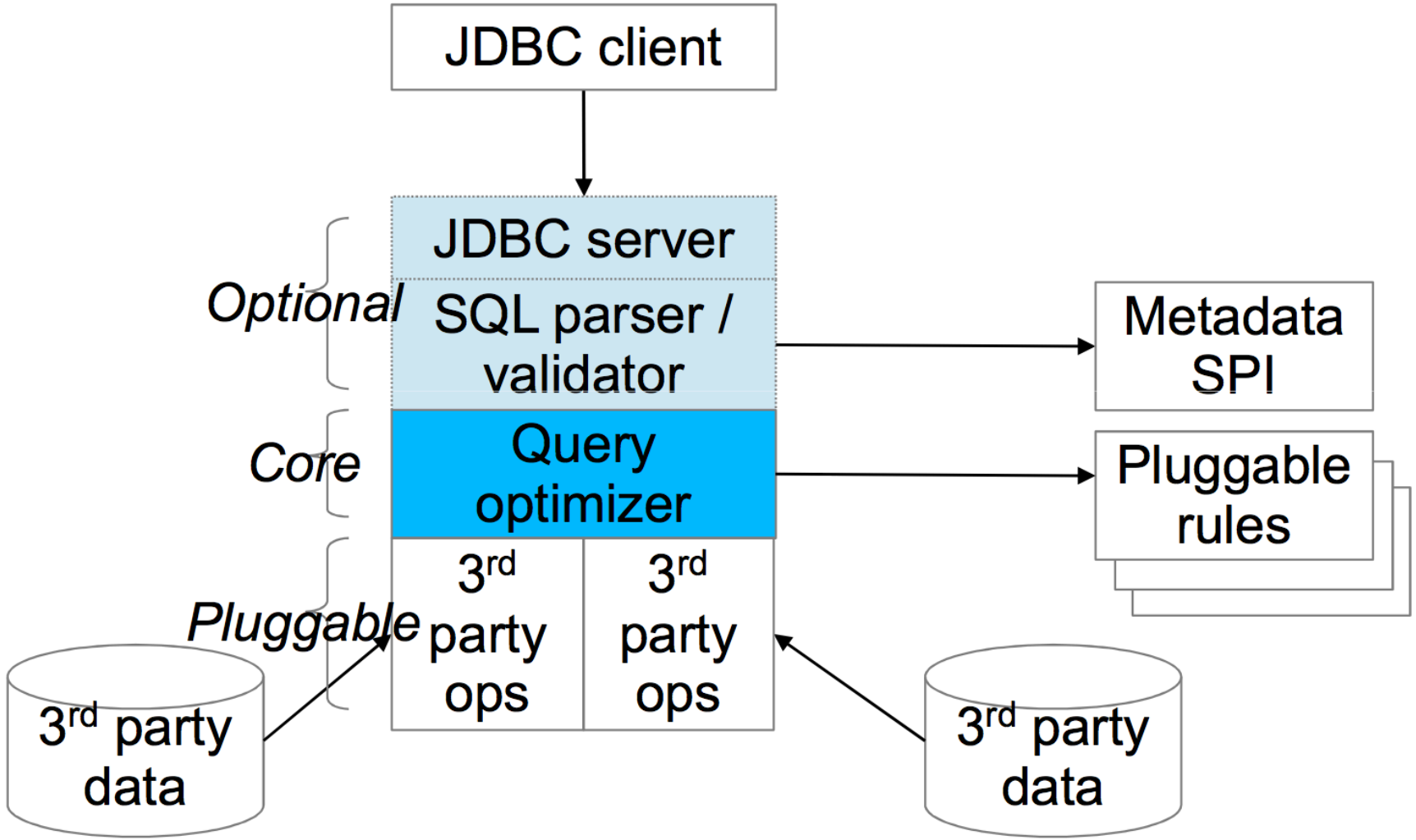
Hortonworks

Conventional DB architecture



Hortonworks

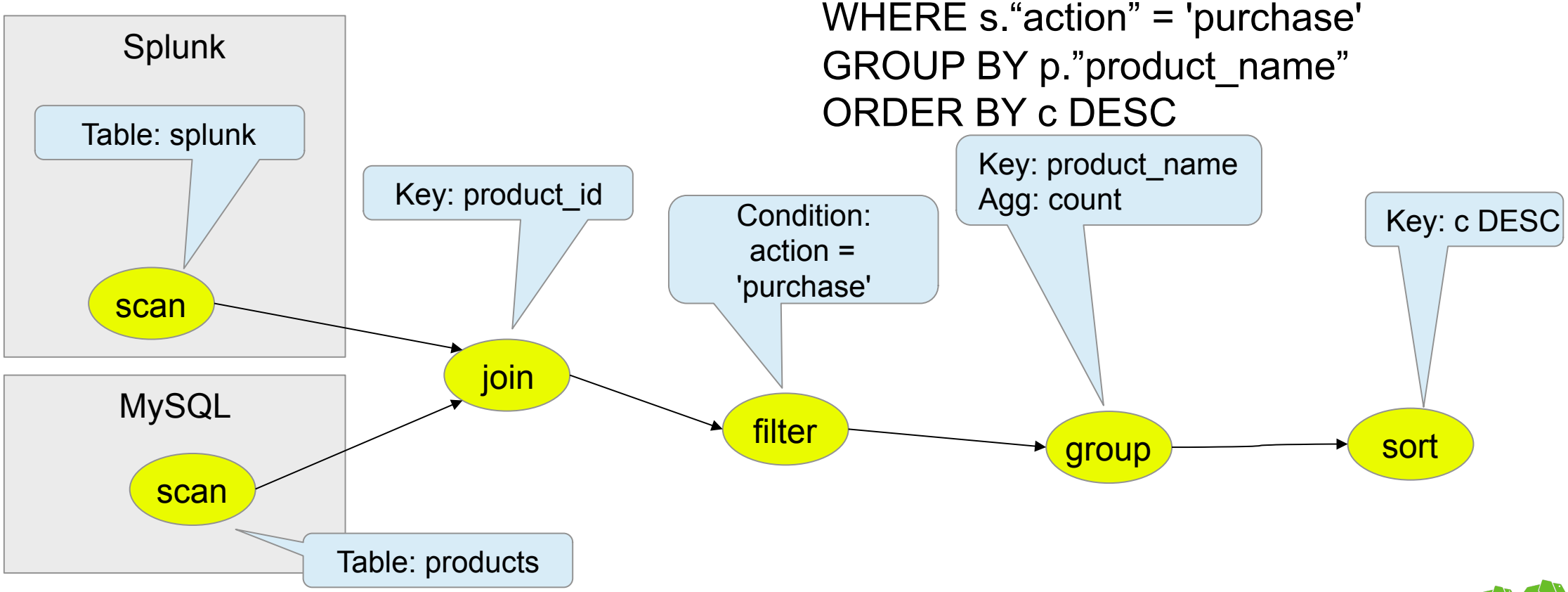
Optiq architecture



Hortonworks

Expression tree

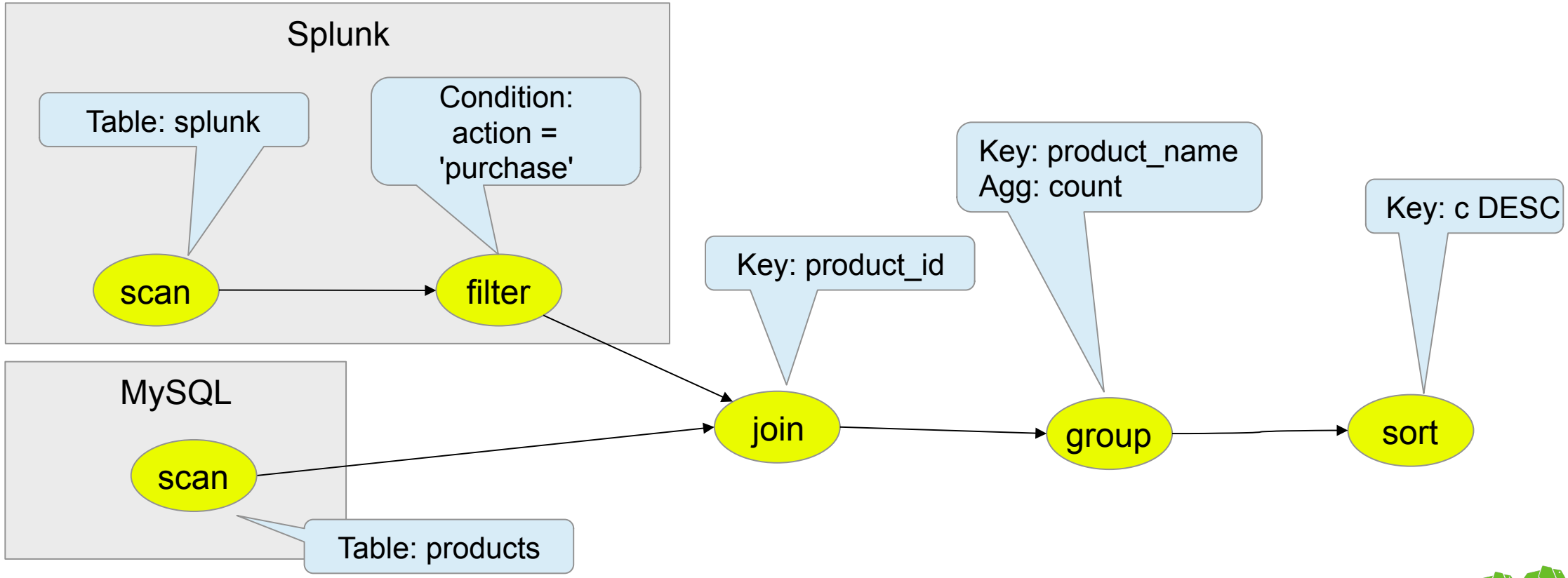
```
SELECT p.“product_name”, COUNT(*) AS c
FROM “splunk”.”splunk” AS s
    JOIN “mysql”.”products” AS p
    ON s.”product_id” = p.”product_id”
WHERE s.“action” = 'purchase'
GROUP BY p.”product_name”
ORDER BY c DESC
```



Hortonworks

Expression tree (optimized)

```
SELECT p.“product_name”, COUNT(*) AS c
FROM “splunk”.”splunk” AS s
    JOIN “mysql”.”products” AS p
    ON s.”product_id” = p.”product_id”
WHERE s.“action” = 'purchase'
GROUP BY p.”product_name”
ORDER BY c DESC
```



Optiq – APIs and SPIs

Relational algebra

RelNode (operator)
- TableScan
- Filter
- Project
- Union
- Aggregate
- …

RelDataType (type)
RexNode (expression)
RelTrait (physical property)
- RelConvention (calling-convention)
- RelCollation (sortedness)
- TBD (bucketedness/distribution)

Transformation rules

RelOptRule
- MergeFilterRule
- PushAggregateThroughUnionRule
- RemoveCorrelationForScalarProjectRule
- 100+ more

Unification (materialized view)
Column trimming

Cost, statistics

RelOptCost
RelOptCostFactory
RelMetadataProvider
- RelMdColumnUniquensss
- RelMdDistinctRowCount
- RelMdSelectivity

SQL parser

SqlNode
SqlParser
SqlValidator

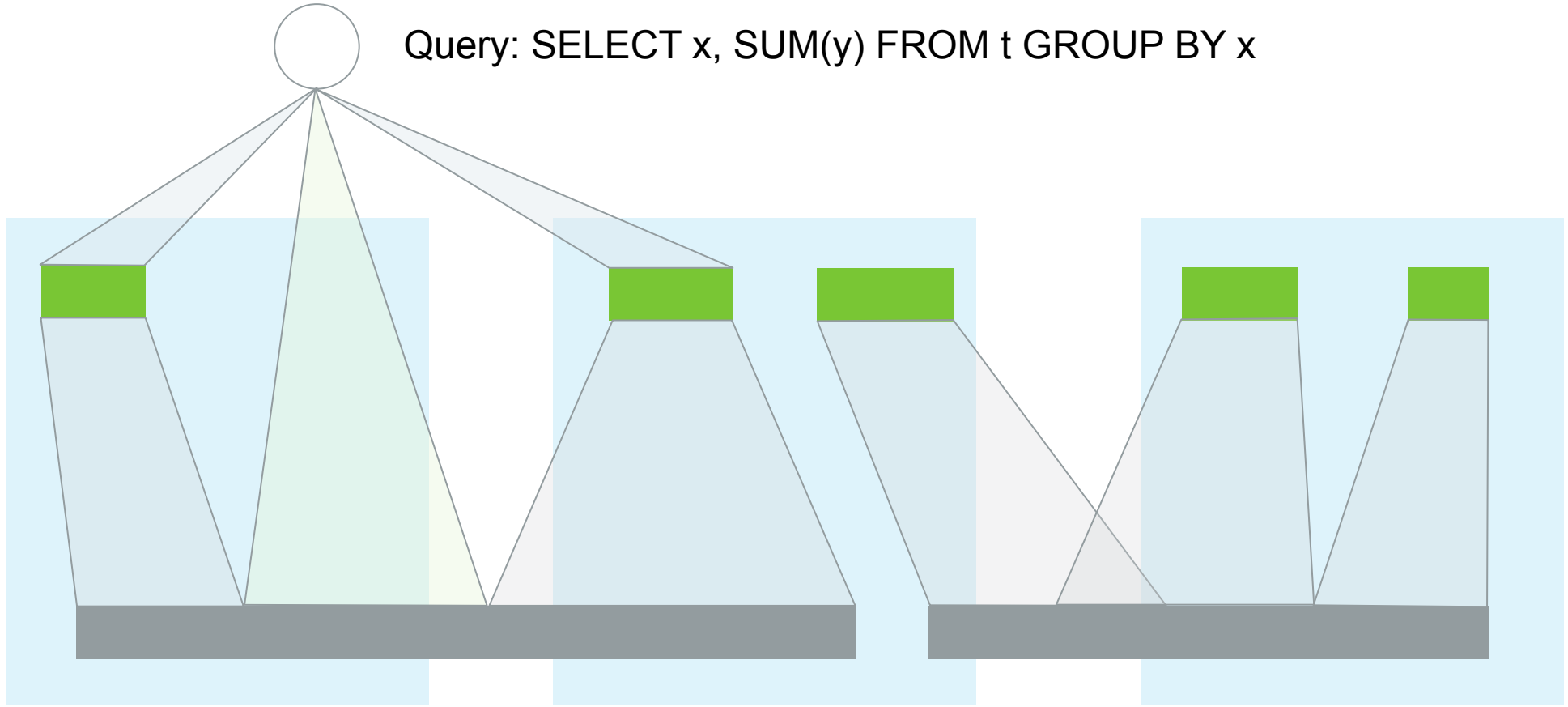
Metadata

Schema
Table
Function
- TableFunction
- TableMacro

JDBC driver

Hortonworks

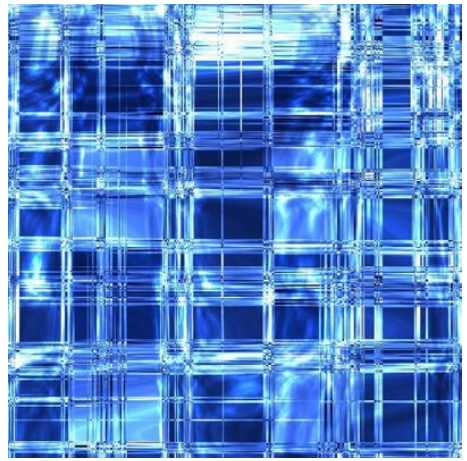
Under development in Optiq - materialized views

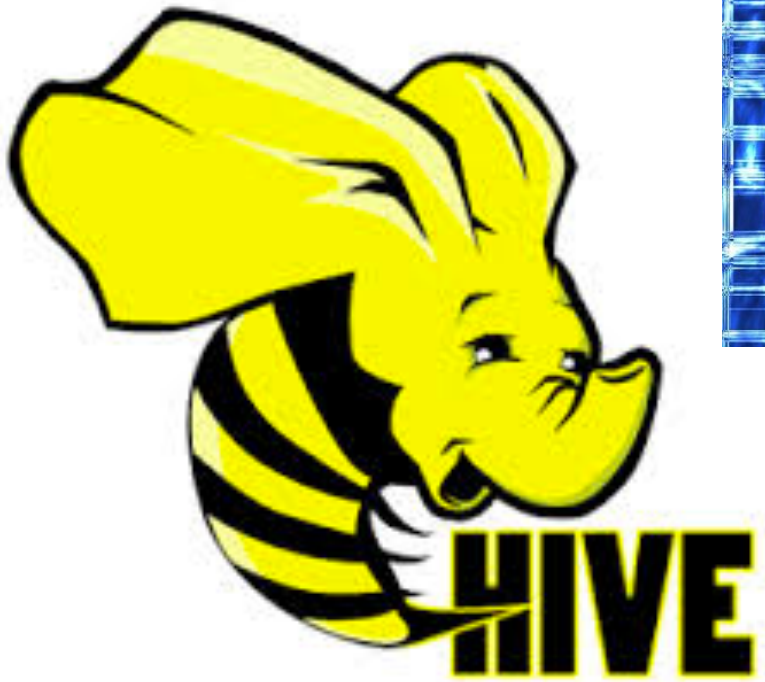


http://hortonworks.com/blog/dmmq/

Hortonworks

Now… back to Hive





Hortonworks

CBO in Hive

Why cost-based optimization?

Ease of Use – Join Reordering

View Chaining

Ad hoc queries involving multiple views

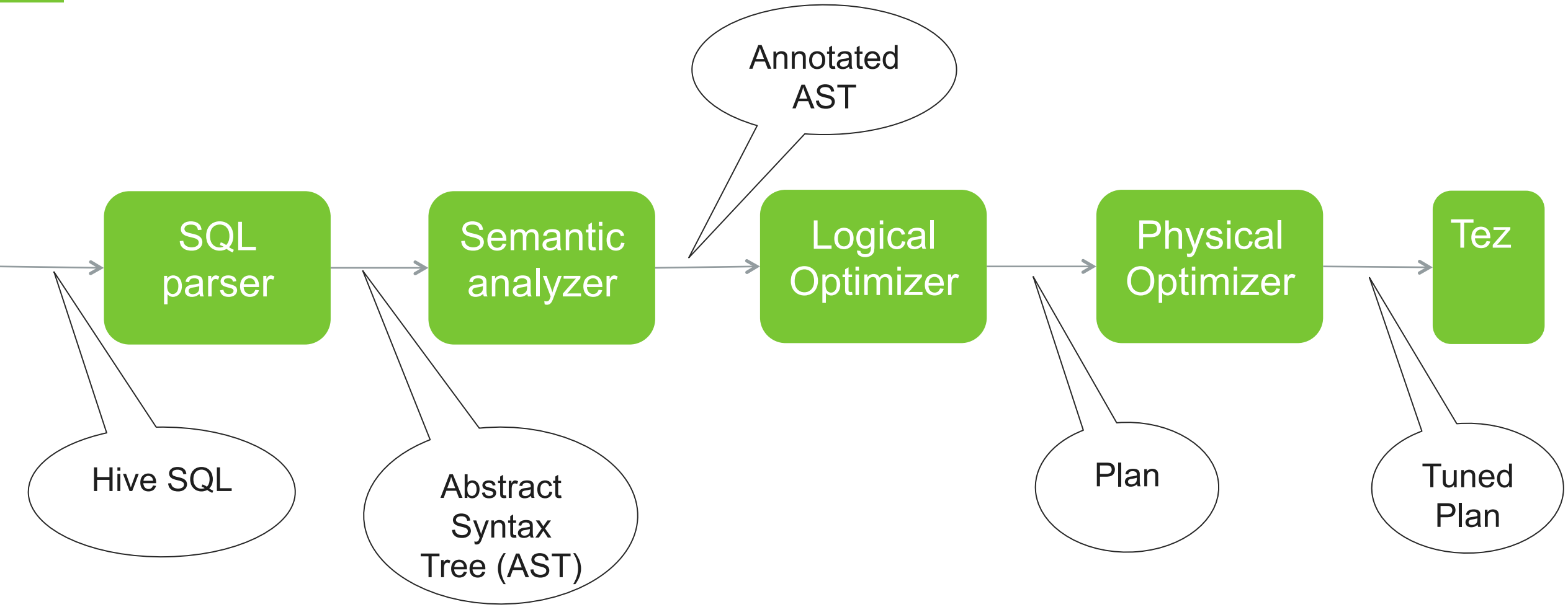
Enables BI Tools as front ends to Hive

More efficient & maintainable query preparation process

Laying the groundwork for deeper optimizations, e.g. materialized views

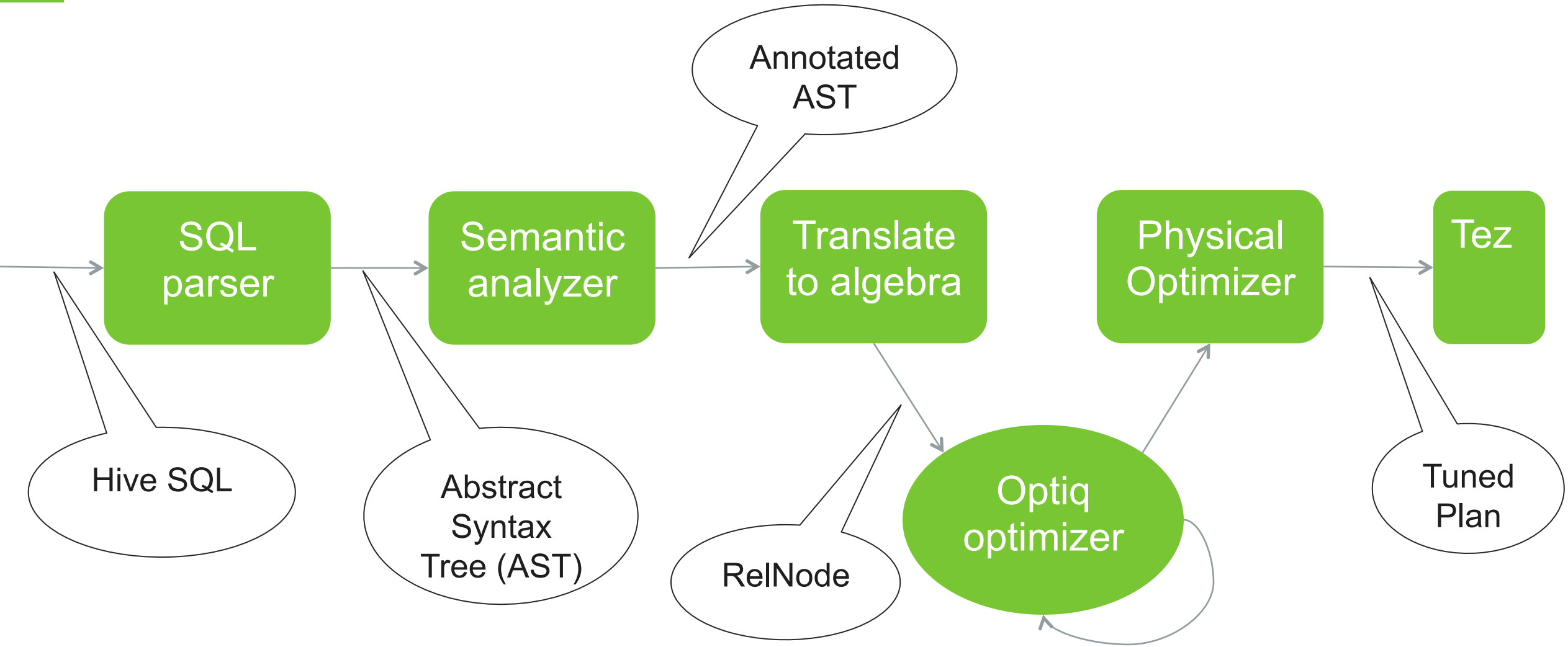
Page 16

Query preparation – Hive 0.13



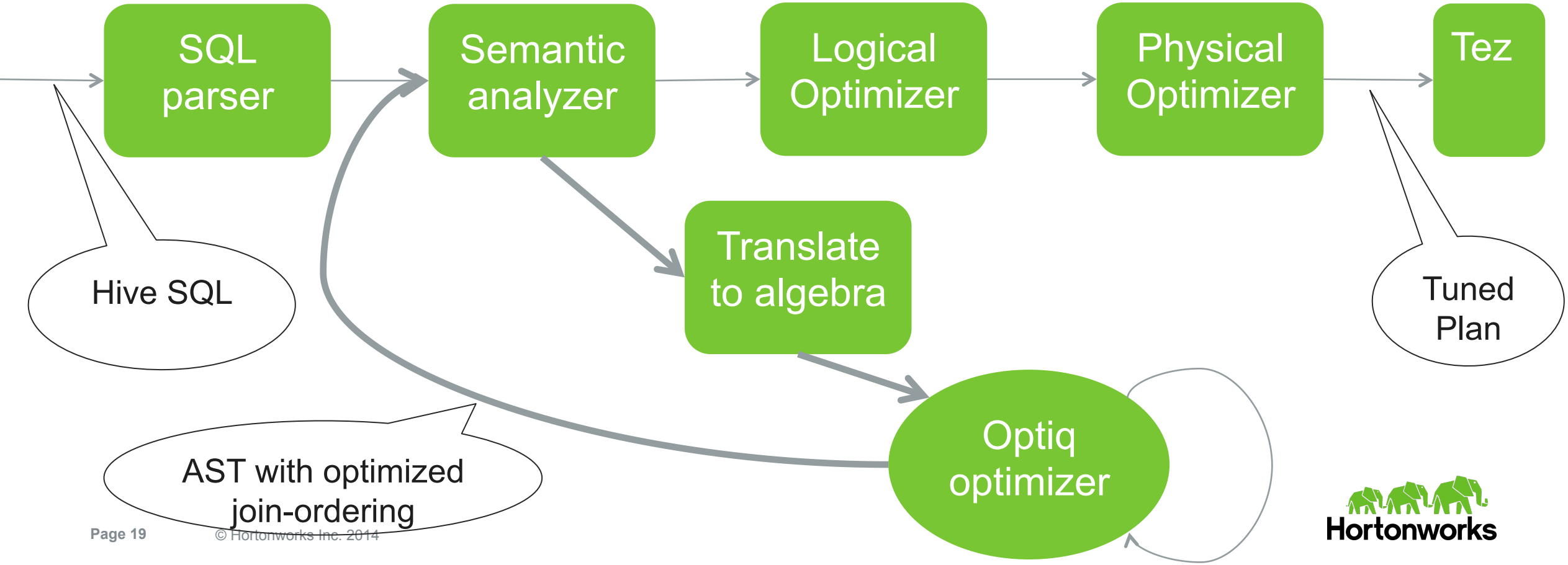
Hortonworks

Query preparation – full CBO



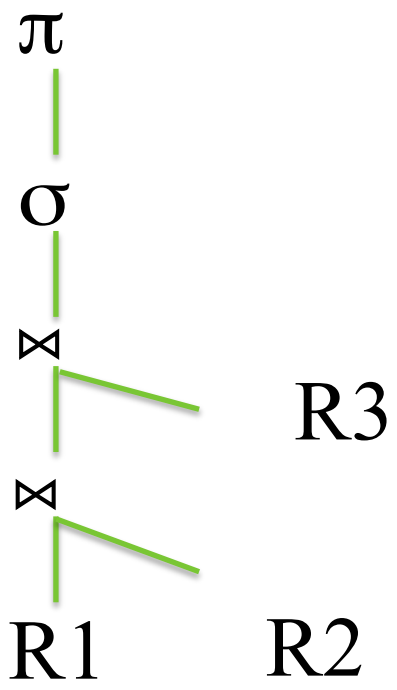
Hortonworks

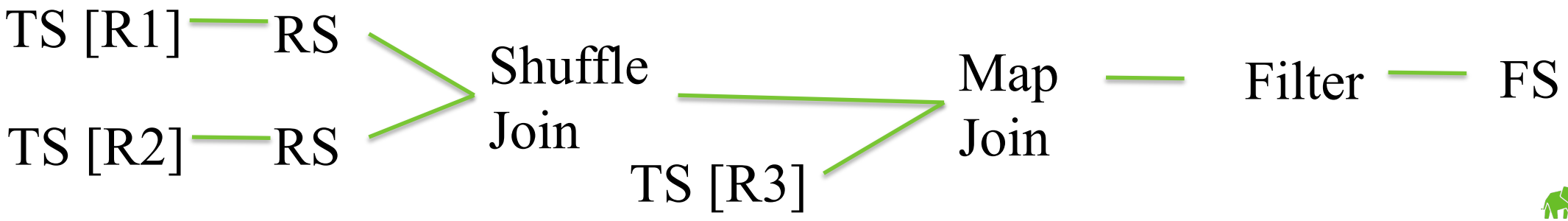
Query preparation – Hive 0.14



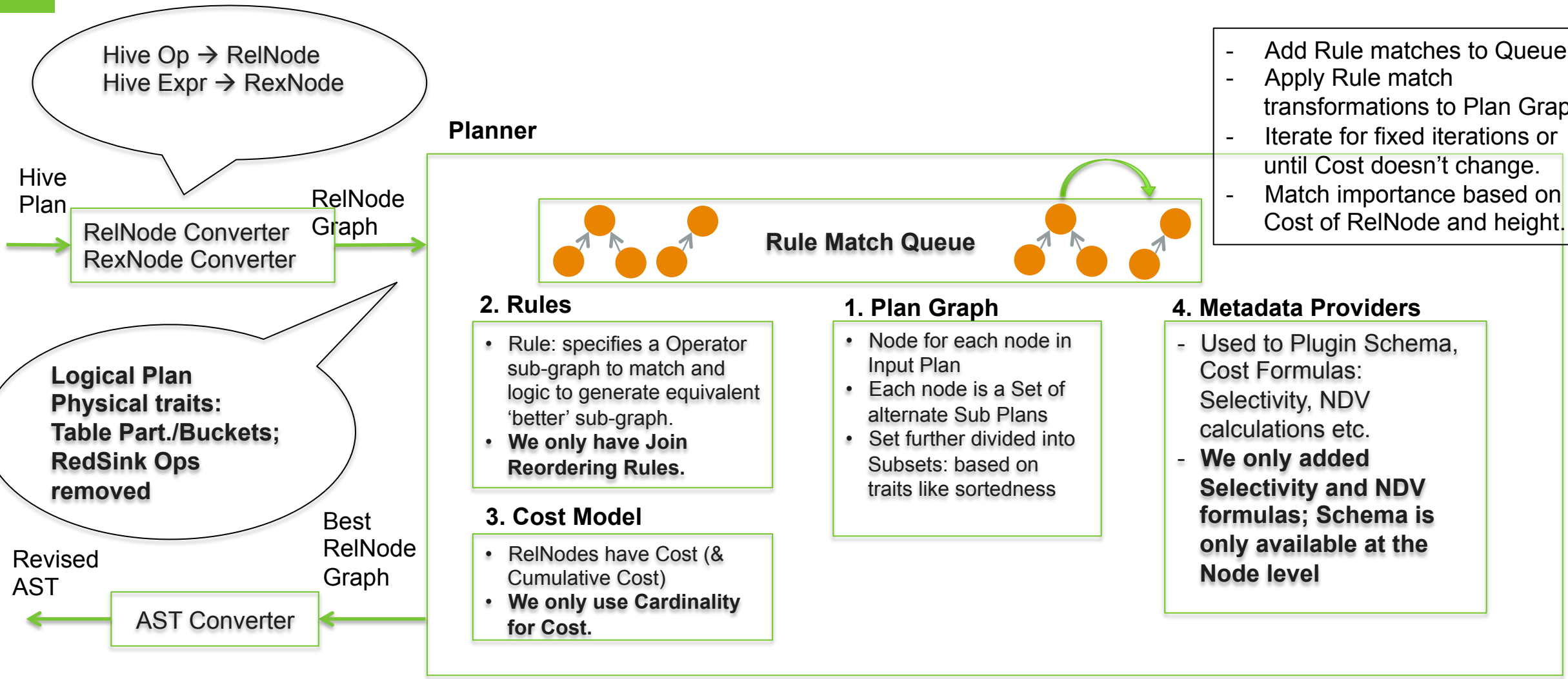
Query Execution – The basics

```
SELECT R1.x
FROM R1
  JOIN  R2 ON R1.x = R2.x
JOIN R3 on R1.x = R3.x AND R2.x = R3.x
WHERE R1.z > 10;
```

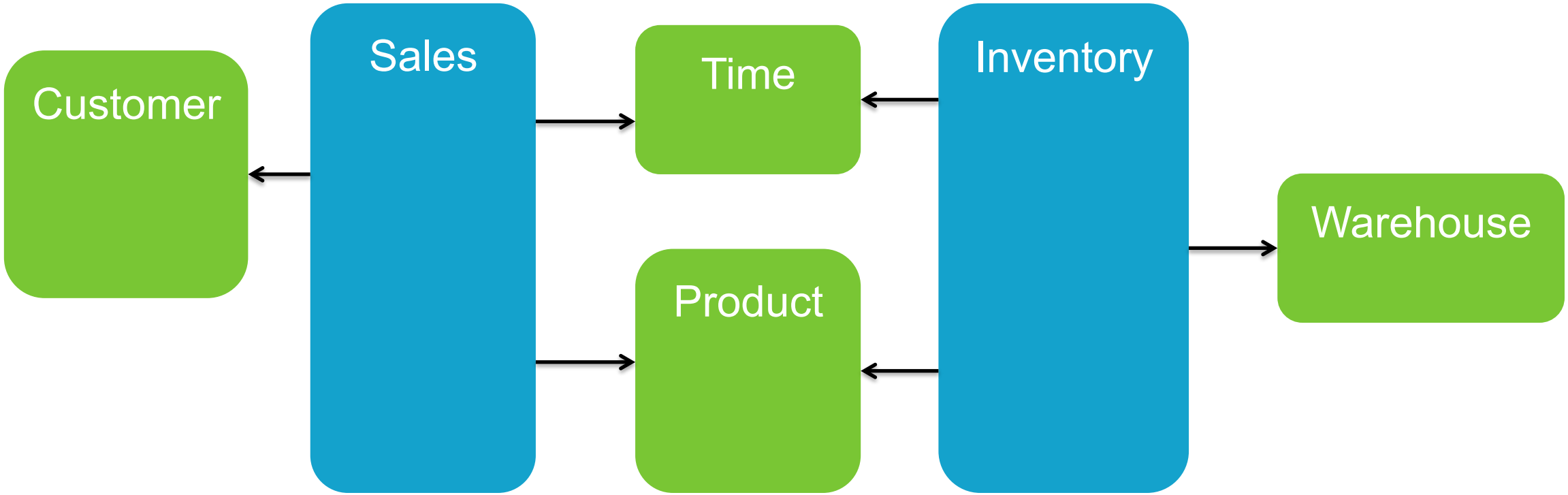




Optiq Planner Process



Star schema

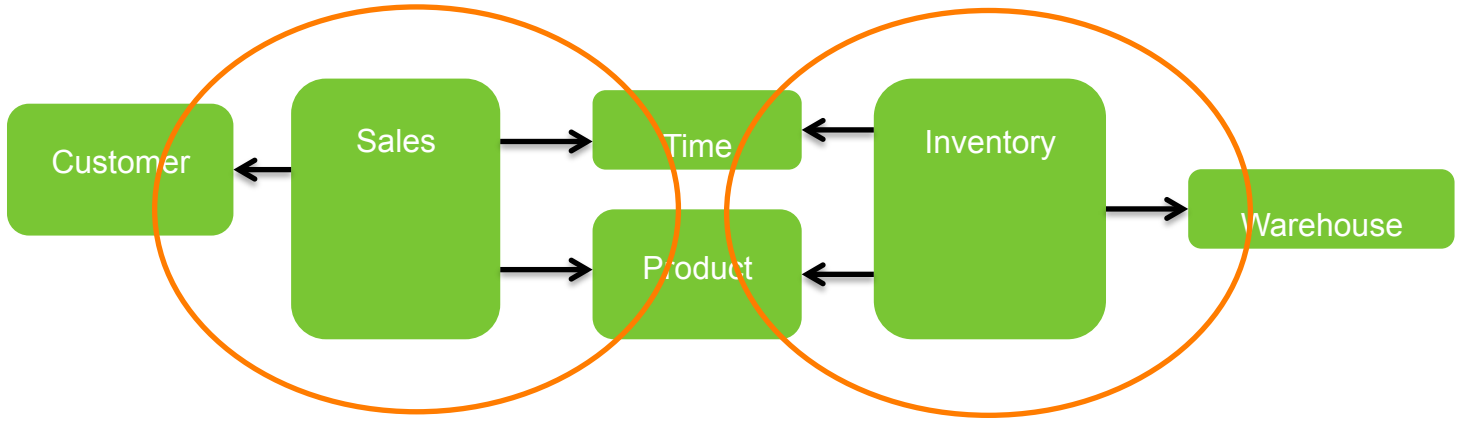


Key
Fact table
Dimension table
→ Many-to-one relationship

Hortonworks

Query combining two stars

```
SELECT product.id,  sum(sales.units), sum(inventory.on_hand)
FROM sales ON …
JOIN customer ON …
JOIN time ON …
JOIN product ON …
JOIN inventory ON …
JOIN warehouse ON …
WHERE time.year = 2014
AND time.quarter = 'Q1'
AND product.color = 'Red'
AND warehouse.state = 'WA'
GROUP BY …
```



Hortonworks

Left-deep tree

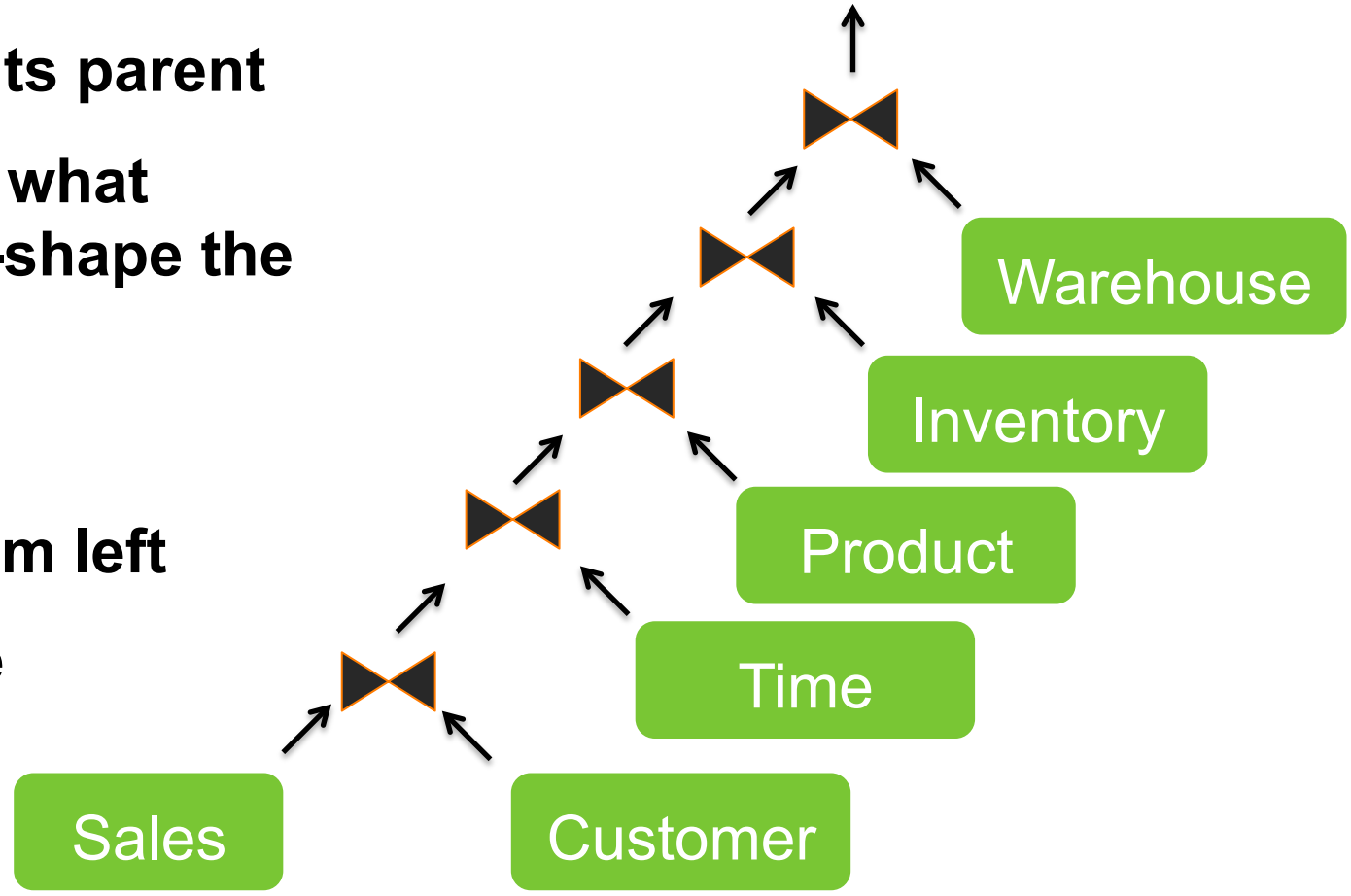
Initial tree is “left-deep”

No join node is the right child of its parent

Join-ordering algorithm chooses what order to join tables – does not re-shape the tree

Typical plan:

- **Start with largest table at bottom left**
- **Join tables with more selective conditions first**



Hortonworks

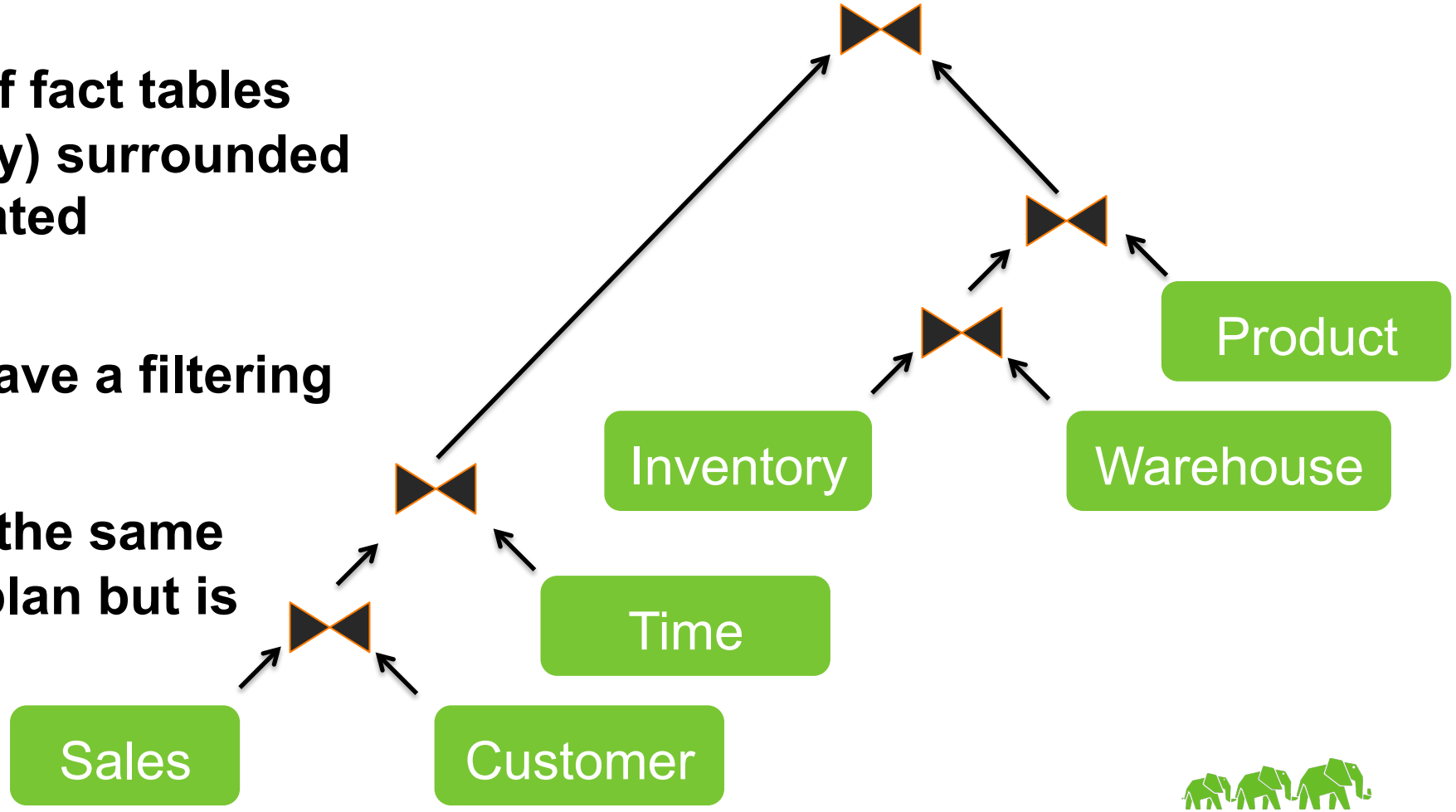
Bushy tree

No restrictions on where join nodes can occur

“Bushes” consist of fact tables (Sales and Inventory) surrounded by many-to-one related dimension tables

Dimension tables have a filtering effect

This tree produces the same result as previous plan but is more efficient



Hortonworks

Two approaches to join optimization

Algorithm #1 “exhaustive search”

Apply 3 transformation rules exhaustively:

- SwapJoinRule: A join B → B join A
- PushJoinThroughJoinRule: (A join B) join C → (A join C) join B
- CommutativeJoinRule: (A join B) join C → A join (B join C)

Finds every possible plan, but not practical for more than ~8 joins

Algorithm #2 “greedy”

Build a graph iteratively

Use heuristics to choose the “best” node to add next

Applying them to Hive

We can use both algorithms – and we do!

Both are sensitive to bad statistics – e.g. poor estimation of intermediate result set sizes

Statistics

Feeding the beast

CBO disabled if your tables don't have statistics

- No longer require statistics on all columns, just join columns

Better optimizations need ever-better statistics… so, statistics are getting better

Kinds of statistics

Raw statistics on stored data: row counts, number-of-distinct-values (NDV)

Statistics on intermediate operators, computed using selectivity estimates

- Much improved selectivity estimates this release, based on NDVs
- Planned improvements to raw statistics (e.g. histograms, unique keys, sort order) will help
- Materialized views

Run-time statistics

- Example 1: 90% of the rows in this join have the same key → use skew join
- Example 2: Only 10 distinct values of GROUP BY key → auto-reduce parallelism

Stored statistics – recent improvements

ANALYZE

Clean up command syntax

Faster computation

Table vs partition statistics

All statistics now stored per partition

Statistics retrieval

Faster retrieval

Merge partition statistics

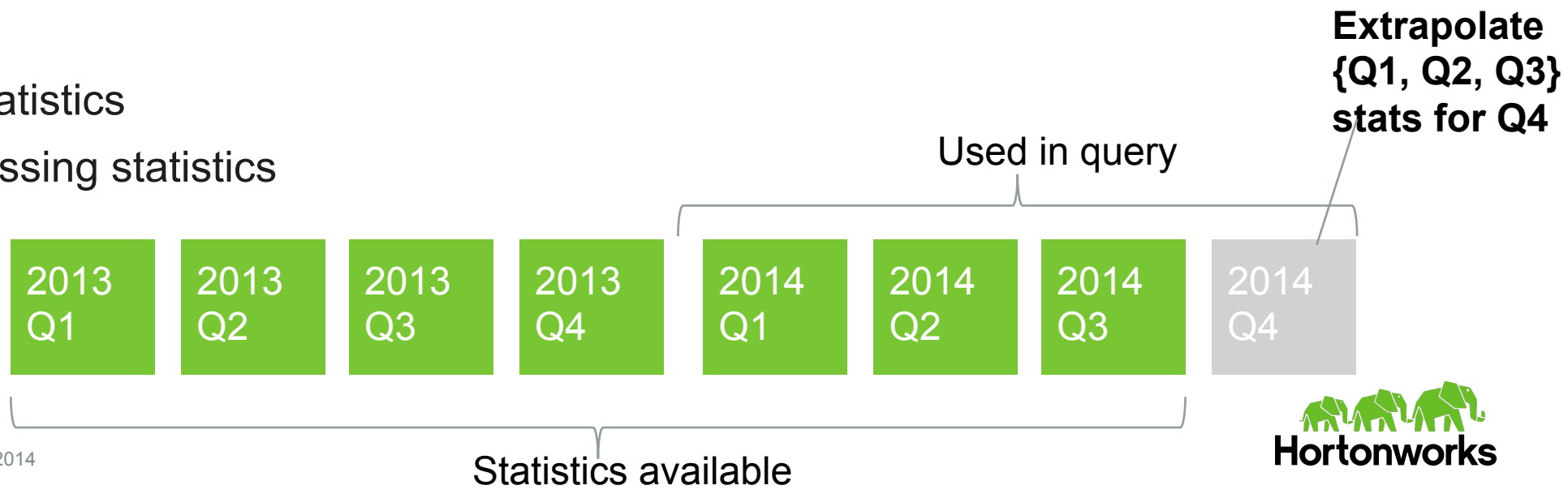
Extrapolate for missing statistics

Extrapolation

SQL:

```
SELECT productId, COUNT(*)
FROM Sales
WHERE year = 2014
GROUP BY productId
```

Required statistic: NDV(productId)

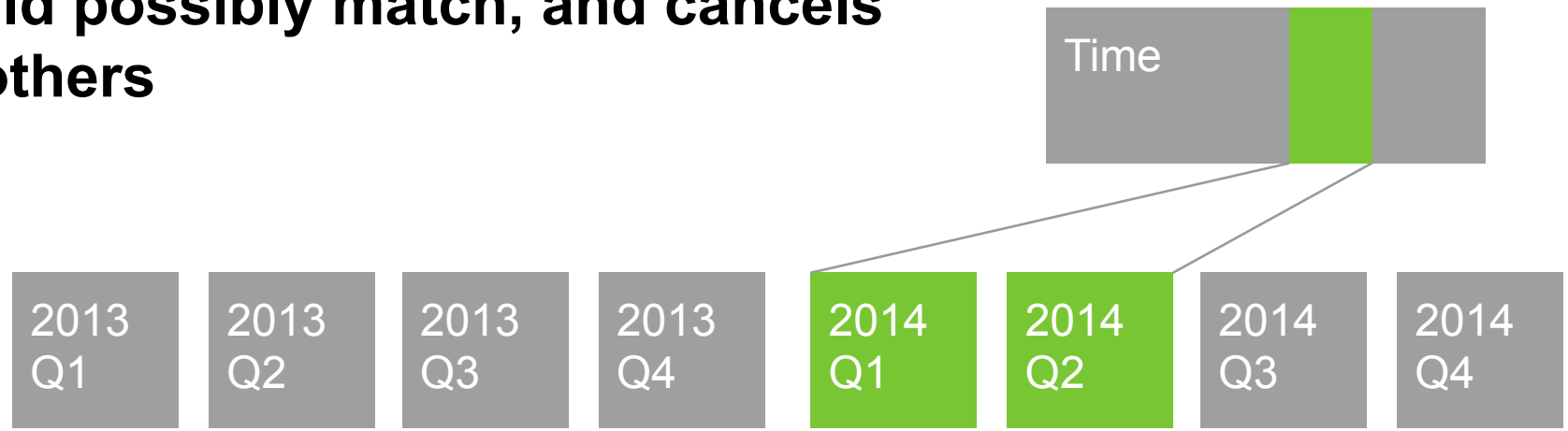


Dynamic partition pruning

Consider a query with a partitioned fact table, filters on the dimension table:

```
SELECT … FROM Sales
JOIN Time ON Sales.time_id = Time.time_id
WHERE time.year = 2014 AND time.quarter IN ('Q1', 'Q2')
```

At execute time, DAG figures out which partitions could possibly match, and cancels scans of the others



Hortonworks

Summary

Join-ordering: (exhaustive & heuristic), scalability, bushy joins

Statistics – faster, better, extrapolate if stats missing

Very few operators that CBO can't handle – TABLESAMPLE, SCRIPT, multi-INSERT

Dynamic partition pruning

Auto-reduce parallelism

Show me the numbers…

Hortonworks

TPC-DS (30TB) Q17

Joins Store Sales, Store Returns and Catalog Sales fact tables.

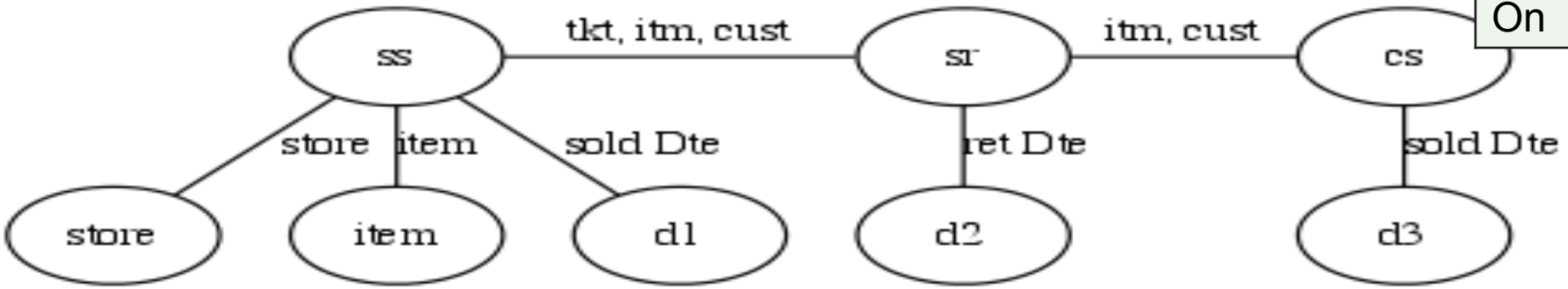
Each of the fact tables are independently restricted by time.

Analysis at Item and Store grain, so these dimensions are also joined in.

As specified Query starts by joining the 3 Fact tables.

```
SELECT i_item_id
      ,i_item_desc
      ,s_state
      ,count(ss_quantity) as store_sales_quantitycount
      ,....
FROM store_sales ss ,store_returns sr, catalog_sales cs,
 date_dim d1, date_dim d2, date_dim d3, store s, item I
WHERE d1.d_quarter_name = '2000Q1'
AND d1.d_date_sk = ss.ss_sold_date_sk
AND i.i_item_sk = ss.ss_item_sk AND …
GROUP BY i_item_id ,i_item_desc, ,s_state
ORDER BY i_item_id ,i_item_desc, s_state
LIMIT 100;
```

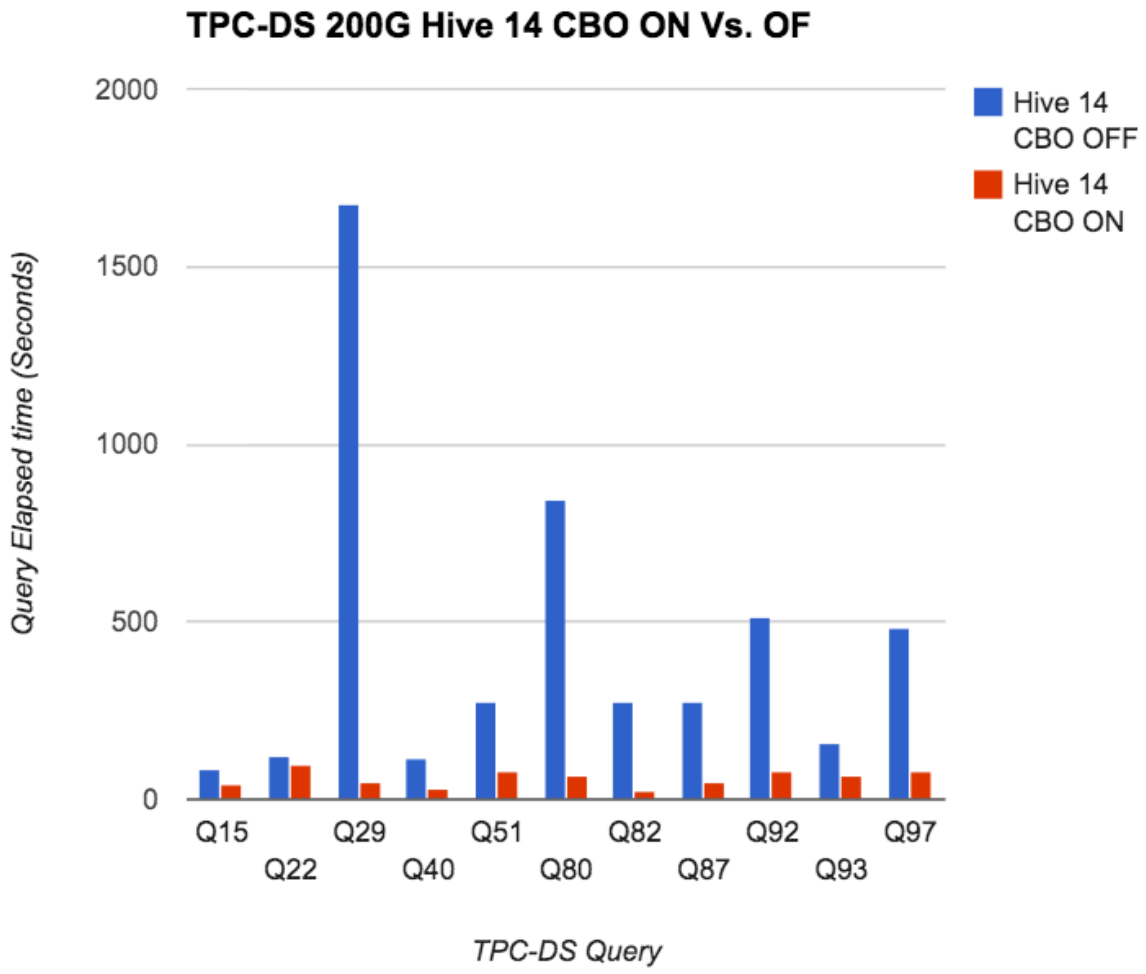
CBO	Elapsed (s)	Intemediate data (GB)
Off	10,683	5,017
On	1,284	275



Hortonworks

TPC-DS (200G) queries

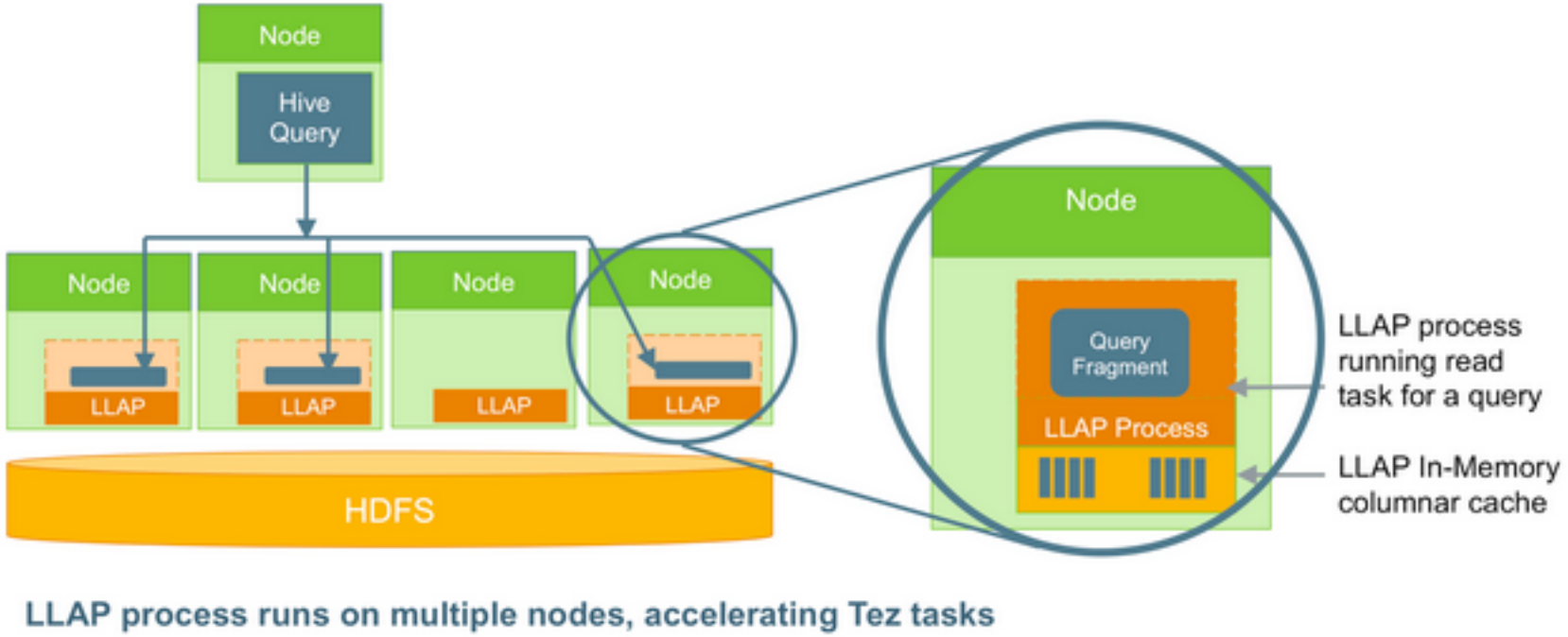
Query	Hive 14 CBO off	Hive 14 CBO on	Gain
Q15	84	44	91%
Q22	123	99	24%
Q29	1677	48	3,394%
Q40	118	29	307%
Q51	276	80	245%
Q80	842	70	1,103%
Q82	278	23	1,109%
Q87	275	51	439%
Q92	511	80	539%
Q93	160	69	132%
Q97	483	79	511%



Hortonworks

Stinger.next

- **SQL compliance: interval data type, non-equi joins, set operators, more sub-queries**
- **Transactions: COMMIT, savepoint, rollback)**
- **LLAP**
- **Materialized views**
  - In-memory
  - Automatic or manual

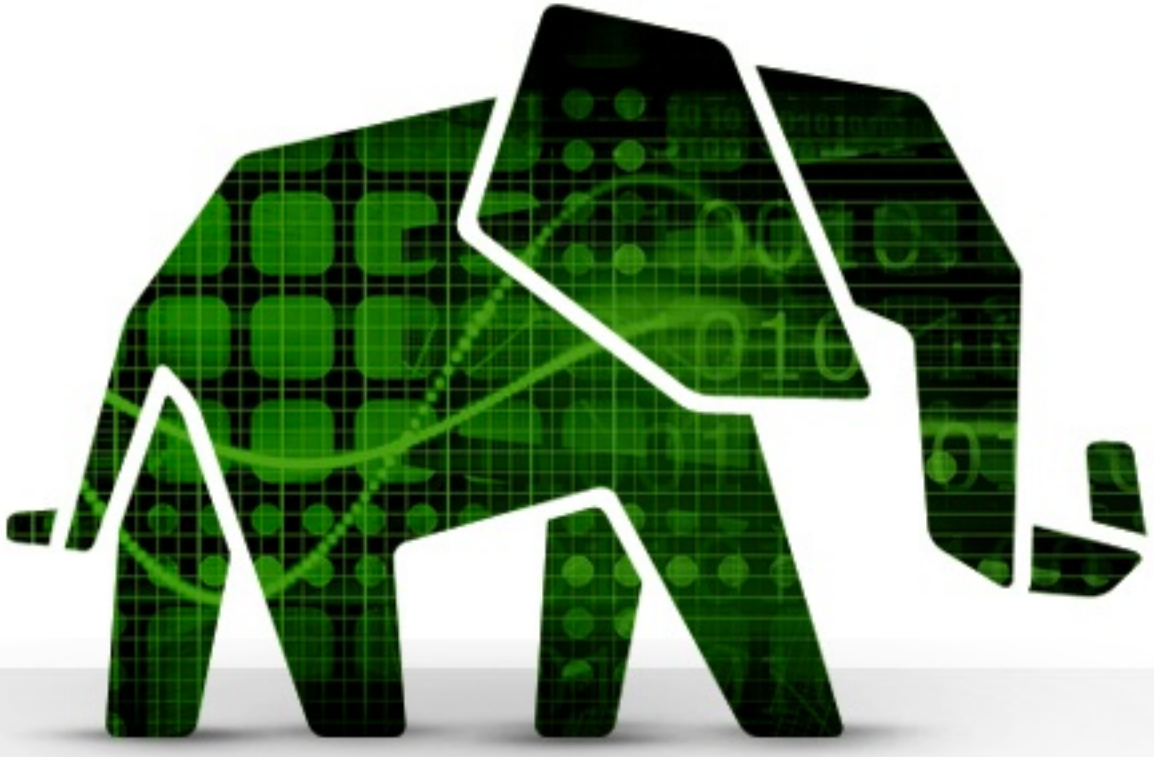


LLAP process runs on multiple nodes, accelerating Tez tasks

http://hortonworks.com/blog/stinger-next-enterprise-sql-hadoop-scale-apache-hive/

Hortonworks

Thank you!



@julianhyde

http://hive.apache.org/

http://optiq.incubator.apache.org/

Hortonworks