Apache Tajo : A Big Data Warehouse System on Hadoop

김형준
babokim@gmail.com

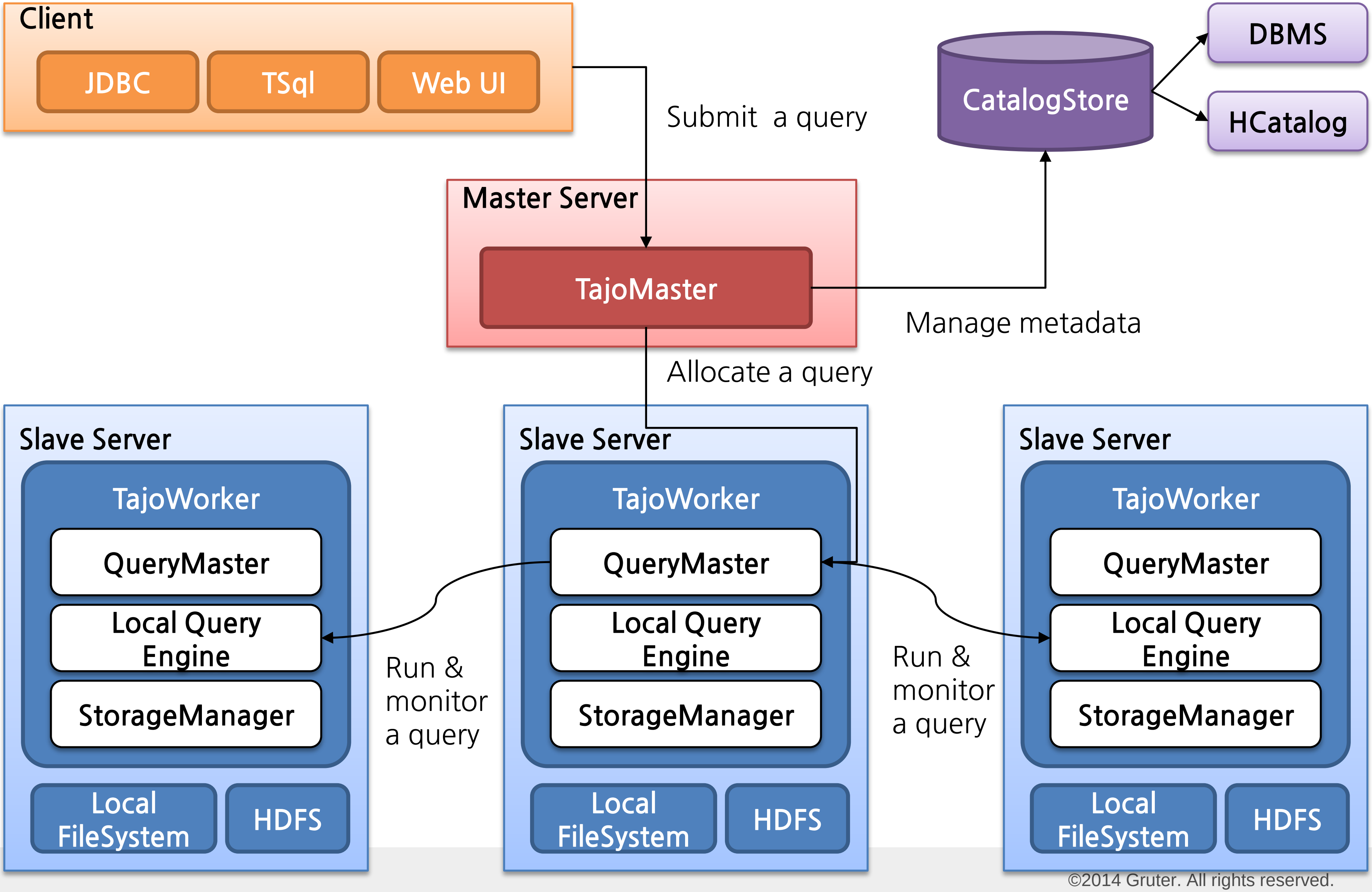
Apache Tajo Overview

- A big data warehouse system on Hadoop
- Apache Top-level project since March 2014
- Supports SQ

- Features
  - – Powerfu te Not MapReduce)
  - – Advance ati techniques
  - – Long rur r
  - – Interactive analysis queries : from 100 milliseconds
- Recent 0.9.0 release

TAJO

Tajo Architecture



Mature SQL Feature Set

- **Fully distributed query executions**
  - Inner join, and left/right/full outer join
  - Groupby, sort, multiple distinct aggregation
  - window function
- **SQL data types**
  - CHAR, BOOL, INT, DOUBLE, TEXT, DATE, Etc
- **Various file formats**
  - Text file (CSV), SequenceFile, RCFile, Parquet, Avro
- **SQL Standards**
  - Non standard features : PgSQL and Oracle

Group by, Order by

- **Group by**
  - – Multi-level distributed Group by
    - tajo.dist-query.groupby.multi-level-aggr
  - – Multiple Count Distinct
    - Select count(distinct col1), sum(distinct col2), …
  - – Hash Aggregation or Sort Aggregation
- **Order by**
  - – Fully distributed order by
  - – Range partition

Join

- Join
  - NATURAL, INNER, OUTER (LEFT, RIGHT, FULL)
  - SEMI, ANTI Join (planned for v0.9)
- Join Predicates
  - WHERE and ON predicates
  - de-factor standard outer join behavior with both predicates

```
SELECT * FROM t1 LEFT JOIN t2
ON t1.num = t2.num WHERE t2.value = 'xxx';
```

```
SELECT * FROM t1 LEFT JOIN t2
WHERE t1.num = t2.num and t2.value = ‘xxx’;
```

Table Partitions

- Column Value Partition
  - Hive Compatible Partition

```
CREATE TABLE T1 (C1 INT, C2 TEXT)
  using PARQUET
    WITH (‘parquet.compression’ = ‘SNAPPY’)
  PARTITION BY COLUMN (C3 INT, C4 TEXT);
```

- Range Partition (planned for 1.0)
  - Table will be partitioned by disjoint ranges.
  - Will remove the partition granularity problem of Hive Partition

Window Function

- OVER clause
  - row_number() and rank()
  - Aggregation function support
  - PARTITION and ORDER BY clause

```
SELECT depname, empno, salary, enroll_date FROM (  SELECT
    depname, empno, salary, enroll_date,
    rank() OVER (PARTITION BY depname
        ORDER BY salary DESC, empno) AS pos
  FROM empsalary
) AS ss WHERE pos < 3;
```

Currently Not Supported

- **IN/Exists SubQuery(2014, 4Q)**
  - Select * from t1 where col1 in (select col1…)
- **Scalar SubQuery(2015, 1Q)**
  - Select col1, (select col2 from …) from …
- **Some analytic function(2015, 1Q or 요청시)**
  - ROLLUP, CUBE, some Window
- **Create/Drop Function(2014, 4Q)**
  - 실행 중에 UDF를 추가하는 기능 미제공
- **Alter meta property(2014, 4Q)**
  - Partition 정보 등

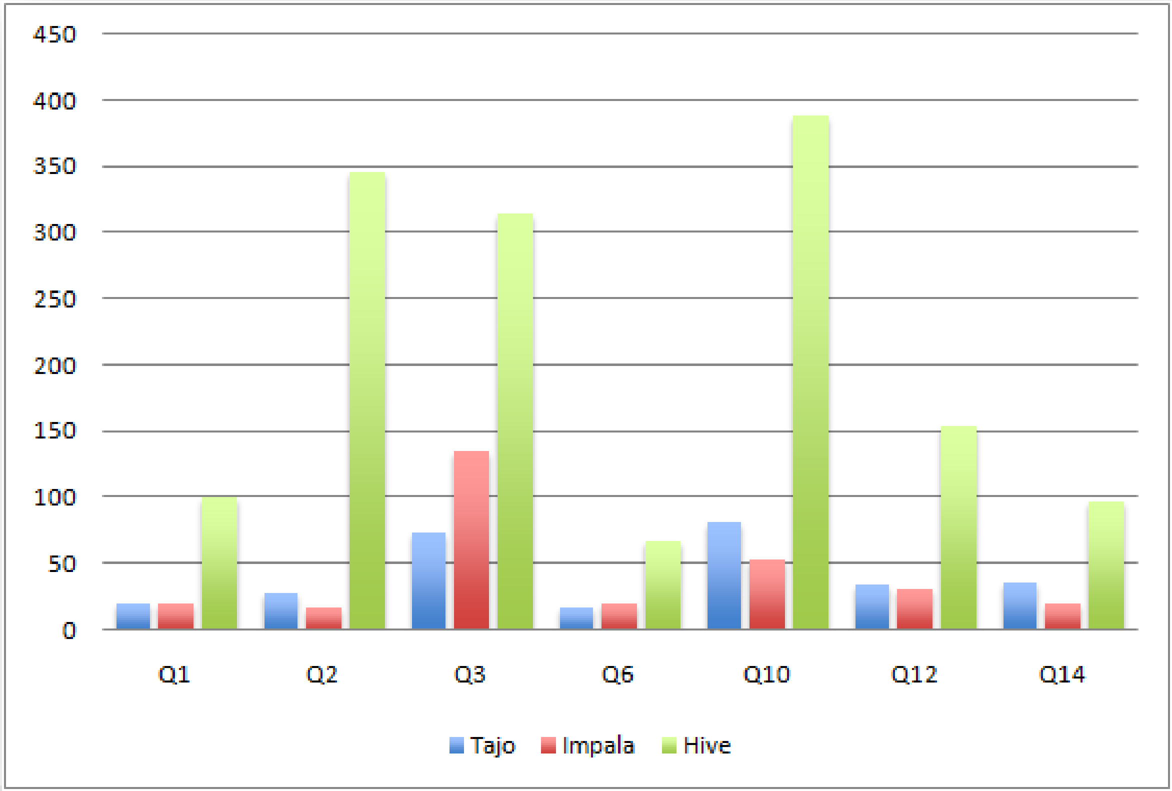
Comparison with other platform

** 버전 업 등에 따라 내용은 틀릴 수 있음*

Function	Tajo	Hive	Impala	Spark
Computing	자체	MapReduce or Tez	자체	자체
Resource Management	자체 or YARN	YARN	자체	자체 or YARN
Scheduler	FIFO, Fair	FIFO, Fair, Capacity	FIFO, Fair	FIFO, Fair
Storage	HDFS, S3, HBase	HDFS, HBase, S3	HDFS, HBase	자체 RDD (HDSF 등)
File Format	CSV, RC, Parquet, Avro 등	CSV, RC, ORC, Parquet, Avro 등	CSV, RC, Parquet, Avro 등	CSV, RC, Parquet, Avro 등
Data Model	Relational	Nested Data Model	Relational	Nested Data Model
Query	ANSI-SQL	HiveQL	HiveQL	HiveQL
구현 언어	Java	Java	C++	Scala
Client	Java API, JDBC, CLI	CLI, JDBC, ODBC, Thrift Server API	CLI, JDBC, ODBC	Shark JDBC/ODBC, Scala, Java, Python API
Query Latency	Long run, Interactive	Long run, (Interactive-Tez)	Interactive	Interactive
컴퓨팅 특징	데이터는 Disk, 중간 데이터는 Memory/Disk 모두 사용	데이터는 Disk, 중간 데이터는 Memory/Disk 모두 사용	중간 데이터가 In-Memory (최근 On-Disk 지원)	분석 대상 데이터가 In-Memory에 로딩
License	Apache	Apache	Apache	Apache
Main Sponsor	Gruter	Hortonworks	Cloudera	Databricks

Performance(1)

- 테스트 장비: 1 master + 6 workers
  CPU: Xeon 2.5GHz, E5, 24 Core,   Memory: 64GB, Network:10Gb
  Disk: 3TB * 6 SATA/HDD (7200 RPM)
- 테스트 데이터: TPC-H Scale 100 or 1000
- Version
  Hadoop: cdh-4.3.0,   Hive: 0.10.0-cdh4.3.0,   Impala: impalad_version_1.1.1_RELEASE
  Tajo: 0.2-SNAPSHOT

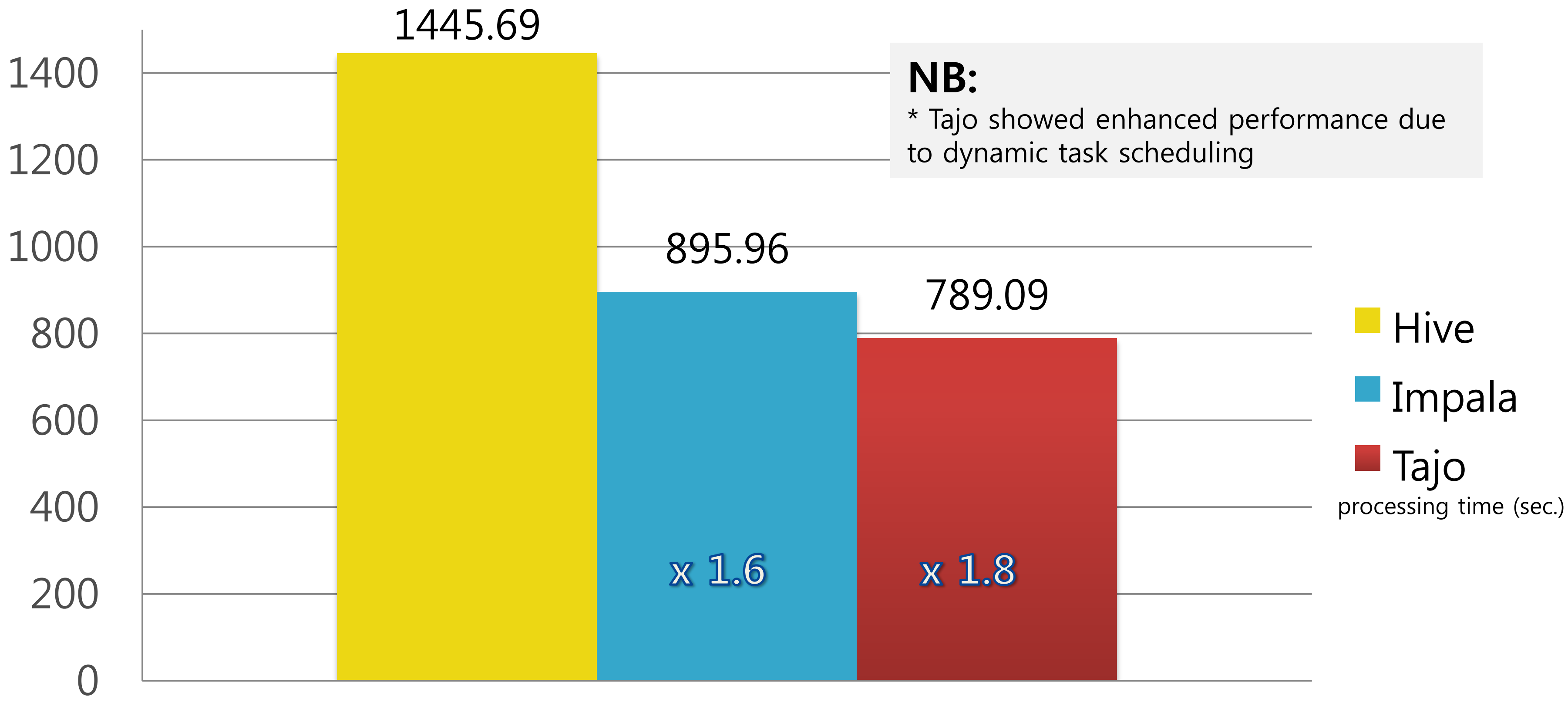


Performance(2)

- **실제 사용 데이터 및 질의**
  - 데이터
    - 1.7TB (4.1B rows, Q1)
    - 8 or less GB (results of Q1, rest of Qs)
  - Query
    - Q1: scan using about 20 text pattern matching filters
    - Q2: 7 unions with joins
    - Q3: join
    - Q4: group by and order by
    - Q5: 30 text pattern matching filters with OR conditions, gr oup by, having, and order by 4

http://www.slideshare.net/gruter/tajo-case-studybayareahugmeetupevent20131105

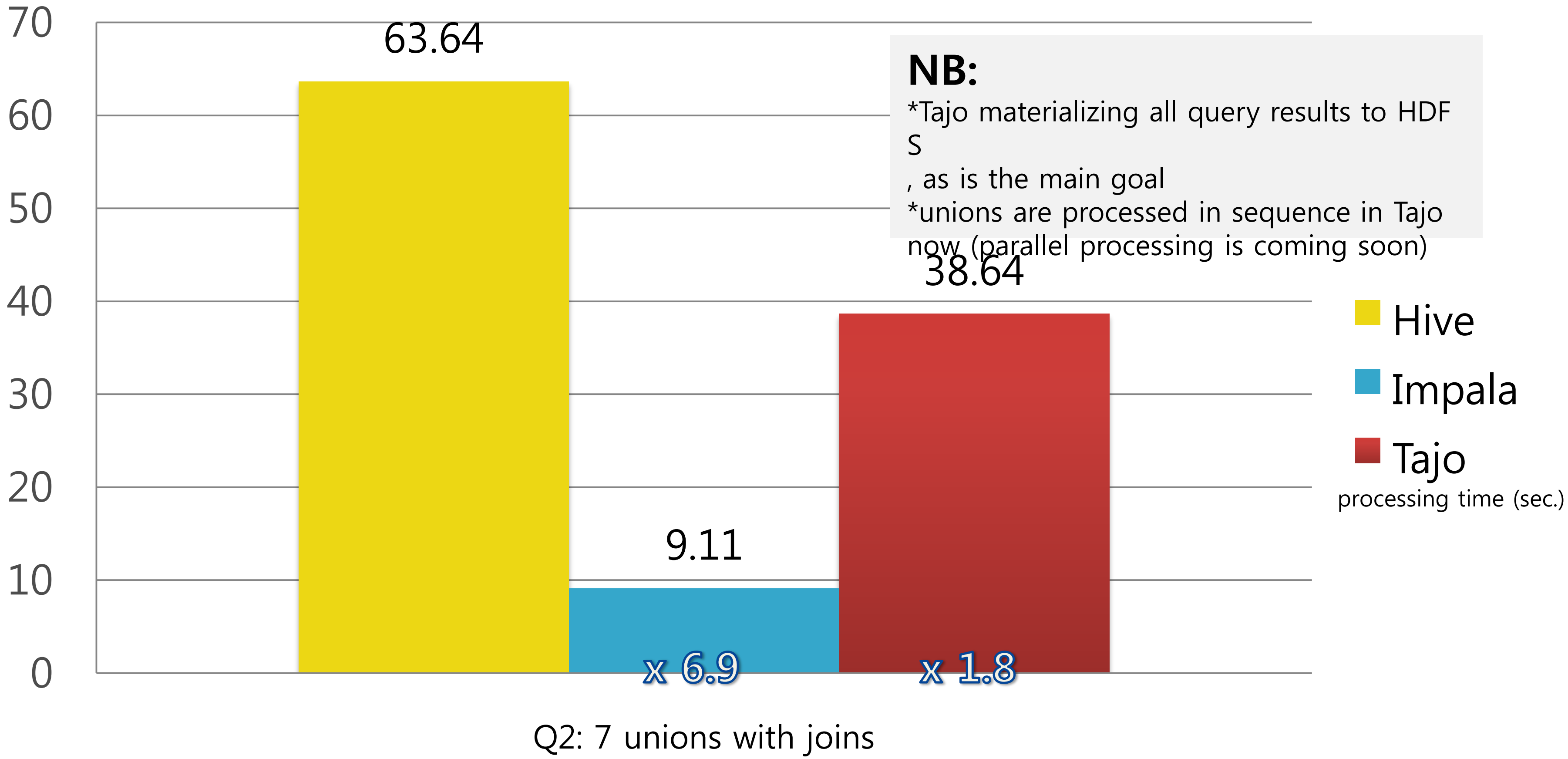
Q1 – filter scan



NB:
* Tajo showed enhanced performance due to dynamic task scheduling

Q1: scan using about 20 text pattern matching filters

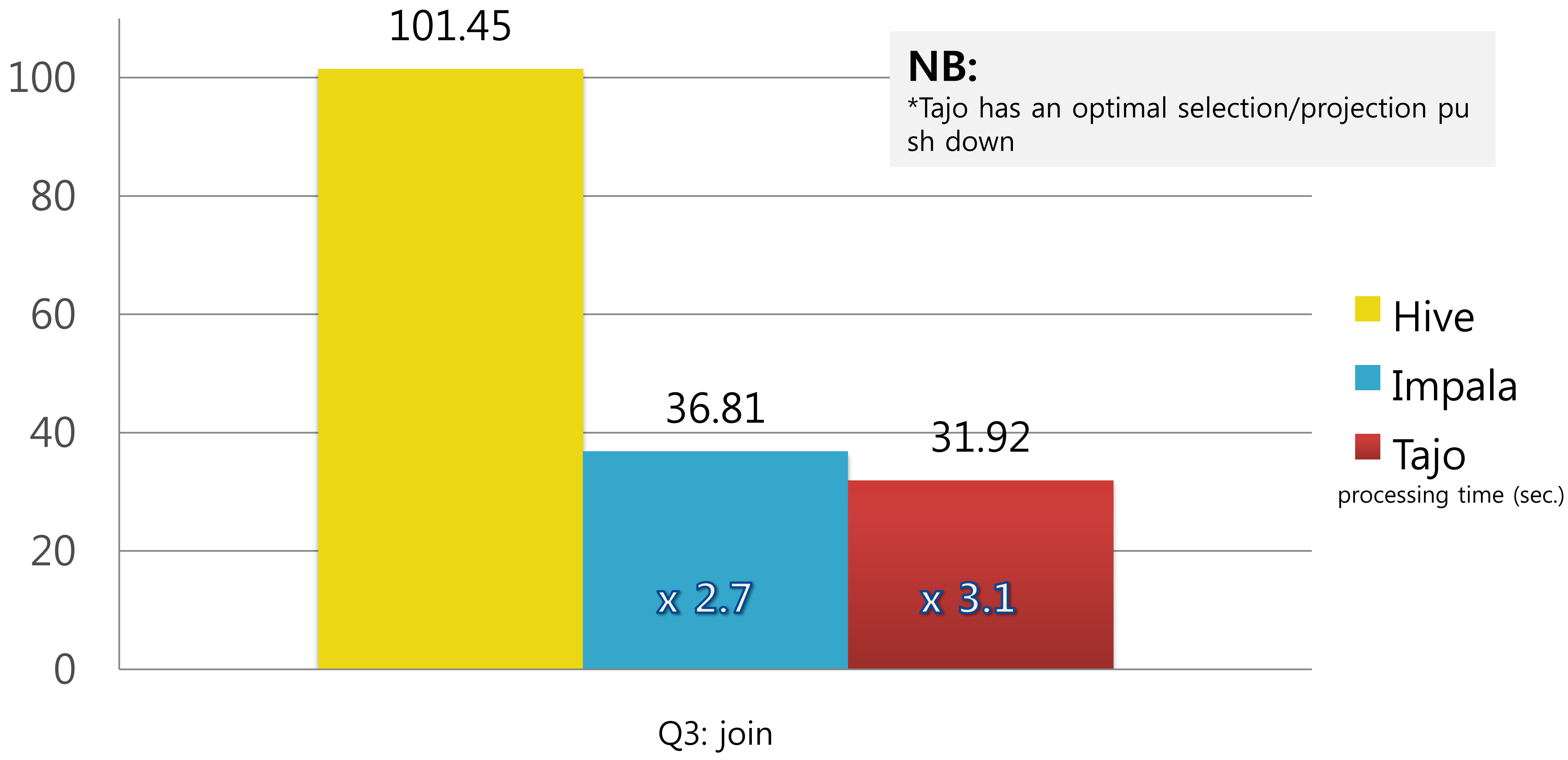
Q2 – unions, joins



NB:

*Tajo materializing all query results to HDF
S
, as is the main goal
*unions are processed in sequence in Tajo now (parallel processing is coming soon)

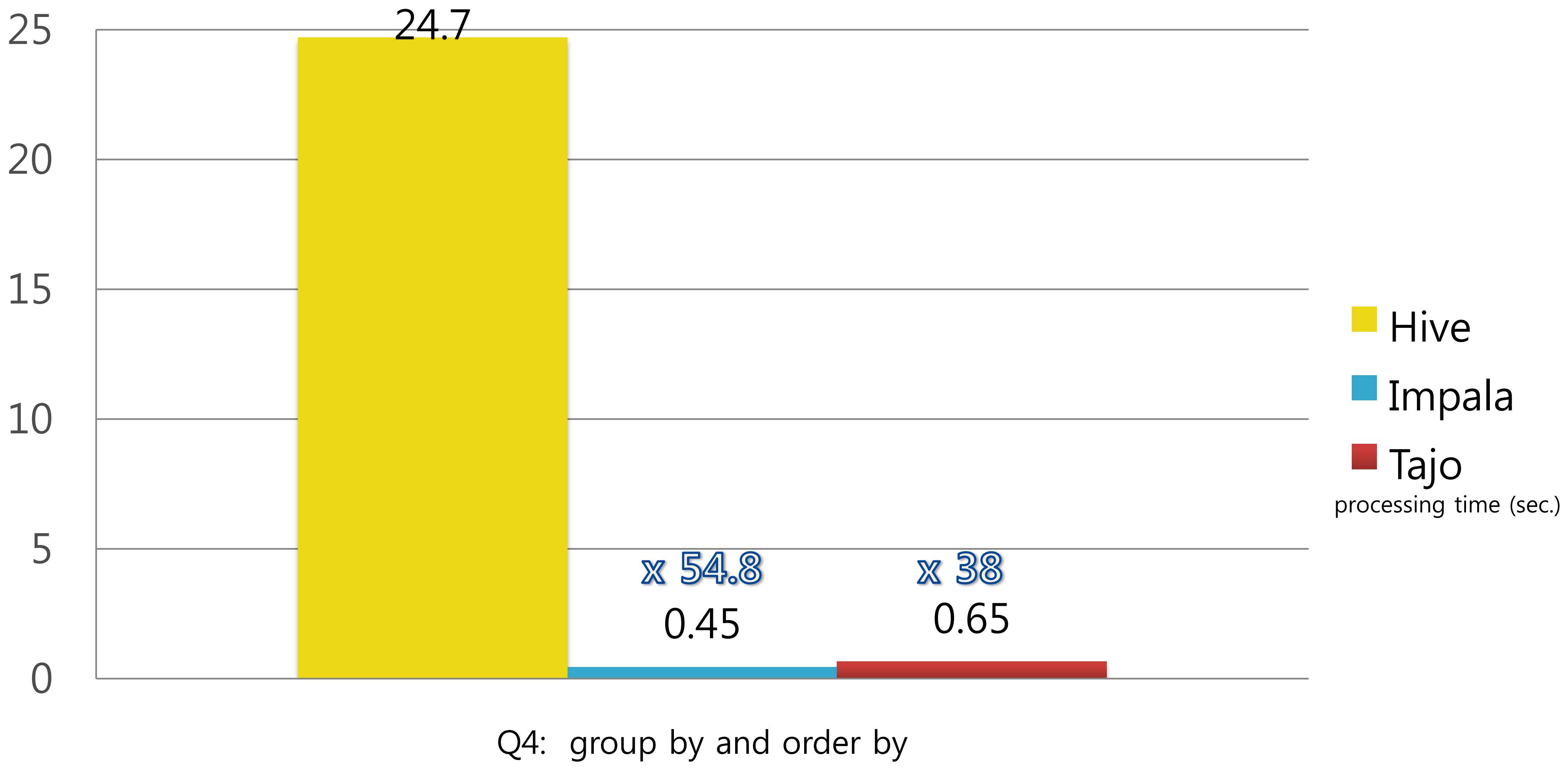
Q3 – join



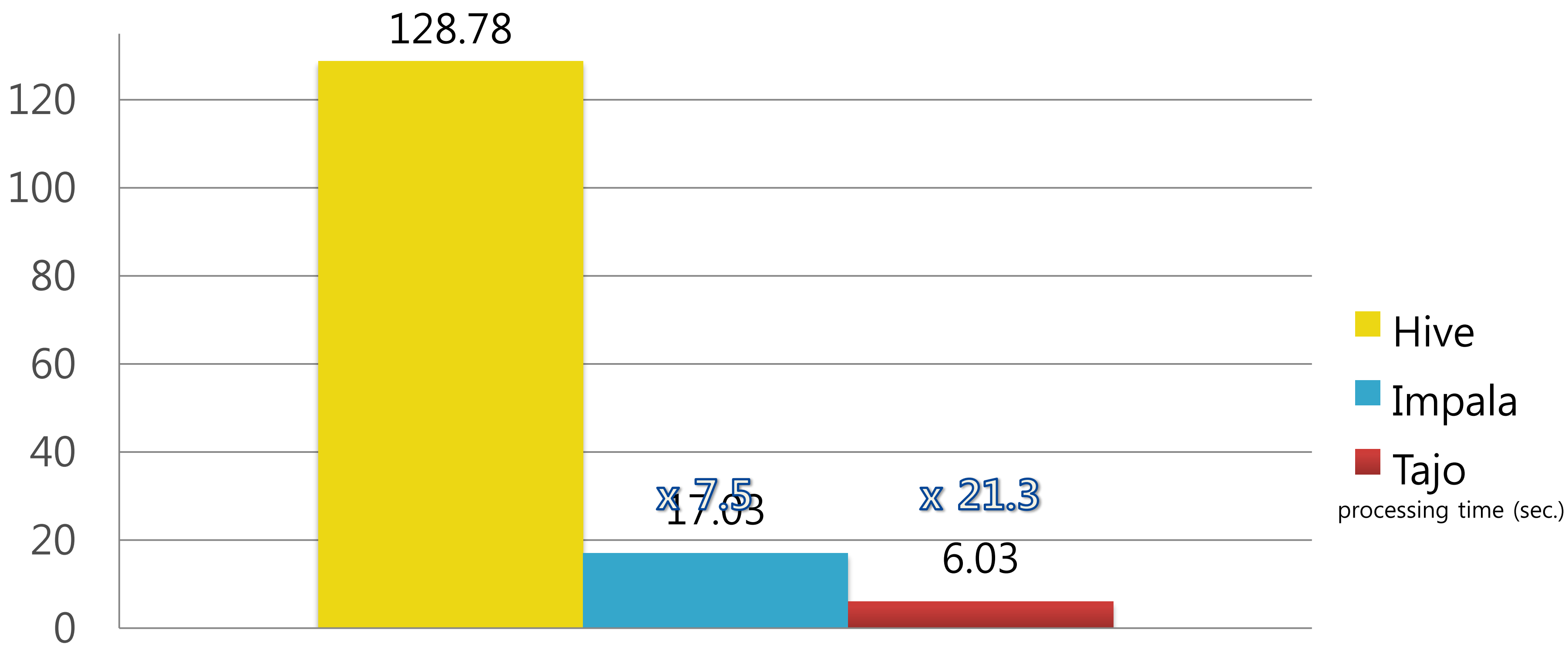
NB:
*Tajo has an optimal selection/projection pu sh down

Q4 – group by and sort

HIVE • cloudera IMPALA • TAJO



Q5 – filters, group by, having and sort

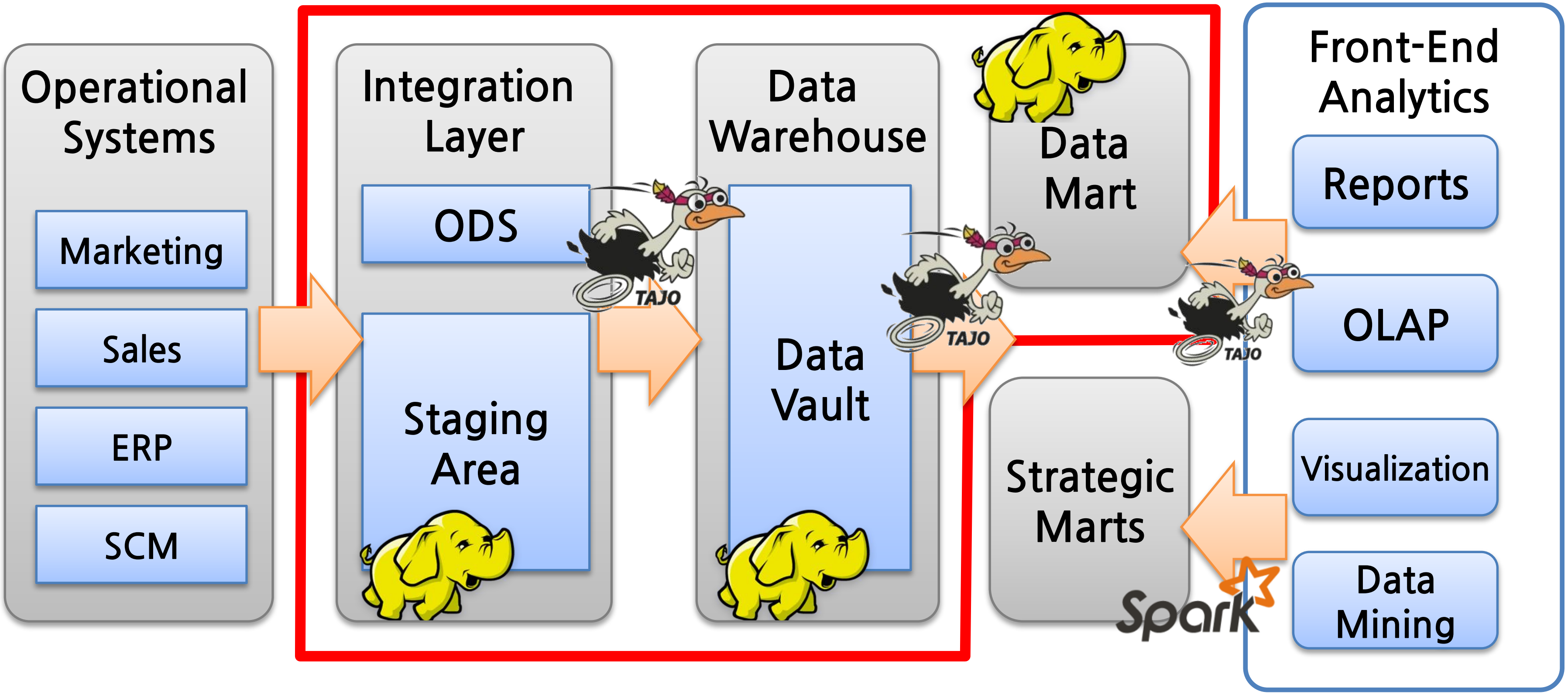


Q5:

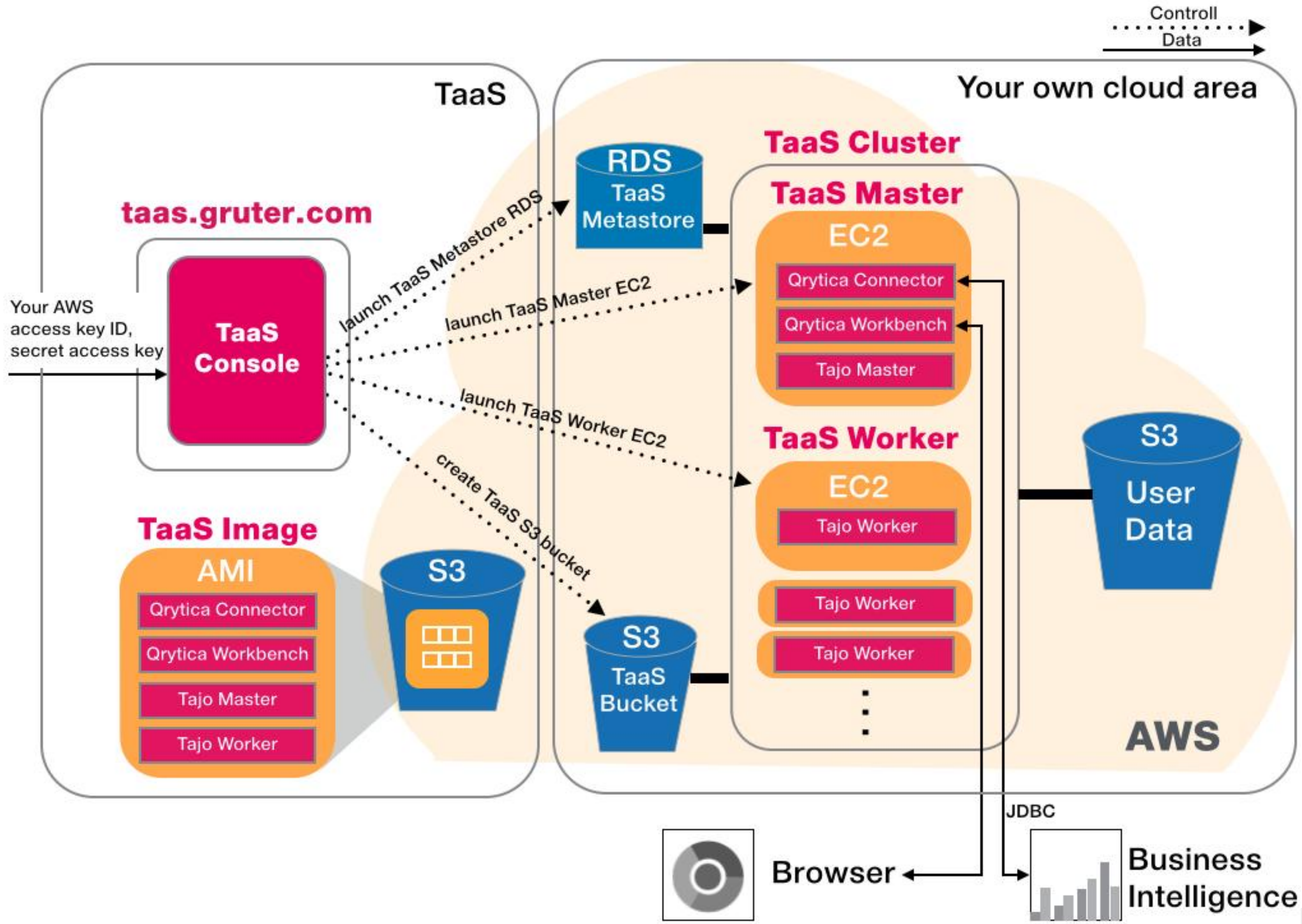
Q6: 30 Text pattern matching filters with OR conditions, group by, having, and order by resulting in smaller set of output

Replace Commercial Data Warehouse

- ETL(Batch) Processing: 120+ queries, ~4TB read/day
- OLAP(Interactive) Processing: 500+ queries



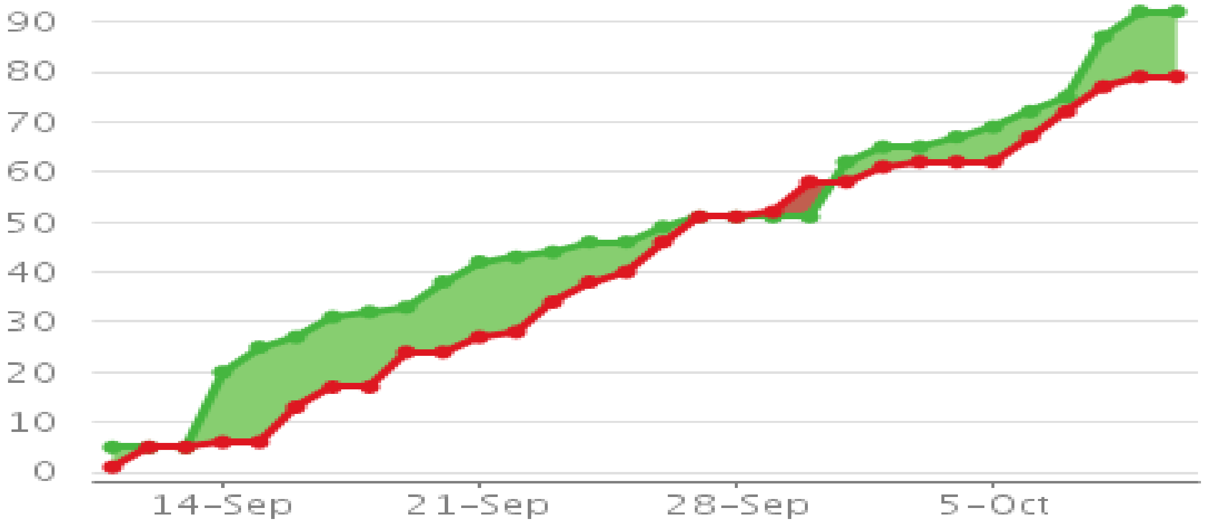
Tajo-as-a-Service on AWS



Active Open Source Community

- Fully community-driven open source
- Stable development team
  - – 17 committers + many contributors

Issues: 30 Day Summary



Issues: 79 created and 92 resolved

Future Works

- 2014 4Q
  - HBase intergation
  - In/Exists SubQuery
  - User defined function
  - Multi-tenant Scheduler
- 2015 1Q
  - Authentication and Standard Access Control
  - Scalar SubQuery
  - ROLLUP, CUBE
- 2015 2Q
  - Vectorized Engine(C++ Operator)

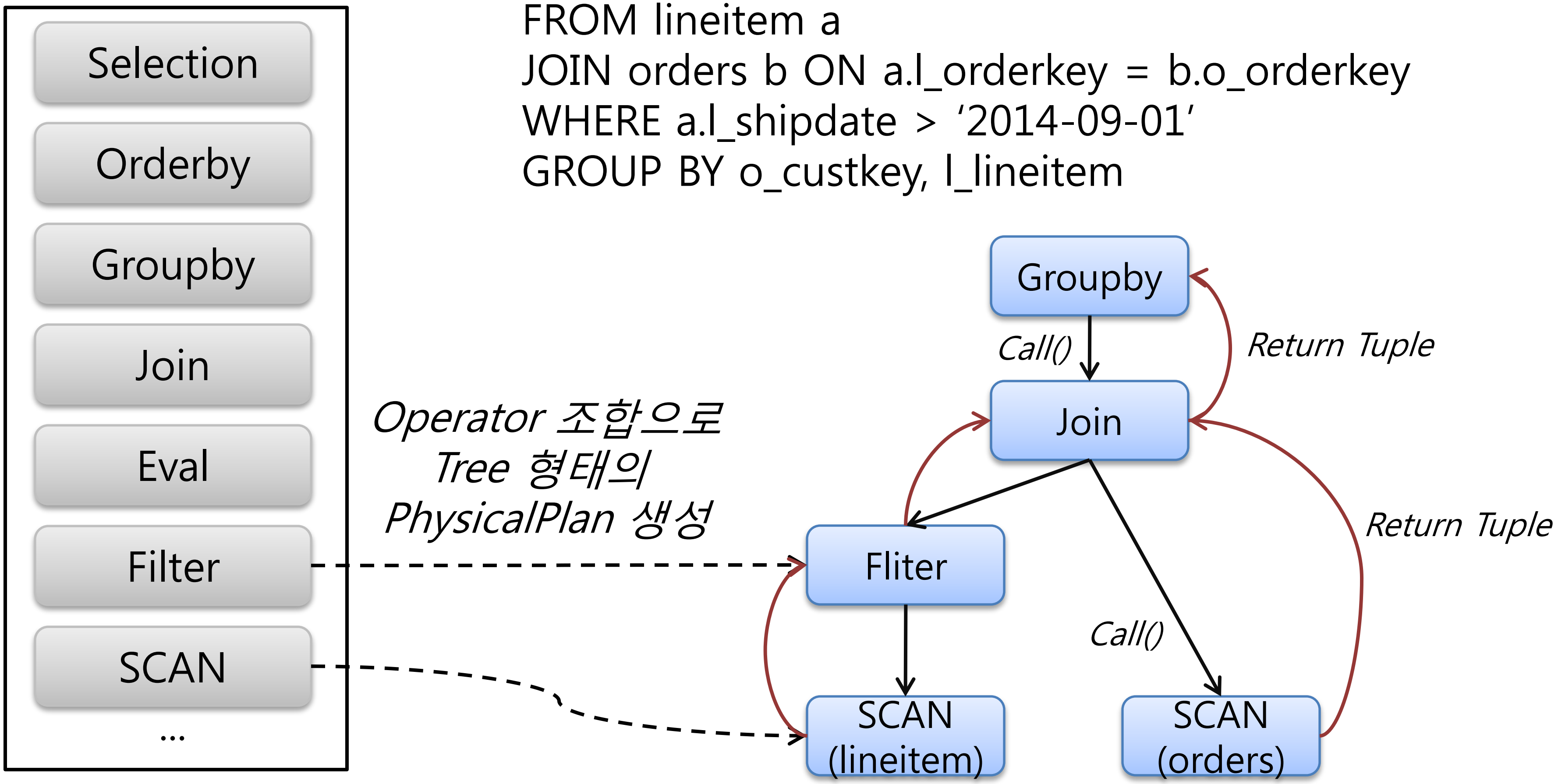
Gruter에서 진행하는 마일스톤으로
Apache Tajo 커뮤니티 방향과는 다를 수 있음

TAJO INTERNAL

DBMS 실행 방식

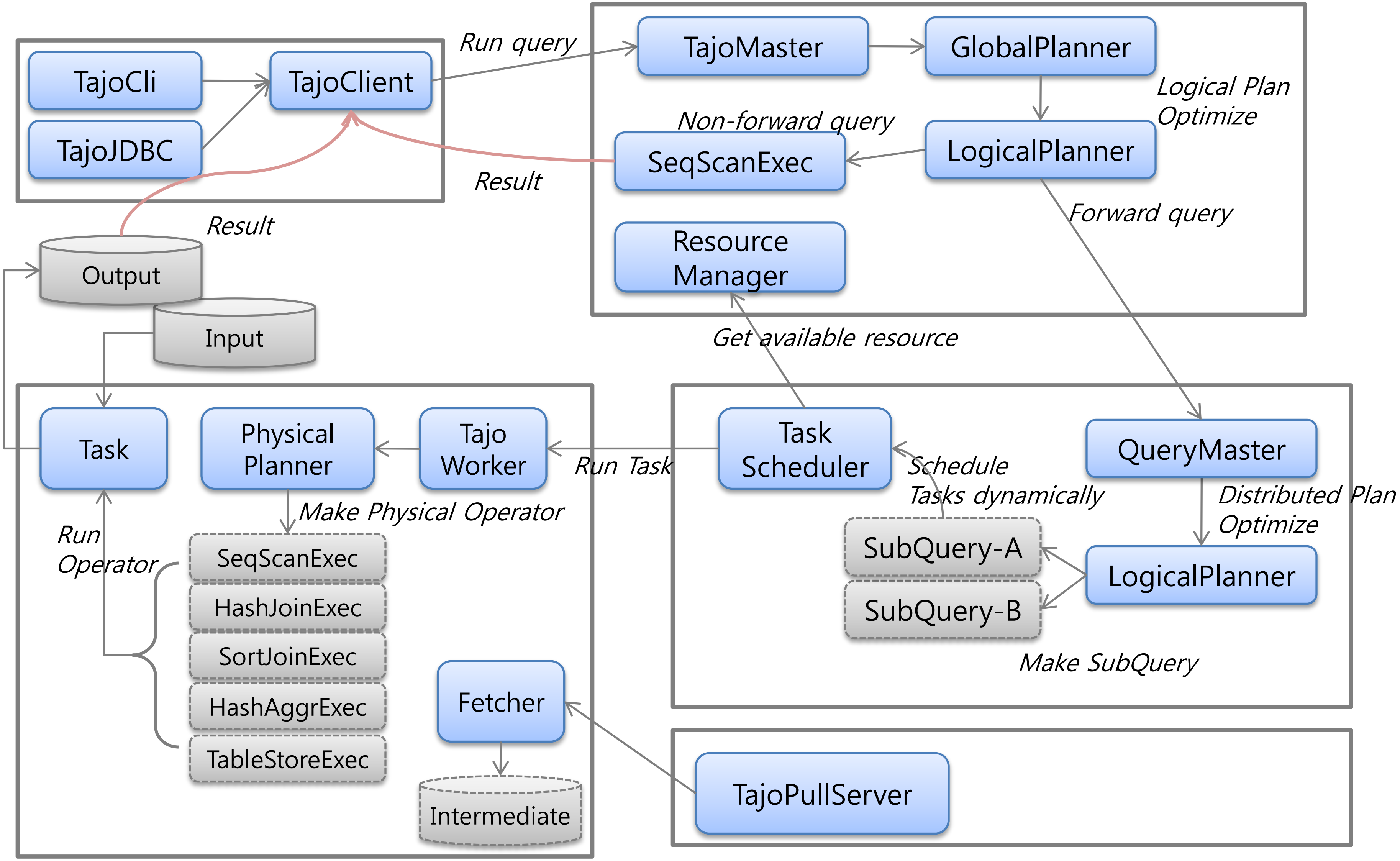
```
SLECT o_custkey, l_lineitem, count(*)
FROM lineitem a
JOIN orders b ON a.l_orderkey = b.o_orderkey
WHERE a.l_shipdate > '2014-09-01'
GROUP BY o_custkey, l_lineitem
```

Basic Operator

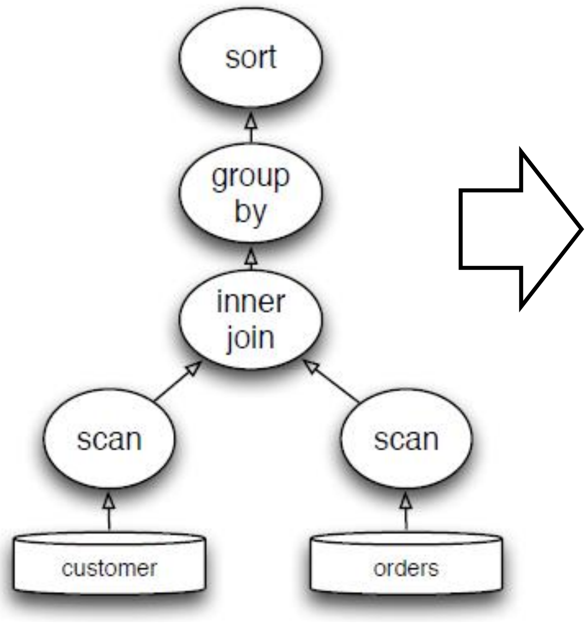


Operator 조합으로 Tree 형태의 PhysicalPlan 생성

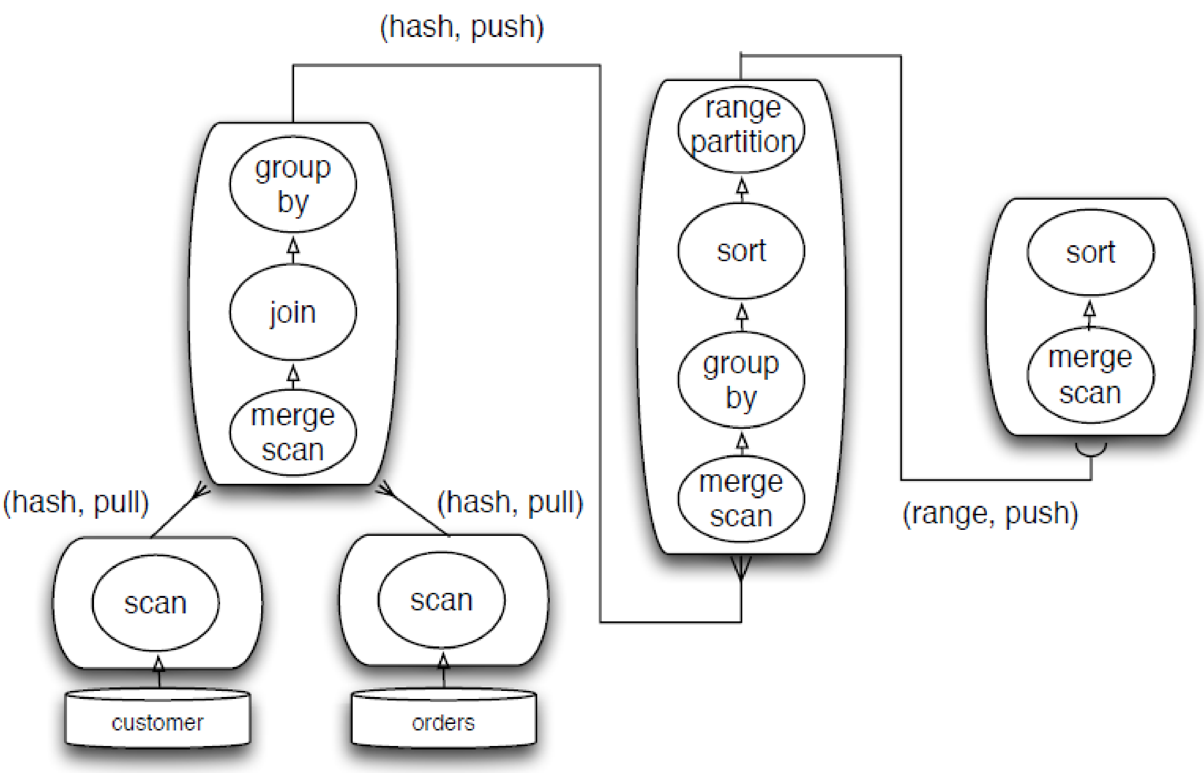
Query Execution Flow



Logical Plan/Distributed Plan



(A join-groupby-sort query plan)



(A distributed query execution plan)

select col1, sum(col2) as total, avg(col3) as averagefrom r1, r2
where r1.col1 = r2.col2 group by col1 order by average;

Logical Plan Optimizer

- **Basic Rewrite Rule**
  - Common sub expression elimination
  - Constant folding (CF), and Null propagation
- **Projection Push Down (PPD)**
  - push expressions to operators lower as possible
  - narrow read columns
  - remove duplicated expressions
    - if some expressions has common expression
- **Filter Push Down (FPD)**
  - reduce rows to be processed earlier as possible
- **Extensible Rewrite Rule**
  - Allow developers to write their own rewrite rules

Logical Plan Optimizer

```
SELECT
  item_id,
  order_id
  sum_price * (1.2 * 0.3)
  as total,
FROM (
  SELECT
    item_id,
    order_id,
    sum(price) as sum_price
  FROM
    ITEMS
  GROUP BY item_id, order_id
) a
WHERE item_id = 17234
```

CF + PPD

FPD

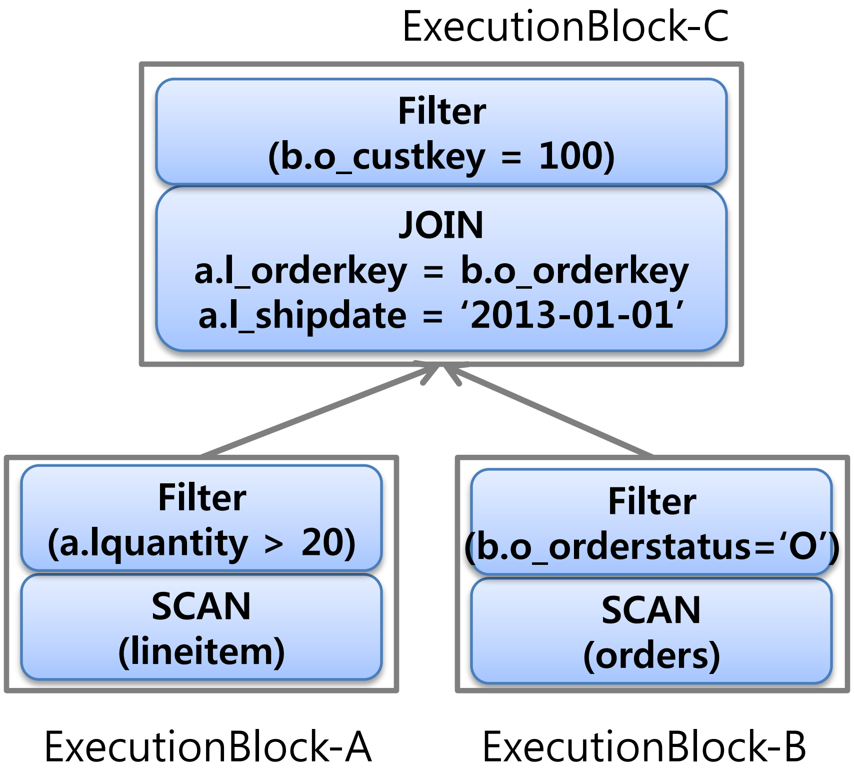
```
SELECT
    item_id,
    order_id,
    sum(price) * (3.6)
FROM
    ITEMS
GROUP BY
    item_id,
    order_id
WHERE item_id = 17234
```

Original **Rewritten**

Filter Push Down Rule(Outer Join)

	Preserved Row Table	Null Supplying Table
Join Predicate	Not Pushed(1)	Pushed(2)
Where Predicate	Pushed(3)	Not Pushed(4)

```
SELECT * FROM lineitem a
  OUTER JOIN orders b ON
    a.l_orderkey = o_orderkey AND
① a.l_shipdate = '2013-01-01' AND
② b.o_orderstatus = 'O'
WHERE a.l_quantity >20 ③
  AND b.o_custkey = 100 ④
```



ExecutionBlock-C

Filter
(b.o_custkey = 100)

JOIN
a.l_orderkey = b.o_orderkey
a.l_shipdate = '2013-01-01'

Filter
(a.lquantity > 20)

SCAN
(lineitem)

ExecutionBlock-A

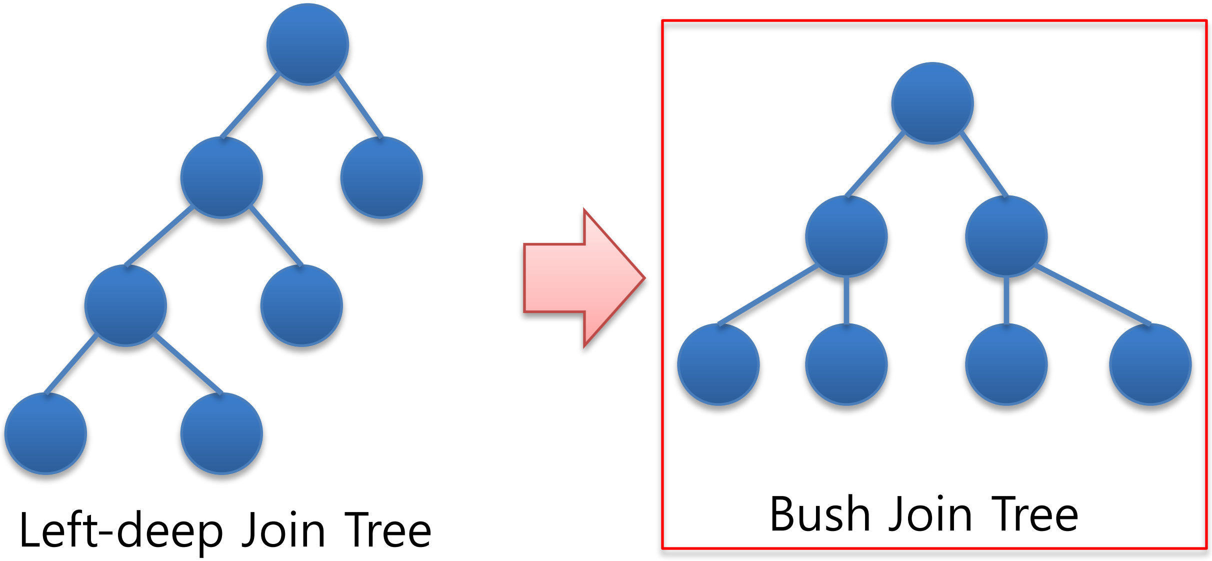
Filter
(b.o_orderstatus='O')

SCAN
(orders)

ExecutionBlock-B

Logical Plan Optimizer

- Cost-based Join Order
  - Don’t need to guess right join orders anymore
  - Greedy heuristic algorithm
    - Resulting in a bushy join tree instead of left-deep join tree



기타

- Union
  - Filter push down
- Repartitioner
  - tajo.dist-query.join.partition-volume-mb
  - tajo.dist-query.groupby.partition-volume-mb
  - tajo.dist-query.table-partition.task-volume-mb
- Broadcast Join
  - tajo.dist-query.join.broadcast.threshold-bytes
- Host/Disk aware scheduler

GRUTER: YOUR PARTNER IN THE BIG DATA REVOLUTION

Asian HQ

Phone	+82-70-8129-2950
Fax	+82-70-8129-2952

US HQ

Phone	+1-415-841-3345

E-mail	contact@gruter.com
Web	www.gruter.com