VectorH: Taking SQL-on-Hadoop to the Next Level

Weixi Li, Yang Yang

4.11.2019

Before we start……

- HDFS: Hadoop Distributed File System (handle large data sets running on commodity hardware)

- YARN: Yet Another Resource Negotiator (resource management and job scheduling)

- SQL-on-Hadoop: analytical application tools combine established SQL-style querying with Hadoop

VectorH

- a new SQL-on-Hadoop system with advanced query execution, updatability, YARN, HDFS and Spark integration.

The Problem

- Keep the vectorized processing while converting Vectorwise system (single-server) to VectorH (shared-nothing).

- Divide the hardware resources between concurrent parallel queries.

- Not viable to send a query to YARN (because the query should run on a thread that is part of the active server process).

From Vectorwise to VectorH

From Vectorwise to VectorH

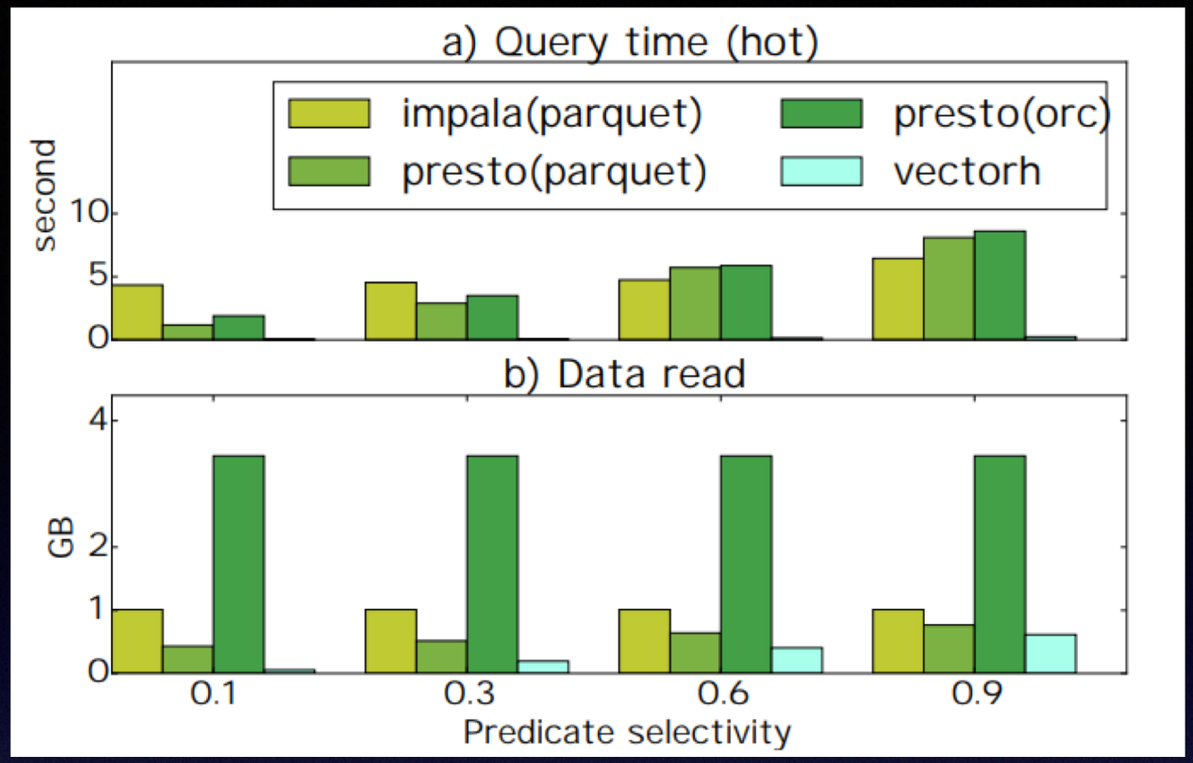
- Why choose Vectorwise system (Vectorized processing)?

From Vectorwise to VectorH

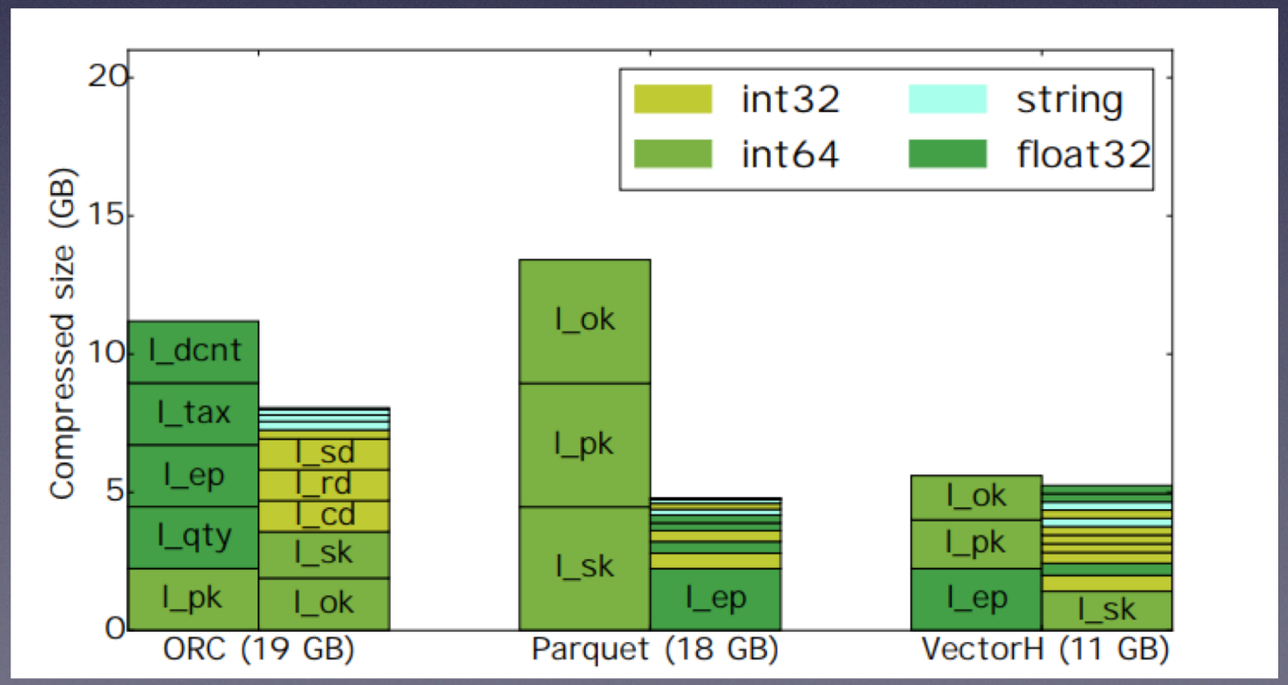
- Reducing query interpretation overhead.
- Increasing data and code CPU cache locality and allowing to use SIMD instructions.

Vectorwise Features

- Compression schemes PFOR, PFOR-DELTA and PDICT.
- Keep so-called MinMax statistics on all columns.



- SELECT max(l_linenumber) FROM lineitem WHERE l_shipdate<X

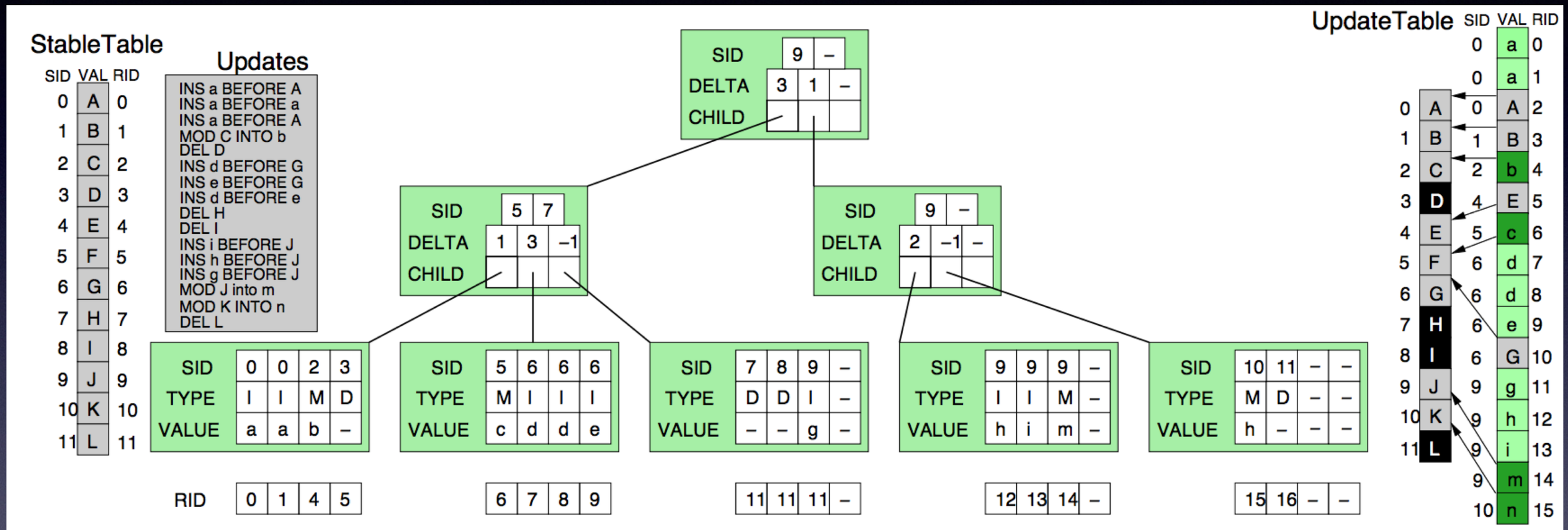


- the PFOR schemes used in VectorH compress better than both the ORC and Parquet formats do

Vectorwise Physical Design Options

- Co-ordered layout is like nested Hadoop data formats which is difficult to insert/delete in.
- Using a differential update mechanism based on Positional Delta Trees (PDT)

Positional Delta Trees



- Merging in differences from a PDT is fast because it identifies tuples by position, rather than by primary key

From Vectorwise to VectorH

- VectorH is a YARN-based cluster version of Vectorwise relying on HDFS for storage and fault-tolerance.
- VectorH integrates with HDFS and YARN

Storage Affinity With HDFS

Storage Affinity With HDFS

- Original Layout
- File-per-partition Layout
- Instrumenting HDFS Replication

Original Layout

- Writing is done in groups of consecutive blocks(4MB).
- Workloads consisting of consecutive updates can lead to significant wasted disk space.
- Appending to a table requires opening many files.

File-per-partition Layout

- All columns of a table partition are stored in the same file.
- Partially filled blocks would use as much space as full blocks.
- Partial blocks are written to a partial chunk file. A subsequent append merges these blocks with new data into new blocks and then frees the previous partial chunk file.

Instrumenting HDFS Replication

node1			node2			node3			node4		
R01	**R02**	**R03**	**R04**	**R05**	**R06**	**R07**	**R08**	**R09**	**R10**	**R11**	**R12**
S01	**S02**	**S03**	**S04**	**S05**	**S06**	**S07**	**S08**	**S09**	**S10**	**S11**	**S12**
R10a	R11a	R12a	R01a	R02a	R03a	R04a	R05a	R06a	R07a	R08a	R09a
S10a	S11a	S12a	S01a	S02a	S03a	S04a	S05a	S06a	S07a	S08a	S09a
R07b	R08b	R09b	R10b	R11b	R12b	R01b	R02b	R03b	R04b	R05b	R06b
S07b	S08b	S09b	S10b	S11b	S12b	S01b	S02b	S03b	S04b	S05b	S06b

after node4 failure:

R01a	R02a	**R03**	**R04**	**R05**	R06a	**R07**	**R08**	**R09**	
S01a	S02a	**S03**	**S04**	**S05**	S06a	**S07**	**S08**	**S09**	
R10	**R11**	**R12**	**R01**	**R02**	R03a	R04a	R05a	**R06**	
S10	**S11**	**S12**	**S01**	**S02**	S03a	S04a	S05a	**S06**	
R07b	R08b	R09b	R10b	R11b	R12b	R01b	R02b	R03b	
S07b	S08b	S09b	S10b	S11b	S12b	S01b	S02b	S03b	
R04b	R05b	R06b	R07a	R08a	R09a	R10a	R11a	R12a	*re-replicated*
S04b	S05b	S06b	S07a	S08a	S09a	S10a	S11a	S12a	*partitions*
node1			node2			node3			

- Partition Affinity Mapping for the 12 partitions of table R,S before (top) & after (bottom) node4 failure. Responsible partitions in bold; a/b are the second/third copy (R=3).

Elasticity With YARN

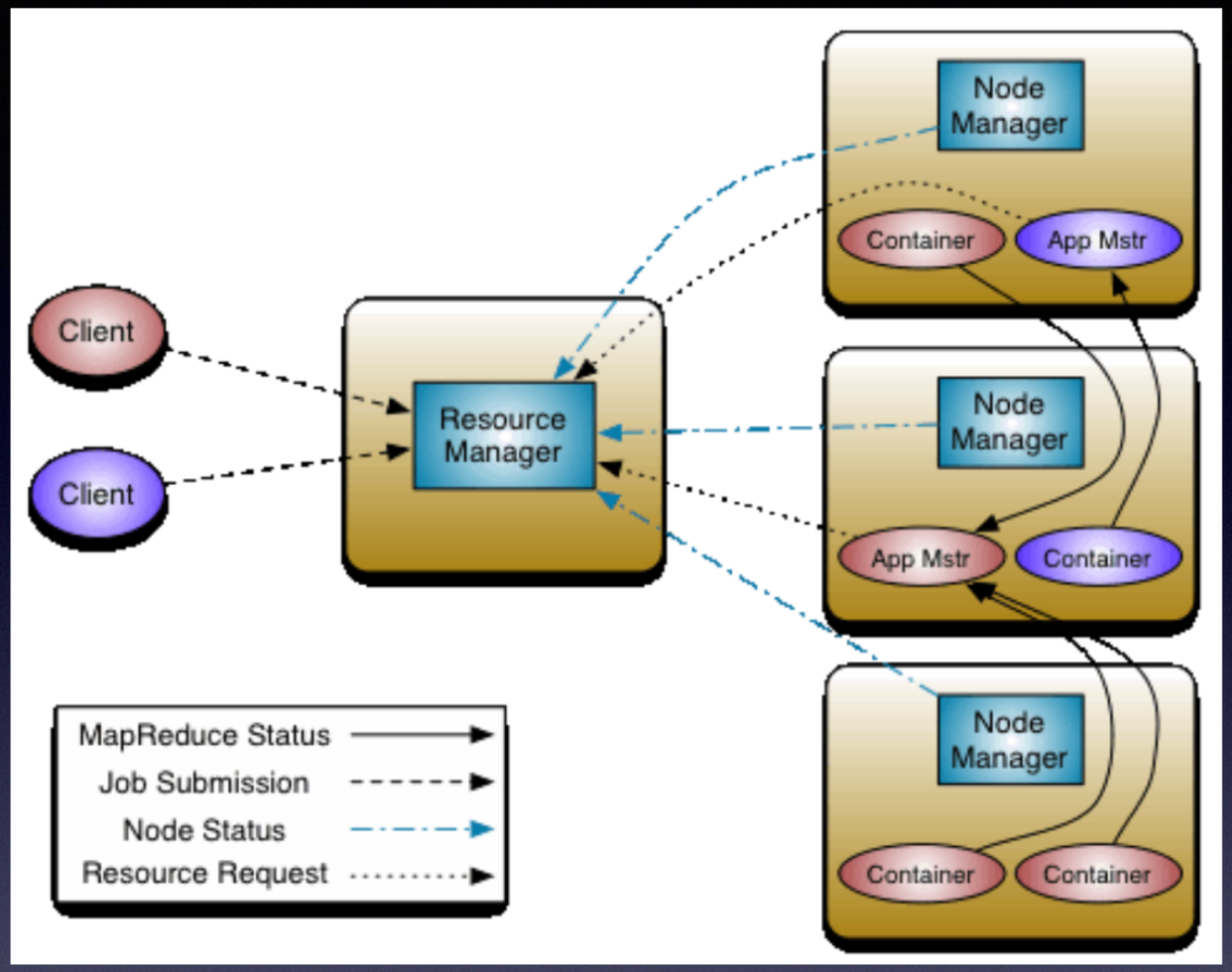
Elasticity With YARN

- Workers and Master
- Out-of-band YARN
- Min-cost Flow Network Algorithms
- Dynamic Resource Management

Workers and Master

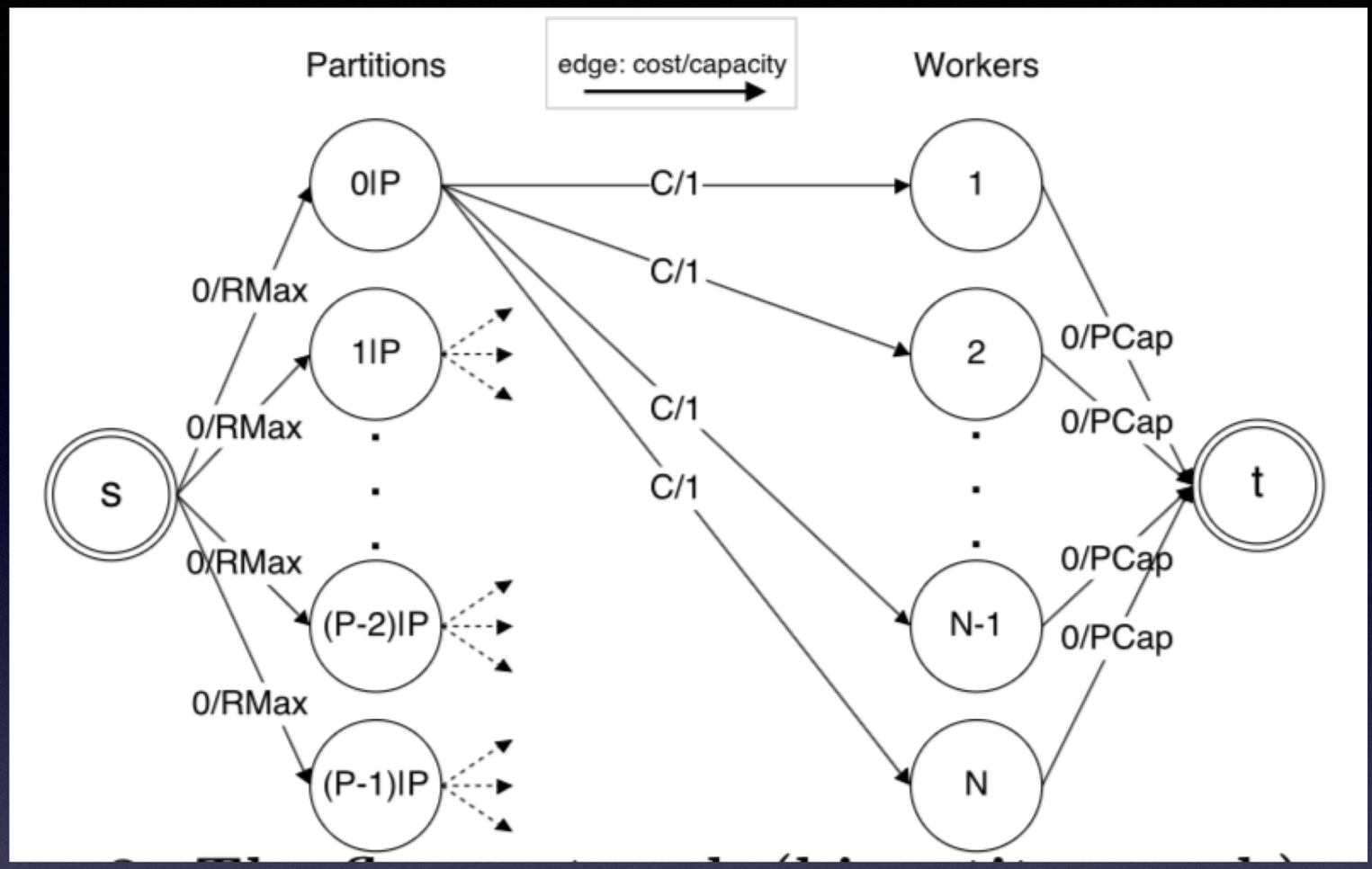
- Workers: A subset of the whole Hadoop clusters to run Vectorwise processes.
- Master: One of the workers becomes master.

Out-of-band YARN



- Why? Long running processes, latency, not allow to modify the resources of a container.
- out-of-band: Separate from its containers.
- How? VectorH runs a dbAgent process that acts as its YARN client.
- VectorH runs multiple AMs with corresponding containers on the worker set, each allocating a slice of its resources.

Min-cost Flow Network Algorithms



- When? VectorH initially starts, or when it recovers from node failures.
- Why? dbAgent must select N machines with most data locality.
- How? Min-cost Flow Network Algorithms.

Dynamic Resource Management

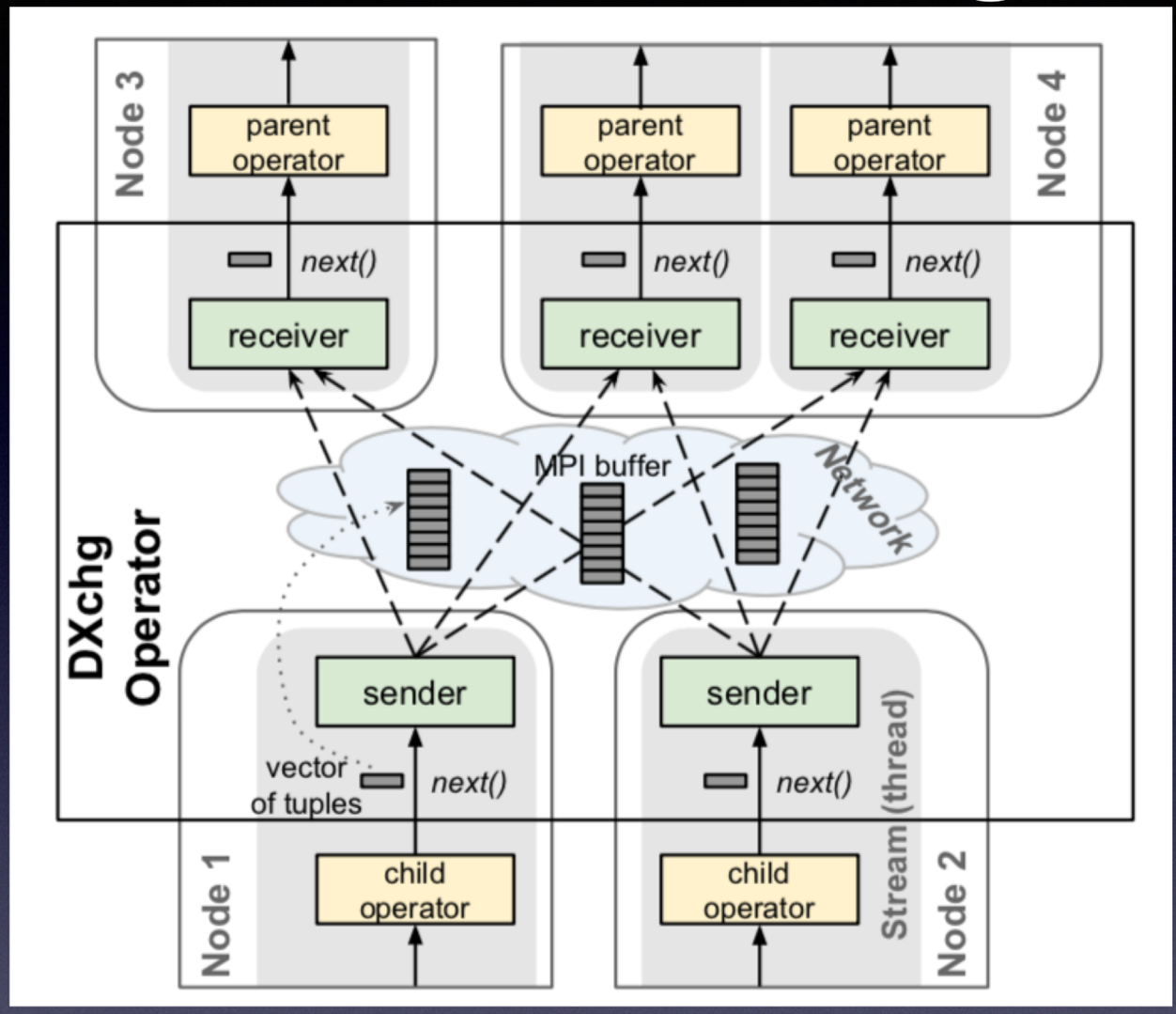
- VectorH will negotiate with YARN to get to its configured target of resources.
- VectorH will start as long as it gets above a configured minimum.
- VectorH will periodically negotiate with YARN to go back to its target resource footprint after its resources been occupied by higher-priority jobs.

Parallelism With MPI

Parallelism With MPI

- Exchange operators are the base of parallelism in VectorH.
- An Xchg operator only redistributes data streams.
- an Xchg operator acts as a synchronization point among a number of producer and consumer threads.

Distributed Exchange



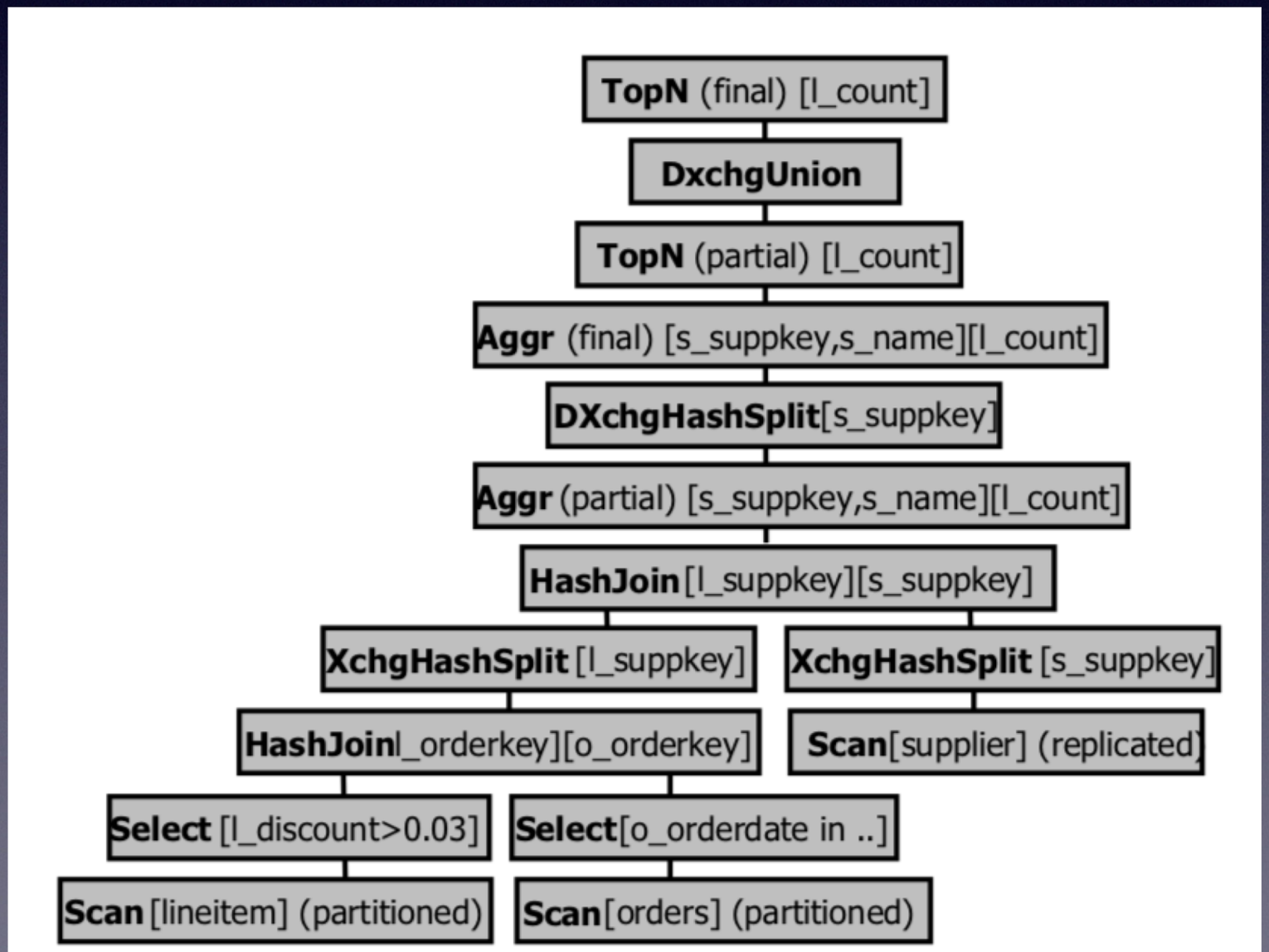
- Using MPI for network communication and intel MPI library.
- For better parallelism, implementing a thread-to-node approach.

Query Optimization

- Most query optimization changes were made in the Parallel Rewriter.
- Cost-based optimization using a dynamic programming algorithm.
- Cost model based on the cardinality estimates taken from the serial plan.

Detecting locality

```
SELECT FIRST 10 s_suppkey, s_name, count(*) as l_count
FROM   lineitem, orders, supplier
WHERE l_orderkey=o_orderkey AND l_suppkey=s_suppkey AND
      l_discount>0.03 AND
      o_orderdate BETWEEN '1995-03-05' AND '1997-03-05'
GROUP BY s_suppkey, s_name
ORDER BY l_count
```



TRANSACTION IN HADOOP

Vectorwise transaction management

- Updates are stored in a Positional Delta Tree (PDT)
- PDT can be stacked.
- while a transaction runs, it gathers changes

Distributed Transaction in VectorH

- Use table partition-specific WALs(Write-Ahead Log) instead of one single WAL.

- 2PC(2-Phase Commit)

VectorH - Log Shipping

VectorH-Update Propagation

VectorH - Referential Integrity

CONNECTIVITY WITH SPARK

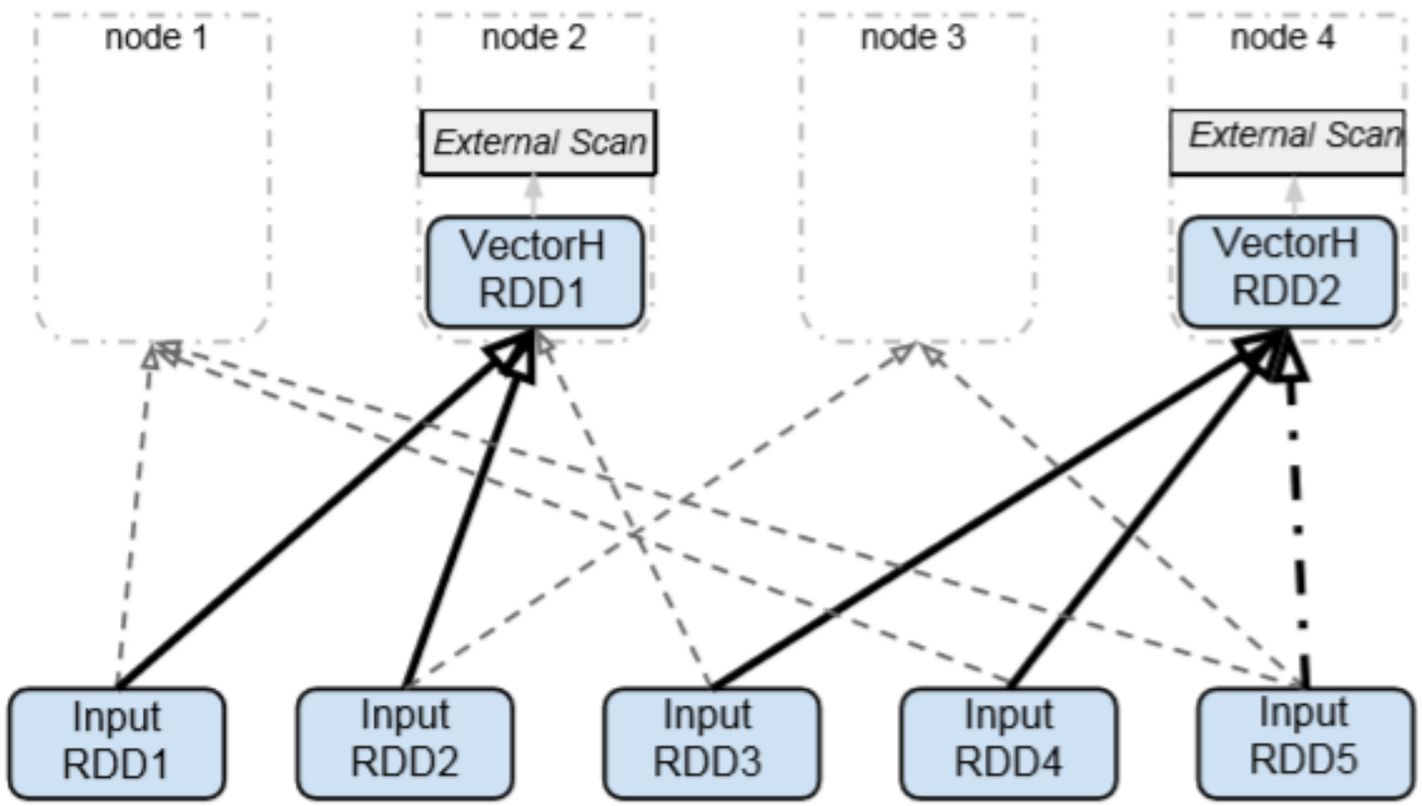
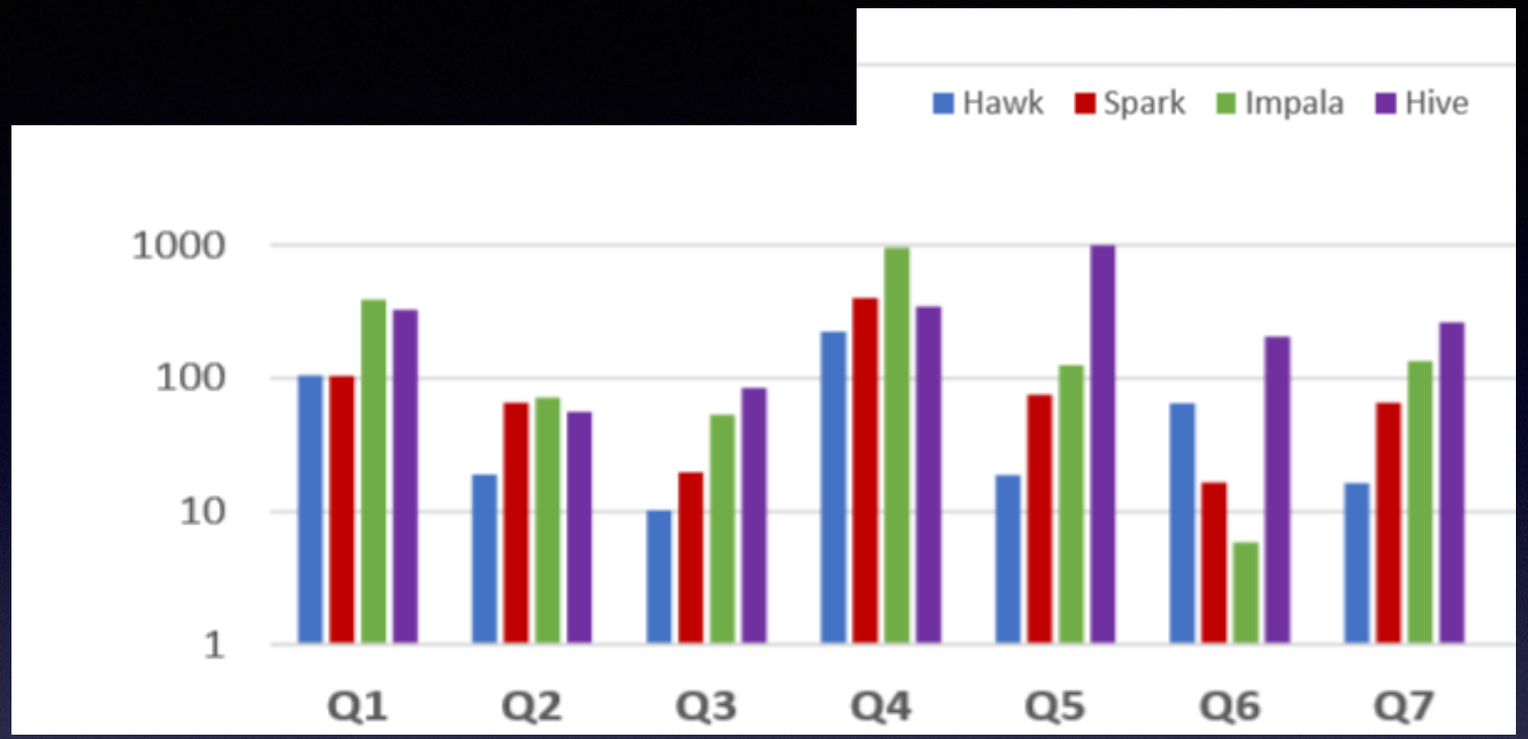


Figure 6: Mapping Spark RDDs to VectorH RDDs and ExternalScan operators

Evaluation


Hawk
Spark
Impala
Hive
1000
100
10
1
Q1
Q2
Q3
Q4
Q5
Q6
Q7

VectorH	**1.5**	**1.14**	**3.16**	**0.17**
HAWQ	158.2	21.46	32.06	38.21
SparkSQL	155.4	74.98	62.38	68.27
Impala	585.4	81.81	167.7	163.18
Hive	490.1	63.57	266.6	59.08

Does the paper prove its claims?

1. More like a design summary than an application of the scientific method

2. Design decisions are justified by comparison