



MapReduce and Parallel DBMSs: Friends or Foes?

Presented by Guozhang Wang

DB Lunch, May 3rd, 2010

Papers to Be Covered in This Talk

- *CACM'10*
 - MapReduce and Parallel DBMSs: Friends or Foes?
- *VLDB'09*
 - HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads
- *SIGMOD'08* (Pig), *VLDB'08*(SCOPE), *VLDB'09*(Hive)

Outline

- Architectural differences between MR and PDBMS (*CACM'10*)
 - Workload differences
 - System requirements
 - Performance benchmark results
- Integrate MR and PDBMS (*VLDB'09*)
 - Pig, SCOPE, Hive
 - HadoopDB
- Conclusions

Workload Differences

- Parallel DBMSs were introduced when
 - Structured data dominates
 - Regular aggregations, joins
 - Terabyte (today petabyte, 1000 nodes)
- MapReduce was introduced when
 - Unstructured data is common
 - Complex text mining, clustering, etc
 - Exabyte (100,000 nodes)



System Requirements:

From order of 1000 to 100,000

- Finer granularity runtime fault tolerance
 - Mean Time To Failure (MMTF)
 - Checkpointing
- Heterogeneity support over the cloud
 - Load Balancing

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
Transactional-level fault tolerance	Checkpointing intermediate results

Architectural Differences

Parallel DBMSs

*Jobs often need to restart
because of failures*

MapReduce

*Cannot pipeline query
operators*

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
Hash/range/round robin partitioning	Runtime scheduling based on blocks

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
<i>Execution time determined by slowest node</i>	<i>Cannot globally optimize execution plans</i>

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
<i>Execution time determined by slowest node</i>	<i>Cannot globally optimize execution plans</i>
Loading to tables before querying	External distributed file systems

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
<i>Execution time determined by slowest node</i>	<i>Cannot globally optimize execution plans</i>
<i>Awkward for semi-structured data</i>	<i>Cannot do indexing, compression, etc</i>

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
<i>Execution time determined by slowest node</i>	<i>Cannot globally optimize execution plans</i>
<i>Awkward for semi-structured data</i>	<i>Cannot do indexing, compression, etc</i>
SQL language	Dataflow programming models

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
<i>Jobs often need to restart because of failures</i>	<i>Cannot pipeline query operators</i>
<i>Execution time determined by slowest node</i>	<i>Cannot globally optimize execution plans</i>
<i>Awkward for semi-structured data</i>	<i>Cannot do indexing, compression, etc</i>
<i>Not suitable for unstructured data analysis</i>	<i>Too low-level, not reusable, not good for joins</i>

Least But Not Last ..

- Parallel DBMS
 - Expensive, no open-source option
- MapReduce
 - Hadoop
 - Attractive for modest budgets and requirements

Benchmark Study

- Tested Systems:
 - Hadoop (MapReduce)
 - Vertica (Column-store DBMS)
 - DBMS-X (Row-store DBMS)
- 100-node cluster at Wisconsin
- Tasks
 - Original MR Grep Task in OSDI'04 paper
 - Web Log Aggregation
 - Table Join with Aggregation

Benchmark Results Summary

2X



	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

Benchmark Results Summary

	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

4X



Benchmark Results Summary

	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

36X



Benchmark Results Summary

	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

- MR: parsing in runtime, no compression and pipelining, etc
- PDBMS: parsing while loading, compression, query plan optimization

Outline

- Architectural differences between MR and PDBMS (*CACM'10*)
 - Workload differences
 - System requirements
 - Performance benchmark results
- Integrate MR and PDBMS (*VLDB'09*)
 - Pig, SCOPE, Hive
 - HadoopDB
- Conclusions

We Want Features from Both Sides:

- Data Storage
 - From MR: semi-structured data loading/parsing
 - From DBMS: compression, indexing, etc
- Query Execution
 - From MR: load balancing, fault-tolerance
 - From DBMS: query plan optimization
- Query Interface
 - From MR: procedural
 - From DBMS: declarative

Pig

- Data Storage: MR
 - Run Pig Latin queries over any external files given user defined parsing functions
- Query Execution: MR
 - Compile to MapReduce plan and get executed on Hadoop
- Query Interface: MR+DBMS
 - Declarative spirit of SQL + procedural operators

SCOPE

- Data Storage: DBMS+MR
 - Load to Cosmos Storage System, which is append-only, distributed and replicated
- Query Execution: MR
 - Compile to Dryad data flow plan (DAG), and executed by the runtime job manager
- Query Interface: DBMS+MR
 - Resembles SQL with embedded C# expressions

Hive

- Data Storage: DBMS+MR
 - Use one HDFS dir to store one “table”, associated with builtin serialization format
 - Hive-Metastore
- Query Execution: MR
 - Compile to a DAG of map-reduce jobs, executed over Hadoop
- Query Interface: DBMS
 - SQL-like declarative *HiveQL*

So Far..

	Pig SIGMOD'08	SCOPE VLDB'08	Hive VLDB'09
Query Interface	Procedural Higher than MR	SQL-like + C#	HiveQL
Data Storage	External Files	Cosmos Storage	HDFS w/ Metastore
Query Execution	Hadoop	Dryad	Hadoop

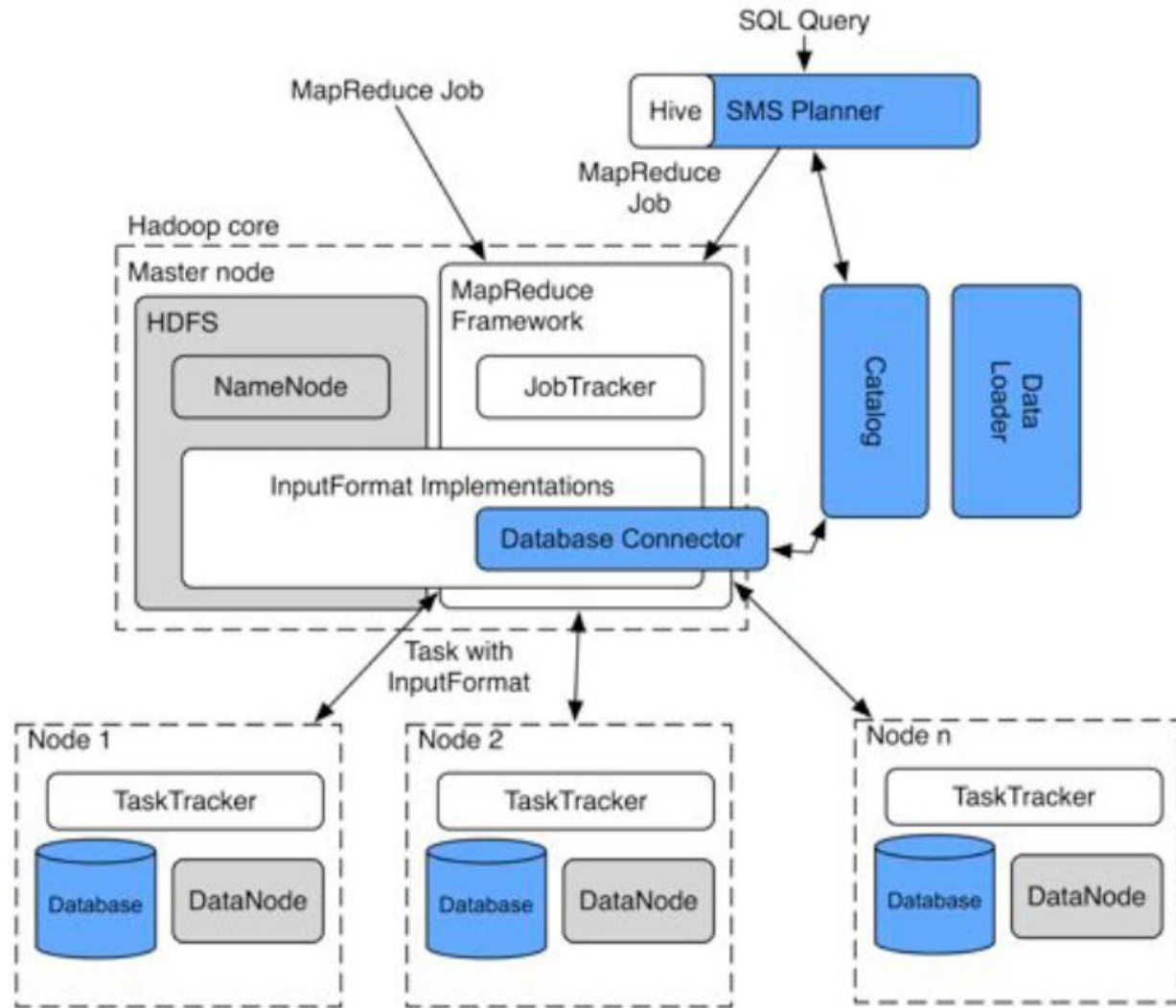
HadoopDB

	Pig SIGMOD'08	SCOPE VLDB'08	Hive VLDB'09	HadoopDB VLDB'09
Query Interface	Procedural Higher than MR	SQL-like + C#	HiveQL	SQL
Data Storage	External Files	Cosmos Storage	HDFS w/ Metastore	HDFS + DBMS
Query Execution	Hadoop	Dryad	Hadoop	As much DBMS as possible

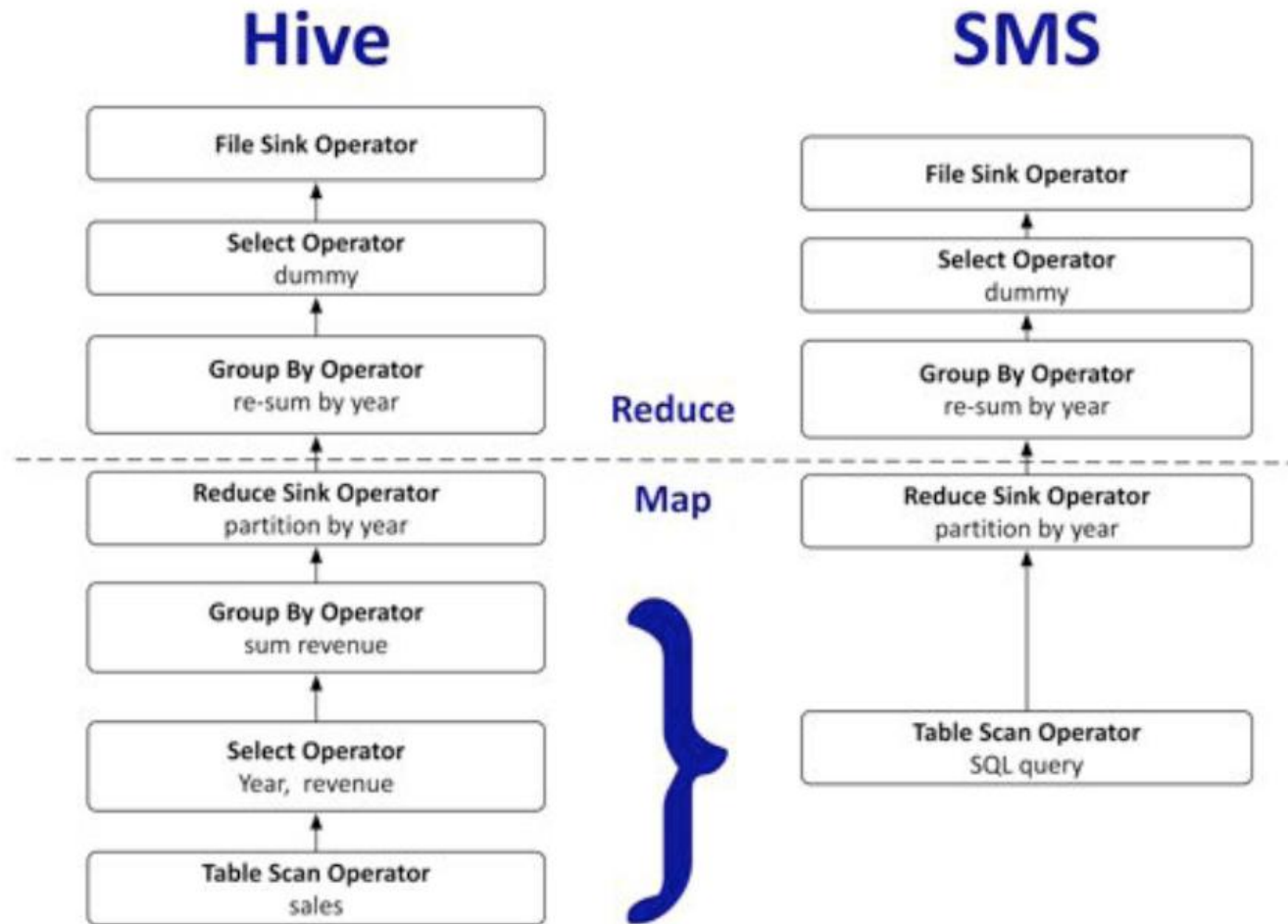
Basic Idea

- Multiple, independent single node databases coordinated by Hadoop
- SQL queries first compiled to MapReduce, then a sub-sequence of map-reduce converts back to SQL

Architecture

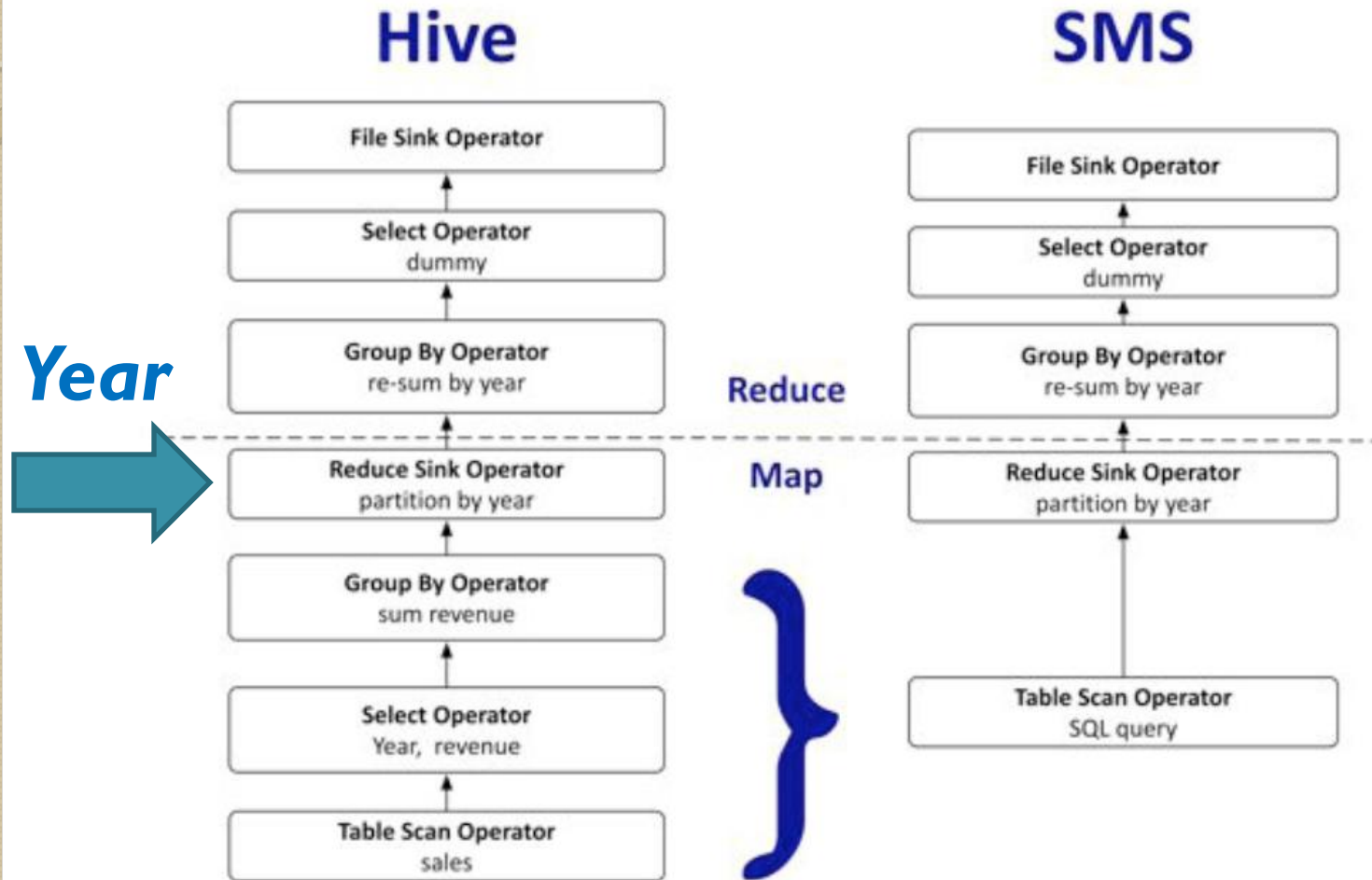


SQL – MR – SQL (SMS)



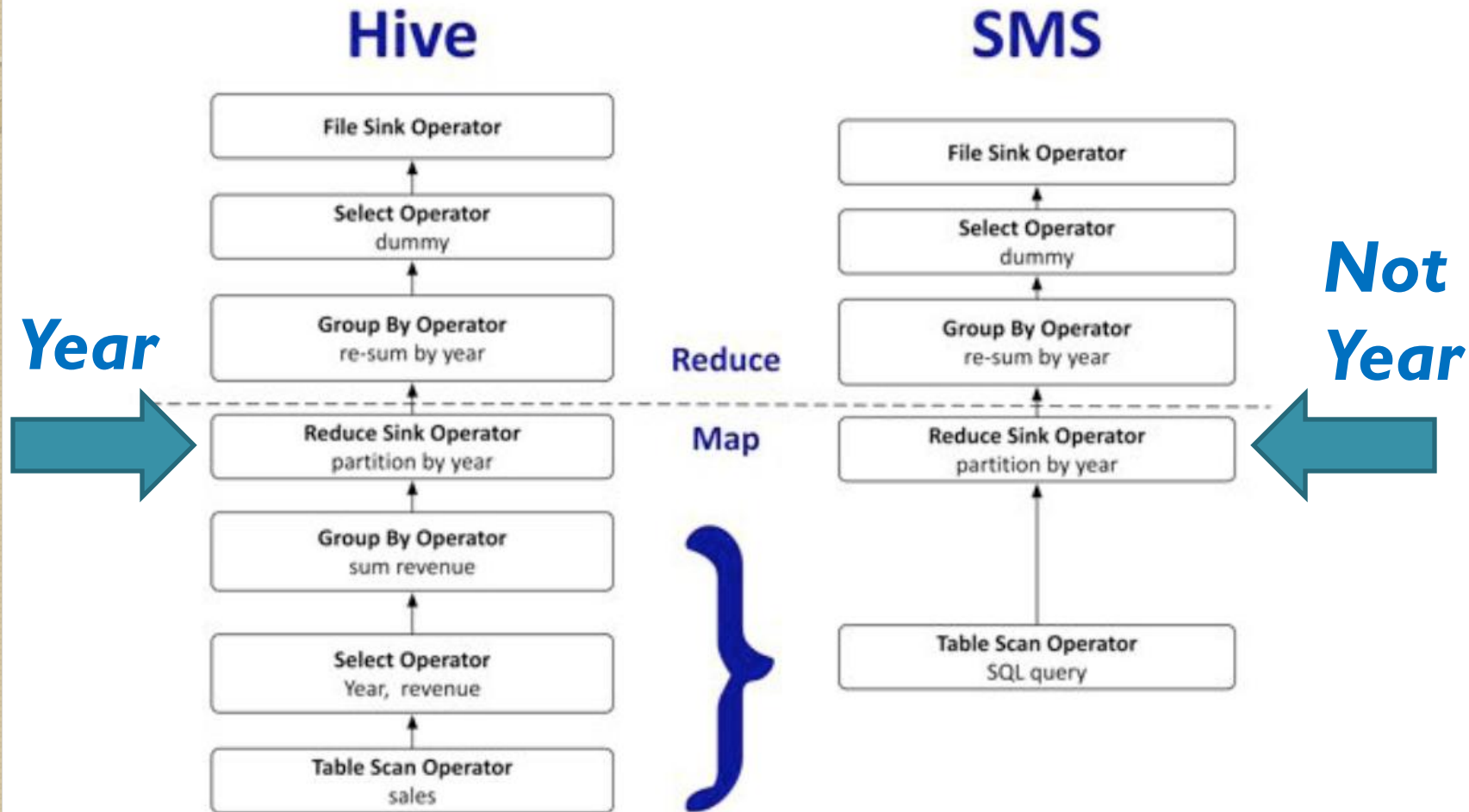
```
SELECT YEAR(saleDate), SUM(revenue) FROM sales GROUP BY YEAR(saleDate);
```

SQL – MR – SQL (SMS)



```
SELECT YEAR(saleDate), SUM(revenue) FROM sales GROUP BY YEAR(saleDate);
```

SQL – MR – SQL (SMS)

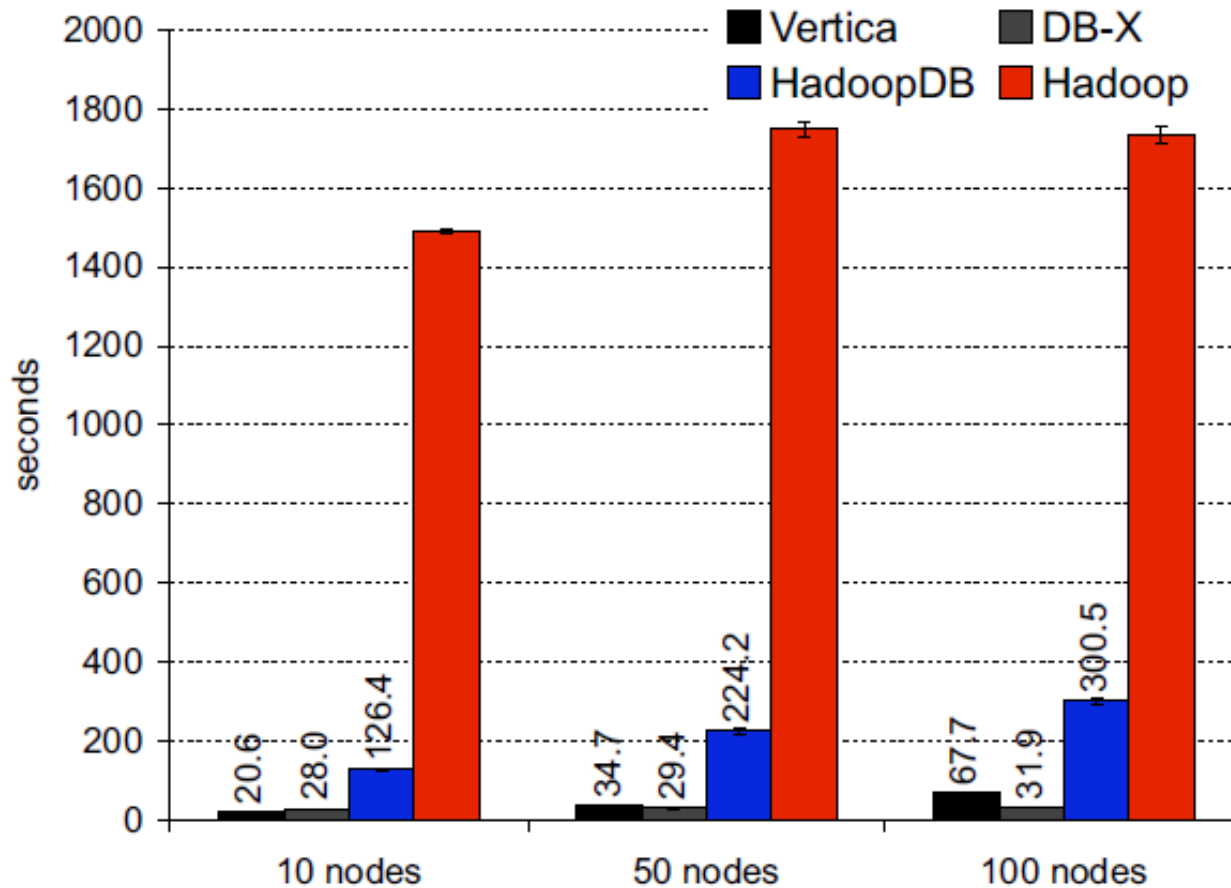


```
SELECT YEAR(saleDate), SUM(revenue) FROM sales GROUP BY YEAR(saleDate);
```

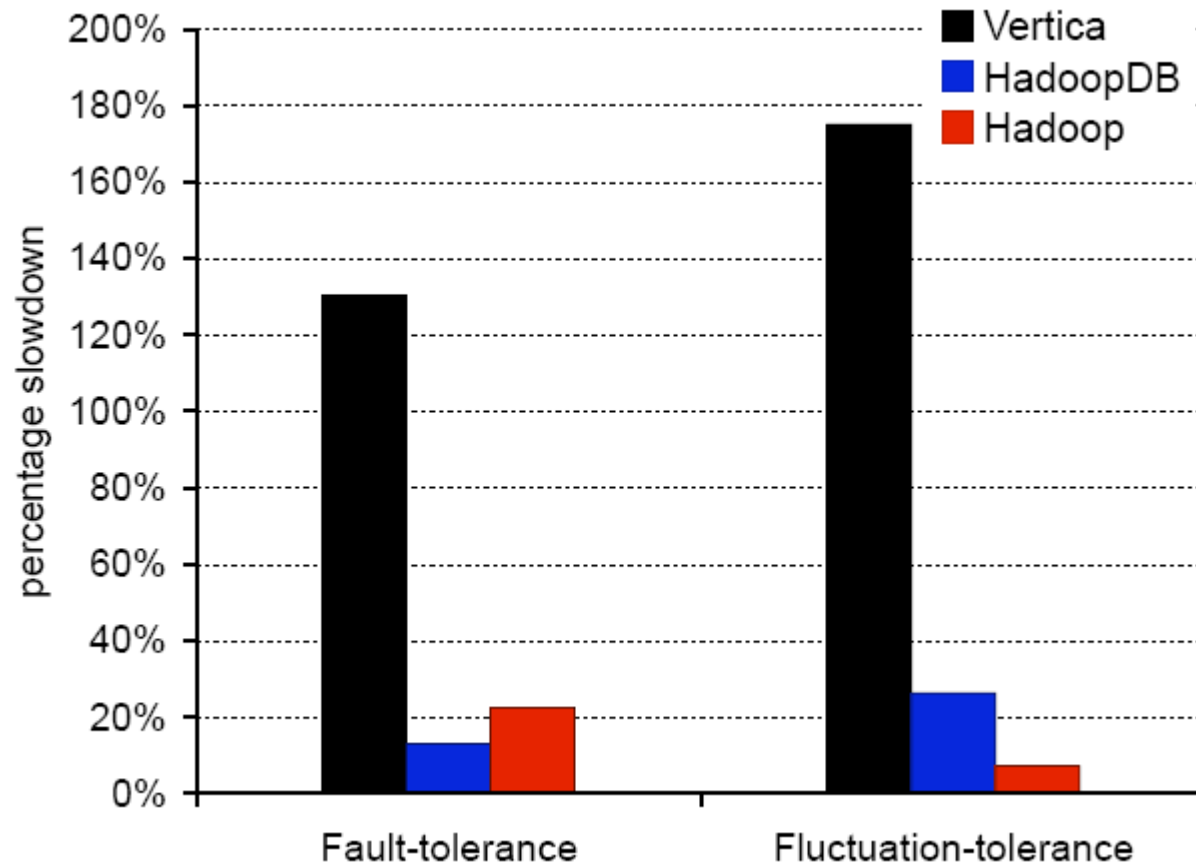
Evaluation Setup

- Tasks: Same as the *CACM'10* paper
- Amazon EC2 “large” instances
- For fault-tolerance: terminate a node at 50% completion
- For fluctuation-tolerance: slow down a node by running an I/O-intensive job

Performance: join task



Scalability: aggregation task



Conclusions

- Sacrificing performance is necessary for fault tolerance/heterogeneity in the case of order 100,000 nodes
- MapReduce and Parallel DBMSs completes each other for large scale analytical workloads.

Conclusions

- Sacrificing performance is necessary for fault tolerance/heterogeneity in the case of order 100,000 nodes
- MapReduce and Parallel DBMSs completes each other for large scale analytical workloads.


Questions?

Other MR+DBMS Work

(part of the slide from Andrew Pavlo)

- Commercial MR Integrations
 - Vertica
 - Greenplum
 - AsterData
 - Sybase IQ
- Research
 - MRi (Wisconsin)
 - Osprey (MIT)


Benchmark Results Summary



	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

- MR: Record parsing in run time
- PDBMS: Record parsed/compressed when loaded


Benchmark Results Summary



	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

- MR: Write intermediate results to disks
- PDBMS: Pipelining

Benchmark Results Summary



	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

- MR: Cannot handle joins very efficiently
- PDBMS: Optimization for joins

Benchmark Results Summary

	Hadoop	DBMS-X	Vertica
Grep Task	284 sec	194 sec	108 sec
Web Log	1146 sec	740 sec	268 sec
Join	1158 sec	32 sec	55 sec

Summary:

- Trade performance to have runtime scheduling and checkpointing
- Trade execution time to reduce load time at storage layer

Architectural Differences

<i>Parallel DBMSs</i>	<i>MapReduce</i>
Transactional-level fault tolerance	Checkpointing intermediate results
Hash/range/round robin partitioning	Runtime scheduling based on blocks
Loading to tables before querying	External distributed file systems
SQL language	Dataflow programming models