HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads

Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, A. Rasin
Yale University
VLDB 2009

Presented by:
Anup Kumar Chalamalla

Outline

- Context: Analytical DBMS Systems
- Background: Parallel Databases and Query Processing
- Key Properties for Very Large Scale Data Analytics
- Architecture of HadoopDB
- Performance and Scalability Results

Context: Analytical DBMS Systems

- Multi-dimensional structured data
  - Star schema: Fact tables and dimension tables
- Types of queries
  - TableScan, Joins, multi-dimensional aggregation (CUBE), Pattern Mining, Top-K and ranking
- Data explosion in terabytes and petabytes

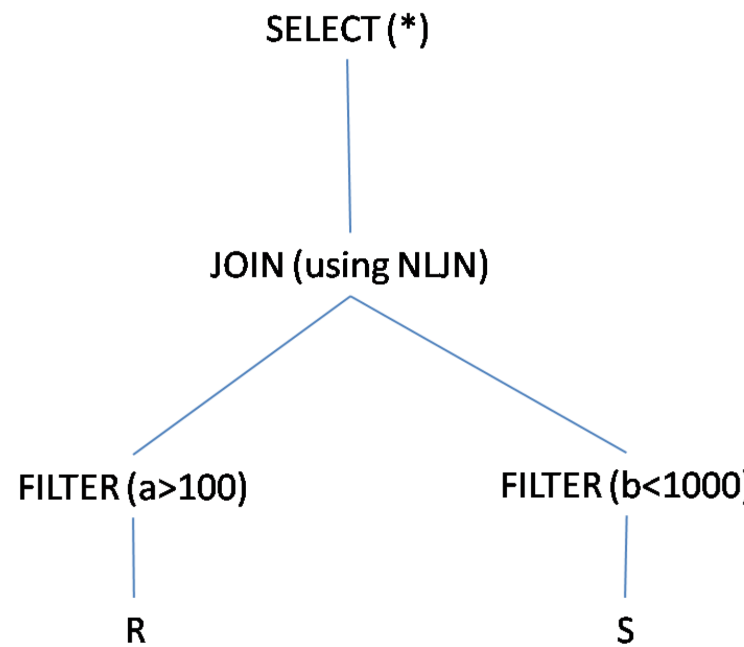
Background: Parallel Databases

- DBMSs deployed on a shared nothing architecture
- Query execution is divided equally among all machines
- Results are computed on different machines and transferred over the network
- Important tasks:
  - Partitioning the tables on to several machines
  - Parallel evaluation of relational query operators

Background: Query Processing

- SELECT *
  FROM R CROSS JOIN S
  WHERE R.a > 100 AND
  S.b < 1000

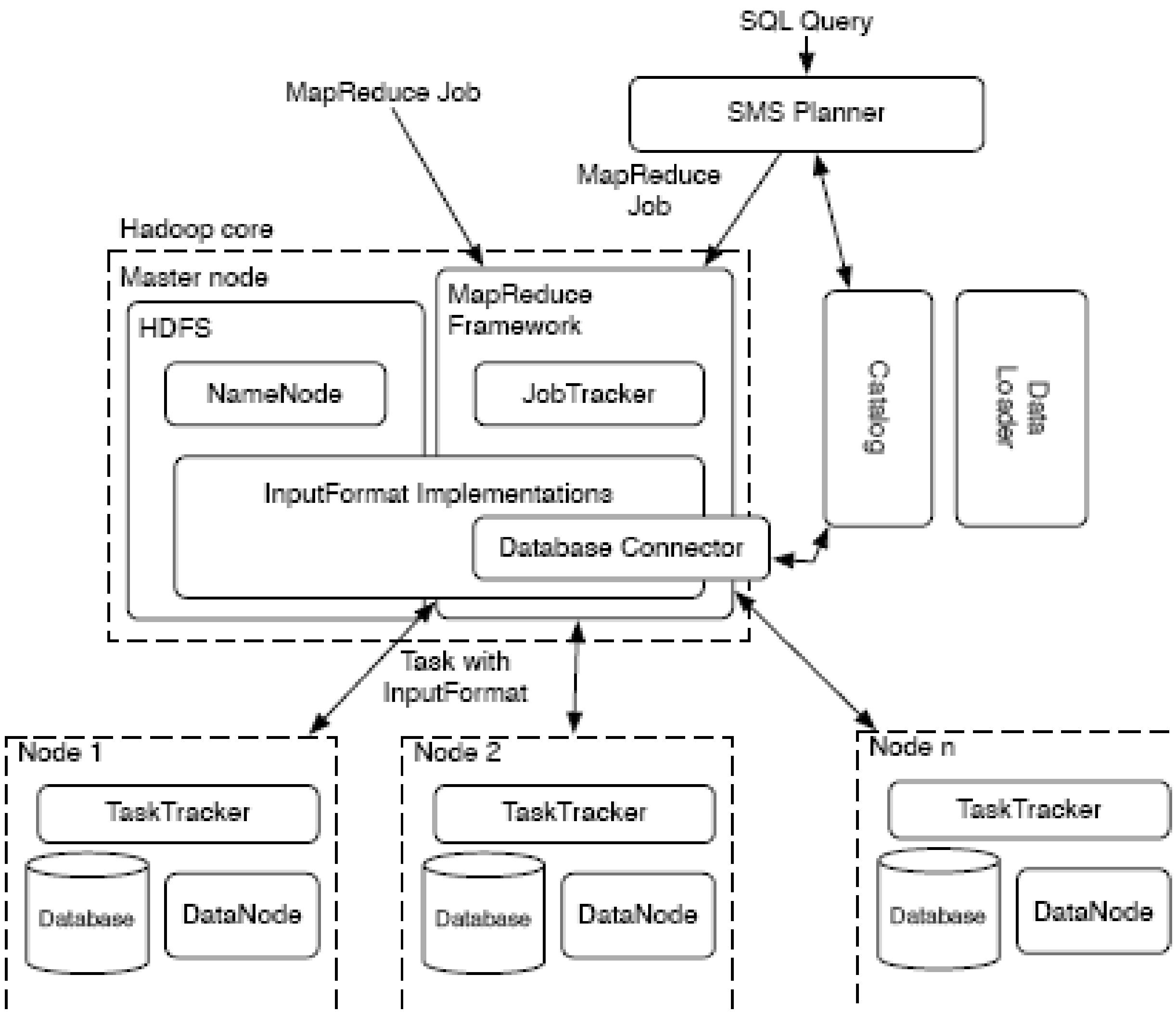
- Pipelining: Transfer intermediate results of one operator to another operator on the fly



Key properties for very large scale data analytics

- Performance: Computing the results of a query faster
- Fault Tolerance: Rescheduling parts of query execution in the case of node failures
- Adapt to heterogeneous distributed environment: Getting the same performance from all the machines is difficult
- Flexible Query interface: Should support ODBC/JDBC and user defined functions

Architecture of HadoopDB



Data Loader

- All data initially resides on the HDFS; table data is stored as raw files
- Tables are partitioned (on-demand) and partitions are loaded on to the nodes' file systems
- Data that comes at each node is re-partitioned in to small chunks
- From there it is bulk-loaded in to the DBMS and indexed if required
- Hash Partitioning :
  - Global Hasher: Partition the tables which are stored as raw files on HDFS and distribute them
  - Local Hasher: Partition the single-node data in to file chunks and store them in to disk blocks for efficient processing

Catalog

- Metadata about tables and their partitions:
  - Attribute on which partition of a table exists in the cluster
  - Size and location of the blocks of a partition on a particular node
  - Replicas, if replicas exist for the partitions
- For each node store the DBMS connection details
  - IP Address, Driver class, username and password, database name, etc.
- MetaStore: Table schema information on the DBMSs in the nodes. Used by SMS Planner for query plan generation

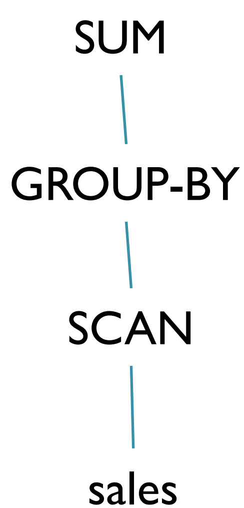
SMS Planner

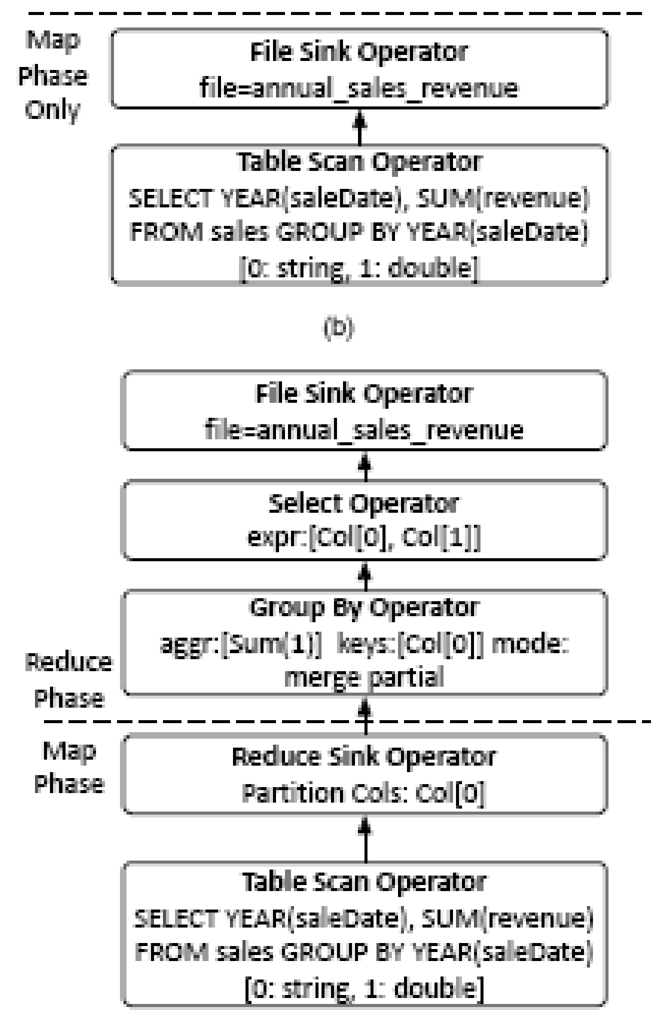
- Extends Hive, an SQL query processor built on top of Hadoop
- Parses the SQL Query, and transforms it in to an operator DAG or the logical plan
- Generates an optimal query plan after doing any transformations
- It breaks up the plan in to a batch of map and reduce functions
- Checks if a partitioning of a table exists on the join or group-by attributes and decides on map and reduce functions

SMS Planner on an example query

- SELECT YEAR(saleDate), SUM(revenue)

  FROM sales GROUP BY YEAR(saleDate);





SMS Planner and Hadoop Jobs

- SMS Planner generates map or reduce functions that encapsulate code about database connection and SQL query to execute

- A DatabaseConnector object is created by a Map function to connect to the database using JDBC and execute SQL query

- Assuming tables are loaded in the database, an execution of a map function triggers a database connection, query execution and transforming the ResultSet in to key value pairs

- Reduce function simply aggregates over the repartitioned tuples and produces output to the files

Salient Features of HadoopDB

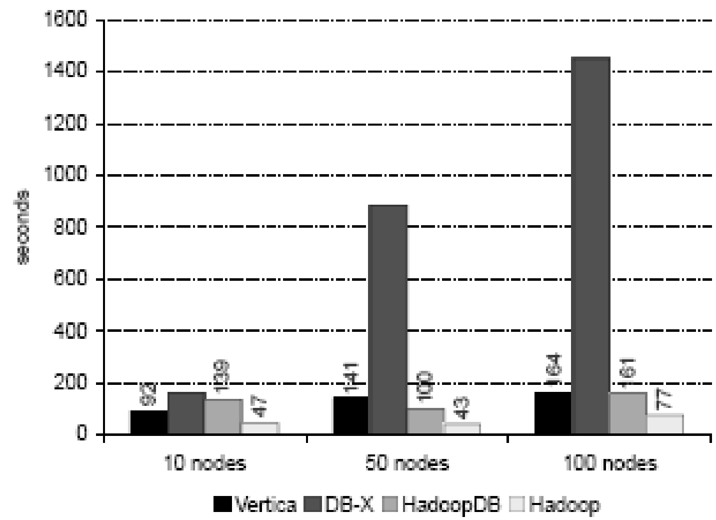
- Hadoop is used :
  - To store the data using the HDFS file system
  - For task scheduling, Hadoop's JobTracker is used to schedule Map and Reduce tasks on the nodes
  - As network communication layer to transfer the intermediate results of SQL query computations between nodes
- An SQL Query is initially broken down in to a batch of MapReduce jobs and then scheduled using Hadoop
- Ultimately execution of relational query operators happens in a single node DBMS
- Queries are embedded in map and reduce functions and executed
- Results are returned as key value pairs after query execution

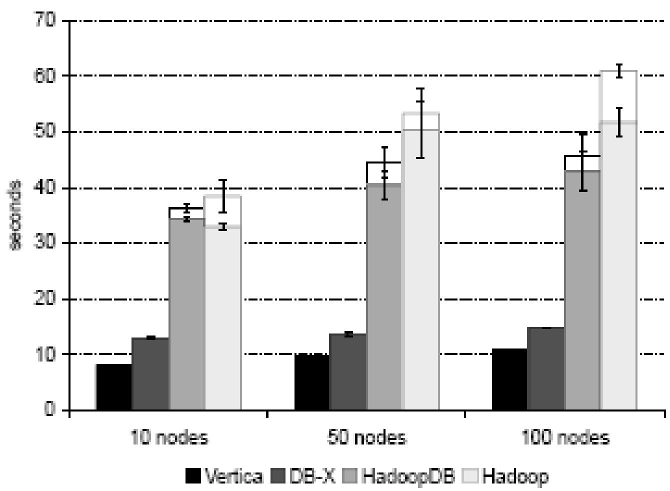
Performance and Scalability Benchmark

- Architectures compared:
  - Hadoop
  - HadoopDB
  - Vertica
  - DBMS-X
- Tasks evaluated in the benchmark:
  - Grep
  - Selection (Filtering)
  - Aggregation
  - Join
  - UDF Aggregation

Grep Task

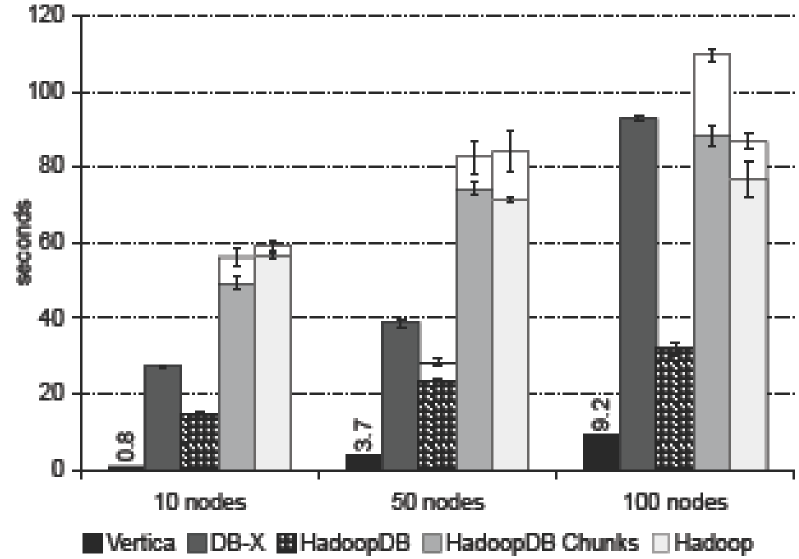
- Data consists of 5.6 million100-byte records per node
- For Hadoop, a map function that performs a simple string match over records stored in a file, one per line
- Vertica, DBMS-X, HadoopDB execute the query:
  - SELECT * FROM Data WHERE field LIKE '%XYZ%';
- HadoopDB performs better than Hadoop because it saves on I/O





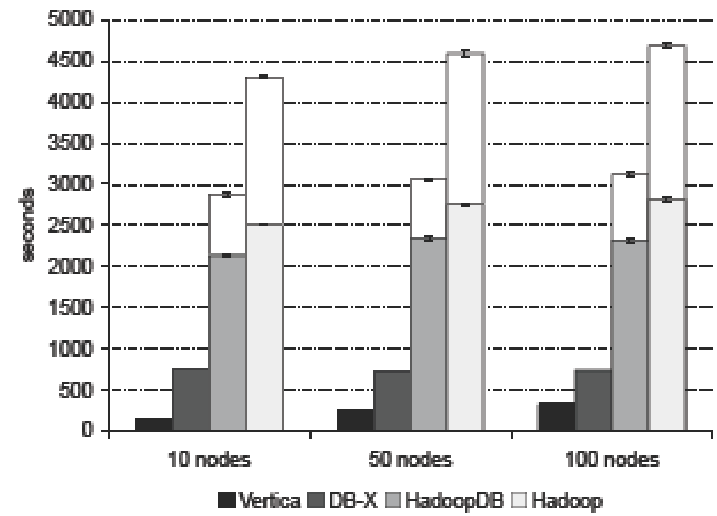
Selection Query

- SELECT pageURL, pageRank FROM Rankings WHERE pageRank > 10;
- Hadoop as usual parses the data files and filters records
- HadoopDB pushes the execution of selection and projection operators in to the PostgreSQL
- Using clustered indices boosts performance of parallel databases and HadoopDB over Hadoop



Aggregation Query

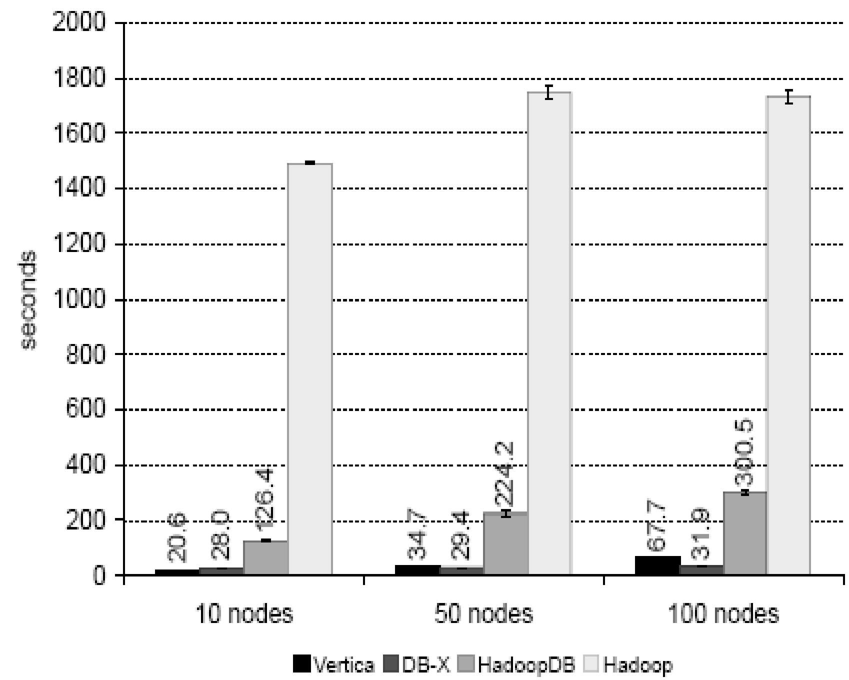
- SELECT sourceIP, SUM(adRevenue) FROM UserVisits GROUP BY sourceIP;
- There is a map and a reduce phase in these queries
- HadoopDB pushes the SQL operators' execution in to the PostGreSQL
- Using Hive's query optimizer helps in choosing either sorting or hashing method to perform aggregation



Join Queries

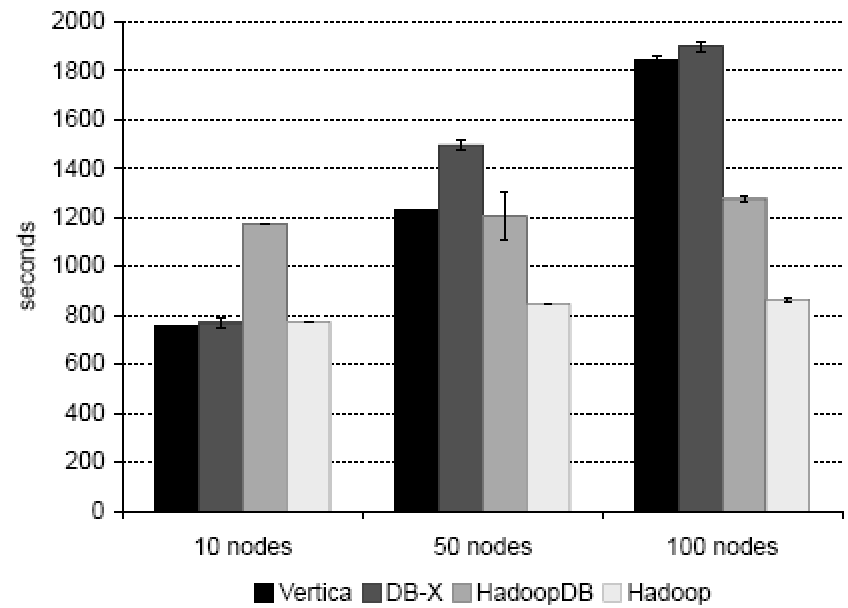
- Hadoop supports a sort-merge kind of algorithm but incurs sorting overhead

- HadoopDB assumes a collocation of tables partitioned on the join attributes

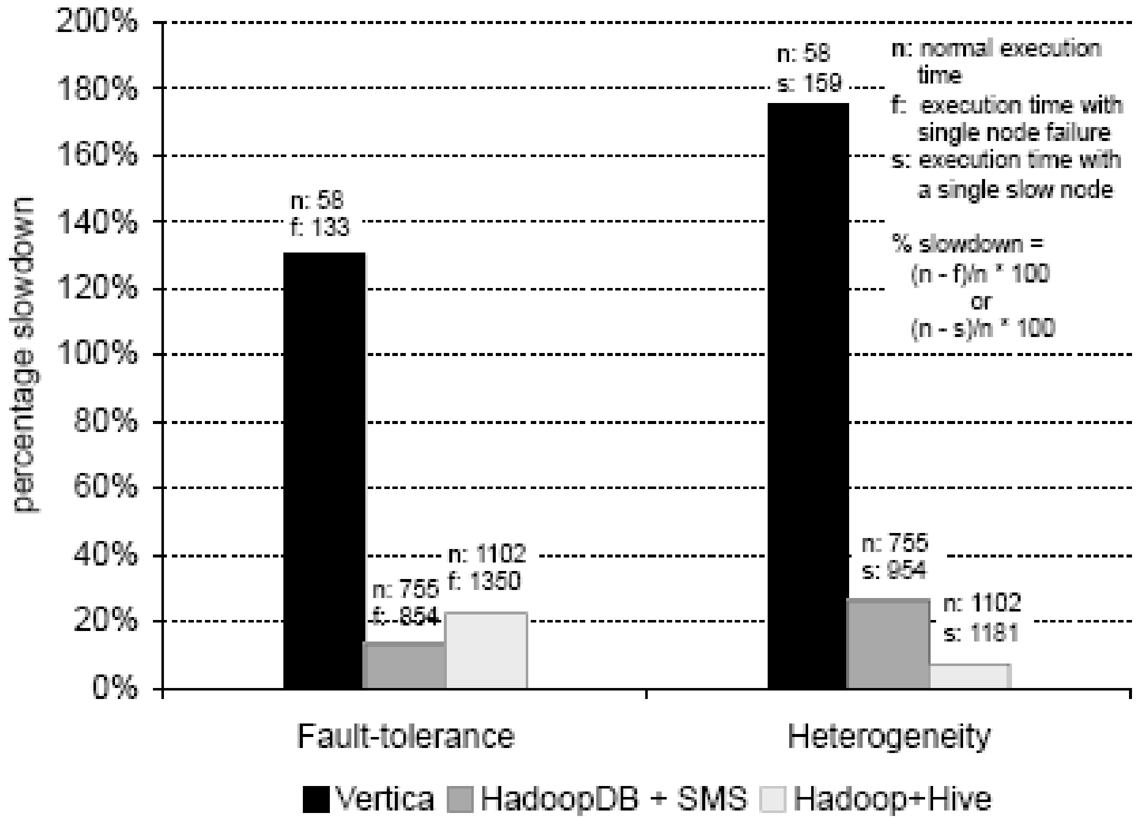


UDF Aggregation Task

- HTML Documents are processed for counting number of out-links
- In parallel DBMS a user defined function accesses chunks of HTML documents and parses them in memory
- Outputs results of chunks on to a temporary table which are later aggregated
- Hadoop and HadoopDB executes the same and Map and Reduce code



Fault Tolerance and Heterogeneity



Conclusions

- HADOOPDB
- Fault Tolerance: In the presence of node failures, Hadoop reschedules the tasks and completes the query
- Hadoop redundantly executes tasks of straggler nodes thus reducing effect of slow nodes on query time
- PostgreSQL is not a column-store and hence a drawback for HadoopDB
- In the event of data explosion and using several hundreds of nodes scalability comes in to picture

- PARALLEL DATABASES
- In case of node failures unfinished queries are aborted and query processing is restarted
- There is no way to counter the slow node's effect on overall query time
- Parallel databases like Vertica achieve much better performance due to column store and data compression
- Parallel databases are not scalable