Hadoop's Entry into the Traditional Analytical DBMS Market

Daniel Abadi
Yale University
August 3rd, 2010

Data, Data, Everywhere

- Data explosion
  - Web 2.0 → more user data
  - More devices that sense data
  - More equipment that produce data at extraordinary rates (e.g. high throughput sequencing)
  - More interactions being tracked (e.g. clickstream data)
  - More business processes are being digitized
  - More history being kept
- Date becoming core to decision making, operational activites, and scientific process
  - Want raw data (not aggregated version)
  - Want to run complex, ad-hoc analytics (in addition to reporting)

Consequences of Scale

- Increasing desire for incremental scale out on commodity hardware (1000s of nodes)
  - Fault tolerance a bigger concern
  - Dealing with unpredictable performance a bigger concern
  - Cost becoming a bigger concern
- Need to bring computation to data, not vice versa
- Hadoop outperforms traditional analytical database systems (Teradata, Oracle Exadata, IBM, Netezza, Vertica, Greenplum, Aster Data, soon Microsoft, etc.) on each of the above 4 dimensions

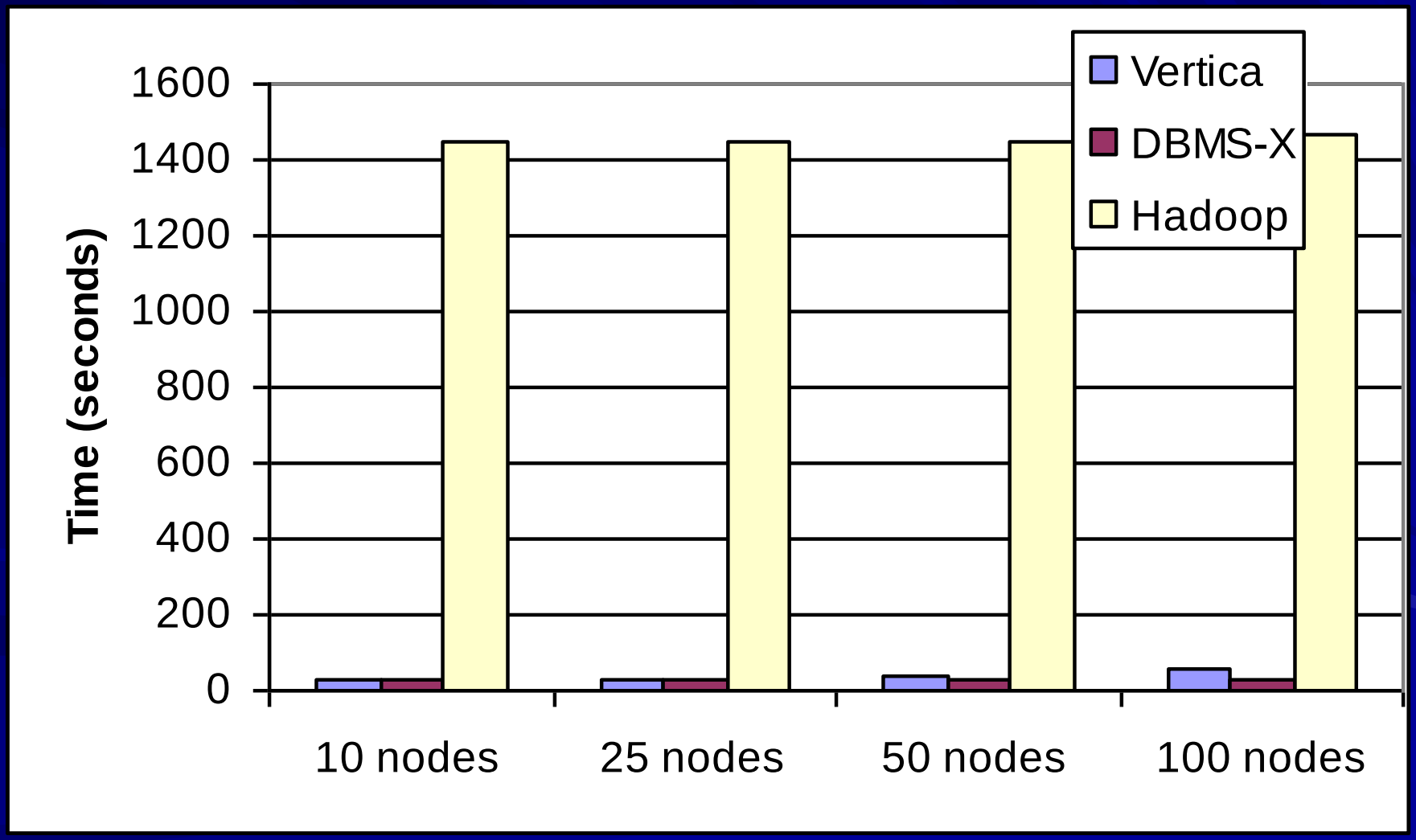
SIGMOD 2009 Paper

- Benchmarked Hadoop vs. 2 parallel database systems
  - Mostly focused on performance differences
  - Measured differences in load and query time for some common data processing tasks
  - Used Web analytics benchmark whose goal was to be representative of tasks that:
    - Both should excel at
    - Hadoop should excel at
    - Databases should excel at

Hardware Setup

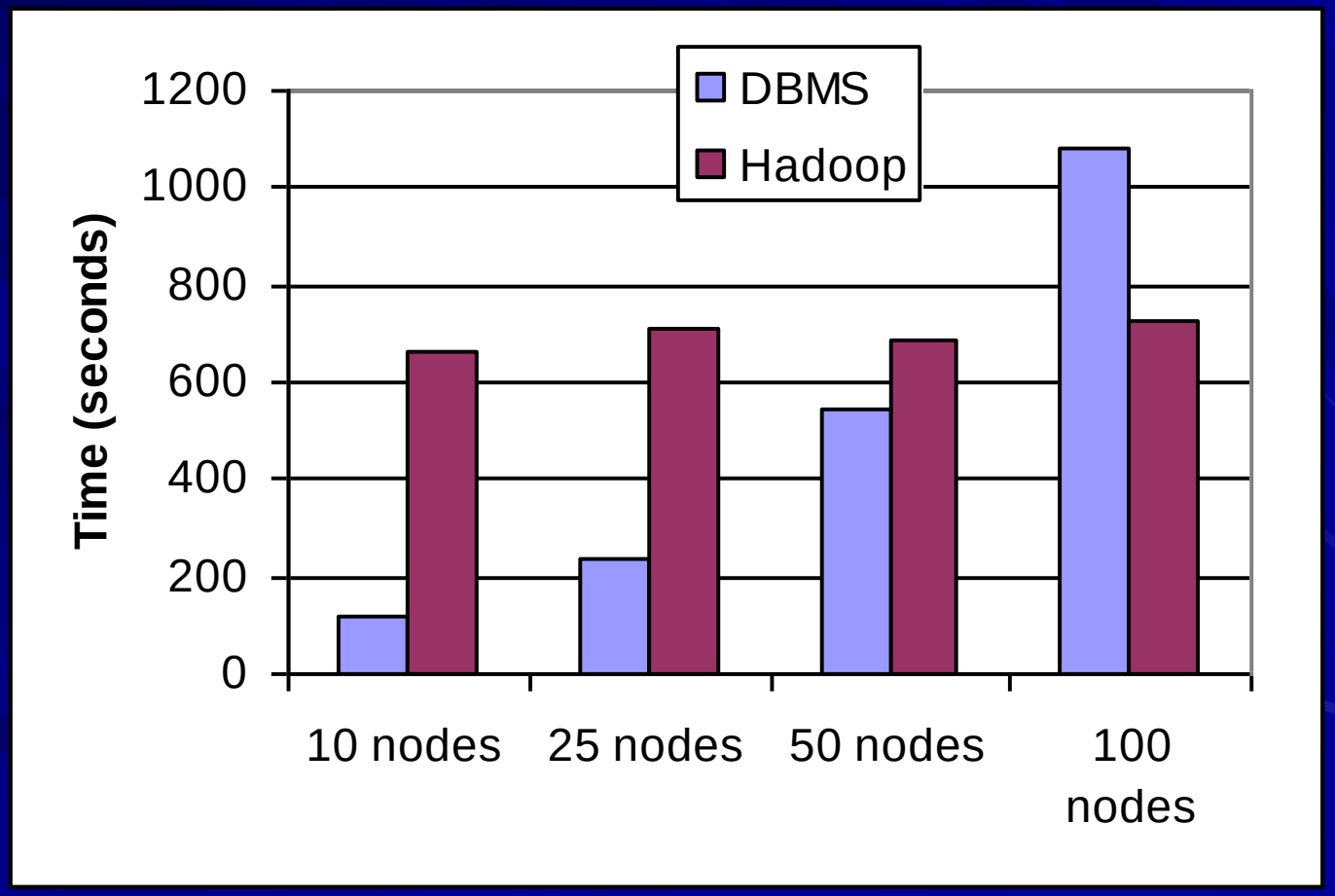
- 100 node cluster
- Each node
  - 2.4 GHz Code 2 Duo Processors
  - 4 GB RAM
  - 2 250 GB SATA HDs (74 MB/Sec sequential I/O)
- Dual GigE switches, each with 50 nodes
  - 128 Gbit/sec fabric
- Connected by a 64 Gbit/sec ring

Join Task


Vertica
DBMS-X
Hadoop
Time (seconds)
1600
1400
1200
1000
800
600
400
200
0
10 nodes
25 nodes
50 nodes
100 nodes

UDF Task

- Calculate PageRank over a set of HTML documents
- Performed via a UDF



Fault Tolerance and Cluster Heterogeneity Results

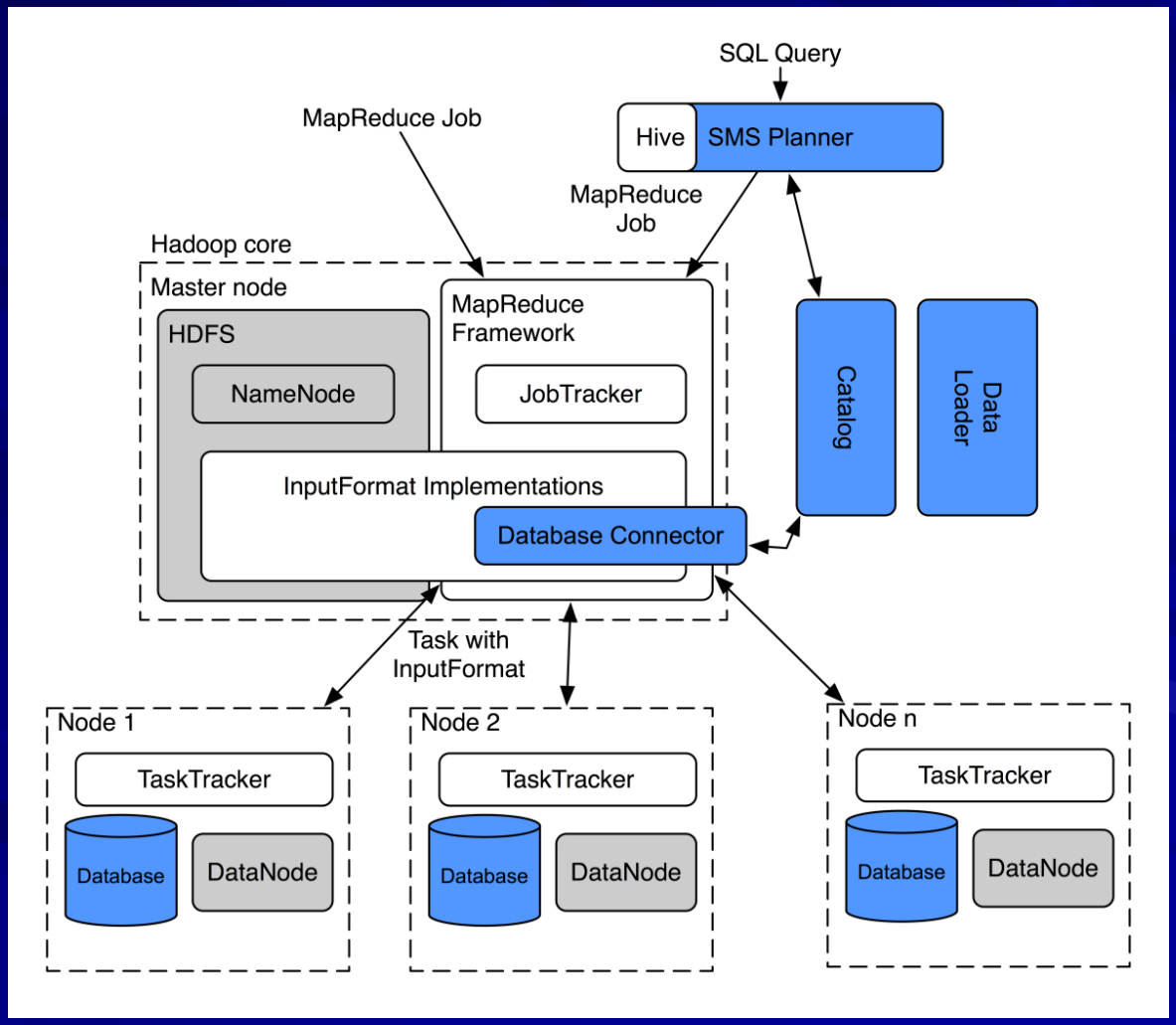
Benchmark Conclusions

- Hadoop is consistently more scalable
  - Checkpointing allows for better fault tolerance
  - Runtime scheduling allows for better tolerance of unexpectedly slow nodes
  - Better parallelization of UDFs
- Hadoop is consistently less efficient for structured, relational data
  - Reasons both fundamental and non-fundamental
  - Needs better support for compression and direct operation on compressed data
  - Needs better support for indexing
  - Needs better support for co-partitioning of datasets

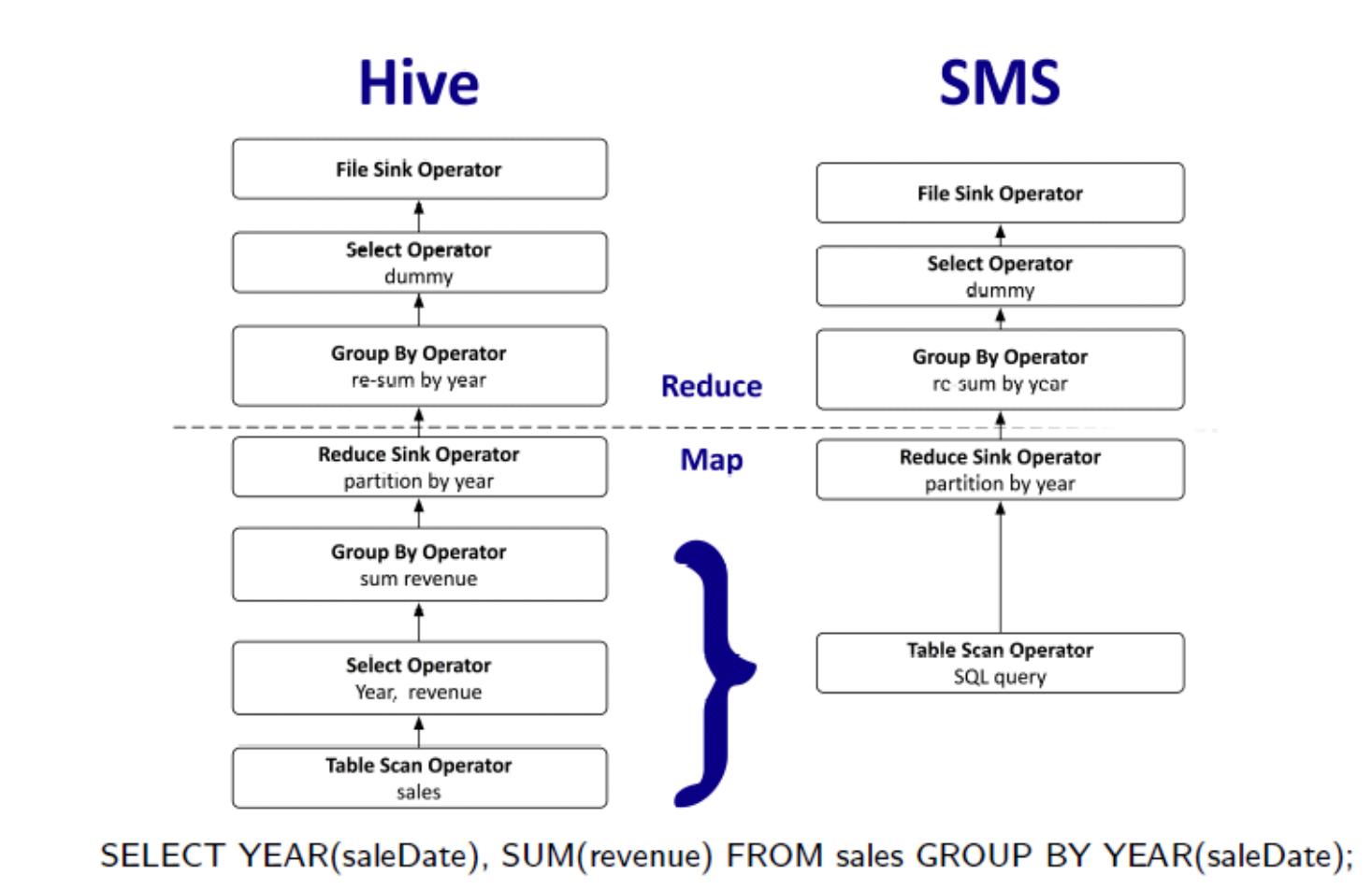
Best of Both Worlds Possible?

- Many of Hadoop's deficiencies not fundamental
  - Result of initial design for unstructured data
- HadoopDB: Use Hadoop to coordinate execution of multiple independent (typically single node, open source) database systems
  - Flexible query interface (accepts both SQL and MapReduce)
  - Open source (built using open source components)

HadoopDB Architecture

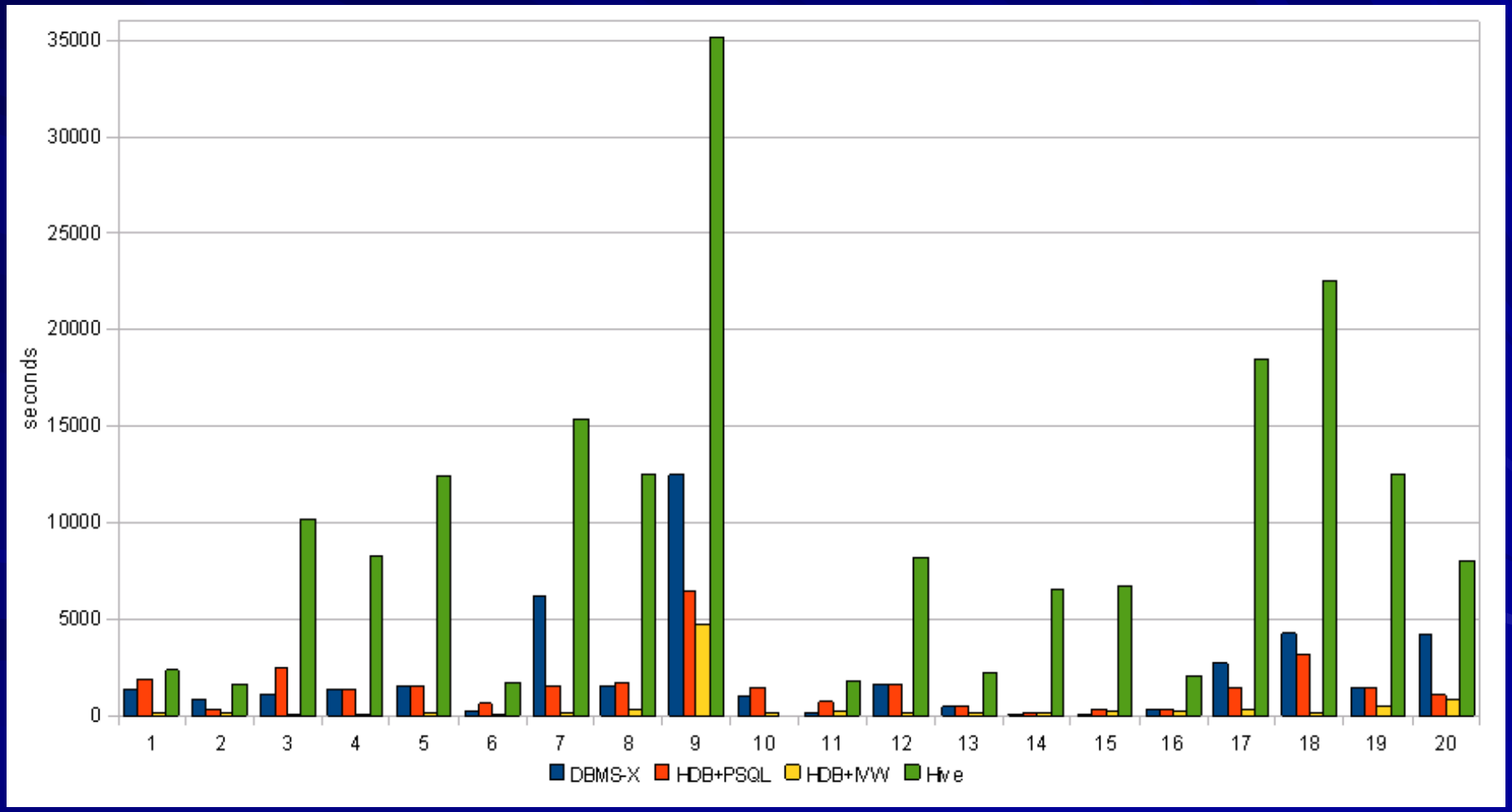


SMS Planner

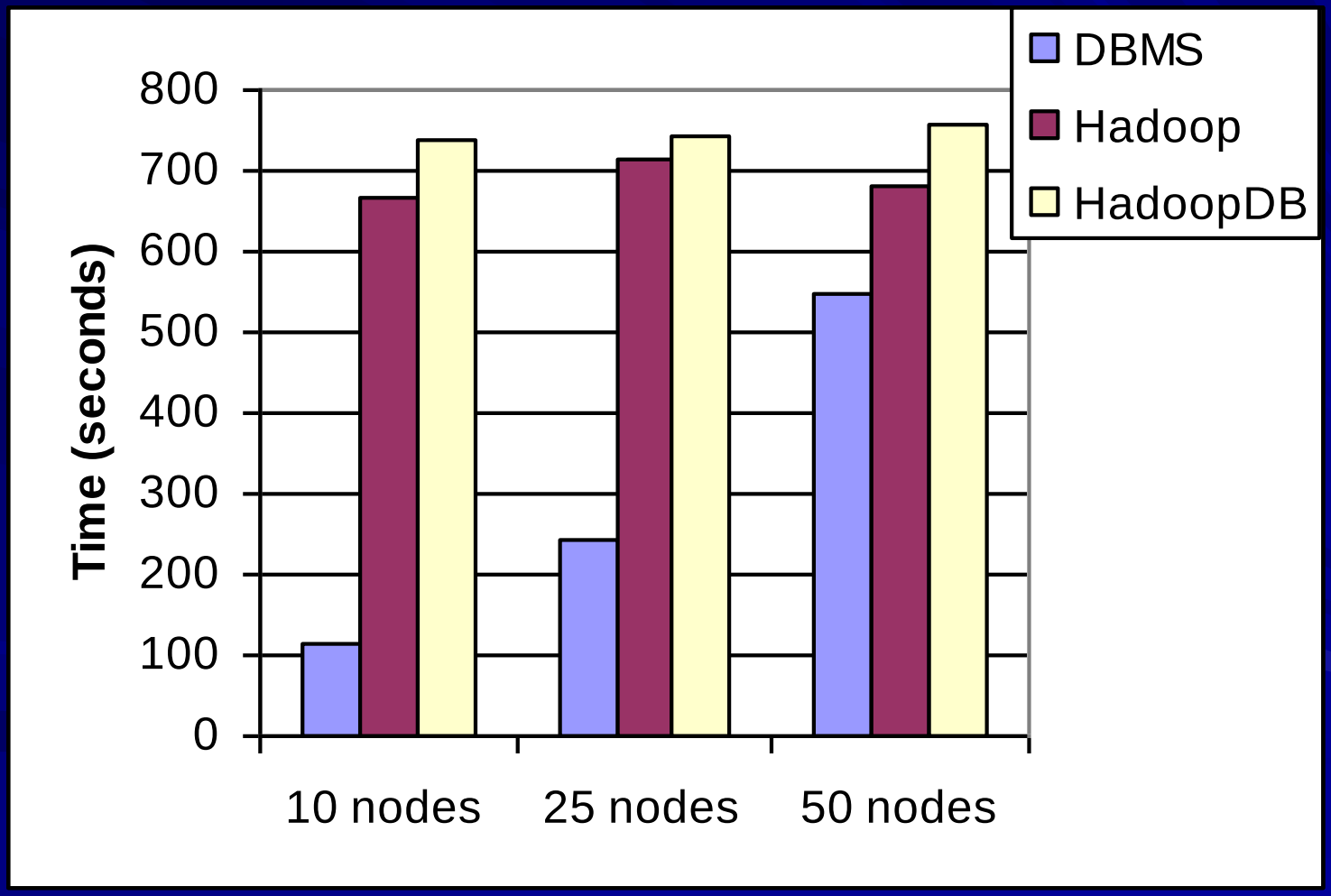


SELECT YEAR(saleDate), SUM(revenue) FROM sales GROUP BY YEAR(saleDate);

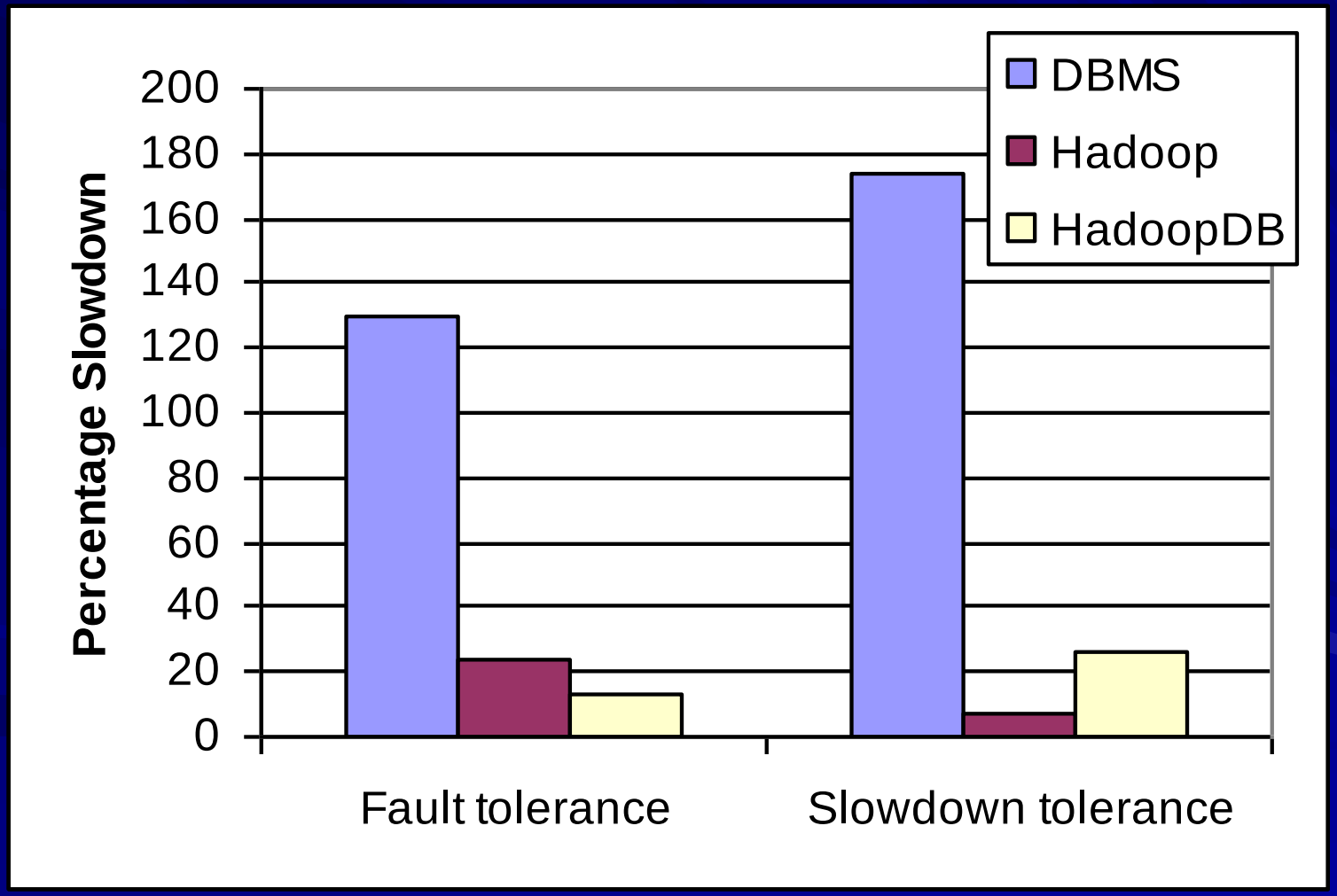
TPC-H Benchmark Results



UDF Task


DBMS
Hadoop
HadoopDB
Time (seconds)
800
700
600
500
400
300
200
100
0
10 nodes
25 nodes
50 nodes

Fault Tolerance and Cluster Heterogeneity Results



Invisible Loading

- Data starts in HDFS
- Data is immediately available for processing (immediate gratification paradigm)
- Each MapReduce job causes data movement from HDFS to database systems
- Data is incrementally loaded, sorted, and indexed
- Query performance improves “invisibly”

Conclusions

- Parallel database systems can be used for many data intensive tasks
  - Scalability can be an issue at extreme scale
  - Parallelization of UDFs can be an issue
- Hadoop is becoming increasingly popular and more robust
  - Free and open source
  - Great scalability and flexibility
  - Inefficient on structured data
- HadoopDB trying to get best of worlds
  - Storage layer of database systems with parallelization and job scheduling layer of Hadoop
- HadoopDB needs additional development before it can be useful in general
  - Full SQL support (via SMS planner)
  - Speed up (and automate) replication and loading
  - Easier deployment and managing
  - Automatic repartitioning about node addition/subtraction