BIG DATA BENCHMARKING:
A REVIEW OF YAHOO! CLOUD SERVING BENCHMARK (YCSB) FRAMEWORK[1], MAPREDUCE PARADIGM, AND PARALLEL DATABASE MANAGEMENT SYSTEM (DBMS) [2]

MD. NAZMUL HAQUE KHAN

UNIVERSITY OF HELSINKI

[1] DOI: 10.1145/1807128.1807152 , [2] DOI: 10.1145/1559845.1559865

BENCHMARKING

In computing, a **benchmark** is the act of running a computer program, a set of programs, or other operations, in order to assess the relative **performance** of an object, normally by running a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

BIG DATA BENCHMARKING …

Benchmarks for big data could be defined by the "Vs"

- Volume
  - Can the benchmark test scalability of the system to very large volumes of data?
- Velocity
  - Can the benchmark test ability of the system to deal with high velocity of incoming data?
- Variety
  - Can the benchmark include operations on heterogeneous data, e.g. unstructured, semi-structured, structured?

INDUSTRY STANDARD BENCHMARKING ORGANIZATIONS

- TPC - Transaction Processing Performance Council (http://www.tpc.org/)
- SPEC - The Standard Performance Evaluation Corporation (https://www.spec.org/)
- CLDS – Centre for Large- scale Data System Research (http://clds.sdsc.edu/bdbc)

TYPES OF BENCHMARKS

- **Micro-benchmarks**: To evaluate specific lower-level, system operations
  - E.g., A Micro-benchmark Suite for Evaluating HDFS Operations on Modern Clusters, Panda et al, OSU
- **Functional \ component benchmarks**: Specific high-level function.
  - E.g. Sorting: Terasort
  - E.g. Basic SQL: Individual SQL operations, e.g. Select, Project, Join, Order-By, …
- **Genre-specific benchmarks:** Benchmarks related to type of data
  - E.g. Graph500. Breadth-first graph traversals
- **Application-level benchmarks:** Measure system performance (hardware and software) for a given application scenario—with given data and workload

YAHOO! CLOUD SERVING BENCHMARK (YCSB) FRAMEWORK

- Different database systems prioritize different operations e.g. some optimize write performance, while others optimize read performance. Such optimization is a trade-off and understandably leads to de-optimization of some other attribute.
- The goal of the YCSB framework is to remedy this situation, by enabling an apples-to-apples comparison among the variety of database systems.
- The paper introducing it also tests and reports for four databases PNUTS, HBase, Cassandra, and a sharded MySQL implementation.
- Entire source code is public (open-source software).
- Further work on the project is possible and encouraged.
- Modular design of the project allows for easy extension e.g. of workloads, tests, etc.

CLIENT ARCHITECTURE

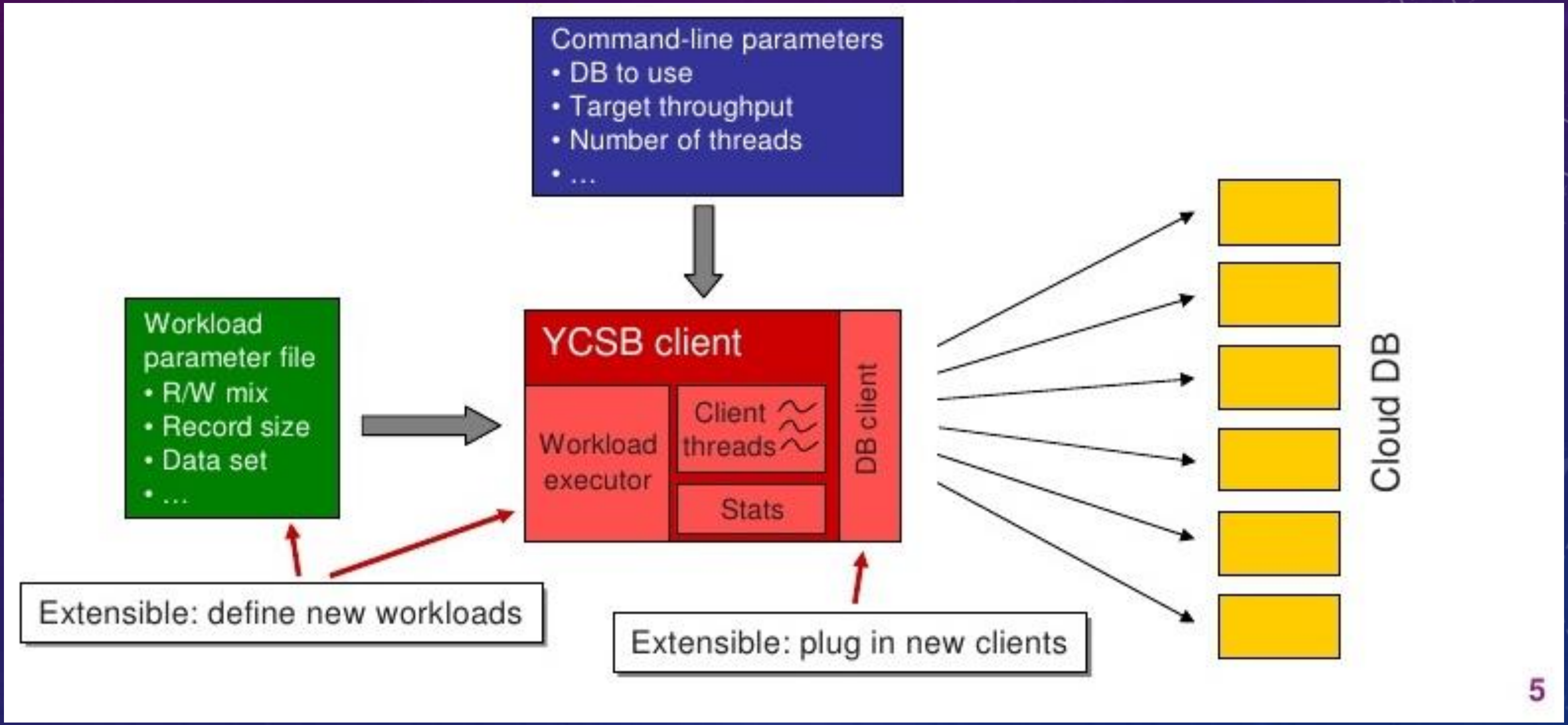
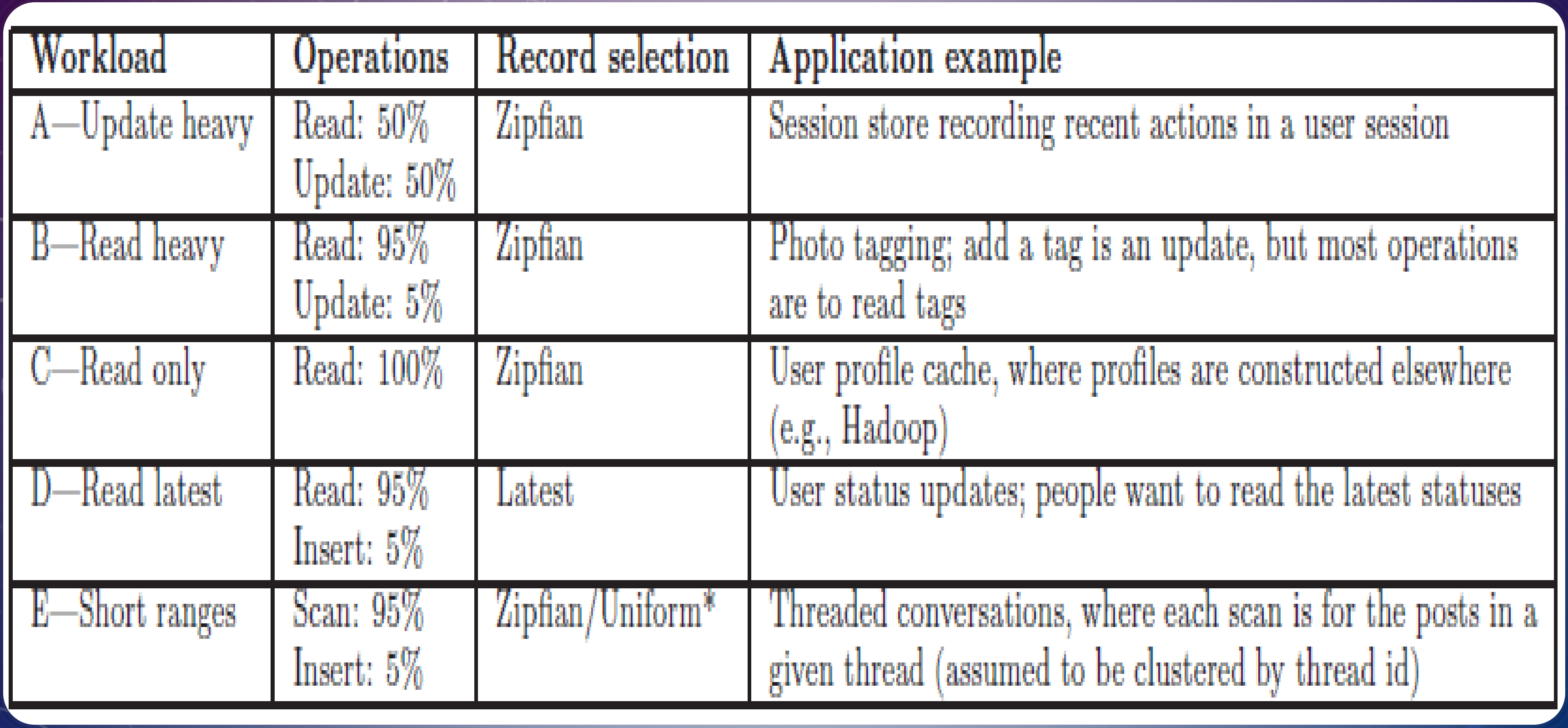


Figure by https://www.slideshare.net/kevinhan/yahoo-cloud-serving-benchmark

TEST SETUP

- Setup
  - Six server-class machines
    - 8 cores (2 x quadcore) 2.5 GHz CPUs, 8 GB RAM, 6 x 146GB 15K RPM SAS drives in RAID 1+0, Gigabit ethernet, RHEL 4
  - Plus extra machines for clients, routers, controllers, etc.
  - Cassandra 0.4.2
  - HBase 0.20.2
  - MySQL 5.1.32 organized into a sharded configuration
  - Sherpa 1.8
  - No replication; force updates to disk (except HBase, which does not yet support this)
- Workloads
  - 120 million 1 KB records = 20 GB per server
  - Reads retrieve whole record; updates write a single field
  - 100 or more client threads

CORE WORKLOADS



Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Insert: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Scan: 95% Insert: 5%	Zipfian/Uniform*	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

Figure by: DOI: 10.1145/1807128.1807152

BENCHMARKING TIERS

Tier 1: Performance (Latency)

- Measure latency as throughput is increased until system is saturated.

Tier 2: Scaling

- **Scaleup.** Increase number of servers, amount of data, and offered throughput scale proportionally. Latency should be constant.
- **Elastic Speedup.** In running system, add more servers. Performance should improve.

WORKLOAD A : UPDATE HEAVY

- 50/50 Read/update

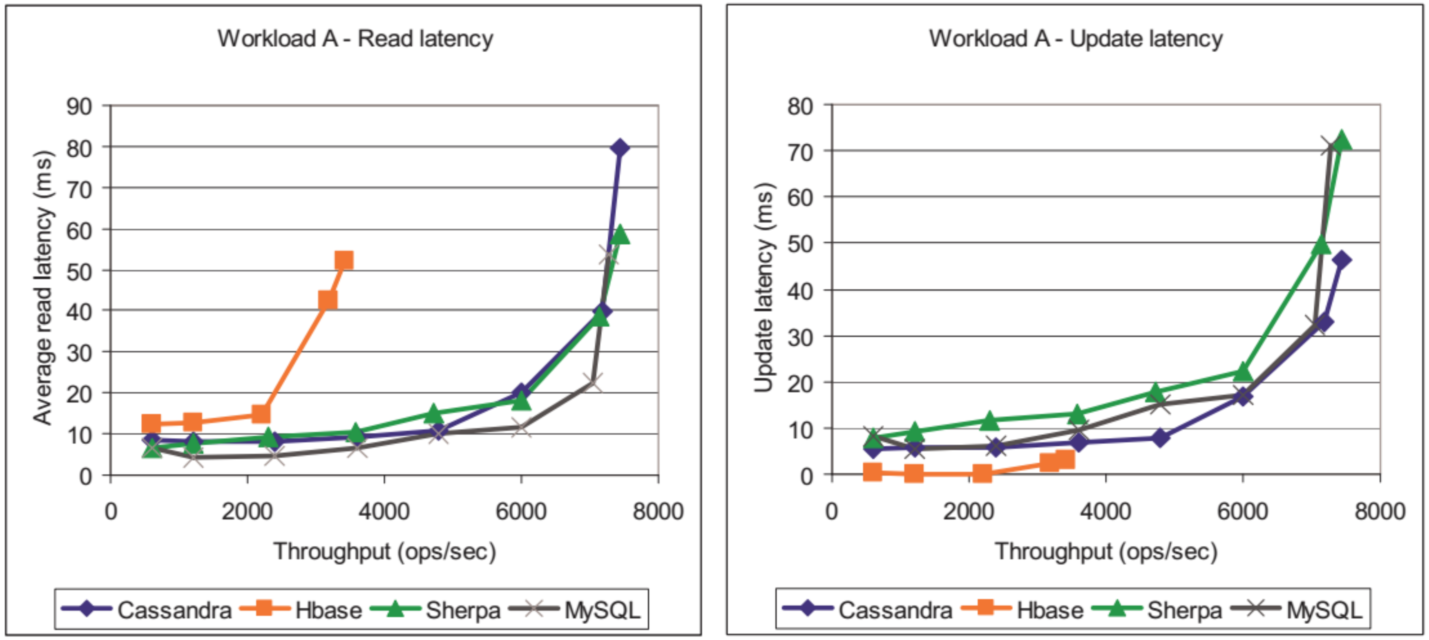
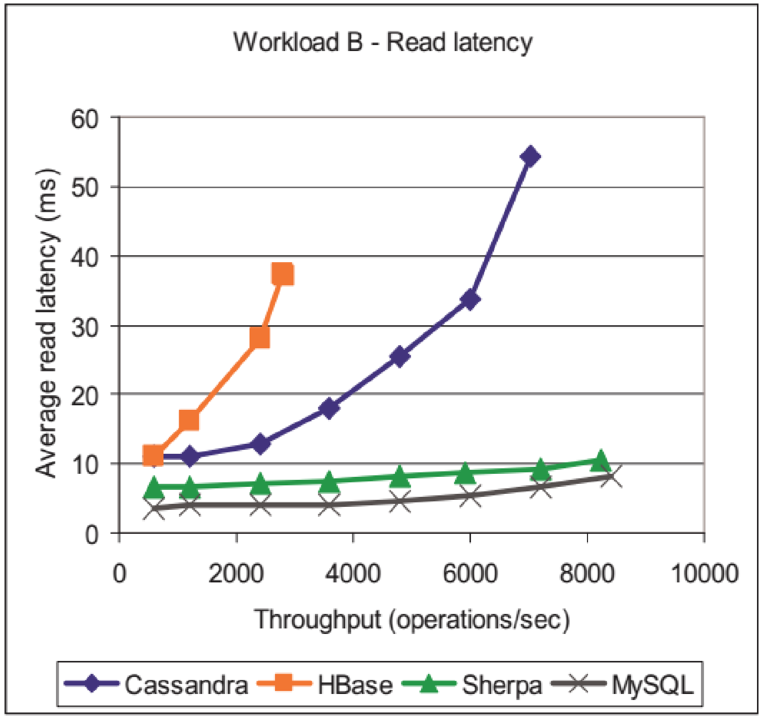


Figure : Workload A—update heavy: (a) read operations, (b) update operations. Throughput in this (and all figures) represents total operations per second, including reads and writes.

Figure by https://www.slideshare.net/kevinhan/yahoo-cloud-serving-benchmark

WORKLOAD B : READ HEAVY



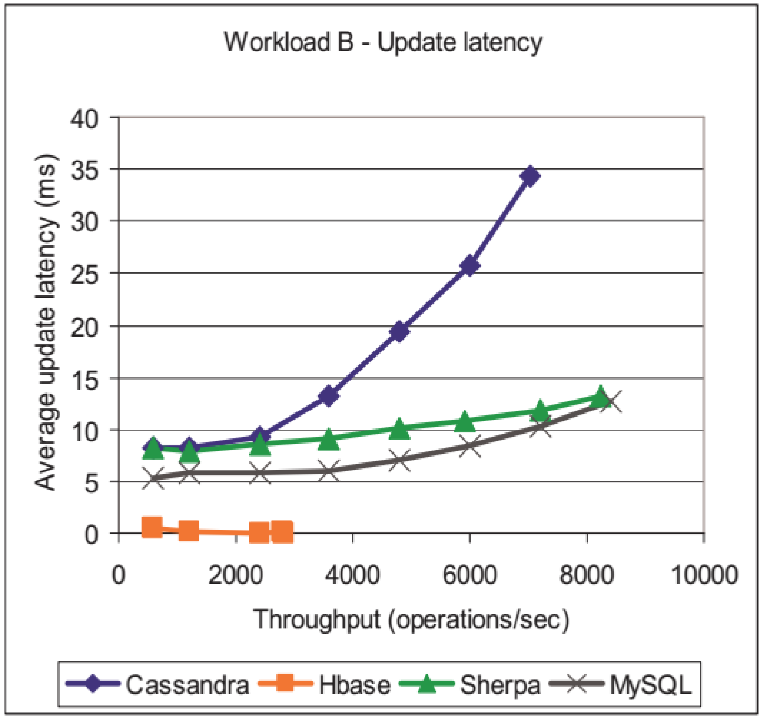


Figure : Workload B—read heavy: (a) read operations, (b) update operations.

Figure by https://www.slideshare.net/kevinhan/yahoo-cloud-serving-benchmark

SCALABILITY

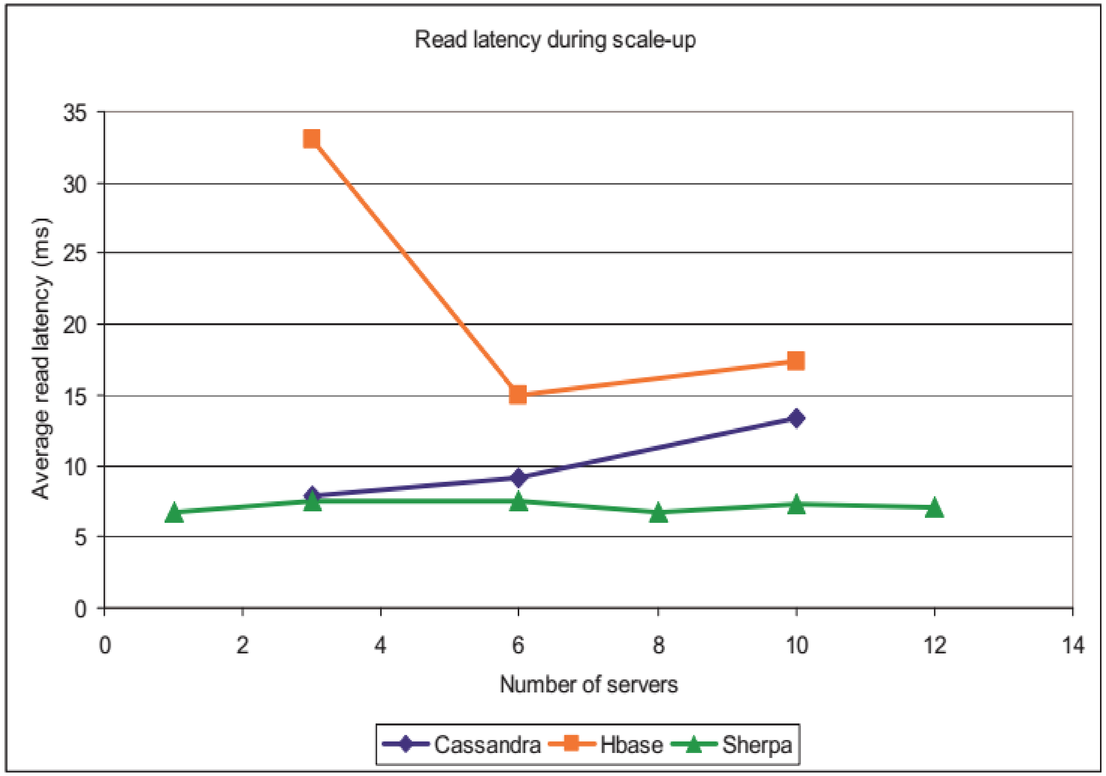


Figure : Read performance as cluster size increases.

Figure by https://www.slideshare.net/kevinhan/yahoo-cloud-serving-benchmark

ELASTICITY

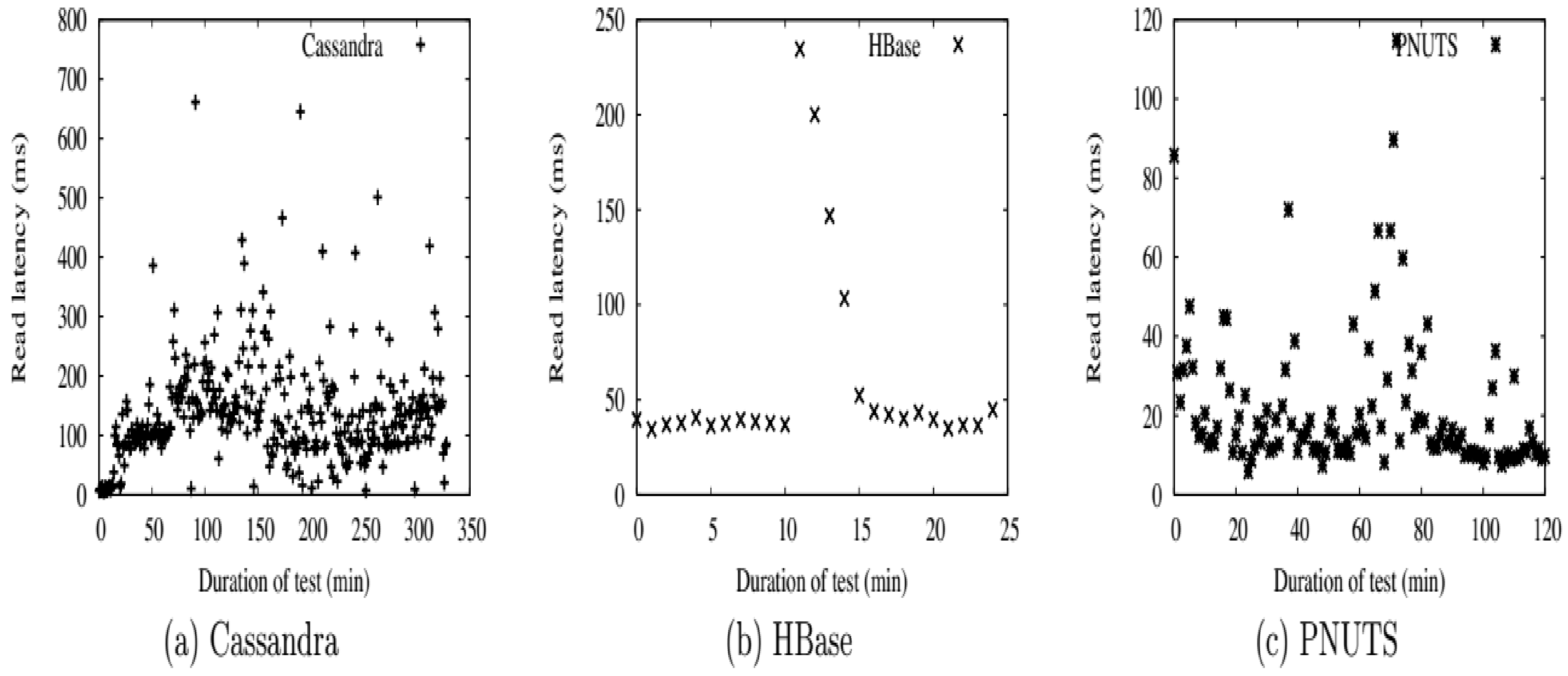


Figure 7: Elastic speedup: Time series showing impact of adding servers online.

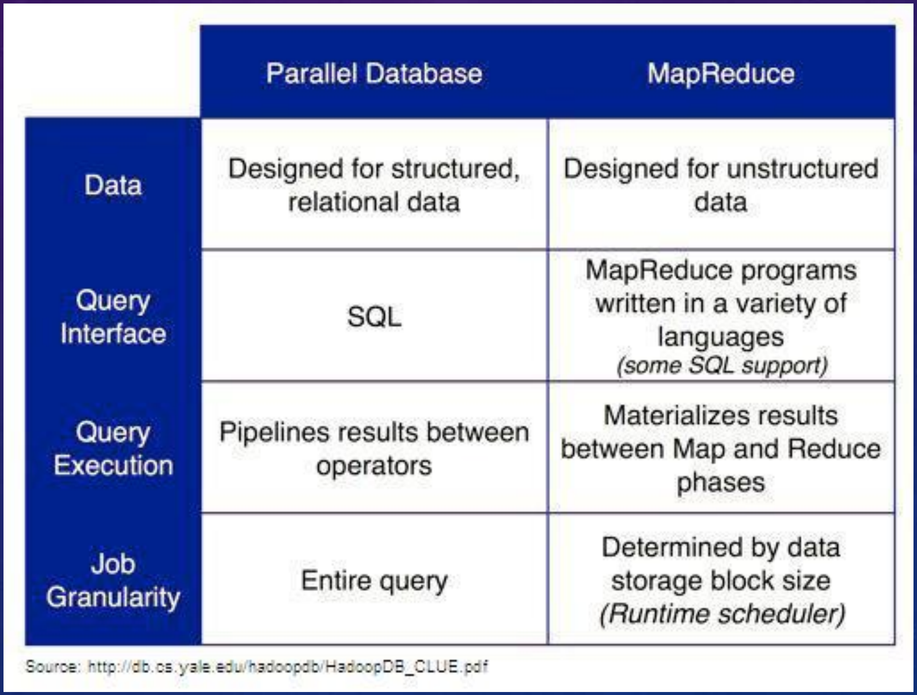
Figure by: DOI: 10.1145/1807128.1807152

YAHOO! CLOUD SERVING BENCHMARK (YCSB) FRAMEWORK – SUMMARY

- In totality, Cassandra and HBase were slower in read-heavy workload and faster in write-heavy workload. PNUTS and MySQL were faster in read-heavy workload.
- Sherpa is efficient for random lokoups and does well. As range increases, HBase begins to perform better since it is optimized for large scans
- PNUTS and Cassandra scaled well as the number of servers and workload increased proportionally. HBase performance was more unpredictable as the system scaled.
- Elasticity of Cassandra, HBase and PNUTS were good during the workload was execution but PNUTS was best with most stable latency while elastically repartitioning data.

MAPREDUCE PARADIGM, AND PARALLEL DATABASE MANAGEMENT SYSTEM (DBMS) - INTRODUCTION

- A comparison of the approaches to analyzing Big Data effectively comparing MapReduce (MR) and Parallel Database Management System (DBMS).
- Systems compared - Hadoop, DBMS-X, and Vertica.



	Parallel Database	MapReduce
Data	Designed for structured, relational data	Designed for unstructured data
Query Interface	SQL	MapReduce programs written in a variety of languages *(some SQL support)*
Query Execution	Pipelines results between operators	Materializes results between Map and Reduce phases
Job Granularity	Entire query	Determined by data storage block size *(Runtime scheduler)*

Source: http://db.cs.yale.edu/hadoopdb/HadoopDB_CLUE.pdf

Table by http://www.hadoopsphere.com/2012/07/dbms-within-hadoop-nodes-for-high.html

MAPREDUCE PARADIGM, AND PARALLEL DATABASE MANAGEMENT SYSTEM (DBMS) - GOALS

- The goal is to compare the two paradigms in their efficacy for analyzing Big Data.
- Parameters evaluated – performance, and development complexity.
- The authors accept that the two approaches have different design decisions.
- Consider the choices made in either approach and the trade-offs incurred due to these decisions.

COMPARING DEVELOPMENT COMPLEXITY

- Schema Support
- Programming Model & Flexibility
- Indexing
- Execution Strategy & Fault Tolerance
- Data transfers

SCHEMA SUPPORT

- MapReduce
  - Flexible, programmers write code to interpret input data
  - Good for single application scenario
  - Bad if data are shared by multiple applications. Must address data syntax, consistency, etc.

- Parallel DBMS
  - Relational schema required
  - Good if data are shared by multiple applications

PROGRAMMING MODEL & FLEXIBILITY

- MapReduce
  - Low level
  - “Anecdotal evidence from the MR community suggests that there is widespread sharing of MR code fragments to do common tasks, such as joining data sets.”
  - very flexible

- Parallel DBMS
  - SQL
  - User-defined functions, stored procedures, user-defined aggregates

INDEXING

- MapReduce
  - No native index support
  - Programmers can implement their own index support in Map/Reduce code
  - But hard to share the customized indexes in multiple applications
- Parallel DBMS
  - Hash/b-tree indexes well supported

EXECUTION STRATEGY & FAULT TOLERANCE

- MapReduce
  - Intermediate results are saved to local files
  - If a node fails, run the node-task again on another node
  - At a mapper machine, when multiple reducers are reading multiple local files, there could be large numbers of disk seeks, leading to poor performance.

- Parallel DBMS
  - Intermediate results are pushed across network
  - If a node fails, must re-run the entire query

AVOIDING DATA TRANSFERS

- MapReduce
  - Schedule Map to close to data
  - But other than this, programmers must avoid data transfers themselves

- Parallel DBMS
  - A lot of optimizations
  - Such as determine where to perform filtering

BENCHMARK ENVIRONMENT

- Hadoop (0.19.0 on Java 1.6.0)
  - HDFS data block size: 256MB
  - JVMs use 3.5GB heap size per node
  - “Rack awareness” enabled for data locality
  - Three replicas w/o compression
- DBMS-X (a parallel SQL DBMS from a major vendor)
  - Row store
  - 4GB shared memory for buffer pool and temp space per node
  - Compressed table
- Vertica
  - Column store
  - 256MB buffer size per node
  - Compressed columns by default

BENCHMARK ENVIRONMENT (CONT.)

- Node Configuration:
  - 100-node cluster
  - Each node: 2.40GHz Intel Core 2 Duo, 64-bit red hat enterprise linux 5 (kernel 2.6.18) w/ 4Gb RAM and two 250GB SATA HDDs.
  - Nodes are connected to Cisco Catalyst 3750E 1Gbps switches. Internal switching fabric has 128Gbps. 50 nodes per switch
  - Multiple switches are connected to create a 64Gbps Cisco StackWise plus ring. The ring is only used for cross-switch communications.

DATA LOADING TIMES

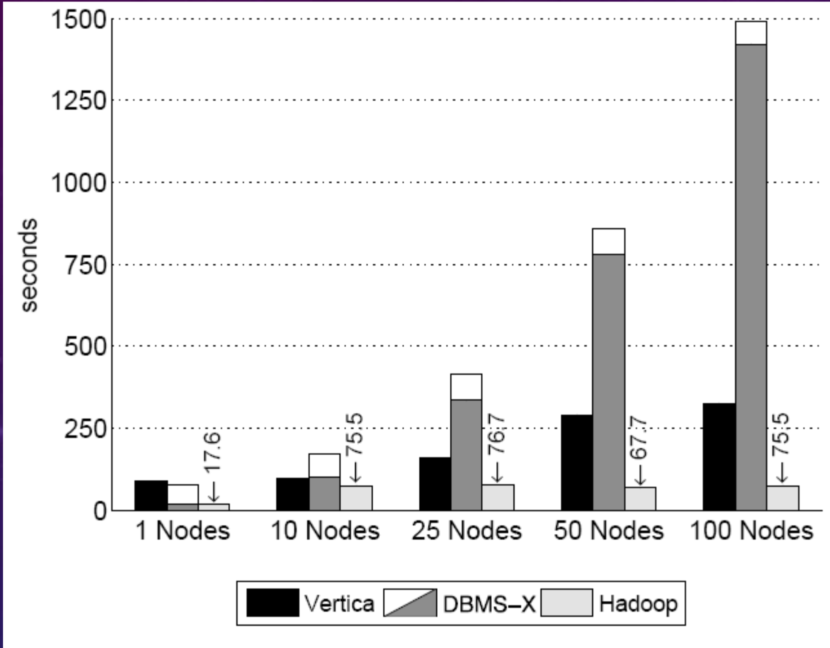


Figure 1: Load Times – Grep Task Data Set (535MB/node)

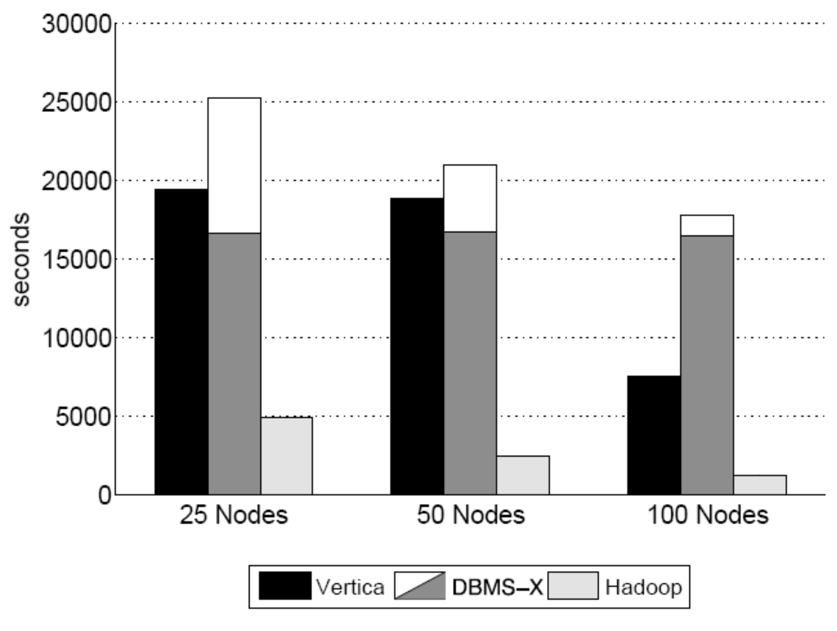


Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

DBMS-X: grey is loading, white is re-organization after loading

Loading is actually sequential despite parallel load commands

Hadoop does better because it only copies the data to three HDFS replicas.

Figure by: DOI: 10.1145/1559845.1559865

PERFORMANCE BENCHMARKS

- Original MR task (Grep)
- Analytical Tasks
  - Selection
  - Aggregation
  - Join
  - User-defined-function (UDF) aggregation

EXECUTION TIME

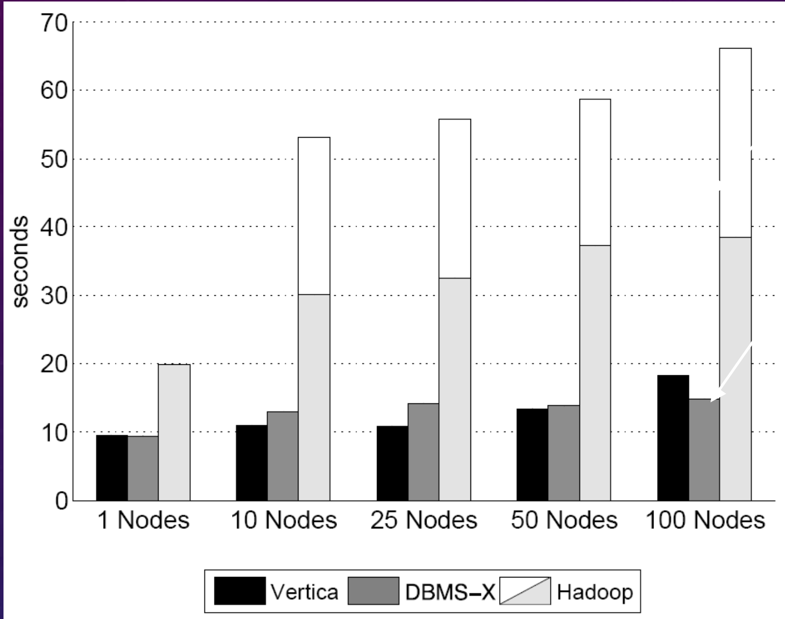


Figure 4: Grep Task Results – 535MB/node Data Set

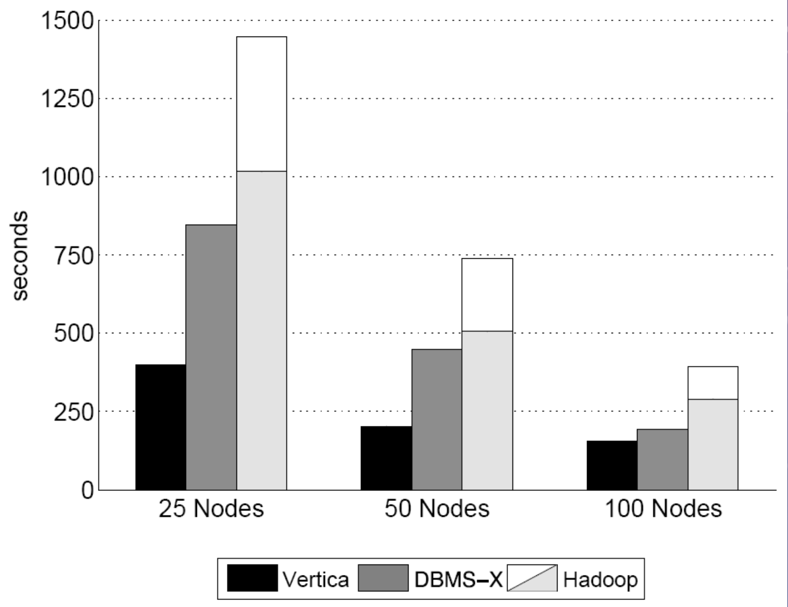


Figure 5: Grep Task Results – 1TB/cluster Data Set

Hadoop's large start-up cost shows up in Figure 4, when data per node is small

Vertica's good data compression

Figure by: DOI: 10.1145/1559845.1559865

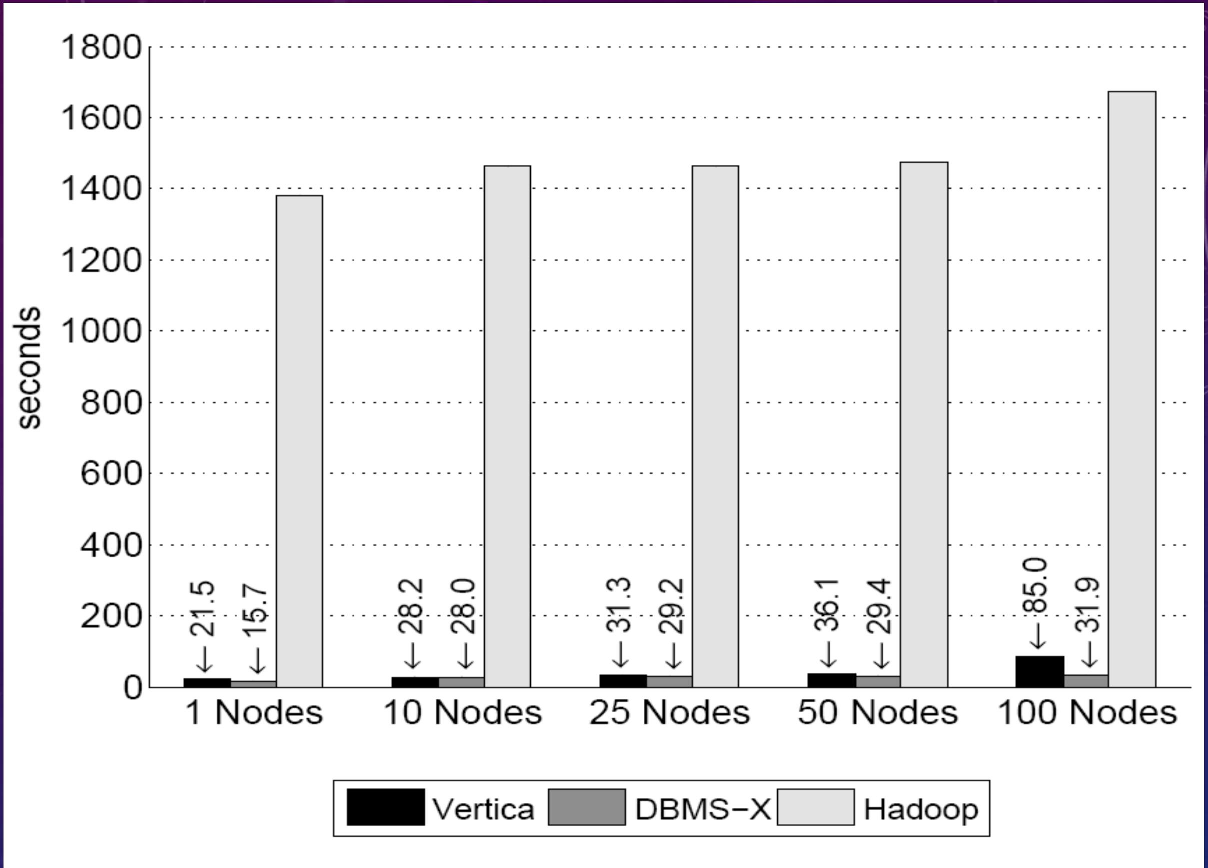


Figure 9: Join Task Results

DBMS can use index, both relations are partitioned on the join key; MR has to read all data.

MR phase 1 takes an average 1434.7 seconds

600 seconds of raw I/O to read the table; 300 seconds to split, parse, deserialize; Thus CPU overhead is the limiting factor

Figure by: DOI: 10.1145/1559845.1559865

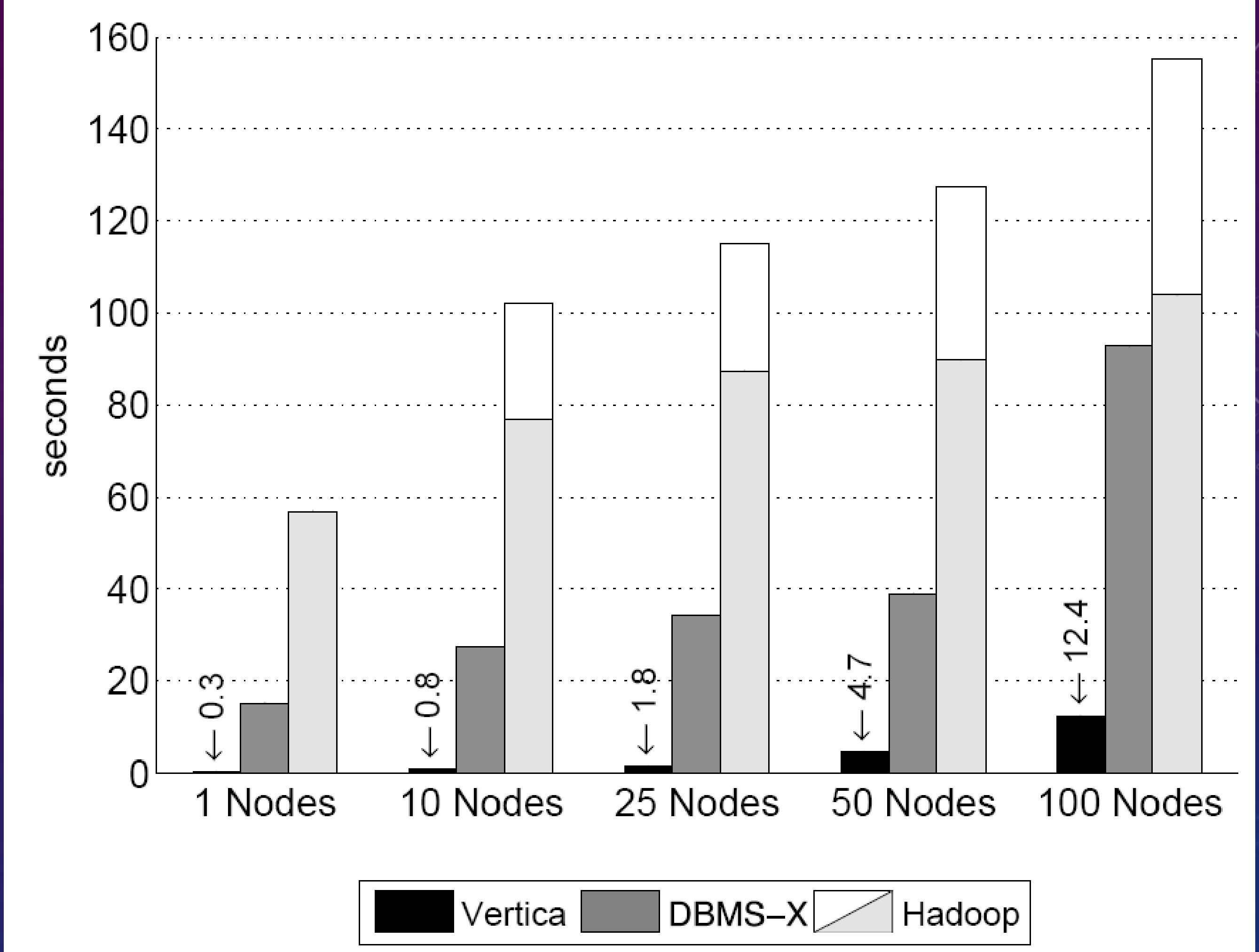


Figure 6: Selection Task Results

Hadoop's start-up cost; DBMS uses index; vertica's reliable message layer becomes bottleneck

Figure by:DOI: 10.1145/1559845.1559865

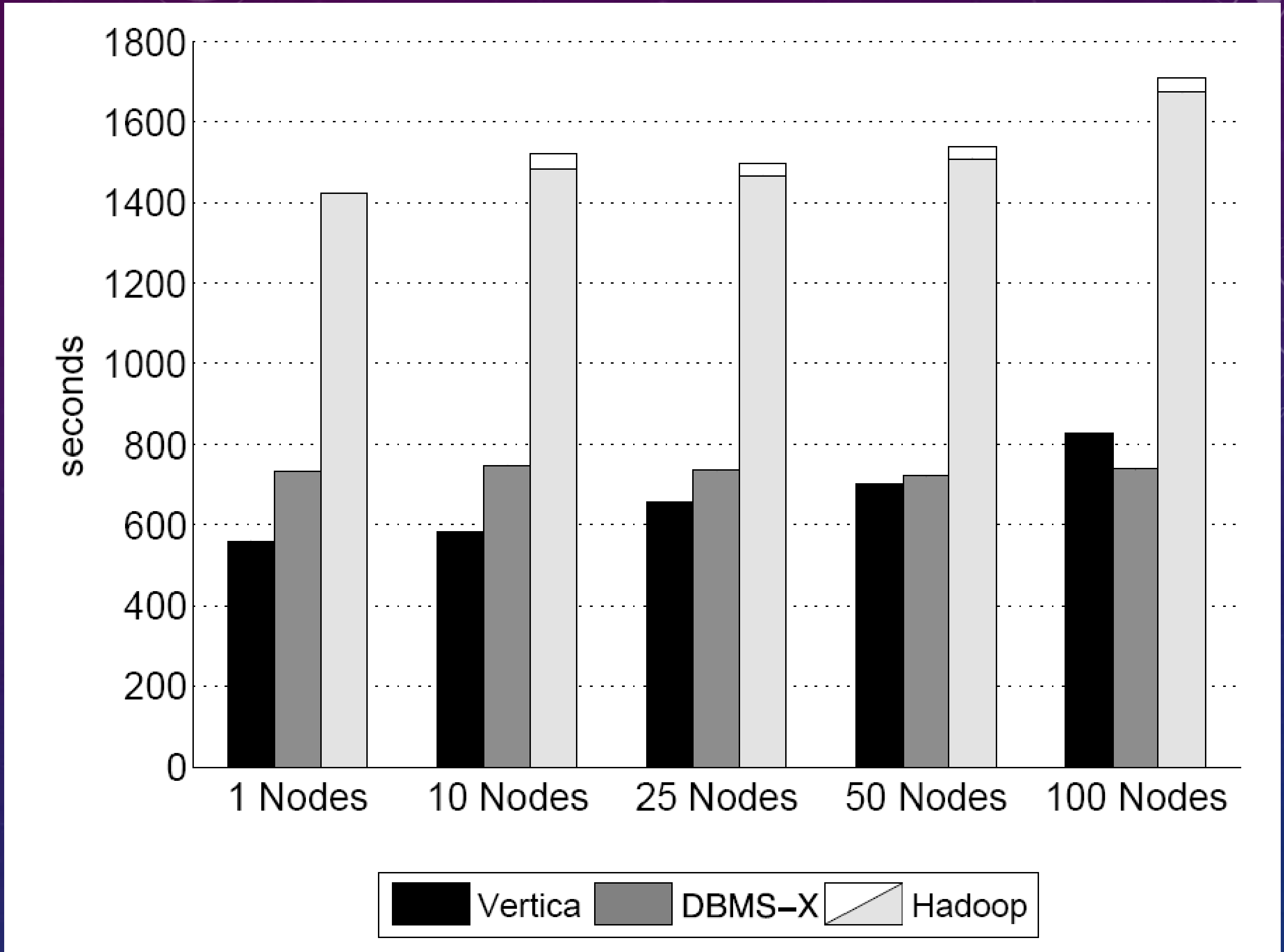


Figure 7: Aggregation Task Results (2.5 million Groups)

DBMS: Local group-by, then the coordinator performs the global group-by; performance dominated by data transfer.

Figure by: DOI: 10.1145/1559845.1559865

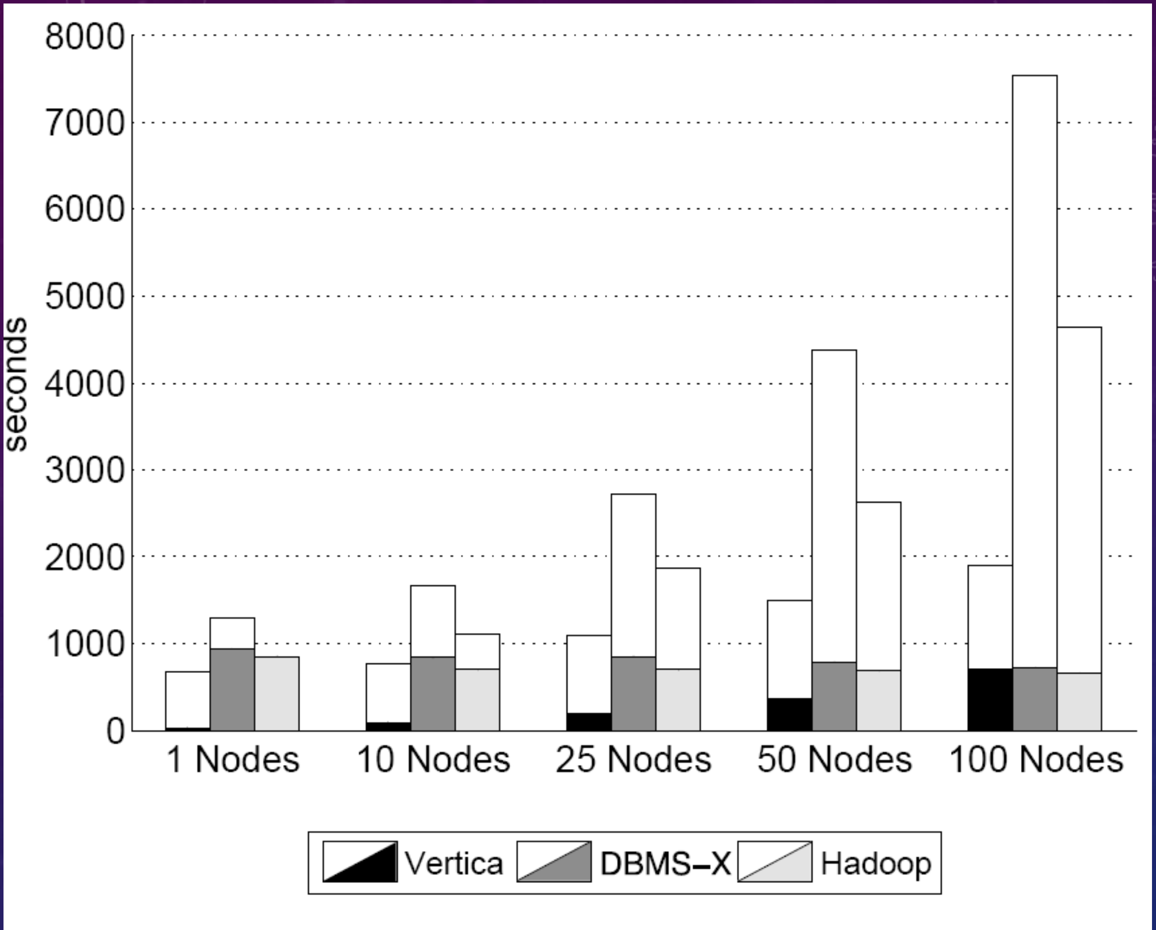


Figure 10: UDF Aggregation Task Results

- DBMS: lower – UDF time; upper – other query time
- Hadoop: lower – query time; upper: combine all results into one

Figure by: DOI: 10.1145/1559845.1559865

MAPREDUCE PARADIGM, AND PARALLEL DATABASE MANAGEMENT SYSTEM (DBMS) - SUMMARY

- In totality, the authors found SQL DBMS to be significantly faster and demanding of less code.
- MR programs took some time before all nodes were running at full capacity. In contrast, parallel DBMSs are started at OS boot time, and thus are considered to always be “warm”, waiting for a query to execute
- Compression does not improve performance in Hadoop.
- Parallel DBMSs much more challenging than Hadoop to install and configure properly and takes longer time to load and tune the data.

MAPREDUCE PARADIGM, AND PARALLEL DATABASE MANAGEMENT SYSTEM (DBMS) – SUMMARY (CONT.)

- MR systems use a large number of control messages to synchronize processing, resulting in poorer performance due to increased overhead
- Fault tolerance is much better in MR system.
- It may be easier to for developers to get started with MR, maintenance of MR programs is likely to lead to significant pain for applications developers over time.

RECENT APPROACHES TO BIG DATA BENCHMARKING

- BigBench
  - It is an industry-wide effort on creating a comprehensive and standardized Big Data benchmark. Commercial supporters include Intel and Cloudera.
- Yahoo! Streaming Benchmark
  - A combined tool to compare Apache Flink, Apache Storm and Apache Spark Streaming to provide an apples-to-apples comparison.
- TPC-DS
  - TPC-DS is the de-facto industry standard benchmark for measuring the performance of decision support solutions including, but not limited to, Big Data systems.

CONCLUSION

- Big data systems differ in architectural designs and their particular requirements.
- Both the papers admit that there is a trade-off in every alternative - if read-time performance is optimized, then the write-time performance is poor.
- The way to approach the requirements for a particular scenario is need-based i.e. what does the current application require?
- Big data benchmarking still lacks unified industry standard.
- Several benchmark tools are developed but each satisfies specific purpose.

REFERENCES

- Cooper, B. (2016). *GitHub - brianfrankcooper/YCSB: Yahoo! Cloud Serving Benchmark*. *Github.com*.
- https://github.com/brianfrankcooper/YCSB
- Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010). Benchmarking cloud serving systems with YCSB. *Proceedings Of The 1St ACM Symposium On Cloud Computing - Socc '10*. http://dx.doi.org/10.1145/1807128.1807152
- Pavlo, A., Paulson, E., Rasin, A., Abadi, D., DeWitt, D., Madden, S., & Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. *Proceedings Of The 35Th SIGMOD International Conference On Management Of Data - SIGMOD '09*. http://dx.doi.org/10.1145/1559845.1559865

THANK YOU

QUESTIONS?