Evaluation of NoSQL and Array Databases for Scientific Applications

Lavanya Ramakrishnan, Pradeep K. Mantha, Yushu Yao, Richard S. Canon

Lawrence Berkeley National Lab
Berkeley, CA 94720
[lramakrishnan,pkmantha,yyao,scanon]@lbl.gov

ABSTRACT

Scientific users are increasingly considering the use of NoSQL and array databases for storing metadata and data. These databases offer various advantages including support for real-time changing schema and performance optimizations for specific operations. However, there is a limited understanding of the strengths and weaknesses of these databases for scientific applications. In this paper, we present an evaluation using standard benchmarks (Yahoo! Cloud Serving Benchmark) and two scientific data sets. Our results indicate that careful understanding of the distribution of the workload as well as aspects such as client side tools and parameters need to be considered to get the optimal performance from these databases.

1. INTRODUCTION

Scientific discoveries for important challenges facing our society require federation or integration of data from multiple research groups and disciplines. The data infrastructure must be able to accommodate diverse data-types and formats from diverse experiment facilities, models and sources.

Today, data and metadata is primarily stored in file systems and in rare cases in relational databases [14]. Scalable file systems are critical as an underlying software layer, but these are not sufficient for user-level data management storage, nor do they offer an adequate interface for queries and data analyseos. Also, it is often difficult to attach schema, or meaning, to the data beforehand and the schema and metadata tends to evolve over time. Thus, datastores need to accommodate real-time changes to the structure of the data while making the data searchable on any attribute.

Cloud applications are increasingly using schema-less (also referred to as Not Only SQL or NoSQL) datastores–which enable capture of semi-structured data and allow attribute-based searches. Increasingly scientific applications are investigating the use of NoSQL databases for their needs (e.g., the Materials Project [9], data from the Advanced Light Source [12]) and/or array databases. NoSQL databases address many of the needs of a data store for storing less structured and evolving data schemas. Current implementations of schema-less databases are based on the ideas proposed by Amazon (Dynamo) [6] and Google (Bigtable) [3]. Similarly, array databases provide performance optimizations for array operations. However, there is a limited understanding of the strengths and weaknesses of these new class of databases.

In this paper, we provide an evaluation of existing cloud schema-less and array databases for scientific data use. For our initial evaluation, we have selected MongoDB [11], Cassandra [2], HBase [10], and SciDB [1, 13]. Specifically, our evaluation includes:

- A comparison of Cassandra, HBase and MongoDB using the Yahoo! Cloud Serving Benchmark (YCSB)
- A comparison of Cassandra, SciDB and PostgresSQL using two scientific data sets
- Investigation of the impact of various factors impacting performance in the context of Cassandra

The rest of the paper is organized as follows. In Section 2, we discuss the methodology of our evaluation and detail results in Section 3. We discuss related work and conclusions in Sections 4 and 5 respectively.

2. METHODOLOGY

In this section, we describe our experiment setup including the databases and the workload and data sets.

2.1 NoSQL and Array Databases

Schema-less databases are typically classified as document-oriented, key-value, and graph-based. A document-oriented database stores, retrieves and manages semi-structured or document-oriented information. Key-value stores use a key-value pair. HBase, part of the Hadoop ecosystem and Cassandra, initially developed by Facebook are key-value stores. MongoDB, developed and supported by 10gen, is a document-oriented store. We Cassandra - 1.1.2, MongoDB - 2.4.5 and HBase - 0.20.3 for our evaluation.

SciDB is an array-based database that has recently gained traction for scientific data that can be represented in an array model. Our selection criteria are based on applicability to scientific data, feasibility, active development, and community support. We use SciDB 13.3 in our evalution.

The goal of our evaluation is not necessarily to determine the best product, but rather to understand the state of the

art and the application of each class of schema-less database to scientific data.

2.2 Workloads and Data Sets

First, we perform a comparison using the Yahoo! Cloud Serving Benchmark (YCSB) framework. YCSB facilitates the performance comparisons of the new generation of cloud data serving systems. The workload supports a number of different workloads from update heavy, read heavy, read only, read latest and short ranges (Table 2.3). The workload consists of 100 million records where each record is 1 KB (10 fields by 100 Bytes).

Second, we use two scientific data sets to compare the performance of Cassandra and SciDB with PostgresSQL. The data sets are from bioinformatics and astronomy.

The first data set consists of non-redundant (nr) protien sequences from bioinformatics. The database had 16 million entries where each entry had the following fields: sequence id, sequence, checksum. We used the sequence id as the key. The query used for our evaluation fetches all fields based on a sequence id.

The second data set is from astronomy. The data set contains a collection of 55K spectrums produced from simulation of different supernova with different input parameters. Each spectrum contains 2500 wavelength steps and two record types. The data set contains 275 million records. The primary key is a compound key consisting of spectrum id, wavelength id and record type. The query used calculates the Chi-square distance between the given observed spectrum to every spectrum in the set, and returns the ID of the spectrum that has minimal Chi-square distance.

2.3 Machines

All experiments were run on the Jesup testbed at National Energy Research Scientific Computing Center (NERSC). Jesup is a 160 node testbed cluster, used to explore emerging hardware and software technologies. The nodes consist of quad-core Intel Xeon X5550 ("Nehalem") 2.67 GHz processors (eight cores/node) with 24 GB of memory per node.

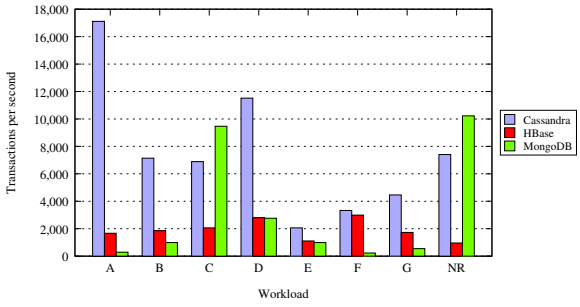


Figure 1: The figure shows the comparison of performance with different workload types (Table 2.3) with the different NoSQL databases.

3. RESULTS

We evaluate the various NoSQL stores and SciDB using the YCSB benchmarks and two scientific data sets.

3.1 YCSB

Workload A: Update heavy workload	Mix of 50/50 reads and writes. Zipifan
Workload B: Read mostly workload	95/5 reads/write mix. Zipifan
Workload C: Read only	100% read. Zipifan
Workload D: Read latest workload	New records are inserted, and the most recently inserted records are the most popular. (read/update/insert ratio: 95/0/5).Zipifan
Workload E: Short ranges	Short ranges of records are queried, instead of individual records. (read/update/insert ration: 95/0/5) Uniform
Workload F: Read-modify-write	The client will read a record, modify it, and write back the changes. (read/read-modify-write ration: 50/50). Zipifian
Workload G: Custom workload	100% write
Workload NR	Record size matching the bioinformatics workoad. Read only similar to Workload C

Table 1: The table shows the configuration of the YCSB workloads

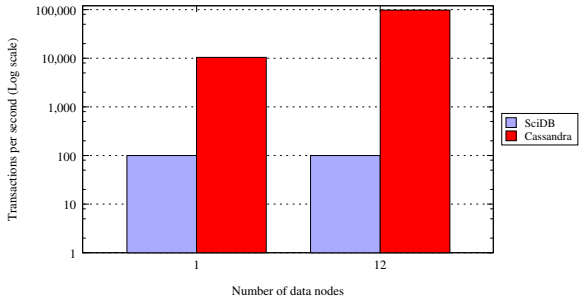


Figure 2: The figure shows the comparison of Cassandra and SciDB with the bioinformatics data set. Cassandra provides higher transactions/second than SciDB at one node. But the difference is significant when we scale to 12 nodes.

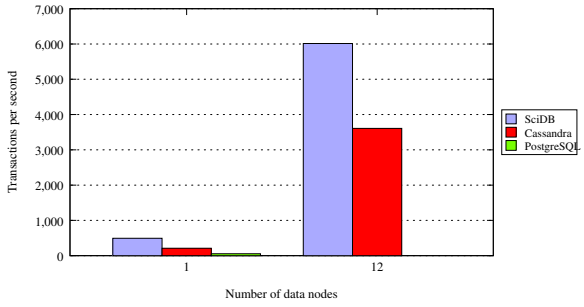
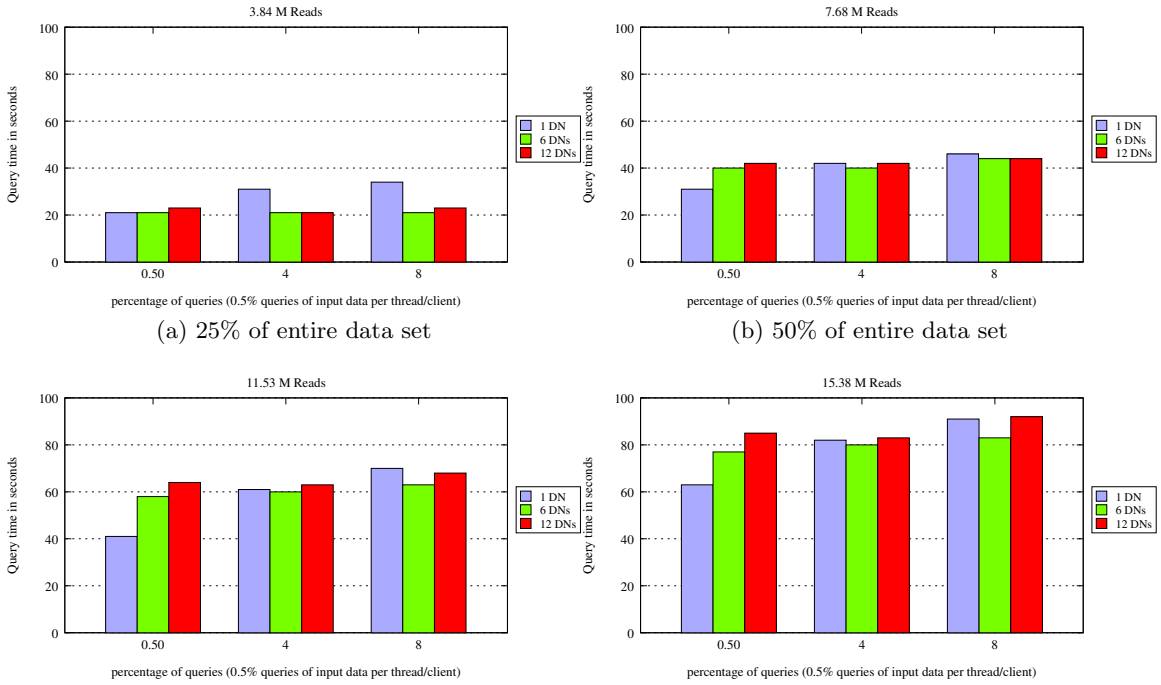


Figure 3: The figure shows the comparison of Cassandra, PyCassa and PostgresSQL for the astronomy data sets. The array-based queries give superior performance with SciDB.



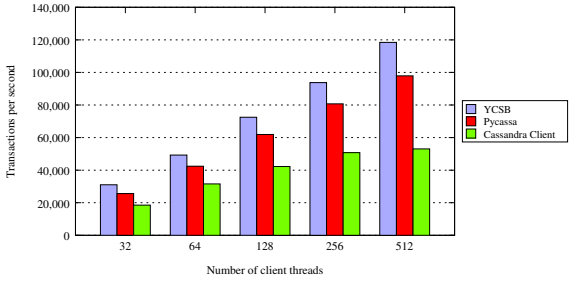
(a) 25% of entire data set

(b) 50% of entire data set

(c) 75% of entire data set

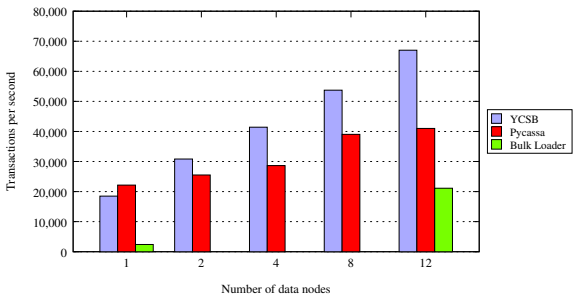
(d) 100% of entire data set

Figure 6: The figure compares the effects of varying database size with Cassandra-Cli tools.



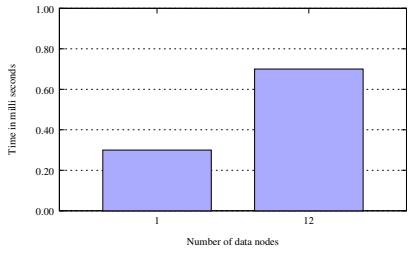
(a)

Figure 4: The figure compares the impact of different client side tools (YCSB, PyCassa and Cassandra Cli) in a case where the number of records in database and the records queried are identical.



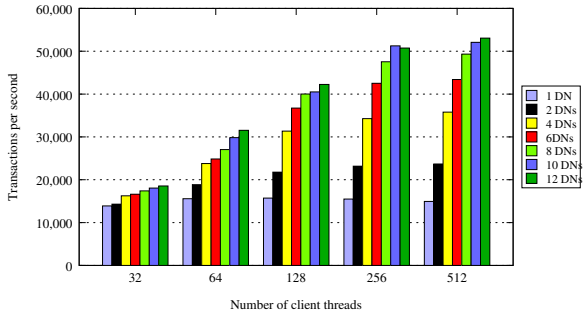
(a)

Figure 5: The figure compares the loading time with YCSB and PyCassa. In all cases 16 million operations are performed and 32 client threads/node are used



(a)

Figure 7: The figure shows the performance of Cassandra when querying for missing reads. The time for a query increases with the number of data nodes when you query for missing records.



(a)

Figure 8: The figure shows the effect of increasing data nodes. The effective throughput from adding additional nodes is evident when there are a large number of concurrent client threads.

Figure 1 shows the comparison of MongoDB, HBase and Cassandra using the standard YCSB benchmarks. MongoDB does better than Cassandra and HBase for the ready-only workload (workload C). Cassandra shows over 8 × the performance of the other databases for Workload A which has 50% reads and 50% writes. Cassandra shows between 3 × and 5 × the performance of the other databases with workloads B and D where reads dominate the workload. Cassandra does only slightly better than the other databases with short range queries and read, modify and write queries (workloads E and F). Cassandra does about 2 × better than the other databases with the write-only workload (workload G).

In addition, we also configured YCSB with a data set that is similar to our bioinformatics sequence data (marked as NR in figure). We matched the record size, fields and record count to be identical to our bioinformatics data set. The workload shows similar trends as workload C since our worload focussed only on the queries/reads.

3.2 Scientific Data Sets

Bioinformatics data. Figure 2 shows a comparison of Cassandra and SciDB with the NR sequence data where the Y-axis is in log scale. SciDB is able to provide only about 100 transactions/second for such queries and thus Cassandra provides significantly higher performance.

Astronomy data. Figure 3 shows the comparison of Cassandra, PyCassa and PostgresSQL for the astronomy data set. SciDB is specifically designed to handle array processing and hence SciDB gives the best performance at one node and is over 1.5 × faster than Cassandra at 12 nodes.

3.3 Miscellaneous

From our early experiments, Cassandra provided the best performance for a wide range of workload types. Thus, we performed additional experiments with Cassandra to understand a number of other characteristics of NoSQL databases.

Client Tools. We initially started out using Cassandra-Cli tools which is a very basic interactive command line interface. However, we noticed that the performance we were getting was well below the advertised performance. We shifted to PyCassa which is the Python client library for Cassandra. Figure 4 shows the comparison of the client side tools at 12 data nodes with the bioinformatics data set. PyCassa perfoms better than Cli and is almost 1.5 × at 512 client threads. Finally, we used the YCSB Java client tool to generate data sets similar to the the bioinformatics workload and the throughput achieved through this is even higher especially at increased number of client threads. This is probably due to known drawbacks in Python based tools due to the Global Intepreter Lock, etc compared to Java.

Database loading. Figure 5 shows our experience with database loading using various tools with varying data nodes. In each case, we performed 16 million operations using 32 client threads on a single node. On a single data node, the PyCassa client perfoms slightly better than using YCSB client tools (i.e., Java based client tools). However, as the number of data nodes increases, the YCSB client tools perform better than PyCassa. The bulk loader tool does not support multiple threads at this time and performed slower than the other tools.

Database size. Figure 6 compares the impact of the varying database size and the number of client threads. Each client thread queries 0.5% of the input data set. In this experiment, we vary the amount of data nodes, the number of clients and the size of the database. Figure 6(a) show that as the number of client threads are increased on a setup with a single node, query time takes longer. This trend is also seen in the other three graphs (Figure 6 (b-d)). Additionally, as the size of the database grows, it impacts the query times. For example, when only 3.84 million sequence reads are loaded, the queries complete in the 20-40 second range. However, at 15.38 million records, query times start at 60 seconds and are sometimes even as high as 90 seconds. The impact of additional data nodes is a little less clear in this experiment.

Missing reads. In addition to queries that return successful records, it is important to understand how datastores perform when querying for items that may not be present in the database. Figure 7 shows the effect on performance of Cassandra using PyCassa when querying for items that are not present in the database. In this experiment, we used a single client with a single thread on 100% of the data set. We query for keys that are not present in the data set. In this case, while queries return in less than one millisecond, the additional datanodes does impact the performance when querying for something that is not present in the database.

Data node scaling. Figure 8 shows the effect of adding additional data nodes. In this case we run 32 threads/client node. As the number of client thread increases, we see that the effective throughput increases with the additional data nodes. For example, at 512 client threads, we see a performance improvement of over 3 × when the number of data nodes were increased from one to 12.

4. RELATED WORK

Cooper et. al [5] describe the YCSB benchmark and compare different NoSQL and relational databases (Cassandra, HBase, Yahoo!'s PNUTS [4], and MySQL). Dory et al. [8] study the elastic scalability of MongoDB, HBase and Cassandra on a cloud infrastructure. These studies do not consider the performance achievable specifically for scientific applications using these frameworks.

Previous work has compared MongoDB and Hive to the relational database SQL Server PDW using YCSB and TPC-H DSS [15] benchmarks. Previous works have evaluated both MongoDB and Hadoop MapReduce performance for an evolutionary genetic algorithm and other data-intensive applications [16, 7].

However, there is no prior work that characterizes the performance of NoSQL and array databases with real scientific applications.

5. CONCLUSIONS

In this paper, we present an evaluation of NoSQL and array databases using standard benchmarks and scientific data sets. Our results demonstrate that it is important to understand the query load distribution and carefully determine the right parameters to get the optimal performance. Our results also indicate that the choice of client side tools

might significantly impact performance and must be considered when making choices.

It is important to note that many of these products are evolving and the performance obtained is improving. The performance is also closely tied to specific hardware and/or software setup. The choice of a database for a particular project might depend on qualitative factors such as the nature of the data, availability of support for the software.

6. ACKNOWLEDGMENTS

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors would like to thank Elif Dede.

7. REFERENCES

[1] P. G. Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 963–968, New York, NY, USA, 2010. ACM.

[2] Cassandra Website. `http://cassandra.apache.org/`.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26:4:1–4:26, June 2008.

[4] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow.*, 1(2):1277–1288, Aug. 2008.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM.

[7] E. Dede, M. Govindaraju, D. Gunter, R. Canon, and L. Ramakrishnan. Semi-structured data analysis using mongodb and mapreduce: A performance evaluation. In *Proceedings of the 4th international workshop on Scientific cloud computing*, 2013.

[8] T. Dory, B. Mejías, P. Van Roy, and N.-L. Tran. Measuring elasticity for cloud databases. In *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 154–160, 2011.

[9] D. Gunter, S. Cholia, A. Jain, M. Kocher, K. Persson, L. Ramakrishnan, S. P. Ong, and G. Ceder. Community accessible datastore of high-throughput calculations: Experiences from the materials project. In *5th workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, 2012.

[10] HBase Website. `http://hbase.apache.org/`.

[11] Mongodb website. `http://www.mongodb.org/`.

[12] L. Ramakrishnan and R. S. Canon. Experiences in building a data packaging pipeline for tomography beamline. In *Big Data Analytics: Challenges and Opportunities (BDAC-13) Workshop, Held in conjunction with Supercomputing*, 2013.

[13] SciDB Website. `http://www.scidb.org/`.

[14] D. Shoshani, Arie amd Rotem. *Scientific Data Management: Challenges, Technology, and Deployment*. CRC Press, 2009.

[15] The TPC-H Benchmark. `http://www.tpc.org/tpch/`.

[16] A. Verma, X. Llora, S. Venkataraman, D. Goldberg, and R. Campbell. Scaling ecga model building via data-intensive computing. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.