NoSQL Approach to Large Scale Analysis of Persisted Streams

Khalid Mahmood, Thanh Truong, Tore Risch

Department of Information Technology, Uppsala University, Uppsala 75237, Sweden
{khalid.mahmood, thanh.truong, tore.risch}@it.uu.se

Abstract. A potential problem for persisting large volume of streaming logs with conventional relational databases is that loading large volume of data logs produced at high rates is not fast enough due to the strong consistency model and high cost of indexing. As a possible alternative, state-of-the-art NoSQL data stores that sacrifice transactional consistency to achieve higher performance and scalability can be utilized. In this paper, we describe the challenges in large scale persisting and analysis of numerical streaming logs. We propose to develop a benchmark comparing relational databases with state-of-the-art NoSQL data stores to persist and analyze numerical logs. The benchmark will investigate to what degree a state-of-the-art NoSQL data store can achieve high performance persisting and large-scale analysis of data logs. The benchmark will serve as basis for investigating query processing and indexing of large-scale numerical logs.

Keywords. NoSQL data stores, numerical stream logs, data stream archival.

1 Introduction

The data rate and volume of streams of measurements can become very high. This becomes a bottleneck when using relational databases for large-scale analysis of streaming logs [1, 2, 3, 4]. Persisting large volumes of streaming data at high rates requires high performance bulk-loading of data into a database before analysis. The loading time for relational databases may be time consuming due to full transactional consistency [5] and high cost of indexing [6]. In contrast to relational DBMSs, NoSQL data stores are designed to perform simple tasks with high scalability [7]. For providing high performance updates and bulk-loading, NoSQL data stores generally sacrifice strong consistency by providing so called eventual consistency compared with the ACID transactions of regular DBMSs. Therefore, NoSQL data stores could be utilized for analysis of streams of numerical logs where full transactional consistency is not required.

Unlike NoSQL data stores, relational databases provide advanced query languages and optimization technique for scalable analytics. It has been demonstrated in [8] that indexing is a major factor for providing scalable performance, giving relational databases a performance advantage compared to a NoSQL data store to speed up the analytical task. Like relational databases, some state-of-the-art NoSQL data stores (e.g. MongoDB), also provide a query language and both primary and secondary indexing, which should be well suited for analyzing persisted streams.

To understand how well NoSQL data stores are suited for persisting and analyzing numerical stream logs, we propose to develop a benchmark comparing state-of-the-art relational databases with state-of-the-art NoSQL data stores. Using the benchmark as test bed, we will then investigate techniques for scalable query processing and indexing of numerical streams persisted with NoSQL data stores.

2 Application Scenario

The Smart Vortex EU project [1] serves as a real world application context, which involves analyzing stream logs from industrial equipment. In the scenario, a factory operates some machines and each machine has several sensors that measure various physical properties like power consumption, pressure, temperature, etc. For each machine, the sensors generate logs of measurements, where each log record has timestamp *ts*, machine identifier *m*, sensor identifier *s*, and a measured value *mv*. Relational databases are used to analyze the logs by bulk-loading them in table *measures (m, s, ts, mv)* which contains a large volume of data logs from many sensors of different machines [3, 4].

Since the incoming sensor streams can be very large in volume, it is important that the measurements are bulk-loaded fast. After stream logs have been loaded into the database, the user can perform queries to detect anomalies of sensor readings. The following query analyzes the values of *mv* from sensor logs for a given time interval and parameterized threshold.

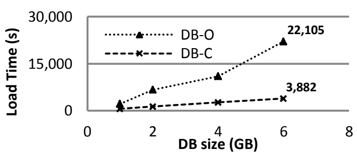
```
SELECT * FROM measures WHERE m = ? AND s = ?
     AND ts > ? AND ts < ? AND mv > @th
```

In order to provide scalable performance of the query, we need an index on the composite key of *m*, *s*, *ts* and a secondary B-tree index on *mv*.

3 Challenges in Analyzing Large Scale Persisted Streams

Analysis of large-scale stream logs in the above application scenario poses the following challenges (C1 to C6) in utilizing relational and NoSQL data stores.

C1. Bulk-loading: In relational DBMSs, the high cost of maintaining the indexes and full transactional consistency can degrade the bulk-loading performance of large volume of data logs. The loading performance of a relational DBMS from a major commercial vendor, called *DB-C* and a popular open source relational database, called *DB-O* for 6GB of data logs is shown in Fig. 1. It took more than 1 hour in a high performance commodity machine for the state-of-the-art commercial DBMS, DB-C to bulk-load data logs consisting of around 111 million sensor measurements. Some of the data logs consist of more than a billion sensor measurements, which require high-performance bulk-loading. To boost up the performance, weak consistency level of a NoSQL or relational database can be utilized.



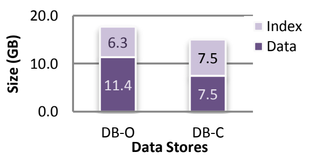


Fig. 1. Bulk-loading performance of 6GB logs **Fig. 2.** Index and database size of 6GB of logs

C2. Index size: Fig. 2 shows the index and database sizes for 6GB of stream logs loaded into the two DBMSs. The size of the index created in both relational DBMSs was larger than the size of the original logs. For high performance and scalable analysis of typical stream logs, hundreds of gigabytes of memory is required in our application. It is interesting to see whether the state-of-the-art NoSQL data store can provide memory efficient indexing strategies. Novel indexing techniques can also be incorporated in order to provide a memory efficient indexing for analyzing persisted streams.

C3. Indexing strategies: Unlike relational databases and MongoDB, most NoSQL data stores do not provide both primary and secondary indexing, which are essential to scalable processing of queries over data logs. Some NoSQL data stores such as Hbase, Cassandra, Memcached, Voldemort, and Riak do not provide full secondary indexing, which is needed for queries having inequalities over non-key attributes. CouchDB has secondary index, but queries have to be written as map-reduce views [7], not transparently utilizing indexes.

C4. Query processing: Unlike relational databases, most NoSQL data stores do not provide a query optimizer. Some NoSQL data stores, e.g. MongoDB, provide a query language that is able to transparently utilize indexes. However, the sophistication of query optimizer still needs to be investigated for scalable analysis of data logs.

C5. Advanced analytics: Relational DBMS features for advanced analytics such as joins or numerical expressions is limited in NoSQL data stores. Therefore, it needs to be investigated how advanced numerical analytics over large-scale data logs could be performed by NoSQL data stores.

C6. Parallelization of data: NoSQL data stores have the ability to distribute data over many machines, which can provide parallel query execution. However, typical queries for analyzing data logs can generate lots of intermediate results that need to be transferred over the network between nodes, which can be a performance bottleneck. Therefore, the performance of both horizontal and vertical partitioning of distributed NoSQL data stores can be investigated for query execution over numerical logs.

4 Proposed Work

There are several investigations that can be performed for large-scale analysis of numerical stream logs.

Stream log analysis benchmark: Typical TPC benchmarks [9] such as TPC-C, TPC-DS, and TPC-H are targeted towards OLTP or decision support, not for log analysis. To benchmark data stream management systems, the Linear Road Benchmark (LRB) [10] is typically used. However, LRB does not include the performance of persisted streams. Analysis of large-scale data logs often requires scalable queries (e.g. [3, 4]) over persisted numerical logs, which should be the focus the benchmark. In the benchmark, several state-of-the-art NoSQL data stores should be compared with relational DBMSs to investigate at what degree NoSQL data stores are suitable for persisting and analyzing large scale numerical data streams. The performance of bulk-loading capacities of the databases w.r.t. indexing and relaxed consistency should be investigated in the benchmark. The queries should be fundamental to log analyses and targeted to discover the efficiency of query processing and utilization of primary and secondary index of the data logs. The benchmark should analyze and compare the performance differences of loading with relaxed consistency, index utilization, and query execution for both NoSQL and relational databases, which can provide the important insights into challenges C1, C3, C4, and C6.

Query processing: Supporting advanced analytics using a complete query language with a NoSQL data store requires the development of query processing techniques to compensate for the limitation of the NoSQL query languages, for example lack of join and numerical operators. The push-down of query operators as generated parallel server side scripts should be investigated. Furthermore, it should be investigated how domain indexing strategies [11] in a main memory client-side database (e.g. Amos II [12] developed at UDBL of Uppsala University and [13]) can improve performance of numerical data log analyses of data retrieved from back-end NoSQL databases. These can provide the insights of the challenges C2 and C5.

References

1. Smart Vortex Project, http://www.smartvortex.eu/
2. Zeitler, E., Risch, T.: Massive Scale-out of Expensive Continuous Queries. In: VLDB (2011)
3. Truong, T., Risch, T.: Scalable Numerical Queries by Algebraic Inequality Transformations. In: DASFAA (2014)
4. Zhu, M., Stefanova, S., Truong, T., Risch, T.: Scalable Numerical SPARQL Queries over Relational Databases. In: LWDM Workshop (2014)
5. Doppelhammer, J., Höppler, T., Kemper, A., Kossmann, D.: Database performance in the real world. In: SIGMOD (1997)
6. Stonebraker, M.: SQL databases v. NoSQL databases. Comm. ACM. (2010)
7. Cattell, R.: Scalable SQL and NoSQL data stores. ACM SIGMOD Rec. 39 (2011)
8. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., Dewitt, D.J., Madden, S., Stonebraker, M.: A Comparison of Approaches to Large-Scale Data Analysis. In: SIGMOD (2009)
9. Council, T.P.P.: TPC Benchmarks, http://www.tpc.org/information/benchmarks.asp
10. Arasu, A., Cherniack, M., Galvez, E., Maier, D., Maskey, A.S., Ryvkina, E., Stonebraker, M., Tibbetts, R.: Linear road: a stream data management benchmark. In: VLDB (2004)
11. Gaede, V., Günther, O.: Multidimensional Access Methods. ACM Comput. Surv. 30 (1998)
12. Risch, T., Josifovski, V., Katchaounov, T.: Functional Data Integration in a Distributed Mediator System. In: Gray, P.M.D., Kerschberg, L., King, P.J.H., Poulovassilis, A. (eds.) The Functional Approach to Data Management (2004)
13. Freedman, C., Ismert, E., Larson, P.-Å.: Compilation in the Microsoft SQL Server Hekaton Engine. In: IEEE Data Eng. Bull. 37, 1 (2014)