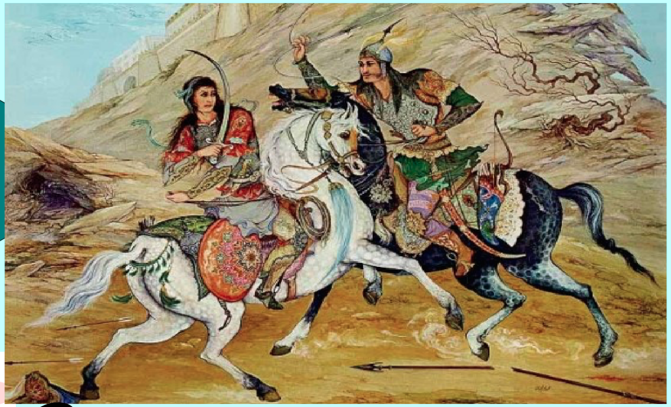
GordSystem* : Approximate Query Processing for Very Large Streaming Data

Samira Elhami Moghadam,
Mehran Sharghi, Reza Azmi
OSSL LAB, Tehran, Iran



1 INTRODUCTION

In today's world, companies face a challenge to manage and analyze their large amount of data. Most of these data are streaming data, that arrive in a stream or streams, and if they are not processed immediately or stored, then they are lost forever. Managers need to query these data to **make real-time decisions**. But scanning or analyzing these amounts of data leads to **extensive execution time**. Therefore *approximate query processing (AQP)*[1] techniques which allow for fast processing of large amounts of data by trading result accuracy for response time would be an appropriate choice. Sampling is one of the general techniques in AQP. In this study,we present **GordSystem**, a distributed approximate query engine that supports continuous SQL-based queries [2] on streaming data. GordSystem answer query **2 times faster** than a combination of Apache Spark Streaming [3] and Apache Spark SQL [4], within an **error rate 1- 6%.**

2 METHODS

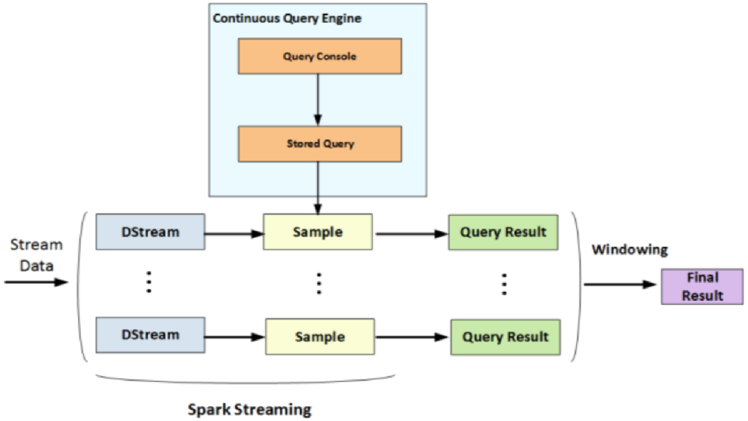
* Based on *DStream* (Discretized Streams), the building block of Apache Spark Streaming.

* Because each DStream contains a small amount of streaming data, its query result is insufficient. Therefore Gordsystem utilized a **state-of-the-art window** *operator which groups all of the records from a range of past time intervals into a single DStream and merges results at the query level.*

* Gives **real-time responses** with *statistical error guarantees*

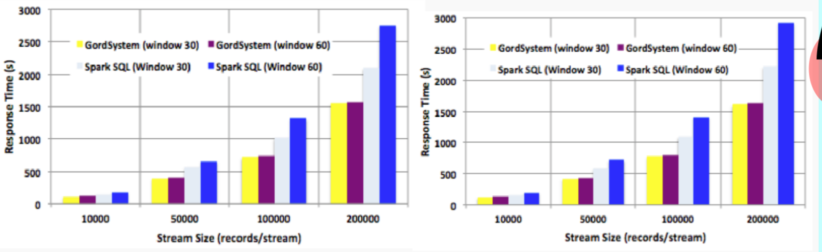
* Tested against NEXMark benchmark [5], server log from 1998 World Cup [6] and 15 cluster nodes

2-1 GORDSYSTEM ARCHITECTURE



1) User defines SQL queries through *Countinuous Query Engine*.
2) The queries will then stored in *Stored Query* file.
3) Upon receiving a stream of data, DStreams are created and stored across the cluster.
4) The *Sampling module* creates a sample from streams.
5) The *query* is then executed on this sample.
6) The *Window operator* merges results at *query level*.

3-1 Comparison of Query Response Times on Spark SQL and GordSystem



Sample contains 500 records

Sample contains 1,000 records

3 PERFORMANCE

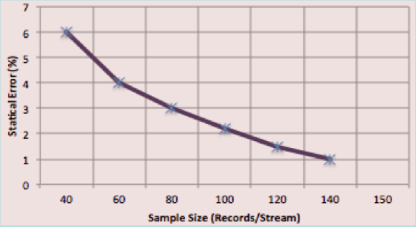
To evaluate the approximate query processing module, different sample sizes including 500 and 1000 records/stream were considered.In all cases, GordSystem **outperforms** the *Spark SQL* (it is faster by a **factor of 2**) because it runs queries on a sample of data, utilizing a *state-of-the-art window technique*.

The entire sampled-based approximate query processing system suffers a trivial statistical error in its results and GordSystem is not an exception.

Every DStream includes 3,000 records.The error rate for these samples are between 1-6 percent.

3-2

Static Error Rate across different sample sizes.



4 CONCLUSION

GordSystem utilizes the sampling-based approximate query processing techniques and a novel Window operator to decrease the query response time over massive amounts of streaming data.

5 REFERENCE

[1] M. Garofalakis, "Approximate Query Processing: Taming the TeraBytes" VLDB, 2001
[2] S. Babu, in Encyclopedia of Datavase Systems, Springer US, 2009, pp. 492-493.
[3] M. Zaharia, T. Das, H. Li, S. Shenker and I. Stoica, "Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters," USENIX, 2012.
[4] "Spark SQL," [Online]. Available: https://spark.apache.org/sql/.
[5] "NEXMark Benchmark," [Online]. Available: http://datalab.cs.pdx.edu/niagaraST/NEXMark/.
[6] "1998 World Cup Web Server Logs," [Online]. Available: http://ita.ee.lbl.gov/html/traces.html.

*GordSystem is derived from GordAfarid, a female legend in Shahnameh poet book.