International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 2 | ISSN : 2456-3307

Query Engine Design and Performance Analysis : A Review

Chemwotie Kipkurui Brian
Department of Computing, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya

ABSTRACT

Distributed real-time computing has been the domain of practical system engineering for many decades. The development of a discipline of real-time programming would allow the construction of programs with analysable and variable timing properties. Such a discipline will need to be built on a well-integrated framework in which different methods are used where appropriate to obtain timing properties to which a high-level assurance can be attached. Time is everything in this current system, everything fast will gain the most valuable achievements especially in the business field. Whoever tries to get the first opportunity to fulfil the market needs will gain most profit. This paper looks into the various query engine models and frameworks that tries to improve on both the design and performance.

Keywords: Distributed, Real-Time Computation, Query Engine

I. INTRODUCTION

We will find many real time systems around us. Their field of implementation range from small to large-scale use, like for instance industrial use or the military. Most of these applications are also safety critical systems that need to be reliable therefore; we need a reliable and fault-tolerant distributed real-time computation system. Real-time computing is any information processing system, which has to respond to externally generated input stimulated within a finite and specified period. We can also define a real-time system as a processing system that processes any information and generates output within a specified time. Storm is an example of a distributed real-time data processing system, which guarantees that the system will continue to operate properly in failure state, and all incoming data will be processed [1].

There has been other notable distributed real-time systems including S4 [2], MillWheel [3], Samza [4], Spark Streaming [5] and Druid [6].With many distributed real-times computation, there are some requirements of real-time processing which are becoming standard for distributed real-time computation systems. For supporting analysis in real-time computation, querying is one of the most used tools.

II. QUERY OPTIMIZATION STRATEGIES

A. Iterative Dynamic Programming (IDP)

Kossman [7] proposed the use of iterative dynamic programming in place of dynamic programming. He argues that IDP is able to produce as good plans as dynamic programming if there are enough resources available, and IDP is, in addition, able to adapt in cases where there are not enough resources available or the query is too complex for dynamic programming.

B. Randomized optimization

Query optimization for relational database systems is a combinatorial optimization problem, which makes exhaustive search unacceptable as the query size grows. Randomized algorithms, such as Simulated Annealing (SA) and Iterative Improvement (II), are viable alternatives to exhaustive search. Ioannidis et al [8] adapted these algorithms to the optimization of project-select-join queries. The author tested them on large queries of various types with different databases, concluding that in most cases SA identifies a lower cost access plan than II.

C. Dynamic Approach

Although. Query optimization is the most critical phase in query processing. Hameurlain et al [9] tries to describe synthetically the evolution of query optimization methods from uniprocessor relational database systems to data Grid systems through parallel, distributed and data integration systems. The author points out a set of parameters to characterize and compare query optimization methods, mainly: size of the search space, type of method (static or dynamic), modification types of execution plans (re-optimization or re-scheduling), level of modification (intra-operator and/or inter-operator), type of event (estimation errors, delay, user preferences), and nature of decision making (centralized or decentralized control).

D. Static Approach

Although. Query optimization is the most critical phase in query processing

III. OTIMIZING QUERY PRACTICE

The following systems are set practices of query optimization on existing systems. The practices have been tested and proven on different systems and all the results recorded and published.

A. JAQL

JAQL is a part system of IBM Big insights to analyze large semi structured datasets in parallel using Hadoop's Map Reduce framework [10]. It has some common features with other data processing languages. JAQL was also developed for scale-out framework architecture such as Pig [11], Hive [12] and DryadLINQ [13].JAQL developed methods that are also implementable in other scale-out framework architectures that have been listed above with some minor modification. JAQL components consists of a declarative scripting language, a compiler and a runtime. JAQL contains a scripting language, a compiler and hadoop's runtime components.

For querying language, JAQL evaluates statements, which are either expressions or assignments. The system in JAQL has an input to produce an output that can feed another expression as input. All of the aggregate functions such filter, join and group by are supported by JAQL. A user can also submit the SQL query that will be translated to JAQL system. Java and the other languages are also supported by JAQL.

For JAQL's compiler, It's designed as a heuristics-base rewrite engine which optimize input scripts applying a set of transformation rules. JAQL will simplify expression to lower level operators so it can be executed by this. Map Reduce framework will evaluate an expression that has been transformed to map reduce function. This rule is the most important rules of JAQL. In the distributed computation, the interpreters evaluates the script locally on the nodes that does the compilation of the script. In order to do parallel execution, JAQL will spawns interpreters to remote nodes using Map Reduce function.

B. JQL

JQL [14] is a java's extension which has a capability for querying the collection of objects. This means the query is applicable on objects in class collections of the system and usable for expressions checking of specific instances types at run-time.

The query engine is allowed by query to run the implementation detail task with abstraction which provided by JQL for handling sets of object, thus the code will be smaller and with permission the query valuator can make a dynamical decision even though the situation keep changing at run time. In program language, run-time execution can be improve by the query optimization strategies, which is come from database domain. They main concept is do the query optimization task at compile-time as many as we can.

Histograms is used by the technique to estimate the selection of join and predicates in a query. The order of query joins and predicates will be ordered according to that estimation. After this technique obtained the plan for query at compile-time, the plan is going to be compiled at run-time. The estimation of errors rate and split merger algorithms are suitable and efficient to maintain the histogram accurately, this is showed by the experimental results which has been done before.

IV.REAL-TIME PROCESSING FRAMEWORK

Real-time processing network has become a new trend in the last couple of years. Most organizations are pursuing implementation of real-time processing systems. With this technology, they want to achieve

what they could not before, real-time analysis to facilitate real-time decision-making.

A. Spark Streaming

Spark is an open-source distributed computing framework that runs on a computer cluster. Spark uses a dubbed Resilient Distributed Datasets (RDD) for repeated query [15] and stored cache datasets in memory. Spark's performance is much faster than Hadoop map reduce which is around 100 times faster for iterative machine application [16]. Spark streaming is a streaming computation as a series of a very small and deterministic batch jobs [17].

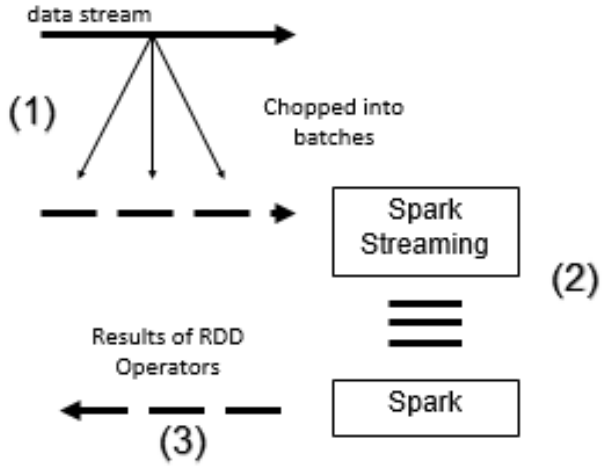

Figure 1. Spark Streaming Process

These are Spark streaming concepts:

1. Dividing live stream into several batches of X seconds.
2. Spark assume each batch of data as RDDs and does the process by using RDD operations.
3. At the end, RDD operation's results are returned in batches.

B. Spark SQL (Shark)

Shark is system that analyze data and an extension of Spark distributed computing framework [18]. This system is combining SQL queries with Spark analytic function at scale and also has a recovery-recovery for query. Shark is able to execute SQL quires faster enough at 100 times than apache hive and Hadoop in machine learning programs. Shark implemented a column-oriented in memory for storage and dynamic strategies for the preplanning mid-query in order to effectively execute the SQL statement.

C. Storm

A distributed real-time computation framework which has a capability to process data in the unbounded streams form [19]. Strom does the same thing with Hadoop except Storm does the real-time processing while Hadoop is a batch processing. Queuing and database technologies are also supported by Storm framework. Streams of data are fed to Storm topology and Storm process those streams in the very random complex ways, stream repartition between each stage of computation is need. Storm support multi-languages (ruby, python, java script, Perl, and PHP). Storm architecture is presented by three nodes which have different functionality.

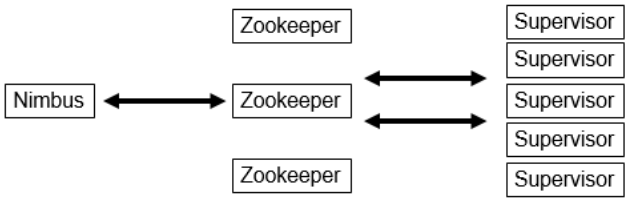

Figure 2 Storm Architecture

a) Nimbus node

Nimbus [20] is a master node that will do the assigning tasks to supervisor nodes, monitoring failures in the cluster and distributing code to be executed around the cluster. Nimbus node is also responsible for monitoring all of the computation and reallocating workers.

b) ZooKeeper

Distributed configuration, synchronized service and naming registry for Storm cluster, Zookeeper [21] responsible for storing all of nimbus and supervisor state in the Znode or in the local disk.

c) Supervisor nodes

Listening and Receiving all distributed task that have been assigned by the nimbus node. Supervisor is able to start/stop the worker processes if necessary. Each Worker in the supervisor nodes will executed a part of the topology, while in Hadoop it called TaskTracker.

D. Storm Trident

Trident [14] is a Storm's extension which provides a simple and easier framework for distributed real-time big data analytics framework. Storm trident is also developed by Twitter.

One of the Twitter's biggest problem is to keep the statistics of how many tweets and tweeted URLs which get retweeted by some millions of followers. Image a famous person whose tweets a URL and has a millions

of followers. Many of his/her followers will do the retweet. Thus how to calculate how many people in tweeter have seen or read the URL? This function called "Top retweet Url's". This feature will display which one is the most tweeted in real-time.

The one and only solution was apache Storm, but with the extension of Trident. Managing this kind of features will became easier with the present of Storm trident. Trident does the simplification for Storm. In the traditional ways of Storm, we must configure a number of Spouts, Bolts and also manage the configuration how the tuple is distributed, which grouping we will use. Tridents exists a lot of feature that will do the complicated part. Trident has four operations that have a different functionality.

1. The local operation will apply to each of partition and there is no need to do network transfer.
2. The operation will be distributed which will cause the stream also being distributed. In this operation, network transfer is involved.
3. Operation for aggregate function which will need a network transfer as part of the operation.
4. Merges and joins.

The fact that using in-memory state as a storing system in Bolt doesn't has fault-recovery or in other word is fault-tolerant. The process in dying nodes will get reassigned by nimbus node while the state can't be retrieved. Although we can use a ticket that has been provided, the thoughtful way is to preserve to a reliable database which is the reason why trident is useful if we need to save the state. Trident does the pre-batch processing to lighten our data store with only one update for each message. Trident also provides an aggregation API.

Although Trident can be used to simplify complex algorithms computation, we must learn how to use trident and use its own function that need to be learned. The other framework which is Spark streaming already provide SQL integration with its system which called Shark. Strom has extension for trident while having no SQL engine query.

E. Spark and Storm Comparison

Although the fact that Storm and Spark streaming are distributed real-time processing, there are several differences that will separate them. Storm and Spark streaming will be compared side-by side and the table below will present their differences.

	Spark	Storm
Origin	BackType, Twitter	UC Berkeley
Implemented in	Scala	Clojure
API language	PHP , Java , Pyhton	Java, Scala
Processing Model,	Arriving events is batched up in the period of shoet time window before the stream is processed	The incoming event will be process in a real-time. There is no waiting time for execution.
Latency	Few seconds	Sub second
Fault Tolerance, Data Guarantees	Has a fault-recovery and statefull computation	Storm will guarantee that the each record is processed once or more. Storm also allows duplicating during the fault-recovery. This will means there is possible mutable state due to two incorrect updates.
Hadoop distribution	HDP , HortonWorks	MapR, Cloudera

V. CONCLUSION

All sections described above are related to this research that we have specially selected. There are several frameworks that have been implemented in a distributed system for real-time computation. Apache Storm and Spark are two of many real-time distributed computation framework that already exist that have gained popularity for its performance for real-time computation. In this paper, we have majorly focused

on Storm and spark engines. Storm has a non-SQL extension called trident whereas Spark have been implemented SQL query engine in their system. SQL is a generic and most used language for queries, with the absence of SQL extension. We have then looked into the differences between the spark and storm engines.

VI. REFERENCES

[1]. A. Toshniwal et al., "Storm @ Twitter," pp. 147–156, 2014.

[2]. L. Neumeyer and B. Robbins, "S4 : Distributed Stream Computing Platform," IEEE Int. Conf. Database Syst.

[3]. S. Chernyak et al., "MillWheel : Fault-Tolerant Stream Processing at Internet Scale," Proc. VLDB Endow., vol. 6, no. 11, 2013.

[4]. J. Samosir, M. Indrawan-santiago, and P. D. Haghighi, "An Evaluation of Data Stream Processing Systems for Data Driven Applications 2 Real-time Data Processing of Big Data," Procedia - Procedia Comput. Sci., vol. 80, pp. 439–449, 2016.

[5]. M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized Streams : Fault-Tolerant Streaming Computation at Scale," Proc. Twenty-Fourth ACM Symp. Oper. Syst. Princ., no. 1, pp. 423–438, 2013.

[6]. F. Yang, E. Tschetter, G. Merlino, X. Léauté, N. Ray, and J. Bieber, "Druid A Real-time Analytical Data Store," Proc. 2014 ACM SIGMOD Int. Conf. Manag. Data, pp. 157–168, 2014.

[7]. D. Kossmann and K. Stocker, "Iterative Dynamic Programming : A New Class of Query Optimization Algorithms," vol. 1, no. 212, 1999.

[8]. Y. E. Ioannidis, "RANDOMIZED ALGORITHMS FOR OPTIMIZING LARGE JOIN QUERIES + ALGORITHMS," ACM, pp. 312–321, 1990.

[9]. A. Hameurlain and F. Morvan, "Evolution of Query Optimization Methods," Trans. on Large-Scale Data- & Knowl.-Cent. Syst., vol. 33, no. 0, pp. 211–242, 2009.

[10]. K. S. Beyer, R. Gemulla, A. Balmin, E. J. Shekita, C. Kanne, and F. Ozcan, "Jaql : A Scripting Language for Large Scale Semistructured Data Analysis," Proc. VLDB Endow., vol. 4, no. 12, pp. 1272–1283, 2011.

[11]. C. Olston, B. Reed, R. Kumar, and A. Tomkins, "Pig Latin : A Not-So-Foreign Language for Data Processing," ACM SIGMOD, 2008.

[12]. A. Thusoo et al., "Hive - A Warehousing Solution Over a Map-Reduce Framework," ACM VLDB Endow., 2009.

[13]. Y. Yu et al., "DryadLINQ : A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language," USENIX Symp. Oper. Syst. Des. Implement., pp. 1–14, 2009.

[14]. D. Willis, D. J. Pearce, and J. Noble, "Efficient Object Querying for Java," Eur. Conf. Object-Oriented Program., vol. 4067, pp. 28–49, 2006.

[15]. M. Zaharia et al., "Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing," Proc. 9th USENIX Conf. Networked Syst. Des. Implement., 2012.

[16]. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," 26th Symp. Mass Storage Syst. Technol., pp. 1–10, 2010.

[17]. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark : Cluster Computing with Working Sets," Proc. 2nd USENIX Conf. Hot Top. cloud Comput., 2010.

[18]. R. S. Xin et al., "Shark : SQL and Rich Analytics at Scale," Proc. 2013 ACM SIGMOD Int. Conf. Manag. Data, pp. 13–24, 2013.

[19]. R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," IEEE Cloud Comput., 2015.

[20]. E. Feller, L. Rilling, and C. Morin, "Snooze : A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds," 12th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput., 2012.

[21]. P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper : Wait-free coordination for Internet-scale systems," USENIX Annu. Tech. Conf., vol. 8, 2010.