Data Optimization using Apache Flink

Vikas S, Thimmaraju S N

Abstract: ***Map Reduce, Flink, and Spark, also become more popular in the processing of big data lately. Flink will be an open platform Big Data processing system for Apache-powered batch storage and streaming of data. Flink's query optimizer is constructed for historical information processing (batch) based on parallel storage systems approaches. Flink query query optimizer interprets the questions into jobs of different tasks that are regularly sent. Therefore, taking advantage of task similarities should prevent redundant computation. In this article, the multi-demand optimization model for Flink, Flink was planned and designed on Flink Software Stack's top priority. It's thought-about as an associate in Apache Flink's nursing add-on to maximize multi-demand information sharing. The Flink system takes advantage of option operators ' information sharing resources to reduce overlap and duplication of multi-query in-network information movement. Research findings show that the leveraging of shared option operations in vast information on multiple requests would offer promising time to perform queries. Therefore, in the stream phase, Without doubt the Flink approach can be used to boost application performance over time periods.***

Keywords: ***BigData, Parallel Processing, Flink, batchprocessing, selection predicates.***

I. INTRODUCTION

Big data emerged in the digital data age as an innovation space to address the enormous amounts of information produced precisely. Usually the word "big data" is used to characterize quantities, size and speed of information (M. Chen et al., 2014). Such information generally encompasses giant Quantities of quasi-structured and unstructured data types that are very difficult to store, control and analyze using ancient information technology. Huge data is considered to be a viable technology (Manyika et al., 2011; McAfee et al., 2012; Rothstein, 2015) and is useful to different organizationsIn general, telecommunications companies use large quantities of information to monetize traffic information (Yazti & Krishnaswamy, 2014). Firms use large-scale data to identify the ANoptimum maintenance period for replacement elements before they fail, time and customer satisfaction (Zhu et al., 2014). Pharmaceutical companies use extensive knowledge to accelerate the development of medicines and provide highly customized care (Greenspan & Valkova 2014). To protect patients from cyber attacks, governmental agencies use massive data (Kim et al., 2014; Lyon, 2014).It is very important to extract important and valuable knowledge from Brobdingnagian datasets to provide new products and services and to raise the standard of either the dominant ones (Kambatla et al., 2014; Talia, 2013). Massive information keeps parallel to an excessively distributed fashion need process (Philip Chen & Zhang, 2014), These new information and technology can be thoroughly exploited at an affordable interval of your time. In a broad range of applications, successful technology has been implemented (Hsu, 2014), For example, data processing, knowledge analysis, computer programming and scientific computing. Nevertheless, these applications have challenged the processing of massive data due to the complexity of the information to be processed and the quantifiability of the underlying algorithms that enable these processes. (Labrinidis & Jagadish, 2012). Map Reduce is probably the primary style model for the storage of the current information on a large scale (Dean & Ghemawat, 2008) (Dean & Ghemawat, 2010) primarily attributable to its vital options embodying quantification, fault tolerance, simple programming, and adaptability. MapReduce is now primarily used to communicate decentralized computations on large amounts of information and a wide-scale processing system on clusters of object databases. Cluster computing provides high performance in distributed system environments, as well as PC power, storage, and network communications (Bollier & Firestone, 2010), but cluster computing will provide a hospitable environment for information growth.As a result of the numerous massive information coming across the globe, several massive models of information, systems and new technologies have been developed to provide additional storageMulti-processing and analyzing various large and diverse sources in real time. Furthermore, new solutions are being built to ensure information privacy and security. These systems provide a lot of flexibility, scalability and performance compared to ancient technologies. Moreover, due to the technological advance of the property, the value of most hardware storage and process solutions is steadily falling (Purcell, 2013). Many models, programs, software, hardware and technologies are designed and projected to extract data from massive information. For large data applications, they try to confirm a lot of correct and reliable results. Nonetheless, distinguishing between various methods in such surroundings should be time new-consuming and difficult. In addition, there are several criteria to consider: Compatibility of code, complexity of planning, cost, performance, value, reliability, risk support and protection. There are many massive data studies in the literature, but many of them seem to target algorithms and tackle customized massive information methods rather than technologies (Ali et al., 2016; Chen and Zhang, 2014; Chen et al., 2014a).

Revised Manuscript Received on December 08, 2019

Vikas S, Assistant Professor, CSE Department, VTU PG Centre, Mysuru, Karnataka, India

Thimmaraju S N, Professor, CSE Department, VTU PG Centre, Mysuru, Karnataka, India

MapReduce framework has been widely used to effectively handle such massive information. MapReduce has existed for the previous couple of years because it is the favored computing model for parallel, batch-style and large-scale data processing (Apache Flink (2016a). Once successfully used by Google, MapReduce gained its quality. In addition, it is a scalable and fault-tolerant methoding device that provides versatility in processing large voluminous information in parallel with many low-end computing nodes (Babu, S. and Herodotou, H. (2013). MapReduce is becoming present due to its usability, quantifiability and sensitivity to faults, gaining major traction from every business and educational world. High performance will be achieved by breaking the process into tiny work units running parallel across multiple cluster nodes (Camacho-Rodríguez (2014)). Within the MapReduce model, information in multiple machines and information is drawn as (key, value) pairs at the starting partitions is a distributed classification system (DFS). The MapReduce framework performs the most on a single master machine wherever we tend to be able to pre-process the input file before map functions are known as or post-process the output of back scale functions. A combination of map and scale back functions is also dead once or varied times because it depends on the associate degree application characteristics (Tzoumas, K. (2015)).

Hadoop might be MapReduce's trendy open-source implementation for analyzing massive datasets. This uses a distributed file system at the user level to handle cluster-wide processing resources (Zhang, C-Y. (2014)). However, the system produces unintended acceleration with small data sets, but it produces an affordable speed with a wider range of information that increases the number of computer nodes and reduces the execution time by half an hour compared to traditional data mining process and alternative processing techniques (Ghemawat, S. (2008)).

Apache Hadoop and Apache Spark are parallel systems samples which provide environments for Development of the programming template for Map Reduce. While Hadoop Map Reduce (Zhou, W. (2015)) runs on the computer, due to increased I / O operations, sometimes it is slow and expensive.

In contrast, as Spark MapReduce (Touriño, J. (2016) is implemented in the cluster's computer node memory, it offers a simpler and cost-effective approach to deploying scalable concurrent and distributed algorithms for large-scale knowledge processing.

Well we just simply present an efficient flink-based algorithm even in this article for the mining sequence from relatively large repositories of breast cancer. We generally apply our algorithm again to the breast cancer datasets to check the efficacy of our algorithm for big data analysis. The major contribution to this paper is our scalable decentralized frequency mining technique, which is a timely imitation of MapReduce's programming and parallel processing system on a Apache flink platform to mine series from large data sequences. Here they concentrate on transaction information, structure encoding, key / value information processing where Requests are translated into information levels, including a collection of data processing on batch processing.

A Big information project, many steps are recommended as shown in Fig.1: First, the proper problem should be chosen. There are three types of issues. The primary quite drawback has al-ready been resolved with traditional technique and there's no have to be compelled to use huge information technologies. The second quite downside is impossible to be resolved with current technologies. We must always specialise in the third quite drawback that's resolvable with current huge information technologies. Second, we'd like to get the information by sen-sors, monitors, molecular identification or extract the information from pub-lic databases/sources when putting in place a practical goal. Third, we'd like to try to to information pre-processing to get clean and significant data. information pre-processing could be a vital step for the success of a big information project.

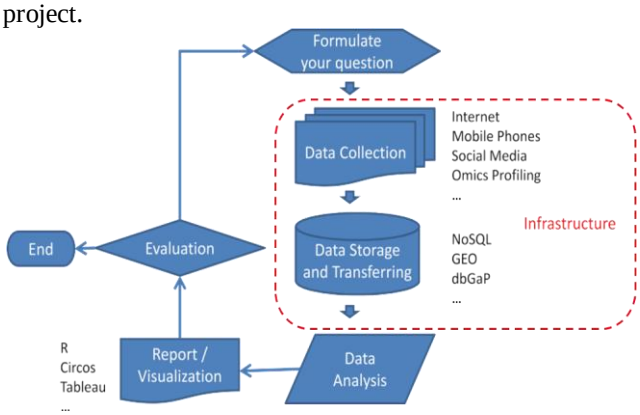

Fig 1: The workflow of a classic Big Data project

Large information could be divided into completely distinct classes predicated on five aspects: data sources, type of document, knowledge inventory, data storage and data serial (see Figure 2) (Hashem et al., 2015).

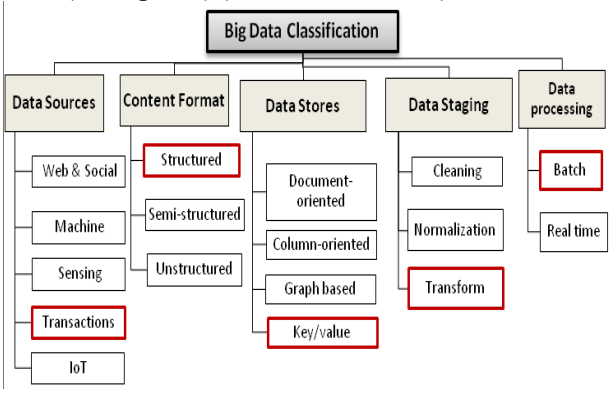

Fig 2: Big data classification

1. RELATED WORKS

Several scientific research efforts were generated throughout the field of Hadoop-based device setup to automate massive information processing, in general. All research activities in the study will be loosely divided into two categories; co-occurringand non-competitive. The same is true of the relative databases (Olston et al. 2008a).

It attempts to find common components between specific queries, such as scanning, computing, shuffling, etc. (Wang and Chan, 2013). The non-competitive is close to materialized display strategies. It materializes and uses the calculation's intermediate and final effects to respond to questions (Elghandour and Aboulnaga, 2012).

MRShare may be a simultaneous distribution mechanism wherever the price of I / O prevails (Nykielet al., 2010).

The interactive possibilities for exchanging maps, map creation and map functions are therefore taken into account. Nevertheless, the relaxed MRShare calms and generalizes the conflicting questions in order to extend sharing incentives over one job (Wang and Chan, 2013).
The mutual map input monitor and map production are studied in accordance with relaxed MRShare, and in the sense of MapReduce, algorithms are implemented to pick a computational scheme for a batch of jobs. In the relaxed MRShare, other methods of optimization (i.e., GGT and MT) are also implemented. In addition, Sahal et al. (2016) provides a comparative analysis of MRShare and relaxed MRShare approaches using predicate-based filters on MapReduce. In step with the comparative results, the relaxed MRShare technique was found to outperform MRShare methodology for sharing information on MapReduce predicate-based filters with appropriate query execution period.
The comparable work, on the other hand, discussed the predicate-based filters of Map Reduce that considered huge knowledge analytical tool with in-disk computation, while this dissertation provides a broad comparative of Flink's predicate-based filter that were considered in-memory computing huge information analytical system.
ReStore platform is among the non-competitive exchanging systems aimed at maximizing the effects of question processing victimization on top of Pig (Elghandour and Aboulnaga, 2012). It uses heuristics algorithmic software to pick even for the complete or part of the map the correct materialized results and rescale the output for each work. The materialized performance provided by the ReStore process will not be recycled in the case of non-perennial request workloads that require overhead storage.
A Lefevre et al. research considers the reuse of previous results stored as materialized views by using MapReduce's intermediary results due to failure resilience wherever Hive's linguistic new user-defined functions (UDF) models were used to efficiently change views if future queries could be evaluated more quickly. then on the other hand, a mechanism, SharedHive, is planned to transform a HiveQL collection of sharing scanning and computing tasks into new optimized query sets (Dokeroglu et al., 2014). For Pig scripts, PigReuse was suggested to classify and recycle CSEs that occur in Pig Latin scripts to tackle the reused-based optimization. Instead, for the most successful ones to be implemented, a cost-based search approach is chosen a linear process problem solver is used to solve this problem (Camacho-Rodríguez et al. 2014). Through extensive knowledge of completely and partly reused ways to take advantage of the gross roughness, the moth template is also suggested. The basic concept of the theoretical moth model is the horizontal analysis of the coarse-grained reused opportunity (i.e., non-equal tuple size) and the horizontally non-uniform distribution of knowledge (i.e., tuple number). (Sahal et al., 2017).

BATCH AND STREAM PROCESING FLINK-SYSTEM

A description of the processing of flink batch and stream processing is given in this paragraph. Then we will discuss the proposed Flink-System.

3.1 Flink batch processing overview

Apache Flink is a partner in Apache massive information analytics platform that enforces a widespread data flow engine to execute massive data analytics for each channel and batch (Carbone et al., 2015). This tends to follow a model for large-scale knowledge processing in-memory computing. In-memory processing refers to using direct storage rather than just disks (i.e. MapReduce) to handle the streaming or batching operation. Thus, Application reliability is often improved through processing and storing information straight from RAMHowever, by storing unlimited data flows for the moment of data analysis, the overhead of disk access is often reduced. The storage of Flink batches is addressed over certain static information in conjunction with this article. For information purposes, delimited information such as conventional data warehouses could be an unique case of similar unbounded information flow, thereby moving traditional information warehouse systems stored on the hard disk to an analytical model in memory like a Flink (Carbone et al., 2015)
In this article, batch processing is called a special streaming case wherever the flow is small and thus record sequence and duration may not matter (i.e. all records belong directly to a single all-encompassing window). In particular, Flink manages batch processing by optimizing its output using a query optimizer and adding memory-free block operators (Mohammed et al., 2016).
Apache Flink offers 3 forms of APIs for extreme data upload, batch processing, and SQL analysis: DataSet API, Table API and DataStream API (Apache Flink, 2016a). The DataStream API involves translating and combining data stream windows to keep the state and partition retrievable. In contrast to Flink, the DataSet API facilitates the storage of batches over static datasetsIn addition, the DataSet apis Flink batch processing utilizes advanced information systems, Algorithms and operators such as being part of, categorizing and using various methods of scheduling. Recently, Flink has provided associate API, specifically Table API, to specify SQL-like expressions for relative streaming and batch processing for victimization of operations. The Flink Table API enables developers to create down a relative table abstraction list of their queries. . related tables will be generated from external information sources or from existing Data sets and Data Streams wherever related operators are included, such as collection, aggregation and tables (Apache Flink, 2016b, 2016c).

3.2 Flink Stream processing overview

Flink's Data stream API implements the complete stream analytics framework on the prime platform of Flink, as well as time management mechanisms like those out of-order event processes, window formation, and customer-defined status monitoring and updating. The stream API is based on a DataStream concept, an infinite (possibly unbounded) set of components of a specific type. While Flink's platform already embraces pipeline information transfers, Continuing state-of - the-art operators and also a failure detection system for continuing state-of - the-art changes, superimposing a stream processor is largely about applying a windowing approach and a state-of - the-art interface. These are opaque to runtime, as noted, which treats windows as merely associating state-of - the-art operators with implementation.
Within the API in Flink State, unique associates are created by supplying: I user interfaces or annotations to dynamically record explicit local variables at an operator's range at intervals, and ii) an operator-state abstract to define divided Key-value

conditions and operations correlated with them. Users even can plan how to store the condition and check the system's State Backend abstractions. This allows extremely versatile administration of custom state in streaming applications. The check-pointing feature of Flink ensures that any recorded state is robust with precisely once semantic updates.

Apache Flink integrates windowing into a state-of - the-art operator controlled by a dynamic statement consisting of 3 processes: a window assigner associating optionally a trigger and an evicter. All three processes can be chosen from the a pool of commonly predefined implementations (e.g. sliding time windows) or can be expressly stated by the user (e.g. user-defined functions).

Breast cancer is just one way that a stream-based architecture with proper message transmission (MapR Streams) and a scalable, high-performance stream processor (Flink) can accommodate a variety of different types of stream-based implementations.

3.3 Highly-Available Flink Clusters

The check-pointing and recovery methods used by Flink avoid data loss only in the case of a job or employee error. But Flink needs a sufficient quantity of process slots to restart the associated application. A streaming program can be implemented with a total parallelism of eight due to a Flink setup with four Task Managers granting 2 slots each. If one of the task managers fails, the amount of available slots will be reduced to six which is not sufficient to recover a streaming application with eight parallels. This problem can be solved for complete cluster setups by getting standby project managers who can take over the job of failed staff. New Task Manager processes can be started mechanically in cluster setups with resource managers such as YARN and Apache Mesos.Link to all available Task Managers and query ZooKeeper's storage locations to get the Job Map, the JAR file, and all state handles from the remote storage of the last checkpoint of all running applications. Then restart all applications and reset to the last completed checkpoints the status of all their tasks using the recovered Job Graphs and job state handles.A stand-alone cluster configuration that is highly accessible requires at least two job managers, one active and one or more stand-by masters. Flink selects the active Job Manager using ZooKeeper. If Flink runs on a resource manager like YARN or Apache Mesos, there is no need for stand-by masters because a new Job Manager is started automatically.

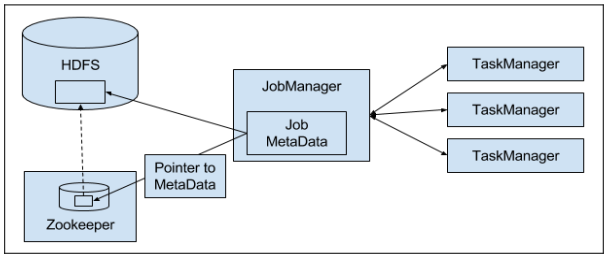

Fig 3. Highly-available JobManager setup with Apache ZooKeeper

1.4 Type of implementation of the flink framework

The proposed Flink architecture optimizes multi-query efficiency by using the relaxed MRShareand MRShare strategies for sharing information. You may describe the measures of the suggested Flink system given below:

1. The predicate extractor module collects the test queries (i.e. the queries sent by broad information analysts) and analyzes them to suit standard SQL language. Operators are then derived attribute names and predicates from the relationship names. Subsequently, interpreted data queries are clustered to help extracted relationships, attributes, and forecast operators in the multi-demand and multi-demand cluster. The mutual multi-demand cluster that carries entirely different groups according to the information operators of the shared option.
2. The replicated multi-demand optimizer module estimates compare or overlap information sharing chances (i.e. reciprocal predicate choice operators) in each shared multi-demand cluster using the MRShare cost model and relaxed MRShare techniques (Nykiel et al., 2010)
3. A request editor module edits the shared queries and creates the optimized multi-demand execution configuration which takes into account the equality and overlap of multi-demand data sharing.
4. The Flink-System discloses the configured multi-query execution attempt to Apache Flink. The Flink Table API interprets the queries as predictions of choice between the optimized multi-query project in a list of relative operators. Such operators have been built into such a list of batch processing tasks transmitted throughout the native system to Flink's runtime environment Flink reads information from a large information network and performs optimized jobs comparable to the lower information volume. The final results of the goal will be sent back to researchers of big data. While the algorithmic program one does not lose its generality, it shows the intended Flink system's pseudo code adapted to MRShare victimization and relaxed MRShare techniques.

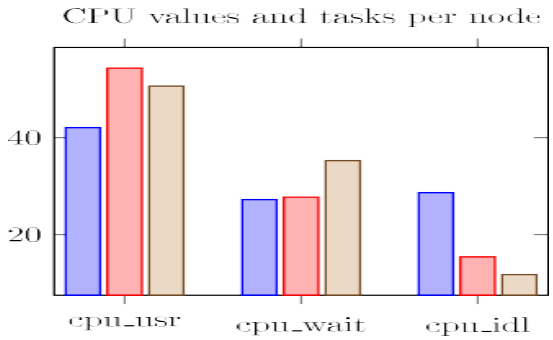

Fig 4: CPU Utilization using support vector machine implementation in Flink.

Algorithm 1: Flink-System

Input: Finput = [Q1, Q2, …, Qn]
Output: FMRShareoutput = [Q1, Q2, …, Qm]
FRelaxedMRShareoutput = [Q1, Q2, …, Qm]
// Step 1: Predicates extractor
Flinkparsed = ParseQuery(Finput)
FlinkPredicates= ExtractPredicates(Flinkparsed)
// Step 2: Sharing classification
FSharedGroup=GetSharedGroup(FlinkPredicates)
// Step 3: Reused-based figurer
FMRShare = MRShare(FSharedGroup)
FRelaxedMRShare=RelaxedMRShare(FSharedGroup)
// Step 4: redaction Flink set up for each F in FMRShare do
F′ = RewiteQuery(F)
FMRShareoutput = FMRShareoutput ∪ F′

for each Q in QRelaxedMRShare do
F′ = RewiteQuery(F)
FRelaxedMRShareoutput=FRelaxedMRShareoutput ∪ F′
// Step 5: Generate Flink set up
FMRShareoutput=FMRShareoutput ∪FnonSharedGroup
FRelaxedMRShareoutput=FRelaxedMRShareoutput∪
FnonSharedGroup

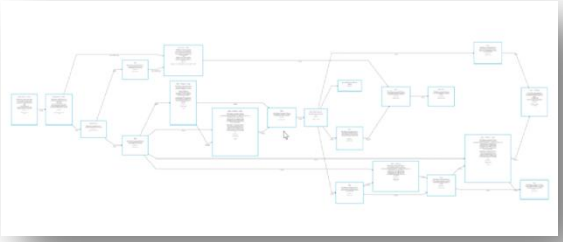

Fig 5: System Architecture in Flink IDE

II. EXPERIMENTAL RESULTS

It tests the efficacy of the expected batch processing of the Flink Framework and the stream processing that was implemented in Java. We appear to show the effectiveness of Flink for parameter knowledge about the variety of tuplesA comparative analysis was commonly performed between naã¯ve, F-MRT and F-MT,Flink relaxed FT andFlink MRShare techniques (Guoping, 2014). Analyzed and contrasted muti-demand execution plans are as follows:

1 FT plan independently running queries
2 F-MT project based on similar opportunities for data sharing
3 F-MRT project based on specific opportunities for data sharing with overlap consideration.

In the end, two parameters are calculated; the application execution period and thus the filtered data reduction. Typically, database execution time will be a common metric for information management systems wherever the request execution period was the duration from sending queries to completing queries. Whereas the information filtered displays the tuples which could be filtered to respond to a request running on data files such as input data or the result folder reuse

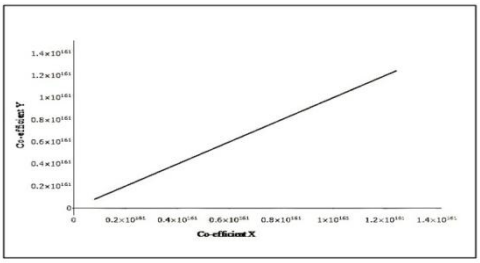

Fig 6: Data Visualization in vector format in Flink

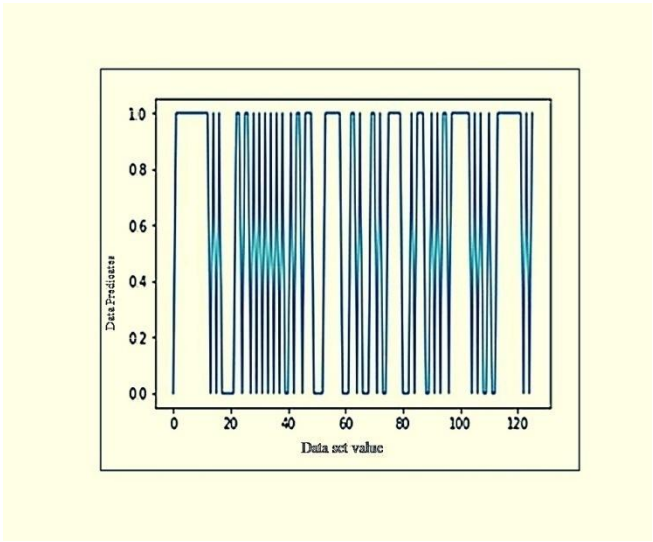

Fig 7: predicted data in Binary representation.

\

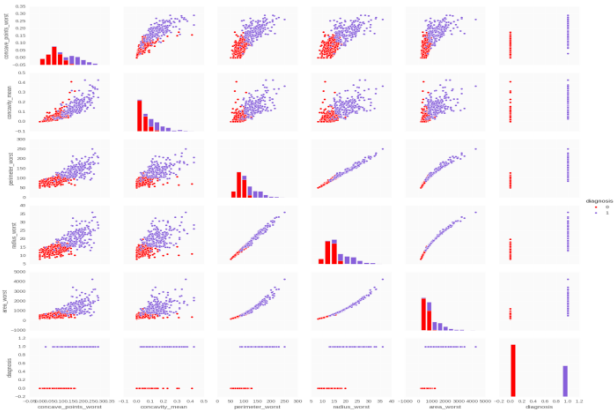

Fig 8: A matrix visualization.

We can easily distinguish the difference between **Malignant** and **Begnin** of given Breast cancer Data set.

5.1 Performance measurement

Table1: Performance variation with respect the network bandwidth

BandWidth(in Mbps)	Runtime(In Seconds)
110	2s
80 to 90	3s
50	7s
20 - 30	9s
5 - 10	12s or 13s

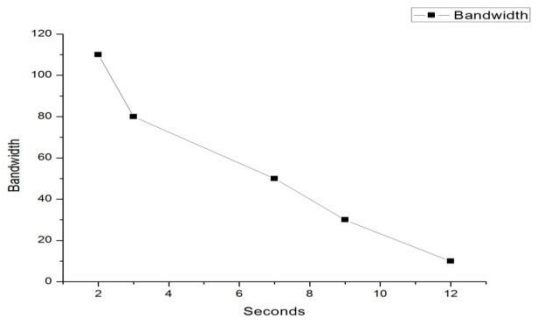

Fig 9: Performance flow with respect to time and Network bandwidth

III. CONCLUSION

The Flink system is built in this article on top of Processing of flink batches to maximize multi-query sharing over massive data. The proposed system of Flink uses two proven techniques for sharing; MRShare and relaxed MRShare to reveal the benefit of using mutual option predicates on multi-query. It can allow critical improvements in multi-query efficiency by increasing duplication overheads due to the collection of useless information which cause extra unnecessary work. The experimental analysis showed that, relative to the Naive methodThe Flink platform can simplify and speed up the exchange of information across a wider range of queries.
The data volume and sections of multi-query overlap will greatly improve the efficiency of the Flink process by eliminating redundant filtration tasks.

REFERENCES

1. Akerkar, R. (2013) 'Big data computing',in Business & Economics, 564pp, December, CRC Press.
2. Alhajj, R. and Polat, F. (1999) 'Using object-oriented materialized views to answer selection-based complex queries', *Information Sciences*, Vol. 118, No. 1, pp.75–99.
3. Apache Flink (2016a) *Scalable Batch and Streaming Data Processing* [online] https://flink.apache.org/ (accessed 18 November).
4. Apache Flink (2016b) *Table API – Relational Queries Beta* [online] https://ci.apache.org/projects/ flink/flink-docs-release-0.9/libs/table.html (accessed 8 August).
5. Apache Flink (2016c) *Table API and SQL Beta* (2016c) [online] https://ci.apache.org/projects/flink/ flink-docs-release-1.1/apis/table.html (accessed 13 November).
6. Babu, S. and Herodotou, H. (2013) 'Massively parallel databases and MapReduce systems', *Foundations and Trends in Databases*, Vol. 5, pp.1–104.
7. [online]https://www.sciencedirect.com/science/article/pii/S0020025514000346.
8. Council, T.P.P. (2008) *TPC-H Benchmark Specification* [online] http://www.tcp.org/hspec.htm (accessed 26 December 2016).
9. Dokeroglu, T., Ozal, S., Bayir, M.A., Cinar, M.S. and Cosar, A. (2014) 'Improving the performance of Hadoop Hive by sharing scan and computation tasks', *Journal of Cloud Computing*, Vol. 3, No. 1, pp.1–11.
10. Dong, Y., He, J., Yao, S. and Zhou, W. (2015) 'The skip-octree: a dynamic cloud storage index framework for multidimensional big data systems', *International Journal of Web Engineering and Technology*, Vol. 10, No. 4, pp.393–407.
11. Eiras-Franco, C., Bolón-Canedo, V., Ramos, S., González-Domínguez, J., Alonso-Betanzos, A. and Touriño, J. (2016) 'Multithreaded and Spark parallelization of feature selection filters',
12. *Journal of Computational Science*, Part 3, Vol. 17, pp.609–619 [online] https://www.sciencedirect.com/science/article/pii/S1877750316301107
13. Gkoulalas-Divanis, A. and Labbi, A. (2014) Large-Scale Data Analytics, National University of Singapore, Springer.
14. Guoping, W. (2014) Optimization Techniques for Complex Multi-query Applications, National University of Singapore.
15. Lee, K-H., Lee, Y-J., Choi, H., Chung, Y.D. and Moon, B. (2012) 'Parallel data processing with MapReduce: a survey', ACM SIGMoD Record, Vol. 40, No. 4, pp.11–20.
16. Lefevre, J., Sankaranarayanan, J., Hacigumus, H., Tatemura, J., Polyzotis, N. and Carey, M.J. (2014a)
17. 'Opportunistic physical design for big data analytics', Proceedings of the 2014 ACM
18. D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, et al. The design of the Borealis stream processing engine. CIDR, 2005.
19. T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernandez-Moctezuma, R. Lax, S. McVeety, D. Mills, ´ F. Perry, E. Schmidt, et al. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. PVLDB, 2015.
20. A. Nadkarni, D. Vesset, Worldwide Big Data Technology and Services Forecast,2016–2020, International Data Corporation, IDC, 2016.
21. Dynamic allocation in spark, http://spark.apache.org/docs/latest/job-scheduling.html/.
22. Álvaro Brandón Hernández a, María S. Perez a, Smrati Gupta b, Victor Muntés-Mulero b" Using machine learning to optimize parallelism in big data Applications" Future Generation Computer Systems 86 (2018) 1076–1092
23. M.A. Alsalem a , A .A . Zaidan a , B.B. Zaidan a , M. Hashim a , H.T. Madhloom a , N.D. Azeez a , S. Alsyisuf " A review of the automated detection and classification of acute leukaemia: Coherent taxonomy, datasets, validation and performance measurements, motivation, open challenges and recommendations" Computer Methods and Programs in Biomedicine 158 (2018) 93–112.
24. Mohamed Hosni a , ∗, Ibtissam Abnane a , Ali Idri a , Juan M. Carrillo de Gea b , JoséLuis Fernández Alemán" Reviewing ensemble classification methods in breast cancer" *Computer Methods and Programs in Biomedicine 177 (2019) 89–112*
25. Panagiota Galetsia, Korina Katsaliakia, Sameer Kumarb" Big data analytics in health sector: Theoretical framework, techniques and Prospects" *International Journal of Information Management 50 (2020) 206–216.*
26. Carl Witt, Marc Bux, Wladislaw Gusew, Ulf Leser" Predictive performance modeling for distributed batch processing using black box monitoring and machine learning" Information Systems 82 (2019) 33–52.
27. PekkaPääkkönen,1, DanielPakkala1" Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems" Big Data Research 2 (2015) 166–186.
28. Tanvir Habib Sardar, Zahid Ansari "An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm "Future Computing and Informatics Journal 3 (2018) 200e209.
29. Stefano Tribertia,b, Lucrezia Savionia,b, Valeria Sebria,b, Gabriella Pravettoni "eHealth for improving quality of life in breast cancer patients: a systematic review "Accepted Date: 7 January 2019.
30. *Kee Yuan Ngiam, Ing Wei Khor "*Big data and machine learning algorithms for health-care delivery"www.thelancet.com/oncology Vol 20 May 2019.
31. GASPARD HARERIMANA, (Student Member, IEEE), BEAKCHEOL JANG , (Member, IEEE),JONG WOOK KIM , (Member, IEEE), AND HUNG KOOK PARK" Health Big Data Analytics: A Technology Survey" Digital Object Identifier 10.1109/ACCESS.2018.2878254.
32. Hanjo Jeong 1 and Kyung Jin CHA" An E_cient MapReduce-Based Parallel Processing Framework for User-Based Collaborative Filtering "Symmetry 2019, 11, 748; doi:10.3390/sym11060748.
33. T. Y. J. Naga Malleswari1 and G. Vadivu" MapReduce: A Technical Review" Indian Journal of Science and Technology, Vol 9(1), DOI:10.17485/ijst/2016/v9i1/78964, January 2016.

AUTHORS PROFILE

Mr.Vikas S. received M.Phil degree in Computer Science in the year 2009 and Master of computer Applicaions (MCA) in the year 2007 from Visvesvaraya Technological University and Bachelors Degree in Computer Science in the year 2004 from kuvempu University. He is currently working as Assistant Professor in the **Department of CS&E, Visvesvaraya Technological University,** PG Center, Mysore, Karnataka, where he is involved in research and teaching activities. He is having 11 years of teaching experience and 02 years of Industrial experience. He is a Life member of India Society for Technical Education (LMISTE), Computer Society of India (CSI) and Doing Research work on the Area **Big data Analytics**.

Dr. Thimmaraju S N, He is presently a professor**, CS&E, Visvesvaraya Technological University,** PG Center, Mysore, Karnataka, he has received his Ph.D degree from **VisvesvarayaTechnological University(VTU)**, Belgaum in the year 2010, M.E., degree in Computer Science and Engineering from University Visvesvaraya College of Engineering (UVCE), Bangalore in 2002 and Bachelors Degree in Computer Science and Engineering from PESCE, Mandya in the year 1999. He is involved in research and teaching activities. His major areas of research are Computer Networks, WSN's and Graph theory. He is having 17 years of teaching experience. He has published around 17 research papers which include International Journals, International Conferences and Notional Conferences.