Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 4 (2016) pp. 220-228
© MIR Labs, www.mirlabs.net/jnic/index.html

Data Triggered Programming Model for Text Processing in Big Data

Sandhya N[1], Philip Samuel[2] and Mariamma Chacko[3]

[1]Information Technology,
Cochin University of Science and Technology,
Kochi - 682022,
nairsands@gmail.com

[2]Information Technology,
Cochin University of Science and Technology,
Kochi - 682022,
philips@cusat.ac.in

[3]Department of Ship Technology,
Cochin University of Science and Technology,
Kochi - 682022,
mariamma@cusat.ac.in

Abstract - Large volume of text processing becomes a challenge in recent era. Text processing methods drive much of modern data analysis across engineering sciences and commercial applications. Extraction of useful information from text sources refers to text analytics. This term describes tasks from annotating text sources with meta-information such as places mentioned in the text and a wide range of documents. The key/value pair generation of MapReduce program creates memory overhead and deserialization overhead due to data redundancy. Redundancy of data is one of the most important factors that consumes space and affect system performance while using large set of data. This overhead can be avoided considerably by using a novel approach that we developed named Data Triggered Multithreaded Programming (DTMP) model. In this paper, we demonstrate the use of DTMP model using a large dataset with author details and his publications. The Data Triggered Multithreaded Programming can dynamically allocate the resources and can identify the data repetition occurring during computation. DTMP model when applied to the MapReduce programming model brings performance improvement to the system. The major contributions of this work are a simple and scalable processing of text data that enables automatic parallelization and distribution of large-scale computations.

Keywords: Big Data Computing, Data Centric Architectures, Data Parallelism, MapReduce, Scalable, Data Triggered Multithreading

I. INTRODUCTION

With the development of technology, computing devices, social network sites, ecommerce services, sensors and mobile data has led us to an era of data explosion. Big Data refers to large and complex, structured and unstructured data in the order of exabytes and pentabytes [1]. The main four characteristics of big data are the four V's i.e, volume, velocity, veracity and variety. The term volume refers to the large quantity of data, the term velocity refers to the travelling speed of data and the processing speed, the term variety refers to the diversity of data types and veracity is the ability to trust accurate data and the reliability of data [1].

In traditional high-performance computer applications (e.g., for weather forecasting), it is common place for a high-performance computer to have processing nodes and storage nodes connected together by a high-capacity interconnect. MapReduce programming model is an architecture where processors and storage devices are co-located. In such cases, we can take advantage of data locality by running code on the processor directly attached to the block of data we need [3].

In current computer architectures, it is harder to overcome the data-intensive computation such as the latency gap between multi-core CPUs and mechanical hard disks, which is growing every year [31]. Hence, it is necessary to tackle these problems in a scalable manner with a scalable architecture model. Gray argued that the new trend should focus on supporting cheaper clusters of computers to manage and process all this data instead of focusing on having the biggest and fastest single computer.

In the distributed file system, frameworks have been developed for large volume of data processing. Text processing is one of the big data processing challenges. Basic element in text processing is the document processing. Document can be defined as a unit of discrete textual data in which usually within a collection of text data will correlates with some real world text document such as e-mails, research papers, blog reports, manuscripts and press release. A document can also be a member of different document collections, or different subsets of the same document collection [2].

A text document may be seen from different perspectives, as a structured or semi-structured format of data. Document itself explains a large amount of semantic and syntactical structure, this structure is implicit and to some extend hidden in its textual context [2]. We can consider the elements such as punctuation marks, capitalization, numeric, and special characters such as white space, carriage returns, underlining, asterisks, tables, columns and so on. Text documents are strong typographical layout or mark-up indicators to denote structure [2]. Text documents with extensive and consistent format elements in which field-type metadata can be more easily inferred such as e-mail, HTML web pages, PDF files and word processing files with document templating or style-sheet constraints are described as semi-structured text documents. One of the most important programming models is MapReduce [4], [5]. It is an emerging programming model for data-intensive applications. The idea behind MapReduce is from functional programming, where the programmer defines Map and Reduce functions to process large sets of distributed data. It was originally developed by Google and built on well-known principles in parallel and distributed processing several years back. MapReduce has widespread adoption via an open-source implementation called Hadoop, whose development was led by Yahoo. Today, a vibrant software ecosystem has sprung up around Hadoop, with significant activity in both industry and academia.

The Workflows [9] which are widely used in data-intensive text processing applications since it facilities the composition of individually developed executable or scripts types, making it easier for the experts to focus on their research rather than the management of analysis and computation. Many systems are proposed to execute workflows, including GXP Make [11], Swift. A workflow is generally a DAG with many data processing tasks and their dependencies. Each task is a typical existing binary or executable [40]. For example, workflows in natural language processing (NLP) typically consist of sentence splitters, part-of-speech taggers, named entity recognizers, parsers, data indexers, and so on. Many of parsers are third-party components that received a considerable amount of development efforts in the community [40].

The text mining algorithms operate on the feature based notations of the text document. The first goal of this is to achieve correct calculation of the volume and the semantic level of features to picture the text accurately [2]. This tends to favourably disposed towards text mining pre-processing operations towards selecting or extracting relatively more features to represent text document [4]. The next goal is to find out the computationally efficient features for pattern recognition, which supports validation, normalization, or cross-referencing of features against vocabularies or external sources such as dictionary or thesauri which are semantically rich feature. Different features can be taken to represent text document, the following four features are most commonly used:

- Characters: The character level representation can include the full set of all characters for a text document or some subsets. For text mining applications, character-based representation without positional information are often utilized in a limited manner. This type of representation includes some level of positional information. This feature type can be viewed as the most complete of any representation of a real-world text document [2].
- Words: Word level features are sometimes referred to as existing in the native feature space of the text processing. In the case of linguistic token, a single word level feature should equate with that. Phrases, multiword expressions or multiword hyphenates would not constitute single word level features [2].

The Data Triggered Multithreaded Programming (DTMP) Model inherits the power of eliminating redundant computation from data triggered thread but enhances the design of the programming model and runtime system to demonstrate the ability to support massive data-level parallelism [3] , [4]. The DTMP model provides a new type of data trigger declaration that allows programmers to trigger computation more efficiently. The DTMP model also allows programmers to describe the ordering of triggered computation. The DTMP model supports many threads running at the same time and executes threads in and out-of-order fashion [5].

In traditional distributed high performance computing systems (e.g., Gnome purpose), it is common for a master computer to have nodes for processing and nodes for storage, this is connected through large bandwidth connectivity network. Since workloads are not processor demanding, this storage and data interaction creates large problems. That is where this hadoop type architecture comes in to existence, where processors and storage are co-located. In such a setup, we can take advantage of data locality by running code on the processor directly attached to the block of data we need. The distributed file system is responsible for managing the data over which MapReduce operates.

In this paper, we propose a Data Triggered Multithreaded Programming together with MapReduce programming model [7]. This DTMP model is a data centric programming model obtained from the data-triggered threads (DTT) model to better understand the need for data-centric computing. DTMP model initiates parallel computation when the application changes memory content.

This paper is organized as follows. Section 2 discusses related work. In section 3 we provide the details of Multithreaded MapReduce Model. Section 4 is Case study of the Threaded MapReduce Model using an Author dataset. Section V is the Conclusion of the paper.

II. BIG DATA TEXT PROCESSING CHALLENGES

To handle Big Data in a scalable manner for an intelligent learning database system [32], the essential method is to scale up large volume of data and provide treatments for the characteristics featured by the big data. The main characteristics are volume, velocity, variety and veracity. These characteristics can be considered in different forms to handle different issues. Data volume issues become important in the case of data accessing and computing. Veracity can be issues in data privacy and domain knowledge can be considered. Velocity and variety can be handled by different data mining algorithms. The challenges at volume of data focus on data accessing and actual computing procedures. Because big data are often stored at different locations and data volumes may grow continuously. A scalable computing platform will have to take distributed large-scale scalable data storage into consideration for computing [32].

The challenges of the next characteristics called variety centred on semantics and domain knowledge for different Big Data applications (Eg. Facebook). This information provides additional benefits to the mining process, and also in addition technical barriers to the Big Data access such as volume of data and mining algorithms [31]. The data privacy and information sharing mechanisms between data producers and data consumers can be significantly different depending on different domain applications. Sharing online streaming data for applications like sensor data or gnome data may not be discouraged. In addition to the above privacy issues, the application domains can also provide additional information to benefit or guide Big Data mining algorithm designs.

Like association rule in market basket analysis of data, each transaction is considered independent and the discovered knowledge is typically represented by finding highly correlated items, possibly with respect to different dimensional aspects. In a social network, on the other hand, users are linked and share dependency structures [10]. The knowledge is then represented by user communities, leaders in each group, and social influence modelling etc. For low-level and high level data access understanding of semantics and application knowledge is important in data processing algorithms. The data mining challenges concentrate on algorithm designs in tracking the difficulties raised by the Big Data volumes, distributed data distributions, and by other complex data processing characteristics.

The third part contains three stages such as sparse, diverse, uncertain, complex, and multi-source data are pre-processed by data fusion techniques. Next is that, complex and dynamic data are mined after pre-processing. Finally, the global knowledge obtained is tested and relevant information is fed back to the pre-processing stage.

III. Data Triggered Architecture

The proposed model implements both Data Triggered Multithreaded Programming model followed by the MapReduce model which can improve the system performance in a cost effective manner [24].

As shown in Fig.1. We have data trigger and a trigger point for the DTMP model and this process outputs the tasks to a queue. We can modify the memory content of the data through data trigger process. Trigger point that waits for the completion of all outstanding events of a certain support thread functions.

The queued tasks are the input to the MapReduce model. In the Map Reduce Model the data splits into different groups and these are stored in the distributed file system. In this model the main two phases are Map and Reduce.

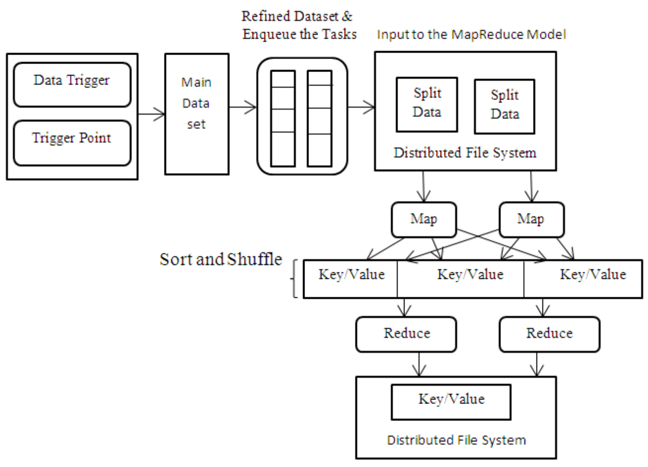

Figure1. Architecture of Multithreaded MapReduce Model

Data from the distributed file system are taken and fed to the Map phase. Here dataset is represented in the form of key/value pairs. These key/value pairs are sorted and shuffled. Thus generate an intermediate key/value pair. This intermediate key/value pairs are input to the Reduce function. The output of the Reduce function is the final result which is stored in the distributed file system.

A. MapReduce Programming Model

MapReduce programming has made large complex text data processing easier to understand, efficient and tolerant of hardware failures during computation [19]. This technology is applied for batch processing of large volumes of data, and it is not suitable for recent demands like real-time processing [8]. MapReduce [10] is a parallel data processing approach for execution on computer cluster [3]. For text file processing MapReduce is one of the important programming model. The key strengths of the MapReduce programming framework are parallelism combined with its simple programming framework and it can be applicable to a large variety of application domains. This requires dividing the workload across more number of machines. The degree of parallelism depends on the input data size [3].

MapReduce programming model defines *Map* function and *Reduce* function. The *Map* function performs sorting and filtering of the input data. The input data to *Map* function is in the form of key/value pair [10]. The *Map* function in the MapReduce programming is fed with data stored on the distributed file system are split across the nodes in the cluster. The Map tasks are started on the compute nodes and Map function is applicable to each Key/Value pair and the output of the Map function is intermediate Key/Value pairs. This intermediate Key/Value pairs are stored in the local file system and sorted by the keys [3].

The Reduce() function performs the addition operation. The Reduce function takes the intermediate Key/Value pair as the input and a list of intermediate values with that key as its input. All key-value pairs for the same key are fed to the same function [2]. A *Reduce* function is applied to all values grouped for one key and in turn generates Key/Value pairs. Key/value pairs from each reducer are written on the distributed file system. The final output will be set of values.

B. Data Triggered Multithreaded Programming

The DTMP programming eliminates redundant computation but improves the design of the programming model and runtime system to describe the capability to support massive data-level parallelism [10]. The DTMP programming provides a triggering of data for declaration that allows programmers to more efficiently trigger computation [12].

The DTMP programming also allows programmers to describe the ordering of triggered computation [24]. The DTMP programming supports multiple threads running at the same time and executes threads in and out of order manner. Based on the changes happen in the memory address locations that triggers the data computation, the runtime system of DTMP can dynamically schedule computation to the most appropriate computing resource to reduce the amount of cache misses and data synchronization traffic. The runtime system can also balance the workload among processing units [12]. The user of the DTMP model achieves parallelism by declaring data triggers and an associated thread function [24].

The Data Triggered Multithreaded Programming model, the user can declare a variable, a data structure and a triggering thread [12]. The triggering thread can be assigned to the variable. The fields in the data structure contain the attributes of data for processing. The supporting thread function describes the computation, when the program changes the value of a data trigger [13]. In our DTMP model we are using data trigger and trigger point. The data trigger operation can modify the data trigger's memory content, and trigger point waits for the completion of all occurrences of all the supporting thread functions [13].
When this application executes data trigger operation, then these following steps are executed [131].

- The DTMP system checks for any changes in the current version. If the system detects any changes, then it will create new thread function event containing the changed address and the supporting thread function connected with the triggering data. DTMP system will increase the trigger point to the next value associated with the thread function. If the system does not detect any changes then it will not trigger any data.
- The changed address is analysed and then enqueue the appropriate task.
- When supporting thread function finishes its execution and the polling thread notifies the system and release the completed tasks in the queue. Then the trigger point will decreases its counter value associated with the supporting thread function.
- When the program reaches the trigger point of the supporting thread function, then it stops its operation and checks for any incomplete tasks associated with the trigger point. Thus the program resumes, if pending task does not exist with the trigger point.

Data repetition creates large amount of unwanted threads results in performance degradation [12]. The data triggered is defined with continuously changing element. In the DTMP model the destination value of the assignment changes and the data thread triggers when a same data fields appears in the data set. If all the attributes or fields of one data match then the triggering of data occurred. This application can be used to trace any particular event or element of large dataset to reduce redundancy.

IV. Implementation on Authors Dataset Model

The proposed model uses a large dataset, which consists of author and his publication details, i.e. authorId, author name, publications details. So for the processing of dataset we are taking authorid and his number of publications in this example.

We process this author dataset using data triggered multithreading first and then this dataset will be input to the MapReduce programming model. The data triggered programming requires declaration of the data triggers to achieve parallelism and associated with each trigger needs a supporting thread function. The following steps are required for the first processing of the data triggered programming. For each author, a data structure field is created.

i. We can declare a variable to the author name.
ii. Value is assigned for each author id as the data trigger which is associated with author name.

As shown in Table 1. Data structure is defined with name 'author' and contains fields such as authorid, papercount, and citations. These are the details of the dataset which we are processing. Next we defined the data triggered thread function which contains thread pointer and filetype pointers for the fields in the data structure [24].

Table 1.Data Triggered Model for Author Dataset

Author Declaration

Typedef struct author {
    author_id;
    papercount;
    citations; …….   }
author;

Data triggered function

int author_thread(void *aid_ptr) {
int i, j, k;
fptype authorid;
fptype publishno;
fptype citation;
return 0; # trigger_point data author_thread Inner loop

This triggering model allows the user to declare a data trigger after an assignment statement. The data triggers multithreaded execution when the destination value of the assignment changes. i.e. when an author_id with same author name and same publications arrives this triggers and redundancy recognized and repeated 'authorid' are not entered to the refined dataset. If both the fields are same then only it is considered as redundant data. Redundant threads can be eliminated to an extent.

As shown in Figure 2. completing redundancy check by the DTMP model, this is entered to a dataset. From main dataset which is of refined form with no redundant data, here we enqueue the tasks and the input is fed to the MapReduce programming model. The author name, publications and citations are available in the dataset. In the Map phase the sorting and shuffling takes place and author names are taken as keys and publication numbers are taken as values.

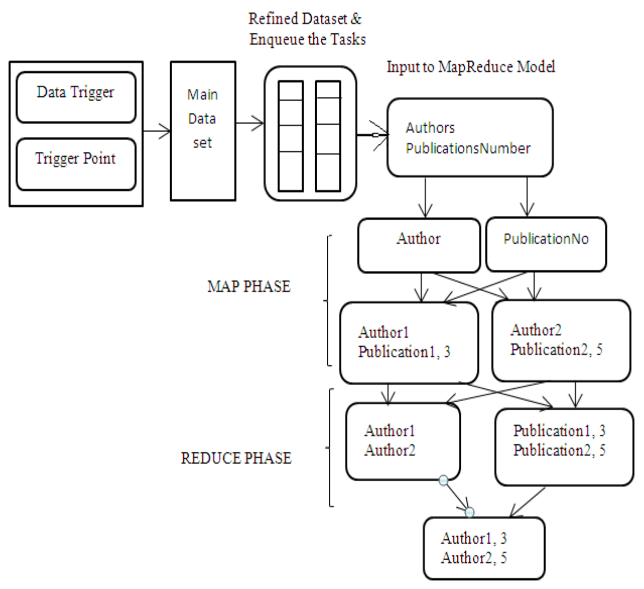

Figure 2. Multithreaded MapReduce Model Design Using Author Dataset

When a MapReduce task is submitted to the system then the map task applies the Map function to every Key/Value pair (author1, publicationno). Zero or more intermediate Key/Value pairs (list(authors, publications)) are generated for the same input Key/Value pair. These are stored in the distributed file system and sorted by the keys.

Table 2. Algorithm Map and Reduce Functions

```
Algorithm 1:  Map Function

Input : String Authorid, publicationno
Key : Messageid, value: document value
String[] twittermessage = value.split("|")
Intermediatekey(article[0], ParseFloat(twittermessage [2]))

Algorithm 2: Reduce Function

Input : String key, Iterator values
Float totalmessage = 0;
While values.hasNext() do
    Totalmessage + = values.next();
End
```

Completion of the map tasks notifies the reducer by the MapReduce engine. In the reduce phase, the output files from the map tasks are taken in parallel and sort the files to combine the Key/Value pairs into a set of new Key/Value pairs i.e, (authors,list(publications)), where all publications with the same author are listed. The reduce function is applied and the final list total number of authors and publications are generated. Based on this author and publication list journal ratings are calculated.

If the above dataset is processing only through MapReduce programming model then redundant data will be more. Since we are taking this dataset of author details from different sites, blogs or databases, repetition of data will be there. Data triggered threads usage avoids this to a minimum. In this data triggered model we are assigning values to some variable. If repetition occurs then the variable changes its value and data triggering occurs.

V. Related Works

As the data grows, the availability of high performance and relatively low-cost hardware database systems are parallelized to run on multiple hardware platforms to manage scalability [8]. MapReduce model, whose basic idea is to simplify the distributed computing platform that offers two main functions Map and Reduce. The MapReduce programming model can be used to solve parallel problems [8]. This model can be used in applications such as data mining, machine learning and scientific computations. Hadoop, a popular big data processing framework implements this MapReduce programming model [6].

One of the important characteristics of Big Data is computing tasks on the large dataset, because of the different sources, large volume, heterogeneous and dynamic characteristics of application data involved in a distributed environment. The terabyte and petabyte level of data with a complex computing process is very difficult. Hence, utilizing a parallel computer infrastructure, its corresponding programming language support, and software models to efficiently analyze and mine the distributed dataset, The data's are the critical goal for Big Data processing to change from "quantity" to "quality". Currently, Big Data processing mainly depends on parallel programming models like MapReduce, as well as providing a cloud computing platform of Big Data services for the public. MapReduce is a batch oriented parallel computing model. There is still a certain gap in performance with relational databases. How to improve the performance of MapReduce and enhance the real-time nature of large-scale data processing is a hot topic in research.

The MapReduce parallel programming model has been applied in many machine learning and data mining algorithms in different applications. To solve and optimise model parameters in machine learning applications data mining algorithms usually need to go through different training datasets. It calls for intensive computing to access the large-scale data frequently. In order to improve the efficiency of algorithms, [32] proposed a general-purpose parallel programming method which is applicable to a huge number of algorithms which support machine learning based on the simple MapReduce programming model on multi-core processors. Many classic data mining algorithms are included such as locally weighted linear regression, k-Means, logistic regression, support vector machines, the independent variable analysis, Gaussian discriminant analysis, expectation maximization and back propagation neural networks [32]. With the analysis of these classical machine learning algorithms, we argue that the computational operations in the algorithm learning process could be transformed into a summation operation on a number of training data sets.

Then various addition operations could be performed on these Mapper nodes to get intermediate results. Finally, text processing algorithms are parallel executed through collecting all Reducer nodes. Ranger [33] proposed a MapReduce based application programming interface Phoenix, which supports parallel programming in the environment of multi-core and multi-processor systems, and realized three data mining algorithms including k-Means, principal component analysis, and linear regression.

The MapReduce's implementation mechanism in Hadoop has been improved by Gillick and evaluated the algorithm's performance of single-pass learning, iterative learning and query-based learning in the MapReduce framework [34]. He also studied how to share data between computing nodes involved in parallel learning algorithms and how to work with distributed storage data, and then showed that the MapReduce mechanisms suitable for large-scale data mining by testing series of standard data mining tasks on medium-size clusters [34].

The number of input data text files greatly affects MapReduce processing time, especially in the big data processing. In order to decrease the MapReduce processing time, we need to pre-process the input data for the mapper. In the context of batch processing data, large semi-structured and unstructured data of different documents, the data itself is a raw data obtain directly from text document.

The common types of errors in text data processing is redundancy readings (duplicate reads). This redundancy problem is recognized as a serious issue in batch processing. Redundancy can happen at two different levels, reader level or data level.

- Reader level redundancy occurred when there are more than one reader or different factors that affecting text data.
- Data level redundancy occurred as unwanted and unreliable data. It happens when text documents or text data taken from social networking sites or blogs.

In protection of massive text data, proposed a multi-layer rough set model, which can accurately describe the granularity change produced by different levels of generalization and provide a theoretical foundation for measuring the data effectiveness criteria in the unremarkable process, and designed a dynamic mechanism for balancing privacy and data utility, to solve the optimization of refinement order for classification algorithms [35].

A recent paper on protection of confidentiality in big data summarizes a number of methods for protecting public release data, including aggregation, suppression, data swapping (i.e., switching values of sensitive data records to prevent users from matching), adding random noise, or simply replacing the whole original data values at a high risk of disclosure with values synthetically generated from simulated distributions [36].

For applications involving Big Data and tremendous data volumes, it is often the case that data are physically distributed at different locations, which means that users no longer physically possess the storage of their data.

In the processing of text or document mining, having an efficient and effective data access mechanism is vital, especially for users who intend to hire a third party to process their data. Under such a circumstance, users' privacy concerns may include:

- No local data copies or downloading of data
- Different data analysis must be deployed based on the existing data storage systems without violating existing privacy settings, and many others.

Wang proposed a privacy preserving mechanism for large scale data storage such as cloud computing systems has been proposed [36]. The public key based mechanism is used to enable Third Party Auditing (TPA), so users can safely allow a third party to analyse their data without breaching the security settings or compromising the data privacy.

For most Big Data applications, privacy concerns focus on excluding the data miners from directly accessing the original data. Common solutions are to rely on some privacy preserving approaches or encryption mechanisms to protect the data.

A recent effort by [37] indicates that users' "data access patterns" can also have severe data privacy concerns and lead to disclosures of geographically co-located users or users with common interests. In their system, it hides data access patterns from the servers by using virtual disks.

As a result, it can support a variety of 20 Big Data applications, such as blog search and social network queries, without compromising the user privacy. Big Data Mining Algorithms, in order to adapt to the multi-source, massive, dynamic Big Data, researchers have expanded existing data mining methods in many ways, including the efficiency improvement of single-source knowledge discovery methods [38], designing a data mining mechanism from a multi-source perspective [39] as well as the study of dynamic data mining methods and the analysis of convection data.

The main motivation for discovering knowledge from massive data is improving the efficiency of single-source mining methods. On the basis of gradual improvement of computer hardware functions, researchers continue to explore ways to improve the efficiency of knowledge discovery algorithms to make them better for massive data.

Text analytics spans across virtually all verticals. We frequently come across text analytics use cases in finance, insurance, media, and retail industries, but even oil and gas companies can derive value from text analytics. A typical text-analytics application in the finance industry focuses on compliance and fraud prevention.

Different data retrieval text processing for full-text search today rely on a data structure called an inverted index, it will give a term which provides access to the list of documents that contain the term. In information retrieval parlance, objects to be retrieved are generically called documents, even though in actuality they may be web pages, PDFs, texts in different social networking sites [2].

When user queries are thrown then the text retrieval process uses the inverted index to achieve documents that contain the query terms with respect to some ranking model, taking into account features such as term matches, term proximity, attributes of the terms in the document, as well as the hyperlink structure of the documents [2].

The web search problem which we are handling in our day today life are decomposes into three components namely gathering web content also called crawling construction of the inverted index and ranking documents, which means if given a query retrieval.

Crawling and indexing share almost same characteristics and requirements, but these are very different from retrieval. Gathering web content and building inverted indexes are for the most part offline problems. Both need to be scalable and efficient, but they do not need to operate in real time. Indexing is usually a batch process that runs periodically: the frequency of refreshes and updates is usually dependent on the 65 design of the crawler.

Dataflow architectures are similar to data triggered programming which tries to achieve parallelism by triggering computation. The data flow architectures require hardware support which is more costly and complex [10].

The data triggered model does not require any hardware support and provides composing applications. The data triggered programming shares the same functionalities as Cilk [16] and CEAL [17] that extends dataflow like programming and execution models on conventional architectures.

Cilk exploits dataflow parallelism like functional programming language. CEAL encourages programmers to use incremental algorithms on changing data to avoid redundant computation. The DTMP model extends Data-Triggered Threads, which is designed to exploit both dataflow-like parallelism and reduce redundant computation [8].

The programming model of DTMP is similar to Habanero [6] in triggering multithreaded computation asynchronously, in allocating tasks and load balancing. The DTMP model, the program can avoid redundant computation that Habanero does not address.

VI. Performance Evaluation

Here as shown in Fig.3. performance test evaluation is done using an author dataset. Our performance measurements are made on Ubuntu personal computer with an Intel Core 2 Quad Q6600 (3 CPUs) 2.4 GHz processor, 4 GB of main memory, Java Runtime Environment 1.7 and Eclipse IDE as development tool.

Test dataset consists of author_id, author name, publication details and citation details with the size of 5GB, 3GB and 2 GB data. The first testing attempt is done by removing redundant data in a 1GB data, number of threads generated is 6 and without removing redundant data number of threads generated is 10.

Second attempt is using a 3 GB data by removing redundant data number of threads generated is 9 and original data for processing number of threads generated is 16.

Third attempt is using 5GB data, by removing redundant data, number of threads generated is 11 and by original data, number of threads generated is 20.

Here the graphs with data triggered multithreaded programming model is coloured red and bar graph without data triggered programming model is shown as blue. From the above performance evaluation graph we could measure the performance improvent of the system with data triggered multithreaded model(DTMP) over without data triggered multithreaded model.

Results show that the algorithm works well in minimizing the dataset by removing redundant data and improves the system parallelism, which improves system performance. Thus we can see that by minimizing the dataset i.e. optimizing the dataset the performance can be improved to an extent, thus processing time decreases. Hence the overall system performance increases .

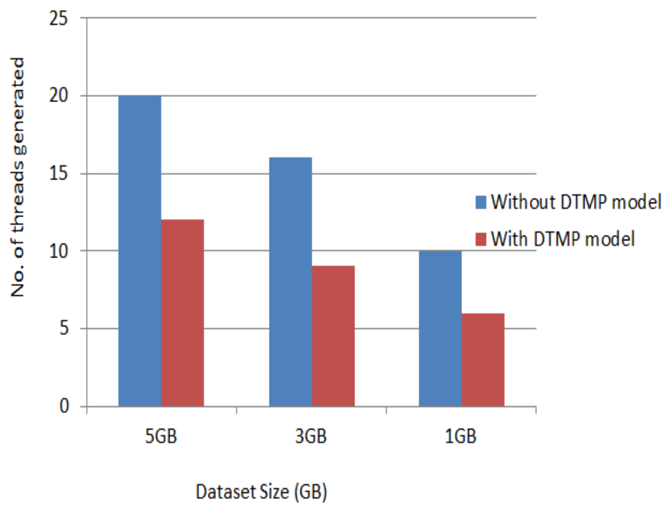

Figure 3. Performance evaluation

VII. Conclusion

Due to the technological growth and data explosion, handling data in scalable and efficient manner is necessary. Usage of large volume of data in different applications, redundancy is one of the important factors which consumes space and affect the system performance. By processing the large dataset using multithreaded data triggering method redundant data can be reduced.

Repetition of data during processing affects the system performance. MapReduce model has limited capability of controlling redundant data. Document mining or text mining dominated by a data-driven, empirical approach, typically involving algorithms that attempt to capture statistical regularities in data for the purposes of some task or application For batch processing types of data processing MapReduce model provides faster computation.

Due to large volume of data the key-value pair generation of MapReduce program creates memory overhead and deserialization overhead. This overhead can be avoided to the maximum by using Data Triggering Programming Model. Processing large volume of data using this DTMP model and then sending to MapReduce Model helps to improve the performance of the system.

Thus the data triggered programming together with MapReduce programming brings performance improvement. Thus our system has the power to reduce redundancy and improve performance.

This proposed system can also achieve better scalability than normal MapReduce programming model. As shown in Fig.3. performance of the system has been improved by using DTMP model when compared with normal MapReduce programming. This type of model can used for social networking sites such as twitter and facebook.

As we use data triggering multithreaded programming model, tracing particular member or particular issue becomes simpler. This proposed system strengthens the MapReduce technology with our DTMP model and serves as a basis for further experimentation.

REFERENCES

[1] Arvind and R.S. Nikhil. "Executing a program on the mit tagged-token dataflow architecture", *IEEE Transactions on Computers*, pp.300–318, 1990.

[2] Lin, Jimmy, and D. Chris. "*Data-intensive text processing with MapReduce*, Synthesis Lectures on Human Language Technologies, vol. 3, pp.177, 2010.

[3] F. Li , B.C. Ooi, Tamer, M. Ozsu, and S.Wu. "*Distributed data management using MapReduce, In ACM Computing Surveys (CSUR)*, vol. 46, Issue 3, 2014.

[4] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar. and R. Pasquin, "*Incoop, MapReduce for incremental computations, In ACM SOCC*", 2011.

[5] H.-W. Tseng, and D. M. Tullsen, Data-triggered threads: Eliminating redundant computation. "*In 17th International Symposium on High Performance Computer Architecture*", pp.181–192, 2011.

[6] V. Cave, J. Zhao, J. Shirako, and V. Sarkar. "*Habanero Java -The new adventures of old x10"*. In Proceedings of the 9th International Conference on Principles and Practice of Programming in Java, PPPJ '11, pp. 51-61,2011.

[7] J. Dean, and S. Ghemawat. "*Mapreduce: simplified data processing on large clusters"*. ACM Proceedings, Jan. pp.107–113, 2008.

[8] Arvind, and R. S. Nikhil. "*Executing a program on the mit tagged-token dataflow architecture"*. IEEE Transactions on Computers, pp. 300–318 1990.

[9] E. Deelman, D. Gannon, M. S. Shields, and I. Taylor, "Workflows and e-science. An overview of workflow system features and capabilities," Future Generation Comp. Syst., vol. 25, no. 5, pp. 528–540, 2009.

[10] J. Steffan, C. Colohan, A. Zhai , and T. Mowry. "*A scalable approach to thread-level speculation*". In 27th Annual International Symposium on Computer Architecture, pp. 1-12, 2000.

[11] K. Taura, T. Matsuzaki, M. Miwa. "*Design and implementation of gxp make – a workflow system based on make,*" in eScience2010, pp. 214–221.

[12] S. Hong, H. Kim. "An analytical Model for a GPU Architecture with Memory-Level and Thread-Level Parallelism Awareness", In: ACM SIGARCH Computer Architecture News, pp. 152–163, 2009.

[13] H.-W. Tseng, and D.M Tullsen. "Data-triggered Multithreading for Near-Data Processing". In 1st Workshop on Near-Data Processing (WoNDP), 2013.

[14] Miner, Donald, Adam. " MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems", 'O'Reilly Media Inc. 2012.

[15] H.C. Yang, A. Dasdan, R.L. Hsiao, D.S. Parker. "MapReduce-merge: simplified relational data processing on large clusters", In ACM SIGMOD '07, pp. 1029–1040, 2007.

[16] Z. Matei, K.i. Andy, D.J. Anthony, H. Randy, Katz, and S. Ion. "Improving MapReduce performance in heterogeneous environments". *In Proceedings of the 8th USENIX Symposium on Operating System Design and Implementation*, pp. 29–42, 2008.

[17] H.-W. Tseng, D.M. Tullsen. "Software data-triggered threads". In ACM SIGPLAN 2012 Conference on Object-Oriented Programming, Systems, Languages and Applications, 2012.

[18] Brunett, J. Thornley, M. Ellenbecker. "An initial evaluation of the tera multithreaded architecture and programming system using the the c3i parallel benchmark suite". In Proceedings of the 1998 ACM/IEEE conference on Supercomputing (SC 1998), pp.1–19, 1998.

[19] B. Lewis, D.J. Berg. "Multithreaded Programming with Pthreads". Prentice Hall, 1998.

[20] M. Frigo, C.E. Leiserson, K.H. Randall. "The implementation of the cilk-5 multithreaded language". *In ACM SIGPLAN 1998 conference on Programming language design and implementation*, pp.212–223, 1998.

[21] M. A. Hammer, U. A. Acar, Y. Chen. " CEAL: A C-based language for self-adjusting computation". In *ACM SIGPLAN 2009* conference on Programming language design and implementation, pp.25–37 2009.

[22] K. Shim. "MapReduce Algorithms for Big Data Analysis Databases in Networked Information Systems". *Springer Berlin Heidelberg*, pp.44-48 2013.

[23] Leung, Carson, Kai-Sang, H. Yaroslav. "Mining frequent patterns from uncertain data with MapReduce for Big Data analytics. Database Systems for Advanced Applications". *Springer* Berlin Heidelberg, 2013.

[24] N. Sandhya, P. Samuel. "Data Dentric Text Processing using MapReduce". *In Proceedings of the 6th International Conference on Innovations in Bio-Inspired Computing and Applications*, IBICA'15, pp.129-137, 2015.

[25] A. Machanavajjhala, P. Jerome, Reiter. "Big privacy: Protecting Confidentiality in Big data", *ACM Crossroads*, 19(1) pp.20-23, 2012.

[26] C. Ranger, R. Raghuraman, A. Penmetsa , G. Bradski, C. Kozyrakis. "Evaluating MapReduce for multi-core and multiprocessor systems", In: *Proceedings of the 13th IEEE International Symposium on High Performance Computer Architecture (HPCA '07*), pp. 13-24, 2007.

[27] D. Gillick , A. Faria, J. DeNero. "MapReduce: Distributed Computing for Machine Learning", Berkley, December 18, 2006.

[28] S. Das, Y. Sismanis, K.S. Beyer, R. Gemulla , P.J. Haas, J. McPherson, Ricardo. "Integrating R and Hadoop", In: *Proceedings of the 2010 ACM SIGMOD* International Conference on Management of data (SIGMOD '10), 2010, pp. 987-998, 2010.

[29] D. Wegener, M. Mock, D. Adranale, S. Wrobel. "Toolkit-Based high-performance data mining of large data on MapReduce clusters", In: *Proceedings of the ICDM Workshop*, pp. 296-301, 2009.

[30] A. Ghoting, E. Pednault. "Hadoop-ML: An infrastructure for the rapid implementation of parallel reusable analytics", In: *Proceedings of the Large-Scale Machine Learning: Parallelism and Massive Datasets Workshop* (NIPS-2009).

[31] G.Bell, J. Gray, and S. Alexander. "Petascale Computational Systems", *IEEE Computer*, 39(1): pp.110-112.

[32] Gandomi, Amir, M. Haider. " Beyond the hype: Big data concepts, methods and analytics", *International Journal of Information Management* , pp. 137-144, 2015.

[33] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, " Evaluating MapReduce for multi-core and multiprocessor systems", In the *Proceedings of the 13th IEEE International Symposium on High Performance Computer Architecture* (HPCA '07), 2007, pp. 13-24.

[34] N. Marz, J. Warren. "Big Data: Principles and best practices of scalable realtime data systems", Manning Publications Co, 2015 Mar 31.

[35] M. Ye, X. Wu, X. Hu, D. Hu. "Anonymising classification data using rough set theory, Knowledge-Based Systems", 43: pp. 82-94, 2013.

[36] Q. Wang, K. Ren, W. Lou. "Privacy-Preserving Public Auditing for Data Storage Security in Could Computing", *IEEE Transactions* on Computers, 62(2):pp. 362-375, 2013.

[37] J. Lorch, B. Parno, J. Mickens, M. Raykova, and J. Schiffman, Shoroud " Ensuring Private Access to Large-Scale Data in the Data Center", In: *Proc. of the 11th USENIX Conference on File and Storage Technologies* (FAST'13), San Jose, CA, 2013.

[38] E.Y . Chang, H. Bai, and K. Zhu. "Parallel algorithms for mining large scale rich-media data", In *the Proceedings of the 17th ACM International Conference on Multimedia* (MM'09), NY, USA, , pp. 917-918, 2009.

[39] X. Wu, and X. Zhang. "Synthesizing High-Frequency Rules from Different Data Sources", *IEEE Transactions on Knowledge and Data Engineering*, vol.15, no.2, pp.353-367.

[40] T. Cheng, and K. Taura. "A Comparative Study of Data Processing Approaches for Text Processing Workflows", In the *Proceedings of High Performance Computing Networking Storage and Analysis*, IEEE Computer Society Washington, DC, USA, pp. 1260-1267.

Author Biographies

Mrs. Sandhya N is Researcher in Information Technology Division, School of Engineering, Cochin University of Science & Technology (CUSAT). She holds a Master Degree (M.Tech) in Computational Engineering and Networking from Amrita Viswa Vidya Peetham (2009-2011). She has 5years of experience in Research & development in IT Company. She has published a paper on 2012 7th International Conference on Computer Science and Education (ICCSE). Her research interest includes Big Data Analytics, Testing and Analysis, Object Oriented Programming.

Dr. Philip Samuel is Reader in Information Technology Division, School of Engineering, Cochin University of Science & Technology (CUSAT). He holds M.Tech in Computer & Information Science from Cochin University of Science & Technology and Ph.D degree in Computer Science & Engineering from IIT Kharagpur. He has more than 17 years of experience in teaching and research as a faculty at Cochin University of Science & Technology. He has published more than 35 research papers in International Conferences and Journals. His research interest includes Big Data Analytics, Distributed Computing and Automated Software Engineering.

Mariamma Chacko was born in 1961 at Changanacherry, India. She received her Bachelor's degree in Electrical Engineering from University of Kerala in 1985, Master's degree in Electronics from Cochin University of Science and Technology in 1987 and PhD in Computer Engineering from Cochin University of Science and Technology in 2012. She has been working as Associate Professor in the Department of Ship Technology at Cochin University of Science and Technology since 2006. She has 12 research publications to her credit. Her research interests include validation and optimization of embedded software, and sensorless control of BLDC motors .