**I J C T A**, 9(26) 2016, pp. 389-395
© International Science Press

# Interpreting the Public Sentiment Variations on Twitter Using Virtual Shuffling for Data Movement in Big Data

**B. Arputhamary[1] and V. Jayapriya[2]**

**ABSTRACT**

The Map Reduce is an open source Hadoop framework implemented for processing and producing distributed large Terabyte data on large clusters. Its primary duty is to minimize the completion time of large sets of Map Reduce jobs. Hadoop Cluster only has predefined fixed slot configuration for cluster lifetime. This fixed slot configuration may produce long completion time (Make span) and low system resource utilization. Virtual shuffling that can enable efficient data movement and reduce I/O for Map Reduce shuffling, thereby reducing power consumption and conserving energy. In order to overcome the disk I/O problem of physical shuffling, virtual shuffling needs to address three important issues: (1) how to scalable represent intermediate data segments in a virtual manner, (2) how to minimize the impact of actual shuffling of data; and (3) how to dynamically coordinate and balance data shuffling and merging without degrading the performance.

*Keywords:* Map Reduce, Make span, Workload, Dynamic Slot Allocation.

## I. INTRODUCTION

Big data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not single technique or a tool; rather it involves many areas of business and technology. Big data involves the data produced by different devices and applications. Given below are the some fields that come under the umbrella big data.

### Black Box Data

It is a component of helicopter, airplane and jets, etc. it captures voices of the light of the crew, recordings of microphone and earphones, and the performance information of the aircraft.

### Social Media Data

Social media such as face book and twitter hold information and the views posted by millions of people across the globe.

### Stock Exchange Data

The stock exchange data holds information about the 'buy' and the 'sell' decisions made on the share of different companies made by the customers.

### Power Grid Data

The power grid data holds information consumed by a particular node with respect to a base station.

[1] Assistant professor, Department of Computer Applications, Bishop Heber College, Tiruchirappalli, India. *E-mail: arputhambaskaran@rediffmail.com*

[2] Research Scholar, Department of Computer Science, Bishop Heber College, Tiruchirappalli, India. *E-mail: jayapriya253@gmail.com*

### Transport Data

Transport data includes model, capacity, distance and availability of a vehicle.

### Search Engine Data

Search engine retrieves lots of data from different databases.

Thus big data involves huge volume, high velocity, and extensible variety of data. The data in it will be three types:

- Structured data: relational data.
- Semi structured data: XML data.
- Unstructured data: word, PDF, Text, Media logs.

### Map Reduce

Map Reduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of a large amount of data(multi tera byte data sets), on large clusters of commodity hardware in a reliable, fault-tolerant manner. The Map Reduce program runs on hadoop which is an apache open source framework.

## II. LITERATURE SURVEY

In literature, there was research study on performance optimization of Hadoop MapReduce jobs. An essential way for upgrading the performance of a Map Reduce job is dynamic slot configuration and job scheduling. J. Polo etal.[1] calculated the map and reduce task completion time dynamically and update it every minute during job execution. Task scheduling policy was based on the priority of each job. Priority was estimated based on the concurrent allocation of jobs. The dynamic scheduler is pre-emptive. It affects resource allocation of low priority jobs.

J. Wolf *et al.* [2] implemented flexible scheduling allocation scheme with Hadoop fair scheduler. A primary concern is to optimize scheduling theory metrics, response time, makespan, stretch, and Service Level Agreement. They proposed penalty function for measurement of job completion time, epoch scheduling for partitioning time, moldable scheduling for job parallelization, and malleable scheduling for different interval parallelization.

J. Dean *et al.* 2008 [3] discussed Map Reduce programming model. The Map Reduce model performs operations using the map and reduces functions. Map function gets input from user documents. It generates intermediate key/value for reducing function. It further processes intermediate key/value pairs and provide output key/value pairs. At an entry level, MapReduce programming model provided the best data processing results. Currently, it needs to process the large volume of data. So it provides some consequences while processing and generating data sets. It takes much execution time for task initialization, task coordination, and task scheduling. Parallel data processing may lead to inefficient task execution and low resource utilization.

Verma *et al.* [4] proposed deadline aware scheduler, called SLO scheduler. The SLO scheduler takes decisions of job ordering and slot allocation. This scheduler's primary duty is to maximize the utility function by implementing the Earliest Deadline First algorithms. It measures how many numbers of slots required for scheduling the slots dynamically with a particular job deadline.

B. Sharma *et al.* [5] proposed a global resource manager for the job tracker and a local resource manager for the task tracker. A global resource manager function is to manage each MapReduce task. It processes resource needs and resource assignments for each task. A local resource manager's duty is to identify each task. It examines resource usage and task completion time of the task. It deals with detecting bottlenecks with resources and resource contention.

J. Wang *et al.* [6] proposed fair slot setting for dynamically allocate available slots to particular tasks. They used FRESH for static and dynamic slot configuration. The static slot configuration slots are allocated before cluster launch based on previous task execution records. It uses deduct workload function to update current workloads of running jobs in the cluster. The fair scheduler was proposed to achieve fairness metric. The dynamic slot assignment slots are allocated during task execution. It used Johnson indices to represent the level of fairness.

## III. EXISTING SYSTEM

We take a new perspective at data shuffling of MapReduce programs. In the default Hadoop implementation, intermediate data segments are pulled by Reduce Tasks in their entirety to local disks, and then merged before being reduced for final results. Because the physical movement of entire segments across disks, we refer to this strategy as physical shuffling.

## IV. PROPOSED SYSTEM

Virtual shuffling that can enable efficient data movement and reduce I/O for MapReduce shuffling, thereby reducing power consumption and conserving energy. In order to overcome the disk I/O problem of physical shuffling, virtual shuffling needs to address three important issues: (1) how to scalable represent intermediate data segments in a virtual manner, (2) how to minimize the impact of actual shuffling of data; and (3) how to dynamically coordinate and balance data shuffling and merging without degrading the performance.

## ADVANTAGE OF PROPOSED SYSTEM

Virtual shuffling significantly relieves the disk I/O contention problem and speeds up data movement in Map Reduce programs. In addition, it reduces power consumption of Map Reduce programs by as much as 12%

**Modules**

1. Three-level segment table
2. Demand merging
3. Dynamic and balanced sub trees

### *1. Three-level segment table*

The classic concept of virtual memory in designing virtual shuffling. To manage many intermediate data segments produced by Map Tasks, we design a three-level segment table to organize them in a scalable manner. The hierarchical table includes three kinds of directories: the Segment Table Directory, the Segment Middle Directory, and the Segment Global Directory. At the completion of a Map Task, its data segment is not physically fetched all at once by a Reduce Task. Instead, a Segment Table Entry (STE) is created at the lowest level-Segment Table Directory (STD)-to represent the segment in a virtual manner. The STE includes several attributes of the segment such as its total length, its source Map Task, as well as its physical location on the remote disk.

### *2. Demand Merging*

Virtual shuffling mimics the concept of demand paging and realizes near-demand merging to reduce the pressure of data spilling and minimize the impact of actual data movement. With near-demand merging, we

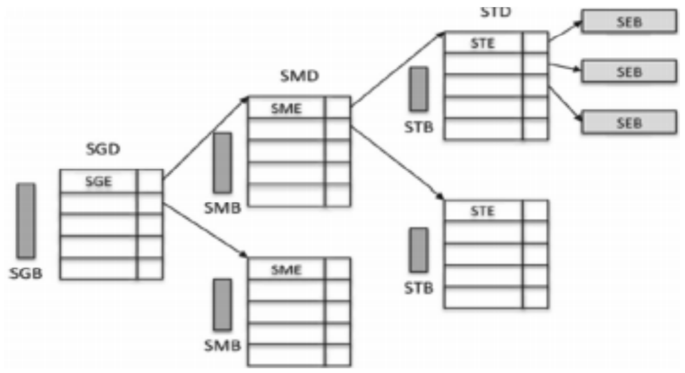


**Figure 1: Three Level segment Table**

wait until it is clear which segments are needed by the reduce function of Reduce Tasks. Near-demand merging does not really wait until the last moment to fetch data. Instead, it works as part of virtual shuffling to form a pipeline of fetching, merging and reducing data segments, with an emphasis on hiding the cost of data transfer over the network.

### *3. Dynamic and Balanced Sub trees*

We employ a dynamic orchestration mechanism to manage the merging of virtual segments. Instead of activating all leaves, we organize the whole tree as many sub trees, each composed of an STD and its SEBs. At any time, only a limited number of sub trees are actively merging data. Two sub trees are currently active in merging its SEBs into STBs. The merged data will be further merged to SMBs and/or SGBs. To balance the merging progress at different sub trees, previously active sub trees will be deactivated to allow other sub trees to make progress.

## IMPLEMENTATION

Implementation modeling consist of mathematical analysis and proof of concept validation under general and customized section of environment of HADOOP

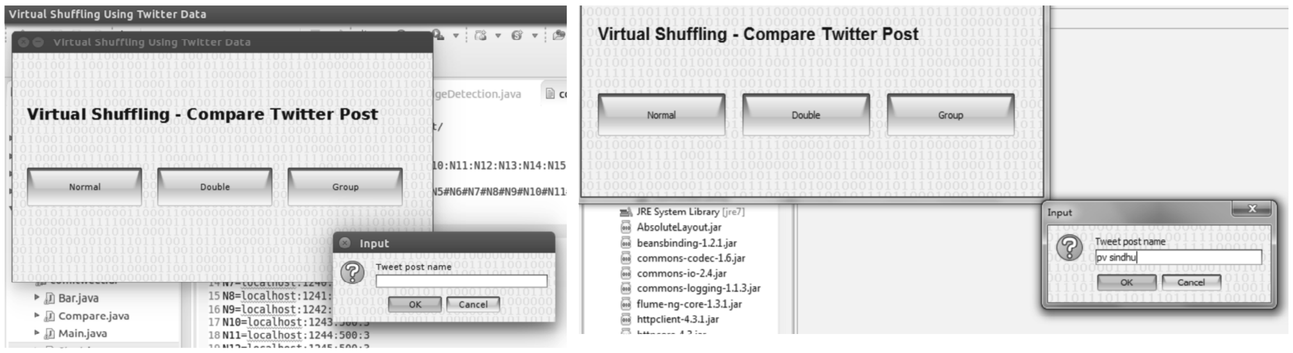
*Algorithm: Novel Implicit Shuffling and Data analysis algorithm*

**INPUT: Keyword**

**Output:Positive/Negative Command**

1. Dataset/ keyword collection and analysis

$$S = \{s_1, s_2, s_3, \ldots s_n\}$$
$$\text{choice} = \{\text{single, double, multi}\}$$



```
10:N11:N12:N13:N14:N15
N5#N6#N7#N8#N9#N10#N11
N8=localhost:1241:
N9=localhost:1242:
N10=localhost:1243:
N11=localhost:1244:500:3
```

Tweet post name: pv sindhu

**Figure 2: Compare twitter post with positive and negative command**

```
if (single/normal)
    $s = s_1$
else if (Double)
    $s = \{s_1, s_2\}$
    acquire ($s_1$)
    acquire ($s_2$)
else (Group)
    for $j = s_1$ to $s_j$ where j is no of nodes per keyword
    {
        acquire ( $j$)   }
```

2. Connection establishment and twitter data requesting

$$S = \{s_1, s_2, s_3, ...s_n\} = \sum_{j=1}^{\text{choice}} Sj$$

$$\text{Packet} \left[\sum_{j=1}^{\text{choice}} (Sj)\right]$$

3. Tweeter-end:

$$\text{Analyse} \left(\sum_{j=0}^{\text{choice}} (Sj)\right)$$

$$\text{Fetch tweets } T = (t_1, t_2, t_3, ...t_n / T_i \text{€ String } (S_j))$$
$$T = \{t_1, t_2, t_3, ...t_n\}$$

$$D = \sum_{j=0}^{\text{choice}} \left(\int_0^n (t_n) \subseteq (Sj)\right)$$

4. Download Implicit Tweets

$$D_v = \left(\sum_{j=1}^{\text{choice}} \int_0^n (t_n) \subseteq (Sj)\right)_I$$

$$\text{Where } I : \text{Index}/I = \{I_1, I_2, I_3, ...I_n, I\,I_n\}$$

5. Analysis and Decision Making

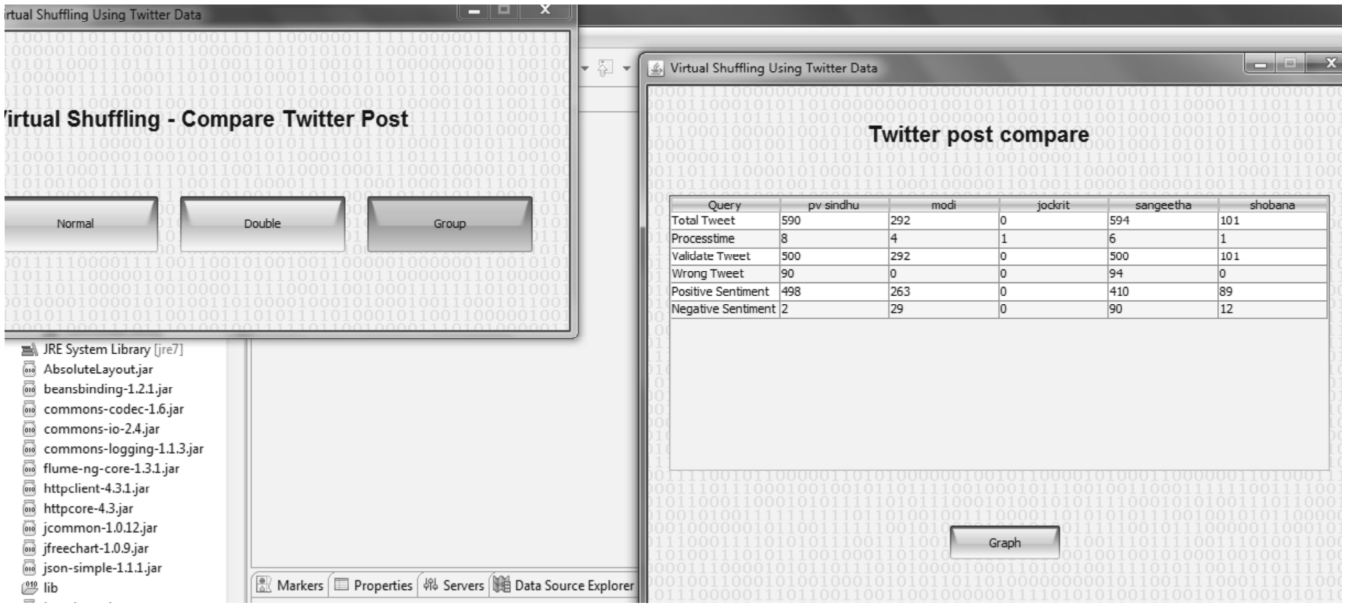
$$A = \int_0^n \left\{ \left( \frac{Ii}{Sj - ti} \right) \frac{d}{dt} (\text{keywords}) \right\}$$

If $\frac{d}{dt}(\text{keywords}) \subseteq \text{Positive}$ else negative

## V. CONCLUSION

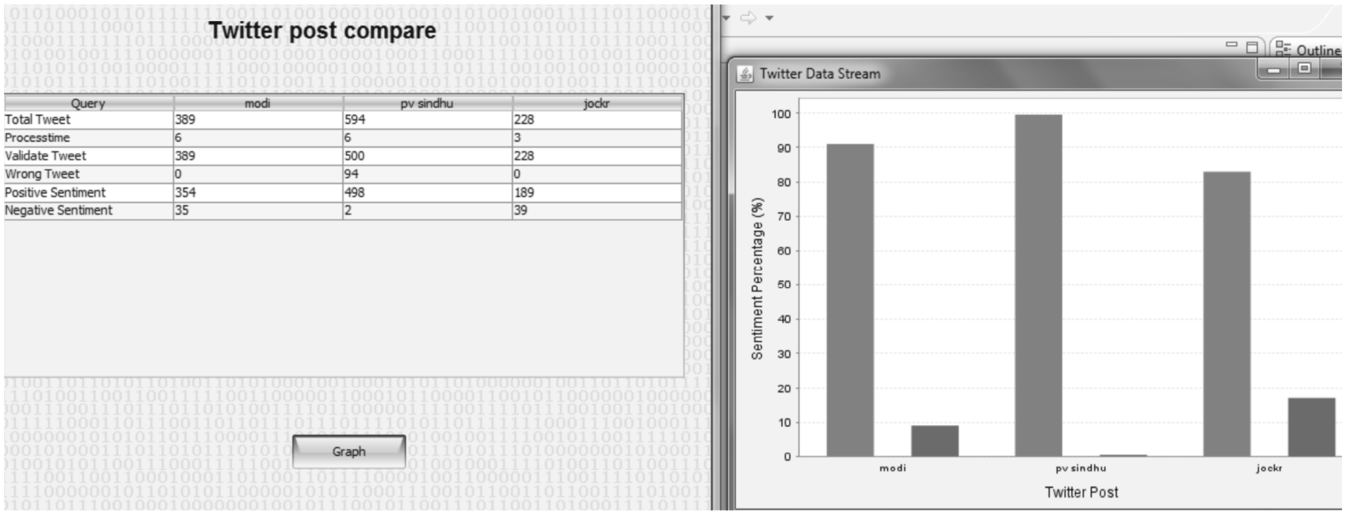
The proposed system is designed for extracting data from online big data repository for analysis and retrieving the same under virtual terminology, *i.e.* the system is designed for retrieving values and process online instead of downloading at local system. The time consummation in retrieving the tweets is drastically reduced and thus the performance is enhanced. The system is analyzed and verified for 200 samples under single, double and multiple modes, thus the system performance is effecting and has higher analysis of response time, the proposed system is programmed from Map Reduce loading balancing technique, thus the ratio of efficiency is improvised under this proposed system. As the system is designed and dedicated for tweeter

## Evaluation and Result



| Query | pv sindhu | modi | jockrit | sangeetha | shobana |
|---|---|---|---|---|---|
| Total Tweet | 590 | 292 | 0 | 594 | 101 |
| Processtime | 8 | 4 | 1 | 6 | 1 |
| Validate Tweet | 500 | 292 | 0 | 500 | 101 |
| Wrong Tweet | 90 | 0 | 0 | 94 | 0 |
| Positive Sentiment | 498 | 263 | 0 | 410 | 89 |
| Negative Sentiment | 2 | 29 | 0 | 90 | 12 |

**Figure 3:Twitter post Compare to Famous Leaders Positive Negative Level**



| Query | modi | pv sindhu | jockr |
|---|---|---|---|
| Total Tweet | 389 | 594 | 228 |
| Processtime | 6 | 6 | 3 |
| Validate Tweet | 389 | 500 | 228 |
| Wrong Tweet | 0 | 94 | 0 |
| Positive Sentiment | 354 | 498 | 189 |
| Negative Sentiment | 35 | 2 | 39 |

**Figure 4:Twitter post Compare to the prediction level**

data handle analysis and perform a good rate of analysis. Thus improvement can be made under an active manner for dynamic load shifting from one social media to another. This media based movement is complicated as the system configuration and proxy settings are varies, hence the challenge can be addressed in future.

## REFERENCES

[1] J. Dean and S. Ghemawat, "Map Reduce: simplified data processing on large clusters", in *Communications of the ACM*, **51**, 107-113, 2008.

[2] J. Polo and D. Carrera and Y. Becerra *et al.*, "Performance-driven task co scheduling for Map Reduce environments", *IEEE Network Operations and Management Symposium in NOMS'10*, 373-380, 2010.

[3] J. Wolf and D. Rajan and K. Hildrum and R. Khandekar and V. Kumar and S. Parekh, K.-L. Wu, and A. Balmin, "Flex: A slot allocation scheduling optimizer for Map Reduce workloads", *ACM/IFIP/USENIX 11th International Middleware Conference*, 1-20, 2010.

[4] Apache Hadoop. Reference link: http://hadoop.apache.org.

[5] A. Verma and L. Cherkasova, and R. H. Campbell, "ARIA: Automatic Resource inference and allocation for Map Reduce environments", *ICAC'11 Proceeding of the 8th ACM International Conference on Autonomic Computing*, 234-244, 2011.

[6] Apache Hadoop YARN (yet another resource negotiator) Reference

Link: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoopyarn-site/YARN.html.

[7] B. Sharma and R. Prabhakar and S.-H. Lim *et al.*, "Mrorchestrator: A finegrained resource orchestration Framework for Map Reduce clusters", *Colud Computing (CLOUD) 5th international Conference'12*, 1-8, 2012.

[8] V. K. Vavilapalli and A. C. Murthy and C. Douglas and S. Agarwal and M. Konar, R. Evans and T. Graves and J. Lowe and H. Shah and S. Seth *et al.*,"Apache Hadoop yarn: Yet another resource negotiator", *SOCC'13 Proceedings of the 4th annual Symposium on Cloud Computing. ACM*, 2013.

[9] Jiayin Wang and Yi Yao and Ying Mao and Bo Sheng, "FRESH: Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters", *IEEE 7th International Conference on Cloud Computing*,761-768, 2014.

[10] Capacity scheduler Reference Link:https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

[11] S. Tang and B. Lee, and B. He," Dynamic MR: A Dynamic Slot Allocation Optimization Framework for Map Reduce Clusters", *IEEE Transactions on Cloud Computing*, 2,333-347, 2014.

[12] A. Bansal, A. Deshpande, P. Ghare, S. Dhikale, B. Bodkhe, " Healthcare Data Analysis using Dynamic Slot Allocation in Hadoop", *International Journal of Recent Technology and Engineering*,3,2277-3878, 2014.

[13] A.U.Patil and T.I Bagban and A.P. Pande, "Recent Job Scheduling Algorithms in Hadoop Cluster Environments: A Survey", *International Journal of Advanced Research in Computer and Communication Engineering*, **4**, 127-149, 2015.

[14] Z. Li, Q. Zhang, M. F. Zhani, R. Boutaba, Y. Liu, Z. Gong *et al.*, "DREAMS: Dynamic Resource Allocation for Map Reduce with Data Skew", *Integrated Network Management IFIP/IEEE International Symposium*, 2015.

[15] Y. Yao, J. Wang, B. Sheng, C. Tan and N. Mi, "Self-Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters", *IEEE Transactions on Cloud Computing*, 2015.