DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.5210

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

PARALLEL AND DISTRIBUTE PROCESSING FOR VIRTUAL MAPREDUCE CLUSTERS BY USING IMPROVISED HYBRID JOB SCHEDULING ALGORITHM

Dr N Sandhya
Professor, Department of CSE
VNRVJIET Hyderabad, India

Sravan Kumar Kanthi
M. Tech, Department of CSE
VNRVJIET Hyderabad, India

Abstract: MapReduce is a programming model that defines a MapReduce job for instance, a map perform task then reduce perform task. This model splits the job into several map perform tasks and reduce perform tasks at run time. It also accomplishes these tasks in parallel on a MapReduce cluster. We have researched a resourceful and appropriate scheduling scheme called as hybrid job-driven scheduling scheme (JoSS) which source higher map and reduce data-locality. But, in this existing JoSS scheduling scheme, virtual MapReduce cluster does not provide flexibility over multiple workloads for load balancing. For this purpose, we have enriched native JoSS with advanced JoSS by adding a new functionality called virtual MapReduce cluster to provide flexibility to JoSS. This enhanced work achieves load-balancing and also improves job performance.

Keywords: MapReduce, Virtual MapReduce Cluster, Data Locality, JoSS.

I. INTRODUCTION

In current years, MapReduce [1] is a well-known version designed for data-extensive computation [2]. Considering specific behaviors and global performance objectives over numerous jobs, job performance can be increased through schedulers and this is vital in MapReduce/Hadoop [2]. For those several jobs which run very slow for Hadoop mapreduce, resource aware scheduling methodology was enhanced by these schedulers. Based on process outline and employment of utilities, we need to dynamically alter the slots allocation where the present algorithm has such impact on profiling data. Besides the workload placement, it also increases the resource utilization of the cluster.

In a Hadoop cluster and also in a big cluster the assets are commonly placed far from one another because the community link sources have various bandwidth capabilities when compared to each other. Here with heterogeneous sources, communication expenses could be high if a project's distribution is maximized in a huge cluster. For this, the Hadoop device distributes responsibilities to multiple sources to reduce a process's total time. MapReduce is an information processing and a software technique for distribute computing which is developed in java. In MapReduce methodology Map and the Reduce tasks are the two main tasks. Map takes raw data and splits the data into different data sets and these data set elements are categorized as tuples (key/value pairs). Furthermore, Map task output is taken as input for reduce task and joins the information tuples into reduced set of tuples [2]. MapReduce indicates that, Map job and reduce job is performed one after other. For scalability, data processing above numerous computing nodes we adopt MapReduce framework. In MapReduce algorithm, data processing jobs/tasks are referred as mappers and reducers. Mappers and reducers are crucial in data processing splitting utility. Implementing an approach in MapReduce model, which scales the software to run above multiple loads or several multiple of nodes in a cluster, is just a configuration change. The MapReduce model's high scalability has fascinated many programmers attention to adopt this model.

Many MapReduce Frameworks like Google MapReduce, Dryand, are adopted by the users. However the free source supply Hadoop MapReduce is widely used. In maintaining the global performance of MapReduce Applications scheduling plays a prominent role. FIFO Scheduler is the standard scheduler in Hadoop MapReduce, Fair Scheduler is used by facebook, and Capacity Scheduler is used by Yahoo [2]. These schedulers are regular examples of schedulers for MapReduce application. But these schedulers unable to handle the functions distressed by virtualization used in cloud environments. Therefore, based on software capabilities, Virtual Machines and locality of records, by enhancing native schedulers with a dynamic scheduler could schedule MapReduce packages substantially. Besides this, the dynamic scheduler successfully executes these packages in hybrid cloud environment.

II. RELATED WORK

Jongse Park, Daewoo Lee, Bokyeong Kim, Jaehyuk Huh, Seungryoul Maeng planned and evaluated Dynamic Resource Reconfiguration (DRR) [3]. Their work evaluated for distributed data-intensive platforms on virtualized cloud environments and developed this DDR using a dynamic VM reconfiguration mechanism. DRR improves record segment of a digital MapReduce cluster by concise increase in VMs to run local responsibilities. DRR schedules locality and fine-tune the working ability of virtual nodes. This methodology differs from earlier approaches assuming a cluster that continually features a fastened quantity of procedure resource in every node. For balancing unfair distribution, dynamic VM reconfiguration is extended to differing kinds of load for distributed data-intensive platforms which are not balanced properly. Necessities by different jobs or tasks for different resources may accompany to inappropriate utilization of every single virtual node resource [3]. With VM reconfiguration, every node is adjusted to produce the mandatory quantity of resource demanded for the node [3]. A framework supporting dynamic VM reconfiguration is their future work.

A. Matsunaga, M. Tsugawa, and J. Fortes researched and investigated over bioinformatics applications. Their research is an operative method because it validates its low outlays and increased performance. They have implemented CloudBLAST, a distributed implementation of NCBIBLAST [4] which achieves high performance. The purpose of this implementation

is that it executes parallel by integrating MapReduce applications where software environment is encapsulated and virtual networks connect data in virtual machines [4].

In Cloud Computing configurations like Hadoop, MapReduce etc [5] large scale processing is a gradual process and is common. In such systems, files are riven into tiny blocks and every single block is duplicated over many servers. To increase files efficiency, each single job is riven into several tasks and every single task is circulated to a server to override a file block [5]. For the task scheduler it is vital to improve data locality. Vaishali W. Thawari, Sachin D. Babar, Nitin A. Dhawas presented data locality driven task scheduling algorithm [5] also known as the Balance-Reduce algorithm. According to the workload and network state, BAR algorithm fine-tunes task data locality and also schedules task using its global view. With this algorithm data locality can be improved in poor network environment.

In a planet platform, the network state and the cluster work modification occur very often. Therefore it's essential to update the programming strategy by an efficient rescheduling algorithmic program to handle machine failure and network anomaly. Yet, the rescheduling occurred regularly by the scheduler, the rescheduling algorithm ought to be less quality.

Chen He, Ying Lu, David Swanson developed a novel technique [6] to increase the information locality. This technique allocates tasks to a node. A native map tasks are invariably most popular over non-local map tasks. A neighborhood marker is utilized to mark nodes and also to confirm every node is in a good probability to acquire its local tasks. Experimental results show that this technique achieves the best information locality rate and therefore the minimum delay for map tasks [6]. This technique is an alternative to the delay algorithmic program [7] where their technique doesn't need the tuning of the parameter [6].

The MapReduce programming version has been used at Google for its special features. Jeffrey Dean and Sanjay Ghemawat attribute these features to many motives. Firstly, this methodology is used by the programmers who don't have experience in parallel and allotted systems since it hides the principle factors of fault-tolerance, loads balancing, parallelization and section optimization [1]. Secondly, an outsized form of issues is simply represented as MapReduce computations. For example, Google's production internet search provider uses MapReduce technology of records for sorting, data processing, device gaining knowledge and masses of opportunity structures [1]. Finally, they need advanced application of MapReduce programming paradigm which balances large clusters of machines. This application makes cost-effective use of those machine resources and is appropriate to use on several big procedure issues encountered at Google [1].

J. Polo, D. Carrera, et al. represented a model designed on task scheduling for MapReduce applications which is implemented on top of Hadoop. Apache introduced a free source application of MapReduce framework which is called as Hadoop [8]. The performance time is estimated dynamically through the scheduler for every MapReduce process within the system. Each MapReduce activity is composed of a high range of tasks (maps and reduces) recognized earlier through the initialization section activity and the results of the tasks are calculated at runtime [8]. The scheduler takes both submitted and incomplete Hadoop tasks where it displays the average undertaking duration for already completed obligations [8]. This information services in guessing the best task execution time.

An interference and locality-aware scheduler for virtual MapReduce clusters [9] is employed by Xiangping Bu, Jia Rao, and Cheng-Zhong Xu. IASM and LASM are the two main design elements of this ILA. Through a performance prediction model, an interface-free design is performed with IASM. By exploitation apt able Delay planning algorithmic rule [7] [9], task information section is improved by LASM. Experimental outcomes exhibit that ILA pc hardware may accomplish a speeding of 1.5-6.5 times for person jobs and yield an improvement of up to at- least 1.9 instances in system turn-out as compared with four opportunity schedulers [9]. It improves statistics locality of map tasks. Though ILA planning algorithmic rule is meant for MapReduce framework, it can be applicable to virtual cluster schedulers.

Bikash Sharma, Timothy Wood, Chita R. Das give a two-phase hierarchical scheduler known as HybridMR, for hybrid server structures along with a combination of local and digital systems to governor the aids of every paradigms [10]. In the first part, HybridMR outlines incoming MapReduce jobs to see the calculable virtualization overheads and make use of these records to automatically guide placement among bodily systems and digital systems [10]. In the second part, HybridMR builds dynamic resource prediction which plays dynamic aid planning to ease the interference amongst collocated MapReduce and interactive packages [10]. Problematic opinions on a hybrid cluster inclusive of 24 bodily servers, forty eight digital machines with numerous workload aggregate of interactive and batch MapReduce applications display that HybridMR achieves as much as 40% progress in process of completion time of MapReduce jobs over a digital Hadoop [10]. Further, HybridMR offers development in aid usage and energy savings as compared to a local Hadoop with smallest overall performance penalty. Besides, we have a tendency to show that it's run-time modification ability of the local and virtual cluster configurations to deal with versions in workload integration for increasing the performance.

Engin Arslan, Mrigank Shekhar, Tevfik Kosar propose the LoNARS formula for reduce task programming in MapReduce [11]. Experimenting with 12-server cluster to check the LoNARS performance as a micro benchmarking they proved that the existing Hadoop programming formula for reduce outperformed by LoNARS [11]. Besides this, a 100-server cluster is used to simulate macro-benchmarking and compared LoNARS to inventory accounting, Rack Aware, and CoGRS algorithms. The outcome showed that up to 25% of network traffic is reduced every time by LoNARS over above 3 algorithms and that marks as a vital effect in power consumption of network switches [11]. To overcome the worst job execution interval through LoNARS the shuffle transfer time should be over one heartbeat time because in most cases shuffle time is being reduced by LoNARS. This can be achieved by what quantity circulation a job shuffle within the shuffle quantity and this dependency is called the reduction quantitative relation.

III. FRAMEWORK

In this paper, to increase the JoSS flexibility, we are enriching the native JoSS with heterogeneous virtual MapReduce cluster [12]. In this proposed model, we can balance the different workloads of virtual MapReduce clusters. Through this we can generate multiple servers' equals to multiple jobs. Proposed system categorized into two sections - section A shows existing system and section B shows the load balancing for proposed system.

A. Existing System

Previously, in native JoSS, the job classification is done based on the ratio of predefined chunk size of map and reduce job where it addresses both map-data locality and reduce-data

locality in a digital MapReduce cluster. This job organization can be classified into either a Map-Heavy (MH) or Reduce-Heavy (RH) job or large job which is presented in figure 1. Figure 1 explains that existing JoSS uses three different benchmarks to conduct our trials.

1. Word-Count which counts the amount of incidences of each word occurred in data files.
2. Inverted-Index produces word-to-file indexing [12] by receiving one or more data files as an input.
3. Sort performs sorting and results the data by taking a data file as input.

These benchmarks are classified into either MH or RH or Large Jobs based on their filtering percentage value (FPJ value).

There are two categories present in JoSS [10] and are known as

1. Task-driven Task Assigner (TTA)
2. Job-driven Task Assigner (JTA)

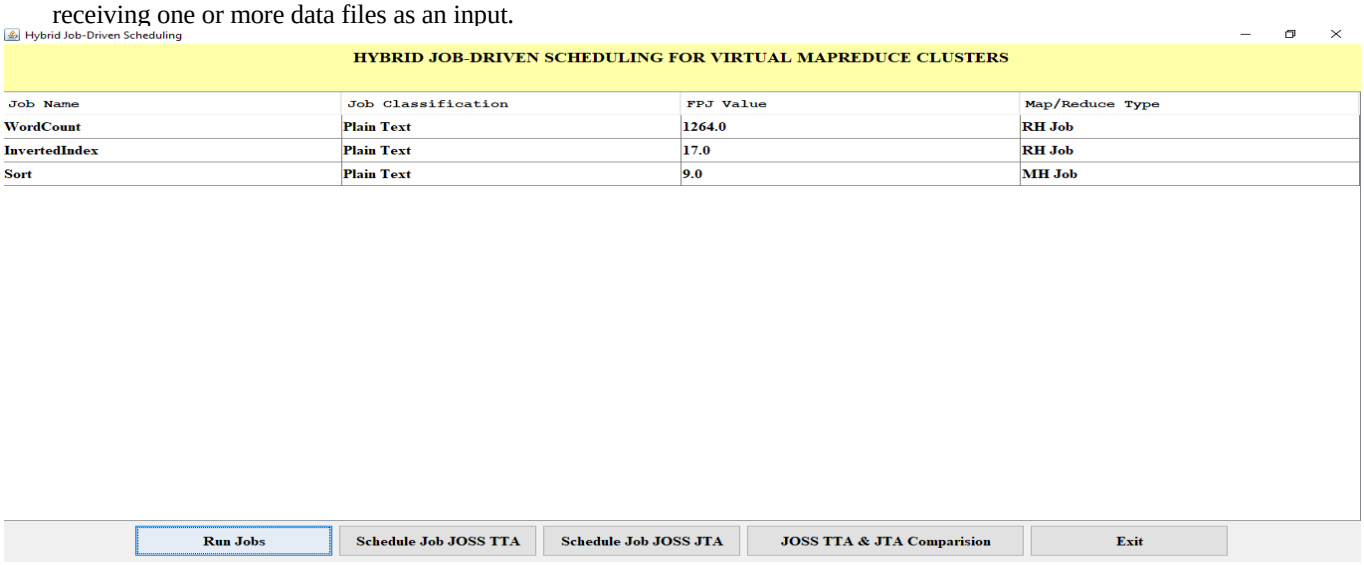

Figure 1 JoSS job categorization into MH, RH and Large jobs

1) Task-driven Task Assigner(TTA): Once the job classification is done based on the FPJ value we run the schedule job JoSS TTA which is presented in figure 1 and the above three benchmarks are processed through TTA scheduler for fast task assignment. Figure 2 exhibits the processed results of three benchmarks with TTA by simulatig virtual private severs. TTA assigner uses Hadoop FIFO algorithm where a map task is consigned from a map queue to VPS [12] which is its functionality. The main aim is to govern job cataloging by earning filtering percentage values and execution of all newly submitted jobs consequently. In TTA, other data center's map-task queue's, first Map task is assigned in a round-robin procedure and through this the consignment of tasks can be finished quickly.

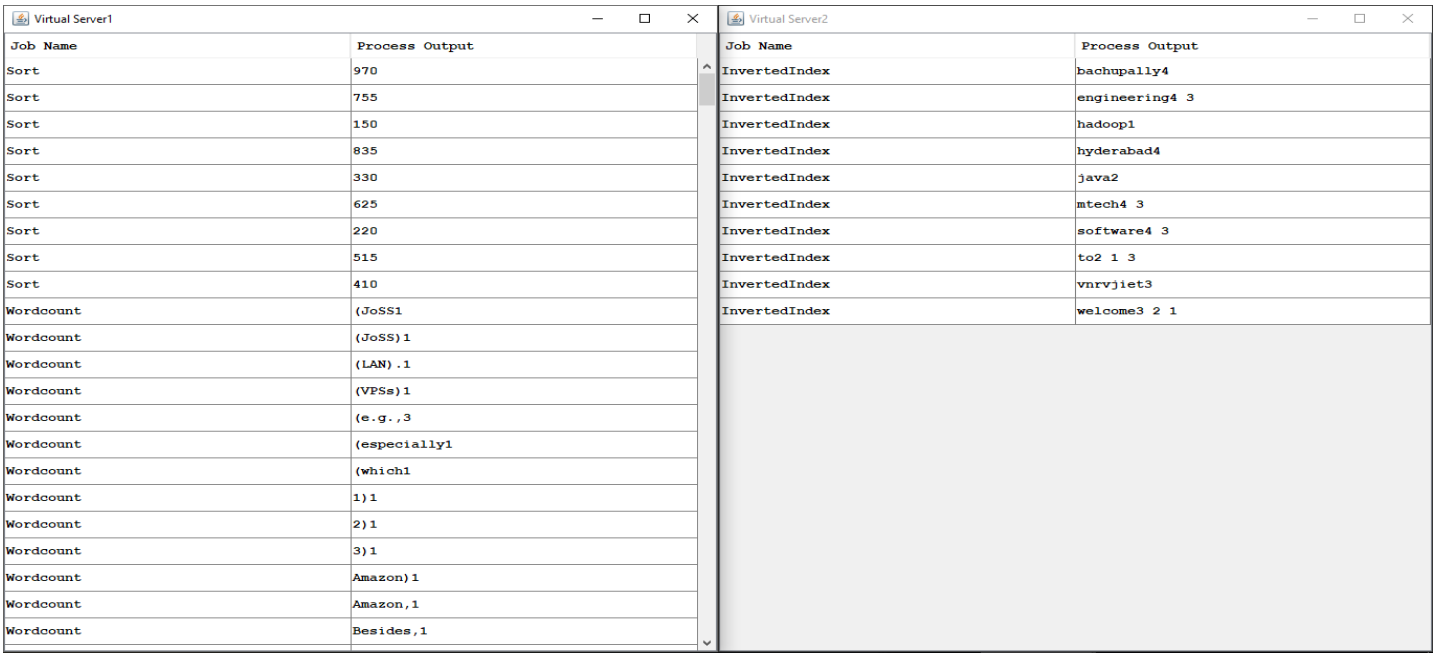

Figure.2 JoSS Task-driven Task Assigner (TTA)

2) Job-driven Task Assigner(JTA): Schedule JoSS JTA, which works same as of TTA which is shown in figure 1 and the above three benchmarks are processed through JTA scheduler presented in figure 3. Figure 3 exhibits the processed results of three benchmarks with JTA by simulating virtual private severs. In JTA, for allocating a map task Hadoop FIFO algorithm is opted which consigns from every single map-task queue which is a key difference between JTA and TTA and also improves VPS-locality [12]. Furthermore, with the aid of categorizing jobs into large jobs, MH jobs, RH jobs and through round-robin model scheduling, job starvation can be avoided and increases performance of job [12] presented in figure 4. In figure 4, we estimate and equate both TTA and JTA with each other. This tentative outcome validate that both TTA and JTA provide a improved map-data locality, succeed a higher reduce-data locality [12]. It also results that when mapreduce jobs are all small, TTA is highly suitable than JTA for virtual mapreduce cluster. Besides, when mapreduce jobs not all small, JTA is more applicable because the workload improvement time is very short. However, the JoSS scheme doesn't provide flexibility for load balancing.

Virtual Server1

Job Name	Process Output
Wordcount	(JoSS1
Wordcount	(JoSS)1
Wordcount	(LAN).1
Wordcount	(VPSs)1
Wordcount	(e.g.,3
Wordcount	(especially1
Wordcount	(which1
Wordcount	1)1
Wordcount	2)1
Wordcount	3)1
Wordcount	Amazon)1
Wordcount	Amazon,1
Wordcount	Besides,1
Wordcount	But1
Wordcount	But,1
Wordcount	Cen-locality,1
Wordcount	Cenlocality1
Wordcount	Consequently,1
Wordcount	Due3
Wordcount	Each1
Wordcount	Face1
Wordcount	For1

Virtual Server2

Job Name	Process Output
InvertedIndex	bachupally4
InvertedIndex	engineering4 3
InvertedIndex	hadoop1
InvertedIndex	hyderabad4
InvertedIndex	java2
InvertedIndex	mtech4 3
InvertedIndex	software3 4
InvertedIndex	to1 2 3
InvertedIndex	vnrvjiet3
InvertedIndex	welcome3 2 1

Figure 3 JoSS Job-driven Task Assigner (JTA)

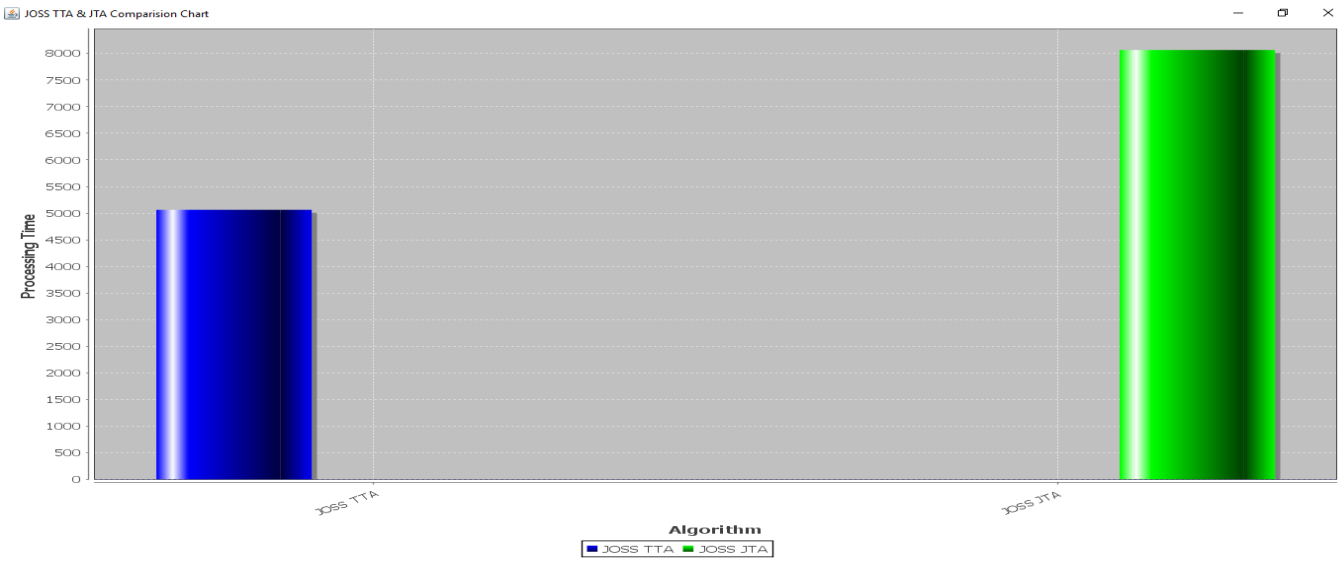

Figure 4 Comparison chart of TTA & JTA

B. Load Balancing

To disperse the load equally throughout the idle nodes Load balancing [13] method is used, while a node is loaded over its threshold degree. Load balancing is important while managing immense documents for processing and during this process hardware assets usage is vital because load balancing is not ample in MapReduce design ethics. Besides this, it achieves adequate progress in global performance and also hardware employment in resource need circumstances. In HDFS [14] cluster few data nodes become full or empty, so to balance the disk space utilization load balancing technique is presented. In this method to determine whether the cluster is balanced or not we need to calculate the threshold value where it varies between 0% - 100% and the default value is 10%.

To determine a cluster is balanced if- "for every record node, the ratio of used area on the node to the total capability of node (node usage) differs from the cluster's used space ratio to the total capability of the cluster (cluster usage) not higher than the threshold value." Here the balancer is time-consuming for its execution but it balances cluster extremely if the value is small.

IV. EXPERIMENTAL RESULTS

In this experiment, we are also taking three types of jobs named as WordCount, Inverted Index and Sort presented in figure 5. We have to run these three jobs by using MapReducers (explained in previous section existing system). The proposed or enhanced work can assign an individual virtual MapReduce server for every individual job presented in figure 6. Figure 6 exhibits all the results of each benchmark in an individual server for load balancing where in previous homogeneous virtual mapreduce cluster the same results are exhibited with predefined virtual severs (2 virtual servers).

We evaluate and compare Virtual mapreduce cluster with both JoSS TTA and JoSS JTA presented in figure 7. In figure 7 the extensive experiment outperforms both JoSS-TTA and JoSS-JTA. It works same as existing homogeneous virtual mapreduce cluster but our experiment achieves fast task performance than TTA. In figure 7 we demonstrated that the implemented scheduling scheme can enhance the performance time of jobs and balancing of different workloads.

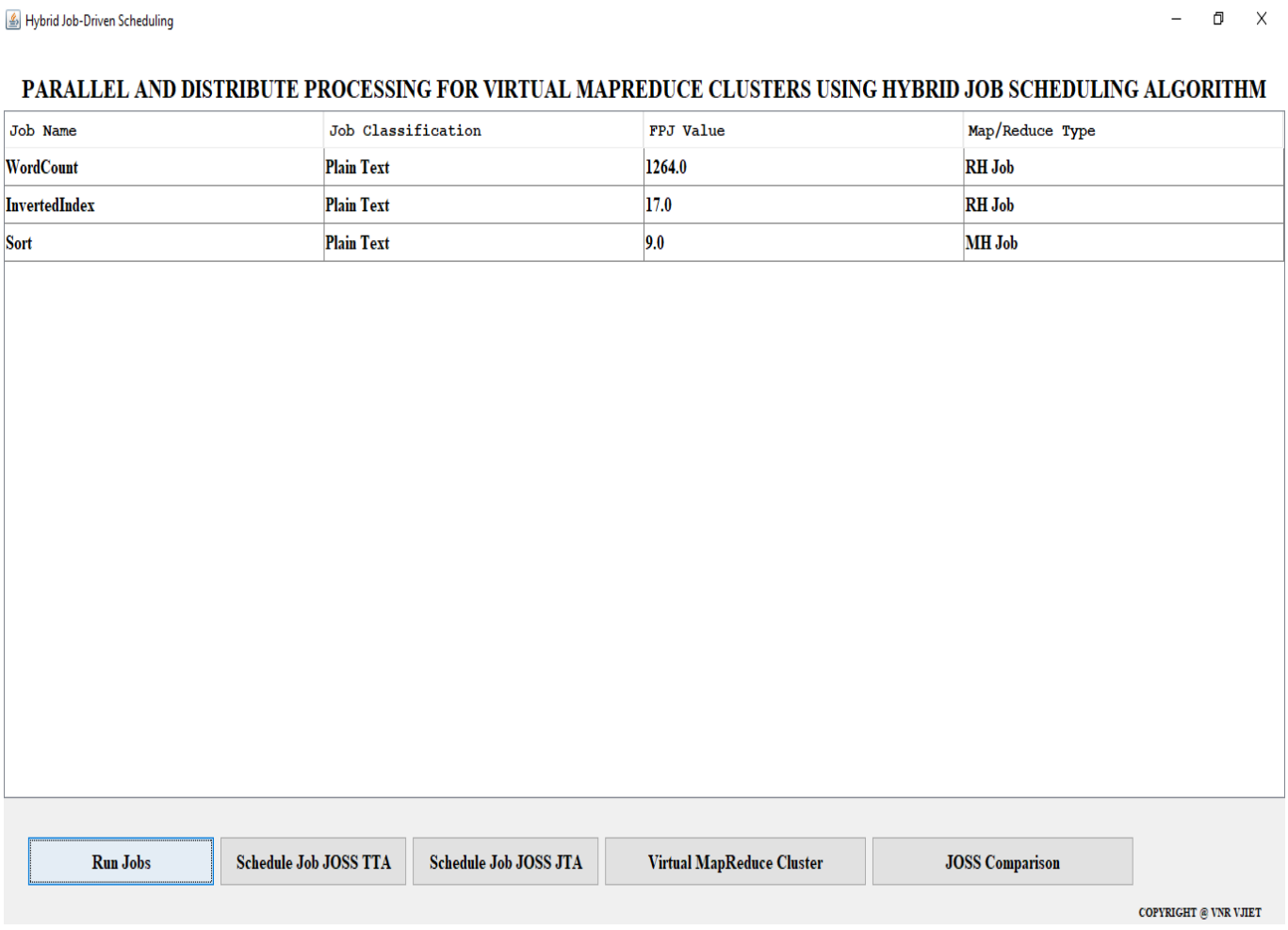

Figure 5 Heterogonous Virtual MapReduce cluster

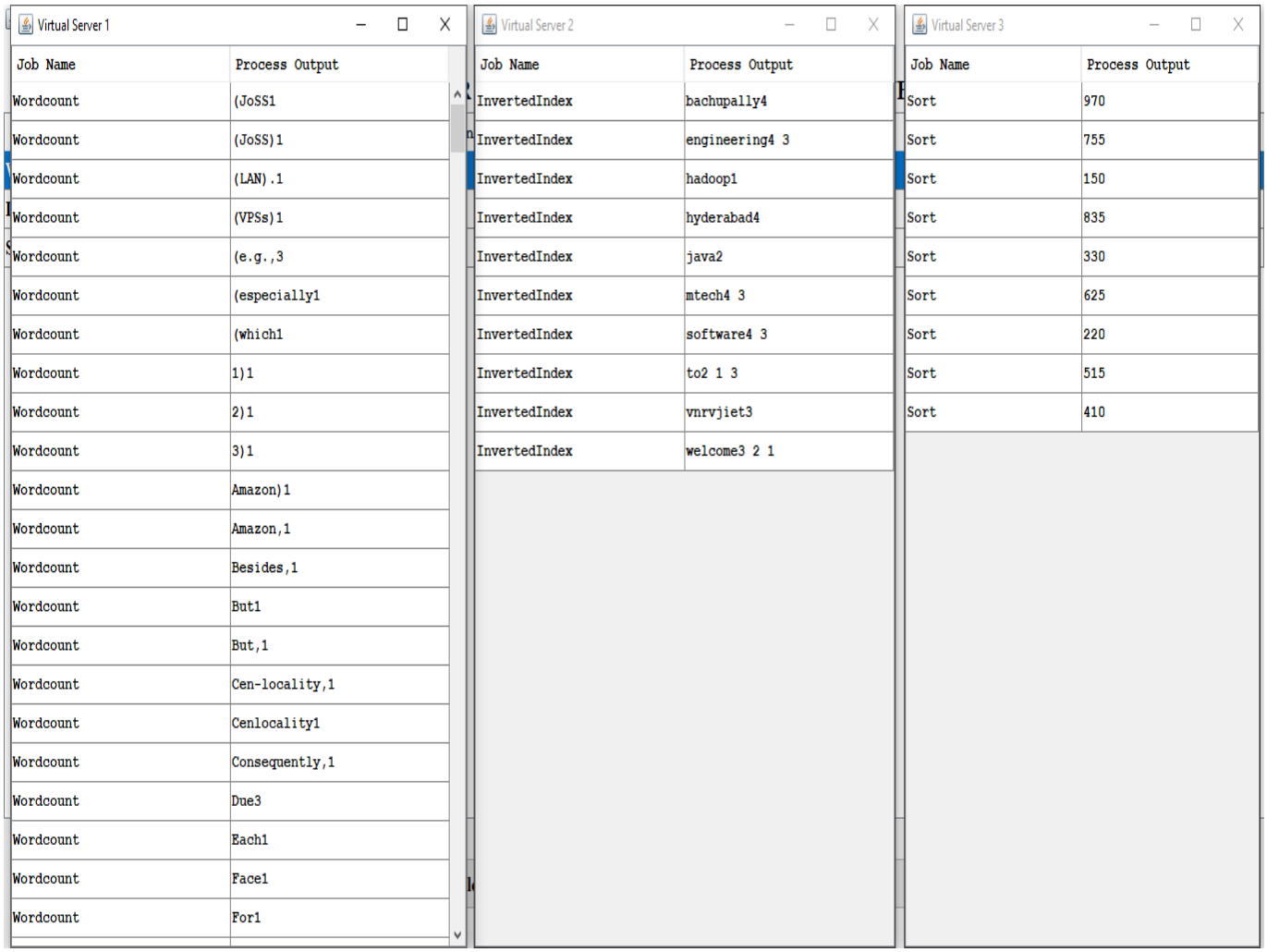

Figure 6 JOSS flexibility using Virtual MapReduce clusters

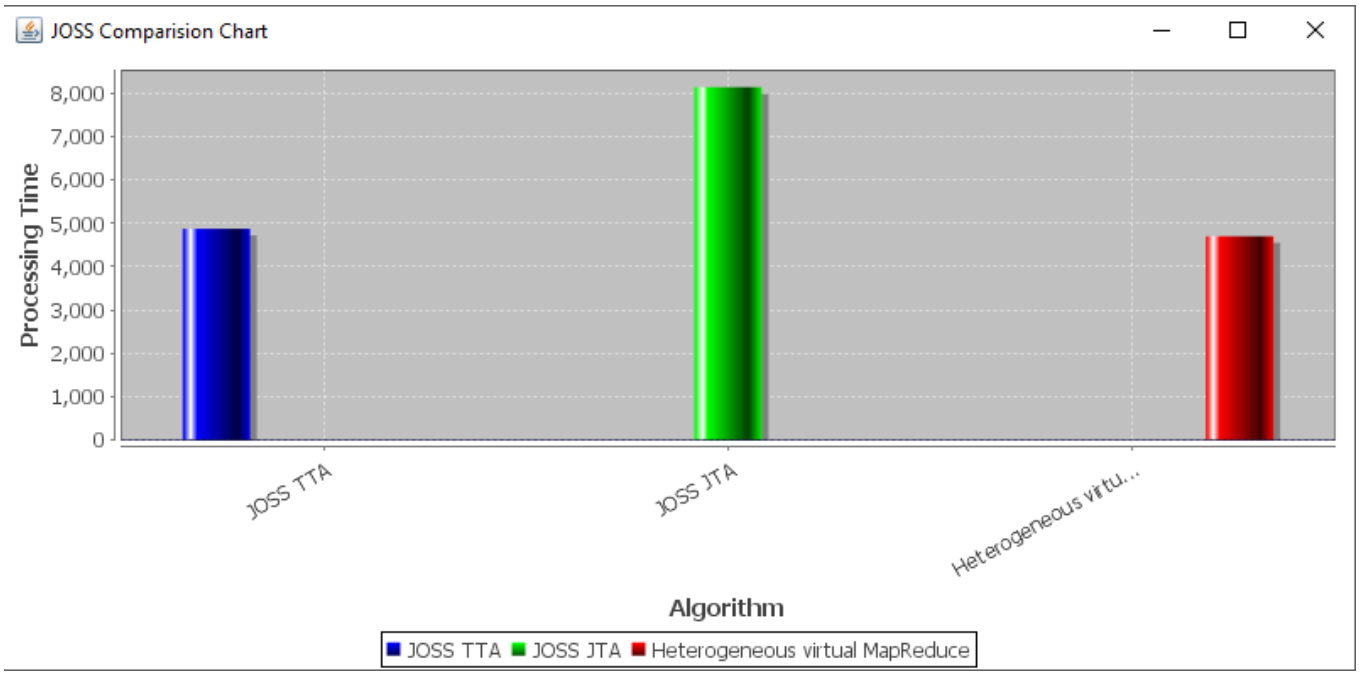

Figure 7 Comparison chart of TTA, JTA & Heterogeneous Virtual MapReduce cluster

V. CONCLUSION

In this paper, we have enriched the native JoSS work with an advanced virtual MapReduce cluster technique. In the existing JoSS, the virtual MapReduce clusters are homogeneous so we cannot balance the workloads and we need to improve the JoSS flexibility. So, we have enriched native JoSS by adding a new functionality called virtual MapReduce cluster which provides flexibility to native JoSS and also increase in the performance. Our experiments also proved that the new approach increases the performance than the previous approach.

VI. REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, 1 Jan. 2008, pp. 107–113, doi:10.1145/1327452.1327492.

[2] B. Anusha and S. Afroz. "Mitigating network traffic while job execution and improving the job performance for mapreduce clusters." International Journal of Engineering Research in Computer Science and Engineering (IJERCSE), Vol. 4, no. 11, Nov. 2017, pp. 304-308.

[3] J. Park, D. Lee, B. Kim, J. Huh, and S. Maeng, "Locality-Aware dynamic VM reconfiguration on MapReduce clouds." Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing - HPDC 12, 2012, pp. 27–36, doi:10.1145/2287076.2287082.

[4] A. Matsunaga, M. Tsugawa, and J. Fortes, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," 2008 IEEE Fourth International Conference on eScience, 2008, pp. 222–229, doi:10.1109/escience.2008.62.

[5] V. W. Thawari, S. D. Babar, N. A. Dhawas, "An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing," International Journal in Multidisciplinary and Academic Research (SSIJMAR), vol. 1, no. 3, 2012.

[6] C. He, Y. Lu, and D. Swanson, "Matchmaking: A New MapReduce Scheduling Technique," 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Nov. 2011, pp. 40–47, doi:10.1109/cloudcom.2011.16.

[7] M. Zaharia, D. Borthakur, et al. "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," Proceedings of the 5th European conference on Computer systems - EuroSys 10, Apr. 2010, pp. 265–278, doi:10.1145/1755913.1755940.

[8] J. Polo, D. Carrera, et al. "Performance-Driven task co-Scheduling for MapReduce environments," 2010 IEEE Network Operations and Management Symposium - NOMS 2010, 2010, pp. 373–380, doi:10.1109/noms.2010.5488494.

[9] X. Bu, J. Rao, and C.-Z. Xu, "Interference and locality-Aware task scheduling for MapReduce applications in virtual clusters," Proceedings of the 22nd international symposium on High-Performance parallel and distributed computing - HPDC 13, 2013, pp. 227–238, doi:10.1145/2493123.2462904.

[10] B. Sharma, T. Wood, C. R. Das, "HybridMR: A Hierarchical MapReduce Scheduler for Hybrid Data Centers," 2013 IEEE 33rd International Conference on Distributed Computing Systems, 2013, pp. 102–111, doi:10.1109/icdcs.2013.31.

[11] E. Arslan, M. Shekhar, and T. Kosar, "Locality and Network-Aware Reduce Task Scheduling for Data-Intensive Applications," 2014 5th International Workshop on Data-Intensive Computing in the Clouds, 2014, pp. 17–24, doi:10.1109/datacloud.2014.10.

[12] M. Lee, J. Lin, and R. Yahyapour, "Hybrid Job-Driven Scheduling for Virtual MapReduce Clusters," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 6, Jan. 2016, pp. 1687–1699, doi:10.1109/tpds.2015.2463817.

[13] S. Shaheena, SD. Afzal A and P. Sham. "Solving load rebalancing for distributed file system in cloud." International Journal of Advances in Applied Science and Engineering (IJAEAS), vol. 1, no. 3, 2014.

[14] Hadoop Distributed File System, http://hadoop.apache.org/hdfs/, 2012.