Task Selection for Scheduling using Hadoop Scheduler

D C Vinutha, G T Raju

Abstract: MapReduce is a prevalent model for data intensive applications. This covers the difficulties of parallel programming and provides an abstract environment. Hadoop is a benchmark for Big Data storage by being able to provide load balancing, scalable and fault tolerance operation. Hadoop output is mainly dependent on scheduler. Various algorithms for scheduling [6-10]have been suggested for various types of environments, applications and workload. In this work new task selection method is developed to facilitate the scheduler, if a node has several local tasks. Experimental result shows an improvement of 20% in respect of locality and fairness.

Keywords: Map Reduce, Hadoop Fair Scheduler, LATE.

I. INTRODUCTION

The complex problems in various fields have been solved using advancement in computing systems. These applications produce and processes huge amount of data. Cloud and Grid computing are developed to efficiently resolve the large scale data intensive applications. Conventional models are not effective to handle the Load balancing, debugging and scheduling of data-intensive applications. MapReduce is commonly used for data intensive applications. This covers the user difficulties in parallel programming and provides an abstract environment. Hadoop has features like load balancing, fault tolerance, scalability over clusters of commodity hardware. Hadoop performance is dependent on scheduler. Scheduling algorithms are designed based on environments, applications and workloads. This work introduces a new method to select the optimal tasks from several available local tasks in a node. The function with a minimum number of input replicas, loading of an individual node and Maximum average waiting time for the next local node to begin a task is considered to increase the percentage of launching a job's local tasks. Structure of the remaining content is. Section 2 provides related work details. Section 3 gives proposed work explanation. Section 4 includes proposed work performance evaluation. Paper concludes at section 5 with further work.

II. RELATED WORK

Hadoop uses of master-slave architecture and slave nodes sends the heartbeat messages frequently to master node, after the heartbeat message is received from the node, the scheduler schedules the tasks to a node. Hadoop default scheduler takes high response time for small jobs and lacks in sharing the resources fairly among the users. To share the cluster across different organizations, firstly, creates a list of queue with minimum capacity and shares the cluster. To fairly share the clusters among users/applications, Facebook has developed a Hadoop Fair Scheduler (HFS) that creates a set of pools and each pool has a minimum share. To effectively launch the speculative copy of the straggler's tasks, LATE and Delay scheduler increases data locality by compromising the fairness moderately. Authors in [10] considered the deadline constraint of a job. Several authors have proposed several algorithms but still there is a scope to improve optimal task selection from the set of available tasks.

III. PROPOSED SYSTEM

Scheduler assigns a slot to the job depending on the scheduling policy and the availability of the node, and then it identifies and allocates to a node from the unscheduled local task of a job. If the local tasks are not available, scheduler assigns non-local task or prioritizes the jobs present in queue depending on the scheduling strategy. Every block of input is replicated to 3 nodes, because Hadoop default replication factor is configured to 3. Input files are fragmented into input and output blocks of fixed size and stored into Hadoop Distributed File System (HDFS).The number of blocks generally exceeds number of nodes and replicates each block to several nodes, the input data blocks of different jobs are stored in one node. An application which takes different files as input and improves the chances of assigning job's local tasks to a node. To select the maximum number of local tasks, it is necessary to know in advance, remaining time to execute the tasks in a node, allocating the task to a less load node, achieving a high transfer rate and balancing the load of a node. Progress Rate is used to analyze the free slots of a node in advance to allocate the tasks, it considers the replication factor of input block, predicted available slots of a node in advance, present load of a node where the input of a task is stored.
Let UT = { ut1, ut2, ..., uti } is the unscheduled tasks lists and the input is stored in nodei (List of local tasks in Nodei), ALTi is the average length of tasks executing on slots with the available task output Ii.

Revised Manuscript Received on December 22, 2019.

D C Vinutha, Associate Professor, ISE Department, Vidyavardhaka College of Engineering, Mysuru.

G T Raju, Vice-Principal, Professor & Head, Department of Computer Science & Engineering, RNS Institute of Technology, Bengaluru, Karnataka.

RP is input replication factor of UTi and slotik is the available slots in a node, predicted time for the availability of next free slot to assign a task is computed using equation 1.

$$FT(UTi) = \frac{ALTi}{\sum_{k=1}^{Rp} slotik} \quad (1)$$

The load of a node and the request for input / output from the local task in a node is identical to the request for input/output from non-local tasks executing on some nodes by reading the disk content via the network. In order to dissiminate the load equally between nodes, a task is chosen where the input data allocated to a node is selected to store the minimum number of input data from the executing task.

$$DL(Ti) = 1 - \frac{S_R}{S_N} \quad (2)$$

The total slots of a node is SN, and the executing tasks are SR, which reads data from disks where UTi is allocated to the task input. Weightage for replication depends on the replication factor which is computed using equation 3.

$$WR(t) = 1 - \frac{R_P - \min(R_P)}{(\max(R_P) - \min(R_p))} \quad (3)$$

Where the maximum and min imum number of task block replicas is max(Rp) and min(RP). For WR (t) higher replica count values give less value and vice varsa.

The values of WR (T) and load (Ti) ranges between 0 to 1, the value FT(UTi) is normalized between 0 and 1. Finally task selection is computed using equation 4 and 5.

$$\text{TS(T)=Max}\{\alpha * FT(UTI) + \beta * FT(UTi) + \gamma * WR(Ti)\} \quad 1 \le i \le n \quad (4)$$

$$\text{TS(T)=Max}\{\alpha * \frac{ALT_i}{\sum_{k=1}^{Rp} slotik} + \beta * (1 - \frac{SR}{SN} + \gamma * [1 - \frac{R_p - \min(R_p)}{\max(Rp) - \min(Rp)} \quad 1 \le i \le n \quad (5)$$

α, β, γ are ratio variables, n is the quantity of local map tasks to be performed in a node that is further scheduled base d on the scheduling strategy. T(s) is represented as a combination of multiple goals that provide flexibility depending on the requirements to easily include or delete goals. Depending on the specifications, α, β, γ values can be constant and their sum equals to one. For the objective function, larger value has a higher weight.

IV. RESUTLS AND DISCUSSIONS

Hadoop cluster is created to conduct an experiment with 20 nodes and the systems configurations are 8 GB RAM, 2.4 GHz and 500 GB HDD. Experimental review of the proposed method was conducted on Click count application on research and academic web server log file dataset such as NASA and rnsit.ac.in .The local map tasks percentage executed on a node using the proposed method and HFS for different job size is shown in Figure 1.Experimental results show that, relative to Hadoop Fair Scheduler (HFS), the data locality is improved by 20 percent. There is no significant improvement in the data locality using the proposed work for small and larger jobs, as the data blocks are stored in limited number of nodes. Alternatively the experimental result shows an improvement of 20% in the data locality for medium job size (5-50 map tasks).

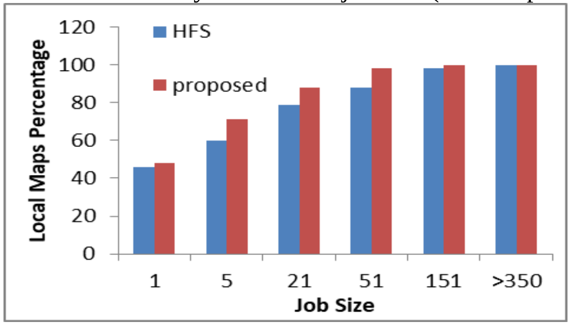

Fig 1: Performance Analysis of number of local tasks launched for different job size

Zaharia et al. [9] presented a delay scheduler to enhance the locality by slightly compromising in allocating the resources. Experiments are also conducted using HFS with delay scheduling and compared with proposed work. Figure 2 shows a significant improvement in allocating the resources fairly among the jobs. The proposed method improves locality by decreasing the relaxation of fairness of delay scheduling method. An improvement of fairness and locality results in improvement in response time and load balancing of a job.

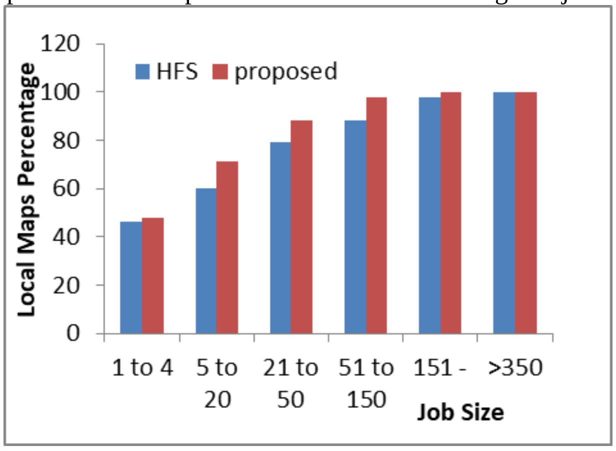

Fig 2: Performance Analysis of fairness using proposed method with Delay Scheduler

V. CONCLUSION

This work suggested a new method for selecting the task from available local tasks lists depending on the input block replica count, predicted available time of node to start the tasks and the current node load. Experimental results revealed that the proposed work improves data locality by 20% compared to HFS and fairness is also improved without having an impact on locality with delay scheduling. Future work may be carried out by considering the parameters (like cluster size) for selection of task.

REFERENCES

[1] V Jeffrey Dean and Sanjay Ghemawat, " MapReduce: Simplified data processing on large clusters", Communications of the ACM, 51(1):107–113, 2008.

[2] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. "Dryad: distributed dataparallel programs from sequential building blocks", ACM SIGOPS Operating Systems Review, 41(3):59–72, 2007.

[3] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox, "Twister: a runtime for iterative MapReduce", In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 810–818, 2010.

[4] Yunhong Gu and Robert L Grossman, " Sector and sphere: the design and implementation of a high performance data cloud", Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367(1897):2429–2445, 2009.

[5] Apache Hadoop. http://hadoop.apache.org/.

[6] Hadoop's Capacity Scheduler: http://hadoop.apache.org/docs/stable/capacity_scheduler.html

[7] Hadoop's Fair Scheduler: http://hadoop.apache.org/docs/stable/fair_scheduler.html

[8] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments", In Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2008.

[9] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", In Proceedings of the 5th European Conference on Computer systems (EuroSys), 2010.

[10] K. Kc and K. Anyanwu, "Scheduling Hadoop Jobs to Meet Deadlines", *in Proc. CloudCom,* 388-392,2010.

[11] B.Thirmala Rao, LSS.Reddy," Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments". *International Journal of Computer Applications*,34 (9):29-33,2011

AUTHORS PROFILE

Mrs.Vinutha D C, Research Scholar, pursuing PhD at R&D Centre, CSE Dept., RNSIT and working as Associate Professor, ISE Dept., VVCE, Mysuru. Her research interests include Big Data Systems, Hadoop, Parallel Processing mechanisms, Scheduling. She has published more than 12 research papers.

Dr. G T Raju has received M.E. (CSE), Degree from Bangalore University in 1995 and Ph.D (CSE) from Visvesvaraya Technological University (VTU), Belagavi, Karnataka in 2008. Currently working as Vice-Principal, Professor & Head in the Department of Computer Science & Engineering , RNS Institute of Technology, Bengaluru, Karnataka – 560 098. He has 25 years of teaching and research experience. His areas of research interests include Web Mining, Semantic Web, Artificial Intelligence, Machine Learning, Knowledge Data Discovery, Internet of Things, Image Processing and Pattern Recognition. He has published 100+ research papers in reputed International Journals and conferences. He has authored 5 technical text books. He has completed two funded projects. 10+ Research Scholars have been awarded Ph. D degree under his supervision.