International Journal of Engineering & Technology, *7 (2.33) (2018) 843-846*

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A hybrid and optimized resource scheduling technique using map reduce for larger instruction sets

Syed Thouheed Ahmed [1] *, Ashwini S [2], Divya C [2], Madhura Shetty [2], Pravina Anderi D [2], Amit Kumar Singh [3]

[1] *Assistant Professor, Department of Computer Science & Engineer, Dr. T Thimmaiah Institute of Technology, VTU, K.G.F*
[2-5] *UG-Scholars, Department of Computer Science & Engineer, Dr. T Thimmaiah Institute of Technology, VTU, K.G.F*
[3] *Assistant System Engineer, TATA Consultancy Services (TCS), Bengaluru*
**Corresponding author E-mail: syed@drttit.edu.in*

Abstract

MapReduce is a structural form to address larger-applications for handling tremendous data generated in parallel. These larger tasks is car-ried out by master and salve node architecture, where the master node judges all the available resources and manages the distributed applica-tions and the slave node is responsible to maintain the resources usability and conveys the information to the master node but the problem encountered is the varying of their resource which should be optimized. In today's business applications resources optimization can either be solved as FIFO queues or using priority scheduler algorithms, thus supports both FIFO and priority algorithm by a concept known as PRISM i.e., Phase and Resources Information Aware Scheduler for map reduce. This incorporates implementation functionali-ties for both FIFO & priority algorithms named as hybrid algorithm where it optimizes the resources based on scenario of evaluation and parameters such as resource time, time to live and resource demand is considered. The importance of phase level scheduler is that it shows the resources usage variability with a particular time of a task. As a result the phase level scheduling algorithm will improve the execution parallelism and resources utilizations such that it ensures the data is not being lost or tampered

Keywords: *Map reduce; Optimization; Scheduling Algorithm; Resource Allocation*

1. Introduction

In recent years MapReduce is become popular for the data inten-sive computations thus is done by breaking the jobs into small maps and reduce their task, that maps are executed across number of machines simultaneously which helps in reducing the running time of the jobs. One of the key features of MapReduce is fault tolerance. To simplify fault tolerance, many implementations of map reduce materialize the entire output of each map and reduce task before it can be consumed. Modified MapReduce architecture has been proposed that allows the data to be pipelined between the operators. This extends the MapReduce programming model be-yond batch processing, and can reduce completion times and im-prove system utilization for batch jobs. The performance of map reduce system is job scheduler. The role of job scheduler is to create map and reduce task for one or more jobs. So that it can minimize job completion time and maximize resource utilization. If the scheduler is running for too many tasks on single machine it leads to heavy resource contention and long completion time. Conversely if few tasks are running on single machine it causes poor utilization of resources.
The main aim of resource aware scheduling for MapReduce is to increase the resource utilization. In existing MapReduce tech-nique, the fixed number of slots will be given to each and every machine to execute their own task. This technique holds good for homogenous workloads, but it fails when individual machine ask for the different resources. Prior to MapReduce that was designed for overall system goals, thus user-specified and resource utiliza-tion was preferred as secondary. so RAS is used i.e., resource aware adaptive scheduler were this RAS adjust the number of slots on each machine dynamically so that they will be increased in resource utilization in one of the cluster which in turn can be done within the given time completion, this technique is performed by using the concept of job slot. The job slot is an execution slot that is bound to particular job to performed particular task within job. In most of MapReduce application, runtime resource consumption varies from phase to phase, by considering the resource demand at the phase level, it is possible for the scheduler to achieve higher degrees of parallelism while avoiding resource contention. For this purpose we have developed a phase level scheduling algorithm where this algorithm aims in achieving high of performance and resource utilization.
The existing system under our research agenda is the trivial meth-od of looking forward for data to be requested as jobs and each job has to be scheduled around the other for completion. The existing system is an unbalanced system for unpattern job collection under an active state of merging under stacking operations. This de-creases the performance and thus increases the overall system delay. The existing system divides the job into threads and then processes it according to the instance availability of the processor and critical section. In this regards the overall system consumes more among of processing time to stack and unstack the job as the parametric conditions of processing and efficiency management.
The proposed system is designed under HADOOP environment and has varies node cluster infrastructure. This improves the over-all system behavior and thus magnifies the ratio of performance and system behavior. The jobs are collected and reduced under JobReduce approach of HADOOP for simplifying the behavior model of load overhead created on the system processor under unbalanced resource sharing.

The proposed system incorporates the overall drawbacks of the existing system and the previous trivial systems used in windows and basic LINUX operating system. The proposed system makes use of Job Reduce operations for instance reduction of jobs into respective tasks and future fetches each job as an independent task for processing. In this approach a dedicated technique of streamlining the overall job is done according to which the processors availability is dynamically shared with respect to the tread of task running in it. This makes the proposed system more efficient and user effective interms of efficiency and performance.

2. Background

2.1. Hadoop & structure review

Hadoop is open source framework based on java programming used for processing and storing of large datasets in a distributed computing hardware. The core of apache Hadoop consists of storage part known as HDFS (Hadoop distributed file system) and the processing part which is MapReduce programming model. Hadoop divide the files into large blocks and distribute them over the cluster of nodes and then transfer the packaged code to process the data in parallel. The main advantage of Hadoop is that is fault tolerant that is, when the data is sent to a node, that data is also replicated to another node in a cluster so in case of any failure there’s another copy available for use. Some of the other advantages of Hadoop is scalability, cost, effective, flexible and fast.
Hadoop consists of computer clusters built from commodity hardware where it has master node (also known as job tracker) as well as slave node (also known as task tasker) master node runs as resource manager the resource manager is the master that attributes all the available cluster resources and help to manage the distributed applications that is running on YARN system. Slave node which runs as local node manager. The node manager is per-machine framework agent which is responsible for containers to monitor their resource usage, and report to resource manager i.e., the master node. Whenever a slave node is being applied to map ( ) method of local data, it launches a JVM and writes output to temporary storage. Master node ensures that only one copy of redundant data is present and runs on resource manager. The slave node then redistributed data based on the output. The slave node then processes each group of output data in parallel and then transmits the message to master node in order to convey the information.

2.2. Map reduce job phases

The Hadoop job schedulers perform task level scheduling where task is considered as the number of simultaneous copies. If task is executed each task consists of multiple phases. Map task is divided into phases. They are map and merge. Map reduces input is given as data blocks which is stored in Hadoop distributed file system (HDFS). Data blocks are stored across multiple slave nodes. In map phase: the mapper takes the data blocks as the input which is stored in HDFS and then applies those data blocks to user defined map function. The map function generates the serialized records and collects it into buffer. If the buffer is full then content is written to local disk. In merge phase: the mapper gets executed in merge phase. The output record is grouped using intermediate keys and stored in multiple files. In shuffle phase the reducer takes the local map task as output file and then place in buffer (buffer can be either the memory or disk depending on size of the content). At same time the reducer launches threads which perform local merge sort it is used to reduce the running time. Sort phases perform final sorting procedure when it collects all the output records and put them in order. Finally in reducer phase all the records and processed according to the user-defined function and final output is written in HDFS. Each and every phase has built-in resource requirements shuffle phase uses significant network I/O resources for collecting output from map task.

2.3. Phase level resource requirement

Apache Hadoop 0.20.2 is used to monitor the execution of phases inside each task. The experiment evaluate phase level resource requirement across hadoop jobs which also includes standard example which is provided by hadoop map reduce distribution given by bench marks that is gridmix2 and PUMA (these are sets of bench marks that are used for comparisons, where the PUMA benchmark are used to replace the existing ones).
The CPU and memory usage are collected by Linux top command whereas I/O usage is collected by MapReduce I/O counters at runtime. Normally CPU usage shows high variance whereas the average CPU usage of mask task remains relatively stable over time. At the same time, the I/O usage increases significantly and each of the task progresses from the map phase to merge phase. Therefore the result of this I/O usage is the result of map phase incrementally reading the input key value pairs from HDFS system. Where the merge phase has high I/O usage and its responsible for grouping all intermediary key value pairs within a short period of time. The runtime resource consumptions of reduce task changes from shuffle phase to reduce phase, where the shuffle phase fetches the intermediate key value pair from the map task, and performs the partial merge on the fetched key value pairs. Hence as the result the shuffle phase consumes both CPU and network I/O resources, once the reduce phase begins the reducers needs to focus on the reduce function in order to produce final output.

3. System architecture

System architecture consists of proposed terminology of scheduling and restructuring of jobs under Hadoop environment. The proposed system, shown in Fig 1 consists of job starter and each stacker is aided with job collection scheme for processing and regulating the jobs on demand a resource monitor and scheduler is designed for monitoring and maintaining the jobs incoming and retrieving the system resources for requested incoming job. Each job is subdivided into multiple tasks and such tasks are assigned to an independent node under Hadoop clustering environment, the system parametric behaviour is critical and thus retrieves the efficient and most significant region of resource utilization. The phase based scheduler is threaded under out system design. The incoming request is processed under a validated system behaviour for resource sharing and thus the request is processed under a validated system behaviour for resource sharing and thus request is diverted towards nodes and slave nodes. This proposed system demonstrates the overall segmented result attributes under load sharing and job management.
The system data flow diagrams are discussed and processed as shown. The process of DFD is clearly mentioned in above Fig 2. This includes the job collection from the triggered and thus store in stacker, each stacker is responsible for data sharing and resource requesting.
An independent trigger is initialed and thus the system behavior under active and hyper active states is analyzed. Primarily, the request is received and stored in stack and resources scheduler approaches for job division and thus achieves tasking of each job segment and this improves the system pattern behavior for analyzing and achieving consistency for a given model. The task is now assigned for independent nodes under HADOOP environment for processing and threads sharing. Each task is executed at the low level of system behavior and thus achieve faster processing rate. The process is divided into slaves and master nodes for processing and thus achieves a faster rate. The process is divided into slaves and master nodes for processing and thus achieves a faster rate of processing under an active and hyper active behavior of system. The overall system is under HADOOP environment and hence the system behavior is constantly monitored for active and safe resource scheduling

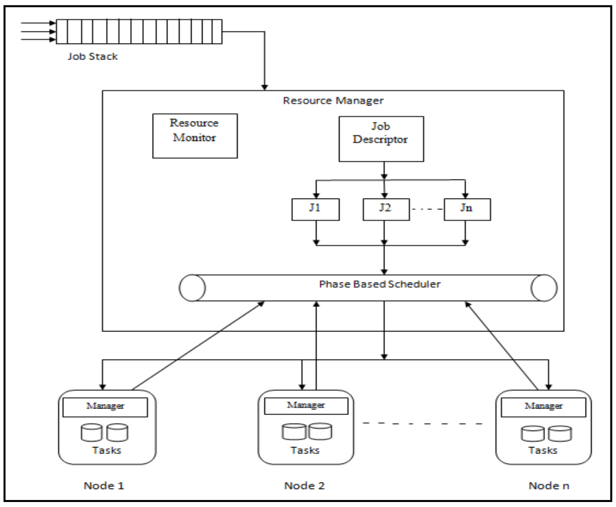


Fig. 1: System Architecture under Hybrid Approach.

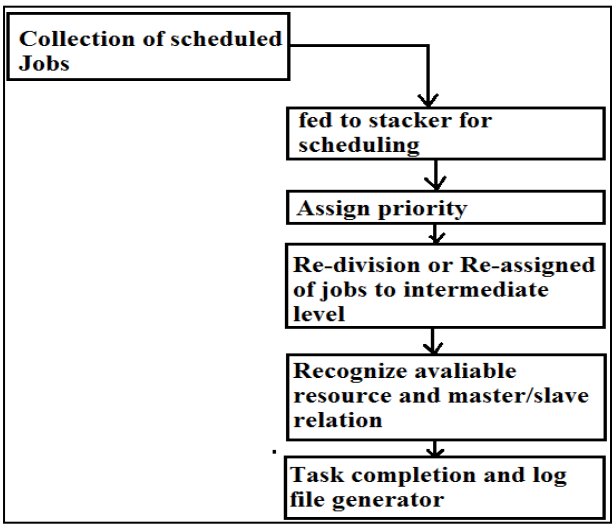

Collection of scheduled Jobs
fed to stacker for scheduling
Assign priority
Re-division or Re-assigned of jobs to intermediate level
Recognize avaliable resource and master/slave relation
Task completion and log file generator

Fig. 2: Block Diagram of Proposed System.

The proposed system is performing under active and hyperactive state of behaviors; this improves the system efficiency and overall system behavior for generating and modulation the behavioral approach for scheduling a job under requested resources. This improves the system behavior for agile transmission of task and control from one master to another master, and from one master to another slave. This is clearly mentioned in above sequence diagram for clear and detailed description.

4. Mathematical modeling

Job collector:
Collect job (J) from all processor P from ‘POST’ (Power On Self-Test). Consider incoming job (J) = $\{J_1, J_2, J_3 \ldots\ldots\ldots\ldots J_n\}$ store each job J_i where i =$\{0\ldots\ldots\ldots\ldots n\}$ such that

$$J_i \in K / K \neq \mathrm{NULL} \tag{1}$$

Where K is clustered job set. K is dedicated index portion of job requester and the equal partition and distributed section of initial processing.
Job scheduler:
Step:1- if K $\not\in$ NULL and K$\in$ $\{J_i\}$&& K=$\{K_1, K_2, K_3 \ldots\ldots\ldots\ldots Km\}$
Each of $K_j \in [{}_0^i\ J$ On summation

$$K=\sum_{i=0}^{n} J_i\ /K_j \text{ --} \tag{2}$$

Step:2- Range analysis and internal prioritization

$$k= R\ [{}_0^m(K_j)_i \tag{3}$$

Where R is cluster constant and k is an interval

Assign Priority:
Under hybrid signification any few process is FIFO and priority Each of K in a range of cluster and its bound is $\sum k \subseteq K$

$$\sum_{i=0}^{n} J_i\ /K \subseteq \sum_{i=0}^{n} \sum_{j=0}^{m} R\ [{}_0^m(K_j)^i \tag{4}$$

Consider P is priority such that

$$P= P\ (\sum_{i=0}^{n} \sum_{j=0}^{m}\ R\ [{}_0^m(K_j)^i)$$

Such that, a differential function is appended to retrieve the below break-down for Reducing function.

$$P=\frac{d(P.Ji)}{dK}\ \&\&\ P=P.\frac{d(J_i)}{dK}$$

The above shown equations are predicted reduce function parameter for single job scenario. The major observation of any job scheduled in our system is based on hybrid approach for sharing processes into smaller partitions for execution. Thus on resembling, the summarized partition for given universal tasks are as shown in Eq. (6).

$$P=P_k.\sum_{i=0}^{n} \frac{d(Ji)}{dk} \tag{5}$$

$$P_i = \sum_{i=0}^{K_i} (P_{k_i}[\ \sum_{i=0}^{n} \frac{d(Ji)}{dk}]) \tag{6}$$

$$\therefore \text{ In general } P=\sum_{i=0}^{K_i} (p_i) \tag{7}$$

Where $p_i=\{(k1)_{p,}(k2)_p,$----------------$(kn)_p\}$ From the above equations, the formulation for priorities is performed to schedule the processes based P
Resource Allocator:
Based on the priorities formulated by Eq. 6, a descriptive allocation for resources is carried in this session, the allocation is shown in Eq. 8 Consider incoming R=$\{R_1, R_2, R_3$-------------------$R_z\}$

$$R=\frac{\partial(J_i)}{\partial P}\ ]_0^n \tag{8}$$

5. Results and discussions

The hybrid scheduler is proposed in this paper and has successfully achieved the architecture based implementation and mathematical proof of concept in session 3 and 4 respectively. In this session, a brief description of Software based framework design is demonstrated.

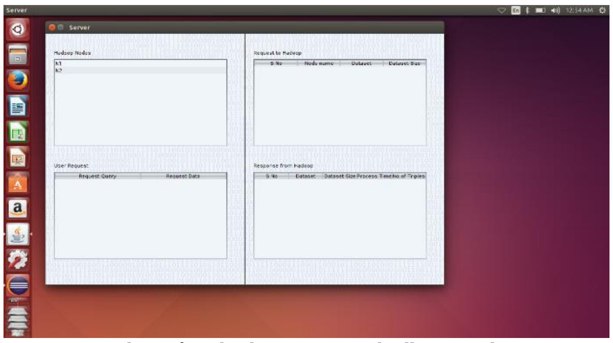
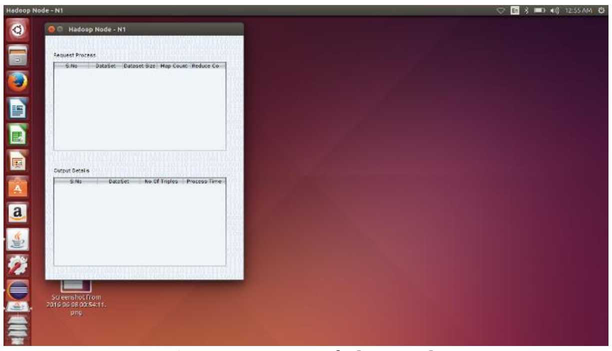


Fig. 3: Screenshot of Hybrid Resource Schudler Based on Optimised Resource Selection, the Picture Demostrates the Activation of Server (Master).


Hadoop Node - N1

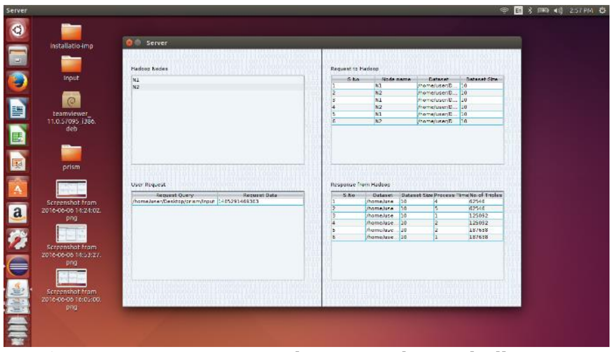
Request Process

S.No	DataSet	Dataset Size	Map Count	Reduce Co...

Output Details

S.No	DataSet	No Of Triples	Process Time

Fig. 4: Activation of Slave Node.


Server

Hadoop Nodes

N1
N2

Request to Hadoop

S.No	Node name	Dataset	Dataset Size
1	N1	/home/user/D...	10
2	N2	/home/user/D...	10
3	N1	/home/user/D...	10
4	N2	/home/user/D...	10
5	N1	/home/user/D...	10
6	N2	/home/user/D	10

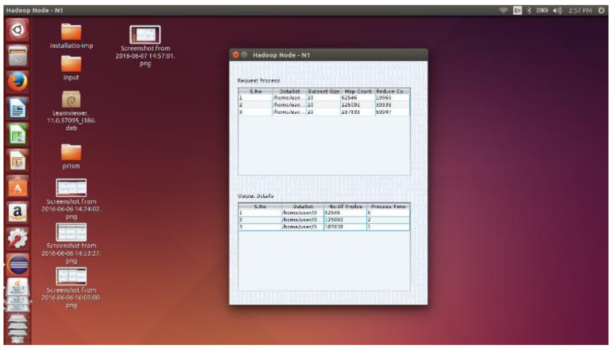
User Request

Request Query	Request Date
/home/user/Desktop/prism/Input	1465291469363

Response from Hadoop

S.No	Dataset	Dataset Size	Process Time	No.of Triples
1	/home/use...	10	4	62546
2	/home/use...	10	5	62546
3	/home/use...	10	1	125092
4	/home/use...	10	2	125092
5	/home/use...	10	2	187638
6	/home/use...	10	1	187638

Fig. 5: Server or Master Based Resource Share and Allocation.


Hadoop Node - N1

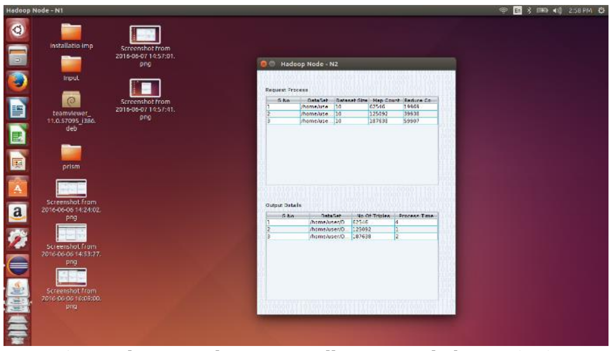
Request Process

S.No	DataSet	Dataset Size	Map Count	Reduce Co...
1	/home/use...	10	62546	19969
2	/home/use...	10	125092	39938
3	/home/use...	10	187638	59907

Output Details

S.No	DataSet	No.Of Triples	Process Time
1	/home/user/O	62546	5
2	/home/user/O	125092	2
3	/home/user/O	187638	1

Fig. 6: Slave Based Resource Allocation and Sharing (N1).


Hadoop Node - N2

Request Process

S.No	DataSet	Dataset Size	Map Count	Reduce Co
1	/home/use	10	62546	19969
2	/home/use...	10	125092	39938
3	/home/use...	10	187638	59907

Output Details

S.No	DataSet	No Of Triples	Process Time
1	/home/user/O	62546	4
2	/home/user/O...	125092	1
3	/home/user/O...	187638	2

Fig. 7: Slave Based Resource Allocation and Sharing (N2).

The above figures demonstrates the master slave based approach for re-allocation of resources in hybrid terminology which incorporates, priority scheduling and FIFO based on job scenario and incoming priority for execution. The master shares the incoming load on N1 and N2 based on preset priorities while execution in Hadoop.

6. Conclusion

The proposed system is designed and evaluated for job scheduling and resources managing application under HADOOP cluster environment, the system is aided with master slave configuration for collecting a request and delivering the output of allocation according to requested achieved. This project is on 7 node cluster infrastructure and hence it retrieves the system performance. The job is rescheduled according to the allocated master slave behavior to reduce the system overload and thus the efficiency is improved. According to the proposed system the jobs are split into task and task into threads on demand is successfully achieved under our simulated protocol of master slave architecture. The system generates positive results for the overall job scheduling including a live operation assignment.

References

[1] GridMix benchmark for Hadoop clusters [Online]. Available: http://hadoop.apache.org/docs/mapreduce/current/gridmix. Html, 2018.

[2] Qi Zhang, Mohammed Faten Zhani, et.al, "PRISM: Fine Grained Resource-Aware Scheduling for MapReduce" IEEE Transitions on Cloud Computing, Vol 3, No.2, June 2015. Pp 182-194.

[3] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computational models and the heterogeneity challenge," J. Internet Serv. Appl., vol. 3, no. 1, pp. 1–10, 2012.

[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[5] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguad_e, "Resource-aware adaptive scheduling for MapReduce clusters," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 187–207.

[6] A. Verma, L. Cherkasova, and R. Campbell, "Resource provisioning framework for MapReduce jobs with performance goals," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 165–186.

[7] Syed Thouheed Ahmed, Divya P Bhadrapur, Bibhu Prasad Mohanty, Lakshmi Bai G, Thanuja K, "Tweeter Data Analysis Using Real time optimized sentimental machine learning algorithm using Hadoop" in International Journal of Pure and Applied Mathematics, Vol 118, No. 19, 2018, pp-2397-2406.

[8] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, vol. 8, pp. 29–42.

[9] Syed Thouheed Ahmed "A study on multi objective clustering technique for medical datasets" in Proc. IEEE International Conference of Intelligent Computing and Control Systems, 2017, pp 174-177.