International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878, Impact Factor (2014): 3.05

Implementation of Improving MapReduce Performance Using Dynamic Slot Allocation Strategy

Pranoti K. Bone[1], A. M. Wade[2]

[1, 2]Department of Computer Engineering, Smt. Kashibai Navale College of Engineering, Vadgaon , Pune, Maharashtra, India

Abstract: ***In many data centers and clusters data processing is a very essential task. For addressing this need, recently many researchers have been working. MapReduce has become one of the well liked techniques for high performance computing tasks. Many large scale clusters of hundreds of machines utilize an open source implementation of MapReduce technique known as Hadoop. However the traditional mapreduce do suffer from sever underutilization of map and reduce slots and straggler machines, i.e. the machines taking very long time to finish their tasks. We propose a dynamic slot allocation strategy which will decrease the execution time of the traditional mapreduce system. The unoptimized map and reduce slots are allocated to the tasks to resolve the underutilization of the slots. We proposed slot pre scheduling technique to improve data locality with no impact on fairness.Smart Executive Performance Balancingcan balance the performance tradeoff between a single job and a batch of jobs dynamically. By building these techniques together we improved the performance of the mapreduce and handled the workload substantially, also the execution time has been decreased.***

Keywords: MapReduce, Big Data, Slot Allocation, Hadoop Distributed File System

1. Introduction

The increasing use of internet leads to handle lots of data by internet service providers. MapReduce is one of the goodsolutions for implementing large scale distributed data application. Mapreduce framework is used for processing terabytes of data. Mapreduce help to build scalable and fault tolerant application.

A. Hadoop

Hadoop[1] is java based framework that allows to process large data sets in distributed environment as shown in figure 1. Hadoop has been used by many large scale companies like Amazon, Facebook, and Yahoo. Hadoop consist of two important concepts:Hadoop Distributed File System (HDFS) and Hadoop MapReduce.

B. Hadoop Distributed File System

The file system that is oriented on blocks is HDFS. Every single file is divided into 64MB size of blocks. There are clustered machines having a huge data storage capacity, where these blocks are kept. Every independent machine in the cluster is known as a DataNode.

As discussed the files are divided into number of blocks, and it is not necessary that all the blocks are stored on a single machine. The target DataNode selection is done randomly based on block-by-block approach. While the user needs to access a file, there is multiple numbers of machines who cooperatively provide the blocks making a single file. But what if a machine in the cluster fails? This issue is handled by Hadoop Distributed File System. The HDFS solves the problem by making replicas of each block on some other machines in the cluster, by default on three machines. In an HDFS cluster there is always a single node known as NameNode having the ability to manage the file system namespace and synchronizes the clients accessing the files. This situation is depicted in figure. The DataNodes stored blocks of files as data. Then it is the responsibility of the NameNode to map the data blocks to the independent DataNodes. NameNode also manages many file system operations like opening, closing, renaming files, etc. The situation where the NameNode might fail is handled by keeping multiple copies of data on several machines called as secondary NameNode[2].

C. Hadoop MapReduce

Controlling and managing of large scale web search applications is a crucial task. There was an attempt made by Google for handling these applications, and it is called as MapReduce. For effectual programming skills for developing data mining, machine learning and search applications in data centers, MapReduce is an bestapproach[3].

D. Challenges of Traditional system

Even if there are several researches going on optimizingthe MapReduce and Hadoop techniques, there are some keychallenges related to the issue of improving theperformanceof the Hadoop cluster. Initially, the basic compute units areconfigured by the administrator in prior to any other operation.The basic units are nothing but the computer resources whichare mapped and reduced in several slots. There are twounique constraints of the MapReduce job execution: First, it is assumed that the map slots are issued to map task and thereduce slots are issued to reduce tasks. Second, the assumptionof general execution where the reduce tasks are executed afterthe map tasks are executed. Due to these constraints, thereare two observations: Firstly, the performance and systemutilization for a MapReduce workload varies as the slotconfigurations differs. Second, even if there is optimal mapor reduce settings, there can be several idle slots indirectlyaffecting the system performance. Sometimes the disputes forprocessor, memory, network bandwidth and other resources,may consume unnecessary time causing delay of the entiremap and reduce tasks. For efficient MapReduce workloads,slot utilization performance

is very important and it sometimesdepends on data locality maximization.

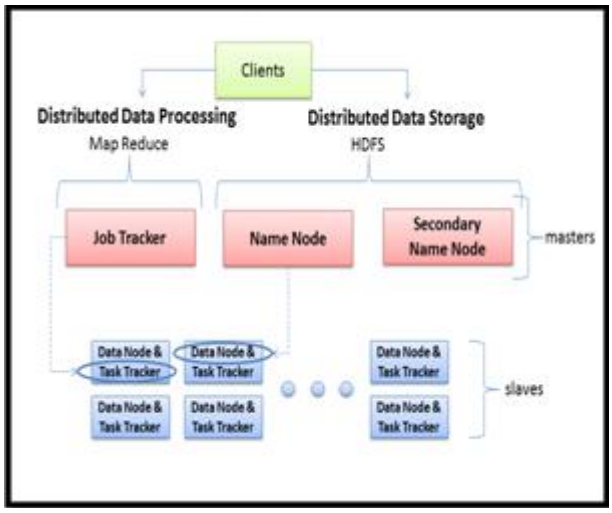


Figure 1: Hadoop Framework

2. Related Work

Performance optimization for MapReduce jobs is a very attention captivating topic for researchers. We survey some of the relating topic to our proposed work.

A. Scheduling and Resource Allocation Optimization

Many researchers have worked on optimization work for MapReduce jobs, and paid attention on computation scheduling and resource allocation topics of the same. Also many authors considered job ordering optimization for MapReduce workloads [4], [5], [6], [7], [8]. The modeling of the MapReduce as a two-stage hybrid flow is described in [9]. This hybrid flow shop has multiprocessor tasks, where job submission orders affect the results of cluster utilization and system performance. The execution time for mapping and reducing the tasks for each job must be known earlier, but this phenomenon is not implemented in the applications. Also this method has not considered for the dependent jobs and suitable only for the independent jobs. Example of such method is MapReduce workflow. Compared to this phenomenon, our proposed DHSA is suitable for all types of jobs. Starfish [10] framework can modify the hadoop configuration automatically for the MapReduce jobs. By using sampling technique and cost based model we can maximize the utilization of hadoop cluster. But still we can improve the performance of this technique by maximizing the utilization of map and by reducing slots. Polo et al. [11] proposed a technique for MapReduce multi job workloads based on resource aware scheduling technique. This technique focus on improving resource utilization by expanding the abstraction of existing task slot to job slot.YARN[12] solve the inefficiency problem of the Hadoop MRv1 in the perspective of resource management. Instead of using slot, it manages resources into containers. The Map and Reduce operation are performed on any container.

B. Speculative Execution Optimization

In MapReduce we need task scheduling strategy for dealing with problems such as straggler problem for a single job, which include LATE [13], BASE [14], Mantri [15],MCP [16].Speculative execution is such an important task scheduling strategy. The speculative execution algorithm speculates the task by prioritizing and pays attention on heterogeneous environments. To run, selecting the fast nodes and the speculative tasks are covered over, this speculative execution algorithm is a longest approximate time to end (LATE) [13], and the prioritizing of task is required for speculation. Guo et al. [14] proposes a Benefit Aware Speculative Execution (BASE) algorithm which evaluate the potential benefit of the speculative tasks and the unnecessary runs are eliminated. This BASE algorithm of the evaluating and elimination can improve the performance for LATE. The speculative execution strategy magnifies its focus mainly on saving cluster computing resource, is provided by Mantri [15].Maximum Cost Performance (MCP) is a new speculative execution algorithm proposed by the Chen et al. [16] proposed for fixing the problem that was affecting the performance of the prior speculative execution strategies. We proposed speculative Execution Optimization strategy that balances the tradeoffs between a single job and a group of jobs.

C. Data Locality Optimization

Many previous work on improving the performance and efficiency of the utilization of cluster have proven to be critical task (for example in [17]-[18]). There are two data locality approaches for MapReduce, reduce-side and map-side. In the map-side data locality approach, the map task computation is moved near to the input data (for example in [17]-[19]).

In [17], the concept of Delay Scheduler is used to refine the data locality task. The jobs of MapReduce are classified into three kinds, map-input heavy, map-and-reduce-input heavy and the reduce-input heavy by Purlieus [20]. They have proposed the work for improving the runtime performance. The tradeoff between fairness and data locality is adjusted by an algorithm provided by Guo et al. [21], [19]. This work is supported by a mathematical model given by the author. The author in [20], [22], [23] have worked on reduce-side data locality optimization task by proposing greedy algorithms.

3. Proposed Work

A. Problem Definition

To maximize the slot utilization for MapReduce and balancethe performance tradeoff between a single job and a batch of jobs with fair scheduling and improving the performance of MapReduce cluster in Hadoop.

B. Goals and Objective

The objective is to utilize the slots in MapReduce cluster. The slot utilization remains a challenging task due to fairness and resource requirements. It is fair when all pools have been allocated with the same amount of resources. The resources requirements between the map slot and reduce slot are generally different. This is because the map task and reduce task are often exhibit completely different execution patterns.

C. System Overview

Proposed approach consist of three optimization techniques, namely, Dynamic Hadoop Slot Allocation, Speculative Execution Performance Balancing and Slot Pre Scheduling

as shown in figure 2. After executing these 3 techniques, results into slot utilization and utilization efficiency optimization. It Also improves the data locality and load balancing.

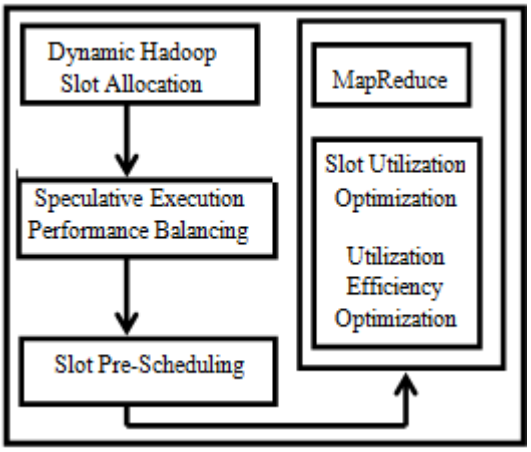


Figure 2: Overview of the Proposed System

Dynamic Hadoop Slot Allocation

It attempt to maximize slot utilization while maintainingthe fairness, when there are pending tasks (e.g., map tasks orreduce tasks). We break implicit assumption of MapReducethat the map tasks can only run on map slots & reduce taskscan only run on reduce slots. In our proposed system wemodify it that map and reduce tasks can berun on either mapor reduce slots.

There are 4 cases,

Consider,

NM = Total number of Map tasks

NR = Total number of Reduce tasks

SM = Total number of map slots

SR = Total number of reduce slots

Case 1: NM ≤ SMandNR ≤ SR

The map tasks which are running on map slots and reduce tasks are run on reduce slots, There is no borrowing of map and reduce slots.

Case 2: NM>SMandNR<SR

We satisfy reduce tasks for reduce slots first and then use those idle reduce slots for running map tasks.

Case 3: NM<SMandNR>SR

We can schedule those unused map slots for running reducetasks.

Case 4: NM>SMandNR>SR

The system should be in completely busy state.

1) Speculative Execution Performance Balancing:

It identifies the slot resource in-efficiency problem for aHadoop cluster, caused by speculative tasks. It works on top ofthe Hadoop speculative scheduler to balance the performancetradeoff between a single job and a batch of jobs. SlotPre-Scheduling improves the slot utilization efficiency andperformance by improving the data locality for map taskswhile keeping the fairness.Speculative tasks are competing for certain resources includingnetwork, map slots and reduce tasks. For maximizing theperformance we should complete the pending tasks first beforeconsidering the speculative tasks. When node is having idlemap slot, then we should consider pending map task firstlyand then we consider speculative map task. For an idle mapslot, we first check jobs J1, J2 ... Jn for map task.For every job we check the total number of pending map andreduce tasks by considering all jobs from Ji and Jj. Where,i=1,2,3,4....

j= i + maxNumOfJobsCheckedForPendingTasks − 1.

We checked each jobJi by considering 3 conditions: 1) Totalpending map tasks are greater than zero. 2) No failed pendingmap tasks and map tasksfor job Ji. 3) Total pending reducetasks is greater than zero.

2) Slot Pre-Scheduling

It improves the slot utilization efficiency and performance by improving the data locality for map tasks while keeping the fairness.

Step 1: Compute load factor mapSlotsLoadFactor = Pending map tasks +running map tasksfrom all jobs divided by the cluster map slot capacity.

Step 2: Compute current maximum number of usable map slots = number ofmap slots in a tasktracker * minmapSlotsLoadFactor, 1.

Step 3: Compute current allowable idle map (or reduce) slots for a tasktracker= maximum number of usable map slots - current number ofused map (or reduce) slots

D. Mathematical Model

The mathematical terminology of proposed system is explained below:

Let S be the proposed system

S = {I,O,F,Fs,F*l*,ϕ}

Identify the inputs I.

I = {T1, M, T2, R,U,E}

Where,

T1 = Pending Map Tasks.

M = Idle Map Slots.

T2 = Pending Reduce Tasks.

R = Idle Reduce Slots.

U = Utilized Slots.

E = Empty Slots.

Identify set of Function. Let F be the set of Functions.

F = {F1,F2,F3}

Where,

F1 = Verify Information.

F2 = Dynamic Slot Allocation.

F3 = Balance the Performance of Job.

Identify the Outputs. Let O be the set of outputs.

O = {O1,O2}

Where,

O1 = Slots Allocated Successfully.

O2 = Successfully Balance the Performance of Job.

Final State:

FS = Increase the Performance of Mapper & Reducer

Failure case:

F*l*= Errors in measuring the input parameters

Constraints:

Let ϕ be the constraints ϕ =C*l*

Where,

C*l* = Accuracy in measuring the input parameters.

E. Experiments and Result

In This section we have shown the working of the proposed system. The figure 3 shows the total required time for the completion of task for proposed system in contrast with the existing system which is Maximum Cost Performance. In figure 3 it also shows the specific time required for all the three techniques Dynamic Hadoop Slot Allocation (DHSA),

Speculative Execution Performance Balancing (SEPB) and Slot Pre-Scheduling.

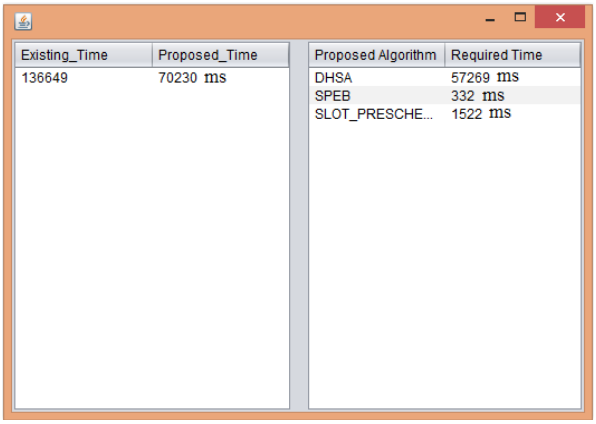
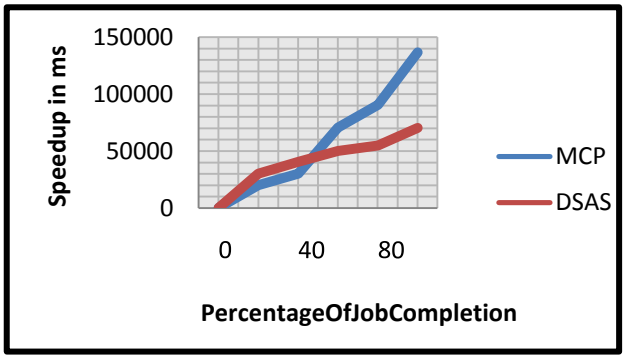


Figure 3: Required time for the proposed system

The graph 1 shows the time required to complete the tasks in MCP is higher as compared to DSAS. The performance of the MCP degrades as the time speeds up.



Graph 1: Performance improvement of the system

4. Conclusion

The aim of the proposed system is to improve the performance of MapReduce workloads. It considered three techniques: Dynamic Hadoop Slot Allocation, Speculative Execution Performance Balancing, and Slot Pre-Scheduling. Dynamic Hadoop Slot Allocation uses allocation of map to maximize the slot utilization and it reduces the task dynamically. It does not require any prior information or any assumption and it can be run on any kind of MapReduce jobs. Speculative Execution Performance Balancing identifies the slot inefficiency problem. It manages the balance between single and batch of jobs dynamically. Slot Pre-Scheduling are used to enhance the efficiency of slot utilization by maximizingdata locality. We can enhance the utilization by adding above concept in traditional system. In future we plan to implement above mentioned concept in cloud environment.

5. Acknowledgement

We thank all the anonymous reviewers and editors for their valuable comments and suggestions to improve the quality of this manuscript.

References

[1] Apache Hadoop. http://hadoop.apache.org.
[2] Hadoop Distributed File System, http://hadoop.apache.org/hdfs
[3] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. OSDI '04, pages 137150, 2004
[4] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and flow-shops. In SPAA'11, pp. 289-298, 2011.
[5] A. Verma, L. Cherkasova, R.H. Campbell, Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan, IEEE Transaction on dependency and secure computing, 2013.
[6] A. Verma, L. Cherkasova, R. Campbell. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance. In IEEE MASCOTS, pp. 11-18, 2012.
[7] S.J. Tang, B.S. Lee, and B.S. He. MROrder: Flexible Job Ordering Optimization for Online MapReduce Workloads. In Euro-Par'13, pp. 291-304, 2013.
[8] S.J. Tang, B.S. Lee, R. Fan and B.S. He. Dynamic Job Ordering and Slot Configurations for MapReduce Workloads, CORR (Technical Report), 2013.
[9] C. Oguz, M.F. Ercan, Scheduling multiprocessor tasks in a twostage flow-shop environment. Proceedings of the 21st international conference on Computers and industrial engineering, pp. 269-272, 1997.
[10] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A Self-tuning System for Big Data Analytics. In CIDR11, pp. 261C272, 2011.
[11] J. Polo, C. Castillo, D. Carrera, et al. Resource aware Adaptive Scheduling for MapReduce Clusters. In Middleware'11, pp. 187- 207, 2011.
[12] Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop yarn/hadoopyarn- site/YARN.html.
[13] M. Zaharia, A. Konwinski , A.D. Joseph , R. Katz , I. Stoica,Improving MapReduce performance in heterogeneous environments.In OSDI'08, pp.29-42, 2008.
[14] Z.H. Guo, G. Fox, M. Zhou, Y. Ruan.Improving ResourceUtilization in MapReduce. In IEEE Cluster12. pp. 402-410,2012.
[15] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y.Lu, B. Saha, and E. Harris, Reining in the outliers in map-reduceclusters using mantri, in OSDI10, pp. 1-16, 2010.
[16] Q. Chen, C. Liu, Z. Xiao, Improving MapReduce PerformanceUsing Smart Speculative Execution Strategy. IEEE Transactions on Computer, 2013.
[17] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I.Stoica, Delay scheduling: A simple technique for achievinglocality and fairness in cluster scheduling. In EuroSys10, pp.265-278, 2010.
[18] J. Tan, S. C. Meng, X. Q. Meng, L. Zhang. Improving ReduceTaskdata locality for sequential MapReduce jobs. In IEEEInfocom13, pp. 1627-1635, 2013.
[19] Z. H. Guo, G. Fox, and M. Zhou. Investigation of Data Localityin MapReduce. In IEEE/ACM CCGrid12, pp, 419-426, 2012.

[20] B. Palanisamy, A. Singh, L. Liu and B. Jain, Purlieus: LocalityawareResource Allocation for MapReduce in a Cloud, InSC11, pp. 1-11, 2011.
[21] Z. H. Guo, G. Fox, and M. Zhou.Investigation of data locality and fairness in MapReduce. In MapReduce12, pp, 25-32, 2012.
[22] M. Hammoud and M. F. Sakr. Locality-Aware Reduce TaskScheduling for MapReduce. In IEEE CLOUDCOM11. pp. 570-576, 2011.
[23] M. Hammoud, M. S. Rehman, M. F. Sakr. Center-of-GravityReduce Task Scheduling to Lower MapReduce Network Traffic.In IEEE CLOUD12, pp. 49-58, 2012.