A MODERN APPROACH ON MAP REDUCE CLUSTER USING DYNAMIC SLOT ALLOCATION OPTIMIZATION FRAMEWORK

B.Kayalvizhi*

S.Karpagaselvi*

Abstract—

Emerging technology and effective mechanism oriented approach makes Map Reduce a popular as well as important computing paradigm which dealing with big data process in cloud computing. The mechanism of un-optimized resource allocation is the severe drawback for slot based Map Reduce System. To overcome the failures of the optimize resource allocation we identify three aspect in this process in order to improve the resource allocation in an effective manner. The initial one is pre-configuring the map slots by reduce the slots which are under-utilized and fungible. The key behind these methods is it makes the map slots fully utilized which reduces the empty slots and vice-versa. We deal this process with a proposed advanced technique known as Dynamic Hadoop Slot Allocation. It can be achieved by implementing the slot based model in this proposed mechanism. It makes every slot reallocated either on map or minimizing the task based on the requirements. The next things are handling the stagger problem. It can be involved in making every single job highly impact without affecting the efficiency of the clusters. Our proposed Speculative Execution Performance Balancing is developed with the motto of building a tradeoff between the single jobs with the batch of jobs. Th third one in our proposed approach is dealing with delay scheduling that makes the data locality in a better way without raising the cost. Based on the above mentioned methodology a combined approach enables a step-by-step slot allocation by the proven name of DynamicMR that shows the massive impact in the working of Map Reduce subsequently dealing

* RVS Educational Trust Group of Institutions, Dindigul-624001

with their workloads. The performance of DynamicMR shows high impact on Hadoop MRv1, by examining the outputs achieving the accuracy of 46% ~ 115% gradually on single jobs as well as 49% ~ 112% effectiveness on multiple jobs. To justify the result in technology aspect our proposed work is compared with YARN experiments that's shows the DynamicMR performance 2% ~9% on multiple jobs as the betterment of ratio control mechanism based on the current map reduce approach.

Keywords— Slot Allocation, Slot Pre-Scheduling, Map Reduce, Delay Scheduler, Hadoop Fair Scheduler, DynamicMR.

1 INTRODUCTION

Map Reduce raised its popularity in the field of big data processing based on clustering and data centers (1). Map Reduce is a coding pattern that abstracts much of the tricky bits of scalable computations. HFDS can be part of a Hadoop cluster or can be a stand-alone general purpose distributed file system. An HFDS cluster primarily consists of

- Name Node that manages file system metadata
- Data Node that stores actual data
- Stores very large files in blocks across machines in a large

Cluster. Reliability and fault tolerance ensured by replicating data across multiple hosts. Has data awareness between nodes and designed to be deployed on low-cost hardware Hadoop can work directly with any distributed file system which can be mounted by the underlying OS. However, doing this means a loss of locality as Hadoop needs to know which servers are closest to the data Hadoop-specific file systems like HFDS are developed for locality, speed, fault tolerance, integration with Hadoop, and reliability. Code usually written in Java- though it can be written in other languages with the Hadoop Streaming API. Two fundamental pieces:

Map step: Master node takes large problem input and slices it into smaller sub problems; distributes these to worker nodes. Worker node may do this again; leads to a multi-level tree structure. Worker processes smaller problem and hands back to master

Reduce step: Master node takes the answers to the sub problems and combines them in a predefined way to get the output/answer to original problem.

Being an open source Hadoop can be implemented with Map Reduce that can be applied

on large data base such as face book, government organization.etc. That makes the process more effective on Map Reduce workloads. In recent studies the optimizing Map Reduce/Hadoop facing the problems on utilization and performance that affects cluster performance. During the starting state the resource can be segmented into map and slots which can be configured by the administrator (2). A Map Reduce process involves two sections one is allocating the map slots which can be used to reduce the tasks. The next section is execution that can be executed before the reducing the task. Based on the above sections we have two outcomes first one is varying performance and utilization Map Reduce workloads on changing the slot configuration. The next case under the optimal map/reduce slot configuration in this it strategically affects the system utilization and performance on keeping the slots unoccupied.

In certain case the run-time contention on processor, memory, network bandwidth and other resources is necessary on that case it may be straggled map or reduce tasks that affects the whole job (3). Another important thing on dealing with Map Reduce is data locality maximization because it improves the concert and operation on Map Reduce workloads in a better way. That is the concept of Hadoop cluster among multiple users (4).

In above we have discussed about the problems we are facing currently, based on that challenges we proposed a DynamicMR which can be developed in the main intention of improving the performance of dynamic slot allocation framework on Map Reduce cluster with slot utilization. The major thing in this DynamicMR is Hadoop Fair Scheduler (HFS). The recent issue is considering the Map Reduce jobs under HFS with FIFO scheduler HFS shows worst result. So only the proposed DynamicMR has three optimization techniques, which are Dynamic Hadoop Slot Allocation (DHSA), Speculative Execution Performance Balancing (SEPB) and Slot Pre-Scheduling from different key aspects:

Dynamic Hadoop Slot Allocation (DHSA): On discussing about DHSA it more contrast to YARN that act as containers on managing map reduce and task. In this slots are pre-configured by numbering it and reduce the slots which are idle. Similarly the tasks are reduced by using the unallocated map slots. Another advantage is it controls the ratio of running map and reduces tasks during runtime because this mechanism is better than the YARN.

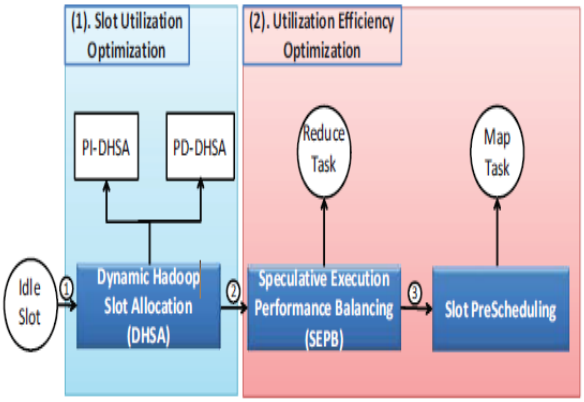
2 SPECULATIVE EXECUTION PERFORMANCE BALANCING (SEPB):

2.1 Introduction

SEPB is based on Speculative execution which solves the problem of slow-running tasks on single jobs and enables backup task on another machine. So we proposed a dynamic slot allocation technique by name Speculative Execution Performance Balancing (SEPB) which balances the single job's execution time and a batch of jobs execution time based on the speculative tasks which are allocated on the slots.

2.2 Slot Pre-Scheduling:

On our proposed system the slot Pre-Scheduling maximize the performance of data locality without affecting its impact on fairness. This effectiveness can be achieved by balancing the nodes between the slave nodes. In runtime due to some constrains the idle slots are not allocated. By doing these pre-allocating slots maximize the data locality on their nodes. ……



2.3 Existing work:

There are various research works states the performance of MapReduce jobs and works which are in consideration are Scheduling and Resource Allocation Optimization, Speculative Execution Optimization, Data Locality Optimization and MapReduce Optimization on Cloud Computing. On dealing with Scheduling and Resource Allocation Optimization the major work is job ordering optimization based on MapReduce workloads. Here they discussed two-stage hybrid flow shop with multiprocessor tasks (5).The major issue is in real time application that does not avail the execution time for map and reduce tasks for each job that should be known in advance.

According to Guo et al.(6) propose a resource stealing method to enable running tasks to steal resources reserved for idle slots by adopting multithreading technique for running tasks on multiple CPU cores but it cannot work for the utilization improvement of those purely idle slave nodes without any running tasks. YARN [7] is a new version of Hadoop with totally different architecture. In contrast to our DynamicMR, it overcomes the inefficiency problem of the Hadoop MRv1 from the resource management perspective. There is no more concept of slot.

Another important task scheduling strategy in MapReduce is Speculative execution that deals with major problem known as straggler on single job. LATE BASE [7]. Longest Approximate Time to End (LATE) [3] is a speculative execution algorithm that focuses on heterogeneous environments by prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks.

Data locality optimization is nothing but improve the data locality by delaying the scheduling of map tasks whose data locality cannot be satisfied for a short period of time, by the process called as Delay Scheduler(3). They are unable to solve the problem of reduce task assignment problem as a stochastic optimization problem theoretically (8) (9). On considering the Cloud Computing MapReduce Optimization deals with deadline and budget (10) (11).The drawbacks in task scheduling and manage the resource allocation for MapReduce workloads put route to the proposed work.

3 OVERVIEW OF DYNAMICMR

To build an effective Map Reduce cluster by optimizing the slot utilization, we follow two methodologies first segment the slots into two type's busy slots and idle slots. The optimization approach involves in reducing the slots according to the total number of map. The architecture of propose DynamicMR is given below.

Fig 1: Architecture of DynamicMR Framework.

As we discussed earlier our proposed architecture of DynamicMR consist of Dynamic Hadoop Slot Allocation (DHSA), Speculative Execution Performance Balancing (SEPB), and Slot Pre-Scheduling which are clearly explained in the fig1.In this proposed system each factor has a unique thing that increase the overall performance. DHSA which involves in maximize slot utilization in concentrating with its fairness. In the same manner the SEPB handles the drawbacks of slot resource in Hadoop cluster that happens due to speculative tasks. So our proposed Hadoop speculative scheduler balances the single job and a batch of jobs in an effective manner. The third one is Slot Pre-Scheduling which concentrated in capitalize on slot utilization efficiency and performance by making fair on data locality.

4 PROPOSED METHODOLOGY:

The proposed model is designed with drawbacks of Map Reduce such as under-utilization of the respective slots on map numbers and reduce task according to the time variation. We consider our dynamic slot allocation policy with idle slots on different period of time. Here we handle the Fairness of Hadoop Fair Scheduler (HFS) which is allocated with the same amount of resources (12).We concentrate on reduce task which consumes more resources such as memory and network bandwidth. This can be achieved by allowing reduce tasks to use map slots simply. The proposed Dynamic Hadoop Slot Allocation (DHSA) is designed with two methodologies such as pool independent DHSA (PI-DHSA) and pool-dependent DHSA (PD-DHSA).

Pool-Independent DHSA (PI-DHSA):

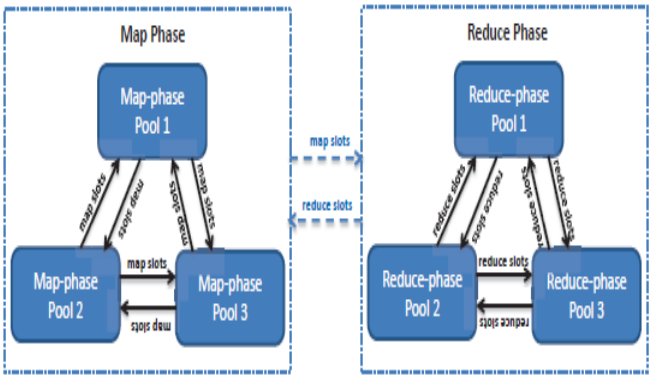


Fig2: System mechanism of Pool-Independent DHSA (PI-DHSA)

We can say that Pool-Independent DHSA (PI-DHSA) is extension of HFS by overcoming its drawbacks such as allocating slots from the cluster global level. Fig 2 represents the slot allocation flow for PI-DHSA. We separate the slot allocation as Intra-phase dynamic slot allocation and Inter-phase dynamic slot allocation. In Intra-phase dynamic slot allocation the pool is segmented into map-phase pool and reduce-phase pool and the slot is shared to each pool. By this way the exceeded pool dynamically borrow unused slots from other pools. In considering the Inter-phase dynamic slot allocation in the same manner with map-phase pool and reduce-phase pool to maximize the cluster utilization the idle reduce slots are borrowed from reduce slots at the reduce phase and the number of map slots, when it is insufficient for map tasks.

4.1 Pool-Dependent DHSA (PD-DHSA):

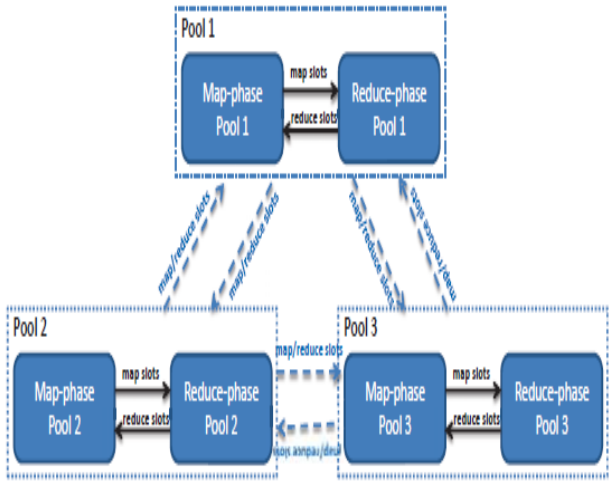


Fig. 3: Example of the fairness-based slot allocation flow for PDDHSA. The black arrow line and dash line show the borrow flow for slots across pools.

As the name indicates it is contrast to the (PI-DHSA) mechanism because Pool-Dependent DHSA (PD-DHSA) considers fairness for the dynamic slot allocation which can be described in fig 3 it states that map-phase pool and reduce-phase pool are selfish that reduce slots for its own needs. It achieves the fairness only on numbers of map and reduce slots allocated across pools are equal. Here also we have two sections such as Intra-pool dynamic slot allocation and Inter-pool dynamic slot allocation. First in Intra-pool dynamic slot allocation it receive a typed-slots based on max-min fairness at each phase based on the relationship between map Slots Demand, reduce Slots Demand. But in Inter-pool dynamic slot allocation it maximize its own need if possible by borrow some unused map and reduce slots from other pools.

4.2 Speculative Execution Performance Balancing (SEPB):

Straggler is a sensitive issue on MapReduce job's execution time. The reasons behind these are Hard Straggler and Soft Straggler (12) (13).

4.2.1 Hard Straggler:

It is the mechanism of deadlock status based on endless waiting for certain resources and we cannot stop until we kill it.

4.2.2 Soft Straggler: a successful computation of a task is achieved by longer time when compared to the common tasks.

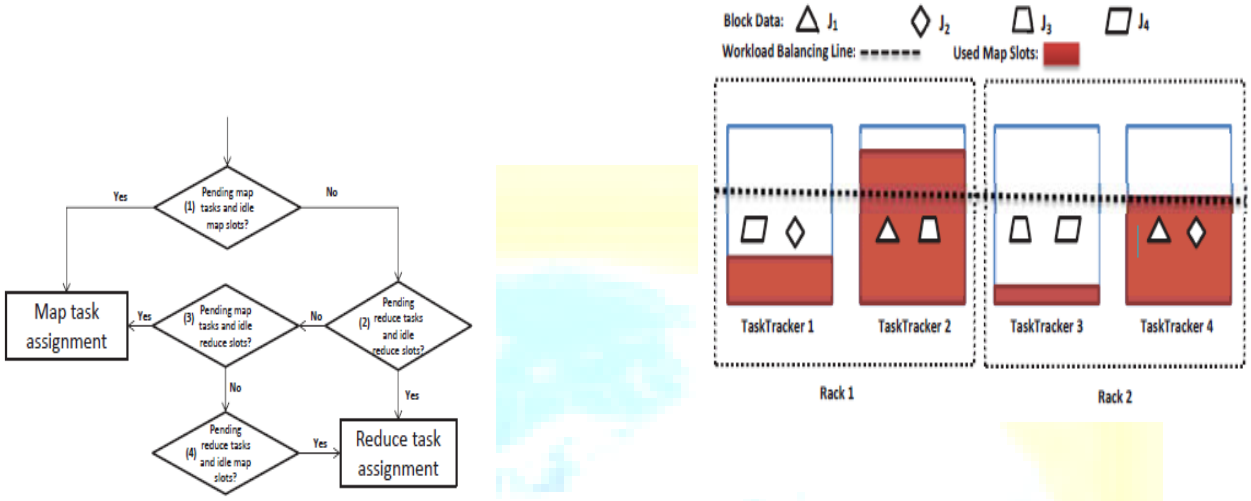


Fig 4: The overall slot allocation process for PD-DHSA

Some of the common drawbacks are Hadoop does not have a mechanism to distinguish between the hard straggler and the soft straggler and we cannot judge the straggler which belong to which during the running section. It also leads to mitigate its negative impact for the performance of a batch of jobs. So in order to maximize it performance for a batch of job it satisfy pending tasks first before considering speculative tasks. It can be employed in following order such as pending map task, pending reduce task, speculative map task, and speculative reduce task that achieves in maximize the performance for a batch of jobs. Fig 5 shows the computation policy for the total Num Of Pending Map Tasks and total Num Of Pending Reduce Tasks.

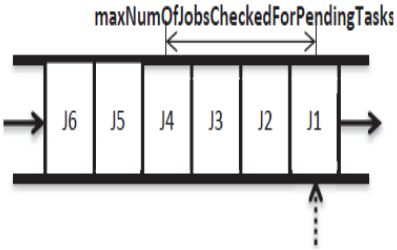


Fig. 5: The computation policy for the total Num Of Pending Map Tasks and total Num Of Pend-

ing Reduce Tasks.

Slot Pre-Scheduling: To achieve Data locality in a effective manner we implementing Slot Pre-Scheduling in order to improve the efficiency of slot utilization and performance. For this process we proposed a Delay Scheduler scheduling for a job by a small amount of time and also it detects map tasks from that job on a node with the input on its. The delay scheduler is also act as a tradeoff between the data locality and fairness optimization. As it overcomes the problem faced during HFS. The basic idea behind this is idle slots which cannot be allocated during run-time because of load balancing constrain so we can maximize the data locality by pre-allocating those slots to the node.

Fig 6: An example of unbalanced workload distribution of running map tasks in a Hadoop cluster

The above figure shows current number of map slots that can be used under an ideally balanced case and also act as the tradeoff between data locality and load balancing. On this image load balancing between machines is represented in white area and below the workload balancing line improving the data locality considering with the load balancing issue. The general fact of these observations is Lots of Map Reduce clusters, varied computing loads to the varied input data sizes and execution time for map (or reduce) tasks may still not be the same, due to the skew caused by an uneven distribution of input data to tasks.

5 EXPERIMENTAL EVALUATION:

We implement our proposed work on cluster consisting of 10 compute nodes, each with two Intel X5675 CPUs (4 CPU cores per CPU with 3.07 GHz), 24GB memory and 56GB hard disks. We configure one node as master and name node, and the other 9 nodes as slaves and data nodes. In our experiment we use Hadoop version 1.2.1 and generate workloads by choosing 9 benchmarks arbitrarily from Purdue Map Reduce Benchmarks Suite [14].

If you are using Word, use either the Microsoft Equation Editor or the MathType add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation or MathType Equation). “Float over text” should not be selected.

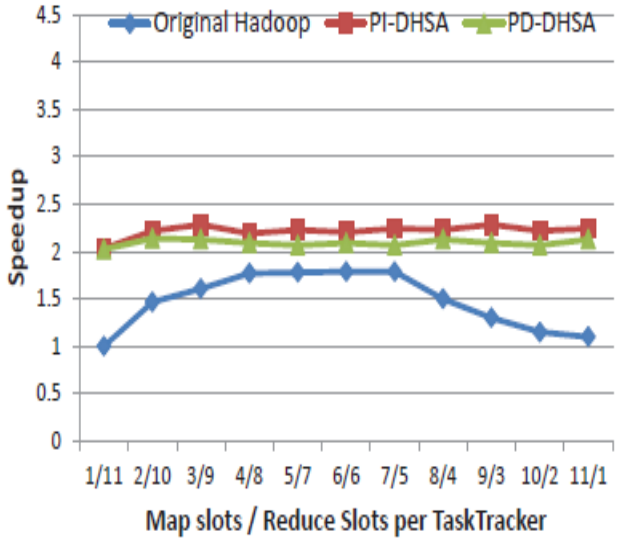
6.1 RESULT AND DISCUSSION:

The dynamic tasks execution processes for PI-DHSA and PD-DHSA is got using the benchmark dataset as given below. The outcome result is comparison with original Hadoop under various slot configurations with our proposed DHSA. In which 12 CPU cores per slave node and we assume that one Map Reduce slot corresponds to a CPU core. We can make the process effective **by** vary the number of map slots per slave node from 1 to 11 based on the ratio of the execution time of the original Hadoop under 1/11 map/reduce slot configuration per slave node, to the current execution time.

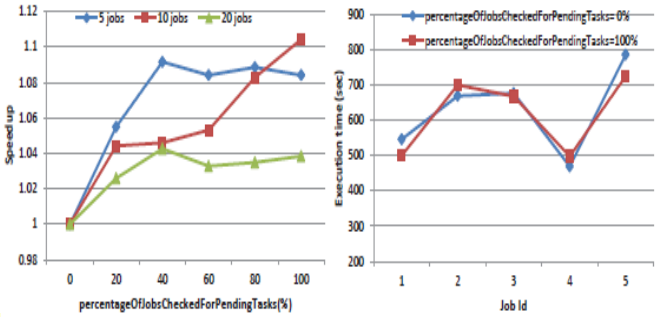
Benchmark		Input Data	
Name	Description	Data Source	Data Size (GB)
WordCount	computes the occurrence frequency of each word in a document.	wikipedia [29]	10
Sort	sorts the data in the input files in a dictionary order.	wikipedia [29]	20
Grep	finds the matches of a regex in the input files.	wikipedia [29]	30
InvertedIndex	takes a list of documents as input and generates word-to-document indexing.	wikipedia [29]	40
Classification	classifies the input into one of k pre-determined clusters.	movie ratings dataset [29]	20
HistogramMovies	generates a histogram of input data and is a generic tool used in many data analyses.	movie ratings dataset [29]	20
HistogramRatings	generates a histogram of the ratings as opposed to that of the movies based on their average ratings.	movie ratings dataset [29]	20
SequenceCount	generates a count of all unique sets of three consecutive words per document in the input data.	wikipedia [29]	30
SelfJoin	generates association among k+1 fields given the set of k-field associations.	synthetic data [29]	40

Table 1: The job information for Purdue MapReduce benchmarks and data sets.

Fig 8: Performance Improvement Comparison



The DynamicMR consisting of three different dynamic slot allocation policies such as DHSA, SEPB, and Slot Pre-Scheduling and the comparison results are based on different metrics that is fairness, slot utilization, and performance. The improvement of Map Reduce on workloads along with the slot utilization and optimization is shown in below figure. In the case of data locality the Slot Pre-Scheduling map tasks on the Hadoop cluster mentioning that Delay Scheduler has been added to the default HFS by overcoming the challenges on traditional Hadoop. The fig 10 shows the batch submission on all jobs submitted at the same time under the optimized map/reduce slot configuration for the original Hadoop, and our DynamicMR system can still further improve the performance of Map Reduce jobs significantly by assigning workloads on multiple jobs.

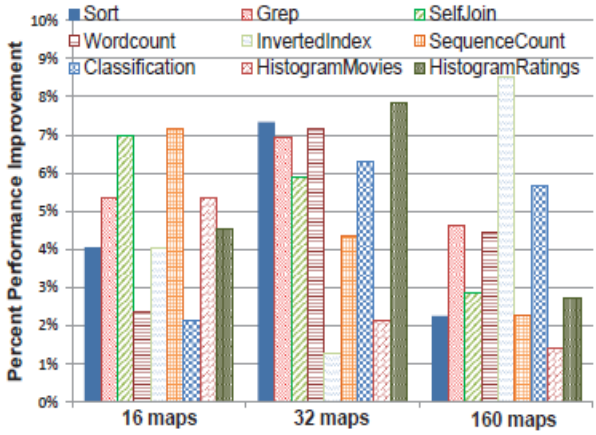


(a) The performance improvement under different percentages.

(b) The detailed job execution time for the workload of 5 jobs.

Fig 10: The performance improvement under Slot Pre-Scheduling.

Fig 9: The performance results with SEPB.



7 CONCLUSION AND FUTURE WORK:

Our proposed DynamicMR framework is employed with the motto of high performance of Map Reduce workloads with achieving its fairness. To make it possible we include three strategies such as DHSA, SEPB, and Slot Pre-Scheduling all are focused on slot utilization optimization for Map Reduce cluster from different perspectives. We have the experimental results of DHSA with its two factors namely PI-DHSA and PD-DHSA that achieves its fairness on on different levels. The slot utilization performance were also increased by SEPB which identifies the slot inefficiency problem of speculative execution and improve its process by building a tradeoff between a single job and a batch of jobs dynamically. As well as Slot Pre-Scheduling also improves its efficiency of slot by maximizing its utilization on data locality. As stated the experimented results proves the effectiveness of our proposed strategy is far better than the tradition hadoop on slot allocation process. The future work can be extended by implementing DynamicMR on cloud computing environment with addition metrics such as budget, deadline on different platforms.

REFERENCES

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters, In OSDI'04, pp. 107-113, 2004.

[2] Hadoop. http://hadoop.apache.org.

[3] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E.Harris, Reining in the outliers in map-reduce clusters using mantri, in OSDI'10, pp. 1-16, 2010.

[4] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica, Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In EuroSys'10, pp. 265-278, 2010.

[5] C. O˘guz, M.F. Ercan, Scheduling multiprocessor tasks in a two-stage flow-shop environment. Proceedings of the 21st international conference on Computers and industrial engineering, pp. 269-272, 1997.

[6] Z.H. Guo, G. Fox, M. Zhou, Y. Ruan.Improving Resource Utilization in MapReduce. In IEEE Cluster'12. pp. 402-410, 2012.

[7] Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apac .org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

[8] J. Tan, X. Q. Meng, L. Zhang. Coupling task progress for MapReduce resourceaware scheduling. In IEEE Infocom'13, pp. 1618-1626, 2013.

[9] J. Tan, S. C. Meng, X. Q. Meng, L. Zhang. Improving ReduceTask data locality for sequential MapReduce jobs. In IEEE Infocom'13, pp. 1627-1635, 2013.

[10] Y. Wang, W. Shi, Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds, IEEE Transaction on Cloud Computing, 2014

[11] J. Polo, Y. Becerra, et al. Deadline-Based MapReduce Workload Management, IEEE Transactions on Network and Service Management, 2013.

[12] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica, Job Scheduling for Multi-user Mapreduce Clusters. Technical Report EECS-2009-55, UC Berkeley Technical Report (2009).

[13] T. White. Hadoop: The Definitive Guide, 3rd Version. O'Reilly Media, 2012.

[14] M. Zaharia, A. Konwinski , A.D. Joseph , R. Katz , I. Stoica, Improving MapReduce performance in heterogeneous environments. In OSDI'08, pp.29-42, 2008.

[15] F. Ahmad, S. Y. Lee, M. Thottethodi, T. N. Vijaykumar. PUMA: Purdue MapReduce Benchmarks Suite. ECE Technical Reports, 2012.