International Journal For Technological Research In Engineering
Volume 3, Issue 9, May-2016 *ISSN (Online): 2347 - 4718*

A SIMPLE AND EFFICIENT PROTOTYPE FOR DYNAMIC SLOT ALLOCATION FOR HADOOP CLUSTER

Kawsar Jahan[1], Sujatha. K[2]
[1]M. Tech Scholar, [2]Asst. Professor, Advanced Information Technology School of Computing & IT, REVA University, Bangalore, India

Abstract: Hadoop is developed as a remedy to perform large scale data parallel application in cloud computing environment. Hadoop framework is basically defined with three factors: Cluster, workload and User. Each of these is either homogeneous or heterogeneous which represents the heterogeneity level of the Hadoop. In this work we considered the heterogeneity impact for every element using the scheduler outcome. Performance evaluation is showed for the Hadoop attributes such as makes span and also resource utilization. .
Keywords: Hadoop, Homogeneous, Heterogeneous, Makespan, Scheduler.

I. INTRODUCTION

Hadoop is inspired from an earlier Apache project known as "Nutch" related to designing of open source web based search engine. This project was affected by various issues like expensive hardware and financial support to manage monthly expenses. In 2002 a working prototype was released, but a issue of scalability arise when the developer's stated the architecture would fail to handle billions of pages in the web. In 2003 a paper was released describing about Google File System (GFS). This provided the Nutch group a clue that they can utilize that order in order to store pages as to overcome the issue of scalability. Later a new projected was initiated to make the implementation of their own open source similar to the GFS. Another paper was published by the Google describing Map Reduce programming paradigm and Nutch group designed the algorithms to run in that standard. Finally with a support of Yahoo a new project known as Hadoop was emerged [1]. Hadoop is a data-intensive cluster computing system, in which incoming jobs are defined based on the Map Reduce programming model. Map Reduce is a popular paradigm for performing computations on Big Data in Cloud computing systems A Hadoop system consists of a cluster, which is a group of linked resources. Organizations could use existing resources to build Hadoop clusters - small companies may use their available (heterogeneous) resources to build a Hadoop cluster, or a large company may specify a number of (homogeneous) resources for setting up its Hadoop cluster. There can be a variety of users in a Hadoop system who are differentiated based on features such as priority, usage, guaranteed shares, etc. Similarly, workload in the Hadoop system may have differing numbers of users' jobs and corresponding requirements. Therefore, a Hadoop system can be specified using three main factors: cluster, workload, and user, where each can be either heterogeneous or homogeneous. Cluster: Is a group of linked resource, where each resource consists of a computation unit and also a data storage unit. The computation unit contains a set of slots wherein every slot is having a specific execution rate. In majority of hadoop systems, every CPU core is accounted as single slot. Likewise, data storage unit as specific capacity as well as data retrieval rate. In hadoop system data are organized as files usually which are large in size. Every file is spitted into tiny pieces, known as slices. Generally every slice in the system have similar size. User: submits jobs to the system. Hadoop assigns a priority and a minimum share to each user based on a particular policy (e.g. the pricing policy in the user's minimum share is the minimum number of slots guaranteed for the user at each point in time. Workload: consists of a set of jobs, where each job has a number of map tasks and reduces tasks. A map task performs a process on the slice where the required data for this task is located. A reduce task processes the results of a subset of a job's map tasks. The value defines the mean execution time of job Join resource Investigations on real Hadoop workloads show that it is possible to classify these workloads into classes of "common jobs" We define the class of jobs to be the set of jobs whose mean execution times (on each resource) are in the same range. There are various Hadoop schedulers, where each scheduler may consider different levels of heterogeneity in making scheduling decisions. Moreover, schedulers are differentiated based on different performance metrics (e.g., fairness, minimum share satisfaction, locality, and average completion time) that they address. There is a growing demand to use Hadoop for various applications which leads to sharing a Hadoop cluster between multiple users. The paper is organized as follows, Section II discusses about Literature survey performed highlighting the contribution of different authors in the respective domain. Section III provides a discussion on the problem statement. Section IV illustrates the research methodology. Section V illustrates the Implementation aspects of the project. Whereas result discussion is provided in Section VI and discussion on conclusion is provided in section VII. In the following section various contributions from different authors in hadoop domain is discussed.

II. LITERATURE SURVEY

Following section highlights the different research works and their contribution in the hadoop domain. Mirajkar et al. [3] Performed a word count Map-Reduce Job in Single Node Apache Hadoop cluster and compress data using Lempel-Ziv-Oberhumer (LZO) algorithm.

Shi et al. [4] Presented a toolbox from IBM, called MRTuner, to facilitate holistic optimization for Map Reduce tasks. Specifically, authors suggested Producer-Transporter-Consumer (PTC) model, which describes the tradeoffs in the parallel execution between tasks. Authors additionally explored the complex relations among around twenty parameters, which have huge effect on job performance. Authors have outlined a effective search algorithm to identify the ideal execution plan. At last, led an intensive test assessment on two distinct sorts of clusters using the HiBench suite which covers different Hadoop workloads from GB to TB size levels. The outcomes demonstrated that the search latency of MRTuner quicker than that of the state of art cost-based optimizer, and the effectiveness of the optimized execution plan is also drastically improved.

Rao and Reddy [5]. Reviewed different possibilities of scheduler improvements with Hadoop and also demonstrated certain guidelines on improving the scheduling in hadoop on distributed environments.

Nayak et al. [6] Proposed the Adaptive Scheduler (AS).In AS the client's needs to submit a Service Level Agreement (SLA) along with the job. Using the SLA it is checked whether the vendor is possible to accommodate the job in order to meet the SLA.If it is achieved the AS schedules and executes the job through SLA. If not, client is informed to negotiate with AS so as to both parties can agree on. This pre-agreement in between the vendors and client will be advantageous for both. The benefit of the proposed AS is demonstrated in comparative review available in many existing schedulers in Hadoop.

Rasooli and Down [7]. Examined the performance of typically used hadoop schedulers that consists of FIFO and Fair Sharing (FS).Authors also performed the comparison of the aforementioned algorithms with COSHH (Classification and Optimization Based Scheduler for Heterogeneous Hadoop) developed by the authors. On the basis of their judgment a hybrid solution is presented which chooses the appropriate scheduling algorithm for scalable and heterogeneous Hadoop system in regard to the incoming jobs and resource availability.

Xie et al. [8] imported a pre-fetching mechanism into Map Reduce model by preserving its compatibility with the resident Hadoop. Provided a application using massive data is using a Hadoop cluster, this strategy will estimates the time required for executing every task and also preloads data adaptively to the memory prior to the new task allocated to computation node.

Xia et al. [9] based on the node health degree nodes are grouped into three categories so as to assign relevant job according to the load and guarantee resource load balance. By comparison with FIFO and Fair scheduling algorithm through simulation it is seen that proposed algorithm ensures to minimize job fail rate and enhances cluster throughput.

Yao et al .[10] Suggested a Hadoop scheduler that leverages the information of workload patterns to minimize average job response time through dynamically tunings the resource shares between the users and the each users scheduling algorithm. From the simulation as well as real experiment from Amazon EC2 cluster it is evident that the proposed scheduler minimizes the average Map Reduce job response time in different system workload scenario in comparison to FIFO and Fair Schedulers.

Wang et al. [11] proposed an improved Hadoop system known as FRESH which can provide the best slot setting, configure slot dynamically and assign tasks to the slots appropriately. From the experimental result it is seen that when serving a batch of Map Reduce job, FRESH drastically enhances the make span and also fairness between jobs.

Yao et al. [12] Suggested YARN scheduler known as HaSTE which will effectively minimize the make span of Map Reduce jobs in YARN by leveraging information of requested resource, capacity of resource and task dependency. Authors have used HaSTE as a pluggable scheduler in recent Hadoop YARN and carried evaluation with traditional Map Reduce benchmarks. Experimental results showed that the suggested YARN scheduler effectively minimizes the makes span and enhances resource utilization in compared to the present scheduling policies.

Verma and Cherkasova [13]. Presented a basic abstraction wherein every Map Reduce job is represented as a couple of map and reduce stage durations. Due to such representation the authors were enable to apply traditional Johnson algorithm which is designed for creating an ideal two-level job schedule. Authors evaluated the performance advantages of the proposed schedule through an detail set of simulations across different realistic workloads. Results are dependent on the cluster size and workload. Authors have designed a heuristic called Balanced Pools which significantly enhances Johnson's Schedule result in the scenarios where it produce sub-ideal make span. Through the simulation authors have validated the experiments in 66 node hadoop cluster.

Huang et al. [14] suggested two speculative technique Such as "Estimate Remaining Time Using Linear Relationship Model" (ERUL) and "Extensional Maximum Cot Performance (exMCP)" ,these techniques are designed to enhance the prediction of the job's pending time. ERUL is a dynamic load-aware mechanism that is used by the authors to provide a solution to the "Longest Approximate Time to End" (LATE), exMCP takes various slot values. Through the investigation it is seen that ERUL and exMCP are linked to accurate estimation of running task's remaining execution time and reduces the execution time of job.

Bardhan and Mensace [15]. Presented a mathematical model on the basis of closed Queuing Networks to predict the execution time of the Map stage of a Map reduce job. The model traps contention in the compute node and gains parallelism due to the raised number of slots available to map tasks. Model is validated on both single and a two node hadoop environment through experiments. Authors performed the experiments using different split sizes for inputs and also different size of map slot to validate their model.

Luo et al. [16] presented a Hierarchical Map Reduce framework which collects the computational resources from various clusters and executes Map Reduce jobs over them. The framework consists of a global controller which

performs the splitting of dataset and sends them to multiple local Map reduce clusters and balances the workload by allocating task as per the cluster capacity and node capacity. Global reduction is performed on the basis of local results sent back to the global controller. Using Auto Dock in Map Reduce the experiments is carried out which demonstrated that load balancing algorithm provides a reliable workload distribution over multiple clusters and also minimizes the total execution time of the complete Map Reduce execution.
Zhu et al. [17] Studied logged offline scheduling of reducing make span and reducing overall completion time, respectively. Authors have considered both pre-emptive as well as non-preemptive reduce tasks. For make span reduction in pre emptive side authors have provided a guide line in the form of algorithm and provided its optimality for non-preemptive side. Authors composed an approximation algorithm with worst ratio of 3/2-1/2h where number of machines is represented by h. On overall complete time reduction, for non-preemptive side authors have devised a heuristic. Authors likewise affirm that their algorithm outperforms state of-art schedulers through experiments.

Malekimajd et al. [18] Presented a fresh upper as well as lower bounds for Map Reduce job execution period in shared Hadoop cluster, Authors have also presented a linear programming model that is capable of reducing cloud resource costs and job rejection penalties for the multiple class job execution with deadline guarantees. From the simulation it is seen that execution time of Map Reduce jobs drop within 14% of the suggested upper bound in average. From the numerical analysis it is seen that that the suggested method is capable of determining the global optimal solution of linear issue to the system consists of 1000 user class within 0.5sec.

Zhang et al. [19] Developed a efficient as well as a fast simulation framework to evaluate and select the appropriate underlying platform to achieve the require "Service Level Objectives" (SLOs). From the evaluation study performed using Amazon EC2 platform reveals that an ideal platform selection may result in 45-68% cost saving for different workload mixes. More ever based on the workload the homogeneous cluster is outperformed by the heterogeneous solution by 26-42%.Simulation results are validated by experiments by deploying Hadoop cluster on variety of Amazon EC2 instances.

III. PROBLEM DESCRIPTION

This section provides a brief discussion on the research problem pertaining to the work. Hadoop provides the adaptability to modify the cluster for different applications since it is configured with huge set of system parameters. It is little challenging task for the user comprehend and set the ideal values to those parameters. A typical Hadoop cluster incorporates a single master node and more than one slave nodes. The master node runs the Job Tracker routine which is in charge of scheduling jobs and organizing the execution of assignments of every job. Every slave node runs the Task Tracker daemon for facilitating the execution of Map Reduce jobs. The idea of "slot" is utilized to provide the ability of each node to accommodate task.

In a Hadoop framework, a slot is allocated as a map slot or a reduce slot serving map task or reduce tasks, respectively. At any given time, only single assignment can be running per slot. The number of accessible slots per node gives the high level of parallelization in Hadoop, the Hadoop system, in any case, utilizes fixed number of map slot and reduces slots at every node as the default setting all through the lifetime of a cluster. In the fixed designs the values used are generally the heuristic number which does not take into account the job features. therefore static settings are not properly customized and may prevent the performance enhancement of the complete cluster.

IV. RESEARCH METHODLOGY

This section briefs about the research methodology of the project. In order to minimize the makes span time of job's, hadoop system allocates the task among the resources. Although hadoop systems do not consider communication cost. In heterogeneous resources with large cluster increasing a task's distribution will result in large communication overhead. This in turn results in an increased completion time. COSHH takes into account of heterogeneity and resource distribution during task assignment.

In order to maximize locality, it's needed to increase the probability of the task that are allocated to resources, which store their input data. Based on the suggested set of job classes for every resource the decision in COSHH scheduling is performed. Therefore, the necessary data of the suggested classes of a resource is replicated on that resource. This can help to enhance locality, in specific in huge hadoop clusters where locality is more crucial.

From the result it is seen that COSHH provides notably better performance in minimizing the average completion time, and fulfilling the required minimum shares. Compared to other two schedulers the performance for locality and fairness metric is competitive.

Queuing Process uses two main strategies known as classification and optimization based strategy. At the high-level at the arrival of new job classification strategy specifies the job class, and performs the storing of the job in appropriate queue. In case if the arrived job does not fit any of the present classes, list of classes is updated to add a class for incoming job. Optimization approach is utilized to identify an appropriate matching of job classes and feature of the resources. The result of queuing process is forwarded to the routing process consisting the list of job classes as well as suggested set of classes for every resource

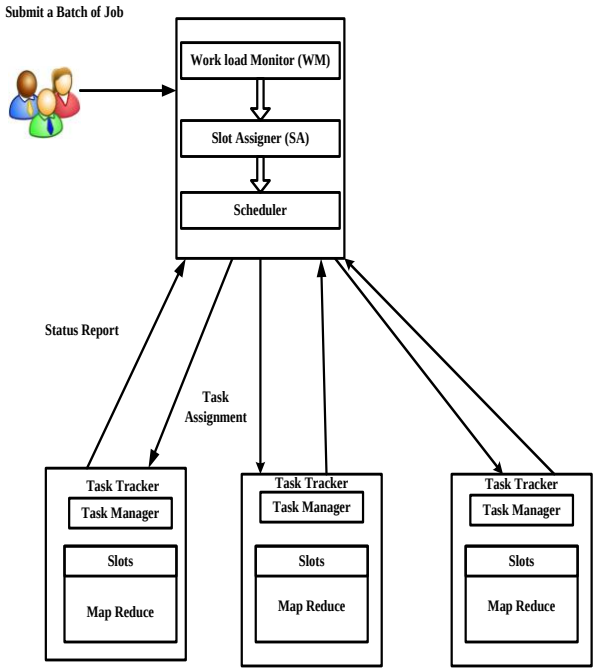

Figure 1: Architecture of the Proposed System.

Routing Process: When a free resource sends a heartbeat message to the scheduler, triggers the routing process. The job selection for the present free resource is done on the basis of the suggested set of classes from the queuing process that is sent to routing process. This process chooses a job for every free slot in resource and the selected job is sent to the task scheduling process. Task scheduling process selects a task of the selected job and allocates the task to its respective slot. Here the scheduler is not restricted to just one resource. After the selection of the job task scheduling process allocates number of appropriate task of this job to the available slots of present free resource. If the available slot is lesser than the number of incomplete task for the selected job, the job remains in the queue. During the next heart beat message from the free resource this job is taken into account at the time of decision making. Already assigned task will not be considered. Once every task of the job is assigned the job will be eliminated from the waiting queue. Routing process uses the algorithm. The selection of job for the available slot of the present free resource is done in two stages. In the initial stage, jobs in the classes in SCR are taken into account wherein the job selection is achieved on the basis of their minimum share satisfaction. Which implies that user with highest distance to achieve minimum share will be allocated resources early. In the next stage jobs in the classes SCR are taken to account in the order specified by the present shares and user priorities.

V. IMPLEMENTATION AND ALGORITHM

This section discusses about the algorithm and program flow process used in the implementation process. The figure 2 illustrates the flowchart of the dynamic slot allocation process in hadoop cluster.

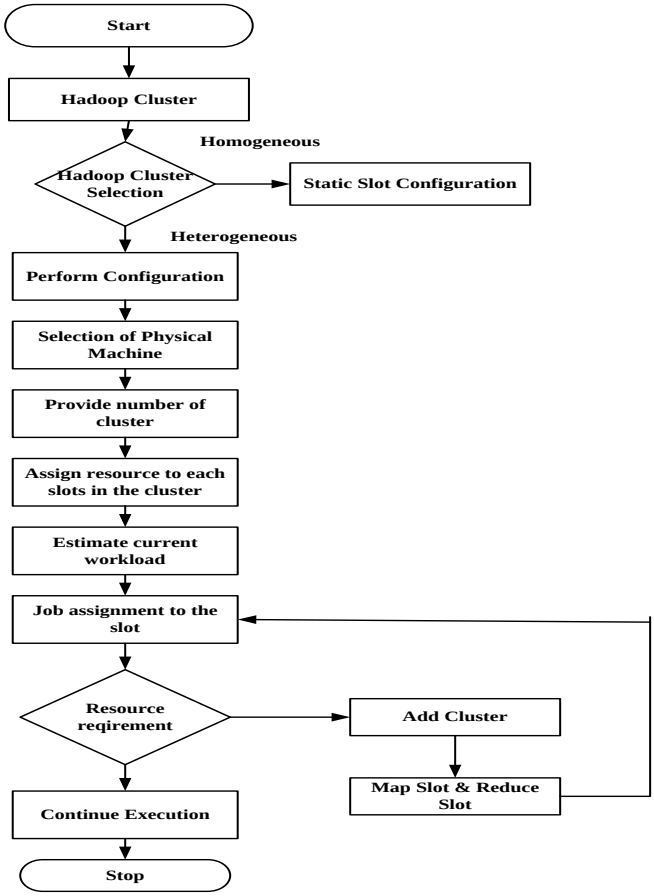

Figure 2: Flow process of dynamic allocation in Hadoop cluster.

The figure below illustrates the flow diagram of the process, initially the hadoop cluster are selected. On selection if the choice s homogenous cluster, then the configuration used is the statically available one. In case of the heterogeneous cluster the configuration of involves providing Physical machine, allocating cluster as well as providing the resources to the slots. After the resource allocation, slot assignment module in the work load monitor will estimate the current workload. Job tracker will perform the job assignment. The jobs are randomly generated in the simulation. During the processing if the requirement of the resource arises the job tracker will dynamically provide the resources by adding a cluster and allocating resource if not the job execution continues.

Table 1: Algorithm for dynamic allocation in Hadoop Cluster.

Input: Job task request
Output: Execution time
Start:
1.select hadoop environment
2.if (H_e == $H_{m)}$
3.{
4. Static Config= {Physical Machine,Cluster,Slots)
5.}
6.else

7. if (H_e==H_t)
8.{
9.Dynamic Config={Physical Machine,Cluster,Slots)
10.estimate the slots in map task S_m and S_r
11.Check if (S_m + $S_r \leq S_T$)
12.check if Sm has sufficient resource to execute workload
13.if (Yes)
14. {
15.Continue
16. else
17.Reschedule resource from S_r to S_m
18.Stop

The above table explains the algorithm used for the dynamic allocation in hadoop cluster. In case of the homogenous the configuration is statically performed as depicted in the process flow diagram. In heterogeneous the configuration is performed in dynamically which involves the selection of physical machine, cluster and also slots. Here the slots are also divided in between the map task and reduce task. The algorithm checks for the total slots required for the execution of the task. It also estimates available slots in map task as well as the reduce task. The slots in the map task (Sm) and reduce task (Sr) add up to the total slot (ST). it then checks the individual slots in the map task as well as the reduce task. It performs the analysis to check if the Sm is having sufficient resource to execute the job ,if yes it continues execution, if not the rescheduling of the resource is performed by allocation the resource from Sr without effecting the performance. The same process is performed in case of Sr if it resource deprived.

VI. RESULT ANALYSIS

This section discusses the evaluation of the proposed algorithm and graphical illustrates the outcome. In figure 3 below provides the comparison of the resource utilization in homogeneous and heterogeneous environment. From the graphical presentation it is seen that the resource utilization in the heterogeneous environment is far better than the homogeneous environment. This is due to the fact that the resource utilization is improved by avoiding the starvation of resource which results in poor performance. The evaluation is performed theoretically using hypothetical observation.

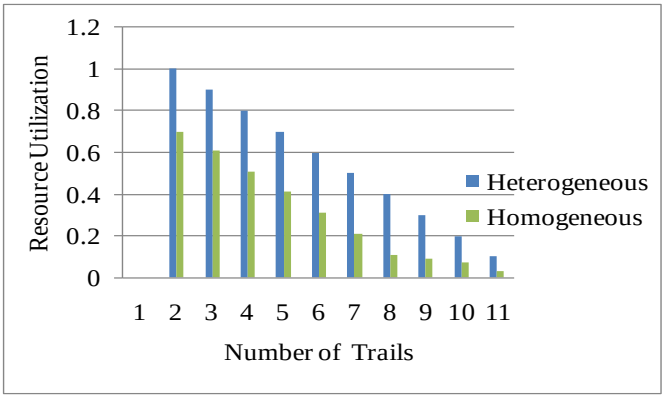

Figure 3: Resource utilization comparison.

Another crucial observation carried out in work is the impact of make span time which also contributes to the performance of the system.

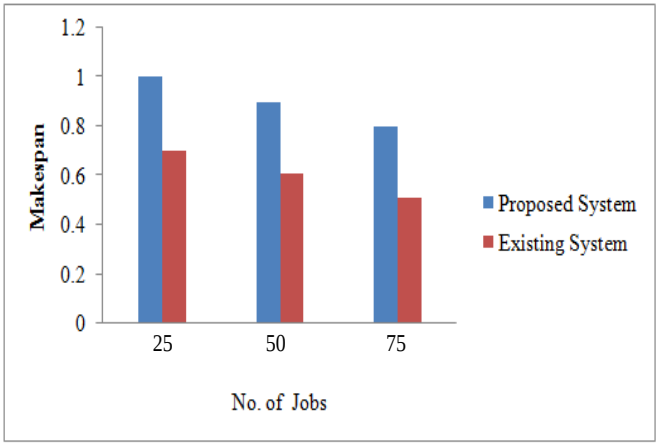

Figure 4: Make span Comparison

The above figure illustrates the make span comparison of the proposed system with the existing system. The comparison is performed for three trails of different number of jobs. From the graphical observation it is seen that the proposed system provides a better make span compared to the existing system. The evaluation is performed hypothetically.

VII. CONCLUSION

Dynamic Adjusting slot configuration is a significant aspect at the time of processing a huge dataset using Map Reduce Paradigm. This will optimize the performance of the Map Reduce fraework. Every job is scheduled by the job tracker using any one of the scheduling strategy. Slot assignment to the jobs is performed by the task manager present in the task tracker. The efficiency as well as robustness of our proposed slot management method is validated in both homogeneous and also heterogeneous cluster environment. The experimental results demonstrate the effectiveness and robustness of our schemes under both simple workloads and more complex mixed workloads. The project analyzed the impact of heterogeneity in every aspects of the hadoop scheduler performance.

REFERENCES

[1] Rasooli, Aysan, and Douglas G. Down. "Guidelines for selecting hadoop schedulers based on system heterogeneity." Journal of Grid Computing 12.3 (2014): 499-519.

[2] Krish, K. R., Ali Anwar, and Ali R. Butt. "[phi] Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler." Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on. IEEE, 2014.

[3] Mirajkar, Nandan, Sandeep Bhujbal, and Aaradhana Deshmukh. "Perform wordcount Map-Reduce Job in Single Node Apache Hadoop cluster and compress data using Lempel-Ziv-Oberhumer (LZO) algorithm." arXiv preprint arXiv:1307.1517 (2013).

[4] Shi, Juwei, et al. "MRTuner: A toolkit to enable holistic optimization for mapreduce jobs." Proceedings of the VLDB Endowment 7.13 (2014): 1319-1330.
[5] Rao, B. Thirumala, and L. S. S. Reddy. "Survey on improved scheduling in Hadoop MapReduce in cloud environments." arXiv preprint arXiv:1207.0780(2012).
[6] Nayak, D.; Martha, V.S.; Threm, D.; Ramaswamy, S.; Prince, S.; Fatimberger, G., "Adaptive scheduling in the cloud — SLA for Hadoop job scheduling," in Science and Information Conference (SAI), 2015 , vol., no., pp.832-837, 28-30 July 2015
[7] Rasooli, Aysan, and Douglas G. Down. "A hybrid scheduling approach for scalable heterogeneous Hadoop systems." High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:. IEEE, 2012.
[8] Xie, Jiong, et al. "Research on scheduling scheme for Hadoop clusters."Procedia Computer Science 18 (2013): 2468-2471.
[9] Xia, Yang, et al. "Research on job scheduling algorithm in hadoop." Journal of Computational Information Systems 7.16 (2011): 5769-5775.
[10] Yao, Yi, et al. "LsPS: A Job Size-Based Scheduler for Efficient Task Assignments in Hadoop." Cloud Computing, IEEE Transactions on 3.4 (2015): 411-424.
[11] Wang, Jiayin, et al. "Fresh: Fair and efficient slot configuration and scheduling for hadoop clusters." Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on. IEEE, 2014.
[12] Yao, Yi, et al. "Haste: Hadoop yarn scheduling based on task-dependency and resource-demand." Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on. IEEE, 2014.
[13] Verma, Abhishek, Ludmila Cherkasova, and Roy H. Campbell. "Orchestrating an ensemble of MapReduce jobs for minimizing their makespan." Dependable and Secure Computing, IEEE Transactions on 10.5 (2013): 314-327.
[14] Huang, Xin, et al. "Novel heuristic speculative execution strategies in heterogeneous distributed environments." Computers & Electrical Engineering (2015).
[15] Bardhan, Shouvik, and D. Menasce. "Queuing network models to predict the completion time of the map phase of mapreduce jobs." Proceedings of the Computer Measurement Group International Conference. 2012.
[16] Luo, Yuan, et al. "A hierarchical framework for cross-domain Map Reduce execution." Proceedings of the second international workshop on Emerging computational methods for the life sciences. ACM, 2011.
[17] Zhu, Yuqing, et al. "Minimizing makespan and total completion time in mapreduce-like systems." INFOCOM, 2014 Proceedings IEEE. IEEE, 2014.
[18] Malekimajd, Marzieh, et al. "Optimal map reduce job capacity allocation in cloud systems." ACM SIGMETRICS Performance Evaluation Review 42.4 (2015): 51-61.
[19] Zhang, Zhuoyao, Ludmila Cherkasova, and Boon Thau Loo. "Exploiting cloud heterogeneity to optimize performance and cost of MapReduce processing."ACM SIGMETRICS Performance Evaluation Review 42.4 (2015): 38-50.

