ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

A Survey of Hadoop Map Reduce Scheduling Algorithms

Mani Bushan Dsouza

Asst. Prof, Department of MCA, AIMIT, St. Aloysius College (Autonomous), Mangalore, Karnataka, India

ABSTRACT*:* Hadoop is an open source framework for distributed data storage and processing big data on cluster of commodity hardware. MapReduce is programming model that allows us to write functional-style code which gets scheduled over distributed data across multiple machines. However, the network sharing among various applications can lead to constrains on network bandwidth. Effective scheduling of the reduce tasks to the resources becomes especially important for the performance of data-intensive applications where largeamounts of data are moved between the map and reducetasks. There are three important scheduling issues in MapReduce namely, as locality, synchronization and fairness. The most common objective of scheduling algorithms is to minimize the completion time of a parallel application and also solve the said issues. In this paper, we focus on important scheduling algorithms used in Hadoop and highlighting advantages and disadvantages.

KEYWORDS*:* MapReduce, Scheduling, Adaptive, Non-Adaptive, Hierarchical, CapacityScheduling,First-Come First-Serve Scheduling ,Fairness Scheduling , SLA-based Scheduling, Map-Task, Reduce-Task, Speculative-Task.

I. INTRODUCTION

Hadoop MapReduce follows master/slave architecture. The master process is called JobTracker, it distributes tasks and co-ordinates their input/output among the HDFS DataNodes. The slave process called TaskTracker runs on each node in cluster. Each slave is assigned a fixed number of map slots and reduce slots based on configuration and/or the number of cores in the node. The TaskTracker executes map and reduce tasks, one per corresponding slot, on a DataNode as per master's instructions. It also manages the data flow to/from map/reduce tasks. Hadoop assumes a tree-style network topology. Nodes are spread over different racks contained in one or many data centers. The bandwidth between two nodes is dependent on their relative locations in the network topology. So nodes that are on the same rack have higher bandwidth betweenthem than those that are off-rack. This feature is used while scheduling data-local tasks among the nodes.

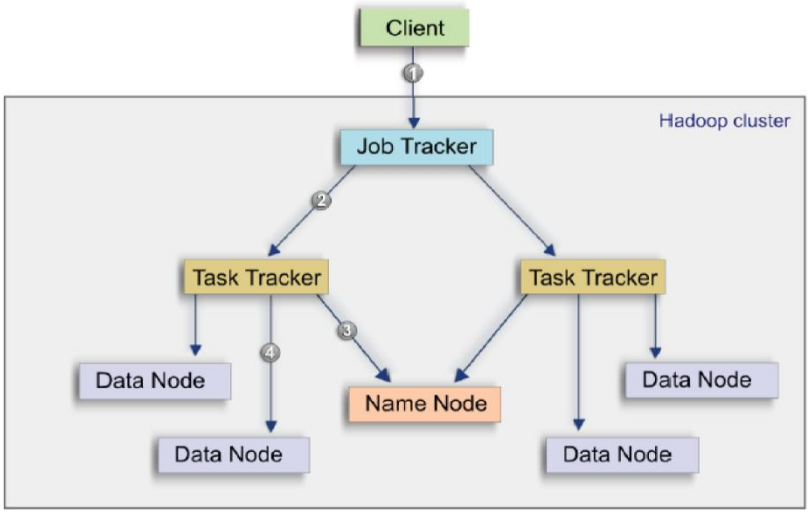

Figure 1. Hadoop Map-Reduce Implementation Architecture

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

Hadoop MR provides an easy and quick implementation framework for efficiently processing huge volume of data on distributed cluster. Users submit jobs by specifying the data location and map/reduce implementations which consisting of a map function and a reduce function to Hadoop MR. HadoopJobTracker distributes that program instance to TaskTrackers, breaks it into tasks, schedules them and tracks them till end.
Hadoop Map-Reduce, when used in shared cluster environments, handles multiple jobs from multiple users on multi-core machines. Hadoop MR provides a few pluggable/configurable job schedulers to enable different types of scheduling algorithms based on the data center's needs.

II. HADOOP SCHEDULING TECHNIQUES

In Hadoop, every TaskTracker sends frequent heartbeats containing the number of free map and reduce slots on that slave to the JobTracker. The JobTracker then assigns a task to the TaskTracker having free slots according to the configured scheduling policy. Hadoop schedulersmostly do a (2-level) hierarchical scheduling. First a job is selected from the pool of pending jobs, then that job's task is scheduled. The task scheduling is done based on the type of the slot available. If it is a map slot, then map tasks are scheduled as follows:

1. A map task bucket is selected from the 3 types of map tasks (3 buckets) in following orderof priority failed, normal, and speculative.
2. From a bucket then a map task is selected based on data locality. The order of preference for selecting a task is a map task having local chunk on available node; a map task that has data on another node within the same rack; a task having data on another node outside the rack.

If free slot is a reduce slot, the reduce tasks from the queue are randomly selected and scheduled.Multiple algorithms are available with Hadoop MapReduce like First-come First-serve, Priority based, and Capacity based. As there are multiple objectives to be met and many variablesavailable for scheduling, there is scope of improvement in these algorithms. Many variations andimprovements of the default scheduling algorithm are available.

III. OVERVIEW OF SCHEDULING ALGORITHMS FOR HADOOP MAP-REDUCE

A HADOOP MAP-REDUCE SCHEDULING PROBLEM

The key objective of Map-Reduce programming model is to parallelize the job execution across multiple nodes.Scheduling is a highly important factor, an inappropriate scheduling of tasks would fail to exploit the true potential of the system and offset the gain from parallelization. The scheduling policy is used to decide when and where (on which machine) a task is to be executed. The most common objective of scheduling is to minimize the completion time of a parallel application by properly allocating the tasks to the processors.
The MapReduce tasks scheduling is a NP-Hard problem as it needs to achieve a balance between the job's performance, data locality, user fairness/priority, resource utilization, network congestion and reliability. If scheduling policy considers data locality for selecting a task, it may have to compromise on the fairness as the node available may have data of some job which is not on head-of-line as per the fairness policy. Similarly if a task is scheduled based on job's priority, it is not necessary that it would have local data on the available node. This would impact job's performance, network utilization and thus also other job's performance. Map-Reduce cluster is used to execute multiple jobs of different users. So, the scheduling policy needs to decide which user, which job and then which task to be executed on which machine. The users/jobs may have different priorities. The jobs would have different complexity, characteristics and requirements. The scheduling needs to consider all these.

B TYPES OF SCHEDULING ALGORITHMS

There are various ways of classifying scheduling algorithms, Subsequent subsections we discuss about various categories of scheduling algorithms and types within them.

C ADAPTIVE VS. NON-ADAPTIVE

MapReduce scheduling algorithms can be based on their runtime nature i.e. how they work. There are two main categories of runtime behaviour based algorithms, namely adaptive and non-adaptive scheduling.An adaptive scheduling algorithm uses the previous, current and/or future values of parameters to make scheduling decisions. A non-adaptive scheduling algorithm, on the other hand, does not take into consideration the changes taking place in

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

environment and schedules job/tasks as per a predefine policy/order. The scheduling algorithms falling in each category are also shown in the Figure 2.

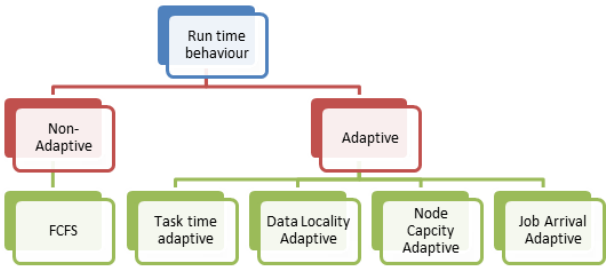

Figure 2: Taxonomy for Map-Reduce Scheduling Algorithms

D. HIERARCHICAL SCHEDULING ALGORITHMS

As the name suggests, Hierarchical Scheduling Algorithms follow a Hierarchical order. It works in shared environments, wherein a user, whose job needs to be scheduled, is first selected. Then from the list of jobs for the selected user it selects the job to be scheduled. Once the job is selected its map, reduce or speculative task is scheduled. Different decision making algorithms can be used at each level depending on the objective to be achieved. Multiple scheduling algorithms exist for each hierarchy level as shown in Figure 3.

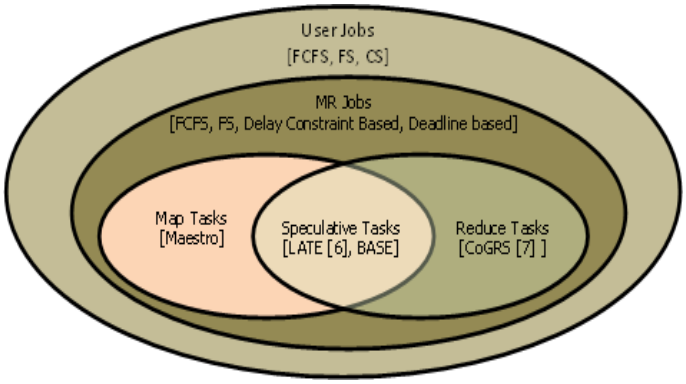

Figure 3: Hierarchical Scheduling policies proposed for Map-Reduce Clusters

E. USER LEVEL

User Level scheduling is categorized into following types

F. FAIR SCHEDULING ALGORITHM

The objective of Fair scheduling (FS) algorithm is to perform equal distribution of compute resourcesamong the users/jobs in the system [2]. In case of multiple users, one pool is assignedto each user [2]. The scheduling algorithm divides resources equally among these pools. The jobs within a pool can then be scheduled based on priority, FCFS or FS basis. Next thedata locality is considered for selecting a task of the selected job.FS scheduling algorithm provides a minimum share guarantee for each pool. If a pool doesnot get its minimum share for long time, FS pre-empts the most recently started task of anover allocated job from some other pool. This ensures that long batch jobs do not block theexecution of some production jobs and also lesser amount of resources are wasted (that spent inthe pre-empted task). The FS provides option to limit the number of jobs submitted in a poolto avoid concurrent execution of large number of jobs competing for cpu and memory resources.When Fair Scheduling algorithm is configured to consider job priorities, the priorities are usedto assign weights to the jobs and resources are allocated as per the normalized fractions to thejobs[8].Fair scheduling algorithms are useful in environments with different types of jobs. So thata lengthy batch job submitted first does not continuously occupy all the resources and a shortjob submitted later gets some amount of resources to finish off in relatively lesser time instead of having to wait for that long time.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

G.CAPACITY SCHEDULING ALGORITHM

The objective of Capacity scheduling is to maximize the resource utilization and throughput inmulti-tenant cluster environment. The design of capacity scheduling algorithm is very similarto Fair scheduling [2]. Here instead of pools, queues are used. Each queue is assigned toan organization and resources are divided among these queues. Additional security mechanismsare built for control access to the queues, so that each organization can access only its queueand cannot interrupt with other organization's queues, jobs or tasks.Similar to Fair scheduling, this algorithm offers minimum capacity guarantee by havinglimits on running/pending tasks and jobs from a single user/queue. For maximizing resourceutilization, it allows re-allocation of resources of free queue to queues using their full capacity.When jobs arrive in that queue, running tasks are completed and resource s are given back tooriginal queue. It also allows priority based scheduling of jobs in an organization queue.

H. JOB LEVEL

The job level scheduling policies available with default Hadoop Map-Reduce are First-ComeFirst serve (FCFS) and Priority based. These may not be efficient for all the scenarios andrequirements. There some new policies have been proposed for selecting a job for meetingthe different users/administrators requirements. The preliminary algorithms, their improvedversions and some newly proposed scheduling algorithms are described in this section.

I.FIRST-COME FIRST-SERVE SCHEDULING ALGORITHM

The objective of FIFO scheduler to schedule jobs based on their priorities in first-come firstserve order. FIFO is the default Hadoop scheduler which includes five priority levels as well. When the scheduler receives a heartbeat indicating that a map or reduce slot is free:

- It scans through list of jobs to find a job with highest priority and then oldest submit time.If this job has a pending task of the slot type (map/reduce) then that job is selected. Elsenext job in this order is picked. This goes on till a matching task (type as per slot) isfound.
- Next if its a map slot, then to achieved data locality the scheduler consults NameNodeand picks a map task in the job which has data closest to this free slave (on the samenode, otherwise on the same rack, or on a remote rack).
- If its a reduce slot, any of the reduce task is scheduled on the node. Hadoop MapReducedoesn't wait for all map tasks to finish for scheduling a reduce task, so the map taskexecution and shuffling of intermediate data can run in parallel to have better turn-aroundtime. This is known as early-shuffle.
-

J.FAIRNESS SCHEDULING ALGORITHM

The naive HadoopMapReduce Fair Scheduling at Job level always assigns free slots to the job that has the fewest running tasks to ensure that cluster is shared fairly between jobs. The newly submitted jobs are considered by fairness criteria whenever the next task-scheduling is to beperformed. Once the job is selected, one of its tasks which has local data on the node, rack or within LAN is launched on the available node.

Two locality problems can occur with this nave fair scheduling [3]

• The head-of-line scheduling - The head-of-line problem is that the job at head-of-line has low data locality then most of its tasks cannot be scheduled on local nodes. This problem mostly occurs for small jobs, which have small input files and hence has less number of data blocks to read. The probability of finding their data on a given node is low whenever they come to head-of-line, still some of their tasks get scheduled.

• The sticky slots - This problem can happen for both small as well as large jobs. The problem is that there is a tendency for a job to be assigned the same slot repeatedly. Suppose multiple jobs each having 10 tasks are submitted together on 10 node cluster. So each job is allocated 1 node, in order of their arrival. When job J finishes a task on node N, Node N requests a new task. At this point, J has 9 running tasks while all the otherjobs have 10. The nave Fair scheduling algorithm assigns that slot on node N to same job J again. Consequently, in steady state, jobs never get to leave a slot/node. This leads to poor data locality as the input files are striped across the cluster while all tasks of job are executed on same machine. To avoid the above mentioned data locality problem simple strategy to delay the execution of the job till it can't be executed on local data is proposed [3].

K.SLA-BASED SCHEDULING ALGORITHMS

The huge enterprise data is now being stored in distributed file systems for higher reliability and availability. This data is required as input not only for long batch jobs but also for many user jobs which many be online jobs or interactive

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

jobs. As these are user tasks, meeting their SLAsis important. The default Map-Reduce implementation does not have SLA as a input/conditionto be met. The default scheduling algorithm therefore does not consider these while schedulingjobs. So few SLA-based scheduling algorithms have been proposed for scenarios.One such algorithm is the constraint based hadoop scheduling algorithm proposed by Kamalet. al. in [4]. This algorithm maintains the minimum number of map/reduce tasks,provided by the job execution cost model, in the cluster to meet the SLA. The job cost executioncost model determines the minimum number of map/reduce tasks required to meet deadlines

L.TASK LEVEL

The third level of hierarchical scheduling is to select a task for scheduling on the available node.MR programming model creates two types of tasks map and reduce which needs to scheduled inrespective slots. One more type of task created by MR during runtime is speculative task tohandle straggler map/reduce tasks. These needs to be scheduled on scheduled in a map or reduceslot depending on the type of task it is duplicating. Different scheduling algorithms have beenproposed for selecting a task for the given slot type available to achieve different objectives likedata locality, performance improvement and improve resource utilization. The section describessome of the scheduling algorithms proposed for each of the task type.

M. MAP-TASK LEVEL

The default MR scheduling algorithm uses data locality criteria to select a map task for a given node. Initially JobTracker assigns the map tasks to the slaves depending on the TaskTrackerslots capacity, considering data locality. At run time, when a TaskTrackerreports an empty slot to process map task, the JobTracker checks for map tasks in its pool and consults the storage meta-data service to get the hosted chunks. It selects a map task for this node in following order of preference (1) a map task that has local chunk on that node, else (2) a map task that has data on another node within the same rack, else (3) a map task that has data on another node outside the rack. The map tasks pool has three kinds of map tasks: failed map tasks which have highest priority, normal map tasks and speculative map tasks with lowest priority.

N.REDUCE-TASK LEVEL

Once even a single map task of a job is completed, Hadoop MR starts scheduling its reduce job so that the map task execution and shuffling of intermediate data can run in parallel to have better turn-around time. This is known as early-shuffle. Hadoop MR randomly selects the reduce the task of the selected job for scheduling on the available reduce slot. Very few improvements have been suggested in literature for reduce task scheduling.

O.SPECULATIVE-TASK LEVEL

In Hadoop , if a node is available but is performing poorly, a condition that we call a straggler, MapReduce runs a speculative copy of its task (also called a backup task) on another machine to finish the computation faster. The goal of speculative execution is to minimize a jobs response time. A speculative task is run based on a simple heuristic comparing each tasks progress to the average progress. Hadoop monitors task progress using a parameter called Progress Score which has value between 0 and 1. For a map, the Progress Score is the fraction of input data read. For a reduce task, the execution is divided into three phases, each of which accounts for 1/3 of the score. Hadoop looks at the average Progress Score of each category of tasks (maps and reduces) to define a threshold for speculative execution. When a tasks Progress Score is less than the average for its category minus 0.2, and the task has run for at least one minute, it is marked as a straggler.

The speculative task scheduling in Hadoop is based on multiple assumptions, one is that data center is homogeneous, all tasks progress at same rate (while some may be local, some remote, some more compute intensive etc), and all reduce tasks process same amount of data. So if any of these is invalidated, their execution can cause competition and may cause Hadoop to perform poorly. Thus scheduling of speculative tasks which actually help minimize delay is complex. First because it is difficult to select the task for which to run speculative task as it would be difficult to distinguish between nodes that are slightly slower than the mean and stragglers especially in heterogeneous environment. Then it is useful in decreasing response time only if stragglers are identified as early as possible, so it needs to scheduled at right time. Few other points that need to be considered while deciding and scheduling them would be the competition of resources (network, cpuetc) created by speculative (nothing but duplicate) tasks and selecting node to run them.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

IV. COMPARISON OF SCHEDULING ALGORITHMS

The job and task scheduling for different users in a shared Hadoop environment is done hierarchically for meeting objectives like reducing the latencies, meeting SLAs, improving data locality and improving resource utilizations. To meet these requirements, many improvisations/changes have been proposed for the default Hadoop scheduling techniques. As different algorithms work towards different goals and work at different levels, they are not compared for efficiency related metrics. The algorithms studied here are compared based on the objectives targeted, the type of tasks scheduled and the key factors considered. The Table 1 below shows objectives considered in the algorithms in their priority order. The type of task - Map, reduce or speculative, whosescheduling is improved by the give n algorithms. The heterogeneous column is used to show if the algorithms consider the presence of heterogeneous no des in cluster or not. Analysis scheduling algorithms based on Taxonomy, Idea to implementation, advantages and disadvantages are listed in the Table 2 below

Scheduling Algorithm	Mini-mizing latency	Meetin g SLA	Maxi-mizing Data Locality	Maxi-mizing Resource Utilization	Fairness	Task Types (Map (M)/ Reduce(R)/ Speculative(S))	Heterogeneou s Nodes
LATE [6]	1					S	Yes
Deadline-based [11]		1			2	M	
Delay [6]			1		2	M	NA
Constraint-based [4]		1			2	M, R	Yes
CoGRS [7]	2		1			R	
BASE [10]	1			2		S	Yes
Maestro [12]			1	2		M	Yes

Table 1: Comparison of Scheduling Algorithms based on the criteria used by them

Scheduling Algorithm	Taxonomy	Idea to Implementation	Advantages	Dis-advantages
FIFO	Non-adaptive	Schedule jobs based on their priorities in first-come first- out.	1. Cost of entire cluster scheduling process is less. 2. Simple to implement and efficient.	1. Designed only for single type of job. 2. Low performance when run multiple types of jobs. 3. Poor response times for short jobs compared to large jobs.
Fair Scheduling	adaptive	Do a equal distribution of compute resources among the users/jobs in the system.	1. Less complex. 2. Works well when both small and large clusters. 3. it can provide fast response times for small jobs mixed with larger jobs.	1. Does not consider the job weight of each node.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 7, October 2015

Capacity	adaptive	Maximization the resource utilization and throughput in multi-tenant cluster environment.	1. ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues over large cluster .	1. The most complex among three schedulers.
delay scheduling	adaptive	To address the conflict between locality and fairness.	1. Simplicity of scheduling	1. No particular
context-aware scheduler	adaptive	To optimizations for jobs using the same dataset	1. Optimizations for jobs using the same dataset.	-

Table 2: Analysis of scheduling algorithm based of Taxonomy, advantages and disadvantages

V. CONCLUSION

In this paper described the overview of Hadoop MapReduce and their scheduling issues. Various categories of scheduling algorithms based on taxonomy, implementation are explained. Advantages and disadvantages pf selected algorithms are also listed. The Hadoop MR has a centralized global scheduler which needs to track all the jobs and tasks running in the cluster. As the workload intensity and size of the clusters are increasing, the single job scheduler may get overwhelmed with the volume of work it has to do. As interactive, short response time jobs are being increasingly used, the scheduler needs to make quicker decisions at least before a small task gets completed. Also to achieve different objectives of decreasing task times, increasing data locality, improving resource utilization, meeting SLAs of different jobs etc. complex decision making algorithms need to be implemented. From the discussions we can see that most of these schedulers address one or more problem. And choice of this scheduler for a particular job is up to the user. There is no one algorithm that fits all situations.

REFERENCES

[1] J Jeffery Hanson. An introduction to the Hadoop Distributed File System. IBM DeveloperWorks, 2011. http://www.ibm.com/developerworks/web/library/waintrohdfs /index.html
[2] The Apache Hadoop Project. Reference Manual http://www.hadoop.org.
[3] MateiZaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, Ion Stoica. Improving MapReduce performance in heterogeneous environments. In *Proceedingsof the 8th USENIX conference on Operating systems design and implementation*, p.29-42, December, 2008.
[4] Kamal Kc and KemaforAnyanwu. Scheduling hadoop jobs to meet deadlines. In *Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science*. CLOUDCOM '10. pages 388-392. IEEE Computer Society, 2010.
[5] M. Chowdhury, Y. Zhong, and I. Stoica. Efficient Coflow Scheduling with Varys. In ACM SIGCOMM, 2014.
[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," (2008) in 8th Usenix Symposium on Operating Systems Design and Implementation, (New York), pp. 29–42, ACM Press.
[7]M. Hammoud, M. Rehman and M. Sakr, " Center-of-Gravity reduce task scheduling to lower MapReduce network traffic ", In: International conference on cloud computing. IEEE, 2012, p. 49–58.
[8] Hadoop's Fair Scheduler. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler
[9] Jord'a Polo, David de Nadal, David Carrera, Yolanda Becerra, Vicenc Beltran, Jordi Torres and Eduard Ayguade. Adaptive Task Scheduling for MultiJobMapReduce Environments. In *Proceedings of Jornadas de Paralelismo conference (JP)*, pp 96-101A Corua. 2009.
[10] ZhenhuaGuo, Geoffrey Fox, Mo Zhou, Yang Ruan. Improving Resource Utilization in MapReduce. *Indiana University Report*. May 2012.
[11] Jord'a Polo, David de Nadal, David Carrera, Yolanda Becerra, Vicenc Beltran, Jordi Torres and Eduard Ayguade. Adaptive Task Scheduling for MultiJobMapReduce Environments. In *Proceedings of Jornadas de Paralelismo conference (JP)*, pp 96-101A Corua. 2009.
[12] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu and S. Wu, " Maestro: replica-aware map scheduling for MapReduce", In: The 12th international symposium on cluster, cloud and grid computing. IEEE/ACM; 2012, p. 435– 477. [28] L. Lei