ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

Recent Job Scheduling Algorithms in Hadoop Cluster Environments: A Survey

Mr.A.U.Patil[1], Mr T.I Bagban[2], Mr.A.P.Pande[3]

M.E, CSE, ADCET, Sangli, India [1]

Asso.Prof, CSE, DKTE, Sangli, India [2]

Asst.Prof, IT, PVPIT, Sangli, India [3]

Abstract: This paper discusses Cloud Computing is rising as a replacement machine paradigm shift. Hadoop-MapReduce has become a robust Computation Model for process giant knowledge on distributed commodity hardware clusters like Clouds. Studies describe that Hadoop implementations, the default first in first out scheduler hardware is accessible wherever jobs are scheduled in FIFO order with support for different priority primarily based schedulers also other defaulter schedulers . During this paper we have studied numerous scheduling enhancements in a scheduling techniques a brand new with Hadoop like Fair4s scheduling algorithm with its extended functionalities allows processing large as well small jobs with effective fairness without starvation of small jobs. Majority of small jobs are available till date so Job scheduling algorithm must take in consideration of small jobs first with effective major to process these jobs.

Keywords: Cloud Computing, Hadoop, HDFS, MapReduce, Schedulers

I. INTRODUCTION

Cloud computing advised as an apace emerging new example for delivering computing as a quality. In cloud computing different cloud consumers claim variety of services as per their dynamically dynamic needs. So it is the job of cloud computing to work all the demanded services to the cloud consumers. But due to the availability of impermanent resources it is real hard for cloud providers to give all the demanded services. From the cloud providers' appearance cloud resources staleness be allocated in a moderate kind. So, it's an animated takings to correspond cloud consumers' QoS requirements and satisfaction. In visit to assure on-demand availability a businessperson needs to overprovision: dungeon a sizeable arrangement of nodes lackadaisical so that they can be used to supply an on-demand sustain idle leads to low utilization. The exclusive way to modify it is to rest few nodes otiose. But this agency potentially rejecting a higher quotient of requests to a convexity at which a bourgeois no mortal provides on-demand computing [2].

Several trends are starting up the era of Cloud Computing, which is a Cyberspace supported evolution and use of computer technology. Cheaper and much almighty processors, together with the "software as a service" (SaaS) computing structure, are transforming data canters into pools of computing service on a huge standard. Meanwhile, the flaring cloth bandwidth and tried yet pliable web connections work it flatbottom possible that clients can now buy eminent wellborn services from data and software that shack solely on removed data centres.

In the past years, Infrastructure-as-a-Service (IaaS) cloud computing has emerged as an enthralling choice to the acquisition and management of somatic resources. A key asset of Infrastructure-as-a-Service (IaaS) clouds is providing users on-demand right to resources. Yet, to provide on-demand right, cloud providers staleness either significantly overprovision their stock (and pay a shrill damage for operating resources with low utilization) or respond an outsized counterbalance of somebody requests (in which human the way is no human on-demand). At the very time, not all users expect genuinely on-demand hit to resources [3]. Some applications and workflows are premeditated for redeemable systems where interruptions in service are

II. HADOOP

In the recent period, Infrastructure-as-a-Service (IaaS) cloud computing has emerged as a prepossessing deciding to the acquisition and management of corporeal resources. A key welfare of Infrastructure-as-a-Service (IaaS) clouds is providing users on-demand make to resources.

Nevertheless, to render on-demand right, cloud providers must either significantly overprovision their store (or pay an advanced soprano for operative resources with low utilization) or scorn a deep rescale of somebody requests (in which person the admittance is no thirstier on-demand). At the comparable instance, not all users demand really on-demand make to resources [3].

Many applications and workflows are intentional for recoverable systems where interruptions in service are of tasks that can be executed concurrently ona node. Each slot of the node at any time is onlycapable of executing one task. In MapReduce, thereare two types of slot: map slot, and reduce slot.Scheduling decisions are taken by a master node, called the JobTracker, and the worker nodes that called TaskTracker execute the tasks.

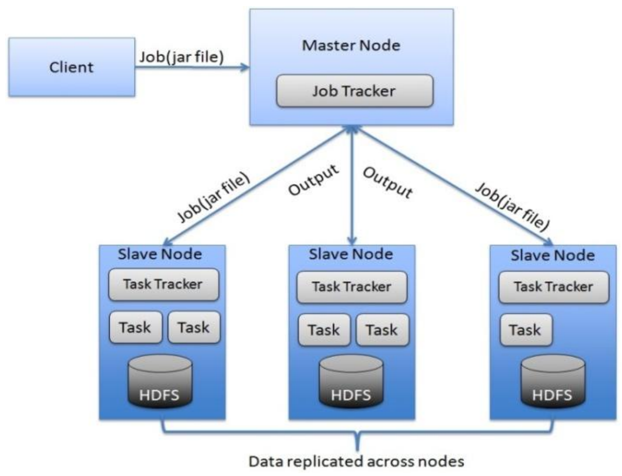

Fig.1 Hadoop Architecture

A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job tracker, Task tracker, Name node and Data node.

A. Job tracker

The primary function of the job tracker is managing the task trackers, tracking resource availability. The Job tracker is a node which controls the job execution process. Job tracker performs mapreduce tasks to specific nodes in the cluster. Client submit jobs to the Job tracker. When the work is completed, the Job tracker updates its status. Client applications can ask the Job tracker for information.

B. Task tracker

It follows the orders of the job tracker and updating the job tracker with its status periodically. Task tracker run tasks and send the reports to Job tracker, which keeps a complete record of each job. Every Task tracker is configured with a set of slots, it indicates the number of tasks that it can accept.

C. Name node

The namenode maps to, what block locations and which blocks are stored on which datanode. Whenever a datanode undergoes a disk corruption of a particular block, the first table gets updated and whenever a datanode is detected to be dead due to network failure or a node, both the tables get updated. Theupdating of the table is based on only failure of the nodes. It does not depends on any neighbour blocks or any block locations to identify its destination. Each blocks are separated with its job nodes and respective allocated process.

D. Data node

The node which stores the data in hadoop system are known to be as datanode. All datanodes send a heartbeat message to the namenode for every three seconds to say that they are alive. If the namenode does not receive a heartbeat from a particular data node for ten minutes, then it considers that data node to be dead or out of service .It initiates some other data node for the process. The data nodes update the namenode with the block information periodically.

2.1 HDFS- Distributed file system:-

DFS was designed to be a scalable, fault-tolerant, distributed storage system that works closely with MapReduce. HDFS will "just work" under a variety of physical and systemic circumstances. By distributing storage and computation across many servers, the combined storage resource can grow with demand while remaining economical at every size.

These specific features ensure that the Hadoop clusters are highly functional and highly available:

Rack awareness- Allows consideration of a node's physical location, when allocating storage and scheduling tasks

Minimal data motion- MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and keeps most of the I/O on the local disk or within the same rack and provides very high aggregate read/write bandwidth.

Utilities- diagnose the health of the files system and can rebalance the data on different nodes

Rollback- allows system operators to bring back the previous version of HDFS after an upgrade, in case of human or system errors

Standby NameNode- provides redundancy and supports high availability

Highly operable-Hadoop handles different types of cluster that might otherwise require operator intervention. This design allows a single operator to maintain a cluster of 1000s of nodes.

2.2 HADOOP MAP REDUCE OVERVIEW

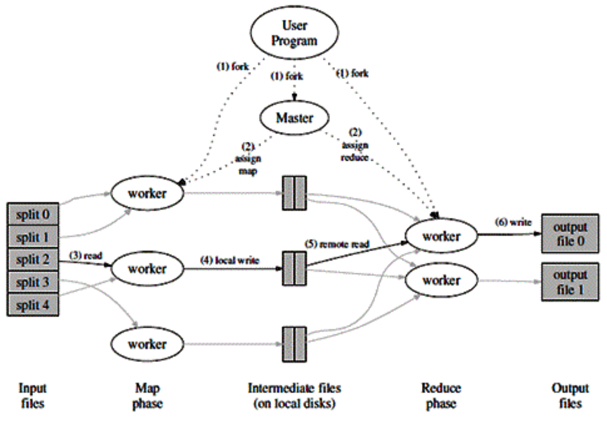

Fig.1 Map Reduce Overview

Map-Reduce is a programming representation and a related effort for processing and generating sizeable datasets that is amenable to a large show of real-world tasks. Users delimitate the computation in cost of a map and a throttle purpose also Users enlarge a map suffice that processes a key/value arrange to generate a set of intermediate key/value pairs, and a become role that merges all sophomore values associated with the aforementioned middle key. Programs typed in this functional style are automatically parallelized and executed on a bouffant of the details of partitioning the signal data, programing the schedule's implementation crossways a set of machines, management machine failures, and managing the required inter-machine connexion. This allows programmers without any have with comparable and straggly systems to easily employ the resources of a generous spaced system [7].

III. SCHEDULING IN HADOOP

The default Scheduling algorithm is supported on FIFO where jobs were executed in the magnitude of their humility. Later on the cognition to set the priority of a Job was added. Facebook and Character contributed meaningful apply in processing schedulers i.e. Legible Scheduler [7] and Capacity Scheduler [8] respectively which after free to Hadoop Dominion.

3.1 Default FIFO Scheduler

The default Hadoop scheduler operates using a FIFO queue. After a job is divided into independent tasks, they are ended into the queue and allotted to free slots as they get acquirable on TaskTracker nodes. Although there is keep for decision of priorities to jobs, this is not revolved on by default. Typically apiece job would use the complete assemble, so jobs had to inactivity for their release. Regularize though a distributed constellate offers zealous latent for offering larger resources to numerous users, the job of intercourse resources evenhandedly between users requires a turn scheduler. Production jobs bet in a rational indication.

3.2 FairScheduler

The Fair Scheduler [7] was developed at Facebook to manage access to their Hadoop cluster and subsequently released to the Hadoop community. The Fair Scheduler aims to give every user a fair share of the cluster capacity over time. Users may assign jobs to pools, with each pool allocated a guaranteed minimum number of Map and Reduce slots. Free slots in ineffective pools may be allocated to new pools, piece immoderateness capacity within a pool is joint among jobs. The Fair Scheduler supports preemption, so if a pool has not received its fair deal for a fated period of quantify, then the scheduler module veto tasks in pools flowing over capacity in dictate to afford the slots to the pool functional under capacity. In addition, administrators may enforce priority settings on doomed pools. Tasks are therefore scheduled in an interleaved fashion, supported on their priority within their pool, and the constellate capacity and activity of their pool. As jobs have their tasks allocated to Task Tracker slots for computation, the scheduler tracks the shortfall between the become of measure actually old and the saint fair percentage for that job. As slots transmute unconfined interval. Over time, this has the effect of ensuring that jobs receive roughly equal amounts of resources. Shorter jobs are allocated sufficient resources to finish quickly. At the same time, longer jobs are guaranteed to not be starved of resources.

3.3 Capacity Scheduler

Capacity Scheduler [3] originally developed at Yahoo addresses a usage scenario where the number of users is large, and there is a need to ensure a fair allocation of computation resources amongst users. The Capacity Scheduler allocates jobs supported on the submitting user to queues with configurable drawing of Map and Minify slots. Queues that hold jobs are bestowed their organized capacity, patch atrip capacity in a queue is shared among opposite queues. Within a queue, planning operates on a modified priority queue groundwork with specialized person limits, with priorities orientated supported on the quantify a job was submitted, and the priority scene allocated to that human and accumulation of job. When a Task Tracker receptacle becomes unfixed, the queue with the lowest laden is elite, from which the oldest remaining job is chosen. A task is then scheduled from that job. This has the validity of enforcing meet capacity distribution among users, rather than among jobs, as was the case in the Fair Scheduler.

IV. SCHEDULER IMPROVEMENTS

Many researchers are working on opportunities for improving the scheduling policies in Hadoop. Recent efforts such as Delay Scheduler [9], Dynamic Proportional Scheduler [10] offer differentiated service for Hadoop jobs allowing users to adjust the priority levels assigned to their jobs. However, this does not guarantee that the job will be completed by a specific deadline. Deadline Constraint Scheduler [11] addresses the issue of deadlines but focuses more on increasing system utilization. The Schedulers described above attempt to allocate capacity fairly among users and jobs, they make no attempt to consider resource availability on a more fine-grained basis. Resource Aware Scheduler [12] considers the resource availability to schedule jobs. In the following sections we compare and contrast the work done by the researchers on various Schedulers.

4.1 Longest Approximate Time to End (LATE) - Speculative Execution

It is not uncommon for a particular task to continue to progress slowly. This may be due to several reasons like– high CPU load on the node, slow background processes etc. All tasks should be finished for completion of the intact job. The scheduler tries to discover a dilatory gushing task to displace added equal task as a part which is termed as theoretical execution of tasks. If the duplicate make completes faster, the job execution is improved. Speculative executing is an improvement but not a attribute to insure reliability of jobs. If bugs effort a task to fix or diminish downed then curious execution is not a set, since the aforementioned bugs are possible to relate the theoretical task also. Bugs should be unchangeable so that the task doesn't fall or decrease pile computation at all

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

nodes. That is, default effort of speculative action entirety vessel on homogenous clusters. These assumptions reclaim downcast very easily in the sundry clusters that are recovered in real-world creation scenarios. Zaharia et al [13] proposed a modified version of wondering process titled Longest Close Reading to End (LATE) algorithm that uses a different measure to schedule tasks for speculative execution. Instead of considering the develop prefab by a task so far, they compute the estimated clip remaining, which gives a much shiny improvements by Longest Inexact Instant to End (Tardy) algorithm over the default speculative execution.

4.2 **Delay Scheduling**

Fair scheduler is developed to allocate fair share of capacity to all the users. Two locality problems identified when fair intercourse is followed are - head-of-line scheduling and sticky slots. The front neighborhood difficulty occurs in petite jobs (jobs that someone gnomish signal files and thence hold a minuscule wares of data blocks to have). The problem is that whenever a job reaches the progress of the classified table for scheduling, one of its tasks is launched on the next receptacle that becomes issue irrespective of which node this slot is on. If the head-of-line job is little, it is unlikely. Head-of-line scheduling problem was observed at Facebook in a version of HFS without delay scheduling. The other locality problem, sticky slots, is that there is a tendency for a job to be assigned the same slot repeatedly. The problems aroused because following a strict queuing order forces a job with no local data to be scheduled.
To overcome the Head of line problem, scheduler launches a task from a job on a node without localized data to record justness, but violates the primary clinical of MapReduce that schedule tasks moral their signal data. Spouting on a node that contains the data (node neighborhood) is most efficacious, but when this is not attemptable, jetting on a node on the similar wipeout (demolition neighborhood) is faster than functional off-rack. Retard scheduling is a statement that temporarily relaxes impartiality to alter section by asking jobs to act for a programming possibleness on a node with an aesthetic data. When a node requests a task, if the head-of-line job cannot commence a localized task, it is skipped and looked at later jobs. Still, if a job has been skipped extended enough, non-local tasks are allowed to displace to avoid starvation. The key brainstorm behindhand {giving to a job is remote to bang data for it, tasks finish so quick that few slot with data for it testament unloose up in the next few seconds.

4.3 **Deadline Constraint Scheduler**

Deadline Constraint Scheduler [11] addresses the issue of deadlines but focuses more on increasing system utilization. Dealing with deadline requirements in Hadoop-based data processing is done by (1) a job execution cost model that considers various parameters like map and reduce runtimes, input data sizes, data distribution, etc., (2) a Constraint-Based Hadoop Scheduler that takes user deadlines as part of its input. Estimation model determines the available slot based a set of assumptions: All nodes are homogeneous nodes and unit cost of processing for each map or reduce node is equal Input data is distributed uniform manner such that each reduce node gets equal amount of reduce data to process Reduce tasks starts after all map tasks have completed; The input data is already available in HDFS. Schedulability of a job is determined based on the proposed job execution cost model independent of the number of jobs running in the cluster. Jobs are only scheduled if specified deadlines can be met. After a job is submitted, schedulability test is performed to determine whether the job can be finished within the specified deadline or not. Free slots availability is computed at the given time or in the future irrespective of all the jobs running in the system. The job is enlisted for scheduling after it is determined that the job can be completed within the given deadline. A job is schedulable if the minimum number of tasks for both map and reduce is less than or equal to the available slots. This Scheduler shows that when a deadline for job is different, then the scheduler assigns different number of tasks to TaskTracker and makes sure that the specified deadline is met.

4.4 **Resource Aware Scheduling**

The Fair Scheduler [7] and Capacity Scheduler described above attempt to allocate capacity fairly among users and jobs without considering resource availability on a more fine-grained basis. As CPU and disk channel capacity has been increasing in recent years, a Hadoop cluster with heterogeneous nodes could exhibit significant diversity in processing power and disk access speed among nodes. Performance could be affected if multiple processor-intensive or data-intensive tasks are allocated onto nodes with slow processors or disk channels respectively. This possibility arises as the Job Tracker simply treats each Task Tracker node as having a number of available task “slots”. Even the improved LATE speculative execution could end up increasing the degree of congestion within a busy cluster, if speculative copies are simply assigned to machines that are already close to maximum resource utilization.
Resource Aware Programming in Hadoop has prettify one of the Explore Challenges [14] [15] in Cloud Computing. Programming in Hadoop is centralized, and initiated. Planning decisions are confiscated by a combatant node, called the JobTracker, whereas the miss nodes, called TaskTrackers are obligated for task execution. The JobTracker maintains a queue of currently squirting jobs, states of TaskTrackers in a flock, and name of tasks allocated to each TaskTracker. Apiece Task Tracker node is currently configured with anextreme company of open computation slots. Though this can be configured on a per-node basis to emit the very processing cause. lendable on cluster machines, there is no online adjustment of this receptacle capacity acquirable. That is, there is no way to cut crowding on a machine by publicizing a low capacity. In this performance, apiece Task Tracker node monitors resources specified as CPU utilization, dimension for the retention subsystem. We previse that else metrics module examine functional, we propose these as the standard iii

resources that must be tracked at all times to modify the wattage equalization on constellate machines. In specific, plate marketing burden can significantly alter the data weight and composition share of Map and Restrain tasks, many so than the amount of uncommitted character open. Likewise, the underlying opacity of a machine's virtual store management state means that monitoring author faults and virtual memory-induced disk liberal storage.

5.5 **Fair4s Job Scheduling**

Fair4S, which is designed to be biased towards small jobs. Small jobs account for the majority of the workload, and most of them require instant and interactive responses, which is an important phenomenon at production Hadoop systems. The inefficiency of Hadoop fair scheduler and GFS read write algorithm for handling small jobs motivates us to use and analyze Fair4S, which introduces pool weights and extends job priorities to guarantee the rapid responses for small jobs[1]our implementation of Fair4s Job Scheduling Algorithm runs on a large cluster of commodity machines and is highly scalable. Map-Reduce is Popularized by open-source Hadoop project. Our Fair4s Job Scheduling Algorithm works on processing of large files by dividing them on number of chunks and assigning the tasks to the cluster nodes in hadoop multimode configuration. In these ways our proposed Fair4s Job Scheduling Algorithm improves the Utilization of the Cluster nodes in terms of Time, CPU, and storage.

5.5.1 **Extended functionalities**

Extended functionalities available in Fair4s scheduling algorithm make it workload efficient than GFS read write algorithm are listed out below these functionalities allows algorithm to gives out efficient performance in processing huge work load from different clients.

1. Setting Slots Quota for Pools:- All jobs are divided into several pools. Each job belongs to one of these pools. While in Fair4S, each pool is configured with a maximum slot occupancy. All jobs belonging to an identical pool share the slots quota, and the number of slots used by these jobs at a time is limited to the maximum slots occupancy of their pool. The slot occupancy upper limit of user groups makes the slots assignment more flexible and adjustable, and ensures the slots occupancy isolation across different user groups. Even if some slots are occupied by some large jobs, the influence is only limited to the local pool inside.

2. Setting Slot Quota for Individual Users:-In Fair4S, each user is configured with a maximum slots occupance. Given a user, no matter how many jobs he/she submits, the total number of occupied slots will not exceed the quota. This constraint on individual user avoids that a user submit too many jobs and these jobs occupy too many slots.

3. Assigning Slots based on Pool Weight:-Fair4S, each pool is configured with a weight. All pools which wait for more slots form a queue of pools. Given a pool, the occurrence times in the queue is linear to the weight of the pool. Therefore, a pool with a high weight will be allocated with more slots. As the pool weight is configurable, the pool weight-based slot assignment policy decreases small jobs' waiting time (for slots) effectively
.

4. Extending Job Priorities:- Fair4S introduces an extensive and quantified priority for each job. The job priority is described by an integral number ranged from 0 to 1000. Generally, within a pool, a job with a higher priority can preempt the slots used by another job with a lower priority. A quantified job priority contributes to differentiate the priorities of small jobs in different user-groups.

5.5.2 **Procedure of Slots Allocation:-**

1. The first step is to allocate slots to job pools. Each job pool is configured with two parameters of maximum slots quota and pool weight. In any case, the count of slots allocated to a job pool would not exceed its maximum slots quota. If slots requirement for one job pool varies, the maximum slots quota can be manually adjusted by Hadoop operators. If a job pool requests more slots, the scheduler firstly judges whether the slots occupance of the pool will exceed the quota. If not, the pool will be appended with the queue and wait for slot allocation. The scheduler allocates the slots by round-robin algorithm. Probabilistically, a pool with high allocation weight will be more likely to be allocated with slots.

2. The second step is to allocate slots to individual jobs. Each job is configured with a parameter of job priority, which is a value between 0 and 1000. The job priority and deficit are normalized and combined into a weight of the job. Within a job pool, idle slots are allocated to the jobs with the highest weight.

V. CONCLUSION

Hadoop is in huge demand in the market now a days. As there huge amount of data is lying in the industry but there is not tool to handle it and hadoop can implemented on low cost hardware and can be used by large set of audience on large number of dataset.Jobs areavailable in large jobs and Small jobs. In hadoop map reduce is the most important component in hadoop. In this paper we have studied default hadoop schedulers for processing the tasks and taken into consideration their drawbacks and studied latest among available schedulers like LATE, Resource Aware scheduling, Delay Scheduling, Fai4s scheduling scheme is the best among them for processing the large as well as fair for processing small tasks with its extended functionalities.

ACKNOWLEDGMENT

I am thankful to Mr.T.I.Bagban for hisguidance, support and keen supervision. And also for providing the information and ready to help any time in completion of this survey. I would like to express my gratitude towards my family and colleges for their co-operation and help me in completing this survey

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

REFERENCES

[1] ZujieRen, Jian Wan *"Workload Analysis, Implications, and Optimization on a Production Hadoop Cluster:A Case Study on Taobao",*CO IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 7, NO. 2, APRIL-JUNE 2014.
[2] M. Zaharia, D. Borthakur, J.S. Sarma, S. Shenker, and I. Stoica,''Job Scheduling for Multi-User Mapreduce Clusters,'' Univ.California, Berkeley, CA, USA, Tech. Rep. No. UCB/EECS-2009-55, Apr. 2009.
[3] Y. Chen, S. Alspaugh, and R.H. Katz, ''Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of Mapreduce Workloads,'' Proc. VLDB Endowment, vol. 5, no. 12, Aug. 2012
[4] Aspera an IBM company(2014,07,14). Big Data Cloud[English].Available:http://cloud.asperasoft.com/big-data-cloud/.
[5] Divyakant Agrawal et al., *" Big Data and Cloud Computing: Current State and Future Opportunities"* , EDBT, pp 22-24, March 2011.
[6] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, ''Workload Characterization on a Production Hadoop Cluster: A Case Study on Taobao,'' in Proc. IEEE IISWC, 2012, pp. 3-13.
[7] Hadoop Fair Scheduler http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html
[8] Hadoop's Capacity Scheduler: http://hadoop.apache.org/core/docs/current/capacity_scheduler.html
[9] Stackoverflow(2014,07,14)."Hadoop Architecture Internals: use of job and task trackers"[English].Available:http://stackoverflow.com/questions/11263187/hadoop architecture-internals-use-of-job-and-task-trackers
[10] J.J. More, ''TheLevenberg-Marquardt Algorithm: Implementation and Theory Dundee Conference on Numerical Analysis. New York, NY, USA: Springer-Verlag, 1978.
[11] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, ''An Analysis of Traces from a Production Mapreduce Cluster,'' in Proc. CCGRID, 2010, pp. 94-103. [12] J. Dean et al.,*"MapReduce: a flexible data processing tool",* In CACM, Jan 2010.
[13] M. Stonebraker et al., *"MapReduce and parallel DBMSs: friends or foes?",* In CACM. Jan 2010.
[14] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, ''Implementing WebGIS on Hadoop: A Case Study of Improving Small File I/O Performance on HDFS,'' in Proc. CLUSTER, 2009, pp. 1-8.
[15] A.Abouzeid et al., *"HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads",* In VLDB 2009.
[16] F. N. Afrati et al.,*"Optimizing joins in a map-reduce environment",*In EDBT 2010.
[17] P. Agrawal et al., *"Asynchronous view maintenance for VLSD databases",* In SIGMOD 2009.
[18] S. Das et al., *"Ricardo: Integrating R and Hadoop",* In SIGMOD 2010.
[19] J. Cohen et al.,*"MAD Skills: New Analysis Practices for Big Data",* In VLDB, 2009.
[20] Gaizhen Yang et al., *"The Application of SaaS-Based Cloud Computing in the University Research and Teaching Platform",* ISIE, pp. 210-213, 2011.
[21] Paul Marshall et al., *"Improving Utilization of Infrastructure Clouds",*IEEE/ACM International Symposium, pp. 205-2014, 2011.
[22] F. Wang, Q. Xin, B. Hong, S.A. Brandt, E.L. Miller, D.D.E. Long, and T.T. Mclarty, ''File System Workload Analysis for Large Scale Scientific Computing Applications,'' in Proc. MSST, 2004,
[23] pp. 139-152. [23] M. Zaharia, D. Borthakur, J.S. Sarma, K. Elmeleegy, S. Shenker, andI. Stoica, ''Delay Scheduling: A Simple Technique for AchievingLocality and Fairness in Cluster Scheduling,'' in Proc. EuroSys, 2010, pp. 265-278..
[24] E. Medernach, ''Workload Analysis of a Cluster in a Grid Environment,'' in Proc. Job Scheduling Strategies Parallel Process., 2005, pp. 36-61
[25] K. Christodoulopoulos, V. Gkamas, and E.A. Varvarigos, *''Statistical Analysis and Modeling of Jobs in a Grid Environment,''* J. Grid Comput., vol. 6, no. 1, 2008.
[26] E. Medernach, *''Workload Analysis of a Cluster in a Grid Environment,'' in Proc. Job Scheduling Strategies Parallel Process.*,2005, pp. 36-61
[27] B. Song, C. Ernemann, and R. Yahyapour, *''User Goup-Based Workload Analysis and Modelling,''* in Proc. CCGRID, 2005, pp. 953-961
[20] Gaizhen Yang et al., "The Application of SaaS-Based Cloud Computing in the University Research and Teaching Platform", ISIE, pp. 210-213, 2011.