IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Allocation of Phase-Based Scheduler for MapReduce Job Scheduling

Suryakant S. Bhalke

Department of Computer Engineering, JSPM's Imperial College of Engineering & Research, Wagholi, Pune, India

Abstract: Hadoop MapReduce is effective user interface design classic for large scale data handling. MapReduce has two levels: Task-level and Phase level. In existing system, that focuses on scheduling at task level which tasks can have changing resource requirements. There are some difficult to efficiently apply accessible resources to reduce job implementation time. To report this limitation, this project proposes a Phase-Based Scheduler. Map Reduce which allocates resource information about status of every Phase the phase-based to executed job scheduling. The job scheduling of phase based is executed by the Master Node, which handle & service lots of list of jobs in the system. Each Node Manager (slave node) from time to time getting a heartbeat message to the scheduler. Getting the status message from a Node Manager running on machine, the scheduler divides the use for fixed of phases for the tasks using the jobs phase-based resource requirement. This improves to reduce job implementation time. This is achieving high job performance and resource utilization.

Keywords: Big Data, Hadoop, Scheduler, MapReduce, Phase-Based Scheduler

I. INTRODUCTION

Hadoop is an open source under the Apache fund account component, and is an open source application of Google graphs calculation model. It can easily develop and run significant data processing. Two of the most essential part are HDFS (Hadoop Distributed File System) and Map Reduce.

A. HDFS

The Hadoop distributed file system (HDFS) to store large files with streaming data access patterns, to run with managers-workers mode, that is, there is NR Me Node (managers) and multiple Data Nodes (workers). Node manages the file system tree and the tree in all of the files and directories. Data Node is usually a Node in the cluster, a record of every file in every block of Data Node information.

B. MapReduce

MapReduce work process is allocated into two phases. A map function, which is used to put a set of keys for mapping into a new set of key-value pairs. And it points to the Reduce function. MapReduce has four parts: the framework of homework submission and initialization, task allocation, task implementation and completion of the homework. Job Client submits a job, and Job Tracker will get the job of the information will be sent. Job Tracker is the canter of the MapReduce formed, which needs to interconnect with the cluster machine timing (heartbeat), and need to achieve what programs should be run on which machines, to achieve job failed, start again operation. Task Tracker is a measure of every machine in MapReduce. It is considered to following resources of their machines. Task Tracker observing tasks run of the current state of the machine. Task Tracker needs getting s the information through the heartbeat Job Tracker. Job Tracker will collect these information to assign new job submitted a run on which machines.

C. The framework of Hadoop YARN

The framework of Hadoop free services: a global RM and Application Master of every application. The RM is responsible for the resource management and allocation of the whole system, while Application Master responsible for the management of a single application. YARN resources on the Node Manager for unified management and scheduling. YARN is mainly part of the RM, Node Manager, AM and several Container mechanisms.

II. RELATED WORK

A. Existing System

In existing system, the scheduler executed at the task level. The original MapReduce work is to schedule the task in different levels. In a MapReduce method, the group of jobs and can be scheduled parallel on multiple machines, resulting in reduction in job running time. In mapper phase data blocks in HDFS and it maps, merge the data and stored in the many files. In other hand reducer phase will fetch data from mapper output and shuffle, sort the data in a serialized manner.

B. Literature Review

For instance J. Polo, C. Castillo, D. Carrera by [3] represents this literature existing The principles of RAS (Resource Adaptive Scheduler) are resource awareness and constant job executed management. RAS approach offers a unique resource aware scheduling technique several ways: Extends the abstraction of 'task slot' to 'job slot'. A 'job slot' is job specific, and has an associated resource demand profile for MapReduce tasks. Leverages

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

resource profiling information to get better utilization of resources and increase application performance. It familiarizes to changes in resource demand by dynamically allocating resources to jobs. Also subsequently M. Zaharia, D. Borthakur, J. [4] by as groups start to use data intensive cluster work out systems Hadoop developing need to share clusters among users. There is an encounter between fairness in scheduling and information locality. To getting report of the encounter among locality and fairness, this literature proposes when the job that should be arranged next allowing to fairness cannot takeoff a local task, it delays for a small remount of time, allowing other jobs launch tasks instead. Delay scheduling is applicable outside fair sharing. This scheduling only asks that it at times give resources to jobs out of order to increase data locality.

The generalization of delay scheduling in HFS to implement a classified scheduling policy. At the top level, HFS allocates task slots across pools using weighted fair sharing. M. Isard, V. Prabhakaran, J. Currey, U. Wieder, and K. Talwar. Proposed by [5] this literature reports the difficult of scheduling parallel jobs on clusters where application data is stored on the work out nodes.
In which scheduling working outs close to their data is central for executed, is more and more common and arises in systems such as MapReduce, Hadoop, and Dryad as well as many grid-work out environments. Problem of arranging with locality and fairness constraints has not before has widely planned under this typical of resource sharing.

A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker and I. Stoica by [6] Resource allocation is a key building block of any shared computer system. To report this problem, the system proposes Dominant Resource Fairness (DRF), a generalization of max-min fairness to multiple resource types. For every user, DRF work out the share of every resource allocated to that user. The maximum among all shares of a user is called that user's dominant share, and the resource matching to the dominant share is called the dominant resource. Different users may have different dominant resources.

DRF has the following properties: C. Joe-Wong, S. Sen, T. Lan, and M. Chiang by [7] This is significant data equivalent applications such as web indexing, data mining, and scientific simulation. To improve the execute during speculative implementation, this literature designs algorithm for speculative implementation that is strong to heterogeneity and highly effective in practice. The proposed algorithm called as LATE for Longest Approximate Time to End. Subsequently Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. Gunda, and J. Currey by [8] proposed. This literature is to exploit this key observation and explore a new, fine-grained network reservation abstraction called temporally-interleaved virtual clusters. By H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu [9] Starfish is a MADDER and analytics on big data. These automatically select efficient implementation techniques for MapReduce jobs.
A unique feature of the Sampler is that it can sample the implementation of a MapReduce job in order to enable the Profiler to collect approximate job profiles at a fraction of the full job implementation cost.

III. PROPOSED SYSTEM ARCHITECTURE

A. Resource Manager
RM is a global that is responsible for the resource management and allocation of the complete system. It is mainly made up of two components: the Scheduler (Scheduler) and the application Manager (Applications Manager, ASM);

B. Application Master
Every application contains 1 RM. There are the main features: Negotiate with GETTING scheduler for resources, Tasks within the task assigned to further, Interconnect with NM to start/stop the task, the Observer all tasks running state;

C. Node Manager
NM is on every node of resources and task manager. On the one hand, it will report regularly to the getting this node on the resource usage and the running state of every container. On the other hand, it receives and deals with the Container from RM start/stop and other requests;

D. Container
Container is resource abstraction of the YARN. It summarizes the multi-dimensional resources on a node, such as memory, CPU, disk, network and so on. Units new task, or allow a paused task to begin its next phase (e.g., the reduce phase), and then information the Node Manager about the scheduling decision.

Finally, once the task is allowed to execute the next phase, the Node Manager grants the getting permission to the task process. Once the task is finished, the task status is received by the Node Manager and then promoted to scheduler. A fine grained, phase-based that every task is at this time executing.

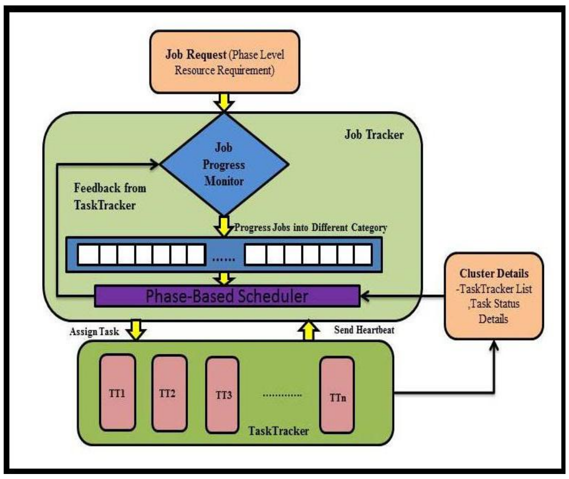
The job scheduling in phase-based is executed by the RM in the Master Node, which continues a list of jobs in the system. The phase-based scheduler will use the delivered info getting action to make scheduling decisions.

A task will scheduled, the scheduler responses to heartbeat message with a task scheduling request. The Node Manager then launches the task. Every time a task finishes executing a particular phase, the task asks the Node Manager for getting permission to start the next phase. The task of every phase is scheduled based on the utility of that phase. The scheduler allocates a utility value to every phase which indicates the benefit of scheduling the phase. The utility function is calculated based on the fairness and job performance of the particular phase.

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016


Job Request (Phase Level Resource Requirement)
Job Tracker
Job Progress Monitor
Feedback from TaskTracker
Progress Jobs into Different Category
......
Phase-Based Scheduler
Cluster Details -TaskTracker List ,Task Status Details
Assign Task
Send Heartbeat
TT1
TT2
TT3
....
TTn
TaskTracker
Fig.1.Proposed Phase-based Scheduler Architecture

IV. SYSTEM BLOCK DIAGRAM

Here present proposed Architecture, a Phase-based resource-aware scheduler that executed scheduling at phase-level. Unlike existing MapReduce schedulers that only allow job owners to specify resource requirements at task-level, proposed architecture allows the job owners to specify Phase-based resource requirements. An overview of the proposed architecture consists of three main components: a phase-based scheduler at the master node, local Node Managers that coordinate phase transitions with the scheduler, and a job progress monitor to capture phase-based progress information. The phase-based scheduling mechanism used by proposed block diagrams is illustrated by Fig.2. [2] Similar to the current Hadoop implementation, Node Manager from time to time getting Node Manager then takes-offs the task. Every time a task finishes executing a particular phase (e.g. shuffle phase of the reduce task), the task asks the Node Manager for a getting permission to start the preceding phase (e.g. reduce phase of the task). Next step the local Node Manager then forwards the getting permission request to the scheduler through the regular heartbeat message. Then given a job's phase-based resource requirements and its current progress info getting action, the scheduler decides whether to start

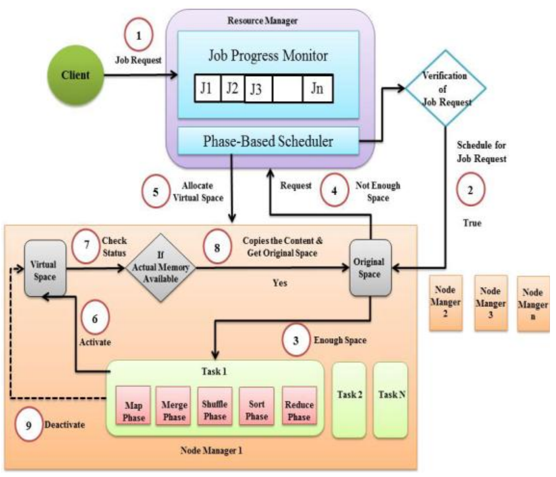


Fig.2. Implementation Workflow

V. DETAIL DESIGN

A. System Algorithm

System Algorithm Following steps will analysis the working of the phase scheduling algorithm.

✓ Step 1: When a task has to be scheduled, the scheduler replies to the heart beat message with scheduling request.

✓ Step 2: The Node Manager then launches the task implementation. When there is no enough space available to execute the job that is scheduled, then the application of the job will be paused.

✓ Step 3: Now the Node Manager will load the content that is paused due to the insufficient resources into the virtual space.

✓ Step 4: when the actual memory that is demanded by the Node Manager from phase-base scheduler is allocated.

✓ Step 5: The Node Manager will deactivate the virtual space and load the content into original space.

✓ Step 6: After finished executing a particular phase, the task asks getting permission to start the next phase from the Node Manager.

✓ Step 7: Then the Node Manager forwards the request for the getting permission to scheduler through heart beat message.

✓ Step 8: If the phase-based resource requirement and the current progress information are known, the scheduler decides whether to start the implementation of new task or paused task to begin next phase, and then this scheduling decision will be information to Node Manager.

✓ Step 9: When the application of all task is completed, the task status is received by Node Manager. Step 10: Then this task status will be forwarded to scheduler by Node Manager

B. Mathematical Expression

This scheduling has get work upon receiving a heartbeat message from the node manager that can be send the resource availability on the node. In the utility function of assigning the phase to the system n as there are J Jobs in system. Each Job $j \in J$ consists of two type approach map tasks M and reduce task R.

$$S = \{ f(i,n) : \{U_{fairness}(i,n), U_{perf}(i,n), r\} \varepsilon U_{(i,n)} \}$$

$$U_{(i,n)} = U_{fairness}(i,n) + U_{perf}(i,n)$$

Where ,

$n = No.\_of\_Machine$

$i = Assigning\_Pahse$

$U_{fairness}(i,n) = \text{Re}\ presents\_the\_utilities\_for\_improving\_fairness$

$U_{perf}(i,n) = \text{Re}\ presents\_the\_Job\_Performanc\ e$

✓ Utility on Fairness

Utility of fairness is calculated based on the Usage of Job Resource c_{jr} and Capacity of Task Tracker to be executed. In phase level scheduling, once a task has completed a phase, the subsequent phase of the task may not be scheduled immediately if the machine does not have sufficient resources to run the subsequent phase. Thus, the execution of a phase may be paused in order to avoid resource contention, at the cost of delaying the completion of the task. Therefore, to avoid the delay of task execution, the utility of fairness is calculated based on the total capacity of the single machine and the resource usage for particular task.

$$U_{fairness}(i,n) = \left(\frac{C_{jr}}{C_r}\right)$$

✓ Utility of Performance
Utility on Performance is calculated based on the currently running map task and reduce tasks and the number of remaining tasks yet to be executed. The performance of a job is measured based on the leading phase and non-leading phase. If the job is a leading phase, the gain of parallelism is measured in terms of the number of running map tasks (or reduces tasks). Otherwise if the job is a non-leading phase, the job performance is measured by the number of seconds that task has been paused due to phase-based scheduling.

Map Phase
Number of Pending Map task / Number of Currently Running Map task
Shuffle Phase
Number of seconds the task has been paused
Reduce Phase
Number of Pending Reduce task / Number of Currently Running Reduce

VI. SYSTEM IMPLEMENTATION

Here project are divide into five models which are explain below implemented model
Module 1:Getting Resource Utilization of a node in Hadoop cluster
Module 2:Utility working out on fairness
Module 3: Utility working out on job Performance
Module 4: Phase-based Scheduling Algorithm
Module 5:Benchmark algorithm Implementation

✓ Module 1:
Getting Resource Utilization of a node in Hadoop cluster
In MapReduce technique a job is allocated into multiple tasks and distributes the tasks of a job to a Task Tracker to be completed, the point of phase-based scheduling algorithm is to schedule the phase on every map and reduce task. The application of a map task can be allocated into two phases: map and merge phases. The Reduce task allocated into three phases: shuffle, sort, and reduce. Phase-based scheduling is to run or schedule the phase based on the resource utilization of that node. The resource utilization of every node is calculated based on the top command.

✓ Module 2:
Utility working out on fairness the fairness of the scheduling algorithm is improved by reducing the delay for implementation of every phase. In phase-based scheduling, once a task has completed a phase, the subsequent phase of the task may not be scheduled immediately if the machine does not have sufficient resources to run the subsequent phase.

Thus, the application of a phase may be paused in order to avoid resource contention, at the cost of delaying the completion of the task. Therefore, to avoid the delay of task implementation, the utility of fairness is calculated based on the total capacity of the single machine and the resource usage for particular task.

✓ Module 3:
Utility working out on job Perform and perform of a job is measured based on the leading phase and non-leading phase. If the job is a leading phase, the gain of parallelism is measured in getting s of the number of running map tasks (or reduces tasks). Otherwise if the job is a non-leading phase, the job perform is measured by the number of seconds that task has been paused due to phase-based scheduling.

✓ Module 4:
The phase-based Scheduling Algorithm is a set of phases that scheduled on a machine; the scheduler allocates a utility value to every phase which indicates the benefit of scheduling the phase.

A scheduler allocates the schedule based upon getting the status message from a Node Manager, the algorithm work out the utilization of the machine using jobs phase-based resource requirement. It then works out a set of applicant phases (i.e. the phases are schedulable on the machine) and selects phases in an iterative manner.

Each iteration, for every schedulable phase of every job, it works out the utility function created on the fairness and job performs. Then it selects phases with the maximum utility for scheduling and updates the resource utilization of the mechanism. Afterwards, the algorithm repeats by recompiling the utility of all the phases in the candidate set, and select the next best phase to schedule.

✓ Module 5:
Benchmarks algorithm Implementation to evaluate to execute of phase-based in Hadoop environment, the projected system implements the benchmark algorithm.

VII. EVALUATION

A. Experimental Setup and Workload

Here Hadoop cluster using 3 computers connected by Ethernet. Each node has different processor Memory and disk which shown in following table. Out of three machines, one machine was used as Job Tracker where as other two were used as Task-Trackers.

All machines were running the Ubuntu 12.04 operating system. We have used Hadoop-0.20.203.0 version which is considered as current stable version.

Table I. Configuration Parameter of each node

Sr. No	CPU	Memory	Disk
Node A (Master)	Intel (CR) Atom™ CPU N270 @ 1.60 GHZ	2.00	160 GB
Node B (Slave-1)	Intel (CR) Pentium ® Dual Core E2180 @ 2.00 GHZ	1.00 GB	160 GB
Node C (Slave-2)	Pentium ® Dual Core CPU E5700 @ 3.00 GHZ	2.00 GB	500 GB

B. Result Analysis

To compile the source code based on Hadoop-0.20.203.0 with Intel J Idea and finally, realizes phase-based scheduler. The compile _le jar is deployed on all nodes. The benchmark is already included in Hadoop-0.20.203.0, and Treasury has been used to sort the records on the Hadoop Cluster nodes. Treasury sort benchmark can generate computation intensive tasks. That's why these experiments are also based on Treasury benchmark instead of real-world traces.

✓ Execution Time of Tasks

To calculated execution time of task choose different sizes: (1024, 2048, and 3072 MB) of data to sort. The average numbers of tasks generated by each Task Tracker. In order to guarantee the fairness of experiments, each set of data is tested three times.

The experimental results are shown in Table below. As shown in Fig.1, phase-based scheduling has less execution time of tasks than both the original task scheduling algorithm of Hadoop in the heterogeneous Hadoop cluster. The execution time of tasks is becoming stable since single tasks are running in the cluster system.
Under the same cluster scale, with the increasing number of tasks, the performance improvement intuitively shows that the optimization effect of phase-based scheduler is obvious, which means that phase-based scheduler can improve the computing ability of the heterogeneous Hadoop cluster.

Table II. Configuration Parameter of each node

Algorithm	Files(MB)	The 1st Run(s)	The 2nd Run(s)	Averag e(s)
ORIGINAL	1024	598	585	591.5
PRISM	1024	554	560	557
ORIGINAL	2048	1520	1542	1531
PRISM	2048	1468	1485	1476
ORIGINAL	3072	2366	2302	2334
PRISM	3072	2297	2310	2303.5

✓ Resource Utilization

One of the nodes is randomly selected as the surveillance object. Meanwhile, 3072-MB input data of Treasury sort benchmark are also selected to verify the average resource utilization, including the CPU utilization, the memory utilization.

The experimental results are shown in Fig.3 below, with the original task scheduling algorithm; the system causes high resource utilization all the time while implementing tasks, particularly the utilization of CPU close to 100%. The long-term overloaded operation leads the system to lower efficiency of task execution. The performance of phase-based scheduler is better than the original task scheduling algorithm.

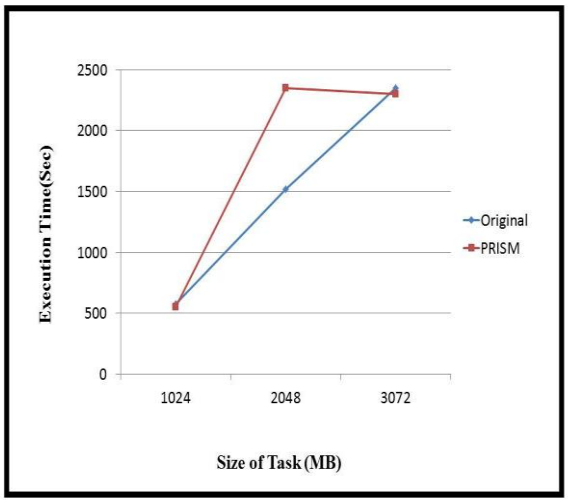


Fig.3. Graph for Job Execution of time scheduling

VIII. CONCLUSION

In this paper project is demonstrates that, if the resources emphasis on task-level, execution of each task may allocated into Phase-Level. Executing these phases, map and reduce tasks will adopt information and execute them in a similar across a large number of machine, so that it will reduce running time of data-intensive jobs. So they will perform resource allocation at the phase-level.

This project introduces Phase-Level [1] at the Phase. Phase-Level consist of how run-time resources can be used and how it varies over the long life time. Phase-Level-Improves job execution algorithm-Performance of resources

REFERENCES

[1] Qi Zhang, Student Member, IEEE, Mohamed FatenZhani,"PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce" IEEE Transactions On Cloud Computing, Vol. 3, No. 2, April/June 2015.
[2] Suryakant S. Bhalke,"Survey on Resource Allocation inPhase-Level using MapReduce in Hadoop" IJSR ISSN (Online): 2319-7064, Volume 4 Issue 11, November 2015.
[3] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley,Steinder,J. Torres, and E. Ayguad,Resource-aware adaptive scheduling for MapReduce clusters, in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 187–207.
[4] M. Zaharia, D. Borthakur, J. SenSarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving

locality and fairness in cluster scheduling," in Proc. Eur. Conf. Comput. Syst., 2010, pp. 265–278.
[5] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, and K. Talwar, "Quincy: Fair scheduling for distributed computing clusters,"in Proc. ACM SIGOPS Symp. Oper.Syst. Principles, 2009, pp. 261–276.
[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIXSymp. Netw. Syst. Des. Implementation, 2011, pp. 323– 336.
[7] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and Stoica,"Improving MapReduce performance in heterogeneous environments," in Proc. USENIX Symp.Oper. Syst. Des. Implementation, 2008, vol. 8, pp. 29–42.
[8] D. Xie, N. Ding, Y. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in Proc. ACM SIGCOMM,2012, pp. 199–210.
[9] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, andS. Babu, Starfish: A self-tuning system for big data analytics,"in Proc. Conf. Innovative Data Syst. Res.,2011, pp. 261–272.
[10] Hadoop MapReduce distribution [Online]. Available: http://Hadoop.apache.org, 2015.
[11] Hadoop Capacity Scheduler [Online]. Available: http://Hadoop.apache.org/docs/stable/capacity_scheduler.html/, 2015.
[12] Hadoop Fair Scheduler [Online]. Available:http://Hadoop.apache.org/docs/r0.20.2/fair_scheduler.html, 2015.
[13] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computationalmodels and the heterogeneity challenge," J.Internet Serv. Appl.,vol. 3, no. 1, pp. 1–10, 2012.
[14] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, andR. Sears, "MapReduce online," in Proc.USENIX Symp. Netw. Syst.Des. Implementation, 2010,
[15] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processingon large clusters," Commun. ACM, vol. 51, no.1, pp. 107–113, 2008.
[16] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. "Multi-resource allocation: Flexible tradeoffs in a unifying framework," in Proc. IEEEInt. Conf.Comput. Commun.,2012, pp. 1206–1214.
[17] A. Rasmussen, M. Conley, R. Kapoor, V. T. Lam, G. Porter, and A.Vahdat, "ThemisMR: An I/O-Efficient MapReduce," in Proc.ACMSymp. Cloud Comput., 2012,
[18] A. Verma, L. Cherkasova, and R. Campbell, "Resource provisioningframework for MapReduce jobs with performancegoals," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011,pp. 165–186.
[19] Y. Yu, M. Isard, D. Fetterly, M. Budiu, _ U. Erlingsson, P. Gunda,and J. Currey, "DryadLINQ: A system for general-purpose distributeddata-parallel computing using a high-level language,"in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, pp.1–14.