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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

PRISM: Phase and Resource Information aware Scheduler for MapReduce using Hadoop Cluster

Nilima P. Sonawane

P.G. Student, Department of Computer Engineering, S.S.V.P.S.'s B.S.Deore College of Engineering, Dhule, India

ABSTRACT: MapReduce has become a popular model for intensive data computing in recent years. By dividing each job into a small map and reducing tasks and running them in parallel on machines, MapReduce can significantly reduce the execution time of jobs that require a lot of data. However, despite recent efforts to design map reducers with efficient use of resources, existing solutions focused on activity-level programming still offer suboptimal work performance. This is due to the fact that the activities may have very varied resource requirements during their useful life, making it difficult for task-level programmers to effectively use the resources available to reduce work execution time. To overcome this limitation, PRISM is introduced. It is an extremely accurate map reduction planner that divides tasks into phases, where each phase has a constant resource usage profile and performs phase-level scheduling. PRISM improves execution parallelism and resource utilization.

KEYWORDS: MapReduce, Hadoop, Scheduling, Resource allocation.

I. INTRODUCTION

Now a Days, companies i.e businesses are entirely dependant on large-scale data analytics, so they can make critical business decisions day by day. This is directed to the development of Map-Reduce, i.e. a parallel programming model that has become equivalent with large-scale and data-intensive calculations. Map-Reduce consists of a job, which is a collection of map and reduce activities. These activities can be synchronously programmed on several machines, with a substantial reduction in work time.

An essential component of a Map-Reduce system is task planner i.e. job scheduler. The main role of the activity planning program is to create a mapping and reduction planning activity that includes one or more jobs, minimizes job completion time and maximizes resource utilization. In many situations, the containment of heavy resources and the time of completion of long processes take place due to a planning with too many tasks performed simultaneously on a single machine. On the contrary, hunger occurs because of the improper use of resources and also because of a planning with very few simultaneous activities in a single machine.

The problem of job scheduling becomes significantly simpler to solve, assuming that all map activities and all reduction activities have consistent resource requirements, such as CPU, memory, disk, and network bandwidth. However, this hypothesis is used to simplify the programming problem with current systems of Map Reduction, such as Hadoop Map-Reduce Version 1.x. This system uses a simple slot-based resource allocation scheme, in which the physical resources of each machine take on the amount of indistinguishable slots that can be allocated to activities.

This paper offers PRISM, which is a fine-stage programmer and resource information for map reduction clusters. PRISM realizes the conscious planning of resources at the level of the phases. Specifically, this document show's that, for Map-Reduce applications, the consumption of resources of the activity during the execution time can vary considerably from one phase to another. Therefore, it is possible that the planner has a greater degree of parallelism even if it avoids the containment of resources, only when it comes to the demand for resources at the phase level. Therefore, in the end, this document has developed a phase-level programming algorithm with the aim of obtaining high work performance together with the appropriate use of resources.

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

Paper is organized as follows. Section II describes MapReduce and Hadoop and also the previous related workdone by many. PRISM and its scheduling mechanism is explained in Section III. Section IV presents algorithm and its description . In section V, implementation and experimental results are shown. Finally, Section VI presents conclusion.

II. RELATED WORK

This section provides an overview of various studies and surveys, which is related to PRISM.

The application works on MapReduce a large cluster of raw materials machines and is highly scalable. A typical MapReduce computer processes a large amount of data in thousands of machinery. Many programs have been implemented in MapReduce Google cluster every day. Hadoop MapReduce[1] implements resource allocation scheme based on slots, which does not consider consumption of resources time of execution of the task. As a result, many recent ones newspapers have reported the inefficiency introduced because such a simple design and solutions proposed. For example, Polo et al. RAS has proposed [10], an adaptive scheduler resources uses a resource-conscious programming technique for MapReduce multi-job workloads aimed at improving resource utilization on all the machines it observes the objectives of the completion time.
However, RAS still exhibits activity-level planning, not consider changes in task use of resources at runtime. Subsequently, Hadoop yarn represents a great effort planning of resources in MapReduce cluster. Offers possibility to specify the size of each container of the activities in terms of requirements for each type of resource. In this context, a key challenge is to define the notion of equity when more resource types are considered Ghodsi et al. proposed key equity of resources as a measure of equity, in the presence of different types of resources, and provided simple programming algorithm for obtaining an almost optimal DRF However, the DRF[7] The scheduling algorithm is still focused on activity level programming and does not consider the variation in the consumption of resources inside His individual tasks after model, that is to say. Dominant Resource Fair Queuing (diabetes FC), aims to achieve DRF planning of packages over time. However, the algorithm is CFRD It is designed primarily for programming packages, which is different from the type of activity-based "bin-packing" planning model Consider in this role. Therefore, it cannot be applied directly to MapReduce programming. Use profiles to improve MapReduce work performance has received considerable attention in the past years. For example, Varma et al. [17] developed a framework that outlines the execution time of the activities and uses the work profiles to reach the expiry program in MapReduce cluster. Herodotus et al. newly developed starfish [8], a job profile generator that collects the characteristics of use of fine-grained tasks it can be used to adjust the configuration parameters of the job. However, the goal of the profiles in these studies is to optimize the work parameters, instead of optimizing work programs. Another the related search direction is the MapReduce funnel.
In particular, MapReduce Online [5] is a framework for processing based on the MapReduce workflow. It allows partial results of each phase that will be sent directly to the next stage, therefore, it allows the overlap of the phase execution. To minimize I / O, ThemisMR [11] is another scheme that makes the design decisions are fundamentally different from the previous ones MapReduce implementations. Themis does an extended Variety of MapReduce jobs almost at the speed of Triton Sort record classification performance. However, both these Solutions do not concern programming. Moreover, they are not aware of the resources. Introducing the awareness of resources into MapReduce Online is another interesting alternative, the programming model for MapReduce online is very different from the current MapReduce. Further research will be needed identify programming problems for Map-Reduce online.

III. PRISM

3.1 Prism Architecture

As it is cleared from the definition that, PRISM is a resource information-aware Map Reduce scheduler that distributes tasks into phases in a fine-grained manner, where each phase has a persistent resource usage profile and implements scheduling at the level of phases. During the execution time of a task, resource usage analysis may lead to ineffective scheduling decisions. Because of this, at run-time, if the resource allotted to a task is higher than the existing resource usage, then the idle resources are wasted. On the other hand, if the resources allotted to the task is much less than the actual resource demand, then the resource can suffer from a situation called, bottleneck, which may slow down task execution.

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

Therefore, a fine-grained, phase-level scheduling mechanism has been introduced. This allocates the resources according to the demand of the phase that each task is currently executing. Due to this fine-grained resource allocation, not a single task suffers from either bottleneck or starvation problem.
An overview of the PRISM architecture is shown in Fig. 1. PRISM comprises of four main modules: resource manager, local node managers, a job progress monitor and a phase-based scheduler. Initially, Resource Manager (also known as a job tracker), is responsible for scheduling tasks on each local node. Then, Local Node Manager, (also known as a task tracker) that coordinate phase transitions with the scheduler. Next is Job Progress Monitor, which is responsible to capture phase-level progress information. Finally, Phase-Based Scheduler, i.e., a fine-grained, phase-level scheduling mechanism that allocates resources according to the demand of executing phase (neither overflow nor underflow).

3.2 Phase level Scheduling Mechanism

In this mechanism, there are some steps which are followed during the execution of PRISM. These steps are:

1. Each local node manager sends a heartbeat message to the phase-based scheduler periodically. As soon as a task requests to be scheduled, then the scheduler immediately responses to the heartbeat message with a task scheduling request.
2. Then, the local node manager initiates the task.
3. As and when a task completes implementing a particular phase (shuffle phase), then the task requests the local node manager for permission to start the next phase (e.g. reduce phase).
4. The local node manager then forwards this permission request to the phase-based scheduler.
5. Finally, once the task is permitted to execute the next phase (reduce phase), the local node manager grants permission to process that task and once the task is completed; the task status is received by the local node manager and then dispatched to the phase-based scheduler.

PRISM requires constant phase-level resource information for each job to perform phase-level scheduling. In this way, the entire task is implemented. Each phase travels through all the above steps and finally get completed successfully.

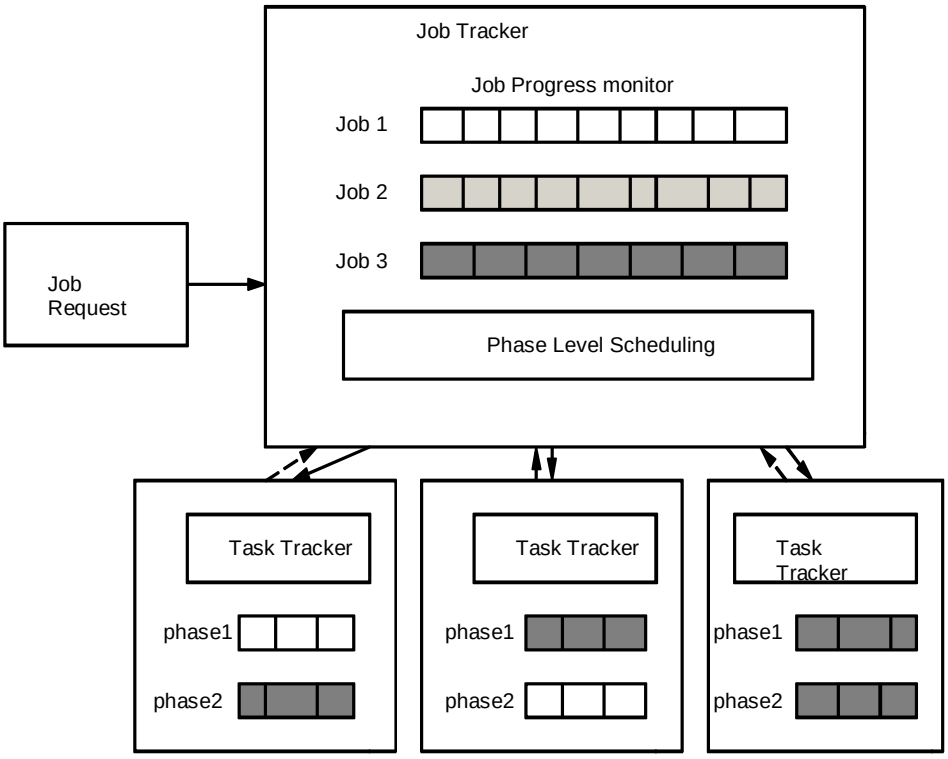

Fig. 1 System architecture

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

IV. ALGORITHM AND ITS DESCRIPTION

4.1 Phase level Sceduling Algorithm

1.Upon receiving a status message from machine n:
2. Obtain the resource utilization of machine n
3. PhaseSelected **PS** ← {∅}
4. CandidatePhases **CP** ← {∅}
5. **repeat**
6. **for** each job j ∈ jobs that has tasks on **n do**
7. **for** each schedulable phase i ∈ j **do**
8. CP ← CP ∪ {i}
9. **end for**
10. **end for**
11. For each job **j** ∈ top k jobs with highest deficit n **do**
12. **if** exist schedulable data local task **then**
13. CP ← CP ∪ {first phase of the local task i}
14. **else**
15. CP ← CP ∪ {first phase of the non-local task i}
16. **end if**
17. **end for**
18. **if** CP ≠ ∅ **then**
19. **for** i ∈ CP **do**
20. **if** *i* is not schedulable on *n* given current utilization **then**
21. CP ← CP ∪ {i}
22. Continue;
23. **end if**
24. Compute the utility **U** (i, n)
25. **if** U(i, n) <= 0 **then**
26. CP ← CP ∪ {i}
27. **end if**
28. **end for**
29. **if** CP ≠ ∅ **then**
30. **i** ← task with highest U (i ,n) in the CP
31. PS ← PS U {i}
32. CP ← CP ∪ {i}
33. Update the resource utilization of machine *n*
34. **end if**
35. **end if**
36. Until CP = = ∅
37. Return ***PS***

4.2 Algorithm Description

This algorithm describes the scheduling algorithm used by the phase-based scheduler. In this algorithm, two important concepts are used, which are Efficiency and Fairness [8], [14]. However, a Map-Reduce scheduler is responsible to assign each task to an appropriate machine along with the consideration of both Efficiency and Fairness.

Efficiency is achieved only by maintaining high utilization of resources in a cluster by the job schedulers. Another effective measure for efficiency is job running time because, during an execution of a task, minimum job running time indicates maximum utilization of resources in an efficient manner.

Secondly, Fairness provides an assurance that, all the resources are fairly distributed among each and every job. This aspect ensures that, there will be neither a bottleneck situation (i.e. overflow of resources) nor a starvation situation (i.e. underflow of resources).

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

However, achieving both the aspects, i.e. Efficiency and Fairness concurrently seems to be very challenging with respect to the multi-resource scheduling.

Initially in the above algorithm, the local node manager sends the status message to the Phase-Level Scheduling Algorithm in order to allocate necessary resources. Then, Line 2 states that, upon receiving the status message from a local node manager running on machine n, the algorithm computes the utilization u of the machine using job's phase-level resource requirement. Next, Lines 4-10 consists of a set of candidate phases (i.e. the schedulable phases) and selects these phases in an iterative manner.

Then, in Lines 11-23, for each job j, if phase has highest deficit n, then first phase of local task i is executed; otherwise first phase of the non-local task i is executed. These steps are iterated for each job.

Next, in Lines 24-28, for each schedulable phase i of each job j in each iteration, the algorithm computes the utility function U (i, n) according to the following equation:

$$\mathbf{U\ (i,\ n) = U\ fairness(i,\ n) + a\ .U\ perf\ (i,\ n)}$$

where, U fairness and U perf signifies the utilities for improving the fairness and job performance, respectively, and 'a' is an adjustable weight factor.

The fairness of each phase is calculated as

$$\mathbf{U\ fairness(i,\ n) = U\ before\ fairness(i,\ n) - U\ after\ fairness(i,\ n)}$$

where, U before fairness and U after fairness signify the fairness measures of the job before and after scheduling phase i on machine n.

Then, in Lines 29-32, the phase with the highest utility for scheduling is selected and Line 33 updates the resource utilization of the machine n.

Subsequently, in Lines 34-37, the algorithm repeats the above steps by re-computing the utility of all the phases in the candidate set, and selects the succeeding best phase to schedule. Finally, the algorithm concludes when the candidate set becomes empty, which means that, there is no suitable phase to be scheduled.

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Implementation

The PRISM architecture is implemented as comparison between existing and the proposed system.All the implementation and experiments are performed on a machine running Ubuntu operating system with hadoop version 1.2. The algorithm for executing the project is implemented in Java, i.e. JDK1.7.0 version.

The Net-Beans tool is used for proposed system implementation. It consists of a master node and slave node. The important component, i.e. a Phase-Based Scheduler is appeared in Master node. This Phase-Based Scheduler is responsible for scheduling the phases and allocating the resources. This scheduler is also known as a Fine-Grained Resource-Aware Scheduler because; it allocates the resources as per the demand of the phases, (i.e. neither overflow nor underflow).

5.2 Experimental Results

The Proposed System (i.e. Phase-level Scheduling) introduces PRISM, which is a fine-grained resource-aware map-Reduce scheduler. PRISM divides tasks into phases, where each phase has a persistent resource usage profile and implements scheduling at the level of phases. The following Fig. shows Time Graph of fair scheduling and phase based scheduling.

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

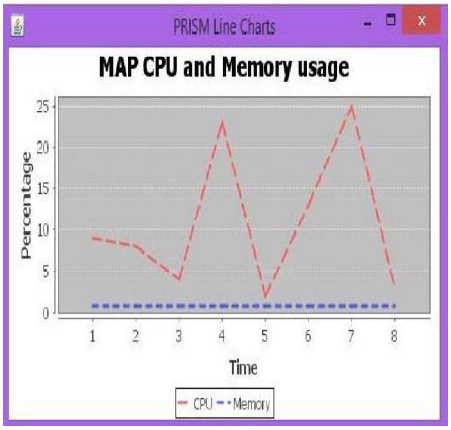

Fig. 2 Map CPU and Memory Usage(existing system)

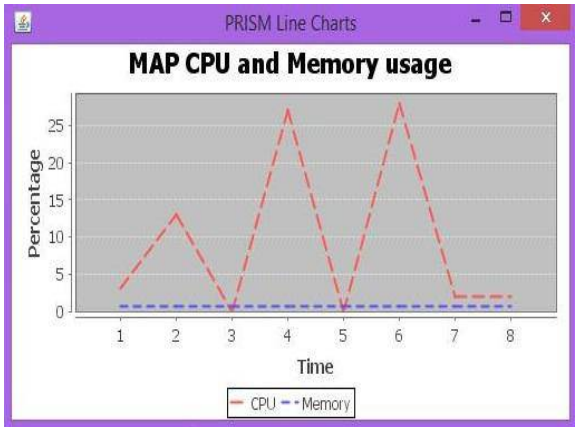

Fig. 3Map CPU and Memory Usage (proposed system)

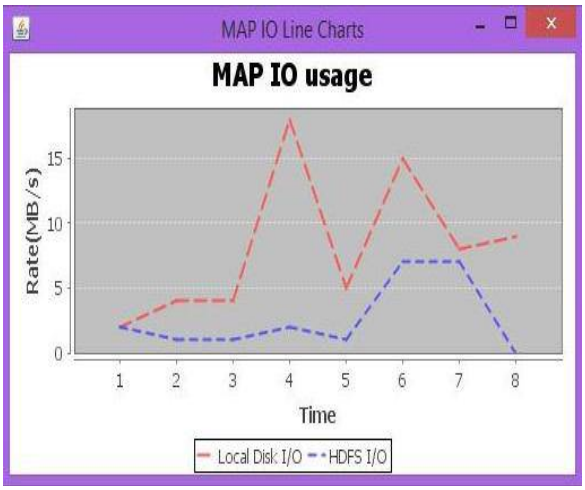

Fig. 4Map IO usage(existing system)

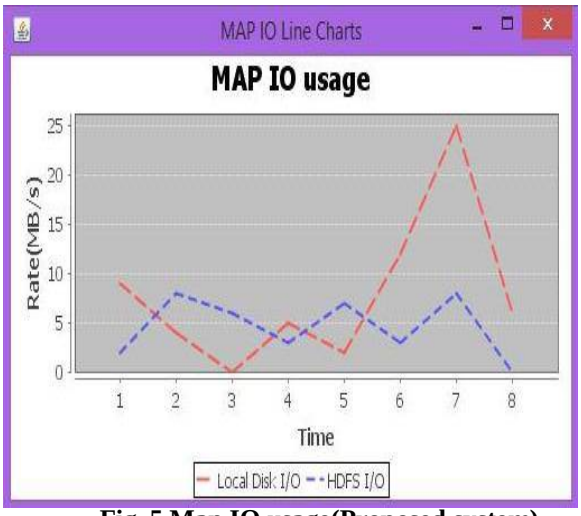

Fig. 5 Map IO usage(Proposed system)

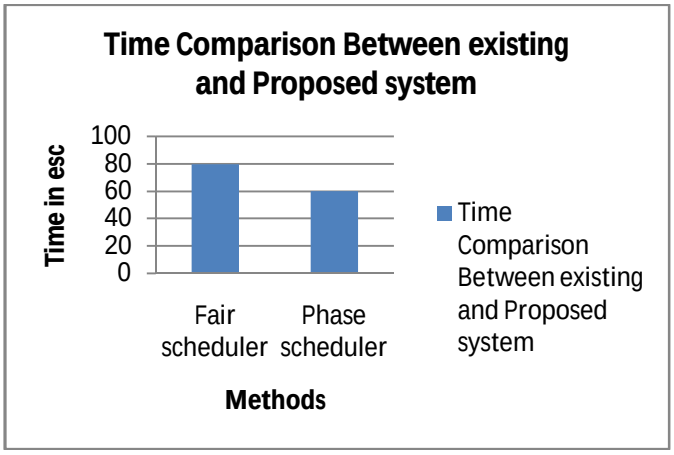

Fig.6 Running time of each system

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

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 6, June 2018

VI. CONCLUSION

MapReduce is programming model for hadoop cluster to perform a data-intensive computing. PRISM, which is a fine-grained resource allocation and reduction planner, divides tasks into phases where each phase has a constant resource usage profile and also performs phase-level scheduling. The planning algorithm used by PRISM contributes to minimize work execution time compared to current Hadoop schedulers and improves resource utilization.

REFERENCES

[1] Hadoop MapReduce distribution [Online]. Available: http://hadoop.apache.org, 2015.
[2] Hadoop Fair Scheduler [Online]. Available: http://hadoop.apache.org/docs/r0.20.2/fair_scheduler.html, 2015.
[3] Hadoop Distributed File System [Online]. Available: hadoop.apache.org/docs/hdfs/current/, 2015.
[4] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computational models and the heterogeneity challenge," J. Internet Serv. Appl., vol. 3, no. 1, pp. 1–10, 2012.
[5] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," in Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2010, p. 21.
[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2011, pp. 323–336.
[8] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics," in Proc. Conf. Innovative Data Syst. Res., 2011, pp. 261–272.
[9] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, and K. Talwar, "Quincy: Fair scheduling for distributed computing clusters," in Proc. ACMSIGOPS Symp. Oper. Syst. Principles, 2009, pp. 261–276.
[10] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguad_e, "Resource-aware adaptive scheduling for MapReduce clusters," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 187–207.
[11] A. Rasmussen, M. Conley, R. Kapoor, V. T. Lam, G. Porter, and A. Vahdat, "ThemisMR: An I/O-Efficient MapReduce," in Proc. ACM Symp. Cloud Compute. 2012, p. 13.
[12] A. Verma, L. Cherkasova, and R. Campbell, "Resource provisioning framework for MapReduce jobs with performance goals," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 165–186.
[13] 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.
[14] Y. Yu, M. Isard, D. Fetterly, M. Budiu, _ U. Erlingsson, P. Gunda, and J. Currey, "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, pp. 1–14.
[15] M. Zaharia, D. Borthakur, J. Sen Sarma, 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.
[16] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008, vol. 8, pp. 29–42.
[17] Qi Zhang, Student Member, IEEE, Mohamed Faten Zhani, Member, IEEE, Yuke Yang, Raouf Boutaba, Fellow, IEEE, and Bernard Wong, "PRISM: Fine-Grained Resource-Aware Scheduling for Map-Reduce," in ieee transactions on cloud computing, vol. 3, no. 2, april/june 2015.