International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 06 | June 2019 www.irjet.net p-ISSN: 2395-0072

# Evaluating and Comparing the Two Variation with Current Scheduling Algorithms

**P.Chenna Kesava[1], P.Subhan Basha[2]**

*[1]PG Student, Department of C.S.E, SSITS, Rayachoti Kadapa, A.P, India.*
*[2]Assistant Professor Dept. of C.S.E, SSITS, Rayachoti Kadapa, A.P, India.*

---

**Abstract -** It is cost-efficient for a tenant with a limited budget to establish a virtual MapReduce cluster by renting multiple virtual private servers (VPSs) from a VPS provider. To provide an appropriate scheduling scheme for this type of computing environment, we propose in this paper a hybrid job-driven scheduling scheme (JoSS for short) from a tenant's perspective. JoSS provides not only joblevel scheduling, but also map-task level scheduling and reduce-task level scheduling. JoSS classifies MapReduce jobs based on job scale and job type and designs an appropriate scheduling policy to schedule each class of jobs. The goal is to improve data locality for both map tasks and reduce tasks, avoid job starvation, and improve job execution performance. Two variations of JoSS are further introduced to separately achieve a better map-data locality and a faster task assignment. We conduct extensive experiments to evaluate and compare the two variations with current scheduling algorithms supported by Hadoop. The results show that the two variations outperform the other tested algorithms in terms of map-data locality, reduce-data locality, and network overhead without incurring significant overhead. In addition, the two variations are separately suitable for different MapReduce-workload scenarios and provide the best job performance among all tested algorithms.*)*

***Key Words***: MapReduce, Hadoop, virtual MapReduce cluster, map-task scheduling, reduce-task scheduling

## 1. INTRODUCTION

MapReduce works by dividing input files into chunks and processing these in a series of parallelizable steps.

More and more applications with their data relying on cloud platforms are geo distributed, for many reasons. Working on such remote outsourced data sets present in those cloud data centers most efficiently with MapReducing is much more complex. Assume that there are substantially sized web caches on multiple continents and that a web administrator needs to execute a query across this data set. Some methods are as follows Gathering all subdata sets into a single data center before handling them with MapReduce is one possibility. Execute individual instances of the MapReduce job separately on each subdata set in respective data centers and then aggregate the results. Perform the MapReduce jobs as a single remote outsourced operation where mappers and reducers may be deployed in different data centers, via random allocation in the considered data centers. One & Two Solutions are inefficient and complex but third one is optimal. Current MapReduce implementations such as Hadoop are not designed to work across multiple data centers and will perform poorly in such deployments. So a better system is required to achieve that.

We implement Joss-T and JOSS-J in a Custom MapReduce Framework conduct extensive experiments to compare them with several known scheduling algorithms supported by Hadoop, including the FIFO algorithm, Fair scheduling algorithm, and Capacity-scheduling algorithm. The experimental results demonstrate that both JOSS-T and JOSS-J outperform the other tested algorithms in terms of map-data locality, reduce-data locality, and network overhead without causing too much overhead, regardless of job type and scale.

### 1.1 PURPOSE OF THE PROJECT

- We analyze the problem of executing remote outsourced MapReduce job sequences as arising in distributed applications.
- We present a job-driven scheduling scheme JOSS, a system for efficiently executing sequences of MapReduce jobs on remote outsourced data sets using a novel algorithm named data transformation that determines an effective execution path to execute a given sequence of MapReduce jobs on a remote outsourced data set, optimizing for either the execution time or the cost.
- JOSS is a Hadoop based framework that can efficiently perform a sequence of MapReduce jobs on a remote outsourced data set across multiple data centers.
- With current frameworks for "the cloud" operating only in single data centers, JOSS thus—metaphorically speaking—acts much like the atmosphere surrounding the clouds.
- Processing Steps
- A given node of a cluster describes the number of MapReduce phases that have been applied on input data and the location of the derivative of each partition.
- Each row of nodes of the cluster belongs to the same stage.

- So, a node in the cluster is described as Ns, where s is the number of MapReduce operations in the sequence that have been applied so far and x is a p-tuple of integers of the form d1; d2... di describing the current distribution of data across a data center.
- To and From Transfer of Data across geo-distributions and Main Processor with respect to above states is required.
- JOSS scheduling algorithm involves constructing a graph named a data transformation cluster, representing possible execution paths for performing MapReduce phases on input data.
- JOSS scheduling algorithm can minimize execution time or cost. Using Virtual MR Clusters reduces execution time or cost accordingly.
- A user can specify if the JOSS algorithm should determine an optimized solution for execution time or the (monetary) cost, where cost involves both cost to maintain nodes and for transferring data. We implement with respect to time as default metric and Performance results are highlighted.

### 1.2 MAPREDUCE

MapReduce is a distributed programming model proposed by Google to process vast amount of data in a parallel manner. Due to programming-model simplicity, built-in data distribution, scalability, and fault tolerance, MapReduce and its open-source implementation called Hadoop have been widely employed by

Many companies, including Face book, Amazon, IBM, Twitter, and Yahoo!, to process their business data. MapReduce has also been used to solve diverse applications, such as machine learning, data mining, bioinformatics, social network, and astronomy. Other MapReduce-like implementations can be found.

Each VPS is a virtual machine with its own operating system and disk space. Due to some reasons, such as availability issue of a datacenter or resource shortage on a popular datacenter, a tenant might rent VPSs from different datacenters operated by a same VPS provider to establish his/her virtual MapReduce cluster.

In this paper, we concentrate on a virtual MapReduce cluster of this type. For a person/organization that establishes a conventional MapReduce cluster, map-data locality (which is defined as how close a map task and its input data) in the cluster is classified into node locality, rack locality, and off-rack since the person/organization is aware of the physical interconnection and placement among all nodes and all racks. However, for a tenant who establishes a virtual MapReduce cluster, the tenant only knows each VPS's IP address and each VPS's datacenter location (e.g., city name).

VPS-locality, which means that a map task and its input data are co-located at the same VPS.

Well data intensive is nothing but big data and distributed applications are the application that works on network by communicating and coordinating with each other by passing messages. . Again remember it's a framework that handles large amount of data for processing. You will get to know the difference between Hadoop and Databases as you go down the line in the coming tutorials. Hadoop was derived from the research paper published by Google on GFS and Google's MapReduce. There are two integral parts of Hadoop HDFS.

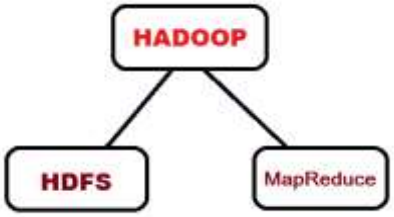



**Fig.1.Hadoop**

### HADOOP DISTRIBUTED SYSTEM (HDFS)

- **Cluster**

A hadoop cluster is made by having many machines in a network; each machine is termed as a node, and these nodes talks to each other over the network.

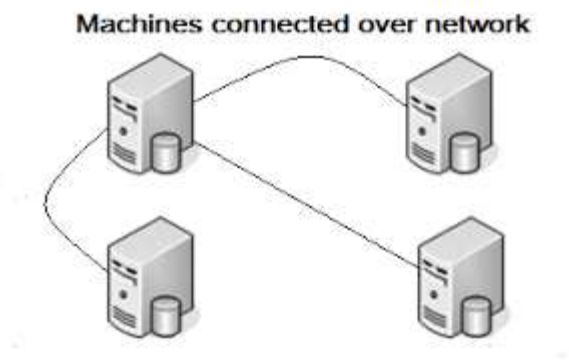



**Fig. 2.Hadoop distributed file system (HDFS)**

- **Block size**

This is the minimum amount of size of one block in a file system, in which data can be kept contiguously. The default size of a single block in HDFS is 64Mb.In HDFS, Data is kept by splitting it into small chunks or parts. Let's say you have a text file of 200 MB and you want to keep this file in a Hadoop Cluster. Then what happens is that, the file breaks or splits into a large number of chunks, where each chunk is equal to the block size that is set for the HDFS cluster (which is 64 MB by default). Hence a 200 Mb of file gets split into 4 parts, 3

parts of 64 MB and 1 part of 8 MB, and each part will be kept on a different machine

- **Master node**

The name node manages the file system namespace. It maintains the file system tree and the metadata for all the files and directories in the tree. So it contains the information of all the files, directories and their hierarchy in the cluster in the form of a Namespace Image and edit logs.

### Worker node

These are the worker that does the real work. And here by real work we mean that the storage of actual data is done by the data node. They store and retrieve blocks when they are told to (by clients or the name node), and they report back to the name node periodically with lists of blocks that they are storing. Here one important thing that is there to note: In one cluster there will be only one Name node and there can be N number of data nodes.

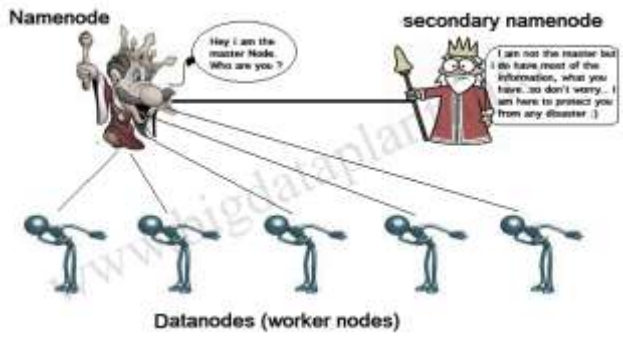



**Fig.3.Worker nodes**

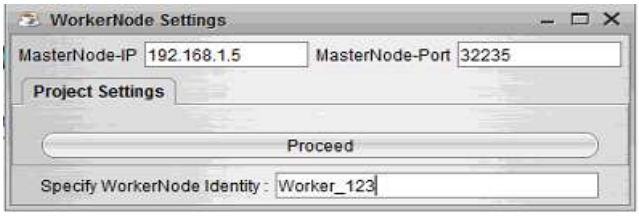
## 2. PROCESS MODEL USED WITH JUSTIFICATION

Irjet Template sample paragraph .Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and RMS do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

## OUTPUT SCREENSHOTS

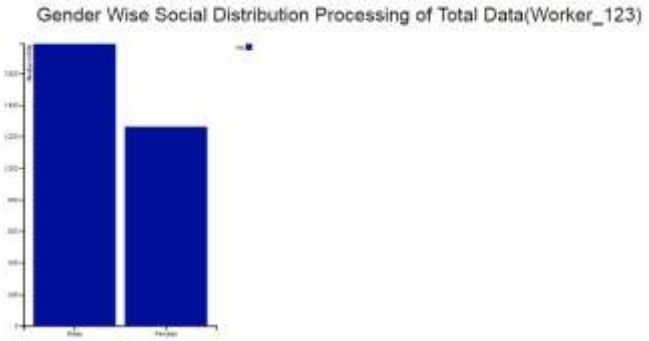
### MASTER NODE


### WORKER NODE

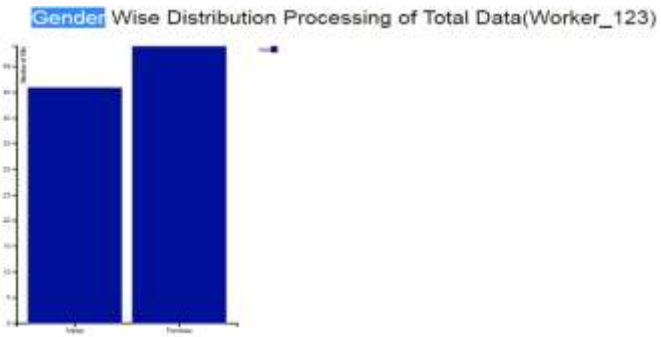



## MAPREDUCE OUTPUT

### GENDER WISE SOCIAL DISTRIBUTION PROCESSING OF TOTAL DATA

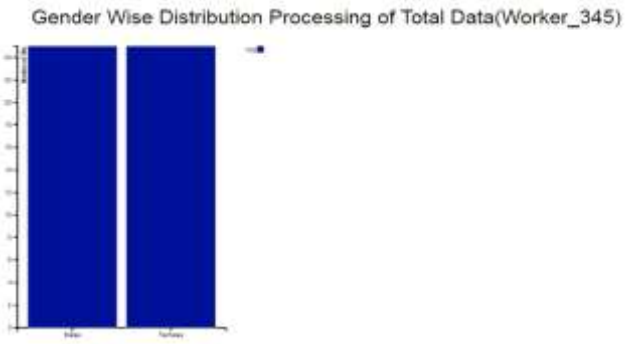



### GENDER WISE DISTRIBUTION PROCESSING OF TOTAL DATA



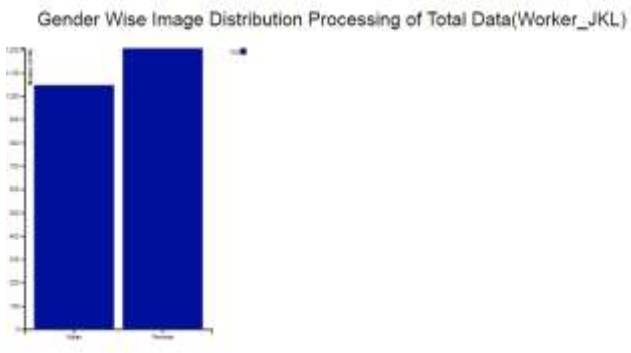

## OUT PUT

### GENDER WISE DISTRIBUTION PROCESSING OF TOTAL DATA

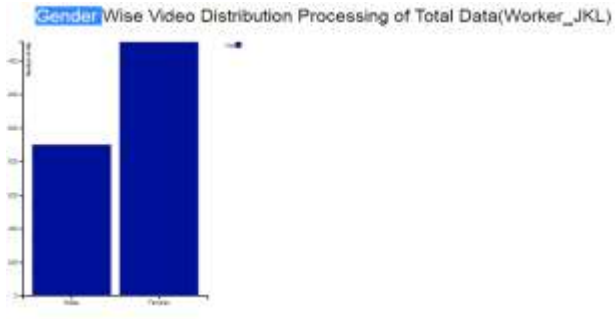


**DYNAMIC GROUPING VM – JOSS OUTPUT**

**IMAGE DISTRIBUTION**




**VIDEO DISTRIBUTION**




**INTRODUCTION TO TESTING**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform

## 3. TESTING IN STRATEGIES

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

### Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two Strategies

### Black Box Testing

In this strategy, some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

### White Box Testing

In this, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.
  Execute all logical decisions on their true and false Sides.
- Execute all loops at their boundaries and within their operational bounds
- Execute internal data structures to ensure their validity.

### Integrating Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. In this, case the communication between the device and Google Translator Service.

### System Testing

Involves in-house testing in an emulator of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

### Acceptance Testing

It is a pre-delivery testing in which entire system is tested in a real android device on real world data and usage to find errors.

**TEST APPROACH**

Testing can be done in two ways**:**

- Bottom up approach
- Top down approach

**Bottom up Approach**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing, a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention, turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down Approach**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs be written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

## 3. CONCLUSIONS

In this paper, we have introduced JOSS for scheduling MapReduce jobs in a virtual MapReduce cluster consisting of a set of VPSs rented from a VPS provider. Different from current MapReduce scheduling algorithms, JOSS takes both the mapdata locality and reduce-data locality of a virtual MapReduce cluster into consideration. JOSS classifies jobs into three job types, i.e., small map-heavy job, small reduce-heavy job, and large job, and introduced appropriate policies to schedule each type of job. In addition, the two variations of JOSS are further introduced to respectively achieve a fast task assignment and improve the VPS-locality.

The extensive experimental results demonstrate that both JOSS-T and JOSS-J provide a better map-data locality, achieve a higher reduce-data locality, and cause much less inter-datacenter network traffic as compared with current scheduling algorithms employed by Hadoop.

The experimental results also show that when the jobs of a MapReduce workload are all small to the underlying virtual MapReduce cluster, employing JOSS-T is more suitable than the other algorithms since JOSS-T provides the shortest job turnaround time. On the other hand, when the jobs of a MapReduce workload are not all small to the virtual MapReduce cluster, adopting JOSS-J is more appropriate because it leads to the shortest workload turnaround time. In addition, the two variations of JOSS have a comparable load balance and do not impose a significant overhead on the Hadoop master server compared with the other algorithms.

## REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2] Hadoop. (2014, Dec. 3) [Online]. Available: http://hadoop. apache.org TABLE 10 The Average VPS Loads when the Five Algorithms Were Individually Used to Perform the Mixed Workload Algorithm Average number of tasks run by each VPS Standard deviation JoSS-T 98.23 7.78 JoSS-J 98.23 11.06 FIFO 98.23 18.30 Fair 98.23 9.46 Capa 98.23 14.74 Fig. 16. The CPU idle rate of the Hadoop master server when the five algorithms are individually used to execute the mixed workload. Fig. 17. The Memory load of the Hadoop master server when the five algorithms are individually used to execute the mixed workload. 1698 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 27, NO. 6, JUNE 2016

[3] S. Chen and S. Schlosser, "Map-Reduce meets wider varieties of applications," Intel Res., Santa Clara, CA, USA, Tech. Rep. IRPTR-08-05, 2008.

[4] B. White, T. Yeh, J. Lin, and L. Davis, "Web-scale computer vision using mapreduce for multimedia data mining," in Proc. 10th Int. Workshop Multimedia Data Mining, Jul. 2010, pp. 1–10.

[5] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in Proc. IEEE 4th Int. Conf. eScience, Dec. 2008, pp. 222–229.

[6] X-RIME. (2014, Dec. 3) [Online]. Available: http://xrime. sourceforge.net/

[7] K. Wiley, A. Connolly, J. Gardner, S. Krughoff, M. Balazinska, B. Howe, Y. Kwon, and Y. Bu, "Astronomy in the cloud: using mapreduce for image co-addition," Astronomy, vol. 123, no. 901, pp. 366–380, 2011.

[8] Disco. (2014, Dec. 3) [Online]. Available: http://discoproject.org

[9] Gridgain. (2014, Dec. 3) [Online]. Available: http://www. gridgain.com

[10] MapSharp. (2014, Dec. 3) [Online]. Available: http://mapsharp. codeplex.com

[11] Amazon Web Services. (2014, Dec. 3) [Online]. Available: https:// aws.amazon.com/elasticmapreduce/

[12] Linode. (2014, Dec. 3) [Online]. Available: https://www.linode. com/

[13] Future Hosting. (2014, Dec. 3) [Online]. Available: http://www. futurehosting.com/

[14] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in mapreduce," in Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., May 2012, pp. 419–426.

[15] C. He, Y. Lu, and D. Swanson, "Matchmaking: A new mapreduce scheduling technique," in Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci., Nov. 2011, pp. 40–47.

[16] T. White, Hadoop: The Definitive Guide. Sebastopol, CA, USA: O'Reilly Media, Jun. 5, 2009.

[17] 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. 5th Eur. Conf. Comput. Syst., Apr. 2010, pp. 265–278.

[18] J. Jin, J. Luo, A. Song, F. Dong, and R. Xiong, "BAR: An efficient data locality driven task scheduling algorithm for cloud computing," in Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., May 2011, pp. 295–304.

[19] Fair Scheduler Guide. (2014, Dec. 3) [Online]. Available: http:// archive.cloudera.com/cdh/3/hadoop-0.20.2+737/fair_scheduler. html

[20] Capacity Scheduler Guide (2014, Dec. 3) [Online]. Available: http:// archive.cloudera.com/cdh/3/hadoop-0.20.2+737/capacity_scheduler.html [21] M. Ehsan, and R. Sion, "LiPS: A cost-efficient data and task co-scheduler for MapReduce," in Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. Workshops PhD Forum, May 2013, pp. 2230–2233.

[22] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: Localityaware resource allocation for MapReduce in a cloud," in Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal., Nov. 2011, pp. 58.

[23] J. Park, D. Lee, B. Kim, J. Huh, and S. Maeng, "Locality-aware dynamic VM reconfiguration on MapReduce clouds," in Proc. 21st Int. Symp. High-Perform. Parallel Distrib. Comput., Jun. 2012, pp. 27–36.

[24] X. Bu, J. Rao, and C.-Z. Xu, "Interference and locality-aware task scheduling for Mapreduce applications in virtual clusters," in Proc. 22nd Int. Symp. High-Perform. Parallel Distrib. Comput., Jun. 2013, pp. 227–238.

[25] S.-Y. Ko, I. Hoque, B. Cho, and I. Gupta, "Making cloud intermediate data fault-tolerant," in Proc. ACM Symp. Cloud Comput., 2010, pp. 181–192.

[26] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in Proc. IEEE Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst., 2009, pp. 1–11.

[27] Statistical workload injector for mapreduce. (2014, Dec. 3)[Online]. Available: https://github.com/SWIMProjectUCB/SWIM/wiki

[28] Apache. (2014, Dec. 3). Hadoop wiki, powered by [Online]. Available: http://wiki.apache.org/hadoop/PoweredBy

[29] F. Ahmad, S. Lee, M. Thottethodi, and T. N. Vijaykumar. (2012). PUMA: Purdue MapReduce benchmarks suite. ECE Tech. Rep., Purdue Univ. [Online]. Available: http://docs.lib.purdue.edu/ ecetr/437

[30] enwiki. (2014, Dec. 3) [Online]. Available: http://dumps.wikimedia.org/enwiki/

[31] Pseudomonas Genome Database. (2014, Dec. 3) [Online]. Available: http://www.pseudomonas.com/strain/list

[32] L. Kleinrock, "Queueing systems," in Theory, vol. 1. New York, NY, USA: Wiley, 1975.

[33] MapReduce Benchmarks. (2014, Dec. 3) [Online]. Available: https://878262af-a-62cb3a1a-s-sites.googlegroups.com/site/farazahmad/home/puma.pdf [34] Free txt mobile ebooks downloads. (2014) [Online]. Available: http://www.umnet.com/mobile-ebooks/0-0-0-txt-0

[35] C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic mapreduce scheduler for heterogeneous workloads," in Proc. IEEE 8th Int. Conf. Grid Cooperative Comput., 2009, pp. 218–224.

[36] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguade, M. Steinder, and I. Whalley, "Performance-driven task co-scheduling for mapreduce environments," in Proc. IEEE Netw. Oper. Manage. Symp., 2010, pp. 373–380.