World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 3.419

# JOB SCHEDULING SCHEME FOR IMAGE PROCESSING TO IMPROVE PERFORMANCE IN HADOOP

**[1*]Malatesh S. H., [2]Pallavi S. T., [3]Raveena B., [4]Shreyanka G. P. and [5]Vanitha Y.**

[1]Head of the Department, Computer Science and Engineering, M. S. Engineering College, Bengaluru.

[2,3,4,5]Computer Science and Engineering, M. S. Engineering College, Bengaluru.

**Article Received on 31/05/2016 Article Revised on 20/06/2016 Article Accepted on 10/07/2016**

***Corresponding Author**
**Malatesh S. H.**
Head of the Department, Computer Science and Engineering, M. S. Engineering College, Bengaluru.

**ABSTRACT**

With the growth of technology, the number of images being uploaded to the internet is exploding. Most current image processing applications, designed for small and local computation, do not integrate well to web-sized problems with their large requirements for resources used in computation and storage. Hadoop and its Mapreduce paradigm are emerging as an important standard for large and data-intensive processing in both industry and academia. A Mapreduce cluster is typically shared among many users with various types of workloads. One challenging issue is to efficiently schedule all the jobs in shared Mapreduce environment in Hadoop. However, we find that prior scheduling algorithms supported by Hadoop cannot ensure good performance for different Image processing workloads. To address this we have developed the Hadoop Image Processing Framework that provides a library that is Hadoop based to support large-scale image processing using Mesos, the resource manager. We propose a new Hadoop scheduler that leverages the study of workload patterns to improve the performance of the system by tuning the resource share dynamically among users and the scheduling algorithms for each user in Hadoop. Mesos is designed using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel will be running on every machine and will provide applications (e.g., Hadoop, Spark, Kafka, and Elastic Search) with API's for scheduling and resource management across the whole data center and all cloud

environments. This new framework enhances the performance of about 5-10% of image processing in hadoop.

**KEYWORDS:** MapReduce, Hadoop, Scheduling, Performance, Mesos.

## INTRODUCTION

Apache Hadoop based cluster is built for not only homogenous or single job at a time, once cluster is commissioned; cluster in real time scenario will be shared by many jobs. Classic implementation of Hadoop uses FIFO based schedulers, which is a default scheduler. When multiple jobs are submitted by users simultaneously, the default job scheduler gives precedence for the job which takes precedence in submission order. There is considerable performance bottleneck in this approach.

To overcome the shortcomings of FIFO scheduler, Fair scheduler was introduced to deal with small jobs and user heterogeneity and to manage access to their Hadoop cluster. Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average, an equal share of resources over time. The main disadvantage is that jobs will be allocated to all the slots in the cluster with maximum slot capacity and the algorithm does not consider the job weight of each node.

Moreover, recent advances in multi core processor, requires trade-off between performance and power budget. Hadoop job loads can be classified as large batch jobs, iterative jobs, interactive jobs, etc.., while large batch job require throughput, interactive job require performance to speed up the execution time. On heterogeneous cluster platform with multi core processors, jobs happened to be treated with even trade off for slow and fast cores. Current implementation has fixed approach which by cluster level job execution degrades its performance.

## REQUIREMENT

### Hardware Requirements

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- Floppy Drive: 44 MB.
- Monitor: 15 VGA Colour.
- Ram: 512 MB.

**Software Requirements**

- Operating system: Ubuntu.
- Coding Language: Java 1.7.
- Software: Apache Hadoop 1.2.1, Mesos.
- IDE: Eclipse

**Software Overview**

**Mesos**

A distributed systems kernel, Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elastic Search) with API's for resource management and scheduling across entire datacenter and cloud environments.

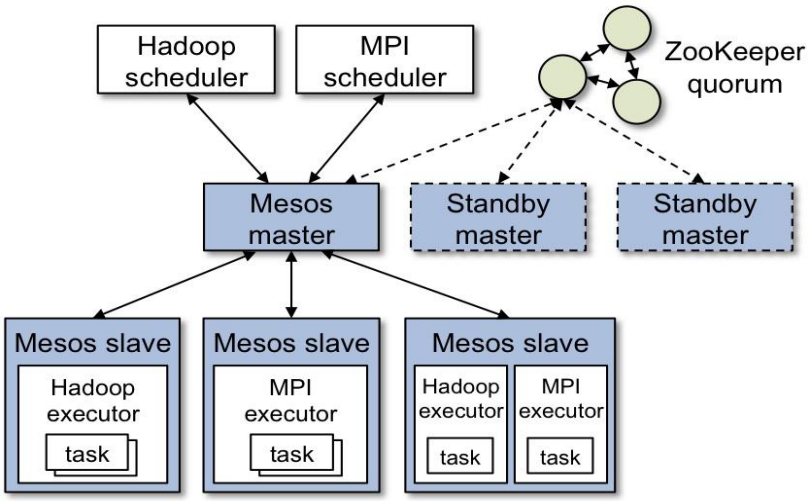


**Figure 1:** ***Architecture of Mesos.***

The above figure 1 shows the main components of Mesos. Mesos consists of a master daemon that manages slave daemons running on each cluster node, and Mesos frameworks that run tasks on these slaves.

**Hadoop Image Processing Interface (HIPI)**

The primary input object to a HIPI program is a Hipi Image Bundle (HIB) as shown in the figure 2. A HIB is a collection of images represented as a single file on the HDFS. The HIPI distribution includes several useful tools for creating HIBs, including a MapReduce program that builds a HIB from a list of images downloaded from the Internet.

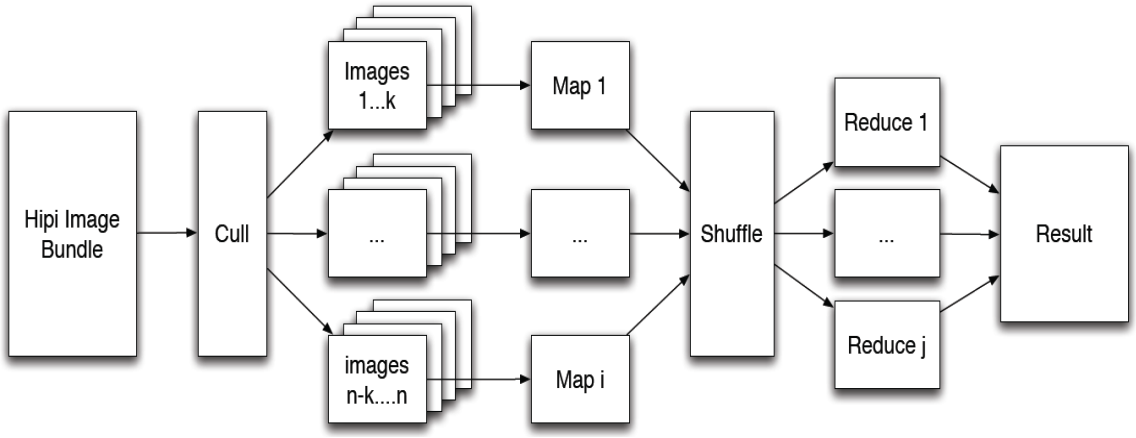


**Figure 2:** ***Processing of image in HIPI.***

## Objective

- To minimize job scheduling time.
- To exploit new hadoop scheduler, that exploits capabilities offered by heterogeneous cores with a single-multi processor for achieving a variety of performance objectives.
- To facilitate efficient and high-throughput image processing with map reduce style parallel programs typically executed on a cluster.

## System Architecture

The following Figure 3 shows the system architecture of job scheduling of different workloads effectively to improve performance using image processing application.

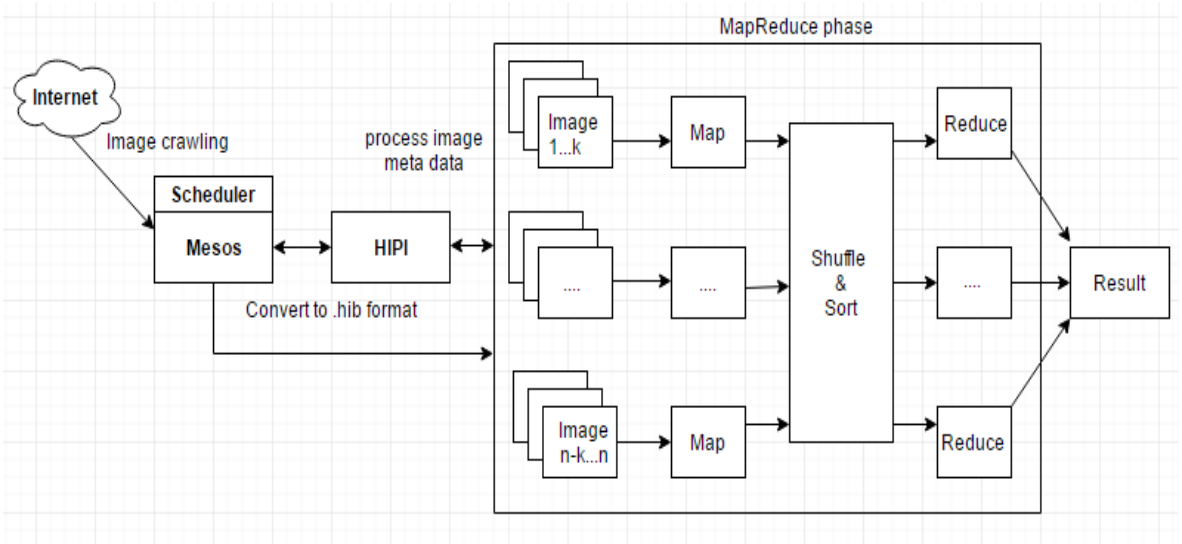


**Figure 3:** ***System Architecture.***

The system architecture consists of different modules to process images. Here the images are being crawled from the web. The scheduler Dyscale is placed on a Mesos framework which is used to efficiently manage the resources. The images downloaded are sent to HIPI to

convert them to.hib format which is in turn sent to MapReduce phase for generation of key-value pairs, processing and storing images.

**Work flow**

**Map Reduce Algorithm with Pseudo Code**

- Synchronization is perhaps the most tricky aspect of designing MapReduce algorithms (or for that matter, parallel and distributed algorithms in general).
- Other than embarrassingly-parallel problems, processes running on separate nodes in a cluster must, at some point in time, come together—for example, to distribute partial results from nodes that produced them to the nodes that will consume them.
- Within a single MapReduce job, there is only one opportunity for cluster-wide synchronization—during the shuffle and sort stage where intermediate key-value pairs are copied from the mappers to the reducers and grouped by key.

**Mesos algorithm**

- This technique for explicit reconciliation reconciles all non-terminal tasks, until an update is received for each task, using exponential backoff to retry tasks that remain unreconciled. Retries are needed because the master temporarily may not be able to reply for a particular task. For example, during master failover the master must re-register all of the slaves to rebuild its set of known tasks (this process can take minutes for large clusters, and is bounded by the --slave_reregister_timeout flag on the master).
- Steps:
- let start = now()
- let remaining = { T in tasks | T is non-terminal }
- Perform reconciliation: reconcile(remaining)
- Wait for status updates to arrive (use truncated exponential backoff). For each update, note the time of arrival.
- let remaining = { T in remaining | T.last_update_arrival() < start }
- If remaining is non-empty, go to 3.

**Hadoop Image Processing Interface (HIPI)**

The primary input object to a HIPI program is a Hipi Image Bundle (HIB). A HIB is a collection of images represented as a single file on the HDFS. The HIPI distribution

includes several useful tools for creating HIBs, including a MapReduce program that builds a HIB from a list of images downloaded from the Internet.

**Downloading and storing the image**

Hadoop uses the Hadoop Distributed File System (HDFS)[15] to store files in various nodes throughout the cluster. One of Hadoop's significant problems is that of small file storage.[16] A small file is one which is significantly smaller than HDFS block size. Large image datasets are made up of small image files in great numbers, which is a situation HDFS has a great deal of trouble handling. This problem can be solved by providing a container to group the files in some way. Hadoop offers a few options.

- HAR File
- Sequence File
- Map File

**Processing image bundle using Map Reduce**

- Hadoop MapReduce program handles input and output data very efficiently, but their native data exchange formats are not convenient for representing or manipulating image data.
- For instance, distributing images across map nodes require the translation of images into strings, then later decoding these image strings into specified formats in order to access pixel information.
- This is both inefficient and inconvenient. To overcome this problem, images should be represented in as many different formats as possible, increasing flexibility.
- The framework focuses on bringing familiar data types directly to user. As distribution is important in MapReduce, images should be processed in the same machine where the bundle block resides.
- In a generic MapReduce system, the user is responsible for creating InputFormat and RecordReader classes to specify the MapReduce job and distribute the input among nodes.

**Extracting image bundles using Map Reduce**

- In addition to creating and processing image bundles, the framework provides a method for extracting and viewing these images. Generally, Hadoop extracts images from an image bundle iteratively, inefficiently using a single node for the task.

- To address this inefficiency, we designed an Extractor module which extracts images in parallel across all available nodes. Distribution plays a key role in MapReduce;
- we want to make effective and efficient use of the nodes in the computing cluster. As previously mentioned in the description of the Processor module, a user working in a generic Hadoop system must again devise custom InputFormat and RecordReader classes in order to facilitate distributed image extraction.
- The Hadoop Image Processing Framework provides this functionality for the extraction task as well, providing much greater ease of use for the development of image processing applications.

**Related work**

In this work, we reduced the overall performance time of the Job by scheduling the prioritized job queues that are awaiting CPU time and by determining which job is to be taken from which queue and the amount of time to be allocated to the job. We designed a framework for creating virtual Hadoop clusters with different processing capabilities (i.e., clusters with fast and slow slots) and we implement a new scheduling scheme to support jobs with different performance objectives for utilizing the created virtual clusters and sharing their spare resources to achieve efficient workload performance.

**Performance Analysis**

The below Figure 4 shows the Output Graph indicating the CPU time taken to process the different sets with the modified Hadoop scheduler and the existing Fair scheduler.

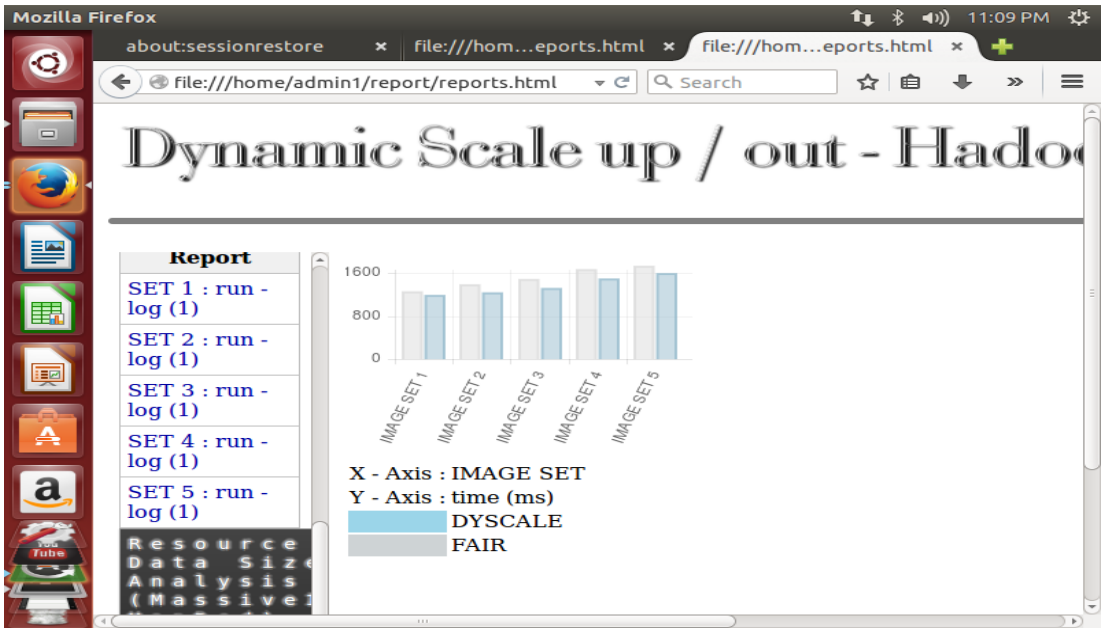


**Figure 4: Performance evaluations with respect to CPU time.**

The below Figure 5 shows the size of the different image sets which are being processed in MB.

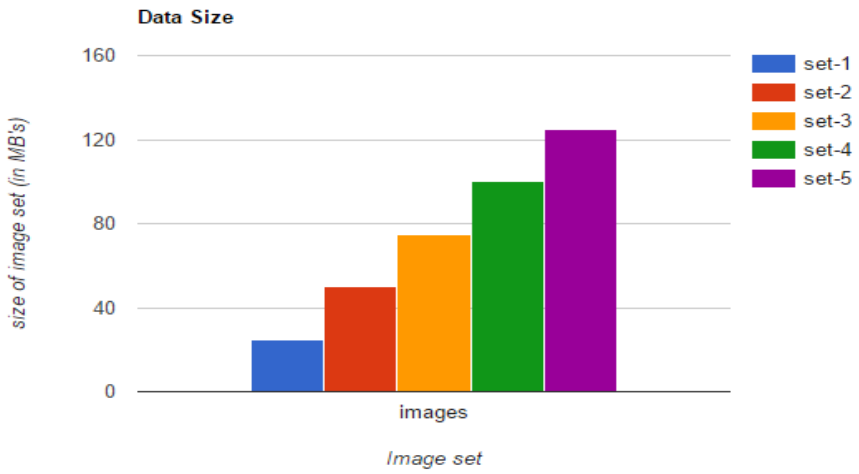


**Figure 5:** ***Data Size Versus Image Sets.***

## CONCLUSION AND FUTURE ENHANCEMENTS

We propose a new framework for scheduling using Dyscale Scheduler in combination with Mesos Resource manager for Image Processing in Hadoop. Dyscale is easy to use because the created virtual clusters have access to the same data stored in the underlying distributed file system, and therefore, any job and any dataset can be processed by either fast or slow virtual resource pools, or their combination. We implement scheduling algorithm for image processing. The comparison of the existing fair scheduler is done against the modified Hadoop scheduler. Our experimental results show 5-10% improvement in the CPU time using Dyscale in combination with Mesos compared to a default Fair scheduler in Hadoop. A comparison of the implementation results to our default hadoop configured with similar features show that they closely match. These results points to significant opportunities for image processing in hadoop.

In the future we plan to conduct more testbed experiments using Dyscale with the combination of Mesos and a variety of Job ordering scheduling policies for different kinds of applications for achieving fairness guarantees and better job completion time. We also plan to quantify the impact of the number of nodes used in order of improvement comparison in processing.We plan to exploit the possible migration features, compare our schedulers to more state of-the-art schedulers and under more challenging scenarios. We would like to introduce a more diverse set of applications including I/O bound applications; data skew work flows and explore new methods for improving their performance with respect to

throughput and time taken.The current dynamic tuning model will also be enhanced to address the challenges associated with the applications.

**REFERENCES**


1. T. White, Hadoop: The Definitive Guide. Yahoo press.
2. F. Ahmad et al., "Tarazu: Optimizing MapReduce of Heterogeneous Clusters," in Proceedings of ASPLOS, 2012.
3. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, 2008; 51(1).
4. M. Zaharia et al., "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in Proceedings of EuroSys, 2010.
5. Apache, "Capacity Scheduler Guide," 2010. [Online]. Available: http://hadoop.apache.org/common/docs/r0.20.1/ capacity scheduler.html
6. Z. Zhang, L. Cherkasova, and B. T. Loo, "Benchmarking approach for designing a mapreduce performance model," in ICPE, 2013; 253–258.
7. S. Rao et al., "Sailfish: A Framework For Large Scale Data Processing," in Proceedings of SOCC, 2012.
8. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a highlevel dataflow system on top of mapreduce: The pig experience," PVLDB, 2009; 2(2): 1414–1425.
9. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic Resource Inference and Allocation for MapReduce Environments," in Proc. of ICAC, 2011.
10. "Play It Again, SimMR!" in Proceedings of Intl. IEEE Cluster'2011.
11. S. Ren, Y. He, S. Elnikety, and S. McKinley, "Exploiting Processor Heterogeneity in Interactive Services," in Proceedings of ICAC, 2013.
12. H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking back and looking forward: power, performance, and upheaval," Commun. ACM, 2012; 55(7).
13. Bienia, S. Kumar, J. Singh, and K. Li, "The PARSEC benchmark suite:Characterization and architectural implications." in Technical Report TR-811-08, Princeton University, 2008.
14. "PassMark Software. CPU Benchmarks," 2013. [Online]. Available: http://www.cpubenchmark.net/cpu.php?cpu=Intel+ Xeon+E3-1240+%40+3.30GHz
15. F. Yan, L. Cherkasova, Z. Zhang, and E. Smirni, "Optimizing power and performance trade-offs of mapreduce job processing with heterogeneousmulti-core processors," in

Proc. of the IEEE 7th International Conference on Cloud Computing (Cloud'2014), June, 2014.

16. Verma et al., "Deadline-based workload management for mapreduce environments: Pieces of the performance puzzle," in Proc. of IEEE/IFIP NOMS, 2012.
17. R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-isa heterogeneous multi-core architectures for multithreaded workload performance," in ACM SIGARCH Computer Architecture News, 2004; 32(2).
18. K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in Proceedings of the 39th International Symposium on Computer Architecture, 2012.
19. M. Becchi and P. Crowley, "Dynamic thread assignment on heterogeneous multiprocessor architectures," in Proceedings of the 3rd conference on Computing frontiers, 2006.
20. Shelepov and A. Fedorova, "Scheduling on heterogeneous multicore processors using architectural signatures," in Proceedings of the Workshop on the Interaction between Operating Systems and Computer Architecture, 2008.
21. K. Van Craeynest and L. Eeckhout, "Understanding fundamental design choices in single-isa heterogeneous multicore architectures," ACM Transactions on Architecture and Code Optimization (TACO), 2013; 9(4).
22. M. Zaharia et al., "Improving mapreduce performance in heterogeneous environments," in Proceedings of OSDI, 2008.
23. Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment," in IEEE 10th International Conference on Computer and Information Technology (CIT), 2010.
24. R. Gandhi, D. Xie, and Y. C. Hu, "Pikachu: How to rebalance load in optimizing mapreduce on heterogeneous clusters," in Proceedings of 2013 USENIX Annual Technical Conference. USENIX Association, 2013.
25. J. Xie et al., "Improving mapreduce performance through data placement in heterogeneous hadoop clusters," in Proceedings of the IPDPS Workshops: Heterogeneity in Computing, 2010.
26. G. Gupta, C. Fritz, B. Price, R. Hoover, J. DeKleer, and C. Witteveen, "ThroughputScheduler: Learning to Schedule on Heterogeneous Hadoop Clusters," in Proc. of ICAC, 2013.

27. G. Lee, B.-G. Chun, and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, Hot Cloud, 2011.
28. J. Polo etal., "Performance management of accelerated mapreduce workloads in heterogeneous clusters," in Proceedings of the 41st Intl. Conf. on Parallel Processing, 2010.
29. W. Jiang and G. Agrawal, "Mate-cg: A map reduce-like framework for accelerating data-intensive computations on heterogeneous clusters," in Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, May 2012; 644–655.
30. Apache, "Apache Hadoop Yarn," 2013. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/ hadoop-yarn-site/YARN.html
31. Verma, L. Cherkasova, and R. H. Campbell, "Resource Provisioning Framework for Map Reduce Jobs with Performance Goals," Proc. of the 12th ACM/IFIP/USENIX Middleware Conference, 2011.