Job Schedulers for Machine Learning and Data Mining algorithms distributed in Hadoop

[1] Cornejo, Félix Martin; [2] Zunino, Alejandro; [3] Murazzo, María

1,3 Departamento de Informática, Universidad Nacional de San Juan;

2 Instituto Superior de Ingeniería de Software de Tandil, Universidad Nacional del centro de la provincia de Buenos Aires

1 fmartin.cornejo@gmail.com; 2 azunino@gmail.com; 3 maritemurazzo@gmail.com

Abstract [1]

The standard scheduler of Hadoop does not consider the characteristics of jobs such as computational demand, inputs / outputs, dependencies, location of the data, etc., which could be a valuable source to allocate resources to jobs in order to optimize their use. The objective of this research is to take advantage of this information for planning, limiting the scope to ML / DM algorithms, in order to improve the execution times with respect to existing schedulers. The aim is to improve Hadoop job schedulers, seeking to optimize the execution times of machine learning and data mining algorithms in Clusters.

Keywords: Big Data, Hadoop, schedulers of Hadoop, ML/DM algorithms, machine learning.

1. Introduction

Recently, the valuable knowledge that can be extracted from the analysis of sets of large-scale data, has given rise to the so-called "Big Data". The term "Big Data" [1,2] refers to a collection of large data sets that can not be processed using traditional database administration tools. This generated numerous scientific challenges such as how to provide support for storage, manipulate and then retrieve information. In this sense, it is necessary to have capable infrastructures to process large volumes of data, in acceptable times and with limited computational resources. Solutions typically rely on concepts of parallel and distributed computing, where through Clusters and Computational Grids [3] allow processing Big Data at relatively low costs [4,5].

Although the fact that parallel and distributed computing appears as one of the most interesting alternatives to store and manipulate Big Data, some of its characteristics prevent its use by part of common users [6]. Aspects such as data dependencies, integrity, load balancing, planning and fault tolerance, although extremely difficult toaverage programmers, are crucial for Big Data solutions to be operational. For this reason, several frameworks have been proposed that abstract these aspects and provide high-level solutions for users and programmers [5,6,7,8,9,10,11]. Some of those frameworks they are based on specific programming paradigms such as Fork-Join, MapReduce and MPI. Recently, the MapReduce paradigm attracted attention due to its applicability in parallel computing, being widely used by Google in its distributed file system GFS [12]. The great novelty of the paradigm and its implementation in a framework consists of in that it conceals to the programmer great part of the complexity of parallelization and distribution. Thus, programmers can focus on the functionality of their programs, while the framework abstracts the complexity and control the underlying computational infrastructure. One of the most successful implementations of MapReduce is Apache Hadoop In addition, it provides solutions to store, manipulate and extract Big Data information in different ways. Around Hadoop has flourished an ecosystem of solutions for specific problems. Some examples are Pig, which facilitates the analysis of large datasets, Spark that speeds up MapReduce using structures in RAM and Mahout that aims to scale data mining and machine learning algorithms.

While these efforts have had a significant impact on research and industrial environments the diversity of uses of Hadoop has led to underperforming results [13]. This is due, among other factors, to the scheduler, component responsible for assigning the computational resources of a Cluster to the works, is one of the aspects that most influence the performance of Hadoop. Originally, Hadoop includes three schedulers: FIFO, Fair and Capacity. The default scheduler is FIFO, which queues jobs in order of arrival and executes them. The Fair scheduler, developed by Facebook, seeks to allocate equitable Cluster resources to tasks. The Capacity scheduler was developed by Yahoo! to ensure equitable allocation for a large number of Cluster users.

Although these Hadoop schedulers give flexibility so that users can optimize the execution of their works to the Cluster, the resulting performance is often less than expected [14] resulting in not only delays in execution time, but also low use of Cluster resources. Some efforts have proposed ad-hoc schedulers for Hadoop that consider different aspects of both tasks and computational resources available.

In this paper we analyze and discuss previous efforts on improving Hadoop schedulers and present our current research to improve Hadoop schedulers. The rest of this paper is organized as follows. In Section 2 we provide background about Hadoop and its built-in schedulers. Then, Section 3 overviews and discuss the most representative research on Hadoop scheduling. Section 4 proposes possible current and future research towards improving Hadoop scheduling for machine learning and data mining tasks.

2. Background

A significant amount of research has been done in the field of Hadoop Job scheduling; however, there is still a need for research to overcome some of the challenges regarding scheduling of jobs in Hadoop clusters. Industries estimate that 20% of the data is in structured form while the remaining 80% of data is in semi structured form. This is a big challenge not only for volume and variety but also for data processing, which can lead to problems for I/O processing and job scheduling. Fig. 1 shows the architecture of a Hadoop distributed system.

As we know, this is an era of Big Data where machines process a significant amount of data, in the range of terabytes or petabytes, using various applications in fields such as science, business, and commerce. Such applications require a considerable amount of I/O processing and spend most of the time doing I/O processing, which is a major part of job scheduling. It is reported that at the Facebook and Microsoft Bing data centers, I/O processing requires 79% of the jobs' duration and 69% of the resources. Therefore, here we present a comprehensive study of the most representative Hadoop schedulers.

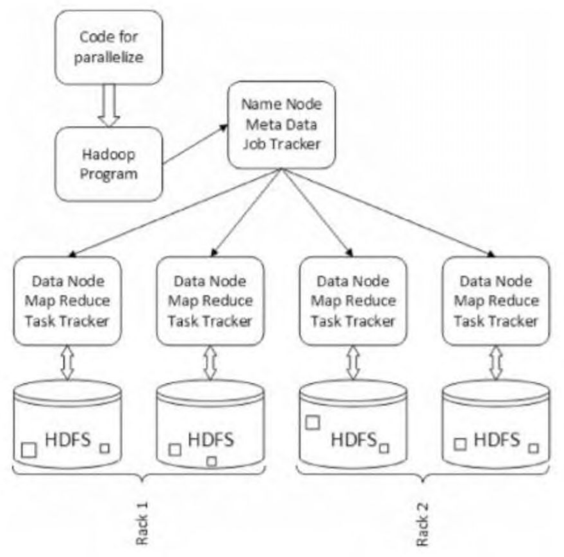

Fig 1. Hadoop distributed system architecture

2.1. Apache Hadoop Ecosystem

The Apache Hadoop is a framework that allows the processing of large volumes of data through Cluster systems, using a simple programming model. In addition, its design allows the scalability from a few nodes to thousands of nodes in an agile way, achieving high efficiency in vertical scaling.

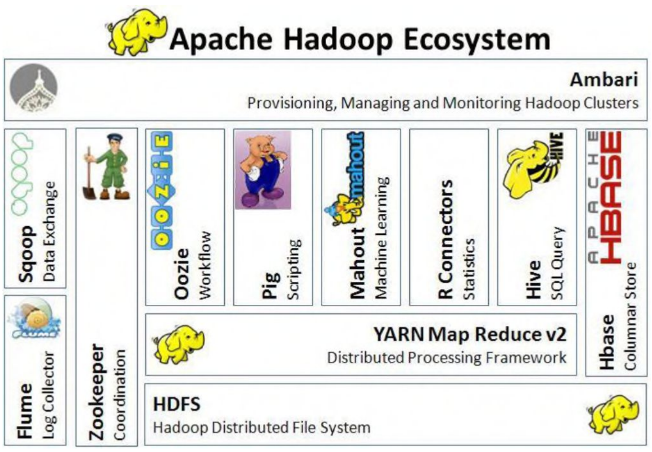

Fig. 2. Hadoop Ecosystem.

Hadoop is a distributed system that uses a Master-Slave architecture. In addition, it provides a distributed file system for storing data called the Hadoop Distributed File System (HDFS) and the Map Reduce programming paradigm to perform the calculations.

Hadoop is a very diverse ecosystem, which grows day after day. Hadoop has grown into a large family of solutions for the storage, management, interaction and analysis of large data, integrated into a rich open source ecosystem created by the community of the Apache Software Foundation (Fig. 2).

2.2. Job schedulers in Hadoop

The aim of job scheduling included in hadoop [15] is to enable faster processing of jobs and to reduce the response time as much as possible by using better techniques for scheduling depending on the jobs, along with the best utilization of resources. FIFO scheduling is default scheduling mode in Hadoop; the jobs coming first get higher priority than those coming later. In some situations, this type of scheduling has a disadvantage, that is, when longer jobs are scheduled prior to shorter jobs, it leads to starvation.

Fair scheduling shares the resources equally among all jobs. Capacity scheduling was introduced by Yahoo. It maximizes the utilization of resources and throughput in Clusters. LATE [16] scheduling policy was developed to optimize the performance of jobs and to minimize the job response time by detecting slow running processes in a Cluster and launching equivalence processes as the background. Facebook uses Delay scheduling to achieve better performance and lower response time for map tasks by applying changes to MapReduce. In deadline scheduler, the deadline constraints are specified by the user before scheduling the jobs in order to increase system utilization.

Resource aware scheduling improves resource utilization; it uses node, master node, and worked node to complete job scheduling. In matchmaking scheduling [17], each node is marked by the locality marker, which ensures that every node gets an equitable chance to seize a local task. Through this scheduling, high data locality and better cluster utilization is achieved.

There have already been a few review papers on job scheduling algorithms for Big data processing. [14] presented a comparative study on job scheduling methods and discussed their strengths and weakness. [18] presented a review on scheduling algorithms and provided guidelines for the improvement of scheduling algorithms in Hadoop MapReduce. [19] presented a survey on Big data management and discussed various scheduling algorithms in Hadoop. They also discussed the latest advancements related to scheduling algorithms. [20] presented a paper on the scheduling policies for Hadoop and performed a comparative study on MapReduce optimization techniques.

2.3 Major challenges for job scheduling in Big data

There is a clear need for efficient scheduling algorithms for the management of Big data on various nodes in Hadoop clusters, in particular for supporting the execution of machine learning and data mining algorithms. There are various factors that affect the performance of scheduling policies such as data volume (storage), format of data sources (data variety), speed (data velocity), security and privacy, cost, connectivity, and data sharing. To achieve better utilization of resources and management of Big data, new scheduling policies should be designed. The next sections describe some of the main challenges that face job scheduling.

2.3.1. Data volume (storage)

An important problem of Big data is that it is very huge and includes data that is in unstructured formats, which makes it difficult to organize the data for analysis. Data includes both structured and unstructured data; the storage of unstructured data is not an easy task.

2.3.2. Format of data sources (data variety)

Data comes into Big data from various homogeneous as well as heterogeneous resources, and this causes many problems due to the heterogeneity of data resources, data format, and infrastructure

2.3.3. Speed (data velocity)

Today, everybody expects everything to be done instantaneously. The speed is an important issue in Big data. Speed of Big data is restricted by various problems such as query/retrieval problem, import/ export problem, real time/offline problem, and statistical analysis problem.

2.3.4. Connectivity and data sharing

Data sharing and connectivity are still issues that need to be considered. At present, a majority of the data points are not yet connected. There are various issues in Big data related to connectivity and data sharing such as data standard and interfaces, access permissions, and shared protocols.

2.3.5. Cost

Cost is also an issue in Big data. The cost up-gradation issues in Big data are cost comparison between master and slave nodes, and upgrade or modification of nodes.

3. Custom Hadoop Schedulers

Some works have studied the behavior of the schedulers included with Hadoop and proposed improvements [14,21] proposed an analytical model for FIFO and Fair schedulers based on experimental measurements and source code analysis. For a class of Map jobs with heavy-tailed service times, the authors found problems of starvation with the Fair scheduler due to the way of launching Reduce tasks. To reduce that, the Coupling scheduler was proposed that couples the advance of Mappers and Reducers, as well as optimizing the locations for both, achieving mitigation of the problem of starvation and improving the data locality.

In [18] proposed an improvement for the FIFO scheduler that takes into account the location of the data and characteristics of the works. In essence, it adopts different policies for jobs linked to CPU or input / output based on data locality. As a result, it is possible to reduce the data transfer and the execution time of the works. Similarly, [19] designed a scheduler based on dynamic priorities whose objective is to reduce the latency for jobs of variable length. The work focuses on intensive data work, so it tries to maintain the data location during execution.

Other authors have investigated how to deal with Clusters formed by heterogeneous hardware. [20] presented a scheduler that uses the heterogeneity and mix of workloads to optimize jobs that use the same data sets. Although the scheduler has only been simulated, it is based on the idea of looking for the best node of the Cluster to run, based on the idea that a large percentage of MapReduce jobs have the same characteristics in terms of CPU, network and disk usage. The scheduler classifies Cluster jobs as linked to CPU or input / output, similarly the nodes classifies them as good for CPU or input / output and then performs the assignment. [22] proposed a self-adaptive scheduler that takes into account that different nodes require different time to complete the same tasks due to their differences such as processing, communication, architectures and memory. The scheduler uses historical information about each cluster node to adjust execution parameters and discover slow tasks. Thus, you can launch backup tasks in other nodes, also taking into account the data locality. [23] develops criteria to schedule schedulers based on restrictions of deadlines specified by the user and discusses the implementation and preliminary evaluation of a scheduler of Deadlines Restrictions for Hadoop that ensures that only jobs whose deadlines can be met are planned for execution.

In [24] on the contrary, the authors focus on the heterogeneity of the tasks, proposing a scheduler that uses information such as job income rates and average execution times to make better decisions.

In [25] a quantitative approach is adopted where a first detailed study of the behavior of several applications on Hadoop that run on four different hardware configurations, to ensure that the data associated with the jobs are available locally to a cluster in a multi-cluster implementation. On the other hand, the work of [26], observe the changes in the load of the nodes to assign jobs more intelligently.

In [27] the conflict between locality and equity is addressed, and a simple algorithm called delay programming is proposed: when the work that must be scheduled according to equity can not start a local task, it waits a small amount of time, allowing others jobs start tasks in their place. It was verified that the schedule of delays reaches an almost optimal data location in a variety of workloads and can increase performance, while preserving fairness. In addition, the simplicity of delay programming makes it applicable under a wide variety of programming policies in addition to fair exchange.

Finally, [28] studied the interactions between jobs and their intermediate results to group multiple jobs into one and thus reduce the number of queries to access shared intermediate data.

4. Current and Future Research

The purpose of this research work is to study the improvement of time in the exploration and analysis of data sets using an improvement over the schedulers implemented in Hadoop for machine learning and data mining jobs. New schedulers algorithms will also be implemented that will make a better assignment of jobs to the cluster, obtaining advantages in terms of better use of resources and execution times with respect to schedulers such as FIFO schedulers. Machine learning and data mining algorithms have a distinctive characteristic: they involve executing the same algorithm many times over different parts of a input dataset.

The CPU load, network usage and input / output will be analyzed, by executing jobs generated by machine learning and data mining algorithms when executed on Hadoop with large and publicly available datasets. It is intended in a first stage to derive models or profiles of resource use of the aforementioned works considering a space of variability in the data sets and using two Clusters with different hardware characteristics.

The methodology will be similar to that of [29, 30,31] where once the characteristics of the works have been outlined, a simulator developed by the working group will be adapted in order to analyze planning alternatives that improve the execution times and resource utilization. Similar to [32], where the scheduler uses estimated information on availability of resources to achieve better performance in the execution of tasks in desktop Grids, in this plan estimates will be used on the tasks, starting from the base that the algorithms have well-known behavior and temporal/spatial complexity.

The starting point to obtain job data and then outline its characteristics will be existing implementations of machine learning and data mining algorithms. A good alternative for this is Mahout[2] that offers Hadoop implementations of Naive Bayes classification, k-means clustering, collaborative filtering and recommendation algorithms based on ALS (Alternating Least Squares), among others. Executions of these algorithms using data sets such as those available in the UC Irvine Machine Learning Repository[3] will provide data on the use of resources and performance. From these data, jobs will be grouped according to their demands, resulting performance, input data characteristics, etc.

In a first stage, optimized schedulers for the profiled jobs will be simulated. Once obtained results in this stage, the improvements in executions in the Cluster available in the UNSJ of San Juan and in the ISISTAN of Tandil will be verified. Simulations / tests will be carried out again until satisfactory solutions are achieved.

5. References

[1] Paul C. Zikopoulos, Chris Eaton, Drik deRoos, Thomas Deutsch, and George Lapis. Understanding Big Data - Analytics for Enterprise Class Hadoop and Streaming Data. 2012.

[2] Alejandro Corbellini, Cristian Mateos, Alejandro Zunino, Daniela Godoy, and Silvia N. Schiaffino.Persisting big-data: The nosql landscape. Inf. Syst., 63:1–23, 2017b. doi: 10.1016/j.is.2016.07.009. URL https://doi.org/10.1016/j.is.2016.07.009.

[3] Daniel S. Katz and Xiaobo Zhou. Leading-edge research in cluster, cloud, and grid computing: Best papers from the ieee/acm {CCGrid} 2015 conference. Future Generation Computer Systems, 72:78 – 80, 2017. ISSN 0167-739X. doi:https://doi.org/10.1016/j.future.2016.09.016. URL http://www.sciencedirect.com/science/article/pii/S0167739X16303557.

[4] A. Tommasel, A. Corbellini, D. Godoy, and S. Schiaffino. Personality-aware followee recommendation algorithms: An empirical analysis. Engineering Applications of Artificial Intelligence, 51: 24–36, 2016.

[5] A. Corbellini, C. Mateos, D. Godoy, A. Zunino, and S. Schiaffino. An architecture and platform for developing distributed recommendation algorithms on large-scale social networks. Journal of Information Science, 41(5):686–704, 2015.

[6] Cristian Mateos, Alejandro Zunino, and Marcelo Campo. Jgrim: An approach for easy gridification of applications. Future Generation Computer Systems, 24(2):99 – 118, 2008a. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2007.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0167739X07000854.

[7] Pieter Hijma, Rob V. van Nieuwpoort, Ceriel J.H. Jacobs, and Henri E. Bal. Generating synchronization statements in divide-and-conquer programs. Parallel Computing, 38(1):75 – 89, 2012. ISSN 0167-8191. doi: http://dx.doi.org/10.1016/j.parco.2011.10.007. URL: http://www.sciencedirect.com/science/article/pii/S0167819111001384. Extensions for Next-Generation Parallel Programming Models

[8] Alejandro Corbellini, Daniela Godoy, Cristian Mateos, Silvia Schiaffino, and Alejandro Zunino. DPM: A novel distributed large-scale social graph processing framework for link prediction algorithms. Future Generation Computer Systems, 2017a. En prensa.

[9] M. Arroqui, J. Rodriguez Alvarez, H. Vazquez, C. Machado, Cristian Mateos, and Alejandro Zunino. JASAG: A gridification tool for agricultural simulation applications. Concurrency and Computation: Practice and Experience, 27(17):4716–4740, 2015.

[10] Rob V. van Nieuwpoort, Jason Maassen, Rutger Hofman, Thilo Kielmann, and Henri E. Bal. Ibis: An efficient java-based grid programming environment. In Proceedings of the 2002 Joint ACMISCOPE Conference on Java Grande, JGI '02, pages 18–27, New York, NY, USA, 2002. ACM. ISBN 1-58113-599-8. doi: 10.1145/583810.583813. URL http://doi.acm.org/10.1145/583810.583813.

[11] Craig Chambers, Ashish Raniwala, Frances Perry, Stephen Adams, Robert R. Henry, Robert Bradshaw, and Nathan Weizenbaum. Flumejava: Easy, efficient data-parallel pipelines. SIGPLAN

[2] http://mahout.apache.org

[3] http://archive.ics.uci.edu/ml/index.php

Not., 45(6):363–375, June 2010. ISSN 0362-1340. doi: 10.1145/1809028.1806638. URL http://doi.acm.org/10.1145/1809028.1806638.

[12] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP ’03, pages 29–43, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945450. URL http://doi.acm.org/10.1145/945445.945450.

[13] Ivanilton Polato, Reginaldo RÃc , Alfredo Goldman, and Fabio Kon. A comprehensive view of hadoop research–a systematic literature review. Journal of Network and Computer Applications, 46:1 – 25, 2014. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2014.07.022. URL: http://www.sciencedirect.com/science/article/pii/S1084804514001635

[14] D. Yoo and K. M. Sim. A comparative review of job scheduling for mapreduce. In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pages 353–358, Sept 2011. doi: 10.1109/CCIS.2011.6045089.

[15] J.V. Gautam, H.B. Prajapati, V.K. Dabhi, S. Chaudhary, A survey on job scheduling algorithms in big data processing, in: IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT’15), Coimbatore, 2015, pp. 1–11.

[16] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, “Improving MapReduce performance in heterogeneous 46 environments”, in: Proc. 8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, San Diego,USA, Dec. 2008.

[17] He, C., Lu, Y., & Swanson, D. (2011). Matchmaking: A New MapReduce Scheduling Technique. In 2011 IEEE Third International Conference on Cloud Computing Technology and Science (pp. 40–47). IEEE. https://doi.org/10.1109/CloudCom.2011.16

[18] Y. Tao, Q. Zhang, L. Shi, and P. Chen. Job scheduling optimization for multi-user mapreduce clusters. In 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, pages 213–217, Dec 2011. doi: 10.1109/PAAP.2011.33.

[19] P. Nguyen, T. Simon, M. Halem, D. Chapman, and Q. Le. A hybrid scheduling algorithm for data intensive workloads in a mapreduce environment. In 2012 IEEE Fifth International Conference on Utility and Cloud Computing, pages 161–167, Nov 2012. doi: 10.1109/UCC.2012.32.

[20] Sreedhar, C., Kasiviswanath, N., & Chenna Reddy, P. (2018). Performance Enhancement of Hadoop for Big Data Using Multilevel Queue Migration (MQM) Technique (pp. 331–342). https://doi.org/10.1007/978-981-10-8237-5_32

[21] Jyoti V. Gautam, Harshad kumar B. Prajapati, Vipul K. Dabhi, Sanjay Chaudhary, A survey on job scheduling algorithms in big data processing, IEE Conf. Pap. (March 2015), http://dx.doi.org/10.1109/ICECCT.2015.7226035

[22] Jian Tan, Xiaoqiao Meng, and Li Zhang. Delay tails in mapreduce scheduling. In Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS ’12, pages 5–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1097-0. doi: 10.1145/2254756.2254761. URL http://doi.acm.org/ 10.1145/2254756.2254761.

[23] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In 2010 10th IEEE International Conference on Computer and Information Technology, pages 2736–2743, June 2010. doi: 10.1109/CIT.2010.458.

[24] Kc, K., & Anyanwu, K. (2010). Scheduling Hadoop Jobs to Meet Deadlines. In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (pp. 388–392). IEEE. https://doi.org/10.1109/CloudCom.2010.97

[25] Aysan Rasooli and Douglas G. Down. An adaptive scheduling algorithm for dynamic heterogeneous hadoop systems. In Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, CASCON ’11, pages 30–44, Riverton, NJ, USA, 2011. IBM Corp. URL http://dl.acm.org/citation.cfm?id=2093889.2093893.

[26] Krish, K. R., Anwar, A., & Butt, A. R. (2014). [phi]Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems* (pp. 255–264). IEEE. https://doi.org/10.1109/MASCOTS.2014.40

[27] Hsin-Han You, Chun-Chung Yang, and Jiun-Long Huang. A load-aware scheduler for mapreduce framework in heterogeneous cloud environments. In Proceedings of the 2011 ACM Symposium on Applied Computing, SAC '11, pages 127–132, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982218. URL http://doi.acm.org/10.1145/1982185.1982218.

[28] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). Delay scheduling. In *Proceedings of the 5th European conference on Computer systems - EuroSys '10* (p. 265). New York, New York, USA: ACM Press. https://doi.org/10.1145/1755913.1755940

[29] Tomasz Nykiel, Michalis Potamias, Chaitanya Mishra, George Kollios, and Nick Koudas. Mrshare: Sharing across multiple queries in mapreduce. Proc. VLDB Endow., 3(1-2):494–505, September 2010. ISSN 2150-8097. doi: 10.14778/1920841.1920906. URL http://dx.doi.org/10.14778/1920841.1920906.

[30] Juan Manuel Rodriguez, Cristian Mateos, and Alejandro Zunino. Energy-efficient job stealing for cpuintensive processing in mobile devices. Computing, 96(2):87–117, Feb 2014. ISSN 1436-5057.doi:10.1007/s00607-012-0245-5.URL http://dx.doi.org/10.1007/s00607-012-0245-5.

[31] Matías Hirsch, Juan Manuel Rodriguez, Cristian Mateos, and Alejandro Zunino. A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids. J. Grid Comput., 15(1):55–80, 2017. doi: 10.1007/s10723-016-9387-6. URL https://doi.org/10.1007/ s10723-016-9387-6.

[32] Sergio Ariel Salinas, Carlos García Garino, and Alejandro Zunino. PFS: A productivity forecasting system for desktop computers to improve grid applications performance in enterprise desktop grid. Computing and Informatics, 33(4):783–809, 2014.URL http://www.cai.sk/ojs/index.php/cai/article/view/878