Towards an Ontology-Based Semantic Approach to Tuning Parameters to Improve Hadoop Application Performance

Ailton Bonifacio
Federal University of Paraná - UFPR
Curitiba, Brazil
ailtons@inf.ufpr.br

Andre Menolli
Universidade Estadual do Norte do Paraná - UENP
Bandeirantes, Brazil
menolli@uenp.edu.br

Fabiano Silva
Federal University of Paraná - UFPR
Curitiba, Brazil
fabiano@inf.ufpr.br

***Abstract*—Hadoop MapReduce assists companies and researchers to deal with processing large volumes of data. Hadoop has a lot of configuration parameters that must be tuned in order to obtain a better application performance. However, the best tuning of the parameters is not easily obtained by inexperienced users. Therefore, it is necessary to create environments that promote and motivate information sharing and knowledge dissemination. In addition, it is important that all acquired knowledge be organized to be reused faster, easily and efficiently whenever necessary. This paper proposes an ontology-based semantic approach to tuning parameters to improve Hadoop application performance. The approach integrates techniques from machine learning, semantic search and ontologies.**

Keywords—Hadoop MapReduce; Hadoop performance; ontology; parameters turning

I. Introduction

MapReduce, a framework introduced by Google Inc., is a programming model designed to process large volumes of data in parallel in a distributed environment (clusters) [1].

Hadoop [2] is an open-source implementation of MapReduce and is currently one of the most widely used implementations for processing large amounts of data. Its programming model is divided into two main phases: the Map and Reduce phases. The reduce phase is divided into phases shuffle, sort and reduce. These are the intermediate phases.

A large number of companies, research centers and researchers in general have used Hadoop MapReduce imple-mentations in applications such as data mining, production of reports, indexing Web pages, analysis of log files, machine learning, financial analysis, scientific simulation, statistics, research in bioinformatics and others [3]. Therefore, Hadoop has been studied to identify several aspects involving the tuning and performance.

Current studies show that the performance of Hadoop MapReduce jobs depends on the cluster configuration, input data type and job configuration settings [1], [3], [4]. Furthermore, some studies point out that most of the applications running on Hadoop, show a large difference between the behavior of Hadoop applications in map and reduce phases. For instance, in some cases the Map phase is computationally intensive, in other is Reduce and others both [4].

This paper proposes an ontology-based semantic approach to improve Hadoop performance. The goal is to provide an environment for tuning parameters of the Hadoop configuration based on semantic knowledge through ontologies. Thus, users would get a better performance from their Hadoop applications. The work focuses on Hadoop configuration parameters that have influence on the job performance. The approach explores workload and cluster characteristics, and analyzes the log history of previous executions. Then, based upon the stored knowledge, it predicts tunable configuration parameters and finds the best tuning for them, in order to improve the Hadoop application performance against default values of the parameters. The T-Box statements of the ontology describe a set of concepts, and properties for these concepts, obtained from a study that evaluated 40 papers that specifically address the configuration parameters impacting the performance of Hadoop.

We identified and explored, by a systematic review, how the tuning parameters of Hadoop impact on the entire system performance. More specifically, the research answered questions about: (1) which Hadoop configuration parameters has influences and impact on system performance, (2) which parameters are influenced by Hadoop phases and (3) which parameters are influenced by workloads characteristics. Thus, it was possible gather knowledge to make a conceptual mapping and propose an ontology that is the basis for this approach.

This paper is organized as follows. In Section II, a background on Hadoop framework and information on the main concepts related to Hadoop configuration parameters are presented. Section III introduces the proposed environment as well as its main structure and the chosen knowledge representation. Section IV presents the final considerations of this paper.

II. Background

This section presents the main concepts related to Hadoop configuration parameters which can be tuned to improve application performance.

A. Hadoop MapReduce

MapReduce is a framework of distributed functional programming. Its processing occurs in two phases: Map

Copyright © Authors

ISSN: 2203-1731

and Reduce. MapReduce framework divides the work into a set of independent tasks and manages communications and data transfers between nodes in the cluster and the related parts of the system.

According to their characteristics, the MapReduce framework is well suited to run in a cloud environment. The cloud-based computing services are already available for use via Hadoop MapReduce [5].

Apache Hadoop [2] is the well known and most widely used implementation of MapReduce. However, the improvement of application performance is directly related to the setting of the Hadoop's parameters. To ascertain the relationship between parameter values and a good performance is not a simple task, even for experienced users.

The Apache Hadoop software library enable the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers up to thousands of machines, each offering local computation and storage. The library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers.

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

The functions *map* are distributed across multiple machines by partitioning the input data into a set *M* of splits. These splits may be processed in parallel by different machines. Moreover, the functions *reduce* are distributed by partitioning the intermediate keys *R* splits by a *hash (key) mod R* function. This number of splits *R* and the partitioning function are defined by the user.

The Hadoop Distributed File System (HDFS) is a distributed filesystem that is designed for the Hadoop MapReduce framework and has a master-slave architecture. Its cluster consists of a single *NameNode* and a master server that manages the *namespace* of file system and regulates the access of clients. In addition, there are a number of *DataNodes* (usually one by node) that manages the storage nodes. Internally, a file is divided into one or more blocks. These blocks are stored in a set of *DataNodes*. The *DataNodes* are responsible for serving the requests of reads and writes in the files system. Moreover, the *DataNodes* perform the creation, deletion and replication blocks on instructions from *NameNode*.

1) Hadoop Parameters Review: In order to run a program such as a job in Hadoop, an object configuration job is created and the parameters of the job must be specified. That is, since the job was created for MapReduce, it must enabled to run in scale on a cluster [4]. Therefore, in order to avoid some problems such as under-performance, it is necessary to tuning some parameters.

Hadoop provides many configurable parameters that lead to better or worse performance on a cluster. Most of these parameters take effect on execution of the job and cluster. In the case of large clusters of machines, these parameters become quite important and provide an indication of the performance of this cluster in relation to the submitted job. Thus, for better system performance, the framework must be further optimized as much as possible.

Hadoop has over 190 parameters that can be specified to control the behavior of a MapReduce job. Among them, more than 25 parameters can impact the performance of the task [5]. Some configuration parameters aim to control various aspects of the tasks behavior at runtime. Some of these aspects include the allocation and use of memory, competition, optimize I/O and network bandwidth usage. If the parameters are not specified, default values are assumed.

The researched papers provided an overview of studies which apply to performance of Hadoop systems in regard to their configuration parameters. This research answered questions about which Hadoop configuration parameters has influences and impact on system performance, which parameters are influenced by Hadoop phases and which parameters are influenced by workloads characteristics.

The results of that study were divided into three main parts:

- Configuration Parameters: focuses on identifying parameters which were used in the detected studies;
- Hadoop Phases x Parameters: focuses on which parameters are affected by each Hadoop phase;
- Workload Characeristics x Parameters: focuses on identifying which parameters are related to the workloads characteristics.

It were identified about 29 configuration parameters which according to studies analyzed, impact the system performance. The Table I summarizes the parameters, Hadoop phases in which the parameters are critical and the main characteristics of the application.

In the column *Parameter Phase*, we classify the parameters which influence the phases of Hadoop: Map, Reduce and intermediate sub-phases of the Reduce phase. The sub-phases are classified as Merge/Shuffle phase. In addition, Core Job was listed as a phase that represents those parameters that directly control essential functions of the job. The parameter *dfs.block.size* was classified by phase *Number of maps* just to demonstrate that the value this parameter is what defines the number of map tasks.

Optimizing Hadoop Performance through the characteristics of workloads are important in order to make full use of cluster resources (CPU, memory, IO and network - see Table I, in the column *Workload Characteristics*).

Among other examples, to set certain parameters can reduce the IO cost and network transmission, but it can cause a CPU overhead. If we configure the *mapred.compress.map.output* to *true* (default is *false*), it will decrease the size of data transferred from the Mapper to the Reducers, but it will add more processing in the data compression and decompression process [4]. Furthermore, some parameters are correlated with each other. The relation between the *io.sort.mb* parameter (the amount of buffer space in megabytes to use when sorting streams) and *mapred.child.java.opts* parameter (Java control options for all mapper and reducer tasks) is an example. The upper limit of the former is smaller than the second size. Configure sufficient Map and Reduce slots to maximize CPU utilization and configure the Java heap size (for Map and Reduce JVM processes) so that ample memory is still available

Copyright © Authors ISSN: 2203-1731

TABLE I: Configuration Parameters

Parameter Category	Parameter Level	Parameter Phase	Workload Characteristics	Parameter
Hadoop	Cluster Level	Merge/ Shuffle	IO	dfs.replication
				dfs.replication.interval
				mapred.min.split.size
			Number of maps	dfs.block.size
	Job Level	Core Job	CPU	mapred.compress.map.output
				mapred.job.reuse.jvm.num.tasks
				mapred.output.compression.type
				mapred.reduce.slowstart.completed.maps
			Memory	mapred.child.java.opts
			Network	mapred.reduce.parallel.copies
		Map	CPU	mapred.map.tasks.speculative.execution
				mapred.tasktracker.map.tasks.maximum
		Merge/ Shuffle	CPU	mapred.map.output.compression.codec
			IO	io.file.buffer.size
				io.sort.factor
				io.sort.mb
				io.sort.record.percent
				io.sort.spill.percent
			Memory	mapred.inmem.merge.threshold
				mapred.job.reduce.input.buffer.percent
				mapred.job.shuffle.input.buffer.percent
				mapred.job.shuffle.merge.percent
		Reduce	CPU	mapred.reduce.tasks.speculative.execution
				mapred.tasktracker.reduce.tasks.maximum
			Memory	mapred.tasktracker.tasks.maxmemory
Workload	Job Level	Map	CPU	mapred.map.tasks
			IO	mapreduce.combine.class
				min.num.spills.for.combine
		Reduce	CPU; IO	mapred.reduce.tasks

for the OS kernel, buffers, and cache subsystems are other examples [6].

Workload characterization by CPU, memory, IO and network (via benchmarks) is essential before performing any tuning parameters. Furthermore, it is important that we deploy the parameters that are related to the characteristics of workloads. This allows identifying the parameters that can be set to obtain best performance.

B. Questions and Considerations

The answers for the questions presented in Section 1 are of great value for modeling the ontology for Hadoop. To repeat the questions: (1) which Hadoop configuration parameters have influences and cause impact on system performance, (2) which parame-ters are influenced by Hadoop phases and (3) which parameters are influenced by workloads characteristics.

Answering the question about which Hadoop configuration parameters has influences and impact on system performance, it is clear that about 29 Hadoop configuration parameters are those which have more impact on the system performance. From those 30 parameters, 10 parameters were covered over 65% of the papers. We observed that the most discussed parameters on those works are *mapred.reduce.tasks* (The suggested number of reduce tasks for a job) with 31 papers of the 40 papers surveyed, mapred.map.tasks (The suggested number of map tasks for a job) with 23 papers, *dfs.block.size* (The basic block size for the file system) with 26 papers and *io.sort.factor* (The number of map output partitions to merge at a time) with 18 papers. These parameters are those that allow the framework to attempt to provide data locally for the task that processes the split. Although some parameters have not been widely exploited by most papers, studies have shown their importance in relation to performance.

Answering the question about which parameters are influenced by Hadoop phases, we identify the parameters that are affected by each Hadoop phase. This will also allow the targeting of tuning the parameters identified at each stage of Hadoop, if required. This approach would be rather useful if we know the application characteristics and at what phase it is CPU, IO or network bound, for example. Thus, the tuning of the parameters would be directed by this prior knowledge.

Answering the question about which parameters are influenced by workloads characteristics, we can observe the parameters are impacted according to workloads characteristics. Thus, it is possible, in future work, observe the workloads characteristics and working with the tuning of Hadoop parameters targeted at them.

C. Analyzed Papers

Analyzed papers have had in common the exploitation of configuration parameters and performance of Hadoop. However, we notice that the studies have different core purposes, as well as different methods of experimentation. Thus, for better understanding, we classified the papers according to the main purpose and the methods applied.

For classifying the experimentation method, it was used the taxonomy proposed by Zelkowitz and Wallace [7]. In this taxonomy there are four approaches toward experimentation:

Copyright © Authors

ISSN: 2203-1731

TABLE II: Purpose and Experimentation Approach Results

Purpose	Approach	Scientific Papers
Energy Prediciton	Analytical	[8]
Performance Analysis	Empirical	[9], [10], [11], [12], [13], [14]
Performance Model	Empirical	[15], [16], [17]
	Analytical	[18], [19], [20], [21], [22], [23]
Performance Penalties	Empirical	[24]
Performance Prediction	Empirical	[25], [26], [27], [28], [3], [29], [30], [31], [32], [33]
	Analytical	[34], [35], [36], [37], [38]
Performance Tuning	Empirical	[5], [39], [6], [40], [41], [42], [43]
	Analytical	[44]

(1) Scientific Method, (2) Engineering Method, (3) Empirical Method and (4) Analytical Method.

Table II shows the main purpose of each work and the main experimentation methods found in the analyzed papers. Importantly, the identification of the method used in the analyzed studies are not always as clear to identify. Some studies show a twofold interpretation. Thus, even with dubious features, in some cases, we classify according to the predominant method.

We observed that many studies focused between the purpose of obtaining performance models (about 22.5%) and performance prediction (37.5%). Others were divided into perfor-mance analysis and tuning performance, and only one job to prediction energy and performance penalties. Furthermore, we observed that 67.5% of the analyzed papers have used the empirical approach as experimentation method, and 32.5%have used the analytical method.

Most studies have adopted an empirical approach in experimental methods. This means that the proposed hypothesis are validate by statistical analysis methods. On the other hand, some studies have utilized the analytical method. These studies have developed a formal theory to a problem, and results derived from this theory were compared with empirical observations.

III. Proposed Approach

This section presents an approach that gathers and organizes the components of the proposed environment as well as its ontology, to create a semantic environment.

As we have seen, to achieve an optimal configuration of Hadoop to obtain the best possible performance of your applications, it is not an easy task. Depends on several factors, including the characteristics of the application, workload and cluster, and others. The big challenge is that users of Hadoop MapReduce get to know, even before submitting their applications to a particular cluster or a cloud computing, the prediction of the most appropriate configuration parameters and the best tuning in order to get the best performance possible. This approach proposes a semantic environment, combining semantic resources, machine learning and Hadoop tuning concepts.

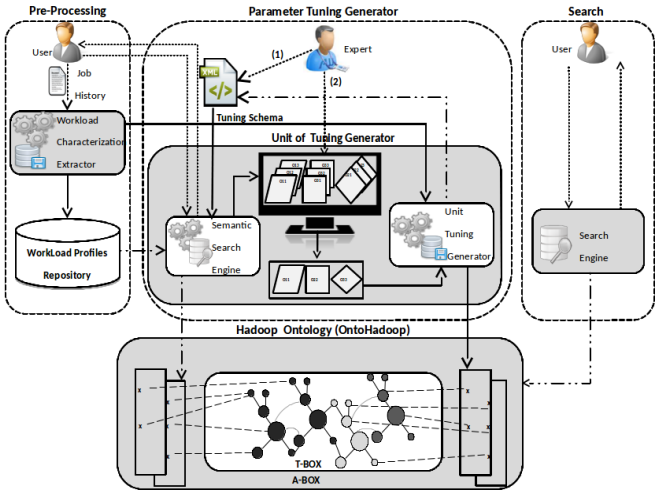

Fig. 1: Proposal Semantic Approach.

Fig. 1 provides a general overview of the ontology-based semantic approach to tuning parameters to improve Hadoop application performance. The approach is based on the Hadoop Ontology and subdivided into three main modules: pre-processing, the parameter tuning generator and search.

Being a semantic environment, it is proposed that all knowl-edge representation generated by collaborative tools and by the insertion of external knowledge should be organized in ontolo-gies. The ontology integrated into the proposed environment to make tuning possible is presented in the next section. The approach of the environment with a detailed description of each component is presented in the next sections.

A. Semantic Knowledge Representation

In that environment, all knowledge generated by the insertion of contents is classified and organized according to the workload characteristics. The characterization of workloads should be based in [45]. The work presented a methodology for understanding the performance of Hadoop MapReduce using traces of Facebook and Yahoo. It tries to answer *what-if* questions are related to system size, data intensity and hardware/software configuration. Thus it is proposed to organize the knowledge in a Hadoop Ontology. The Hadoop ontology was developed by creating a concept taxonomy, which describes the relations between workloads, Hadoop phases, configuration parameters and a set of axioms.

Therefore, after extracting the required attributes (see Section III-B1), the ontology is populated with this data. As shown in the Fig. 1, the OntoHadoop is divided into two parts: a T-Box containing a set of terminology axioms [46] based on knowledge (see Section II-A1 and Section II-B) acquired and modeled; and A-Box containing the set of assertions described for the T-Box. Thus, the unit tuning generator (see Section III-C) populates the A-Box of OntoHadoop with information on units of tune, creating individuals tuning in A-Box.

Copyright © Authors

ISSN: 2203-1731

B. Pre-Processing

The pre-processing is mainly characterized by Workload Characterization Extractor. The user submits the Job History of executing an application on Hadoop. This log is parsed by Workload Characterization Extractor module and stored in the workload profiles repository. This process serves to two purposes: the first one is the characterization of the workload itself and the second one is to instantiate the ontology with those characteristics.

1) Workload Characterization Extractor: When the logs are submitted, a parser extracts information from data ratio, such as number of bytes read and written in the input, shuffle and output stages; plus the total duration of application execution and duration of map and reduce phase. Thus, the performance metrics represent each job according to its size, either in data, runtime, or map and reduce task time.

Then, the module applies rules of machine learning to estimate the type or profile of the workload from the variables already mentioned. A good approach to achieve this goal is the use of linear regression [8], [41] which data are modeled using linear predictor functions, and unknown model parameters are estimated from the data.

In this way, it extracts the main feature of the workload regarding the classification suggested by [45], which classifies workloads in: small jobs, load data, aggregate, aggregate and expand, expand and aggregate, data transform, data summary and transform and expand. All the extracted information is stored in a profile repository for later use in new submission job history. The intention is that the more logs are submitted, the better the characterizations in the future.

C. Parameter Tuning Generator

The generator unit line is divided into two subcomponents: the semantic search engine and unit tuning generator.

1) Semantic Search Engine: The semantic search engine has three purposes:

- Search existing units of tune: After the tuning schema have been defined, the semantic search engine checks if there is already some similar tuning unit and presents it to the expert;
- Search by entering a log history: the log is parsed and classified in the pre-processing (see Section III-B) and, after stored in the repository, is launched to the semantic search engine to find a suitable unit tuning within the ontology;
- Search by statements in SPARQL [47] directly in the ontology by specific statements about type, size, and other characteristics of the workload.

2) Unit Tuning Generator: The Unit Tuning Generator is responsible for generating units tuning that compose the ontology instances. First, an expert is responsible for creating a tuning scheme (see Fig. 1 (1)), which defines a parameter tuning theoretically ideal for a given workload. This tuning scheme will contain information about workload, such as the profile of the workload, which Hadoop phases are critical and if application is CPU, IO or memory bound, among others. So, in Fig. 1 (2), the expert evaluates, selects, and if he wishes, changes the units of tune. Then, this unit goes to the Tuning Unit Generator module to be instantiated in OntoHadoop;

D. Search Component

After the ontology be populated, users can perform searches by workloads or Hadoop configuration parameters. This component has no semantic search support. The search component searches the ontology both by workload type and by units tuning, according to the keywords defined by the user. Results are presented to the user in order to know which informations are contained in the ontology.

IV. Final Considerations

The paper presented an ontology-based semantic approach. This approach aims at building an environment to achieve the Hadoop tuning parameters and improve the performance of applications in this environment.

The most Hadoop MapReduce users do not know exactly which configuration parameters are appropriate or should be tuned to obtain the best performance from their applications. Provide the user the best possible configuration parameters in order to improve the performance of the applications before they are performed could prevent unnecessary waste of time and expense.

Furthermore, the Hadoop MapReduce environment needs environments that argue about techniques for predicting the best definition of the configuration parameters associated with workloads and their specific characteristics semantically modeled. Moreover, to our knowledge there is no existing work in this direction, addressing the ontology-based semantic for this purpose.

Acknowledgment

This work was partially supported by grant from the Coordination for the Improvement of the Higher Level Personnel (CAPES).

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[2] Apache, "Apache hadoop," http://hadoop.apache.org/, Oct 2012, october 24, 2012. [Online]. Available: {http://hadoop.apache.org/}

[3] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics," in *In CIDR*, 2011, pp. 261–272.

[4] J. Venner, "Tuning your mapreduce jobs," in *Pro Hadoop*. Apress, 2009, pp. 177–206.

[5] S. Babu, "Towards automatic optimization of mapreduce programs," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 137–142.

[6] X. Lin, W. Tang, and K. Wang, "Predator — an experience guided configuration optimizer for hadoop mapreduce," in *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, ser. CLOUDCOM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 419–426.

[7] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23–31, May 1998.

Copyright © Authors ISSN: 2203-1731

[8] W. Li, H. Yang, Z. Luan, and D. Qian, "Energy prediction for mapreduce workloads," in *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, ser. DASC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 443–448.

[9] N. Khoussainova, M. Balazinska, and D. Suciu, "Perfxplain: debugging mapreduce job performance," *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 598–609, Mar. 2012.

[10] K. Kim, K. Jeon, H. Han, S.-g. Kim, H. Jung, and H. Y. Yeom, "Mr-bench: A benchmark for mapreduce framework," in *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, ser. ICPADS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 11–18.

[11] M. Koehler, Y. Kaniovskyi, and S. Benkner, "An adaptive framework for the execution of data-intensive mapreduce applications in the cloud," in *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, ser. IPDPSW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1122–1131.

[12] M. Koehler and S. Benkner, "Design of an adaptive framework for utility-based optimization of scientific applications in the cloud," in *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, Nov 2012, pp. 303–308.

[13] T. D. Plantenga, Y. R. Choe, and A. Yoshimura, "Using performance measurements to improve mapreduce algorithms," *Procedia Computer Science*, vol. 9, no. 0, pp. 1920 – 1929, 2012, proceedings of the International Conference on Computational Science, {ICCS} 2012.

[14] G. Wang, A. R. Butt, H. Monti, and K. Gupta, "Towards synthesizing realistic workload traces for studying the hadoop ecosystem," in *19th IEEE Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. Raffles Hotel, Singapore: IEEE Computer Society, Jul. 2011, pp. 400–408.

[15] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "Tarazu: Optimizing mapreduce on heterogeneous clusters," *SIGARCH Comput. Archit. News*, vol. 40, no. 1, pp. 61–74, Mar. 2012.

[16] K. Kambatla, A. Pathak, and H. Pucha, "Towards optimizing hadoop provisioning in the cloud," in *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, ser. HotCloud'09. Berkeley, CA, USA: USENIX Association, 2009.

[17] Z. Zhang, L. Cherkasova, and B. T. Loo, "Benchmarking approach for designing a mapreduce performance model," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '13. New York, NY, USA: ACM, 2013, pp. 253–258.

[18] M. Elteir, H. Lin, and W. chun Feng, "Enhancing mapreduce via asynchronous data processing." in *ICPADS*. IEEE, 2010, pp. 397–405.

[19] J. Han, M. Ishii, and H. Makino, "A hadoop performance model for multi-rack clusters," in *Computer Science and Information Technology (CSIT), 2013 5th International Conference on*, March 2013, pp. 265–274.

[20] X. Lin, Z. Meng, C. Xu, and M. Wang, "A practical performance model for hadoop mapreduce," in *Proceedings of the 2012 IEEE International Conference on Cluster Computing Workshops*, ser. CLUSTERW '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 231–239.

[21] A. Nez and M. G. Merayo, "A formal framework to analyze cost and performance in map-reduce based applications," *Journal of Computational Science*, no. 0, pp. –, 2013.

[22] D. Tiwari and D. Solihin, "Modeling and analyzing key performance factors of shared memory mapreduce," in *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, May 2012, pp. 1306–1317.

[23] X. Yang and J. Sun, "An analytical performance model of mapreduce," in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, Sept 2011, pp. 306–310.

[24] S. Kadirvel and J. A. B. Fortes, "Towards self-caring mapreduce: Proactively reducing fault-induced execution-time penalties," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, July 2011, pp. 63–71.

[25] M. An, Y. Wang, and W. Wang, "Using index in the mapreduce framework," in *Proceedings of the 2010 12th International Asia-Pacific Web Conference*, ser. APWEB '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 52–58.

[26] S. Hammoud, M. Li, Y. Liu, N. Alham, and Z. Liu, "Mrsim: A discrete event based mapreduce simulator," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 6, Aug 2010, pp. 2993–2997.

[27] H. Herodotou, F. Dong, and S. Babu, "No one (cluster) size fits all: Automatic cluster sizing for data-intensive analytics," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 18:1–18:14.

[28] H. Herodotou and S. Babu, "Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs," *PVLDB: Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1111–1122, 2011.

[29] S. Kadirvel and J. A. B. Fortes, "Grey-box approach for performance prediction in map-reduce based platforms," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, July 2012, pp. 1–9.

[30] P. Lama and X. Zhou, "Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 63–72.

[31] G. Wang, A. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, Sept 2009, pp. 1–11.

[32] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "Using realistic simulation for performance analysis of mapreduce setups," in *Proceedings of the 1st ACM Workshop on Large-Scale System and Application Performance*, ser. LSAP '09. New York, NY, USA: ACM, 2009, pp. 19–26.

[33] H. Yang, Z. Luan, W. Li, and D. Qian, "Mapreduce workload modeling with statistical approach," *J. Grid Comput.*, vol. 10, no. 2, pp. 279–310, Jun. 2012.

[34] H. Herodotou, F. Dong, and S. Babu, "Mapreduce programming and cost-based optimization? crossing this chasm with starfish." *PVLDB*, vol. 4, no. 12, pp. 1446–1449, 2011.

[35] R. R. Kompella, Y. C. Hu, and D. Xie, "On the performance projectability of mapreduce," in *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, ser. CLOUDCOM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 301–308.

[36] N. B. Rizvandi, J. Taheri, R. Moraveji, and A. Y. Zomaya, "Network load analysis and provisioning of mapreduce applications," in *Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, ser. PDCAT '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 161–166.

[37] ——, "On modelling and prediction of total cpu usage for applications in mapreduce environments," in *Proceedings of the 12th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ser. ICA3PP'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 414–427.

[38] H. Yang, Z. Luan, W. Li, D. Qian, and G. Guan, "Statistics-based workload modeling for mapreduce," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, ser. IPDPSW '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2043–2051.

[39] D. Heger, "Hadoop performance tuning - a pragmatic & iterative approach," *CMG Journal*, 2013.

[40] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "Hsim: A mapreduce simulator in enabling cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 300–308, Jan. 2013.

[41] W. Premchaiswadi and W. Romsaiyud, "Optimizing and tuning mapreduce jobs to improve the large-scale data analysis process," *Int. J. Intell. Syst.*, vol. 28, no. 2, pp. 185–200, Feb. 2013.

[42] A. Raj, K. Kaur, U. Dutta, V. Sandeep, and S. Rao, "Enhancement of hadoop clusters with virtualization using the capacity scheduler," in *Services in Emerging Markets (ICSEM), 2012 Third International Conference on*, Dec 2012, pp. 50–57.

[43] K. Wang, B. Tan, J. Shi, and B. Yang, "Automatic task slots assignment in hadoop mapreduce," in *Proceedings of the 1st Workshop on Architectures and Systems for Big Data*, ser. ASBD '11. New York, NY, USA: ACM, 2011, pp. 24–29.

[44] Z. Guo, G. Fox, M. Zhou, and Y. Ruan, "Improving resource utilization in mapreduce." in *CLUSTER*. IEEE, 2012, pp. 402–410.

[45] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "A methodology for understanding mapreduce performance under diverse workloads," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-135, Nov 2010.

[46] F. Baader, I. Horrocks, and U. Sattler, "Description Logics," in *Handbook of Knowledge Representation*, F. van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier, 2008, ch. 3, pp. 135–180. [Online]. Available: download/2007/BaHS07a.pdf

[47] W3C, "Sparql query language for rdf," http://www.w3.org/TR/rdf-sparql-query/, 2013. [Online]. Available: {http://www.w3.org/TR/rdf-sparql-query/}

Copyright © Authors ISSN: 2203-1731