A Hadoop MapReduce Performance Prediction Method

Ge Song, Zide Meng, Fabrice Huet, Frederic Magoules, Lei Yu, Xuelian Lin

A Hadoop MapReduce Performance Prediction Method

Ge Song*†, Zide Meng*, Fabrice Huet*, Frederic Magoules†, Lei Yu‡ and Xuelian Lin‡
* University of Nice Sophia Antipolis, CNRS, I3S, UMR 7271, France
†Ecole Centrale de Paris, France
‡Beihang University, Beijing China

***Abstract*—More and more Internet companies rely on large scale data analysis as part of their core services for tasks such as log analysis, feature extraction or data filtering. Map-Reduce, through its Hadoop implementation, has proved to be an efficient model for dealing with such data. One important challenge when performing such analysis is to predict the performance of individual jobs. In this paper, we propose a simple framework to predict the performance of Hadoop jobs. It is composed of a dynamic light-weight Hadoop job analyzer, and a prediction module using locally weighted regression methods. Our framework makes some theoretical cost models more practical, and also well fits for the diversification of the jobs and clusters. It can also help those users who want to predict the cost when applying for an on-demand cloud service. At the end, we do some experiments to verify our framework.**

***Index Terms*—Locally Weighted Regression; Job Analyzer; Performance Prediction; Hadoop; MapReduce;**

I. Introduction

It has been widely accepted that we are facing an information booming era. The amount of data becomes very huge, and traditional ways to manage and to process no longer work. In such a situation, MapReduce[1] has been proved as an efficient way to deal with "Big Data". Hadoop[2] is an open source implementation of MapReduce. A lot of internet companies have deployed many Hadoop clusters to provide core services, such as log analytic, data mining, feature extraction, etc. But usually the efficiency of these clusters is not very high. After studying the performance of some Hadoop clusters, we find some interesting problems as the following:

- How to design an efficient scheduling policy? There are many works done on scheduling policies for Hadoop clusters. But some of the advanced policies (such as [3] [4] [5]) require high level performance estimation in advance to decide scheduling strategies.
- How to tune the parameters? Hadoop provides more than 200 parameters both for the clusters and also for the jobs. But in most of the time, users just choose to use the default values or tune the parameters rely on some empirical values. But if we can estimate the job performance before execution, we can give more reasonable values.
- How to optimize the job performance? More and more companies pay attention on the ROI, which refers to "return of investment". They not only need to solve the problems but also want to optimize the job performance, so that they could use less time and less resources to solve more problems.
- How to balance between the cost and the performance? As IaaS becomes more and more popular, some users prefer to use an on-demand service rather than to deploy their own clusters. In such a situation, they need to precisely decide how long and how many nodes will they use.

To well settle these problems, the main point is to estimate the job performance in advance. That inspires us to create the framework described in this paper to predict the performance of a hadoop job. In order to provide plentiful predictions, our work is based on the following 2 main parts:

- A job analyzer: which is used to analyse the jobs submitted by the users to collect the features related with the jobs, it can also collect the parameters related with the clusters.
- A prediction module: which is used for estimating the performance in using a local weighted linear regretion method.

The main contributions of this paper are as follows:

1) We design a light-weight hadoop job analyzer. It can be used not only as a job performance analyzer but also a parameter collecter.
2) We propose a prediction module, which combines two kinds of information given by the job analyzer and the history traces to predict job performance.

II. Related Work

A. Hadoop MapReduce

The MapReduce programming model was initially developed by [1] for processing large scale of data. Hadoop is an open source framework made of a MapReduce framework and a distributed file system called HDFS. It is very popular not only among the academic institution but also in many real industries such as web search, social network, economic computation and so on. A lot of research work committed to optimize the Hadoop jobs performance and the efficiency of the Hadoop clusters in many different aspects [6] [7] [8].

When running a Hadoop job, the large amount of input data will be first divided into some splits (64M by default). Then each split will be executed by a user-defined map task. Take Word Count as an example, each input split contains

several lines of an article, each line is read as a record and then will be wrapped as key objects and value objects. Then the map function will consume a key and a value object and emit a key and value object as well. During this process, all the records (each line) will be executed by the same map function. After all the map tasks finish, the reduce tasks will pull the corresponding partitions from the output of the map tasks. Then all these data will be sorted and mergered in the reduce tasks to make sure that all the values with the same key will be put together. Finally reduce tasks will run and produce the output data.

B. Hadoop Simulator

To the best of our knowledge, there are two kinds of Hadoop simulators. One is trying to "replace" the Hadoop framework and usually focuses on the scheduling policy design [9] or resource allocation [10]. To simplify their problem and evaluation, they use some simulator to help them analyze the jobs. Usually these simulators are just simply an analog of the MapReduce framework without any complex node communication or process communication. Another one [11] is trying to analyze the application performance on MapReduce cluster through studying the language syntax, logical data-flow, data storage and its implementations. It provides a vivid MapReduce environment and can be used to test the scheduling algorithms. The framework in [11] is designed based on Grid Sim. It will not sample the input data and does not care about the complexity of the jobs. Instead, it will record the performance after executing the whole job in the environment of their simulator. In other words, this simulator is not light weight, and can not meet a quasi-realtime need.

All these works are not well suited for our prediction need. As described before, our job analyzer can not only support real Hadoop jobs but also profile usefull information about jobs and give them to the prediction moduler. Most of the others works do not focus on job analyzing, and cannot provide enough information, such as the complexity of the Map functions and Reduce functions, the conversion rate of the data and so on, for a further module to make prediction. At the same time, our job analyser is light-weight, it will consume only a little additional cost to provide plenty of information. And it is not only a simulator, it is designed to collect information and to analyze job performance, but it could also be used as a Map and Reduce function debugger tool.

C. Predictor

Some predictors are statistic based black box model , while others are cost model based white box model. In [12], the author classify the job into several categories by collecting the history trace of a given cluster. And inside each categories, they use a statistic model to predict job execution time. The authors also compare some clustering technics and feature elimination technics, then propose to use Kernel Canonical Correlation Analysis (KCCA) statistic model to find out the correlation between the features (e.x. inputSize, shuffleInputRatio, outputShuffleRatio, etc.) and the job execution times.

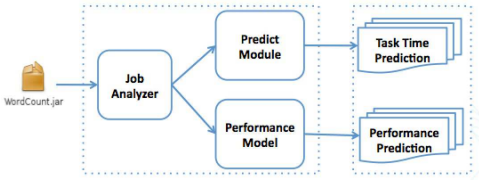

Fig. 1. System Overview

The biggest difference between our work and this kind of work is that we focus on predicting detailed information about jobs. Meanwhile, their features can not be obtained before job execution, and we can use our job analyser to get our features.

Another kind of predictor is based on cost-model. The what-if engine discribed in [13] is focusing on optimizing Hadoop job performance by predicting job performance. This engine is actually a predictor. It gives a corresponding prediction by tuning Hadoop job configurations. But as Hadoop has more than 200 configuration parameters, it's not easy to control the work under a low overhead. And in our work, we only care about a few main features (5 for Map and 6 for Reduce) which can accurately react the performance, so that we can give the prediction of the performance within a reasonable time and help to guide the scheduling policy or any other tuning needs.

III. System Design

In this section, we introduce the design of our system. The purpose of this system is to predict the job performance, hence, i.e. the execution time of Map task and Reduce task. Our work can also help some ideal cost models such as [14] to calculate the CPU, Disk I/O and Network I/O cost as well. Meanwhile, it can also be used to help other optimizing works about Hadoop such as [15] to estimate their cost. All the parts in our system are loosely coupled. Figure 1 presents an overview.

A. Job Analyzer

The purpose of the Job Analyzer is to extract the following information from the use submitted job. First, it measures the data input size and the number of records. Second, it tries to estimate the complexity of the Map and Reduce functions. Finally, it estimates the data conversion rate, i.e. the ratio between the input and output data of a mapper.

And these information should be achieved in a reansonable time within a low latency, because any additional time cost is not welcomed by users. To achieve these goals, we first get a sample of the input data instead of processing the whole data set. Then we extract and modify the procedure code only for Map and Reduce functions, eliminate all the costs for transfering data, initiate the cluster, and so on. Finally, we use the reflection mechanism of Java to instantiate the processing class of Map and Reduce in users' jobs.

1) Feasibility Analysis: We argue that our method is feasible from the following 2 aspects:

The introduction about Hadoop in sectio I shows out that a Hadoop MapReduce job has some special characteristics as shown below,

- **Execution Similarity:** According to the programming model, users only have to provide a map function and a reduce function. And the execution for each Map task (or Reduce task) is very similar to others. In other words, all data will be processed by these functions repeatly. Thanks to this design, we only need to study how each key-value pairs are processed for a particular job, as reading, sorting, transferring and writing data are independent of these two functions.
- **Data Similarity:** MapReduce is well suited for off-line batches processing. And it is usually used to do repeated work in which the input data has very similar format, such as log analysis, inverted index and so on. We can just take a look at a very little sample and then we can estimate the whole dataset.

These two main characteristics make the methods used in our Job Analyzer possible.

2) Implementation: To analyze a job, we first need to get a sample of input data. And we read these data into memory and package them by (key, value) pairs. Then we will execute map function, and sort the output by a hash function, then package the result into (key, value) pairs and give them to Reduce function. After that we will run the reduce function and give the output. Through these processes, we will count the number of the (key, value) pairs, the size of them, and so on.

Figure 2 shows the structure of Job Analyzer.

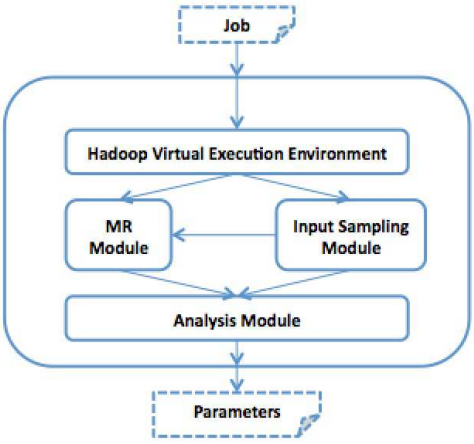

Fig. 2. Job Analyzer

The Job Analyzer mainly contains 4 parts, the Hadoop virtual execution environment, the input sampling module, the MR module and the control module. We will present the details for each module:

- **Hadoop virtual execution environment**
  This is the main part of Job Analyzer. Its main purpose is to support to execute the Hadoop jobs submitted by users. In this virtual execution environment, Jop Analyzer can on one hand use the user configuration variables and on the other hand collect the dynamic parameters in processing map and reduce tasks.
- **Input Sampling Module**
  The main function of this module is to sample and read the input file from HDFS. The most important thing for this module is to know the percentage of the sampling. If the ratio is set too high, to sample will consume too much time; but if it is set too low, then the sample will not reflect the overall of the data set. We use a the sampling algorithm shown below.

Algorithm 1 Sampling Algorithm

Require:
The input file: FILE;
The quantity of lines that we want to sample: k;
The maximum number of iterations (depends on the filesize): MAX;

Ensure:
The sample with k lines (each line is selected with the probability k/n): S_k;

```
1: S_k = [1, 2, ..., k] //Take the top k lines in FILE;
2: if k > n then
3:    return S_k
4: end if
5: for new line i to EndOfFile do
6:    if MAX < 0 then
7:       break;
8:    end if
9:    MAX − −;
10:   int r = random(0, i)
11:   if r < k then
12:      S[r]= new line i;
13:   end if
14: end for
15: return S_k;
```

- **MR module**
  This module is used to execute the Map function and Reduce function. We use the reflection mechanism of Java and the configuration variables collected by the virtual environment to instance the mapper object and reducer object defined in users' job. Then we use the sample data as input to execute the users' map function and reduce function.
- **Analysis Module**
  In this part we will calculate some features defined in our prediction module, for example the number of input records (N); the complexity of the map and reduce functions (MapComplex and ReduceComplex)[14]; and also the conversion ratio of data for map and reduce tasks (MapDataConv and ReduceDataConv).

B. Prediction Module

In order to get an in-depth understanding of Hadoop's job processing, we have to go deep into every step of the implementation details and concurrent relationships. There are a lot researches talking about MapReduce performance model [16] [17] [14]. Different with most of the other cost

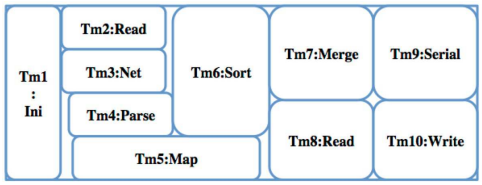

Fig. 3. Map Phase

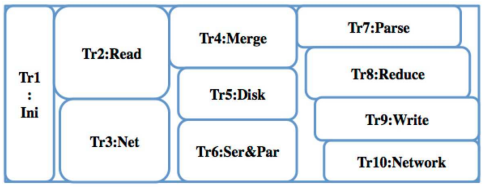

Fig. 4. Reduce Phase

models which divide the processing from the perspective of the execution order, we choose to use the method proposed in [14] to devide the processing from the perspective of resources dimension. This can help us to predict the consumption of different types of resources.

Figure 3 and Figure 4 show the processing steps of map tasks and reduce tasks proposed in [14].

As shown in the figures, we can assume that the total cost for executing Map or Reduce task should be a function of these 10 steps. We suppose that $T_{map} = f(Tm1, Tm2, ..., Tm10)$, and $T_{reduce} = f(Tr1, Tr2, ..., Tr10)$.

Through analysis, we have found that the 10 steps of map tasks are affected by 5 features related to a job. They are: the amount of Map input data (MapInput), the number of Map input records (N), N*log(N), the complexity of Map function (MapComplex), and the data conversion rate for Map function (MapDataConv)

Table I shows the relationship between the 10 steps for Map task and these 5 features.

TABLE I
RELATIONSHIP BETWEEN MAP STEPS AND FEATURES

Feature	Step
MapInput	Tm2, Tm3, Tm7, Tm8, Tm10
N	Tm4, Tm5, Tm6, Tm7, Tm9
N*log(N)	Tm6
MapComplex	Tm5
MapDataConv	Tm7, Tm8, Tm9, Tm10

Similarly, the ten steps of Reduce tasks are affected by following 6 features: the amount of Map input data (MapInput), the number of Map input records (N), N*log(N), the complexity of Reduce function (ReduceComplex), the data conversion rate for Map function (MapDataConv), and the data conversion rate for Reduce function (ReduceDataConv)

Table II shows the relationship between the 10 steps for Reduce task and these 6 features.

TABLE II
RELATIONSHIP BETWEEN REDUCE STEPS AND FEATURES

Feature	Step
MapInput	Tr2, Tr3, Tr5, Tr9, Tr10
N	Tr4, Tr6, Tr7, Tr8
N*log(N)	Tr4
ReduceComplex	Tr8
MapDataConv	Tr2, Tr3, Tr4, Tr5, Tr6, Tr7, Tr8, Tr9, Tr10
ReduceDataConv	Tr9, Tr10

TABLE III
CORRELATION COEFFICIENT FOR TREDUCE

Job Name	Dedup	WordCount	Project	Grep	Total
R^2	0.9982	0.9992	0.9991	0.9949	0.6157

We assume that T_{map} and T_{reduce} have some linear relationship with their corresponding features:
$T_{map} = a_0 + a_1 * MapInput + a_2 * N + a_3 * Nlog(N) + a_4 * MapComplex + a_5 * MapDataConv$
$T_{reduce} = b_0 + b_1 * MapInput + b_2 * N + b_3 * Nlog(N) + b_4 * ReduceComplex + b_5 * MapDataConv + b_6 * ReduceDataConv$

To test our hypothesis, we generate 10 benchmarks with the input data vary from 64M to 8G for 4 kinds of jobs, and we repeat each benchmark for 3 times, and collect the values of the features needed in our linear functions, then we use these values to calculate the correlation coefficient R^2 for T_{reduce}. The correlation coefficient R^2 represents the strength of the linear relationship between the two variables in a regression equation. A value close to 1, indicates that the function is linear, whereas a value close to 0 indicates that there is no linear relationship between the two variables. We use R[1] scripts to calculate this coefficient, the results are listed in table III.

From table III we can see that there is a very good linear relationship for T_{reduce} within each type of jobs (with R^2 bigger than 0,99), but among different types the linear relationship is not good enough (R^2=0.6157). T_{map} shows similar results. So, we will get a very good result if we choose the history trace of the similar type of jobs as a training set to train our linear model. And different feature will give different affect to the model. For example, if the job is compute-intensive, then the complexity of map and reduce function will be the main features; if the job is data-intensive, then the amount of input and the number of records will be more important. This encourage us to choose locally weighted regression method to solve our problem. In this method, we give each feature a weight w. And we measure the weighted distance between the new instance and the ones in the history trace in order to choose the similar type of jobs as a new training set. And then we use this training set to train our model, and use this model to make prediction.

[1]http://www.r-project.org/

The structure of the Prediction Module is shown in Figure 5.

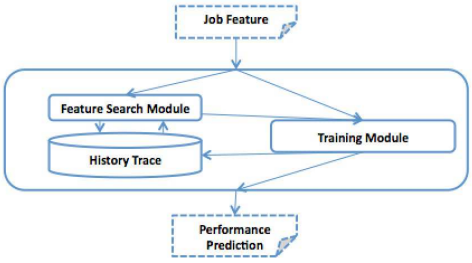

Fig. 5. Prediction Module

IV. Evaluation

In this section we will evaluate the performance and precision of the prediction module. We set our experiment on a 5 nodes Hadoop cluster. The cluster is deployed with Hadoop version 1.0 with one node started as master and the other nodes as slaves. We have chosen a 5 nodes cluster to limit the size of the training set. Indeed, the larger the cluster, the larger the training set we can obtain, increasing the accuracy of the machine learning method. Evaluating our method on a small training set will give insight on its behaviour with limited training.

A. Experiments for Job Analyzer

The Job analyzer is used for sampling and getting properties such as the number of records (N), the complexity of Map function and Reduce function (MapComplex and ReduceComplex), and also the data conversion rate for Map task and Reduce task (MapDataConv and ReduceDataConv).

One of the most important part of the Job Analyzer is sampling. We test our sampling module with data varying from 16 MB to 8 GB. For each data set, we compare the number of records estimated by the job analyzer and the real one from the history trace. Among them, the average error rate is less them 0.02. That means our sampling algorithm is good enough, and the sampling we got can reflect a real situation about the whole data set.

The results are shown in figure 6.

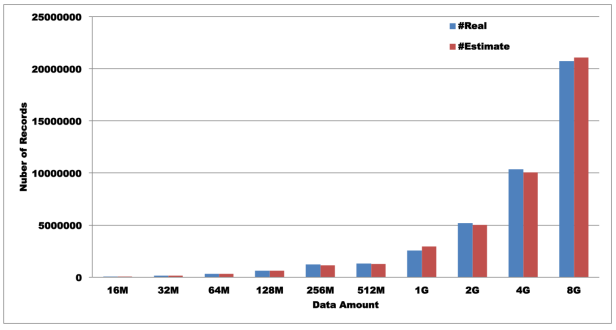

Fig. 6. Precision of sampling

We also compare the complexity of Map and Reduce functions as estimated by the job analyzer to real job history traces. We define the complexity as the ratio of the execution time of a job to the execution time of a standard job, running on the same amount of data in the same environment. Here we choose Word Count as a standard job as it is a typical data-intensive job with an average complex, that will make the value of complexity not too big or too small. Figure 7 shows that for the wc and proj jobs, we obtain a very close value for the complexity. The difference for grep and dedup jobs is because in a real job we use a large amount of data, but in job analyser we only choose a small sample, and grep and dedup depends on the records, different records will give a very different result, but the error is still acceptable.

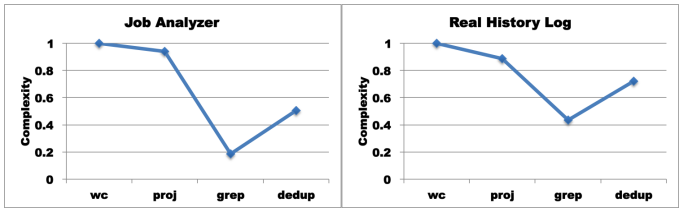

Fig. 7. Estimation of Job Complexity

B. Experiments for Prediction Module

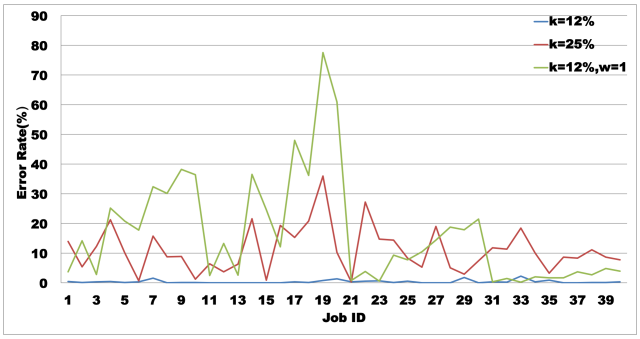

Fig. 8. Map task prediction

1) The prediction about Map task execution time: Figure 8 shows the the prediction about the Map task execution time for various sampling ratio K, K reflects the number of similar jobs being choosen from the history trace. We can see from this experiment that when the sampling ratio is not high, the prediction accuracy is higher, but when the sampling ratio is high, the accuracy rate will decrease. That's because our prediction model is based on a multiple linear regression model, if the sample rate is high, that means we have selected a lot of history traces which is far from the job need to be predicted. And these selected traces are not similar with the job need to be predicted, that's why the accuracy decrease.

We use locally weighted regression method to train our model. When considering a new job need to be predicted, we first need to find out the similar jobs to use as training set. But as we discribe before, different feature gives different effect for the job classification. So, for important features we set a higher weight w in order to find the most similar jobs. We can

see that if we set all the weight w to be 1, it will give us a worse result than giving different features a different weight.

The sampling ratio K and the weight w can be set to some empirical values, or can be infered using machine learning methods.

2) The prediction about Reduce task execution time: The same analysis also applies to the prediction about the execution time of Reduce tasks as shown in figure 9.

From figure 8 and figure 9 we can see that the error rates are high for some job IDs. These particular jobs had the highest possible input size, 8GB. Hence, our trained model had no similar jobs and could not come with a good prediction. This could be solved by improving the training.

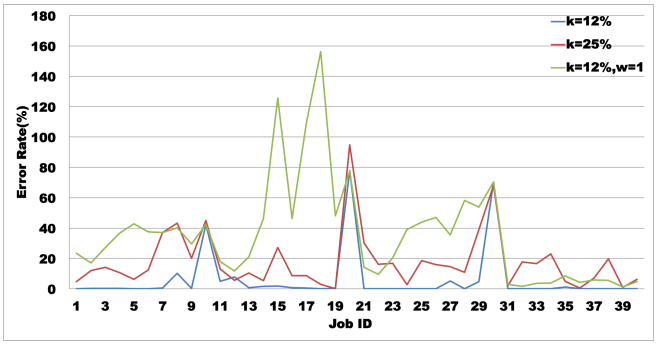

Fig. 9. Reduce task prediction

3) Overhead of prediction: Figure 10 shows the overhead of our framework. We define the overhead as a proportion between the execution time of our framework for prediction and the job execution time get from the job history. As shown in the figure, when the amount of data augments, the overhead becomes smaller.

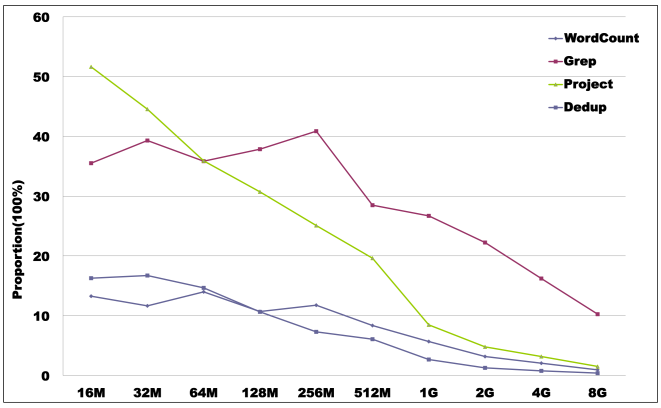

Fig. 10. overhead of the prediction

V. Conclusion

In this paper we have presented a simple model for predicting Hadoop MapReduce jobs' performance. This model is implemented through 2 parts, a job analyzer and a prediction module. The job analyzer is in charge of collecting the important properties related to the jobs for the prediction module. And then the prediction module will use this information to train a locally linear model in using locally weighted regression method. Our work can predict task execution times as well as other job performance metrics. We have shown in experiments the accuracy and efficiency of our framework. It accurately predicts the performance of jobs with a low overhead.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*, 2004, pp. 10–10. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251254.1251264

[2] A. S. Foundation, "Apache hadoop." [Online]. Available: {http://hadoop.apache.org/}

[3] S. Ge, Y. Lei, M. Zide, and L. Xuelian, "A game theory based mapreduce scheduling algorithm," in *Proceedings of the 2nd ICM Conference*, ser. ICM '12. New York, NY, USA: Springer Science+Business Media, 2012.

[4] 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 *Proceedings of the 5th European conference on Computer systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 265–278. [Online]. Available: http://doi.acm.org/10.1145/1755913.1755940

[5] C. He, Y. Lu, and D. Swanson, "Matchmaking: A new mapreduce scheduling technique," in *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, ser. CLOUDCOM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 40–47. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2011.16

[6] M. J. Fischer, X. Su, and Y. Yin, "Assigning tasks for efficiency in hadoop: extended abstract," in *SPAA*, 2010, pp. 30–39.

[7] A. Verma, L. Cherkasova, and R. H. Campbell, "Aria: automatic resource inference and allocation for mapreduce environments," in *Proceedings of the 8th ACM international conference on Autonomic computing*, ser. ICAC '11. New York, NY, USA: ACM, 2011, pp. 235–244. [Online]. Available: http://doi.acm.org/10.1145/1998582.1998637

[8] J. Polo, D. Carrera, Y. Becerra, M. Steinder, and I. Whalley, "Performance-driven task co-scheduling for mapreduce environments," in *NOMS*, 2010, pp. 373–380.

[9] A. Verma, L. Cherkasova, and R. H. Campbell, "Play it again, simmr!" in *CLUSTER*, 2011, pp. 253–261.

[10] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in *MASCOTS*, 2009, pp. 1–11.

[11] F. Teng, L. Yu, and F. Magoulès, "Simmapreduce: A simulator for modeling mapreduce framework," in *MUE*, 2011, pp. 277–282.

[12] A. Ganapathi, Y. Chen, A. Fox, R. H. Katz, and D. A. Patterson, "Statistics-driven workload modeling for the cloud," in *ICDE Workshops*, 2010, pp. 87–92.

[13] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of mapreduce programs," *PVLDB*, vol. 4, no. 11, pp. 1111–1122, 2011.

[14] X. Lin, Z. Meng, C. Xu, and M. Wang, "A practical performance model for hadoop mapreduce," in *CLUSTER Workshops*, 2012, pp. 231–239.

[15] J. Pan, F. Magoulès, Y. L. Biannic, and C. Favart, "Parallelizing multiple group-by queries using MapReduce: optimization and cost estimation." *Telecommunication Systems*, vol. (in press), 2013.

[16] H. Herodotou, "Hadoop performance models," *CoRR*, vol. abs/1106.0940, 2011.

[17] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of mapreduce: An in-depth study," *PVLDB*, vol. 3, no. 1, pp. 472–483, 2010.