Energy Efficiency in Cloud-Based MapReduce Applications through Better Performance Estimation

Seyed Morteza Nabavinejad
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
mnabavi@ce.sharif.edu

Maziar Goudarzi
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
goudarzi@sharif.edu

***Abstract*— An important issue for efficient execution of MapReduce jobs on a cloud platform is selecting the best fitting virtual machine (VM) configuration(s) among the miscellany of choices that cloud providers offer. Wise selection of VM configurations can lead to better performance, cost and energy consumption. Therefore, it is crucial to explore the available configurations and choose the best one for each given MapReduce application. Executing the given application on all the configurations for comparison is a costly, time and energy consuming process. An alternative is to run the application on a subset of configurations (sample configurations) and estimate its performance on other configurations based on the obtained values on those sample configurations. We show that the choice of these sample configurations highly affects accuracy of later estimations. Our Smart Configuration Selection (SCS) scheme chooses better representatives from among all configurations by once-off analysis of given performance figures of the benchmarks so as to increase the accuracy of estimations of missing values, and consequently, to more accurately choose the configuration providing the highest performance. The results show that the SCS choice of sample configurations is very close to the best choice, and can reduce estimation error to 7.11% from the original 16.02% of random configuration selection. Furthermore, this more accurate performance estimation saves 24.3% energy on average.**

Keywords*— *Performance Estimation; Energy Efficiency; Big Data Analysis; MapReduce Applications; Matrix Completion

I. INTRODUCTION

With the proliferation of data centers, their energy consumption has become a main concern. The energy consumption of data centers grows up about 15% annually and the energy cost is about 42% of total operational cost of data centers [1]. The approaches to reduce the energy consumption in data centers can be categorized as *structural improvements* mainly for cooling efficiency, *power improvements* to lower power loss, *VM consolidation* to operate the servers in more energy efficient states as well as to turn off unused resources, and finally *application-level optimizations* where custom solutions are proposed based on application features.

Since the cooling system and power distribution units consume up to 35% of energy in data centers [2] , various types of *structural and power improvements* are presented [3-5] to reduce their share of energy consumption. The emergence of *virtualization technology* and Infrastructure as a Service (IaaS) cloud services provided this opportunity to improve the poor utilization of servers [6] and make them energy proportional by *VM consolidation* [7-9].

Application-level optimizations dive into details of the application in question to improve processes such as resource allocation to consequently reduce the energy consumption. For example [10-14] consider the specifications of MapReduce applications (as the common framework for Big Data analysis) when allocating resources or scheduling jobs.

A common challenge in this last category in the case of resource allocation for MapReduce jobs is that *estimation is needed on the execution time or performance of the application on available VM configurations*. The common framework used in most these works is shown in Fig. 1: the *Estimation Phase* provides an estimate of the processing time of each VM configuration per unit data volume, measured in seconds per GB, s/GB; then the *Resource Allocation Phase* chooses the best configuration and the number of VM instances required to process the whole big data while adhering to the given deadline. Obviously, overestimation of the VM configuration performance results in losing the deadline, and underestimation results in overbooking of resources and waste of energy and money. Thus, accurate and low-cost estimation methods are required to avoid both these cases. Note that for ease of management, usually the same VM configuration is used for all VM instances, and hence, the system is homogeneous.

One common method to address this estimation challenge is *profiling*. In *complete profiling*, the application is executed on ***all*** the accessible configurations for a short period of time (a few minutes) to estimate performance of each configuration. If the state space of the problem is small (i.e., few number of available configurations) this approach will be suitable. Otherwise, when the number of selectable choices is high, which is usually the case with today and anticipated growth in cloud computing providers, running the application on all of them means using a lot of resources that needs lots of money and leads to waste of energy and time. Thus, *complete profiling* is usually not acceptable.

This research is supported by grant number G930826 from Sharif University of Technology. We are grateful for their support.

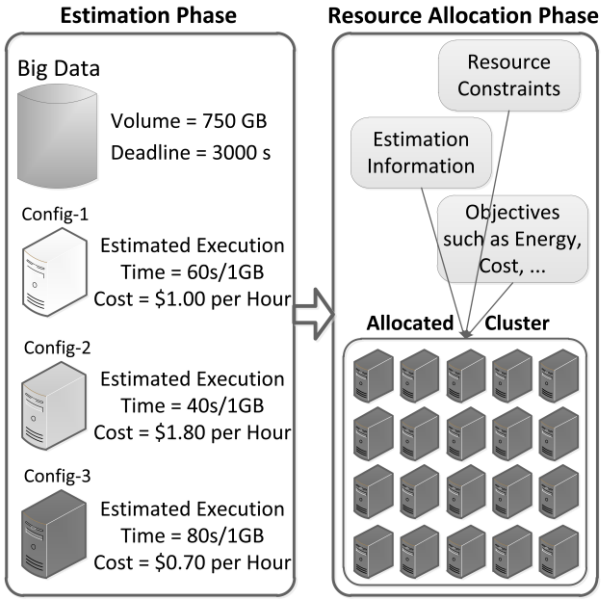

Fig. 1. Common framework in resource allocation for MapReduce jobs in clouds.

In *partial profiling*, only a small subset of the state space (i.e., a few configurations) are actually profiled per new application, and then results are extended to the whole state space based on the results of prior complete profiling of a few representative benchmarks. In [15], the authors first run some applications on all the available configurations (Offline Training), and then, when a new application arrives, its performance is measured on two sample configurations for a short period of time, and then by a reconstruction technique called Matrix Completion [16], the values for other configurations is predicted. Consequently, the Offline Training Phase (see Fig. 2) serves several future applications, and less time and resources are used for the Usage Phase per new application.

Reference [15] chooses the two sample configurations randomly. However, our results in section 2 show that the accuracy of predicting the missing values is widely dependent on the choice of these sample configurations.

In this paper, we propose Smart Configuration Selection (SCS) method to leverage the power of matrix completion by carefully choosing the sample configurations. We apply our method on a set of VM configurations that are already offered by various cloud providers such as Amazon EC2 [17] and Microsoft Azure[18]. We show that this method can increase the accuracy of missing values prediction by more than 2x compared with random selection.

The rest of the paper is organized as follows. Section 2 gives an example to show the motivation beyond this work and discusses it. Section 3 introduces the SCS approach and its mathematical background. Section 4 represents the experimental results. Finally, the paper is concluded and future works are provided in section 5.

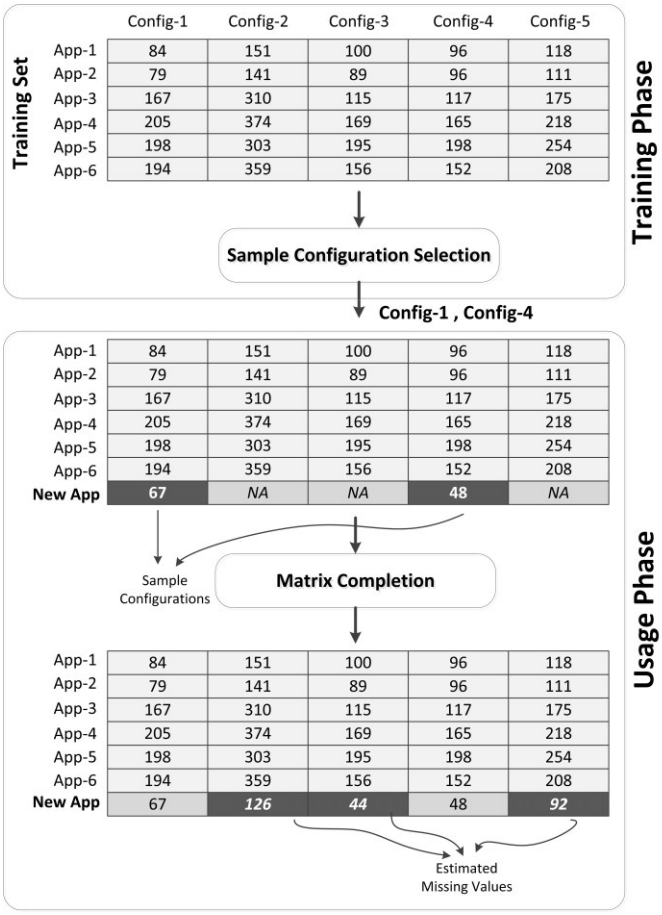

Fig. 2. Overall flow of performance estimation using matrix completion.

II. MOTIVATION

In this section, we employ an example to show motivation behind our research. We ran eight MapReduce applications from PUMA suite [19] on nine VM configurations with different amount of resources (e.g., RAM and CPU). VM configurations are the same as some of the instances from [17] and [18]. We also used datasets from [19] as the input of MapReduce applications. The numbers in TABLE I indicate the execution time of each application on the various configurations (in seconds) for 3GB of data.

TABLE I

EXECUTION TIME OF MAPREDUCE APPLICATIONS ON DIFFERENT CONFIGURATIONS

		VM Configurations (G : RAM (Gigabyte), C : Number of Cores)								
		1	2	3	4	5	6	7	8	9
		7G - 4C	3.5G - 2C	7.2G - 8C	7G - 7C	4G - 4C	14G - 8C	15G - 4C	3.75G - 1C	7.5G - 2C
MapReduce Applications from PUMA Suite	Classification	84	151	100	96	118	53	89	266	146
	Grep	67	126	44	48	92	43	71	207	117
	Histogram_Movies	79	141	89	96	111	50	84	243	134
	Histogram_Ratings	167	310	115	117	175	98	174	508	306
	Inveted_Index	205	374	169	165	218	121	218	663	364
	Kmean	198	303	195	198	254	110	160	436	252
	Sequence_Count	339	536	362	339	500	185	283	843	489
	Word_Count	194	359	156	152	208	111	196	626	350

Before presenting the results, we depict the overall flow of experiments and define necessary concepts for pursuing the rest of paper. Fig. 2 illustrates the flow from training phase to usage phase. First, some applications are executed on all the available configurations. These applications are training set and information obtained from their execution is used for next steps. This step is offline and just need to be done once (Training Phase). After this phase, the new application is introduced to system which we want estimate its performance on configurations. Next, a subset of configurations is employed and the new application is executed on them. From now on this subset is called sample configurations. Finally, from information of training phase and executing the new application on sample configurations, the matrix completion approach estimates the performance of new application on the rest of configurations which we call missing values. This step is done online (Usage Phase).

For showing that selecting the sample configurations randomly might lead to imprecise estimation of missing values, we conducted a set of experiments for each application in TABLE I. In each set, we selected one application and assumed it as the new app we want to sample on configurations and there is no information about it. The other applications are considered as training set that we want to predict the missing values of new application based on their information. Then, we chose a pair of configurations randomly as known samples, which the execution time of application is known on them. The seven left ones are considered as missing values. Finally, we applied matrix completion program from [20] on the matrix with missing values and observed the results. This process repeated ten times for each application and in different iterations we chose a different pair of random sample configurations. Approximation error (the difference between estimated value and actual value) of matrix Completion regarding missing values for each of ten iterations in all the eight applications is demonstrated in Fig. 3. As this figure reveals, the amount of error differs significantly from one sampling pair to the other. Therefore, it can be concluded that the selected sample configurations can affect the outcome of matrix completion remarkably. So, it is essential to choose the sample configurations wisely in order to have an accurate prediction of missing values.

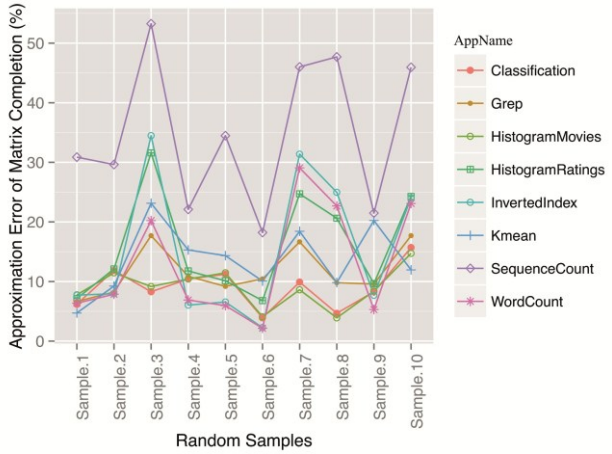

Fig. 3. Approximation Error of Matrix Completion for ten sample configurations.

III. Smart Configuration Sampling (SCS)

In this section, we present the SCS approach. The main goal of SCS is to select the sample configurations in a way that leads to more accurate prediction of missing values in matrix completion process. First, the mathematical background of SCS, including correlation coefficient and its variants is explained and then the approach itself is expanded.

A. Mathematical Background

Correlation Coefficient indicates the linear relationship between two variables and also shows the strength and direction of this relationship. Various versions of correlation coefficient are available such as Pearson and Kendal.

Kendal Rank Correlation Coefficient (KRCC) [21] indicates the portion of ranks that match between two data sets. The proposed method in [22] uses KRCC to evaluate the similarity between users of cloud services and based on that estimates the missing values (response time of cloud services for users who have not used the service yet).

Unlike [22], where the missing values and present ones are not determined by the algorithm because they cannot force users to use specific services, in our work we have this freedom to choose the present values by running the application on counterpart configurations. So, we use the Pearson Correlation Coefficient (PCC). For two variables x and y, the PCC is a value between -1 and +1. +1 and -1 shows that the variables are strongly related to each other. Zero value means that there is no kind of relationship between two variables. The PCC of two variables can be calculated as (1) (N is the total Number of attributes):

$$PCC(x,y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}} \quad (1)$$

Equation (1) can be summarized as (2) for population:

$$PCC(x,y) = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y} \quad (2)$$

Where *cov* stands for covariance and σ stands for standard deviation. We use this form of *PCC* in our approach to measure the correlation between configurations.

B. SCS Approach

Considering TABLE I, we perceive that configurations one and six (7G – 4C and 14G – 8C) have strong relationship with each other (PCC (1, 6) = 0.998). So, what if we choose these two as sample configurations? The results have been shown in Fig. 4 and compared with the average of ten random samples. It shows the approximation error is too high and we can say it is an inauspicious choice. But, what is the reason? The answer

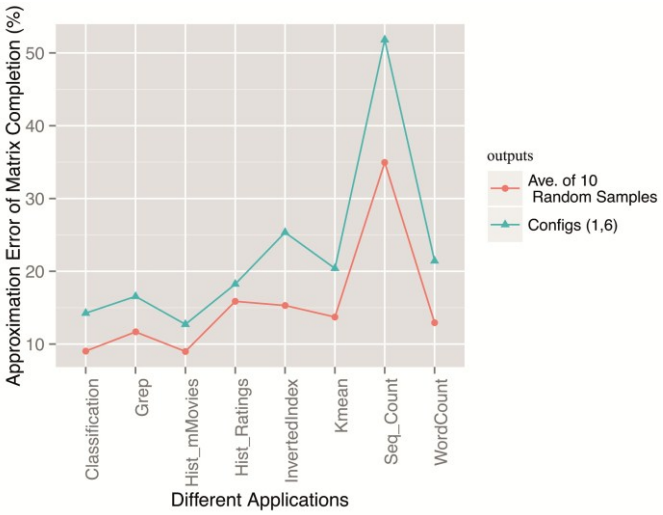

Fig. 4. Impact of PCC on approximation error

is that since these two configurations are correlated to each other, they cannot represent the diversity of all configurations. Hence the estimation leads to poor results. From this observation, we can conclude that the two selected samples must demonstrate the assortment of configurations in order to obtain a precise prediction. From this conclusion, we present the SCS approach which concentrates on finding dichotomy between configurations and picking the two bests that can represent it. This approach is offline and need to be applied just once in *Sample Configuration Selection* part in Fig. 2.

SCS first calculates the PCC for each two configurations (columns in TABLE I) where N stands for all the configurations**.**

$$\forall i, j \in N, PCC(i, j) = \frac{Cov(i, j)}{\sigma_i \sigma_j} \tag{3}$$

Then, it calculates the total amount of correlation be-tween each configuration and other ones. *TCC* stands for Total Correlation Coefficient.

$$\forall i \in N, TCC_i = \sum_{j=1}^{N(j \neq i)} PCC(i, j) \tag{4}$$

After that, the first configuration that will be chosen is the one with least amount of TCC because the one with least TCC shows the most diversity with other configurations.

$$Config1 = \{i \mid TCC_i = Min(TTC_j), j = 1...N\} \tag{5}$$

Finally, for second configuration, the one must be chosen that has the least PCC with first selected configuration. In this way, the too selected configurations have: 1) minimum correlation so are different from each other and 2) at least one of them has the least possible correlation with other configurations and can represent diversity among them.

$$Config2 = \{i \mid \mathrm{PCC}_i = Min(PCC(i, Config1), i = 1...N\} \tag{6}$$

SCS method is simple, yet very effective because it can quickly distinguish the two most proper configurations for sampling by doing a few calculations. The selected configurations are different from each other regarding PCC and they also can represent the diversity among whole set of available configurations.

IV. Experimental Results

Hence the state space of problem in our example (TABLE I) is small enough to be explored completely; we have calculated the amount of approximation error of matrix completion for all the possible pairs of sample configurations for all applications. Fig. 5 illustrates the results. As can be seen, for all the applications there is a wide gap between minimum and maximum error. These observations further prove the necessity for careful selection of sample configurations. In upcoming subsections, we first present the effectiveness of SCS for improving estimation accuracy of applications performance on different configurations. Then, we further explore the problem and depict the impact of estimation accuracy on meeting/missing applications deadline and resource utilization.

A. Impact of SCS on Estimation Accuracy

We ran the SCS for TABLE I and the configurations that are selected by SCS are configurations 4 and 8 (7G – 7C & 3.75G – 1C). The TCC for configuration 4 is 7.3808 and PCC (4, 8) = 0.8481. The comparison between SCS, Average of all possible pairs of sample configurations (all states) and best pair and worst pair of sample configurations per application are depicted in Fig. 6. The average error of SCS over eight applications is 7.11% while the average of worst case is 34.99%, average of best case is 5.46% and average of all states is 16.02%. It shows that while SCS obtains far better approximation than worst case, it answers are near the best case ones. However, SCS cannot always provide the best answer for any application, in average it outperforms any other sample configurations and hence it can be considered as an effective solution for sample configuration selection.

B. Estimation Accuracy Effect on Resource Management

In this section, we want to examine how the estimation accuracy can affect the resource management. Two important factors that are measured in this section are application deadline and energy consumption. For this reason, we compare the results from the SCS approach and four random sample configuration selections. For two applications *word count* and *histogram rating*, which are of different classes and have different input datasets, we want to select a configuration and process 30GB of data on that configuration. The desired deadline is 1800 seconds. The results of matrix completion for these two applications with respect to information from TABLE I under four different random sample configuration selections and SCS approach are presented in TABLE II. The gray cells are the sample configurations which are considered known and matrix completion is done based on them. The real values are also presented in the first two rows so a clear comparison between estimated values and real values can be possible.

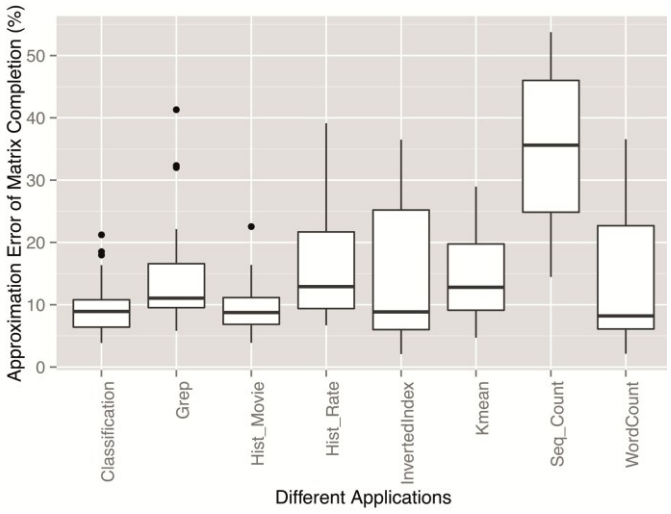

Fig. 5. Scattering of approximation error for all the possible sample configurations.

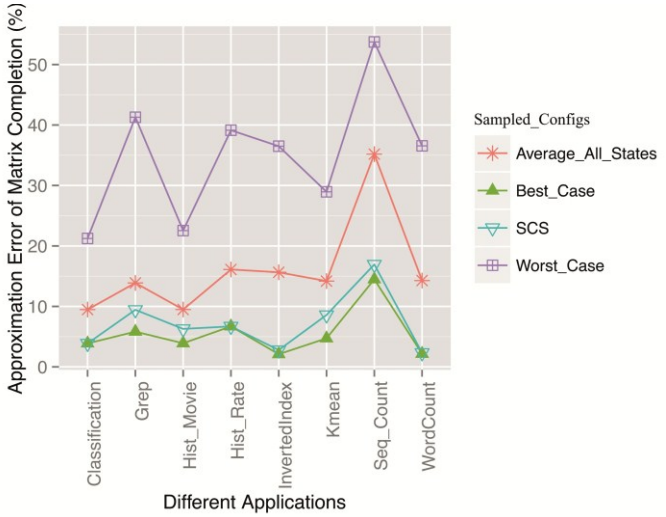

Fig. 6. Comparison of SCS with other sample configurations.

After the matrix completion is done, it is time for choosing the best configuration for executing the applications, based on estimated performances. A pseudo-linear relationship between volume of data and execution time is considered. Hence, a suitable configuration that can execute the 30GB of data in less than or equal to 1800 seconds, must be able to execute the 3GB of it in less than or equal to 180 seconds. So, the candidate configurations are the ones that have execution time of less than 180 seconds in each row of TABLE II. After finding the candidate configurations, it is time for choose the best one. The main factor for choosing the best configuration is number of cores. The less the number of cores, the better the configuration in terms of resource usage and consequently energy consumption. If two candidate configurations have the same number of cores, then the tie breaker will be amount of memory.

According to aforementioned mechanism for choosing the best configuration, configurations with bold number are selected. For example, based on sample configuration (2, 4) in Random 1 and consequently estimated values by it, configuration 7 is suggested for word count application and configuration 1 is suggested for histogram rating application. After that, a real input dataset with size of 30GB for each of applications is executed on selected configurations. The results of execution times are shown in Fig. 7.

Fig. 7 (a) shows that for word count application, the suggested configuration by all the four random ones has missed the deadline, whereas SCS meets the deadline. The reason is that random scenarios overestimated the performance of configurations that considered them faster than they really are. Consequently, their selected configurations couldn't meet the deadline.

On the other hand, in Fig. 7 (b), in addition to SCS, Random 1 also had been able to meet the deadline. But, if we look at the selected configurations by SCS and Random 1, we can see that selected configuration by Random 1 (configuration 1) has more resources than selected one by the SCS approach (configuration 5). Hence, while both of the configurations can meet the deadline, the one selected by Random 1 uses more resources and consequently suffer from resource waste. In fact, Random_1 has underestimated the performance of configuration.

Regarding energy consumption, the SCS again outperforms the random sample configurations. The results are depicted in Fig. 8. As can be seen, the more accurate estimation yield by the SCS approach is able to reduce energy consumption up to 41% compared with worst case (Random_3 in (a)) and 24.3% in average of all the configurations in both applications.

We can conclude from these results that both underestimation as well as overestimation can cause undesirable results in terms of meeting deadline and energy consumptions. So, an accurate estimation of performance would be of interest.

TABLE II

RESULTS OF MATRIX COMPLETION FOR DIFFERENT SAMPLE CONFIGURATIONS

		7G-4C	3.5G-2C	7.2G-8C	7G-7C	4G-4C	14G-8C	15G-4C	3.75G-1C	7.5G -2C
		1	2	3	4	5	6	7	8	9
Real Values	wordcount	194	359	156	152	208	111	196	626	350
	hist-rat	167	310	115	117	175	98	174	508	306
Random 1	wordcount	198	359	155	152	227	112	**178**	500	298
	hist-rat	**167**	310	126	117	185	95	146	443	255
Random 2	wordcount	**143**	242	135	138	208	85	196	382	219
	hist-rat	99	**159**	101	117	134	98	89	235	133
Random 3	wordcount	127	200	131	129	208	111	105	292	**170**
	hist-rat	112	**178**	115	114	175	98	96	261	151
Random 4	wordcount	194	239	156	141	197	80	**114**	331	193
	hist-rat	167	201	115	109	157	66	101	277	**166**
SCS	wordcount	188	347	161	**152**	221	112	202	626	343
	hist-rat	149	279	125	117	**169**	87	160	508	281

V. CONCLUSION

In full profiling, the application is run on all the available configurations. This process is accurate to choose the best case but costly and time and energy consuming. An alternative solution is to run the application on a few number of

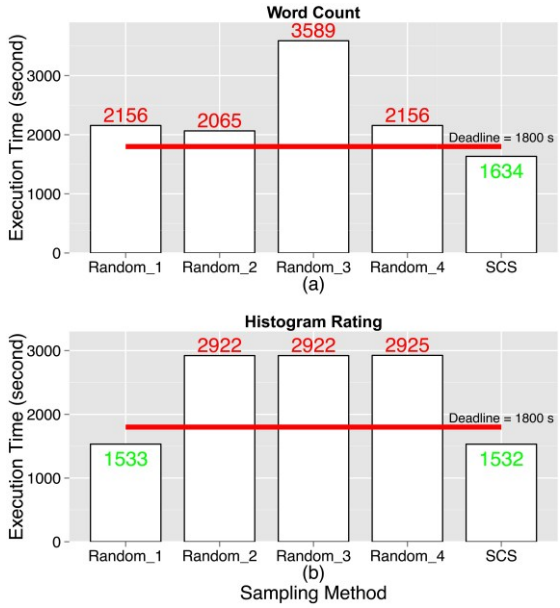

Fig. 7. Impact of estimation accuracy on deadline and resource usage

configurations and then using various prediction methods such as matrix completion to estimate the missing values. The prediction method can be very effective but its accuracy is a great concern. To improve the accuracy, we presented SCS method which uses Pearson Correlation Coefficient to select the most suitable sample configurations. This improves the accuracy of predicted missing values which will lead to better configuration selection for applications and consequently increases the energy efficiency.

The results in Fig. 6 reveal that while SCS provides the best pair of configurations for all the applications on average, it cannot supply the best answer for each specific application. This is a motive for further work in this area because obtaining the best choice per application can further improve the accuracy of estimation and lead to more precise decisions based on this knowledge.

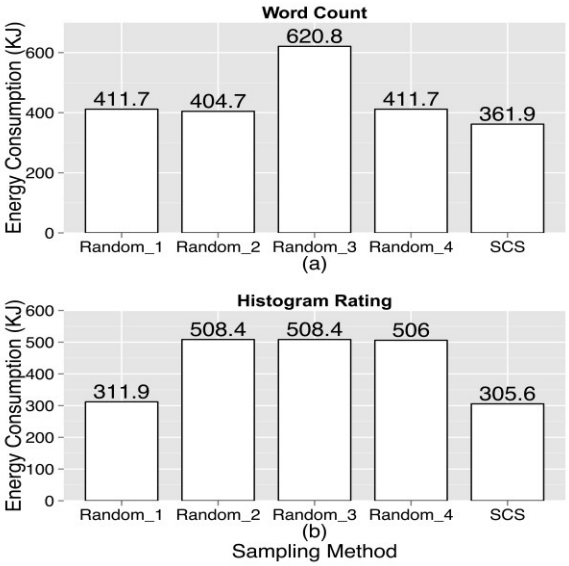

Fig. 8. Energy consumption of different selected configurations.

REFERENCES

[1] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services," in Conference on Innovative Data Systems Research (CIDR09), 2009.

[2] L. A. Barroso, J. Clidaras, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." *Synthesis lectures on computer architecture* 8.3, pp. 1-154, 2013.

[3] A. Faraz, and T. N. Vijaykumar. "Joint optimization of idle and cooling power in data centers while maintaining response time." In *ACM Sigplan Notices*, vol. 45, no. 3, pp. 243-256. ACM, 2010.

[4] V. K. Arghode and Y. Joshi, "Modeling Strategies for Air Flow Through Perforated Tiles in a Data Center," *IEEE Trans. on Components, Packaging and Manufacturing Technology,* vol. 3, pp. 800-810, 2013.

[5] V. Kontorinis and et al., "Managing distributed UPS energy for effective power capping in data centers," in *39th Annual International Symposium on Computer Architecture (ISCA),* pp. 488-499, 2012.

[6] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer,* pp. 33-37, 2007.

[7] I. Takouna, E. Alzaghoul, and C. Meinel, "Robust Virtual Machine Consolidation for Efficient Energy and Performance in Virtualized Data Centers," in *IEEE Conference on IoT, Green Comput. and Communic., and Cyber, Physical and Social Computing*, pp. 470-477, 2014.

[8] Y. Kejiang and et al., "Profiling-Based Workload Consolidation and Migration in Virtualized Data Centers," *IEEE Trans. on Parallel and Distributed Systems,* vol. 26, pp. 878-890, 2015.

[9] A. Varasteh and M. Goudarzi, "Server Consolidation Techniques in Virtualized Data Centers: A Survey," *IEEE Systems Journal*, in press.

[10] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting Spatio-Temporal Tradeoffs for Energy-Aware MapReduce in the Cloud," *IEEE Trans. on Computers,* vol. 61, pp. 1737-1751, 2012.

[11] N. Maheshwari, R. Nanduri, and V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework," *Fut. Gene. Computer Systems,* vol. 28, pp. 119-127, 2012.

[12] L. Mashayekhy, M. Nejad, D. Grosu, Q. Zhang, and W. Shi, "Energy-aware Scheduling of MapReduce Jobs for Big Data Applications," , *IEEE Trans. on Parallel and Distributed Systems,* in press.

[13] Z. Zhang, L. Cherkasova, and B. T. Loo, "Exploiting Cloud Heterogeneity to Optimize Performance and Cost of MapReduce Processing," *SIGMETRICS Perfor. Eval. Rev.,* vol. 42, pp. 38-50, 2015.

[14] M. Malekimajd, D. Ardagna, M. Ciavotta, A. M. Rizzi, and M. Passacantando, "Optimal Map Reduce Job Capacity Allocation in Cloud Systems," *SIGMETRICS Perform. Eval. Rev.,* vol. 42, pp. 51-61, 2015.

[15] C. Delimitrou and C. Kozyrakis, "The Netflix Challenge: Datacenter Edition," *Computer Architecture Letters,* vol. 12, pp. 29-32, 2013.

[16] E. J. CandÃ¨s and B. Recht, "Exact matrix completion via convex optimization," *Foun. of Comput. Math.,* vol. 9, pp. 717-772, 2009.

[17] Amazon EC2, http://aws.amazon.com/ec2/. Accessed Sept. 17th 2015.

[18] Mircrosoft Azure, https://azure.microsoft.com/en-us/. Accessed Sept. 17th 2015

[19] F. Ahmadand, S. Lee, M. Thottethodi, and T. Vijaykumar, "PUMA: Purdue MapReduce Benchmarks Suite," Technical Report Purdue ECE Tech Report TR-12-112012.

[20] S. R. Becker, E. J. CandÃ¨s, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Mathematical Programming Computation,* vol. 3, pp. 165-218, 2010.

[21] J. I. Marden, *Analyzing and modeling rank data*: CRC Press, 1996.

[22] Z. Zibin, W. Xinmiao, Z. Yilei, M. R. Lyu, and W. Jianmin, "QoS Ranking Prediction for Cloud Services," , *IEEE Trans. on Parallel and Distributed Systems,* vol. 24, pp. 1213-1222, 2013.