Performance Prediction Model in Heterogeneous MapReduce Environment

Yuanquan Fan[1], Weiguo Wu[1], Yunlong Xu[1], Yangjie Cao[2], Qian Li[1], Jinhua Cui[1], Zhangfeng Duan[1]
[1]*Department of Computer Science and Technology, Xi'an Jiaotong University,*
[2]*School of Software Technology, Zhengzhou University,*
No. 28, Xianning West Road, Xi'an, Shaanxi, 710049, P.R. China
eie.fyq@stu.xjtu.edu.cn

***Abstract*—MapReduce has emerged as a popular computing model for parallel processing of cloud computing. MapReduce performance analysis and modeling is needed to guide performance optimization and job scheduling. However, we observed that it is difficult to build a performance model due to various aspects of workload behavior and heterogeneity among cluster nodes in heterogeneous MapReduce Environments. To address the above issues, in this paper, we propose a novel performance prediction model for MapReduce in heterogeneous environments. This model consists of two components: (1) performance prediction model based on machine learning and (2) optimal parameters selection based on immune algorithm. Experiment results show that our model can accurately forecast the performance of MapReduce jobs that run in heterogeneous MapReduce systems.**

Keywords—MapReduce, cloud computing, Heterogeneity, performance prediction, machine learning.

I. INTRODUCTION

Along with the explosive development of Internet, Cloud Computing was proposed and has gained wide attention in academic and industrial areas. Cloud computing, which based on the data center, provide computing and storage resources to users dynamically and manage all kinds of workloads to meet the requirement of huge cloud computing applications [1]-[2]. MapReduce is a parallel computing model proposed by Google, which has been applied to cloud computing platform up to now. It is used to express distributed computations on massive amounts of data and processing on clusters of commodity servers [3].

In the MapReduce execution paradigm, a job is broken into two types of tasks: map tasks and reduce tasks. Map tasks are executed by mappers, and reduce tasks by reducers. Each mapper or reducer runs on a computing node. When a MapReduce job begins, each map task processes a partition of raw input data into key/values pairs. The key/values pairs are grouped by their keys, and thus form several partitions. Afterwards, each reduce task takes several partitions as its inputs and processes the partitions in parallel. The execution time of a MapReduce job is determined by the last finished reduce task. It is obvious that the execution time of a MapReduce job can be minimized if all reduce tasks finish at the same time. Hence, if the execution time of each task can be predicted, the scheduler can allocate tasks for each computing node more efficiently, and the performance of MapReduce jobs and the utilization ratio of cluster resource can be improved.

But it is very hard to predict the performance of each task accurately because of two reasons: 1) various aspects of workload behavior, and 2) the heterogeneity in MapReduce systems:

1) There are lots of configuration parameters for MapReduce systems (e.g., Hadoop). These parameters can significantly affect performance during job execution.

2) Performance of each node in MapReduce system can differ due to multiple generations of hardware in non-virtualized data centers or co-location of virtual machines (VMs) in virtualized data centers [4]. The heterogeneity can increase the inaccuracy of performance prediction. For example, the reduce task r_a and reduce task r_b process the same amount of work, however running these two tasks on two nodes with different capacity will lead to different execution times.

To address the performance prediction issue in the current MapReduce system, we propose a novel approach that can predict task execution time more accurately. The contributions of this paper are as follows:

1) Analyze historical tasks and characterize performance information. This performance information can strongly influence jobs or tasks performance, and also can be used to distinguish the computing capability between heterogeneous nodes.

2) Propose a performance prediction model based on least squares support vector repression [5] (LSSVR). The performance information derived from above can be used to determine the LSSVR model.

3) Use immune algorithm (IA) to determine parameter's value for proposed LSSVR prediction model.

The rest of this paper is organized as follows: in section 2, we introduce performance prediction model based LSSVR and IA. In section 3, we present the evaluation. In section 4, we discuss related work. Finally, we conclude the paper in section 5.

II. PERFORMANCE PREDICTION MODEL

During job execution, there are many counters that maintain the performance parameters for each job in MapReduce systems. Many of these parameters can be

978-1-4799-6239-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CIT.2014.122

obtained from logs. Some of these parameters are used to define the heterogeneity between computing nodes, which can then be used to determine and test the proposed LSSVR prediction model. In this section, we give the detailed descriptions to each performance parameter and present the performance prediction model based on LSSVR and IA (PPLI).

A. Performance parameter extraction

(1) Heterogeneity calculation for computing nodes

Computing capabilities of each node is different for different applications. For different applications, the computing capability of each node is different. There may be many reducers running on the same computing node. Hence, for the same application, we first evaluate the heterogeneity of each computing node. We define heterogeneity factor HF as the different computing capability between heterogeneous nodes, and HF can be calculated by

$$HF^i_{reduce} = \frac{N t^i_{reduce} - \sum_{i=1}^{N} t^i_{reduce}}{\sum_{i=1}^{N} (t^i_{reduce} - \frac{\sum_{i=1}^{N} t^i_{reduce}}{N})^2} \quad (1)$$

where N represent the number of heterogeneous computing nodes, and t^i_{reduce} denote the execution time of the reduce task on node i. For the equation *(1),* the size of input loads for each reduce task is the same with each other.

(2) The input loads size of each reduce task

A cluster based on MapReduce framework is composed of two node types: JobTracker node and TaskTracker nodes. The jobtracker, which runs on the JobTracker node, coordinates the job run. The tasktracker, which runs on TaskTracker node, is responsible for running the map tasks or reduce tasks. Every Task-tracker has a fixed number of slots for map tasks and reduce tasks respectively. A slot for map/reduce tasks is called a mapper/reducer.

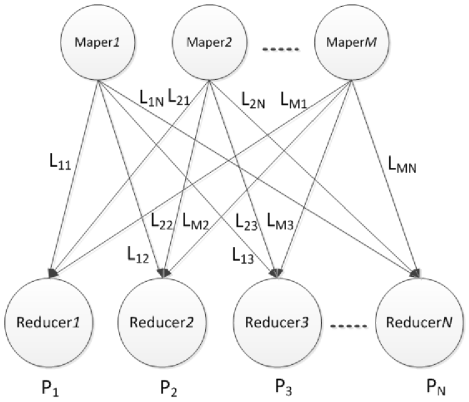


Fig. 1. Work distribution in MapReduce system.

In native MapReduce systems, like Hadoop, the output generated by each mapper is divided into partitions according to the partition function. Each partition is assigned to a reducer. As illustrated in Fig.1, there are M mappers and N reducers and output of mapper m_i is divided into work distribution vector:

$$L_i = (L_{i1}, L_{i2}, L_{i3}, \ldots, L_{iN}) \quad (2)$$

Where L_{ij} is the work fraction allocated to reducer r_j. Hence, the input loads size of reduce task r_i can be expressed as:

$$SL^i_{input} = \sum_{j=1}^{M} L_{ji} \quad (3)$$

B. Performance prediction model based on LSSVR

In machine learning, support vector machines, which are supervised learning models with associated learning algorithms that analyze data and recognize patterns, have been used for classification and regression analysis. We use least squares support vector regression machine to predict the performance of reduce tasks. The LSSVR model can be expressed as:

$$P(V_{input}) = W^T \cdot \varphi(V_{input}) + a \quad (4)$$

$$V_{input} = (H_{reduce}, SL_{input}, SL_{output}, SL_{mapnum}) \quad (5)$$

where $P(V_{input})$ denote the predicted execution time of reduce task I, and SL_{mapnum} denote the number of map tasks which allocate workloads to reduce task i and V_{input} denote the input vector which is composed by the performance parameters H_{reduce}, SL_{input} SL_{output} and SL_{mapnum}, and $\varphi(V_{input})$ denote a nonlinear function which maps the input space into a higher dimensional space, and W denote vector to the hyper plane, and a denote a real constant. Furthermore, the performance prediction model (4) can be solved by the following optimization problem:

$$\begin{aligned} &min \frac{1}{2}\|P\|^2 + \frac{1}{2}\mu \sum_{i=1}^{n} \xi^2 \\ &s.t.\ P\left(V^k_{input}\right) = W^T \varphi\left(V^k_{input}\right) + a + \xi_k \end{aligned} \quad (6)$$

where ξ is a slack variable. This problem can be solved by using Lagrangian function. The final predictive model can be expressed as:

$$P(V_{input}) = \sum_{k=1}^{s} \alpha_k K(V_{input}, V^k_{input}) + a \quad (7)$$

Where $K(V_{input}, V^k_{input})$ is the Kernal function. In this paper, we use radial basis function (RBF) as the kernel function which is expressed as:

$$K(V_{input}, V^k_{input}) = exp\left(-\frac{\left|\left|V_{input} - V^k_{input}\right|\right|^2}{\rho^2}\right) \quad (8)$$

where ρ is the kernel parameter. Moreover, the two parameters μ and ρ can influence the accuracy of performance prediction model. In order to select the best value of these two parameters for avoiding the case of over-fitting or under-fitting of LSSVR model, the immune algorithm [6] is adopted to determine the two parameters μ and ρ.

C. Parameter selection of performance prediction model based on IA

Immune algorithm is a new optimization algorithm that typically exploits the immune system's characteristics of

learning and memory to solve a problem. When using the IA to solve the optimization, the antigen is corresponding to the input of object problem such as objective function, and antibody is used to indicate the solution to the optimization problem [6]. The IA has been successfully applied to various application areas such as machine learning, anomaly detection, and job scheduling. In these application areas, IA demonstrates high efficacy and efficiency [7]. In this section, we use IA to identify optimal value of μ and ρ for performance prediction model based on LSSVR.

(1) Encoding of Antigen

The direct value encoding is used for antibody population because the parameters μ and ρ of LSSVR model are two real numbers. In value encoding, every antibody code is composed of two genes representing the two parameter μ and ρ. The *i*th antibody can be represented by $X_i = \{x_{i1}, x_{i2}\}$, where x_{i1} and x_{i2} is the value of parameter μ and ρ respectively, and x_{i1} *T[m, n]* and x_{i2} *T[c, d]*, and *m, n, c and d* are real numbers. For example, Fig.2 depicts an antibody coding.

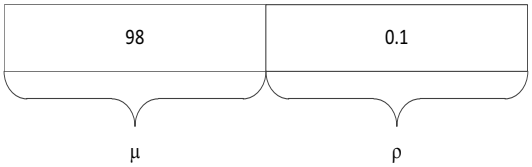


Fig.2 an example of antibody coding

(2) Diversity Calculation

The antibody concentration is represented by diversity. We use Euclidean distance to calculate the similarity between antibodies. Hence, similarity between antibodies can be expressed as:

$$Ant(x,y) = \sqrt{\sum_{i=1}^{2}(x_i - y_i)^2} \tag{9}$$

where x and y are two antibodies, and x_i represents the *i*th gene in antibody x and y_i represents the *i*th gene in antibody y. And the diversity of antibody a_i can be expressed as:

$$\left.\begin{aligned} Div_i &= \sum_{j=1}^{N} Num(a_i, a_j) \\ Num(a_i, a_j) &= \begin{cases} 1 & Ant(a_i, a_j) \le \theta \\ 0 & other \end{cases} \end{aligned}\right\} \tag{10}$$

where N is the number of antibodies and θ is the threshold of diversity.

(3) Affinity Calculation

Affinity is the fitness rate between antibody and antigen. In order to calculate the affinity between antibody and antigen, we need to first define the average error ratio (AERP) of prediction model based LSSVR. AERP represents the average prediction error that can be calculated as follows

$$AERP = \sum_{i=1}^{N} \left|\frac{p_i - e_i}{p_i}\right| \Big/ N \tag{11}$$

where p_i is the predictive execution time of reduce task i and e_i is the execution time of reduce task i, and N is the size of test data set of LSSVR model. The affinity between antibody and antigen can be calculated by

$$Att(s,p) = \frac{1}{1 + AERP} \tag{12}$$

where s is antibody and p is antigen.

(4) Crossover and Mutation

Crossover and mutation are two basic operators of IA. They are used to generate new antibodies. There are many ways to do crossover and mutation. In this work, the arithmetical crossover is adopted. For two selected antibodies $X_i=\{x_{i1}, x_{i2}\}$ and $X_j=\{x_{j1}, x_{j2}\}$, the new antibody $X_s=\{x_{s1}, x_{s2}\}$ created by crossover can be represented as

$$\left.\begin{aligned} x_{s1} &= \theta \times x_{i1} + (1-\theta) \times x_{i2} \\ x_{s2} &= \theta \times x_{j1} + (1-\theta) \times x_{j2} \end{aligned}\right\} \tag{13}$$

where $\theta \in [0, 1]$ is a random value. The Gaussian Mutation Operator is adopted to generate new antibody. For selected antibody $X_k=\{x_{k1}, x_{k2}\}$, the Gaussian mutation operator M changes X_k to

$$X'_k = X_k + N(0,1) \tag{14}$$

where $N(0,1)$ is an unit Gaussian distributed random value.

D. Performance prediction algorithm based on LSSVR and IA

Based on above analysis, we combine LSSVR model and immune algorithm and present a performance prediction algorithm based on LSSVR and IA (PPLI). The PPLI is introduced as below:

Step 1: calculate the heterogeneity factor *HF* of each computing node according to *(1)* and input loads size of each reduce task according to *(2)- (3)*. At the same time, extract the output loads size of each reduce task from task's logs and the number of finished map tasks.

Step 2: construct the input characteristic matrix $X_S = [(V^1_{input})^T, (V^2_{input})^T, \dots, (V^N_{input})^T]$ for LSSVR and V^i_{input} can be obtained according to *(5)*.

Step 3: normalize the input characteristic matrix X_S. For column vector V^i_{input} *in metric* X_S, We first find the minimum value E_{min} and maximum value E_{max} for vector V^i_{input} . Then the normalized value of e_j for vector V^i_{input} in the *j*th rows is calculated as

$$Normalized(e_j) = \frac{e_j - E_{min}}{E_{max} - E_{min}} \tag{15}$$

The remaining variables in the rows of matrix X_S are normalized in the same way.

Step 4: initialize LSSVR model, and build the initial antibodies set A_{init} by random selecting the values of two parameters a and ρ of LSSVR model.

Step 5: For each antibody in set A_{init}, we calculate the *AERP* by equation *(11)* and affinity between antibody and antigen according to *(12)*. When the iterations reached a given scale or the affinity between antibody and antigen reached the given scale, the antibody with the highest affinity to antigen is selected as the optimal parameter for LSSVR

model to predict the performance of reduce tasks. Otherwise, it is necessary to execute **Step 6.**

Step 6: select and save the antibody An_h with the highest affinity from set A_{init}. We then renew the initial set of antibodies A_{init} through randomly generating a new antibody to replace the An_{hi}

Step 7: calculate the similarity between antibodies according to *(9)*, and the diversity of each antibody according to *(10)* and select the antibody a_i with the highest diversity. For antibody a_i, the antibodies with lower similarity than θ are deleted from set A_{init}. We then renew the initial set of antibodies A_{init} through randomly generating new antibodies to replace the deleted antibodies.

Step 8: calculate the affinity of newly generated antibodies, and utilize Roulette wheel selection to select antibodies for crossover and mutation.

Step 9: perform crossover operation on two selected antibodies according to the rule introduced in section 2.3.4. For each new antibody to be produced, the Gaussian mutation operation, which is introduced in Section 2.3.4, is performed according to crossover probability.

Step 10: update the antibody set A_{init} by replacing the antibody, which has the lowest affinity, with the antibody An_i that has the highest affinity, and return to **Step 5.**

III. EXPERIMENT

In order to evaluate the performance and benefits of our performance prediction model, we carried out experiments on a virtualized cluster. We used Xen [8] virtualization software to build heterogeneous virtualized environments. The test bed is composed of three Inspur Yingxin NF5240M3 Servers, each of which is equipped with 16 GB of memory, a single 250 GB SATA driver and four Intel(R) Xeon(R) E5-2420 1.9GHz six-core processors. The operating system is RHL 5.4 (Linux kernel 2.6.18). On each server, we run six to eight virtual machines using Xen software, and each virtual machine is given different amounts of memory and different number of processors. The detailed specification is list in Table I.

TABLE I. THE HETEROGENEOUS VIRTUALIZED ENVIRONMENTS.

VM Name	CPU(s)	Memory (GB)
host	4	3
g01	8	3
g02	3	1
g03 g04 g14 g15 g24 g25	2	1
g05 g16 g17 g18 g26 g27 g28	1	1
g11 g21	6	4
g12 g22	4	2
g13 g23	4	1

To collect test data for PPLI, we used the following applications:

K-means: K-means clustering [9] is a widely used partition algorithm. It partitions the samples into clusters by minimizing a measure between the samples and the centroid of the clusters. We obtain the K-means clustering workload from the Purdue MapReduce Benchmarks Suite and use the free dataset [10] as the workload's input.

Sort: Sort [11] is a strings sorting algorithm that sorts the input sequence files into the output sequence files. We use the RandomWriter [12] to generate 1 GB of random data per node.

CloudBurst: CloudBurst [13]-[14] is a new algorithm optimized for mapping next-generation sequence data to the human genome and other reference genomes. We use CloudBurst workload to process the portions of human genome dataset of 1000 Genomes Project [14].

Ranked-inverted-index (RII): Ranked-inverted-index [9] is an algorithm that takes lists of words and their frequencies in a file, and generates lists of files containing the given words in decreasing order of frequency. We obtained the Ranked-inverted-index workload from Purdue MapReduce Benchmarks Suite, and its input from the free dataset [9].

In our evaluation, we used the following configuration parameters. The HDFS block size is set to 64 MB by default, and each virtual node is configured with two map task slots and one reduce task slot, respectively.

In order to collect application logs, we ran the above four applications. For each application, the total input data size is changed from 10GB to 20GB. So, the execution time of each job can be very different. We collected the logs for above four application, and use MRLA tool [15], which is developed by our research group, to analyze historical jobs and tasks logs and built the data set which composed of 630 samples. This initial data set is divided into two parts: the training set and the testing set. The training set, which is composed of 504 samples, is used to determine the optimal LSSVR model. For each heterogeneous node, we randomly selected two samples in one application. These samples are put into the testing set that is used to measure predicting performance of our model. The size of testing set is 126 samples. We implemented performance prediction model based on java program language.

To evaluate the benefits of PPLI, we considered the following metrics: (1) the validity analysis of our performance prediction model and (2) forecast accuracy.

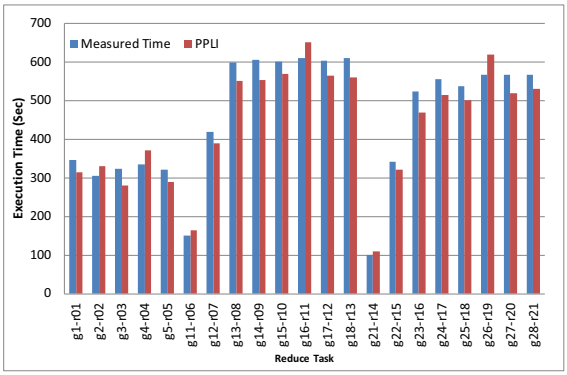
A. Validity analysis

In order to evaluate the validity of our prediction model, we use the RandomWriter to generate 15G of data as the input of Sort and run this application. After execution of one Sort application was completed, we analyzed job logs and predicted the performance of reduce task by using our model. For RII and Cloudburst application, we randomly select a sample for a heterogeneity computing node, and used the model trained by data set composed of 504 samples to forecast execution time of each reduce task. The experiment results are shown in Fig. 3.

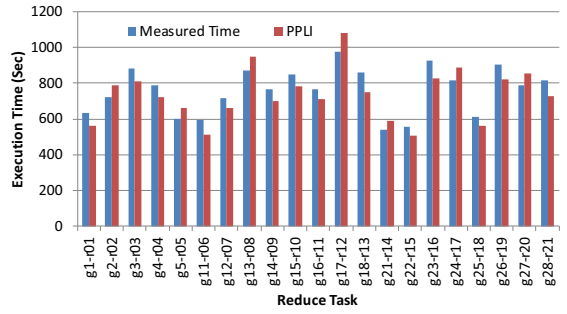
For Sort application, after map function was executed, each reduce task can be allocated work fairly by using hash partition method. But, as shown in Fig.3 (a), because of different computing capability of each nodes, the execution time is different among reduce tasks. For example, the reduce task *r1,* which is executed on high performance node *g01*, can finish work early than reduce task *r18* which is executed on low performance node *g18.* Because of dynamic

characteristics in heterogeneous MapReduce environments, reduce tasks, which are executed on nodes with similar computing capability, does not always finish at the same time. But our prediction model can forecast the execution time of each reduce task accurately. This is because the heterogeneity characteristics of computing node are taken into consideration in our model. We evaluate the heterogeneity of each computing node by using heterogeneity factor *HF*.

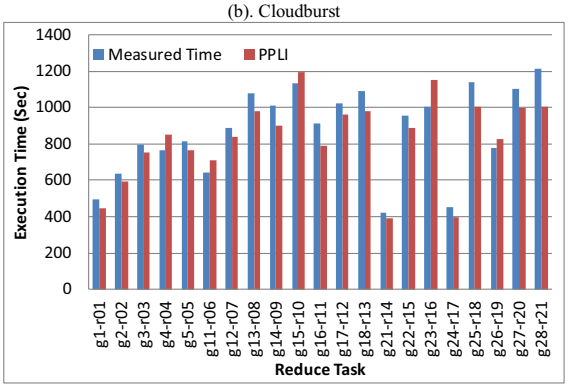
For RII and Cloudburst applications, the size of input load for each reduce task is different. But, as shown in Fig.3(b) and Fig.3(c), the execution time of each reduce can be forecast accurately by PPLI.



(a). Sort



(b). Cloudburst

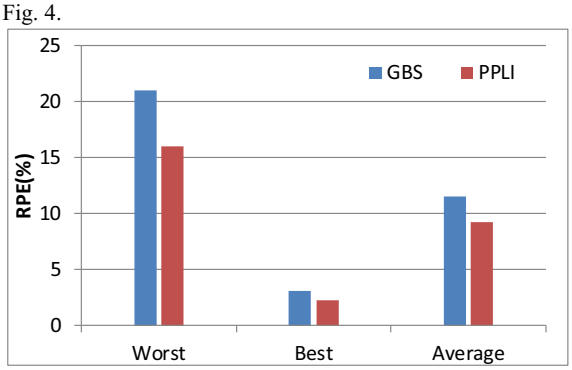


(c). RII

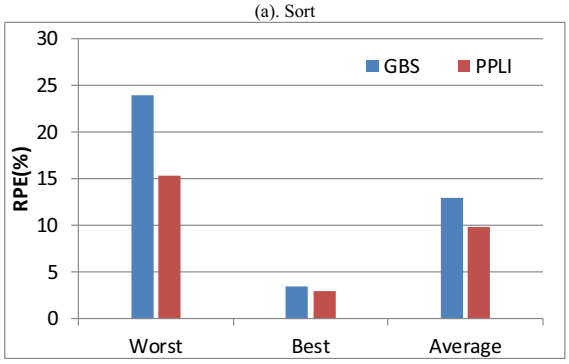
Fig.3 actual execution time and forecasting result of three applications

B. *Forecast accuracy*

In this section, we discuss the necessity of select parameters for performance prediction model based on LSSVR, and evaluate the benefits of our model based on immune algorithm. We use prediction model based on grid search algorithm (GBS) to predict the performance of reduce tasks, and compared the predicted results to PPLI. We use the relative prediction errors (RPE) to evaluate the performance of above two models. The experiment results are shown in Fig. 4.



(a). Sort



(b). RII

Fig.4 Relative prediction errors

As shown in Fig.4, for GBS, it cannot obtain more accurate results. This is because the two parameters μ and ρ of LSSVR model is selected based on grid search algorithm, and the optimal parameters are used to predict the performance of reduce tasks. By using GBS, the largest RPE is 21% for Sort application and 24% for RII application. However, compared with prediction model based on GBS, the forecasting accuracy of PPLI is higher by 5% and 8.7% respectively in the worst case scenarios. When using PPLI, the largest RPE is only 16% for Sort application and 15.3% for RII application. This is because of two reasons. The first one is that PPLI has the ability of global searching. And the searching space can be quickly narrowed by saving the antibody with the highest affinity for the next generation. The second one is that PPLI has the ability of keeping the diversity of solution. In PPLI, the antibody concentration is changed during evolutionary operation. The antibody with highest diversity will be selected. For the highest diversity, the antibodies with lower similarity are replaced by new antibodies. So it is easy for PPLI to escape from the locally optimal solutions. From above two reasons, the convergence rate of PPLI is improved significantly. While the speed at which grid-based searching algorithm provides solution is slower. This is because the GBS is a blind search algorithm. When the mesh size is smaller, it has the opportunity for GBS to search optimal solution. But it is more expensive and time-consuming than PPLI.

IV. RELATED WORK

In heterogeneous MapReduce environments, performance

prediction is an important problem because it can be used to design an efficient scheduling policy to optimize job performance, and tune the system configuration. The current MapReduce system implementation has overlooked this problem. However, there are some studies has been carried out by many organizations and agencies.

Song et al. [16] propose a framework to predict the performance of Hadoop jobs. The execution of a job is divided into different parts, and each part is set different weight. Then a locally weighted regression method is used to predict the performance of Hadoop jobs. In order to optimize configuration parameters for many programs, Herodotou et al. [17] propose a called What-if Engine to estimation the system cost. What-if Engine can collect detailed information from MapReduce programs and predict job performance by turning the configurations of Hadoop. Chen et al. [18] use exponentially weighted moving average to predict process speed and predict a task's remaining time. Ahmad et al. [10] analyze the effect of MapReduce's build-in load balancing and heterogeneity of computing node, and propose a model called predictive load balancing of reduce computation (PLB). However, in PLB, the processing speed of Map computation is used to estimate the performance of Reduce tasks, and it is not accurate because of dynamic characteristics in heterogeneous MapReduce environments. All of these methods are based on cost estimation and not similar to our work.

In order to tackle the challenge of configuring the Hadoop environment, Ganapathi et al. [19] present initial workload generator which is used to evaluate system configurations and proposed a model to predict resource requirements for cloud computing applications based on Kernel Cannonical Correlation Analysis. For homogeneous MapReduce environments, Yang et al. [20] propose a statistical analysis approach to identify the relationships among workload characteristics. This work is similar to our work. However, this work focused on homogeneous MapReduce environments and the heterogeneity among nodes is not considered.

V. CONCLUSION

MapReduce performance analysis and modeling is important for guiding performance optimization and job scheduling. In this paper, we first analyze the historical tasks and characterized performance information that can strongly influence performance of jobs or tasks. We then use the MRLA tool to collect performance information for building the data set. After that we built a performance prediction model based on machine learning method and use immune algorithm to select best parameters for the model. We implemented the proposed performance prediction model. The experiment results show that our approach achieved significantly higher prediction accuracy for three different applications ran on a heterogeneous Hadoop cluster with 22 virtual machines.

VI. ACKNOWLEDGMENT

This work is supported by National High-Tech Research and Development Plan of China under the grant No.2011AA01A204, 2012AA01A306, National Key Technology R&D Program under grant No.2011BAH04B03, National Natural Science Foundation of China under grant NO.61202041, U1304603 and Fundamental Research Funds for the Central Universities under grant NO.08142007.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. "Above the Clouds: A Berkeley View of Cloud computing", *Technical Report* No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[2] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm. "What's inside the Cloud? An architectural map of the Cloud landscape", *In Proceedings of ICSE Workshop on Software Engineer ing Challenges of Cloud Computing*, 2009, pp. 23-31.

[3] J. Dean and S. Ghemawat. "MapReduce: Simplified data processing on large clusters", *in proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation* , 2004, pp.137-150.

[4] M. Zaharia, A. Konwinski, A. D. Joseph, and R. Katz, I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments", *in Proc. 8th USENIX Symposium on Operating Systems Design and Implementation*, USA, 2008.

[5] Johan A K S, Tony V G, Jos D B, Bart D M, Joos V. "Least Squares Support Vector Machines", *in Proc. World Scientific*, 2002.

[6] Dasgupta D. "Artificial Immune Systems and Their Applications", Berlin Heidelberg: Springer-Verlang, 1999.

[7] KP Lin, PF Pai, SL Yang. "Forecasting concentrations of air pollutants by logarithm support vector regression with immune algorithms", *Applied Mathematics and Computation,* 2011,pp: 5318 – 5327 .

[8] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, "Xen and the art of virtualization", *in Proc. ACM*, vol.37, no.5, pp.164-177, 2003.

[9] Esteves, R. M., R. Pais, Rong Chunming, "K-means clustering in the cloud – a Mahout test", *in Proc. 25th IEEE Conf. Adv. Inf. Networking Appl*., Biopolis, Singapore, pp.514-519, 2011.

[10] Farz Ahmad, Srimat T. Chakradhar, Anand Raghunathan, and T. N. Vijaykumar, "Tarazu: optimizing MapReduce on heterogeneous clusters", *in Proc. ACM SIGARCH Computer Architecture News*, pp.61-74, 2012.

[11] Sort, http://wiki.apache.org/hadoop/Sort.

[12] RandomWriter, http://wiki.apache.org/hadoop/RandomWriter.

[13] M. C. Schatz, "CloudBurst: highly sensitive read mapping with MapReduce", *in Proc. Bioinformatics*, vol.25 no.11 pp.1363-1369, 2009.

[14] A Deep Catalog of Human Genetic Variation, http://www.1000genomes.org.

[15] Yuanquan Fan, Weiguo Wu, Depei Qian, Yunlong Xu. "Load Balance in Heterogeneous MapReduce Environments", *in Proc. Proc. 15th IEEE HPCC*, 2013, pp: 820-825.

[16] Ge Song, Zide Meng, Fabrice Huet, Frederic Magoules, Lei Yu and Xuelian Liu. "A hadoop MapReduce Performance Prediction Method", *in Proc. 15th IEEE HPCC*, 2013, pp: 820-825.

[17] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of mapreduce program", *PVLDB*, VLO.4, NO.11, pp.1111-1122, 2011.

[18] Q. Chen, C. Liu, and Z. Xiao, "Improving mapreduce performance using smart speculative execution strategy", *in Proc. IEEE Transactions on Computers*, 2013.

[19] 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.

[20] Hailong Yang, Zhongzhi Luan, Wenjun Li, Depei Qian. "MapReduce Workload Modeling with Statistical Approach", *in Proc. J. Grid Comput.* 10(2), 2012, pp:279-310.