Stable Modeling on Resource Usage Parameters of MapReduce Application

Yangyuan Li
Department of Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Hungary
H-1117, Magyar tudósok körútja 2, Budapest, Hungary

Department of Computer Science, Xi'an Siyuan University, China
710038, Xi'an, China
wlqsb@hotmail.com

Abstract
Currently, Hadoop MapReduce framework has been applied to many productive fields to analyze big data. MapReduce applications based on the MapReduce programming model are used to generate and process such huge data. Due to various computational purpose, MapReduce applications have different resource requirements. For specific applications, the resource bottleneck of the cloud computing platform must inevitably impact its executive performance. Therefore, identification of the bottleneck about the allocated resource for MapReduce applications is crucially needed from the viewpoint of either cloud operators or program developers. In this paper, we model the relationship of resource usage parameters of MapReduce applications using multiple linear regression methods and investigate the minimum sampling time for stable modeling. Based on the analysis, we propose the approach which can be used to build stable performance model to expose the bottleneck resource of Hadoop platform and give the effective optimization suggestion.

Keywords: MapReduce application; resource bottleneck; resource usage parameters; multiple linear regression; stable modeling; minimum sampling time

1. Introduction

As a popular computation framework, the Apache Hadoop (Apache.org, 2017) has been applied broadly to big data processing and analytic in many IT companies, such as Facebook, etc. Map/Reduce (Vavilapalli et al., 2013) is a computational programming model of Hadoop for processing huge amount data either in public clouds or in private clouds. Based on this programming model, the developers can write their MapReduce applications for different big data processing purposes, which may show various computation resource requirements. Though cloud computing techniques (Geneva, 2012) claimed that commodity computers can offer the unlimited resources over the internet, the over-provisioning or unbalanced-provisioning resource to MapReduce application should be avoided. Therefore, the precise identification of the bottleneck problem of allocated resource for MapReduce application is crucially needed from the viewpoint of either cloud operators or developers.

Many efforts have been spent on the related studies. For example, L. Bautista Villalpando et al. (Bautista Villalpando, April, & Abran, 2014) modeled the relationship between performance measurements of big data application and the quality concepts of software engineering. Subsequently, on the basis of Amdahl's law regression methods (Rodgers, n.d.), Issa, J A et al. (Issa, 2015) proposed an estimation model to estimate performance and total processing time versus different input sizes for a given processor architecture. He intended to explore the relationship between processing time and input size of data. Glushkova. et al.(Glushkova, Jovanovic, & Abelló, 2017) built a new performance model for Hadoop 2.x, which use the queuing network model to capture the execution flow of a MapReduce job and take architectural changes into account. These models proposed above only concerned the performance analysis with the given resource and did not mention the allocated resource declining the performance of Hadoop platform. A resource reuse optimization mechanism for MapReduce short jobs was developed by Shi. et al. (Shi et al., 2016), which effectively shortened the execution time of these jobs and significantly improved the resource utilization of cluster. Nghiem. et al. (Nghiem & Figueira, 2016) put forward a novel algorithm for optimal resource provisioning to get the exact amount of task resources, which represented the best trade-off point between performance and energy efficiency for any MapReduce job running on Hadoop. According to the real social network data, Bakratsas. et al.(Bakratsas, Basaras, Katsaros, & Tassiulas, 2017) evaluated the performance of three algorithms when using solid state drives and hard disk drives as underlying storage for Hadoop's MapReduce. However, these papers did not mention the bottleneck resource of Hadoop platform when running an unknown application. As a result, the performance analysis model for exploring the bottleneck resource might be beneficial to gaining better resource provisioning for cloud operators as well as designing high-performance MapReduce application for developers.

We build the regression model for a suite of typical MapReduce benchmark applications, including Wordcount, Wordmean, Wordmedian, Grep, Pi, Teragen, and Terasort. The first four applications respectively calculate the number of occurrence of words, the average length of words, the median length of words, and the matches to a regex in a text file. The Pi application uses the quasi-Monte Carlo methods(Levy, 2016) to estimate the value of the pi number. The Teragen application is used to generate rows of data to a file. Lastly, the Terasort application sorts the generated data from Teragen. All these benchmark applications require various resource intensive requirements. By applying the multiple linear regression methods(Ross, 2017a), the present study models relationships amongst resource usage parameters as well as significantly lagged usage parameters. The obtained model shows the resource bottleneck of MapReduce applications on Hadoop platform. Furthermore, the sampling time for each MapReduce application has a substantial impact on the fit quality of the regression model. The minimum sampling time of each application is the essential condition for stable modeling. Thus, the minimum sampling time of each MapReduce application is also investigated. To the best of our knowledge, this is the first attempt to use multiple linear regression methods to model the relationship of resource usage parameters as well as to investigate the minimum sampling time for Mapreduce applications. This study about exploring the bottleneck resource of cloud computing platform by building the stable performance model for applications is the gap in this era.

In this work, our contributions are:

- We model the relationship of resource usage parameters of several Mapreduce applications using multiple linear regression methods.
- We investigate the minimum sampling time for stable modeling on resource usage parameters for Mapreduce application.
- We present an approach to explore the bottleneck resource of Hadoop cloud computation platform.

This body of the paper is organized as follows. Section 2 presents the methodology of modeling on resource usage parameters. In section 3, we explain the methodology of obtaining the minimum sampling time for stable modeling. The detailed results are presented in section 4. Section 5 shows the discussion and analysis of results. Conclusion and future direction are wrapped into the last section.

2. Methodology of Modeling on Resource Usage Parameters

2.1. Data collection

The workloads for those applications, including Wordcount, Wordmean, Wordmedian, Grep, are generated with the use of hdfswriter.jar (written in java). The corresponding workload is 100 GB text file. The workload of Terasort is 60 GB data generated from Teragen. The usage parameters of Teragen are captured when it generates 180 GB data. Pi is performed with 2000 map tasks in 10000000 times.

Collectl utility is used to measure the total percentage of time spent of CPU processing job, the total memory usage, the total KB read/second from hard disk and the total KB write/second to hard disk of MapReduce applications with time resolution 1 s. Note that Collectl is a lightweight application that only occupies extremely few resources to gather time series data.

2.2. Explore Non-randomness

The collected data in this work are time series data. Empirically, the autoregressive pattern always exists in time series data. Extracting autoregressive term is typically used to eliminate the autoregressive pattern of data. The autocorrelation function (Bhattacharya & Burman, 2016) is used to explore the non-randomness of each resource usage parameter and the partial autocorrelation function (Bhattacharya & Burman, 2016) is applied to determine the number of the significant autoregressive term. A quarter of the amount of observations is used to estimate autocorrelation and partial autocorrelation according to Box and Jenkins (Box, Jenkins, & Reinsel, 1994).

Let x_t denote the value of a time series at time instant *t*. The autocorrelation between x_t and x_{t+k} is given by autocorrelation coefficient, denoted by ρ_k, for $k=\{1, 2, 3, ..., n\}$.

$$\rho_k = \frac{\sum_{t=1}^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2} \qquad (1)$$

The autocorrelation plot (ACF plot) (Box et al., 1994) shows the autocorrelation coefficients of a time series data at various lags. If at least one autocorrelation is significantly non-zero, it gives a strong evidence of non-randomness.

The partial autocorrelation between x_t and x_{t+k} is defined by the conditional autocorrelation coefficient and is conditional on $x_{t-1}, ..., x_{t+k-1}$, denoted by ρa_k, for k=$\{1, 2, 3, ..., n\}$.

$$\rho a_k = \frac{Covariance(x_t, x_{t+k}|x_{t-1}, \cdots, x_{t+k-1})}{\sqrt{Variance(x_t|x_{t-1}, \cdots, x_{t+k-1})Variance(x_{t+k}|x_{t-1}, \cdots, x_{t+k-1})}} \qquad (2)$$

The partial autocorrelation plot (PACF plot) (Box et al., 1994) plots the partial autocorrelation coefficients at various lags. When non-randomness of time series data is significant, PACF plot is used to find the number of the autoregressive terms. In this work, the largest PACF coefficient is used to identify the number of the autoregressive term.

The basic assumption of drawing ACF plot and PACF plot is that time series data is stationary. The Ljung–Box Q test(Shams, Haji, Salman, Abdali, & Alsaffar, 2016) is used to identify the stationary of time series data as well.

2.3. Identify Linear Pattern

2.3.1 Correlation scatter matrix and Correlation Matrix

The correlation scatter matrix is used to intuitively display the correlation of each pair of usage parameters and the correlation matrix is used to numerically show their correlation coefficients which are measured by Pearson Correlation Coefficient (Molugaram & Rao, 2017b).

The Pearson correlation coefficient is denoted by

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad (3)$$

where n is the number of samples, x_i, y_i are the single samples indexed with i, $\bar{x}$ and $\bar{y}$ are the sample means.

Pearson correlation *r* is between -1 and 1. The closer the value of *r* gets to zero, the greater the variation the data points are around the line of the best fit. The absolute value of *r* represents the strength of correlation. The sign of *r* presents the relevant direction of these variables. The positive correlation shows that the pair of variables has the same direction. In contrast, negative one presents the opposite directions of variables.

The correlation matrix exhibits correlation coefficient between pairs of resource usage parameters as well as its autocorrelation coefficient of MapReduce applications.

2.3.2 Feasibility of Linear Regression Model

The variation of the base error rate is used to discriminate the feasibility of multiple linear regression models. The Z-score (standardized coefficient) is a statistical measure to test the effect of dropping that variable from the model. It is used to test the hypothesis of a particular coefficient $\beta_j=0$. The Z-score(Warner, 2016) is denoted by

$$z_j = \frac{\widehat{\beta_j}}{SE} \qquad (4)$$

where *SE* denotes standard error of estimate coefficient $\widehat{\beta_j}$. Under the null hypothesis test of $\beta_j=0$, z_j is distributed as a t distribution with n-m-1 degrees of freedom (Molugaram & Rao, 2017a), where m is the number of predictors in the model, n is the number of observations, a large absolute value of z_j provides evidence to reject this null hypothesis. The absolute value of Z-score greater than 2 refers to approximately the significant level at 5%.

The F statistic (Molugaram & Rao, 2017a) is used to test the significance of a group of coefficients simultaneously. It measures the change of residual sum of squares (RSS) as dropping a group of coefficients simultaneously in the bigger model. Under the null hypothesis, if the smaller model is correct, the F statistic will be distributed as a F distribution. Based on the obtained F statistic, the corresponding p-value (significant level is 0.05) can be calculated. The p-value larger than 0.05 proves that the dropping of insignificant variables would not impact the fit performance of the model.

Based on the given significant variables (the absolute value of Z-score larger than 2), the $Improve_{error_{rate}}$ is used to show the improvement of prediction performance after using the linear fitting. The $Improve_{error_{rate}}$ is denoted by

$$Improve_{error_{rate}} = \frac{Base_{error_{rate}} - Test_{MSE}}{Base_{error_{rate}}} \qquad (5)$$

The $Test_{MSE}$ is the test mean squared error of the true value and the prediction value of the response. In contrast, the $Base_{error_{rate}}$ is the mean squared error of the true test value of response and the mean training value of response and is denoted by

$$Base_{error_{rate}} = \frac{\sum_{i=1}^{n}(y_i - \bar{y}_{train})^2}{n} \qquad (6)$$

where n is the length of test data, y_i is the ith true value of the response, $\bar{y}_{train}$ is the mean training value of the response.

Therefore, the positively higher improvement of $Base_{error_{rate}}$ provides strong evidence to prove the feasibility of the multiple linear regression methods.

2.4. Modeling on Resource Usage Parameters

2.4.1. Multiple Linear Regression Methods

Multiple Linear Regression (Ross, 2017a) is used to model resource usage parameter on associated historical usage parameters as well as other usage parameters. The ordinary least squares approach (Linton, 2017) is used to estimate the coefficient of a model. The estimated coefficients of the model would be able to reveal the quantitative relationships among these usage parameters.
Multiple linear regression model takes the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon. \qquad (7)$$

where X_j represents the jth predictor and β_j quantifies the association between that predictor and the response. β_j is a constant and represents the average effect on Y of one units' increase in X_j, holding all other predictors fixed.

The coefficients of model β_0, β_1, ..., β_p are unknown and must be estimated. The estimated regression coefficients are denoted by $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_p$ and would be obtained by minimizing Residual Square Sum (RSS) of regression model.

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2 \qquad (8)$$

According to given estimated coefficients, prediction can be conducted by using following formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p \qquad (9)$$

Theoretically, the Least Square Regression claims that if:

$$H_{model} = \begin{bmatrix} 1 & x_{11} & x_{12} & & x_{1p} \\ 1 & x_{21} & x_{22} & & x_{2p} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & x_{n2} & & x_{np} \end{bmatrix}, \quad H_{accual} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \beta = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} \qquad (10)$$

then the estimate coefficient vector β will be calculated as (Rizvandi, Nabavi, & Hessabi, 2005) by minimizing above RSS:

$$\beta = (H_{model}^T H_{model})^{-1} H_{model}^T H_{accual} \qquad (11)$$

where $(.)^T$ denotes a transpose matrix. The vector β denotes the set of estimated coefficient of regression model.

2.4.2. Multicollinearity problem

The variance inflation factor (VIF) is used to detect multicollinearity (Yu, Jiang, & Land, 2015) (also collinearity). The elimination of multicollinearity is able to make sure of the independence assumption of predictors for multiple linear regression model. Based on (Gareth James, Daniela Witten, 2013), the VIF is denoted by

$$VIF((\beta_{\downarrow}j)\hat{}) = 1/(1 - R_{\downarrow}(X_{\downarrow}(j) \mid X_{\downarrow}(-j))^{\uparrow}2) \qquad (12)$$

where $R_{\downarrow}(X_{\downarrow}(j) \mid X_{\downarrow}(-j))^{\uparrow}2$ is the R2 from a regression of X_j onto all of the other predictors. If $R_{\downarrow}(X_{\downarrow}(j) \mid X_{\downarrow}(-j))^{\uparrow}2$ is close to one, then collinearity is present, and the corresponding VIF will be a larger one. The smallest VIF value is 1 and indicates the complete absence of collinearity. As a rule of thumb, a VIF of 5 or 10 and above indicates a multicollinearity problem(O'Brien, 2007). In our case, the threshold of VIF is 10. If the VIF value of predictor variable doesn't exceed 10, the model will keep it. Otherwise, it has to be dropped.

Note that the process of eliminating multicollinearity keeps running until no VIF larger than 10. Meanwhile, non-regressive term is preferentially eliminated.

2.4.3. Best-Subset Selection

Best subset selection is used to explore the model with the smallest residual sum of squares from those with the subset of size k, for each k∈ {0, 1, 2, ... *p*}, *p* is the maximum number of predictors. Firstly, the models with the smallest residual sum of squares (RSS) are chosen from all possible various model groups with the fixed subset of size k, for each k∈ {0, 1, 2, ... *p*}. Each model group has same model size (the number of predictors). Secondly, ten-fold cross-validation

(Witten, Frank, Hall, & Pal, 2017) is used to calculate Test MSE (Mean Squared Error) (Theodoridis, 2015) for the given models in each model group.

The one-standard-error rule (Hastie, Tibshirani, & Friedman, 2009) is used to choose the simplest model. The standard error (SE) of a statistic (most commonly the mean) is the standard deviation of its sampling distribution (Everitt & Skrondal, 2010). The standard error of the mean ($SEM_{Test_{MSE}}$) (Theodoridis, 2015) is denoted by

$$SEM_{Test_{MSE}} = \frac{se_{Test_{MSE}}}{\sqrt{k}} \quad (13)$$

where $se_{Test_{MSE}}$ is the sample $Test_{MSE}$ standard deviation, k is the size (number of $Test_{MSE}$) of the sample folds.

2.4.4. Choose Interaction Term

In statistics, an interaction (Usset, Staicu, & Maity, 2016) may arise when considering the relationship between three or more variables and describes a situation in which the simultaneous influence of two variables on a third is not additive. The significant increase for R2(Ross, 2017b) and decrease for residual standard error(RSE) (Usset et al., 2016) is used to determine the join of an interaction term for the regression model. The threshold of increased percentage of R2 is set as 3%.

2.4.5. Estimate coefficients and statistical metrics

To obtain credibly estimated coefficients and statistical metrics, their means are calculated by performing regression calculation for ten times on different resampled training data instead of simply fitting on all training data. The means are closer to the true value than estimates on regression calculation on full training data.

2.4.6. Validate Independence of Residual

One of the most important assumptions of the linear regression model is that the error terms, $\epsilon_1, \epsilon_2, \cdots, \epsilon_n$, are uncorrelated. If the error terms are uncorrelated, residual of the model is random and provides the evidence for the unbiased estimates for the true standard errors. The autocorrelation plot of residual is used to check if the hypothesis is valid.

3. Methodology of Stable Modeling

The stable modeling takes the stability of two metrics into account: estimated coefficient and statistical measures (R^2 and residual standard error) of the given regression model. The minimum sampling time of these two stable metrics is used to represent an essential time for stable modeling.

3.1. Experimental Design

This experiment was designed to repeatedly execute the regression calculation procedure until the full training data is run out. The size of training data is 50 and increases stepwise by 50 for each training model until data is run out. The total number of modeling depends on the total amount of observations divided by 50. The estimated coefficients and the statistic metrics for each modeling were calculated and recorded. Based on these records, the corresponding sensitivity degree of error rate is calculated to decide the minimum sampling time for stable modeling.

3.2. Analyze the stability of estimate coefficient

The minimum sampling time of stable estimated coefficients of a regression model is revealed. The stable condition of estimate coefficients is the value does not change over time. But in real MapReduce environment, the constant estimated coefficients are almost impossible.

Alternatively, the estimate coefficients might fluctuate and gradually converge to the stable state. Theoretically, it will reach a stable state in an infinite time.

The error rate of a coefficient is used to represent the sensitivity degree. The threshold of sensitivity degree is set to 0.1. It means that the estimated coefficients are arriving stability when the percentage of the difference between real coefficient and estimate coefficient over real coefficient less than 10%. The Error rate of Coefficient is denoted by

$$Error\ rate\ of\ Coefficient = \frac{Absolute\ value\ of\ (read\ coefficient - estimate\ coefficient)}{Absolute\ value\ of\ real\ coefficient} \quad (14)$$

where the molecular is the absolute value of the difference between estimates coefficients and approximately true coefficients, the denominator is the absolute value of approximately true coefficients. Note that the threshold of sensitivity degree is able to be a different value according to the requirement of robustness.

3.3. Analyze the stability of Fit Quality of Regression model

The minimum sampling time of residual standard error and R^2 is used to represent the stability of fit goodness of regression models. Likewise, the threshold of sensitivity degree is configured as 0.1. It can be adjusted according to the requirements of real situation. The smaller threshold needs more sampling time while larger one needs less. The corresponding Error rate of RSE and R2 are denoted by

$$Error\ rate\ of\ RSE = \frac{Absolute\ value\ of\ (real\ RSE - RSE\ of\ fitted\ model)}{Absolute\ value\ of\ real\ RSE} \quad (15)$$

$$Error\ rate\ of\ R^2 = \frac{Absolute\ value\ of\ (real\ R^2 - R^2\ of\ fitted\ model)}{Absolute\ value\ of\ real\ R^2} \quad (16)$$

The similar method with the analysis of stability of estimate coefficient is used to get the minimum stable sampling time of statistical metrics.

4. Result

4.1. Experimental environment

The basic experimental setting is as follows:

- Bare metal server with an Intel Core^{TM} i5-4670 CPU 3.40GHz 4 cores, 16GB Kingston HyperX Black DDR3 1600MHz RAM and 250GB 7200RPM hard drive.
- Ubuntu server 16.04.3 LTS, kernel 4.4.0-62-generic has been used to the top of the physical hardware. Automatic update is abandoned.
- Hadoop single node mode is installed on the top of Ubuntu server. The node runs default configuration of Hadoop version 2.7.3 and MapReduce v2, except the block size is set to 512MB.

4.2. Remove Non-randomness

Figure 1 presents the autocorrelation plot and partial autocorrelation plot of each resource usage parameter of Terasort application. The purpose is to check and remove non-randomness. If non-randomness existed, the lag values with the largest partial autocorrelation are chosen to remove non-randomness.

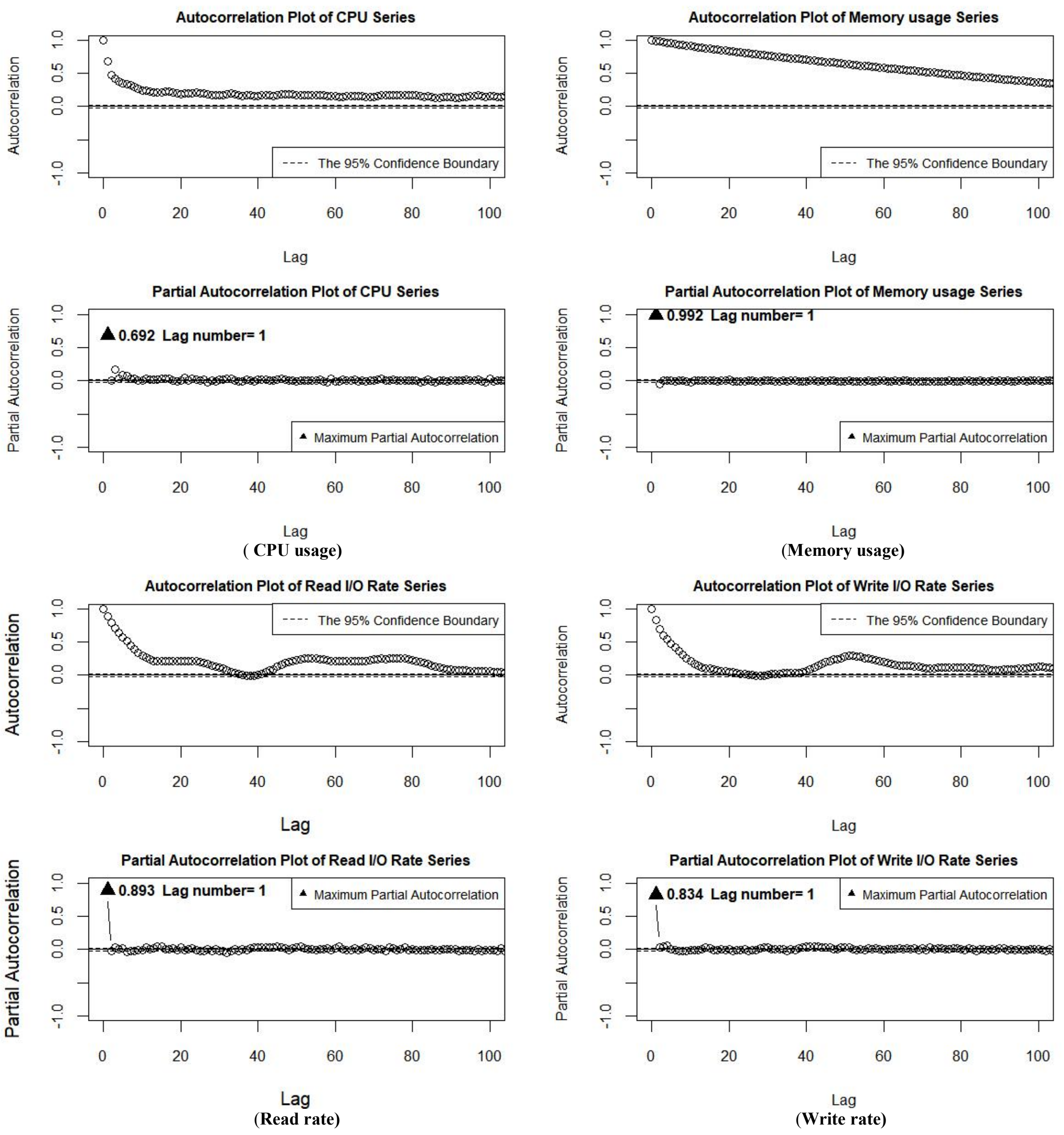

Figure 1. ACF and PACF plot of TeraSort application

In Figure 1, the autocorrelation plots of all resource usage parameters reveal non-randomness because of the corresponding autocorrelations, denoted by the circle, violate the dashed lines (95% confidence boundary) and are statistically significant for lags up to 100. The filled triangle point-up in partial autocorrelation plots marks the largest partial autocorrelation of usage parameters as well as the corresponding lag number.

The largest partial autocorrelation of CPU usage is 0.692 and the corresponding lag number is 1. It shows the highly positive relevance between the CPU usage and its lag 1 values. This result indicates CPU usage is non-random and the main propagating factor is lag 1 values. The same can be observed with memory usage, read rate and write rate. The largest partial autocorrelations and the corresponding lag numbers are listed in Table 1.

According to the lag number in Table 1, the corresponding lag series is added to the predicted equation to remove the non-randomness of associated time series data.

Table 1. The largest partial autocorrelations and the corresponding lag numbers of applications

Application name	**CPU usage**		**Memory usage**		**Read rate**		**Write rate**	
	PACF	**Lag number**	**PACF**	**Lag number**	**PACF**	**Lag number**	**PACF**	**Lag number**
WordCount	0.624	1	0.996	1	0.605	1	0.391	5
WordMean	0.565	3	0.996	1	0.739	1	0.336	5
WordMedian	0.448	6	0.996	1	0.731	1	0.380	5
TeraSort	0.692	1	0.992	1	0.893	1	0.834	1
Grep	0.742	1	0.996	1	0.802	1	0.261	5
TeraGen	0.387	1	0.990	1	0.478	2	0.711	1
Pi	0.983	1	0.932	1	0.732	1	0.307	5

4.3. Feasible Analysis of Linear Regression Model

4.3.1. Correlation Matrix

The Pearson product-moment correlation coefficient was used to quantify the strength of a linear relationship between resource usage parameters. We explored and analysed the significant linear strength between these correlation coefficients of each MapReduce application and chose the Terasort application to describe substantial correlation characteristics. Table 2 shows the correlation matrix of collected data of the Terasort application.

Table 2. Correlation matrix of Terasort application

	CPU	**MEM**	**RIO**	**WIO**	**Prv_cpu**	**Prv_mem**	**Prv_rio**
MEM	-0.089						
RIO	0.046	-0.269					
WIO	0.048	-0.073	-0.495				
Prv_cpu	0.692	-0.086	0.026	0.076			
Prv_mem	-0.092	0.995	-0.272	-0.072	-0.089		
Prv_rio	0.039	-0.262	0.893	-0.396	0.046	0.267	
Prv_wio	0.051	-0.074	-0.413	0.834	0.048	-0.072	-0.495

Table 2 shows the correlation coefficients between resource usage parameters including the corresponding lag series of the Terasort application. All the correlations between usage parameters and their corresponding lag series show the strong positive linear relevance. The range is between 0.692 and 0.995. Another significant correlation comes up between read rate and write rate and it shows the moderate negative linear relevance (-0.495). Meanwhile, the correlations between read rate and lagged write rate and between write rate and lagged read rate both show the similar results with the correlation between read rate and write rate. The results are -0.413 and -0.396. The weak negative relevance is present between memory usage and read rate. The remaining correlations all exhibit extremely weak relationships or even no relationship.

4.3.2. Performance Improvement

The improvement of the base error rate is used to decide the feasibility of linear regression model. The higher this percentage, the bigger the feasibility. Table 3 show the improved percentage of the base error rate of each MapReduce application.

In Table 3, the most significant improvement on the base error rate comes up on the linear models of memory usage as response and are all larger than 87%. The linear models of read rate as the response also have good performance on improving the base error rate. For the linear models of CPU usage as the response, the Teragen application shows a weak decline (-10%) on base rate error and others show positive improvements. Pi application (-3%) using the model of write rate as the response also exhibits a weak decline (-3%). The remaining linear models using write rate as the response improve the corresponding base error rates. The results prove these linear models are the possible adequate ones.

Table 3. Improvement of base error rate of each application

Name of application	**Improve_error_rate**			
	CPU usage as response	**Memory usage as response**	**Read rate as response**	**Write rate as response**
WordCount	52%	99.5%	25%	23%
WordMean	10%	100%	99%	98%
WordMedian	25%	99%	34%	13%
TeraSort	80%	98%	91%	86%
Grep	50%	100%	98%	99%
TeraGen	-10%	99.6%	42%	52%
Pi	96%	87%	41%	-3%

4.4. Multiple Linear Regression Model

Table 4 shows the multiple linear regression models and the corresponding residual standard error (**RSE**) and R square (**R^2**) of each MapReduce application.

Table 4. List of regression models

Name of application	**Regression Model**	**RSE**	**R^2**
Wordcount	$\widehat{CPU} \sim 22.37 + 0.4 \times RIO + 0.6 \times Prv_{cpu}$	13.41	39.5%
	$\widehat{MEM} \sim 0.37 + 0.996 \times Prv_{mem}$	0.36	99.9%
	$\widehat{RIO} \sim 11 - 6.12 \times WIO + 0.6 \times Prv_{rio}$	3.70	38.9%
	$\widehat{WIO} \sim 0.04 + 0.41 \times Prv_{wio}$	0.12	17.1%
Wordmean	$\widehat{CPU} \sim 12.34 + 0.6 \times RIO + 0.57 \times Prv_{cpu}$	16.25	36.1%
	$\widehat{MEM} \sim 0.37 + 0.996 \times Prv_{mem}$	0.36	99.9%
	$\widehat{RIO} \sim 6.28 + 0.02 \times CPU - 3.55 \times WIO + 0.73 \times Prv_{rio}$	2.73	56.7%
	$\widehat{WIO} \sim 0.04 + 0.36 \times Prv_{wio}$	0.13	12.7%
Wordmedian	$\widehat{CPU} \sim 26.09 + 0.51 \times Prv_{cpu}$	21.37	25.6%
	$\widehat{MEM} \sim 0.36 + 0.996 \times Prv_{mem}$	0.40	99.9%
	$\widehat{RIO} \sim 6.63 - 4.65 \times WIO + 0.05 \times Prv_{mem} + 0.62 \times Prv_{rio}$	2.59	57.8%
	$\widehat{WIO} \sim 0.04 + 0.39 \times Prv_{wio}$	0.12	15%
Terasort	$\widehat{CPU} \sim 2.18 + 0.69 \times Prv_{cpu}$	9.41	47.8%
	$\widehat{MEM} \sim 0.77 + 0.99 \times Prv_{mem}$	0.89	99%
	$\widehat{RIO} \sim 1.25 - 0.24 \times WIO + 0.96 \times Prv_{rio} + 0.27 \times Prv_{wio} - 0.01 \times WIO{:}Prv_{rio}$	3.19	90.4%
	$\widehat{WIO} \sim 3.01 + 1.11 \times Prv_{rio} - 1.16 \times RIO + 0.93 \times Prv_{wio} - 0.01 \times RIO{:}Prv_{wio}$	6.39	83.7%
Grep	$\widehat{CPU} \sim 2.93 + 0.74 \times Prv_{cpu}$	11.86	55%
	$\widehat{MEM} \sim 0.22 + 0.01 \times Prv_{cpu} + 0.997 \times Prv_{mem}$	0.40	99.9%
	$\widehat{RIO} \sim 5.51 + 0.80 \times Prv_{rio}$	2.64	64.4%
	$\widehat{WIO} \sim 0.04 + 0.27 \times Prv_{wio}$	0.11	7%
Teragen	$\widehat{CPU} \sim 25.45 - 0.18 \times MEM + 0.32 \times Prv_{cpu}$	8.53	19.2%
	$\widehat{MEM} \sim 0.83 + 0.003 \times CPU + 0.99 \times Prv_{mem}$	0.12	100%
	$\widehat{RIO} \sim 0.003 + 0.58 \times Prv_{rio}$	0.05	33.2%
	$\widehat{WIO} \sim 13.05 + 0.15 \times Prv_{cpu} + 0.72 \times Prv_{wio}$	6.56	53.5%
Pi	$\widehat{CPU} \sim 1.25 + 0.98 \times Prv_{cpu}$	7	96.8%
	$\widehat{MEM} \sim 6.85 + 0.07 \times CPU + 0.31 \times Prv_{mem}$	1.14	92.8%
	$\widehat{RIO} \sim 0.004 + 0.72 \times Prv_{rio}$	0.13	54.2%
	$\widehat{WIO} \sim 0.10 + 0.36 \times Prv_{wio}$	0.28	13%

In Table 4, the estimated coefficient shows the strength of linear dependency and its sign represents the dependent direction. Except for the dependency between response and the previous usage parameter itself, others dependency exposes the resource bottleneck of the corresponding application in Hadoop MapReduce environment. Meanwhile, according to the estimated coefficient, the optimized suggestion could be given for improving associated usage parameters. For example, read rate, denoted by RIO, in the model $\widehat{CPU} \sim 22.37 + 0.4 \times RIO + 0.6 \times Prv_{cpu}$ of Wordcount application has an estimated coefficient 0.4. It means that the average CPU usage might increase 4% when the average read rate increase 10MB/S, as well as the corresponding previous CPU usage keeps the fixed value. But in reality, the average of the previous CPU usage increases with approximately 4%. Thus, the real increase of the means of CPU usage will be 6.4% according to the equation $0.4 \times 10 + 0.6 \times 4 = 6.4$ and the corresponding resource bottleneck is the low read rate. From the cloud operators side, the effective improvement of CPU usage can be expected when the throughput of the disk drive is able to be increased. In the application developers' perspective, the read rate can be improved by optimizing programming of the corresponding application.

The regression models of applications with similar features showed significantly similar form and dependent strength, such as Wordcount and Wordmean, while others exhibited differently. The RSE and R^2 are used to describe the fit quality of models. The smaller RSE and the larger R^2 are the desirable statistical metrics.

4.5. Validation of autocorrelation of error terms

For the linear regression model, one of the most important assumptions is the error terms, $\epsilon_1, \epsilon_2, \cdots, \epsilon_n$, are uncorrelated. If the error term is uncorrelated, it proves that there exists strong randomness in residuals of the model and provides the evidence for the unbiased estimate for the true standard error. The autocorrelation plot is used to check this assumption. Figure 2 shows the autocorrelation plot of residuals of regression models of TeraSort application.

In Figure 2, the horizontal axis represents lag time and the vertical axis indicates the autocorrelation between residual at time t and residual at other lag time. At lag 0, autocorrelation is always equal to 1 and represents time series itself. Most of the autocorrelation at other lag time fall into the 95% confidence interval, only few of them violate the dashed line. Such a shape of ACF plot proves that residuals are uncorrelated and respects to the independent assumption of the linear regression model residuals.

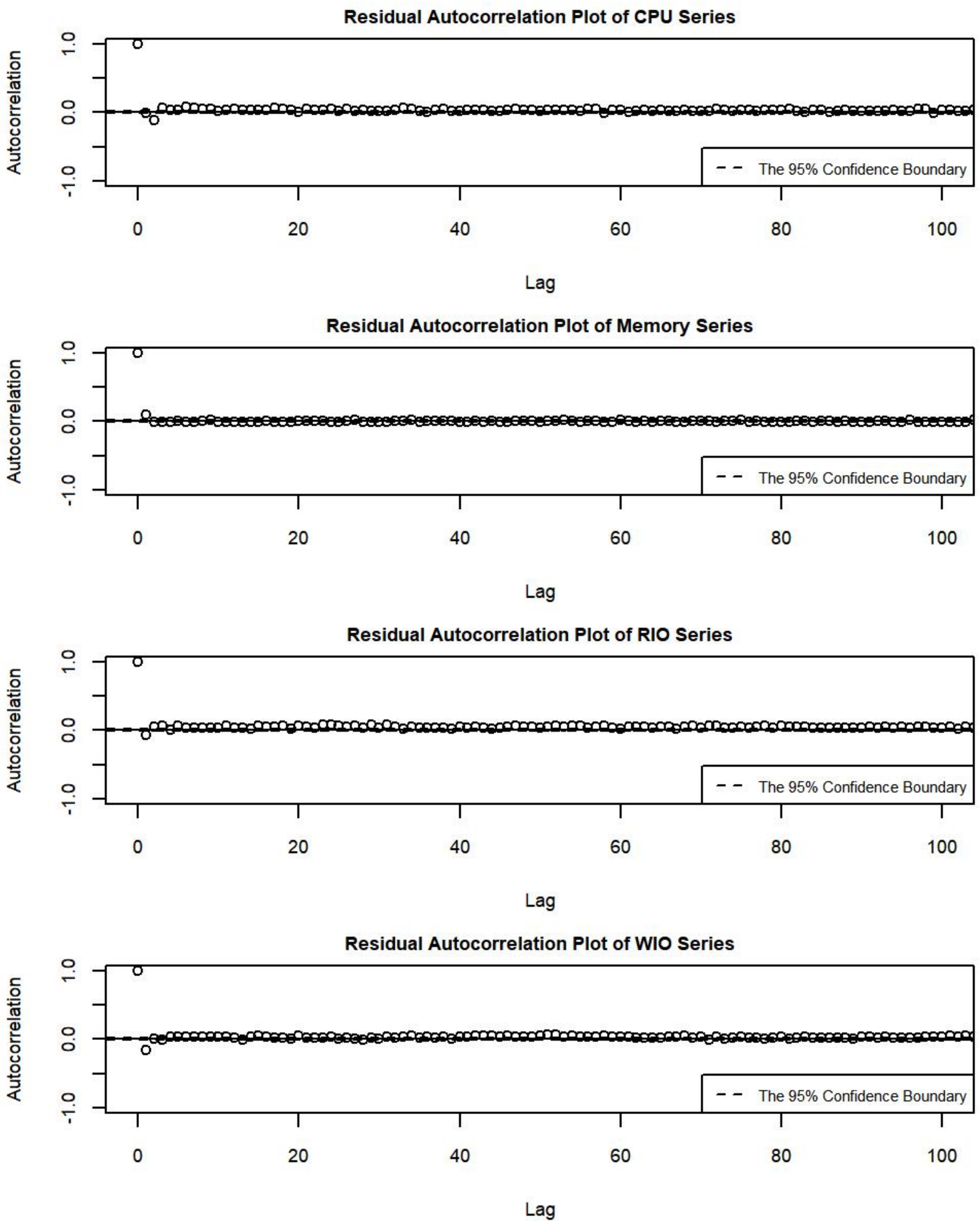

Figure 2. ACF plot of Residuals of Regression Model of Terasort

4.6. Analysis of the minimum sampling time of stable modeling

4.6.1 Analysis of stability of estimate coefficient

Figure 3 shows estimate coefficients distribution of model on read rate as the response of Terasort application. The filled triangle point-up indicates the position of the minimum stable sampling time for the corresponding estimate coefficients as well as the number above it shows the exact position value. The dashed line represents the average estimated coefficient of the regression model.

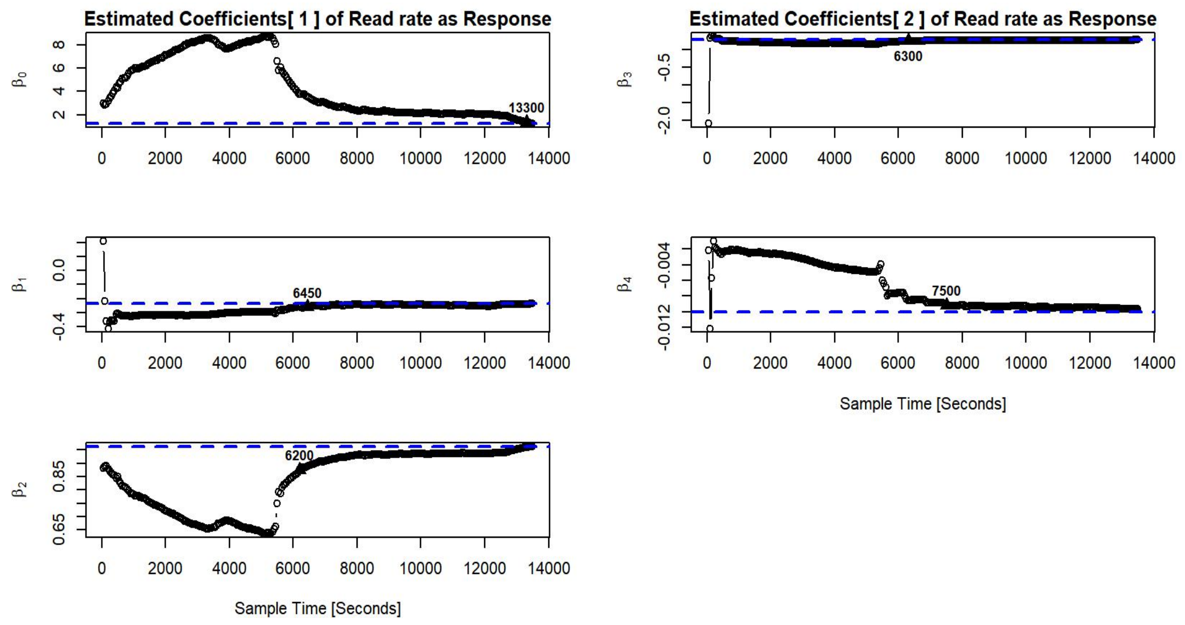

Figure 3. Estimate coefficients distribution of model on read rate as response of Terasort application

Note that each estimated coefficient converges to dashed line gradually and shows the different minimum sampling times as the training data size increases. Different estimated coefficients perform the various minimum sampling time. For stable modeling, the largest sampling time of these coefficients is used to be the minimum sampling time of Terasort application. The minimum sampling time here is 13300 and it belongs to intercept coefficient $\boldsymbol{\beta_0}$. It is found that the bigger difference between estimated coefficients and the average estimated coefficients is exposed as the training data size approximately less than 6000. The result indicates that the adequate training data is needed for training the stable model. Therefore, the minimum sampling time is essential for the model used to reveal the internal relationship between resource usage parameters.

4.6.2. Analysis of stability of Fit Quality of Regression model

Figure 4 shows the statistical metrics distribution of regression model on read rate as the response of Terasort application. The filled triangle point-up indicates the minimum stable sampling time for statistical metrics. The top-half of figure 4 shows the residual standard error (***RSE***) distribution as training data size increase. The remaining half is for the distribution of $\boldsymbol{R^2}$.

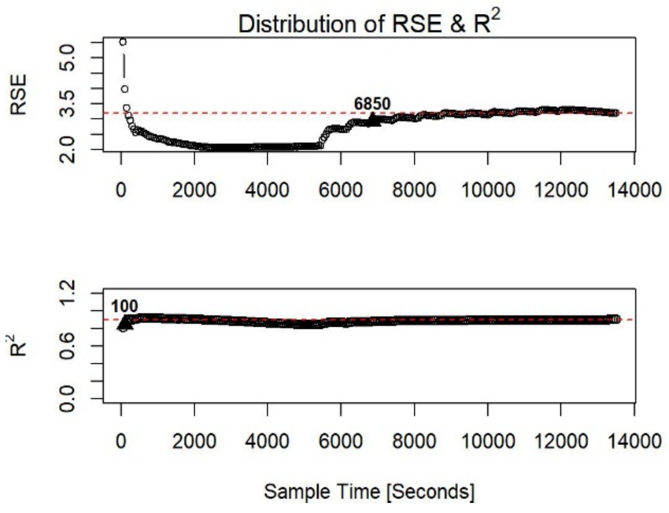

Figure 4. Statistical Metrics distribution of model on RIO as response of Terasort application

The minimum sampling time of residual standard error converges to stability slowly than R^2 for stable modeling. The sampling time is 6850 and 100, respectively. Comparing to the minimum sampling time of estimated coefficients, the sampling times of statistical metrics are more less. Therefore, 13300 is used to be the minimum sampling time to ensure stable modeling.

5. Analysis and Discussion

5.1. The distribution of Estimate Coefficients and Statistical Metrics

Figure 5 presents the strength of estimated coefficients of regression models of each application using bar plot. The height of a bar represents strength of each estimated coefficient as well as the pattern shows the corresponding usage parameter of regression model. The top-left panel of figure 5 displays the estimated coefficients distribution of the model with CPU usage as the response. The top-right panel, bottom-left panel, bottom-right refer to memory usage, read rate and write rate as the response, respectively.

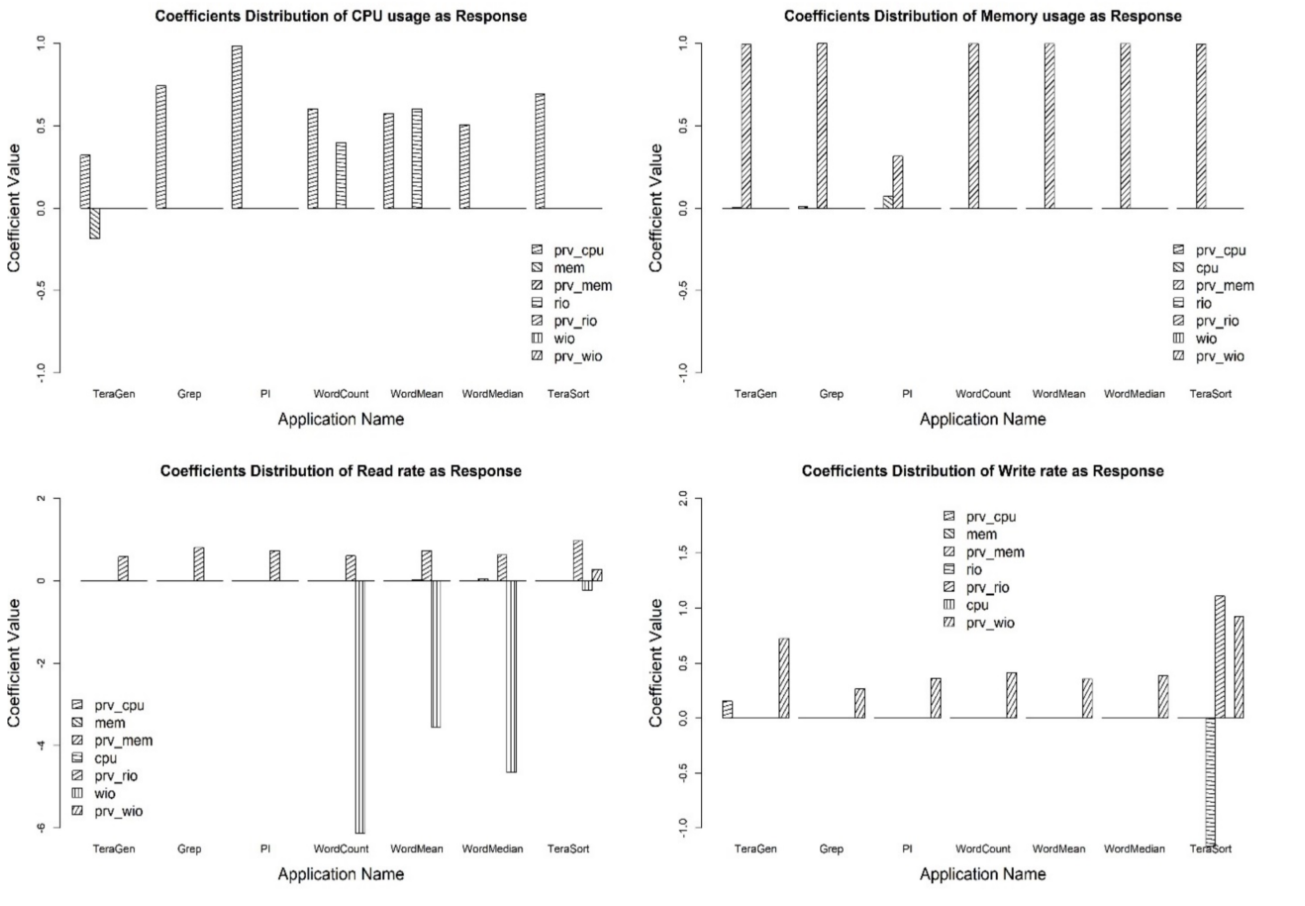

Figure 5. Coefficient Distribution of models of all applications

In Figure 5, the positive dependency of different strength between each resource usage parameter and the corresponding previous usage parameter is exhibited for all MapReduce applications. It indicates that all current resource usage parameters are positively dependent on the previous values to some extent degree. Except for these common dependencies, there exist some special dependencies for different applications.

On the top-left panel of Figure 5, CPU usage of Pi application shows the strongest positive dependency to lagged CPU usage, the Teragen application had the weakest positive dependency, and others exhibit the moderate positive dependency. Except for the dependency between CPU usage and lagged CPU usage, the Wordcount application exhibits moderate positive dependency between CPU usage and read rate, as well as Wordmean application. Meanwhile, the Teragen application shows a weak negative relationship between CPU usage and memory usage.

The top-right panel of figure 5 exhibits the dependent characteristics between memory usage and other usage parameters. Except for Pi application, others showed the extremely highly positive dependency of memory usage to lagged memory usage. Memory usage of Pi application has a weakly positive dependency to CPU usage except for a moderate dependency to lagged memory usage. Memory usage of Grep application exhibits an extremely weak dependency to lagged CPU usage.

The dependent relationships between read rate and other usage parameters are displayed in the bottom-left panel of Figure 5. The dependency between read rate and previous read rate indicates that read rate most likely depends on the lagged read rate for each application. For Wordcount, Wordmean and Wordmedian applications, read rate is negative dependent to write rate. In other words, read rate is going to significantly decrease as write rate increases. Read rate of Terasort application exhibited the moderate negative relationship with write rate and the moderate positive relationship with lagged write rate. It implies that read rate is sensitive to the variation of write rate.

The bottom-right panel of figure 5 exhibits the relationship between write rate and other usage parameters for each application. Terasort and Teragen applications have the highest estimated coefficients on lagged write rate. It is most likely due to their frequent write operations. The remaining applications just exhibit the weak relationship between write rate and lagged write rate. Write rate of Pi application also shows a weak positive relationship with lagged CPU usage. Write rate of Terasort application exhibits a strong negative relationship with read rate and positive relationship with lagged read rate. It means that write rate has a relationship with the variation between read rate and lagged read rate. Therefore, each MapReduce application has the various relationship among resource usage parameters.

Figure 6 shows the fit quality of regression models. It is following:

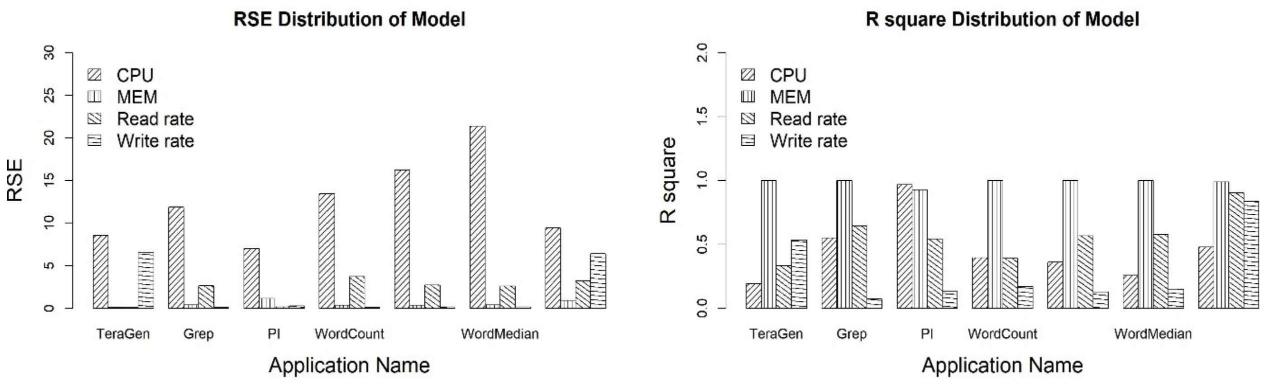

Figure 6. RSE and R^2 of regression models of MapReduce applications

The left panel and the right panel of figure 6 show the residual standard error (RSE) distribution and R^2 distribution of each application. The good fit quality corresponds to a taller R^2 bar and a shorter RSE bar. The R^2 almost 1 and small RSE show the best fit quality of the regression models on memory usage as the response. The overall higher RSE and lower R^2 of regression models on CPU as the response show the worse quality of fitting goodness. The regression models on read rate as the response also show a moderate fitting quality. For the regression models on write rate as the response, Terasort application exhibits the best quality and Teragen application as well. Others show the worse fitting quality. The results show that the regression models on intensive usage parameters as response exhibit the good fitting quality.

5.2. Comparison of Minimum Sampling time for Estimated Coefficients and Statistical Metrics

Figure 7 was used to show the distribution of the minimum sampling time of estimate coefficients for stable modeling for each application.

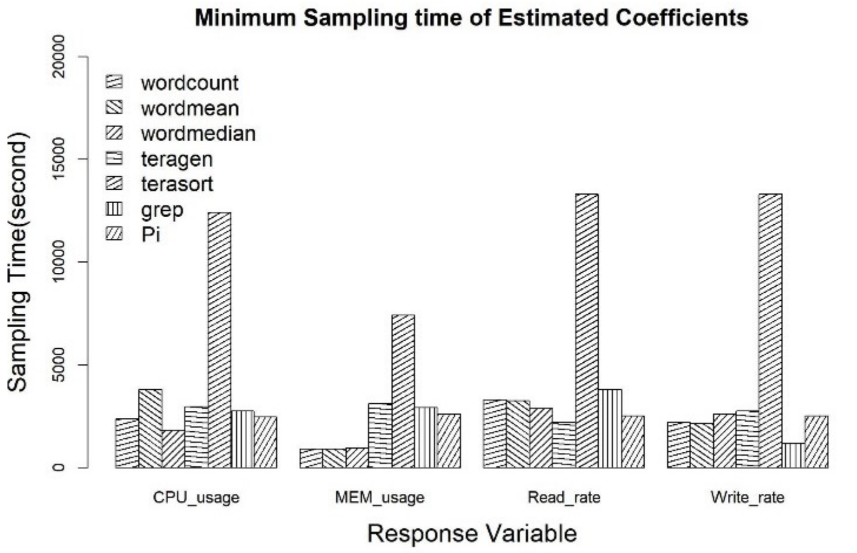

Figure 7. Minimum sampling time of MapReduce applications on various response variable

Figure 7 shows that Terasort application needs the largest sampling time to reach stable state while Pi application needs the smallest sampling time. The remaining applications need the similar minimum sampling time. Overall, the stable regression models on memory usage as response show the least need for sampling time. The results show that various applications have different minimum sampling time to get stable. The application which performs more read/write operations shows larger sampling time need.

Figure 8 presents the minimum sampling time distribution of statistic metrics which ensures the stable modeling. Overall, the minimum sampling time of statistic metrics is smaller than sampling time of estimated coefficients. For different applications, a time-consuming application like Terasort needs the largest sampling time to tend to be stable. The Pi application shows the smallest minimum sampling time to reach stability.

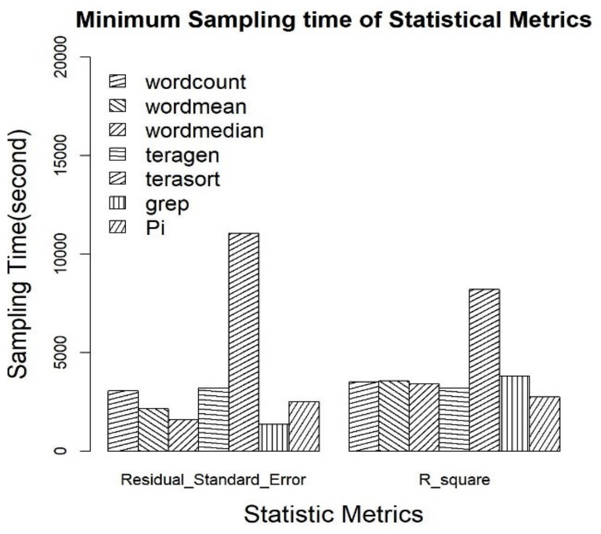

Figure 8. Minimum sample time of statistical metrics of MapReduce applications

The results show that the minimum sampling time of stable modeling in MapReduce environment is related to the features and the complexity degree of the algorithm of applications. Applications with similar features and complexity degree of algorithm always have similar minimum sampling time. The application of higher complexity degree and consumption of resources will need more sampling time to be stable. Therefore, the fixed sampling time is not always reliable.

6. Conclusion and Future Works

In this paper, we explored and dig the dependent characteristics of resource usage parameters, including CPU usage, memory usage, read rate and write rate. Based on the dig characteristics, we model the relationship of usage parameters using multiple linear regression methods for seven MapReduce applications. The results show that the regression model built by using such analytical approach is beneficial to cloud operators to discover the bottleneck resource of the cloud computing platform and to give reasonably optimized suggestion for MapReduce applications. Furthermore, in order to make sure of obtaining the stable relationship, we also investigated the minimum sampling time for each application. We exhibited the influence of sampling time for stable modeling and presented an effective approach to make the corresponding minimum sampling time. This approach effectively provided pieces of evidence against the empirical fixed sampling time for modeling MapReduce applications. In future, we will study the influence coming from the block size and scale of workload and explore the best configuration of related parameters for stable modeling.

References

Apache.org. (2017). Apache hadoop. Retrieved 10 December 2017, from http://hadoop.apache.org/

Bakratsas, M., Basaras, P., Katsaros, D., & Tassiulas, L. (2017). Hadoop MapReduce Performance on SSDs for Analyzing Social Networks. *Big Data Research*.

Bautista Villalpando, L., April, A., & Abran, A. (2014). Performance analysis model for big data applications in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, *3*(1), 19–38.

Bhattacharya, P. K., & Burman, P. (2016). 13 - Time Series. In P. K. Bhattacharya & P. Burman (Eds.), *Theory and Methods of Statistics* (pp. 431–489). Academic Press.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time Series Analysis: Forecasting & Control. Book.*

Everitt, B. S., & Skrondal, A. (2010). *The Cambridge Dictionary of Statistics*. *Journal of Chemical Information and Modeling* (Vol. 53).

Gareth James, Daniela Witten, T. H. and R. T. (2013). *An Introduction to Statistical Learning. Springer New York Heidelberg, page 72.*

Geneva. (2012). ISO/IEC (2012) ISO/IEC JTC 1 SC38:Cloud Computing Overview and Vocabulary. In *International Organization for Standardization*. Switzerland.

Glushkova, D., Jovanovic, P., & Abelló, A. (2017). Mapreduce performance model for Hadoop 2.x. *Information Systems*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. *Bayesian Forecasting and Dynamic Models* (Vol. 1).

Issa, J. A. (2015). Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount. *IEEE Access*, *3*, 2784–2793.

Levy, G. (2016). Chapter 3 - Generation of Random Variates. In G. Levy (Ed.), *Computational Finance Using C and C#* (pp. 35–56). Academic Press.

Linton, O. (2017). Chapter 17 - The Least Squares Procedure. In O. Linton (Ed.), *Probability, Statistics and Econometrics* (pp. 251–261). Academic Press.

Molugaram, K., & Rao, G. S. (2017a). Chapter 10 - Test of Significance—Small Samples. In K. Molugaram & G. S. Rao (Eds.), *Statistical Techniques for Transportation Engineering* (pp. 415–450). Butterworth-Heinemann.

Molugaram, K., & Rao, G. S. (2017b). Chapter 6 - Correlation and Regression. In K. Molugaram & G. S. Rao (Eds.), *Statistical Techniques for Transportation Engineering* (pp. 293–329). Butterworth-Heinemann.

Nghiem, P. P., & Figueira, S. M. (2016). Towards efficient resource provisioning in MapReduce. *Journal of Parallel and Distributed Computing*, *95*(Supplement C), 29–41.

O'Brien, R. M. (2007). A caution regarding rules of thumb for variance inflation factors. *Quality and Quantity*, *41*(5), 673–690.

Rizvandi, N. B., Nabavi, A., & Hessabi, S. (2005). An accurate FIR approximation of ideal fractional delay filter with complex coefficients in Hilbert space. *Journal of Circuits, Systems and Computers*, *14*(3).

Rodgers, D. P. (n.d.). *'Improvements in multiprocessor system design'*. New York: ACM SIGARCH Computer Architecture News. http://doi.org/doi:10.1145/327070.327215.

Ross, S. M. (2017a). Chapter 12 - Linear Regression. In S. M. Ross (Ed.), *Introductory Statistics (Fourth edition)* (Fourth edition, pp. 519–584). Oxford: Academic Press.

Ross, S. M. (2017b). Chapter 3 - Using Statistics to Summarize Data Sets. In S. M. Ross (Ed.), *Introductory Statistics (Fourth edition)* (Fourth edition, pp. 65–138). Oxford: Academic Press.

Shams, M. Bin, Haji, S., Salman, A., Abdali, H., & Alsaffar, A. (2016). Time series analysis of Bahrain's first hybrid renewable energy system. *Energy*, *103*, 1–15.

Shi, Y., Zhang, K., Cui, L., Liu, L., Zheng, Y., Zhang, S., & Yu, H. (2016). MapReduce short jobs optimization based on resource reuse. *Microprocessors and Microsystems*, *47*(Part A), 178–187.

Theodoridis, S. (2015). Chapter 4 - Mean-Square Error Linear Estimation. In S. Theodoridis (Ed.), *Machine Learning* (pp. 105–160). Oxford: Academic Press.

Usset, J., Staicu, A.-M., & Maity, A. (2016). Interaction models for functional regression. *Computational Statistics & Data Analysis*, *94*, 317–329.

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., … Baldeschwieler, E. (2013). Apache Hadoop YARN: Yet Another Resource Negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing* (p. 5:1--5:16). New York, NY, USA: ACM.

Warner, R. A. (2016). Chapter 2 - Using Z Scores for the Display and Analysis of Data. In R. A. Warner (Ed.), *Optimizing the Display and Interpretation of Data* (pp. 7–51). Boston: Elsevier.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). Chapter 5 - Credibility: Evaluating what's been learned. In I. H. Witten, E. Frank, M. A. Hall, & C. J. Pal (Eds.), *Data Mining (Fourth Edition)* (Fourth Edition, pp. 161–203). Morgan Kaufmann.

Yu, H., Jiang, S., & Land, K. C. (2015). Multicollinearity in hierarchical linear models. *Social Science Research*, *53*, 118–136.

Yangyuan LI (September 23, 1980) received his BSc in Computer science (2002) from Tai Yuan University of Technology, MSc in Network information control (2010) from Chang'an University. Now he is PhD in Department of Network system and services, Faculty of Sciences, Budapest University of Technology and Economics, Hungary. His current research interests include different aspects of Artificial Intelligence applied in big data analysis, such as statistical learning, Neural network, cloud computing.