# Performance Evaluation of Hadoop-based Large-scale Network Traffic Analysis Cluster

Ran Tao, Yuanyuan Qiao and Wenli Zhou

*Beijing Key Laboratory of Network System Architecture and Convergence Beijing University of Posts and Telecommunications, Beijing 100876, China*

**Abstract.** As Hadoop has gained popularity in big data era, it is widely used in various fields. The self-design and self-developed large-scale network traffic analysis cluster works well based on Hadoop, with off-line applications running on it to analyze the massive network traffic data. On purpose of scientifically and reasonably evaluating the performance of analysis cluster, we propose a performance evaluation system. Firstly, we set the execution times of three benchmark applications as the benchmark of the performance, and pick 40 metrics of customized statistical resource data. Then we identify the relationship between the resource data and the execution times by a statistic modeling analysis approach, which is composed of principal component analysis and multiple linear regression. After training models by historical data, we can predict the execution times by current resource data. Finally, we evaluate the performance of analysis cluster by the validated predicting of execution times. Experimental results show that the predicted execution times by trained models are within acceptable error range, and the evaluation results of performance are accurate and reliable.

## 1 Introduction

With the rapid development of cloud computing, Hadoop[1] as an advanced big data processing tool, has become the first choice for many researchers and companies. In order to analyze the massive traffic data efficiently, we developed a Hadoop-based Large-scale Network Traffic Analysis Cluster (HBLSNTAC). It consists of one master (running NameNode, ResourceManager), one backup master (also work as client access server, running SecondaryNameNode) and nine slaves (running DataNode, NodeManager, ApplicationMaster). HBLSNTAC stores massive network traffic data collected from operators of several cities in China in HDFS. Moreover, users run various off-line statistical analysis applications using the massive data on the cluster. These applications are aimed at analyzing the basic statistical characteristic of network traffic, such as the time distribution or geographical distribution of network behavior of mobile phone users.

The performance status would greatly affect the work efficiency of HBLSNTAC. A poorer performance means a lower work efficiency, which would slow down the whole analysis work schedule. If we can reliably evaluate performance of HBLSNTAC, the cluster manager could get an intuitive overview of performance status and then make timely adjustment to keep the cluster steady and in high efficiency. However, the performance of HBLSNTAC is affected by various factors, such as hardware device, software versions, Hadoop configuration parameters which are the main factor of performance, and so on. The total amount of Hadoop configuration parameters is excessive, let alone the complex interactive relationships among these parameters. So it is hard to accurately evaluate the performance of HBLSNTAC.

Since several aspects of factors would influence the performance of HBLSNTAC, it is very difficult for normal user to collect all the information of influenced factors. So normal user barely knows the actual performance of HBLSNTAC. Meanwhile, the execution time of the off-line analysis application is directly depended on the performance of HBLSNTAC. A poorer performance would cause a longer execution time. If normal user launches an application at improper timing, for example, when the performance is poor, it would end up lengthening the execution time of his own application, and further worsening the work efficiency of HBLSNTAC. Obviously, that is a dilemma we avoid to get in.

Thus, there is a pressing need for a performance evaluation system to evaluate the performance of HBLSNTAC. This paper aims to design and implement a performance evaluation system to HBLSNTAC. The main contributions of this paper are:

- We propose a performance evaluation system of HBLSNTAC, and the evaluation results are verified to be reliable. Thus we could have a quantitative understanding of performance and then guide user behaviors.

- Since the Hadoop configuration parameters are excessive and complex, we focus on the resource data, which is the essence of configuration parameters. And besides average value, we add three statistical dimensions of resource data, i.e., maximum, minimum and standard deviation.
- We apply principle component analysis and multiple linear regression in modeling phase. After training by historical data, the prediction of this model is relatively accurate with an average error less than 10%.

The remainder of this paper is organized as follows. Firstly, we introduce the related work in section 2. Then we propose the performance evaluation system and detailed introduce the design of the system in section 3. The experimental results and corresponding analysis are presented in section 4. Finally, we draw the conclusion and give the outlook of future work.

## 2 Related Work

Hadoop is a popular open source project. It has shown its significance in big data and cloud computing areas. Since increasing users and companies have applied Hadoop, the study and research of its performance has become more important and globalized.

Performance Optimization is a research focus for Hadoop cluster, because it could improve the work efficiency of cluster. Microsoft IT SES Enterprise Data Architect Team proposes various optimization techniques to maximize performance of Hadoop MapReduce jobs in [2]. Paper [3] proposes a statistic analysis method to identify the relationship among workload characteristics, configuration parameters and Hadoop cluster performance. They both optimize Hadoop configuration parameters in different situations, therefore shorten the execution times of applications to improve the efficiency.

Performance evaluation is another research hot topic. HiBench[4] is a representative and comprehensive benchmark suite for Hadoop, which contains ten applications, classified into four categories. It quantitatively evaluates and characterize the Hadoop development. Paper [5] proposes a performance evaluation for Hadoop cluster on virtual environment, it builds a Hadoop performance model and then examine several influence factors of cluster performance.

In summary, the existed researches of performance of Hadoop cluster mostly focus on Hadoop configuration parameters. However, the configuration parameters are excessive and complex. So in this paper, we study on resource data, which is the essence of Hadoop configuration parameters. Especially, we build a performance model and identify the relationship between resource data and execution times of applications, then we evaluate performance of HBLSNTAC by the validated performance model.

## 3 The Design of Performance Evaluation System

In this section, we introduce the design of performance evaluation system. The framework of system is presented in Figure 1. Firstly we set execution times of three benchmark applications as the performance benchmark of HBLSNTAC, and select total of 40 metrics of resource data, which cover the main resources of HBLSNTAC, such as CPU, memory, disk and network. Then we model historical data of execution times of three benchmark applications and resource data through principal component analysis and multiple linear regression. After that, we predict the execution times by the trained models. Finally, we evaluate the performance of HBLSNTAC based on the predicting of execution times.

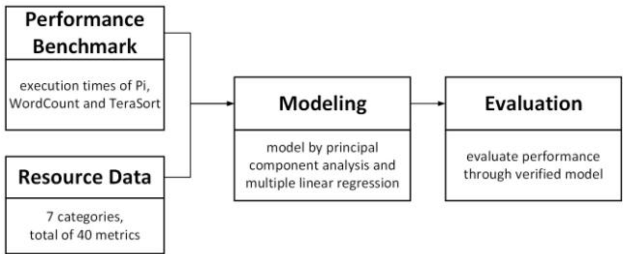
Performance Benchmark
execution times of Pi, WordCount and TeraSort
Resource Data
7 categories, total of 40 metrics
Modeling
model by principal component analysis and multiple linear regression
Evaluation
evaluate performance through verified model

**Figure 1.** Framework of Performance Evaluation System

### 3.1 Performance Benchmark

Hadoop applications can be classified into four categories, CPU intensive, memory intensive, disk I/O intensive and network I/O intensive[6]. Since applications could never finish successfully unless there is sufficient memory resource, we exclude memory intensive applications. Moreover, Network I/O is not a bottleneck in our experiment, because the scale of cluster is small and all nodes are connected to the same switch. As a result, we select a CPU intensive application, i.e., Pi; a disk I/O intensive application, i.e., TeraSort; and a both CPU and disk I/O intensive application, i.e., WordCount. Furthermore, we use the execution time of three selected Hadoop benchmark applications, Pi, WordCount and TeraSort, as the performance benchmark of HBLSNTAC.

More importantly, the main function of HBLSNTAC is running off-line applications to analyze the basic statistical characteristic of network traffic. Both WordCount and TeraSort applications have statistical approaches as counting and sorting separately. So it makes them much more suitable to evaluate the actual performance of HBLSNTAC than other applications.

### 3.2 Statistical Resource Data

Hadoop has more than 200 configuration parameters[7], and the interactive relationships among parameters are complex. Although many parameter researches have proposed various parameter tuning strategies, but with the difference of Hadoop cluster environments, not all the strategies are suitable for any environment. So it is unachievable to propose a universal full-cover parameter tuning strategy for all environments. Thus we can't evaluate performance of HBLSNTAC through configuration parameters without that universal parameter tuning strategy.

Resource data of server, such as CPU, memory, disk and network, is the underlying data of Hadoop cluster. Application on Hadoop cluster would not be launched if not enough resources are allocated to it. Furthermore, most of configuration parameters are specified by resources. For instance, most of configuration parameters of YARN are used for allocations of CPU and memory resources. In other words, resource data is the essence of configuration parameters. The parameters tuning essentially means resources reallocation.

Thus, to conduct a more simplified and essential research of performance of HBLSNTAC, we study on resource data of server instead of configuration parameters. Based on the study of Hadoop resources, combining with accumulated management experience of HBLSNTAC, we select 7 categories of resource data, total of 40 metrics. The list of metrics is shown in Table 1.

**Table 1.** List of Resource Data Metrics

| | | |
|---|---|---|
| CPU | CPU utilization | × (maximum, minimum, average, standard deviation) |
| Memory | memory utilization | |
| Disk | disk utilization | |
| | disk I/O speed | |
| Network | upstream speed | |
| | downstream speed | |
| Open files | number of open files | |
| Load | custom server load | |
| HDFS | upload speed | |
| | download speed | |

Since the resource of Disk, Network and HDFS, are subdivided into two smaller categories, we now have ten small categories. These ten small categories are the common resource data, which are studied frequently within Hadoop researches. Most of the researches only study the average value of resource data. It is generally known that average value does not reflect all of the statistical characteristics of the data. Therefore, for each small category, we add three statistical dimensions, maximum, minimum and standard deviation besides average. Then we have 40 metrics totally. Now, we not only study on the central tendency, but also study on the dispersion degree of resource data, which would improve the reliability of modeling analysis.

Here, the customized Load metric is used to describe the extent of a single server load exceeds the average server load of whole cluster, which would be expressed as $Load_i$. It is easy to understand that for metrics of CPU, Memory, Disk, Network and Open files, the larger the value is, the heavier the server load is. For server $i$, assume that $R_n$ is the value of one of metrics, $R_{avg}$ is the average value of the same metric of the cluster. Then we have $L_n$, i.e., the value of the customized load of a single metric, which could be computed by (1) , and $Load_i$, i.e., the value of the customized server load of server $i$, which could be computed by (2) .

$$L_n=\begin{cases} 0 & (R_n \le R_{avg}) \\ R_n\cdot(R_n-R_{avg}) & (R_n > R_{avg}) \end{cases} \tag{1}$$

$$Load_i=\sum_n L_n \tag{2}$$

## 3.3 Modeling Analysis

The purpose of modeling analysis is to figure out the relationship between performance benchmark, i.e., the execution times of three benchmark applications, and the recourse data. After training the model by historical data, we could predict the execution times by current resource data.

### *3.3.1 Principal Component Analysis*

Principal Component Analysis (PCA) uses dimensionality reduction technique to make a set of possibly correlated original indicators into relatively fewer comprehensive and linearly uncorrelated indicators by linear combination, and retain most of the information of original target[8].

If we have the original data matrix as:

$$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

where $m$ is the number of sites, $n$ is the number of variables in the index system, $x_{ij}(i=1,2,\cdots m; j=1,2,\cdots n)$ is the value of index variable $j$ of site $i$.

(1) Standardization of original data

Usually the variables have different units of measurement (i.e., pounds, feet, rate, etc), that would cause different weight for each variable in the analysis. In order to make each variable to receive equal weight, it is necessary to standardize the original data.

(2) Normalized correlation matrix of data

$$R=(r_{ij})_{m\times n} \tag{3}$$

where $r_{ij}$ is the correlation coefficient of index $i$ and index $j$. The eigenvalues of matrix $R$ are $\{\lambda_1,\lambda_1,\cdots\lambda_n\}$, and $\lambda_1 \ge \lambda_1 \ge \cdots \ge \lambda_n$.

(3) Calculation of variance contribution of each principal component

$$\alpha_i=\frac{\lambda_i}{\sum_{i=1}^{n}\lambda_i} \tag{4}$$

(4) Selection of principal components

Generally speaking, select number of principal components whose cumulative contribution reaches to rate of 85 ~ 100%, meaning these principal components retain 85 ~ 100% information of original data.

### *3.3.2 Multiple Linear Regression*

Multiple Linear Regression (MLR) attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. It is a statistical technique that uses several explanatory variables to predict the outcome of a response variable[9].

The general form of multiple linear regression is defined as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon \quad (5)$$

where $y$ is the dependent or response variable, $x_i \ (i = 1,2,\cdots,n)$ are the independent or explanatory variables, $\beta_i \ (i = 0,2,\cdots,n)$ are the parameter of model and $\varepsilon$ is the deviations of model.

## 3.4 Performance Evaluation

After verifying the trained model reliable, we could evaluate the performance of the HBLSNTAC by the prediction of the three execution times. We propose a score criterion by theses three execution times, then we select a typical practical application, whose execution time is a reflection of the practical performance of HBLSNTAC. By studying the relationship between the score and the execution time of practical application, we can find out several patterns between them. Finally we could evaluate the performance of HBLSNTAC by these patterns, thus to help us manager the HBLSNTAC in a better way.

## 4 Experiment

## 4.1 Experiment Setup

We setup a simple classical Hadoop cluster that consists of one master (running NameNode and ResourceManager) and three slaves (running DataNode, NodeManager and ApplicationMaster). All servers are connected to a 100Mbps Ethernet switch. Hardware and software configurations of the servers in this cluster are the same, as shown in Table 2.

**Table 2.** Configurations of Experimental Servers

| Item | Value |
|---|---|
| Machine | Lenovo M4360 |
| Processor | Pentium(R) E5800 @3.20GHz |
| Cores | 2 |
| Memory | 4G |
| Hard Disk | 500G |
| Network | 100Mbps |
| Operation System | CentOS 6.5 x64 |
| JVM | JDK 1.7.0_45 |
| Hadoop | 2.5.0 |

## 4.2 Modeling and Prediction

We run three benchmark applications for 25 times separately. The input of modeling is the resource data within 3 minutes before application get launched. And the output of modeling is the predictions of execution times of applications. It means that we would finally predict the execution time of an application by the resource data within 3 minutes before this application get launched.

### *4.2.1 Principal Component Analysis*

We get total of 40 metrics of resource data, meaning 40 dimensions for data analysis, that is a quiet large number of dimensions. Moreover, there would be correlations between these metrics. Therefore, we would reduce dimensions from total 40 dimensions to much more fewer uncorrelated dimensions by Principal Component Analysis. The main results of three benchmark applications by Principal Component Analysis using SPSS software[10] are shown in Table 3, Table 4, Table 5 and Figure 2.

For Pi application, we can find out from Table 3 and Figure 2a that the cumulative contribution rate of first 6 Principal Components (PCs) is 98.422%, meaning they retain 98.422% information of original 40 metrics. Here, we have reduced dimensions from 40 metrics to 6 PCs, which we would express as $X_{Pi} = (x_1, x_2, x_3, x_4, x_5, x_6)$. Similarly, for WordCount application, the first 5 PCs expressed as $X_{WC} = (x_1, x_2, x_3, x_4, x_5)$ retain 97.713% information of original 40 metrics. And for TeraSort application, the first 5 PCs expressed as $X_{TS} = (x_1, x_2, x_3, x_4, x_5)$ retain 100.00% information of original 40 metrics. The component matrix would not be shown here due to space limitation.

### *4.2.2 Multiple Linear Regression*

The purpose of MLR is to model the relationship between principal components and execution time of benchmark application, then make a prediction of execution time by a new set of principal components.

As for Pi application, the multiple linear regression equation of principal components as $X_{Pi} = (x_1, x_2, x_3, x_4, x_5, x_6)$ and execution time as $y_{Pi}$ is as follow:

$$y_{Pi} = -0.632 + 0.713x_1 + 0.413x_2 + 0.527x_3 + 0.258x_4 + 0.179x_5 + 0.302x_6 \quad (6)$$

As for WordCount application, the multiple linear regression equation of principal components as $X_{WC} = (x_1, x_2, x_3, x_4, x_5)$ and execution time as $y_{WC}$ is as follow:

$$y_{WC} = -0.491 + 0.726x_1 + 0.619x_2 + 0.492x_3 + 0.325x_4 + 0.173x_5 \quad (7)$$

**Table 3.** Total Variance Explained of Pi

| Component | Initial Eigenvalues | | | Extraction Sums of Squared Loadings | | |
|---|---|---|---|---|---|---|
| | Total | % of Variance | Cumulative % | Total | % of Variance | Cumulative % |
| 1 | 14.636 | 36.590 | 36.590 | 14.636 | 36.590 | 36.590 |
| 2 | 9.452 | 23.630 | 60.220 | 9.452 | 23.630 | 60.220 |
| 3 | 6.694 | 16.735 | 76.955 | 6.694 | 16.735 | 76.955 |
| 4 | 3.950 | 9.875 | 86.830 | 3.950 | 9.875 | 86.830 |
| 5 | 3.010 | 7.525 | 94.354 | 3.010 | 7.525 | 94.354 |
| 6 | 1.627 | 4.068 | 98.422 | 1.627 | 4.068 | 98.422 |
| 7 | .631 | 1.578 | 100.00 | | | |
| … | … | … | … | | | |

**Table 4.** Total Variance Explained of WordCount

| Component | Initial Eigenvalues | | | Extraction Sums of Squared Loadings | | |
|---|---|---|---|---|---|---|
| | Total | % of Variance | Cumulative % | Total | % of Variance | Cumulative % |
| 1 | 16.980 | 42.451 | 42.451 | 16.980 | 42.451 | 42.451 |
| 2 | 11.518 | 28.794 | 71.245 | 11.518 | 28.794 | 71.245 |
| 3 | 5.000 | 12.500 | 83.745 | 5.000 | 12.500 | 83.745 |
| 4 | 3.557 | 8.893 | 92.638 | 3.557 | 8.893 | 92.638 |
| 5 | 2.030 | 5.075 | 97.713 | 2.030 | 5.075 | 97.713 |
| 6 | .915 | 2.287 | 100.00 | | | |
| … | … | … | … | | | |

**Table 5.** Total Variance Explained of TeraSort

| Component | Initial Eigenvalues | | | Extraction Sums of Squared Loadings | | |
|---|---|---|---|---|---|---|
| | Total | % of Variance | Cumulative % | Total | % of Variance | Cumulative % |
| 1 | 16.264 | 40.661 | 40.661 | 16.264 | 40.661 | 40.661 |
| 2 | 14.100 | 35.250 | 75.911 | 14.100 | 35.250 | 75.911 |
| 3 | 4.728 | 11.820 | 87.731 | 4.728 | 11.820 | 87.731 |
| 4 | 3.226 | 8.066 | 95.797 | 3.226 | 8.066 | 95.797 |
| 5 | 1.681 | 4.203 | 100.00 | 1.681 | 4.203 | 100.00 |
| … | … | … | … | | | |

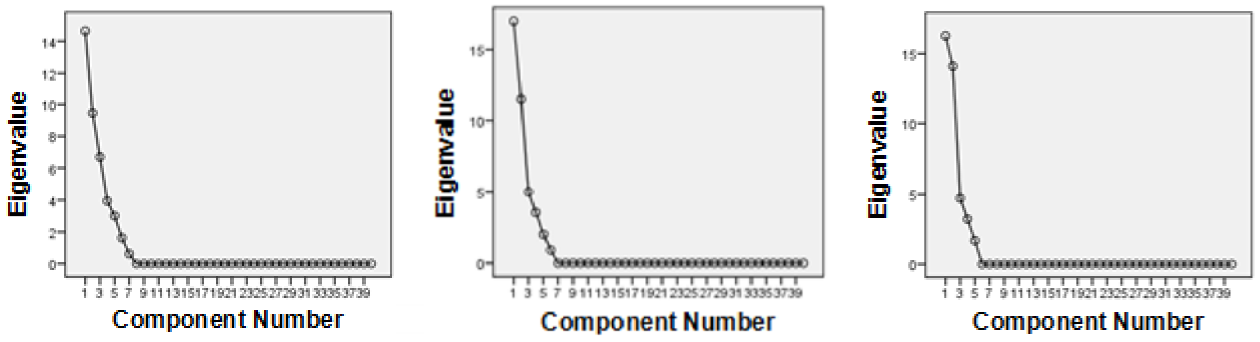

**Figure 2.** Scree Plots of Pi, WordCount and TeraSort (from left to right)

As for TeraSort application, the multiple linear regression equation of principal components as $X_{TS} = (x_1, x_2, x_3, x_4, x_5)$ and execution time as $y_{TS}$ is as follow:

$$y_{TS} = -0.585 + 0.691x_1 + 0.543x_2 + 0.595x_3 + 0.362x_4 + 0.238x_5 \quad (8)$$

### *4.2.3 Prediction Verification*

After training model by historical data, we need to verify the model by current data. We run three benchmark applications for 8 times separately to verify the prediction result of the trained models. The results are shown in Figure 3, Figure 4 and Figure 5. The relative error between predicted execution time and actual execution time of Pi, WordCount and TeraSort are 7.62%, 7.97% and 8.18% respectively. It means that these models can accurately predict the execution time of benchmark applications.

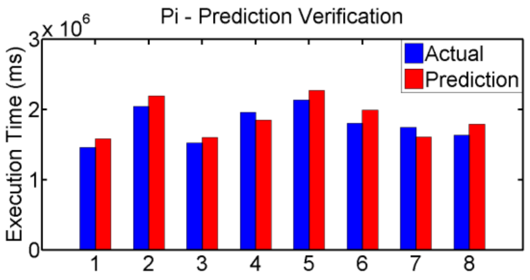

**Figure 3.** Prediction Verification of Pi

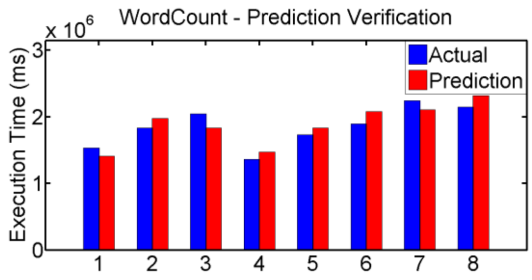

**Figure 4.** Prediction Verification of WordCount

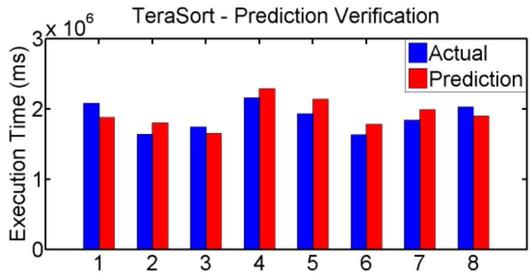

**Figure 5.** Prediction Verification of TeraSort

The trained models, whose prediction average errors are all less than 10%, are verified to be reliable. As a result, they can be used for performance evaluation of HBLSNTAC.

## 4.3 Performance Evaluation

We convert the prediction of each application into a centesimal score, then set the average of the three scores as the evaluation score of the performance of HBLSNTAC. The main function of HBLSNTAC is running off-line statistical applications. And a longer execution time of application means a poorer performance of HBLSNTAC. So we select a typical practical application, whose goal is to find out the top 3 visited websites of 6 time segments of the whole day for each user, and set its execution time as a reflection of the performance of HBLSNTAC. Then study the relationship between the evaluation score and execution time of this practical application.

### *4.3.1 One Single Practical Application*

Relationship between evaluation score and execution time of one single practical application is shown in Figure 6.

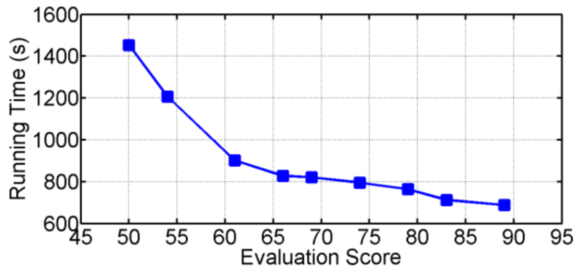

**Figure 6.** Relationship between Evaluation Score and execution time of One Single Practical Application

Roughly, the lower the score is, the longer the execution time is, thus the poorer the performance is, and vice versa. More closely, we can see that there is turning point near 60. When score is higher than 60, a decrease of 5 points of score would cause an increase of 50 seconds of execution time. When score is lower than 60, the same decrease of 5 point of score would cause an increase of 250 seconds of execution time, which is 5 times larger than the increase when score is higher than 60. It suggests that when the performance of HBLSNTAC reaches at a certain poor degree, such as 60 in this case, it is not a good choice to launch an application, because that would cause a much larger increase of execution time.

### *4.3.2 Multiple Practical Applications*

Since the HBLSNTAC is running multiple application parallelly for most of the time, we further study the relationship between evaluation score and total execution time of multiple practical applications. We launch two practical application successively, and launch the second application before the first application completed. The circumstance before the first application launched is the same as one single practical application, so the evaluation score here is predicted before the second application launched. The result is shown in Figure 7.

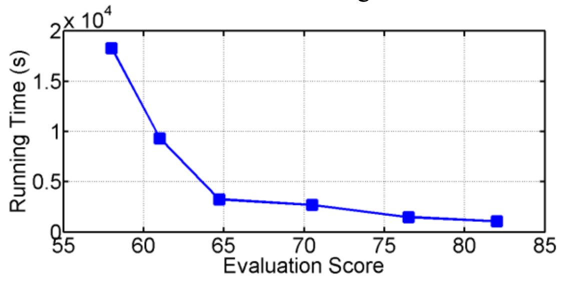

**Figure 7.** Relationship between Evaluation Score and Total execution time of Two Practical Applications

The trend of curve in Figure 7 is about the same as in Figure 6. However, comparing with Figure 6, the turning point is advanced from 60 to 65. It suggests that the greater the number of running applications is, the sooner that certain poor degree is coming. When score is higher than 65, a decrease of 5 points of score would cause an increase of 630 seconds of execution time. When score is lower than 65, the same decrease of 5 points of score would cause an increase of 11210 seconds of execution time, which is more than 17 times larger than the increase when score is higher than 65. Comparing with the circumstance of single practical application, the number of running applications doubles from 1 to 2, however, the increasing range is more than three times from an increase of 5 times to an increase of 17 times. It suggests that the increasing speed of total execution time, in order words, the worsening speed of performance, is more faster than the increasing speed of number of running applications.

### *4.3.3 Summary*

The evaluation results provide a much more straightforward understanding of performance of HBLSNTAC. We used to only have a qualitative understanding of the performance, but with no idea of how exactly the patterns are. Now, we can get a quantitative understanding of performance with the clear evaluation scores. With accumulation of more experimental evaluation results, we can propose a comprehensive user guide strategy. It can guide using behavior of common users, such as suggest the proper timing for users to launch applications and forbid users from launching applications at improper timing, so that to keep HBLSNTAC in a continual good performance status.

## 5 Conclusion and Future Work

In this paper, we have proposed a performance evaluation system of HBLSNTAC. This system associates execution time of three benchmark applications and statistical resource data of servers, then model and analyze these two parts by principal component analysis and multiple linear regression. An evaluation score of performance of HBLSNTAC would be marked by the trained models. The experimental results show that the prediction of execution time of application is reliable with average error less than 10%. Furthermore, the evaluation score can quantitatively reflect the performance of HBLSNTAC, so it could be further used for user behavior guidance. The performance evaluation system is validated to be feasible and reliable.

As for the future work of next stage, firstly, with the expansion of application scenarios of HBLSNTAC, we would include more kinds of applications, such as machine learning and web searching, besides Pi, WordCount and TeraSort. Moreover, the modeling analysis approaches are both linear in this paper, we would change and choose nonlinear modeling approaches to analyze the nonlinear relationship between the execution time of application and resource data. Then study the pros and cons of linear and nonlinear modeling approaches.

## Acknowledgement

This work is supported in part by the Fundamental Research Funds for the Central Universities(2015RC11) , 111 Project of China (B08004) and EU FP7 IRSES MobileCloud Project (Grant No. 612212).

## References

1. T. White, Hadoop: The Definitive Guide. O'Reilly Media, Inc. p. 1-362 (2012)
2. Microsoft IT SES Enterprise Data Architect Team: Hadoop Job Optimization (Microsoft IT white paper), (2014)
3. Hailong Yang, Zhongzhi Luan, Wenjun Li, et al. Statistics-based Workload Modeling for MapReduce. Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW). Shanghai, p. 2043 - 2051 (2012)
4. Shengsheng Huang, Jie Huang, Jinquan Dai, et al. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. Data Engineering Workshops (ICDEW). Long Beach, CA, p. 41-51(2010)
5. M. Ishii, J. Han, H. Makino. Design and Performance Evaluation for Hadoop Clusters on Virtualized Environment. International Conference on Information Networking (ICOIN). Bangkok, p. 244-249 (2013)
6. Jinsong Yin, Yuanyuan Qiao. Performance Modeling and Optimization of MapReduce Programs. Cloud Computing and Intelligence Systems (CCIS). Shenzhen, p. 180-186 (2014)
7. Hadoop Homepage: http://hadoop.apache.org/.
8. Moore, B. Principal component analysis in linear systems: Controllability, observability, and model reduction. Automatic Control. Vol. **26**, No. 1, p. 17-32 (1981)
9. Lu Li. An Internet of Things QoE evaluation method based on multiple linear regression analysis. Computer Science & Education (ICCSE). Cambridge, p. 925-928 (2015)
10. Zhang Liping, Pang Jie, Wang YongChao, et al. SPSS for Water Quality Assessment of Beijing Typical River Based on Principal Component Analysis. Digital Manufacturing and Automation (ICDMA). Changsha, p. 395-398 (2010)