Advances in Computer Science Research, volume 89
2019 International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME 2019)

Research on Preprocessing Method of Performance Monitoring Data in Cloud Environment

Xiao Zhu
College of art and design, Chongqing Vocational Institute of Engineering, Jiangjin, Chongqing, 402260, China

***Abstract*—The analysis of performance monitoring unit(PMU) can not only effectively characterize the operation characteristics of the program, but also provide a basis for performance optimization. Because of the huge number of PMU performance counters in cloud environment, there are some problems such as low monitoring efficiency and low data quality. Based on this, this paper starts with filling missing values and replacing them, uses regression method to complete missing values, uses local filtering method to filter out outliers, and uses the method of data preprocessing based on knowledge base to effectively improve the monitoring efficiency. The experimental results show that the scheme can effectively analyze the possible impact of PMU execution on program operation, and the similarity of the reference value of the processed results is higher than 80%, up to 95%.**

Keywords—PMU; cloud environment; data preprocessing

I. INTRODUCTION

With the development of cloud computing technology, more and more application services are running on cloud platforms. Therefore, the scale and number of data centers are growing at an unprecedented rate. However, large-scale cloud computing infrastructure poses severe challenges to operation and maintenance, and new monitoring and maintenance technologies are needed[1]. The diversity of applications and heterogeneous hardware platforms present in data centers make it more difficult to monitor and optimize performance, which makes performance monitoring in cloud environment particularly important. Compared with the performance monitoring at the system level, the counting results of hardware performance events under PMU can reflect the actual operation of the machine more carefully and directly.

In modern processors, there are usually 2-8 performance counters, which are used to record hardware events that occur at run time[2]. To save chip area and cost, more performance counters can not be designed in processors. To understand the running behavior and optimize the performance of the program, the number of events that need to be monitored is far greater than the number of performance counters, and it is still growing. Because the minimal performance improvements in data centers can be translated into huge cost savings, many researchers are studying how to use PMU monitoring results to optimize performance. There are two ways to map performance events that need to be monitored to performance counters: (1) the number of performance events monitored is less than or equal to the number of counters; (2) the number of performance events monitored is more than the number of counters[3].

The first method can obtain accurate monitoring results, but limited by the number of performance technicians and actual needs, it needs repeated experiments to get the results of all the events required, so this method is inefficient[4]. In addition, the execution time of the same program running multiple times in cloud environment will fluctuate greatly, and the unequal time series results can be obtained, which greatly improves the difficulty of subsequent analysis. The second way can greatly improve the monitoring efficiency and avoid duplication of monitoring work under general needs[5]. However, this method will reduce the quality of monitoring data, which can be shown by the failure to effectively monitor the zero value of events or the sudden increase of data in time series, and the deviation from other data in the same group is more than multiple standard deviations. And the more monitoring events are, the lower the data quality is. In order to improve the monitoring efficiency and quality, it is urgent to improve the existing problems in the second mode.

Considering the differences of server processor architecture, software computing framework and application itself, this paper proposes a data preprocessing method based on knowledge base. For the same running environment and application monitoring tasks under the same computing framework, monitoring events can be different but contain events corresponding to missing values. Combining with the current missing items in monitoring data, a back-up is constructed. For outliers, local filter parameters are configured according to the monitoring data of similar monitoring tasks in the knowledge base, so that the filter can detect and process the outliers in different event results adaptively.

II. EFFICIENCY AND QUALITY OF PMU APPLICATION IN CLOUD ENVIRONMENT

A. Performance Counter

Modern processors are usually equipped with performance monitoring unit (PMU), which consists of two parts: hardware performance event and performance counter. The performance counter is used to measure the number of clock cycles, instructions and cache missing in program execution in micro-architecture. The result can directly reflect the execution of hardware in program execution and analyze the use of hardware resources in program execution[6].

In modern processors, there are usually 2-8 performance counters. For example, Intel Xeon E5 V3 series processors equipped with two dedicated performance counters and four universal performance counters to monitor more than 200

events, which based on Hasw ell-E microarchitecture, the dedicated performance counter can only monitor the number of unpaused clock cycles and the number of successfully executed instructions. The universal counter can be configured to monitor all other events. A performance counter is essentially a special register that records the number of times an event occurs. The number of performance counters owned by different series of processors is different, and the types of hardware events that can support monitoring are different, but the common point is that the number of performance counters in all processor structures is much smaller than the number of events that can be monitored.

B. Application of PMU

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary[7]. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

In the Linux operating system, Perf_events kernel interface acts as a general-purpose, high-level interface to drive performance counters to complete monitoring tasks, and is added to the official kernel after Linux 2.6.31. Perf_event_open( ) is a system call, sysfs is a file system to simplify event naming and tool configuration, and the middle layer includes a general-purpose core computational logic and general interface layer design for different processor architectures.

At present, Perf and Oprofile are popular performance monitoring tools. Perf_events are invoked. In applications, only the events to be tested need to be passed in advance[8]. Event scheduling runs through the whole monitoring process, including binding counter and output record results through trigger mechanism. When the number of events to be tested is more than the actual number of counters, the kernel ensures that every event to be tested has the opportunity to be monitored through time division multiplexing counters. In the process of reuse, polling is adopted and the event occurrence rate is assumed to be the same. The method of calculating event results is formula (1), which infers the counting results in the whole time period according to the proportion of events allocated and the counting value.

$$count_{final} = count_{sample} * \frac{time_{total}}{time_{enable}} \quad (1)$$

Event scheduling can ensure the maximum use of counters and minimize reuse overhead, but it cannot guarantee the accuracy of monitoring results. Some events have specific counter restrictions, which can easily cause event conflicts in the event scheduling process, and aggravate the inaccuracy of monitoring results.

C. Efficiency and Quality of PMU Application in Cloud Environment

The function of performance monitoring unit as program feature description is irreplaceable. Significant results have been achieved in program feature description and performance optimization in cloud environment. However, when using PMU-based monitoring tools, these tasks are limited by the number of performance counters. To ensure the accuracy of monitoring results, relatively conservative monitoring methods are adopted, that is, to ensure that the number of events monitored is not greater than the number of performance counters in each monitoring event. Although this method can ensure the accuracy of monitoring results, it needs repeated operation to obtain more monitoring results, which greatly reduces the monitoring efficiency.

Semantic gap between hardware performance event results and performance has always been a research difficulty. Especially in many events, it is extremely challenging to select a part to participate in monitoring. In extreme cases, all events need to be monitored, which can only be monitored through repeated runs. Some scholars focus on improving the monitoring accuracy of multiplexing performance counters. For example, formula (1) is improved by quantifying errors to obtain higher accuracy. Some scholars improve the formula (1) by changing the rate of events, and improve the accuracy by improving the scheduling method. These studies do alleviate the problem of PMU application, but there are still some phenomena such as over-regulation of monitoring results and missing monitoring results in practical application. The experimental results show that although the number of outliers and missing values is less than 1% in all data, they have a great impact on the feature description of time series.

III. DATA PREPROCESSING METHOD BASED ON KB

To improve the monitoring efficiency of PMU and ensure the quality of monitoring data, a knowledge-based preprocessing method is proposed, as shown in Figure 1. Each monitoring result combines historical monitoring data to detect outliers and missing values, and uses them as reference to replace outliers and supplement missing values. The historical monitoring data mentioned above mainly refers to the monitoring data generated by the same application using the same event. For outliers, an adaptive local filter is designed to automatically judge and process outliers. For missing values, this paper constructs a regression model based on historical data and the missing part of current data, and uses this model to supplement missing values.

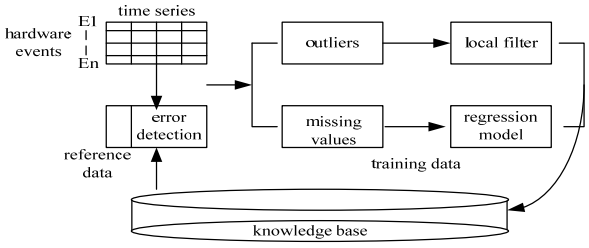

FIGURE I. DATA PREPROCESSING SCHEMATIC DIAGRAM

This paper compared the similarity between different length sequences, and calculate the distance between curves after dynamic time alignment to judge the similarity of event monitoring results. DTW is a similarity measurement method that can better match the shape of time series by bending the time axis. It is used by Bernd and Clifford to measure the similarity of time series. This paper uses the results of historical data as a reference to judge anomalies and missing phenomena. Compared with the monitoring results of the same program under different input data, the monitoring results have high similarity. Big data applications in Hi Bench vary between 50% and 350% in execution time and average level under different input data sets. However, after traversing all hardware events supported by servers, the average DTW distance between multiple monitoring results and reference results is less than 0.07 under different input data sets for the same event, indicating the trend of event monitoring results.

A. Construction of Knowledge Base

Historical data comes from records of the same events when monitoring the same application. This paper designs a knowledge base with two-tier table mechanism, one-tier master table and two-tier data record table. The primary table records the description of the secondary data record table, including event name, application name, record time, record maximum, minimum and average value, and record other events as well as the names of all secondary data record tables. The knowledge base will be updated with monitoring tasks, including replacing monitoring results with identical monitoring events, and adding monitoring results for new monitoring events. For the query of historical records, the records with similar monitoring tasks are returned under the query conditions of monitored programs and problem events. The descriptive information of events is found from the primary table, and then the historical data of specific records and current records are searched twice according to specific needs for the next model training.

B. Local Filter Design

Local filter design includes two cases:

(1) Historical record events: events recorded under the same monitoring application. At this time, the data in the primary table is used to describe the information, and the maximum value of the local record is twice as large as the local threshold value. The value of the local filter is derived from the empirical value in the experiment process.

(2) Initial recording of events: From the observation of the distribution of monitoring data, it can be seen that the data distribution curve is similar to the Gauss distribution, but not a strict Gauss distribution.

Based on the characteristics of Gauss distribution, formula (2) is used as the threshold for data observation, and the sum of mean and n-fold standard deviation is used as the threshold.

$$List.threshold = List.mean + n * List.std \tag{2}$$

C. Construction of Regression Model

The difference of monitoring results does not change dramatically with the change of input data set. Historical data can be used as a basis for supplementing missing values. However, the sequence length in historical records is not necessarily the same as that of current missing items. It is difficult to complete missing values through location information. However, the regression model can be constructed quickly by using the correlation between events, and there is no dislocation when using the regression model to supplement the data. In this paper, we use the data of similar monitoring events in the knowledge base, that is, for the same running environment and application monitoring tasks under the same computing framework, monitoring events may not be identical, but must contain missing values. The common event set is obtained from the intersection of monitoring data and event set in the current results as training data, and the regression model is constructed. The regression model is implemented by KNN regression.

Assuming that only event a has missing item in the result R1 of monitoring event (a, b, c, d) containing missing item, find the result R2 containing event a, and monitor the record result R2 of the same program. R2 contains (a, b, c, e, f) monitoring records.

1. Same monitoring events:

$$R1 \bigcap R2 = (a, b, c) \tag{3}$$

The missing item a is selected as the objective result, and the characteristic events are B and C.

2. Integrating data:

The training set Ts1 chooses the monitoring results of events a, b, C in R2 and the missing parts in R1. The missing part is defined as Ls1.

3. KNN algorithm is applied to establish classification model based on K nearest distance data.

4. According to the subordinate classification of (b, c) prediction a, the average value of all a in the class is the final prediction value of the missing part.

IV. Experimental results and analysis

The experimental cluster consists of four Dell servers, one of which serves as the primary node and the other three as the computing sub-nodes. Each server is equipped with 16-core Intel E5-2630 v3@ 2.4 GHz processors, and the server memory size is 64GB, operating system Ubuntu 14.04.

Cluster management system is Mesos 1.0 and uses Hadoop 2.7 as the computing framework. At the same time, eight typical cloud applications from Hibench are selected as the monitored programs: Pagerank, Aggregation, Scan, Bayes, Kmeans and Sort, Wordcount.

A. Data Quality Assessment

From the experiments, it can be seen that even in the same system environment, repeated monitoring of the same program can only get very similar experimental results, but cannot get exactly the same experimental results. Therefore, the evaluation of data quality in this paper is based on the similarity of data results and reference values as the evaluation criteria of data cleaning effect. By calculating DTW of monitoring data and reference values, the similarity of the results is evaluated, and the difference degree Diff (%) before and after data cleaning is compared to show the effect of processing, such as formula (4).

$$\begin{cases} \text{Distance}_{test} = \text{DTW}(\text{data}_{test}, \text{data}_{ref1}) \\ \text{Distance}_{ref} = \text{DTW}(\text{data}_{ref1}, \text{data}_{ref2}) \\ \text{Diff} = \dfrac{\text{Distance}_{test} - \text{Distance}_{ref}}{\text{Distance}_{ref}} \end{cases} \quad (4)$$

B. Analysis of Experimental Results

When multiplexing is adopted, the system can monitor multiple events simultaneously through time division multiplexing PMU. Taking the Word Count program in Hi Bench as an example, the accuracy of simultaneous monitoring of changes in the number of events is observed (Figure 2). The results show that the degree of difference increases as the number of monitored events increases. After pretreatment, the degree of difference decreased significantly. After pretreatment, the degree of difference did not exceed 42% of the reference value, and the lowest deviation was above 50%. In comparison, the quality of monitoring results is greatly improved, and the difference of DTW distance is less than 20% when the number of events monitored at the same time is less than 24. The maximum number of events that can be monitored can be increased to five times of the mode, and the maximum monitoring efficiency can be increased by five times to ensure that the data can still maintain a high quality.

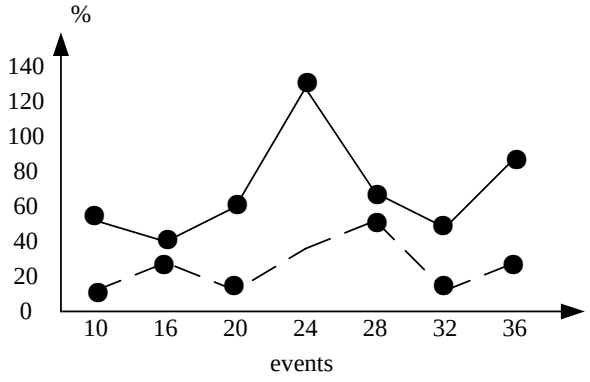

FIGURE II. QUALITY COMPARISON OF SIMULTANEOUS MULTI-EVENT MONITORING

At the same time, other applications in Hibench are validated and compared with the current mean replacement method (GDP) for missing and outliers. When monitoring 10 hardware events, the Database-based preprocessing method (KBDP) greatly improves the similarity with the reference results compared with GDP. The overall average difference is reduced from 52.7% before processing to 8.7%, which means the average similarity is 91.3%.

V. SUMMARY

The pre-processing method based on knowledge base proposed in this paper effectively solves the problem of PMU application in cloud environment, that is, the limitation of reliable data can be obtained by completing data collection by configuring fewer events than the number of performance counters. Through the pre-processing of monitoring results after reusing PMU, the abnormal values are dealt with, the missing values are filled up, and the data quality is improved. The monitoring data can be directly applied to the follow-up analysis work, which greatly improves the work efficiency.

ACKNOWLEDGMENT

Thank you Professor Wang, my colleagues and my family.

REFERENCES

[1] Hennessy J L,Patterson D A.Computer architecture: a quantita-tive approach[M].Elsevier,2012.

[2] Kanev S,Darago J P,Hazelwood K,et al.Profiling a warehouse-scale computer[C].ACM/IEEE 42nd Annual International Sympo-sium on Computer Architecture(ISCA),2015:158-169

[3] Barroso L A,Dean J,Holzle U.Web search for a planet:The Google cluster architecture[J].IEEE Micro,2003,23(2):22-28.

[4] Michael Ferdman,Almutaz Adileh,Onur Kocberber,et al.Clear-ing the clouds:a study of emerging scale-out w orkloads on modern hardware[C].Proceedings of the 17th International Conference on

[5] Architectural Support for Programming Languages and Operating Systems (ASPLOS),2012.

[6] Ren G,Tune E,Moseley T,et al.Google-wide profiling:a con-tinuous profiling infrastructure for data centers[J]. IEEE M icro,2010,30(4) :65-79.

[7] Mars J,Tang L,Hundt R,et al.Bubble-up:Increasing utilization in modern w arehouse scale computers via sensible co-locations[C].Proceedings of the 44th Annual IEEE/ACM International Symposi-um on Microarchitecture(MICRO),ACM,2011:248-259.

[8] Devices A M.BIOS and kernel developer's guide for AMD family 15h models 00h-0Fh processors[M].2013-2015 Advanced Micro Devices,Inc.,2013.