Mining Performance Knowledge in User Logs

Stella Pachidi and Marco Spruit

Department of Information and Computing Sciences, University of Utrecht,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands,
{s.pachidi, m.r.spruit}@uu.nl

Abstract. Software performance is a critical aspect for all software products. In terms of software operation data, it concerns knowledge about the software product's performance when it is used by the end-users. In this paper, we suggest data mining techniques that can be used to analyze software operation data, in order to extract knowledge about the performance of a software product when it operates in the field. Focusing on Software-as-a-Service application, we present the Performance Mining Method to guide the process of performance monitoring (in terms of device demands and responsiveness) and analysis (finding the causes of the identified performance anomalies). The method has been evaluated through a prototype, which was implemented for an online financial management application.

Keywords: Data mining, software performance, software operation knowledge, web analytics.

1 Introduction

The analysis of data collected during the operation of a software product by its end-users may yield significant information about the *performance* of the product. While most research focuses on the recording of software operation data and on the integration, presentation and utilization of the knowledge acquired through general statistics and visualization techniques [1], sufficient support of data analysis functionalities is missing. *Data mining techniques* could be used to analyze operation data, in order to identify performance anomalies and analyze their causes.

By analyzing user log data we could extract information on how well the software meets its performance demands when operating in the field, in terms of device demands (i.e. the fraction of the resources used on the end-user's work unit, or on the servers, etc.), or in terms of responsiveness (i.e. how fast requests, actions and so forth are processed). By monitoring the performance on a regular basis (daily, weekly, monthly), we are also able to compare the values of the metrics and find the causes of performance problems. Performance monitoring and analysis through software operation data mining, could contribute significantly to the performance testing and quality control processes, as it can provide an automated alert system, which identifies problems on the performance of a software product and even analyses their cause. The direct advantage would be the increase of quality of the product, which would also help increase the customer satisfaction. Another advantage would also be a decrease in the costs for performance testing, since many problems would be identified, or even prevented, through the automated analysis of the software operation data.

In practice, a lot of vendors use embedded code to record the operation data, and then analyze and visualize the acquired data in a non-automatic way. However, manual analysis cannot yield interesting patterns. But even in the case where specific tools are used for acquisition of user logs, a recent survey [2] reflects that most acquisition techniques that are currently employed use general statistics, or simple visualization techniques; sufficient support of data analysis functionalities is missing.

A uniform way for analyzing software operation data is needed, in order to automatically monitor and analyze the performance of software products in the field, becomes evident. In this research, we will try to cover this gap by answering the following research question: *How can we detect performance anomalies and their causes in software operation data?*
In order to answer this question, we will first specify which variables will need to be inspected in software operation data to implement performance analysis. Then, we review data mining techniques, in pursuance of techniques that are applicable on software operation data, in order to (1) detect anomalies in the performance of the software on the end-user's side, and (2) reason about the cause of each identified anomaly. We construct a method for mining performance knowledge in software operation data. We also construct a prototype, which will be used to validate this method and to test a subset of the suggested data mining techniques. A case study in a Dutch software vendor will be performed to validate the method and prototype.

2 Related Work

Some preliminary research on testing performance with log file analysis was published by Andrews [3], who suggested a framework for creating and analyzing log files, in order to find bugs in the software, but also to test the system performance, by tracking for example internal implementation details (like memory allocation). Johnson et al. [4] also incorporated the generation of performance log files in the performance testing process, to enable developers and performance architects to examine the logs and identify performance problems.

As far as data mining is involved in the performance analysis task, some related research has been conducted. Imberman [5] identifies the parts of the Knowledge Discovery in Databases process [6], which are currently being followed by performance analysts (regression, graphical analysis, visualizations and cluster analysis). Also, she explains further the parts that are less often used so far, but could be useful for analyzing performance data (association/dependency rule algorithms, decision tree algorithms and neural networks). Jain [7] suggests the regression technique, for modeling performance related variables, and time series analysis, for managing and analysing performance data gathered over a time period. Vilalta et al. [8] study the use of predictive algorithms in computer systems management, for the prediction and prevention of potential failures. Pray and Ruiz [9] present an extension of the Apriori algorithm, for mining expressive temporal relationships from complex data, and test it with a computer performance data set. Finally, Warren et al. [10] suggest an approach for finding the ``hot spots" of a program, i.e. the operations that have the highest impact on performance.

3 Research Methodology

In our research we follow the *Design Science Research (DSR)* approach: questions are answered through the creation and evaluation of innovative artifacts that render scientific knowledge and are fundamental and useful in understanding and solving the related organizational problem [11]. In our case, the research artifacts correspond to the method and to the prototype (which actually constitutes an instantiation of a method) for software performance mining. In order to structure our data mining research, but also to assemble our Performance Mining method, we follow the CRISP-DM Reference Model [12].

This research is delimited by some specific constraints: First of all, we limit our research to the analysis of performance for software-as-a-service applications, although the extensibility to other types of product settings will be considered in the construction of our method. Furthermore, we assume that the software operation data are gathered in a central database through logging tools and libraries, which are embedded in the software product's code. Finally, we have selected to work with the *R* environment end programming knowledge [13], thus the techniques implemented in the method and the prototype are selected accordingly, while certain concepts mentioned in the method are described with terms of objects, techniques and approaches used in R, though this will not affect the adaptability of our method to other data mining systems and tools.

As the aspect of performance is a very complicated and wide domain, it is necessary to set some hypotheses about what types of performance problems we aim to detect and analyze with our suggested techniques and method: The performance anomalies that we aim to inspect in software operation data can be categorized into two types: problems related to *device demands on the servers*, which include unusually high utilization of resources (CPU, memory, disk, etc), in the computers on which the servers are running; and problems related to *responsiveness*, which refer to situations when the response time is much higher than usual. We will try to reason about the responsiveness problems that are detected. We consider the following possible *causes* that could be responsible for increasing the response time: a problem on the server, the seasonal effect, a software problem with a specific application, a processing problem, a problem on the database, a significantly high size of data to be loaded, or a problem on the user’s side.

In order to gather the software operation data that are necessary to monitor and analyze performance, we need to decide which variables will need to be inspected. We distinguish three categories of variables: a) performance metrics related to the resources usage on the machines where the servers are running (e.g. % processor time); b) performance metrics related to responsiveness (e.g. duration of loading a page); and c) variables related to the software operation details (e.g. datetime, customer, application accessed, etc.).

4 Data Mining for Performance Analysis

In this section we present the data mining techniques that may be applied on software operation data, in order to analyze the performance of software-as-a-service products. We suggest three main performance analysis tasks: monitoring performance on the servers, monitoring performance of applications, and analyzing performance of applications. In the following sub-sections, we are presenting the data mining techniques that can be used in each analysis task. More emphasis is set on the techniques which were implemented in our prototype.

4.1 Monitoring Performance on the Servers

Monitoring performance on the servers is related to monitoring the resources utilization on the machines where the servers are running. We are interested in detecting performance bottlenecks related to the resources usage, and we aim to provide an estimation for the future capacity of the resources.

Values of the performance metrics are collected periodically (between time intervals of e.g. 1-3 minutes). Each metric value is stamped with the specific date and time of its measurement; hence we get to have a sequence of values, which can be considered as *time-series data* [14]. By treating each performance counter as a time series involving variable, we apply time series analysis and burst detection to model the time series, and subsequently trend analysis, to forecast future values of the time-series variable.

Time series analysis consists in analyzing a time series into trend or long-term movements, cyclic variations, seasonal movements and irregular or random movements [14]. An example of a time-series decomposition may be viewed in figure 2.

Burst detection [15] can be useful during the auditing of computer system logs for tracking down problems or system bottlenecks. The process of burst detection should be performed after times series decomposition, in order to remove the bias of the burst intervals in the trend analysis.

Forecasting is important for our performance counters monitoring task, in the sense that we want to receive an alert when significant increases or decreases of the performance counters are expected, in order to prevent potential performance problems. *Trend analysis* includes the extraction of the general direction (trend) of a time series, the analysis of the underlying causes which are responsible for these patterns of behavior, and the prediction of future events in the time series based on the past data. We suggest the use of *trend estimation* technique, which employs *linear regression* and *extrapolation* techniques.

4.2 Monitoring Performance of Applications

Monitoring performance of applications consists in monitoring the responsiveness of the system, with regard to individual applications (e.g. .aspx pages, or methods, etc.). We are interested in detecting delays in the response time of applications, or even predicting possible decrease of the responsiveness given the current configuration.

Every action that happens on the end-user's side is represented by a record that includes a timestamp, details of the operation and the response times. Therefore, the responsiveness data are *sequence data*, since they constitute sequences of ordered events, and can be treated as *time-series data*. However, here we have sequences of values which are measured at unequal time intervals. Furthermore, the same application may be accessed by more than one users at the same time. Therefore, the implementation requires a transformation of the measurements into periodical data before the execution of *time series analysis*, *burst detection* and *trend analysis*.

4.3 Analyzing Performance of Applications

Analyzing performance of applications aims to find the causes for events where response time is significantly higher than usual. The categories of performance problem causes were defined in section 3.

We suggest the use of *association rules mining*, which is used to discover interesting relationships that are hidden in large data sets. The relationships constitute relations between attributes on the value-level, which are represented by *association rules*, i.e. expressions in the form of $A \rightarrow C$, where *A* and *C* are sets of items in a transactional database, and the right arrow implies that transactions which include the itemset *A* tend to include also the itemset *C* [16]. Our goal is to mine association rules that are interesting, by using several measures such as *support* (the proportion of transactions in the database that contain both itemsets \$A\$ and \$C\$, over the total number of the transactions in the database) and *confidence* (the conditional probability of having the itemset \$C\$ given the itemset \$A\$). In order to consider the discovered rules interesting, they need to outweigh the thresholds we set for the minimum support and the minimum confidence. The computational complexity can be reduced by the A Priori principle [16].

The performance analysis dataset consists of cases, where each case corresponds to an operation on the software product by an end-user and is represented by a record that includes the application that was used, type of action, timestamp and other operational details, and the corresponding response time. This set of records could be transformed into a set of transactions by following the pre-processing steps suggested by Hahsler [17]. Afterwards, we could perform association rules mining on the performance data, and search for interesting and significant rules in the form of $A \rightarrow C$, where *C* corresponds to significantly high response times.

5 Performance Mining Method

In order to provide guidance to the software vendors in the analysis and monitoring of their software products' performance, we propose the Performance Mining Method. The method has been constructed with the Method Engineering approach [18]. The method includes three activities, which correspond to the three suggested tasks: monitoring the performance on the servers, monitoring performance of applications, and analyzing performance of applications. Each activity consists of several steps (subactivities), which follow the CRISP-DM cycle [12]. The process diagram is presented in figure 1.

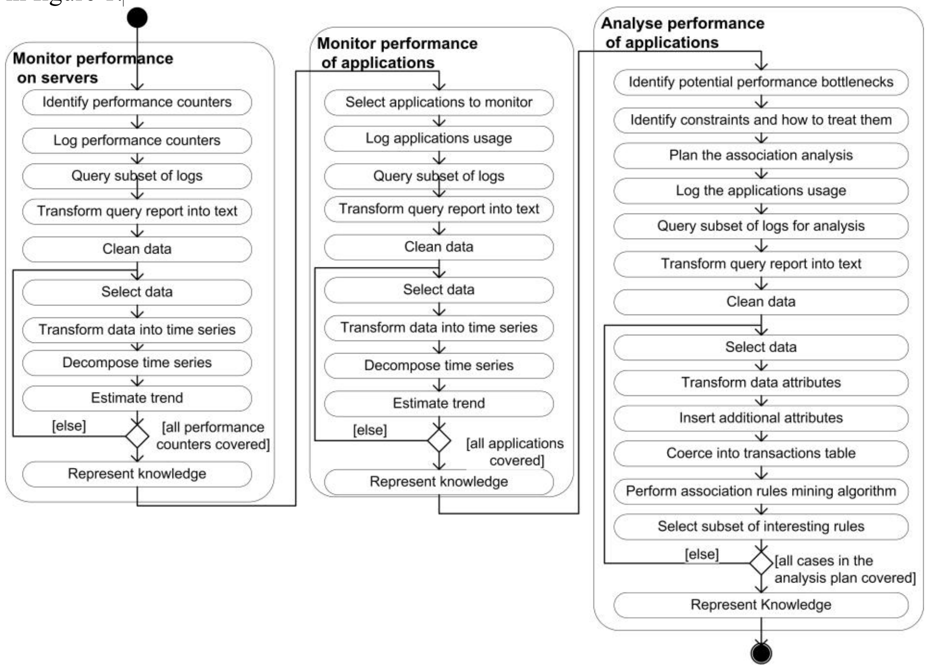
Monitor performance on servers
Identify performance counters
Log performance counters
Query subset of logs
Transform query report into text
Clean data
Select data
Transform data into time series
Decompose time series
Estimate trend
[else]
[all performance counters covered]
Represent knowledge
Monitor performance of applications
Select applications to monitor
Log applications usage
Query subset of logs
Transform query report into text
Clean data
Select data
Transform data into time series
Decompose time series
Estimate trend
[else]
[all applications covered]
Represent knowledge
Analyse performance of applications
Identify potential performance bottlenecks
Identify constraints and how to treat them
Plan the association analysis
Log the applications usage
Query subset of logs for analysis
Transform query report into text
Clean data
Select data
Transform data attributes
Insert additional attributes
Coerce into transactions table
Perform association rules mining algorithm
Select subset of interesting rules
[else]
[all cases in the analysis plan covered]
Represent Knowledge

Fig. 1. Process Diagram of the Performance Mining Method

6 Evaluation

The Performance Mining Method is instantiated in a prototype, which has been created in order to implement and validate the method and the suggested data mining techniques. The prototype is implemented in R [13], and is aimed to analyze operation data which are gathered through logging tools, and are stored in logs, in a central database. The prototype consists of three scripts which correspond to the three main analysis tasks: monitoring performance on the servers, monitoring performance of applications, analyzing performance on applications.

In order to validate our method, we performed a case study in a Dutch software company, where we implemented the Performance Mining Method and ran the prototype in the context of a real software product. The product is a Software-as-a-Service accounting solution with over 10.000 customers and more than 3.500 users per day. The application logs performance and usage data, through a log-generating code that is incorporated in the system layer of the application. However, no sophisticated analysis is performed in any of the logs, and the analysts only perform basic statistical operations on the data and produce common statistical plots for their monitoring.

We tried to observe how our suggested Performance Mining method could be applied for this case, in order to monitor its performance and analyze the performance problems that appear. We also ran our prototype with sample log data collected during the product's operation, in order to validate its functionality.

As far as *monitoring performance of servers* is concerned, we used the related script from our prototype to monitor performance counters that were measured on the machines of two web servers in the period of two months. An example of the output can be viewed in figure 2, which illustrates the time series decomposition for the performance counter *%Processor Time*. In the observed data plot we can clearly identify a burst interval between day 13 and 15. Such a burst would influence the result of our trend analysis and therefore needs to be handled before estimating the trend of the counter. Furthermore, the burst raises a lot of concern to the performance analyst, who will need to conduct a thorough analysis of the logs recorded in that time interval, in order to look for the causes of these anomalous values of the counter. An increasing trend on the performance counter might influence the performance architect to consider the possibility of expanding the hardware, or updating the servers architecture in the future.

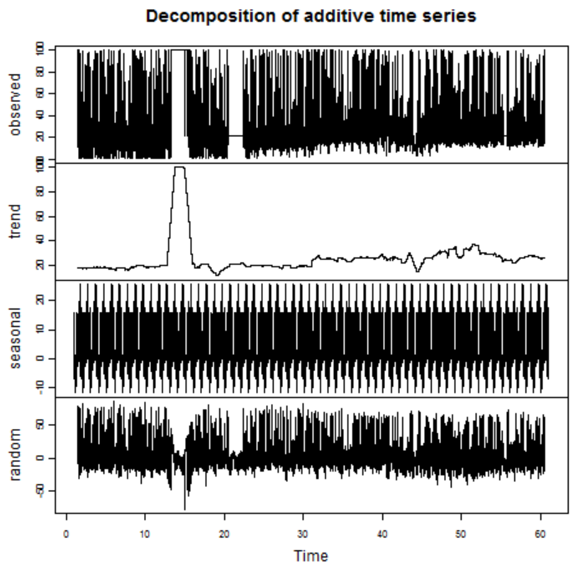

Fig. 2. Decomposition of the time series for the performance counter *% Processor Time*

Regarding monitoring performance of applications, we used the related script from our prototype to monitor performance of applications that were used in the period of 3 months. An example of the output is provided in figure 3, which illustrates the trend analysis for the duration of BankStatus application (after having removed any identified bursts). As we can roughly see from the fitted line on the graph, the duration of the application has increased for more than 180ms. This is a significant increase that the performance analyst should look further into.

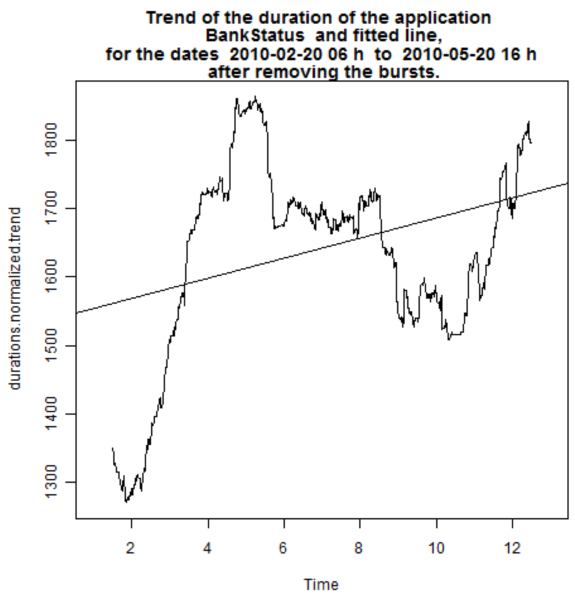

Fig. 3. Trend estimation for the BankStatus application after removing the burst intervals

In order to analyze the performance of applications, we made a plan for the association analysis, which included which variables would be included in the logs, what type of response time we would use to find performance problems, etc. We used the related script from our prototype, to analyze the performance of applications, from the logs that were collected in three consecutive days during which several delays in the response time for processing web requests were observed. After preprocessing the data (which contained 441662 rows), we created the binary incidence matrix, which was mined through the apriori function to find the most frequent itemsets and the rules which had high duration of the right hand side.

7 Discussion and Conclusions

In this paper, we tried to answer the research question of *how we can detect performance anomalies and their cause in software operation data*. We searched for data mining techniques that are particularly applicable for identifying performance problems that appear on the end-user's side; and techniques that could be used to pinpoint the causes of these problems. We suggested a method to apply performance mining in a uniform way. We also validated this method and tested a subset of the proposed techniques through a prototype, which was deployed in the case of an online financial management application.

This research contributes to the domain of user logs, by suggesting data mining techniques to (1) analyze software operation data in a uniform and automated way; and (2) to extract sophisticated knowledge (complex associations, interesting patterns, etc.) about the performance of the software product when it operates in the field. The performance knowledge that can be derived from software operation data, will be supportive to the processes of software product management, development and maintenance. Furthermore, this paper is expected to enrich the domain of Software Performance Evaluation.

From a practical perspective, the method that we suggest for Performance Mining constitutes a reference process model, which can be followed by software vendors, to monitor and analyze the performance of their software in the field. The prototype that we present can easily be adjusted to analyze the performance of specific software products. Finally, the extracted knowledge may support the management, development and maintenance of their software, and yield business advantages.

References

1. van der Schuur, H., Jansen, S., and Brinkkemper, S.: A Reference Framework for Utilization of Software Operation Knowledge. In 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 245--254. IEEE Computer Society, Lille, France (2010)
2. Kristjánsson, B. and van der Schuur, H.: A Survey of Tools for Software Operation Knowledge Acquisition. Technical Report UU-CS-2009-028. Department of Information and Computing Sciences, Utrecht University, Netherlands (2009)
3. Andrews, J. Testing using log file analysis: Tools, methods, and issues. In: 13th IEEE International Conference on Automated Software Engineering, pp. 157-166. IEEE Computer Society, DC (1998)
4. Johnson, M. J., Ho, C.-W., Maximilien, E. M., and Williams, L.: Incorporating performance testing in test-driven development. IEEE Software, vol. 24, pp. 67-73.(2007)
5. Imberman, S. P.: Effective use of the kdd process and data mining for computer performance professionals. In: Int. CMG Conference, pp. 611—620, Computer Measurement Group, Reno, NV (2001)
6. Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P.: From data mining to knowledge discovery: an overview. In: Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (eds.) Advances in knowledge discovery and data mining, pp. 1-34. American Association for Artificial Intelligence, Menlo Park, CA, USA(1996)
7. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, New York, NY (1991)
8. Vilalta, R., Apte, C. V., Hellerstein, J. L., Ma, S., and Weiss, S. M.: Predictive algorithms in the management of computer systems. IBM Systems Journal, vol. 41, pp. 461—474. (2002)
9. Pray, K. and Ruiz, C.: Mining expressive temporal associations from complex data. In: Perner, P. and Imiya, A. (eds.) Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science, vol. 3587, pp. 637-637. Springer, Heidelberg (2005)
10. Warren, C. E., Payne, D. V., Adler, D., Stachowiak, M., Serlet, B. P., and Wolf, C. A.: Software performance analysis using data mining, http://www.google.com/patents/about?id=Xc3OAAAAEBAJ.
11. Hevner, A. and Chatterjee, S.: Design Research in Information Systems, Integrated Series in Information Systems, vol. 22. Springer, New York, NY (2010)
12. CRISP-DM 1.0 Step-by-step data mining guide. Technical report, U.S.A. (2000)

13. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria (2008)
14. Han, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA (2005)
15. Vlachos, M., Wu, K.-L., Chen, S.-K., and Yu, P. S.: Fast burst correlation of financial data. In: Jorge, A., Torgo, L., Brazdil, P., Camacho, R., and Gama, J. (eds.) 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, October, Lecture Notes in Computer Science, vol. 3721, pp. 368--379. Springer, Porto, Portugal (2005)
16. Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules in large databases. In: 20th International Conference on Very Large Data Bases, pp. 487-499, Morgan Kaufmann Publishers Inc, San Francisco, CA (1994)
17. Hahsler, M., Gruen, B., and Hornik, K.: Arules – a computational environment for mining association rules and frequent item sets. Journal of Statistical Software, vol. 14, pp. 1--25. (2005)
18. Weerd, I. v. d. and Brinkkemper, S.: Metamodeling for situational analysis and design methods. Hershey: Idea Group Publishing, Barcelona, Spain (2008)