Server Workload Classification and Analysis with Machine Learning Algorithms

San Hlaing Myint
University of Computer Studies, Yangon
shm2007@gmail.com

Abstract

The main factor in measuring server performance is the accuracy of detection mechanisms. Sever is needed to detect server overload condition accurately. Therefore, it can be satisfied customers by reducing request drop rate. Server overload detection would be an initial step of overload control system. In order to provide such a detection mechanism, it is important to choose the best classifier which is the most suitable for our dataset. Selecting correct classifier maximize the performance of detection mechanism.

In this paper, we present how server workload classification task is performed by using different machine learning classification methods and how the best classifier improve overload detection mechanism. We make a synthetic dataset by using window performance monitor tool. Many classifiers are evaluated over synthetic dataset.

1. Introduction

One of the key factors in customer satisfaction is the application performance. If an application regularly takes too long to respond, the customer may become unsatisfied and he may eventually switch to another service provider. Our aim is to detect performance problems for Ultra-Large-Scale (ULS) system and to provide an overload prevention mechanism. Some existing approaches made their detection based on small amount of performance metric such as response time [10]. Proposed method is based on measuring a wide variety of performance counters such as *…Memory\Available Mbytes* and *…Processor\%processor Time*.

Some conventional control approaches based their request discard decisions on hard thresholds and make admission decision [11]. Traditionally, server utilization or queue length has been the variables mostly used in admission control schemes. In this work, the response of performance counters are chosen as the control parameter since it indirectly affects system utilization and system overloading. Admission controller negatively affects the performance of some customers, therefore, our approach tried to avoid this problem. This approach reduces request drop rates and raises up server performance.

When server gets overloaded, the response time of hardware components become long and the response time of a service become long which affects the many users. When hardware components response so long, server becomes overload vice versa. One of the best solutions is to reduce request drop rate of admission control mechanism by predicting the state of the server. So the admission control mechanism handles some requests with fair admission decision whenever the arriving traffic is too high and thereby maintains an acceptable load in the system.

In server overload control system, an interesting problem is that the underling hardware should be scale up when request rate exceed the limitation of server. In real time situation, it is very difficult to scale up hardware resources before user notices a decrease in performance. Therefore, server overload situation is needed to detect accurately with complete set of performance counter. The acceptance decision of admission control process mostly depends on accuracy of overload detector. So it is very important to take a complete relevant performance metric for overload prediction.

In this paper, server overload detection mechanism is presented as a part of overload control system that we proposed. We present how to select the best classifier and it can improve server overload control. This research is an ongoing research. We make a synthetic dataset by using performance counter patterns. Experimentation is performed based on many classifiers for evaluating the performance of the best classifier. Experimental results demonstrate that selected classifier improve the accuracy of perdition decision.

2. Related Work

The aim of existing research on overload prediction mostly related to admission control and resource scheduling issue for server overload prevention. An admission control mechanism for web servers using neural network (NN) was proposed in [2]. The control decision is based on the desired web server performance criteria: average response time, blocking probability and throughput of web server. A NN model was developed able to predict web server performance metrics based on the parameters of the Apache server, the core of the Linux system and arrival traffic. In [4], Server overload detection method is proposed by using statistical pattern recognition method. The classifier predicted server overload situation (underload, normal, and overload) on 14 performance counters that they assume to be significant for overload detection.

[3] presented a dynamic session management based on reinforcement learning. A learning agent decides the acceptance or rejection of an arriving session by estimating the response time only for service request. In

[5], an efficient admission control algorithm, ACES, based on the server workload characteristics. The admission control algorithm ensures the bounded response time from a web server by periodical allocation of system resources according to the resource requirements of incoming tasks. By rejecting requests exceeding server capacity, the response performance of the server is well maintained even under high system utilization. The resource requirements of tasks are estimated based on their types. A double-queue structure is implemented to reduce the effects caused by estimation inaccuracy, and to exploit the spare capacity of the server, thus increasing the system throughput.

In our experience, most of existing research emphasized response time and other related factors of web service to predict system resource and server overload condition. In our consumption, service response time is directly related to hardware components of physical server. Therefore, it is impossible to lack estimation response time of hardware components. It is indispensible to know the response time value of hardware components to increase estimation accuracy of perdition method. In this work, performance counters are chosen as important variables of detection mechanism and the best classifier is defined on our own experimentation.

3. Proposed System Architecture

Proposed Architecture of overload control mechanism is described in Figure 1.In this architecture; there are three modules such as classification module, overload prediction module, scheduling module and admission control module. The number of requests which will be processed is controlled (scheduling and admission control), and a suitable Queue is dynamically selected for request assignment. In classification module, incoming request are assigned to each classes based on their processing time by using lookup table. This module parses each incoming request URL to extract its file name and searches in a lookup table.

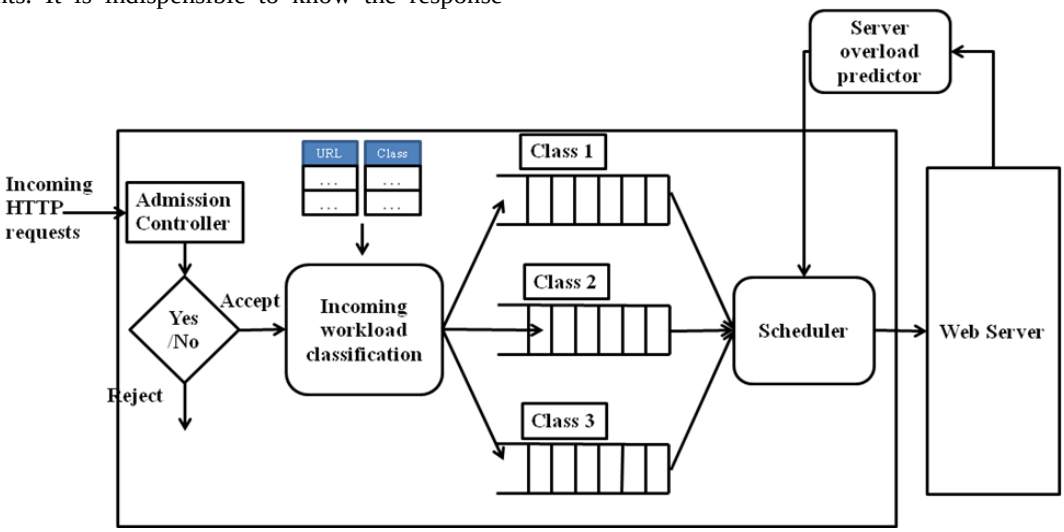

Figure 1.Architecture of proposed overload control mechanism

After a request has been classified, the queue module is invoked. The queue module implements first in first-out (FIFO) queuing policy. The queue module suspends incoming requests and adds each of them to the queue corresponding to its class. After this phase, the scheduler listens the state of server form prediction module and selects requests from queue according to scheduling algorithm. The overload prediction module predicts whether server is in which state (Underload, Normal and Overload) based on performance counter patterns. And then send server situations information to admission control module and scheduling module. Once server overload occurs, admission control module rejects new incoming requests and reconsiders acceptance of new incoming request when server pass overload situation. This research is an ongoing research. Therefore, in this paper, server overload detection mechanism is presented as a part of overload control system and evaluated the performance of the best classifier on based line performance measures.

4. Server Overload Detection

In this section, we will explain how to detect the overload condition of physical server by using classification method. In this approach, the following stages are distinguish,

- Data generation
- Data preparation
- Designing classifier
- Evaluating classifier

The implementation of these stages will present in next section.

4.1. Data Generation

The first step of proposed method is collection data from the server by using performance monitor (PM).PM produces performance counters which describe server states(State1,State2,State3 which is defined in proposed Server Overload Control approach.Here,State1,2,3

means ;Low, Normal and High).Data set is generated on window server which allow to use Performance Monitor tool. In order to train classifier, synthetic data set is created. We avoided collecting data from real server because it can take long time to get enough data. Therefore, synthetic data set is created by using load generator such as **CPU Busy** which performs a stress test on the same server. But the specifications of production server must equal to real server. During stress test, the load will vary from one state to another. Two measurements are interested for training data set;

- Performance counter pattern which is used to describe the server state
- Performance counter values which are used to decide whether a performance counter pattern should be defined as state1 or state2 or state3.

4.2. Data Preparation

4.2.1. Feature Selection

Performance counters are measurements of system state or activity. They can be included in the operating system or can be part of individual applications. Windows Performance Monitor requests the current value of performance counters at specified time intervals. Actually Performance Monitor tool can generate 1948 performance counter patterns, but some of these are not very unlikely to be of interest when monitoring for overload. In [4] 36 counter patterns are selected which are assume to be significant for overload prediction. In our consumption, all performance counters related to physical servers and their processes, some counters may not be significant because of server behavior. We can define which counters patterns are significant or not by examining their counter values. Firstly we calculate information gain of each feature by using information gain ranking filter.

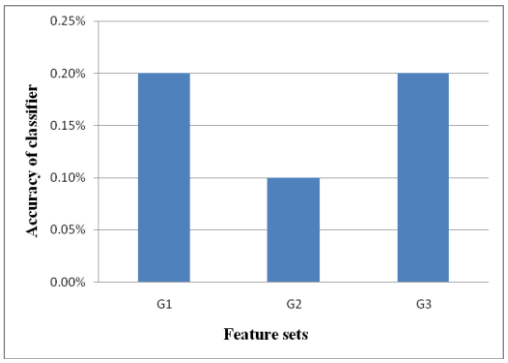

Figure 2. Result of Feature Selection

Here, we can divide features into three groups. First group is >=1 , second group is >=0 and gorup3 (G3)which combine G1 and G2 .The result is shown in Figure 2.Accrodng to the figure; we can see G1 is obviously effective on classifier. Therefore, we selected significant features which contain higher information gain value (>=1).Table I presented some of selected performance counter list. We can improve data set by reducing features from 1948 to 926 features.

4.2.2 Information gain Ranking Filter

Information gain (IG) is a feature ranking method based on decision trees that exhibits good classification performance. Information gain used in feature selection constitutes a filter approach. Filter approaches select features using characteristics of individual features. Advantages of the filter-based techniques are that they can easily scale up to high-dimensional datasets and that they are computationally fast and independent of the learning algorithm. Information gain is a measure based on entropy. Entropy is one of the most commonly used discretization measures.

4.2.3 Data Transformation

To be able to predict server state correctly, it is important to transform it for use with a classifier. Performance Monitor tool generate data collector set which contain performance counter patterns and values. These are need to assigned to a target class (overload, normal and underload) based on their values. We built our data set in the form of (ID, Values).Each performance counter contain about 2000 records which are recorded during 1 minute. The values are used to define which pattern are meet with S1, S2 or S3.

Table 1. Selected Performance Counter List

Main Categories	Counter Name	ID	Value
Physical Disk	Avg.Disk Queue Length	D1571	0.000714245
	%Disk Read Time	D1572	0.396698
	%Disk Write Time	D1574	2.16555466666666
Processor	%Interrupt Time	U1616	1.559920773
	% Idle Time	U1619	98.27500867
	%Processor Time	U1731	2.504948441
Memory	Page Faults/sec	M1746	1655.02885723717
	Available Bytes	M1747	1367638016
	Pages/sec	M1754	20.5277281622
Network Interface	Packets/sec	NT1833	93.93193628
	Packets Received/sec	NT1834	1.020999307
	Packets Sent/sec	NT1835	1.020999307
	Current Bandwidth	NT1836	100000000

4.3 Designing Classifier

Although understanding the data distribution is very helpful for choosing the best classifier, it is very difficult to understand the different data distribution. For small dimensional data set, it can be easy to understand

by plotting the data, but it is not simple for very large scale data set. In this work, in order to know which classifier is the most suitable one for our synthesis dataset many classifiers are tried heuristically with our data set. Some classifiers are sensitive to very large or small data set. Since performance monitor recorded thousands of performance counters contained thousands of features, it is important to choose the best classifier correctly.

4.4 Evaluating Classifier

In this subsection we present the Machine Learning evaluation process conducted. The two main goals of this process are Model selection and Model Accessment.
Model Selection: Comparing different Machine Learning algorithms in order to choose the (approximately) best one.
Model Assessment: Having chosen a final algorithm, estimate its generalization error on new data.

According to the goal (model selection or assessment), different tasks could be conducted. However, in a rich-data scenario, the best approach (if there is enough data) to both model selection and model assessment goals is to divide the data set into three disjoint parts: Training data set needed to build (or fit) the models. Validation data set (or development data test set) used to estimate the test error for model selection. Test data set used for assessment of the generalization error of the finally chosen model.

The main goal is to conduct an evaluation of different Machine Learning algorithms to chose the best algorithm found during current chapter to predict the time to crash and trigger the rejuvenation action in real environments. Due to this goal, we have conducted a model selection. Model selection process requires that ML algorithms given are trained with the training data set, and later compared according to the estimated test error using a completely different data set, called validation data set or development data test set. The training and validation process is presented in Figure 3 and described in detail below.

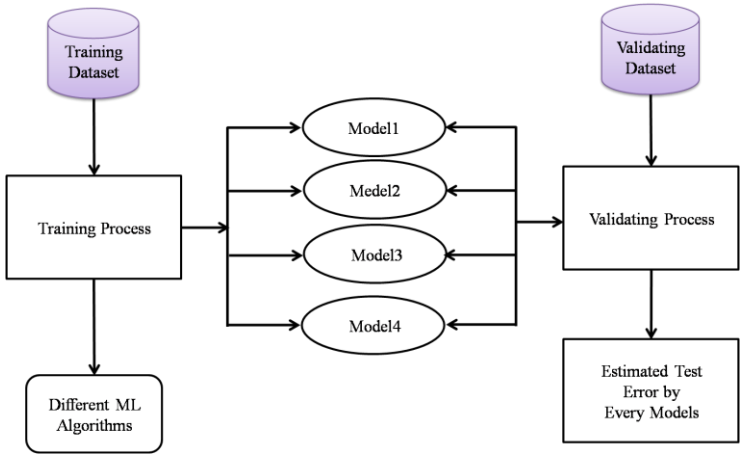

Figure 3. Machine Learning Model Selection Process

5. Analysis of Machine Learning Classifiers

5.1 J48 Decision Tree

Decision trees are a classic way to represent information from a machine learning algorithm, and offer a fast and powerful way to express structures in data. The overall concept is to gradually generalize a decision tree until it gains a balance of flexibility and accuracy. J48 is a decision classifier and also an optimized implementation of C4.5. The C4.5 technique is one of the decision tree families that can produce both decision tree and rule-sets and construct a tree. J48 classifier is among the most popular and powerful decision tree classifiers. J48 are the improved versions of C4.5 algorithms. The J48 algorithm gives several options related to tree pruning. Pruning produces fewer, more easily interpreted results. The basic algorithm recursively classifies until each leaf is pure, meaning that the data has been categorized as close to perfectly as possible. This process ensures maximum accuracy on the training data, but it may create excessive rules that only describe particular idiosyncrasies of that data.

In this experiment, the classification rules are generated by using j48 classifier for the training dataset. The J48 classifier produced the analysis of the training dataset and the classification rules.

- p433 = 974848.0: Low (6.0)
- p433 = 8.0: Normal (366.0)
- p433 = 3.0: High (360.0)

5.2 JRip Classifier

JRip (RIPPER) is one of the basic and most popular algorithms. Classes are examined in increasing size and an initial set of rules for the class is generated using incremental reduced error JRip (RIPPER) proceeds by treating all the examples of a particular judgment in the training data as a class, and finding a set of rules that cover all the members of that class. Thereafter it proceeds to the next class and does

the same, repeating this until all classes have been covered. JRIP classifier generates some useful rules;

- p4 >= 208486400: Low (336.0/0.0)
- Normal (366.0/0.0)
- p257 <= 798.679436:High (360.0/0.0)

5.3 OneR Classifier

OneR is a simple and a very effective classification algorithm frequently used in machine learning applications. ven though OneR is difficult to be improved further due to its simplicity, it can be enhanced by providing better methods for handling some of the exceptions. OneR, short for "One Rule", is a simple classification algorithm that generates a one-level decision tree. OneR is able to infer typically simple, yet accurate, classification rules from a set of instances. Comprehensive studies of OneR's performance have shown it produces rules only slightly less accurate than state-of-the-art learning schemes while producing rules that are simple for humans to interpret. The OneR algorithm creates one rule for each attribute in the training data, then selects the rule with the smallest error rate as its 'one rule'. To create a rule for an attribute, the most frequent class for each attribute value must be determined. The most frequent class is simply the class that appears most often for that attribute value. A rule is simply a set of attribute values bound to their majority class; one such binding for each attribute value of the attribute the rule is based on.

- p257< 432.1987029:Low
- p257>=3.740097533971775E7:Normal
- p257< 3.740097533971775E7:High

5.4 AdaBoostM1 Classifier

Boosting has been a very successful idea for the two-class classification problem. In going from two-class to multi-class classification, most algorithms have been restricted to reducing the multi-class classification problem to multiple two-class problems. AdaBoost.M1 is the most direct extension of the original AdaBoost algorithm to the multiclass case. AdaBoost.M1 only requires the performance of each weak classifier be better than random guessing rather than ½.

- p1164 = 9.0 : High
- p1164 != 9.0 : Low
- p1164 is missing : High

6. Experimental Results of Classifiers

We conduct our experiment on our synthesis performance counter data set.. Our synthesis data set includes 500 up to 1500 records and 150 to 926 features. A summary of some of the properties of these datasets is given in Table 2. For these datasets with no provided test set, we reran each algorithm 10 times (since some of the algorithms are randomized), and averaged the results. We used 10-fold cross validation, and averaged the results over 10 runs of each algorithm on our synthesis dataset. Different classifiers are J48, JRip, OneR and AdaBoostM1. In our experiment, performance of classifiers is evaluated over execution time (second) and percentage of correctly classified instances. In this experiment, all criteria are not different so far. Classifier speed is very important for real time implementation.

Inspection of Figure 4 shows that OneR gave the lowest classification speed and same correctly classify instances. As we can see in Figure 5,all classifiers have the same values in correctly classified instant.

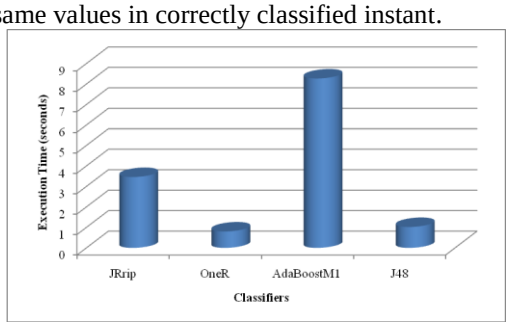

Figure 4. Experimentation on Classifier Speed

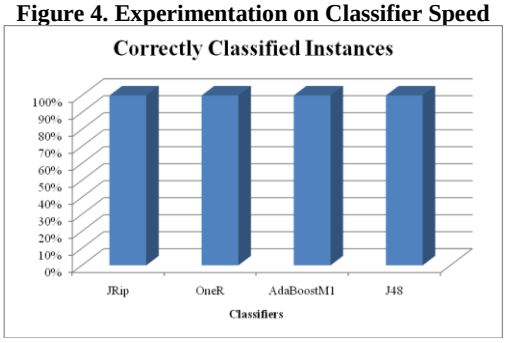

Figure 5. Experimentation on Correctly Classified Instances

7. Conclusion

In this paper, we have practically evaluated performance of various classifiers on a synthetically generated dataset. We presented how selected classifier (Random Tree) is match with our synthesis dataset and how can be useful for server overload detection. In detection method, performance counter values are effective to improve the accuracy of detection mechanism. According to experimental results, the accuracy of selected classifier is obviously increased over number of features .In future work, we will combine random Tree with overload prevention method such as admission control and implement in our overload control mechanism.

References

[1] Saeed Sharifian,Seyed Ahmad Motamedi,Mohammad Kazem Akbari, "Estimation-Based Load-Balancing with Admission Control For Cluster Web Servers," *ETRI Journa*, vol-31. No.2,April 2009.

[2] Lahcene AID,Malik LOUDINI,Walid-Khaled HIDOUCI, "An Admission Control Mechanism For Web Servers Using

Neural Network," *International Journal of Computer Application(0975-8887),* vol. 15-No.5,Feb 2011.
[3] Kimihiro Mizutani,Izumi Koyanagi,Takuji Tachibana, "Dyanmic Session Management Based on Reinforcement Learning in Virtual Server Environment ," *Proceeding of the International MultiConference of Engineers and Computer Scientists, Vols II,IMECS*.March 17-19,2010,Hong Kong.
[4] Cor-Paul Bezemer,Veronika Cheplaygina,Andy Zaidman, "Using Pattern Recognition Techniques for Server Overload Detection,"Report TUD-SERG-2011-009.
[5] Xiangping Chen,Huamin Chen,Prasant Mohapatra, "ACES:An Efficient Admission Control Scheme For QoS-aware Web Servers," Computer Communications 26(2003), 1581-1593.
[6] Andy Liaw,Matthew Wiener, "Classification and Regression by RandomForest," *ISSN(1609-3631),* vol. 2/3,December 2002.
[7] Breiman L, " Random Forests," *Journal of Machine Learning,* vol. 45-No.1,pp.5-32,Springer,October 2001.
[8] Frederick Livingston, "Implementation of Breiman's Random Forest Machine Learning Algorithm,"*ECE591Q Machine Learning Jpurnal Paper*,Fall 2005 .
[9] Xiangping Chen,Huamin Chen,Prasant Mohapatra, "An Admission Control Scheme For Predictable Server Response Time For Web Servers," *ACM 1-58113-348-0/01/0005,WWW10*,May 1-5,2001,Hong Kong.
[10] Y.-C. Ko, S.-K. Park, C.-Y. Chun, H.-W. Lee and C.-H. Cho, "An adaptive QoS provisioning distributed call admission control using fuzzy logic control," *in Proc. of IEEE ICC 2001*, vol. 2, pp. 356-360, Helsinki, Finland, 2001.
[11] Nicolas Poggi,Toni Moreno,Josep Lluis Berral ,Ricard Gavalda, Jordi Torres,"Self-adaptive utility-based session management", *Computer Networks 53(2009) 1712-1721*,© 2008 Elsevier B.V.
[12] Gerald Tesauro, Nicholas K.Jong,"A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation",In the fifth IEEE International conference on Autonomic Computing,ICAC 06,Dublin, Ireland,June 2006.
[13] Apache Software Foundation.http://www.apache.org
[14] Web server survey by Netcraft, http://news.netcraft.com, May 2011.