

Predicting Machine Availabilities in Desktop Pools

Artur Andrzejak
Computer Science Research
Zuse-Institute Berlin
Berlin, Germany
andrzejak@zib.de

Patricio Domingues
ESTG
Polytechnic Institute of Leiria
Leiria, Portugal
patricio@estg.ipleiria.pt

Luis Silva
CISUC
University of Coimbra
Coimbra, Portugal
luis@dei.uc.pt

Abstract—This paper describes a study of predicting machine availabilities and user presence in a pool of desktop computers. The study is based on historical traces collected from 32 machines, and shows that robust prediction accuracy can be achieved even in this highly volatile environment. The employed methods include a multitude of classification methods known from data mining, such as Bayesian methods and Support Vector Machines. Further contribution is a time series framework used in the study which automates correlations search and attribute selection, and allows for easy reconfiguration and efficient prediction. The results illustrate the utility of prediction techniques in highly dynamic computing environments. Potential applications for proactive management of desktop pools are discussed.

Keywords - proactive management, data mining, desktop pool management, grid environments, resource inventory and allocation

I. INTRODUCTION

Analysis of the network, processor or storage availability, current capacity of resources and other characteristics is an essential constituent in the process of systems management. Information derived from these statistics serve as a basis for detection of anomalies or bad configurations, long and short-term capacity planning, and scheduling in resource sharing scenarios, to name a few [2]. A further step is prediction and in general modeling of such characteristics. They enable the transition from reactive to *proactive* management. The advantages of the latter include lower or zero adjustment latency, elimination of system overload or network congestion, downtime reduction, increased scheduling efficiency, and others.

Prediction-based techniques work only if demand characteristics observed in the past are likely to prevail in the future. The nature of the applications in a system must be considered as a deciding factor in this assumption. While applications employed in business or scientific domains are especially likely to exhibit repetitive and thus predictable behavior, desktop pool environments are generally considered less suitable for prediction. In this study we demonstrate that even in such environments important characteristics such as availability and user login behavior can be predicted with good accuracy and efficiently.

Pools of desktop computers are omnipresent in private and public institutions. They pose a challenge for efficient management on one hand, while on the other hand they provide an opportunity of tapping a supply of cheap and currently

underutilized resources. Prediction techniques provide support for the latter endeavor in the following cases:

- management of desktop pools: applications here include proactive power management, discovery of network or system outages, availability scheduling
- desktop Grids: utilizing desktop computers for Grid-like computations [1], [7] can greatly benefit from prediction through proactive resource scheduling, for dealing with failures and minimizing intrusiveness of the computation [5]
- file replica management: in peer-to-peer systems like FARSITE [4] the knowledge of (short-term) component behavior can greatly increase the statistical guarantees of data availability
- security in desktop pools: differences between the modeled and observed network or system load can indicate an intrusion, prediction provides here an early detection mechanism.

For our study presented in this paper we used classification methods known from data mining such as Naive Bayes, decision trees or Support Vector Machines. While these algorithms are well studied, their successful application requires a lot of manual work in data preparation, attribute selection, and result evaluation. Our contributions target the streamlining and automating of these processes, and include methods and software for:

- automated discovery and exploitation of correlations between traces
- computationally efficient selection of predictive attributes
- walk-forward evaluation techniques for higher prediction adaptability and elimination of overfitting
- effortless switching between a variety of classification algorithms depending on accuracy vs. efficiency demands.

The case study has been performed on traces from 32 desktop computers running Windows from two university classrooms with 16 machines each. Various metrics of each machine (availability, CPU idleness ratio, memory load, network traffic rates, and others) have been sampled every two minutes. In this way we obtained two traces: one of 39 consecutive days, and another for 17 consecutive days. As the prediction targets we selected machine availability (switched on/off) and user presence (logged on/off) 30 minutes into the

future. The comparison of the mean square error (mse) of the prediction with the signal variance show good accuracy of the prediction in general (we obtained better values for the first trace than for the second). In the whole study there is only one case where mse is higher than the variance, which shows that here prediction failed. Also we observed significant accuracy differences among the five used classification algorithms, with Support Vector Machines (SMO algorithm) performing best.

II. THE SOFTWARE FRAMEWORK

In this section we briefly describe the two frameworks employed in our study: Distributed Data Collector (DDC) and OpenSeries.

A. Distributed Data Collector

DDC is a framework to automate repetitive executions of console applications (probes) over a set of networked Windows personal computers [3]. DDC schedules the periodic execution of software probes in a given set of machines. The execution of probes is carried out remotely, that is, the probe binary is executed at the remote machine, requiring only the appropriate access credentials. In this way, no software needs to be installed at remote machines. For the purpose of collecting the traces DDC was fitted with an appropriate probe that collects for every execution at a remote machine, the current timestamp, CPU idleness percentage, memory load, sent and received network traffic rates and machine uptime, and other metrics. Collected data are appended to the trace file corresponding to the remote machine.

B. OpenSeries

OpenSeries is an extensible framework which provides existing and new algorithms for modeling, prediction and analysis of timeseries. The framework is a collection of Java packages which provide data structures, persistency layer, and the deployment engine. The latter allows for sophisticated off-line modeling, and is currently being extended with on-line (real time) prediction capabilities. OpenSeries interfaces with Weka [8], a large library for data mining algorithms. In addition to all the algorithms provided by Weka, users of OpenSeries can use multivariate ARIMA algorithms known from econometrics.

III. PREDICTION TECHNIQUES

In our study we have employed classification algorithms provided by Weka, not using the ARIMA methods. This was dictated by the binary nature of the targets and due to better performance of the former algorithms. As an essential step of the preprocessing, numeric attributes relevant for prediction have been selected.

A. Classifiers

Formally, a classifier is a function $f: X \rightarrow Y$ which assigns to a tuple $t \in X$ of attribute values a label $\lambda \in Y$ [8]. In the training phase, f is presented a set of examples (t, λ) (tuples with the correct labels) from which it attempts to build a model of relationships between tuple values and labels. During the

prediction stage, a tuple t with an unknown label is given, and then the label $f(t)$ is computed.

In the setting of this study, attribute values were past sampled values, or some functions thereof, such as moving averages. It is important to note that these values may stem from other machines than the one for which the prediction is performed - this fact allows for exploiting inter-machine correlations.

The set of labels can correspond to discretized target signal values, e.g. different network activity levels for prediction purposes, or to alert-labels (malfunction, likely overload etc.) for anomaly detection. In this study, we have used the values 0 (machine not available / no user) and 1 (machine available / user logged in) as the target values.

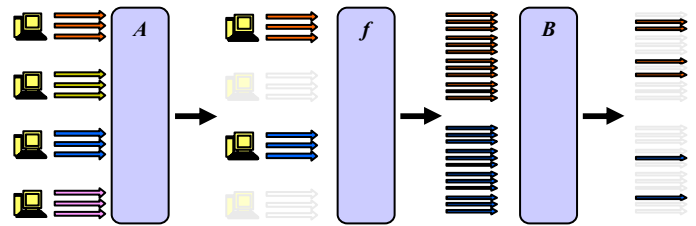


Figure 1: Attribute selection process

B. Attribute selection

An essential step of the model building process is the selection of attributes. Due to the phenomenon called "curse of dimensionality", too many attributes per tuple (in relation to the number of examples) can easily lead to overfitting. We have developed a two-stage process which first selects the relevant (correlated) machines (phase A), then computes a pool of functions on traces of these machines (phase f), and finally selects the final attributes from the pool of functions (phase B), see Figure 1. Since phase A is essentially a trace correlation analysis with low running time, we can specify a large set of machines (on the order of 100) as a potential input. As a by-product of this phase a ranked list of resources potentially influencing the behavior of the target is produced. In phase B we apply the standard mechanism which prefers attributes with high correlation to the target yet low redundancy [6].

C. Walk-forward evaluation

The prediction testing has been done in a walk-forward mode. Here a fitting interval of size F is used for training of the models and searching for best parameters (in this study, always the standard parameter values have been used). An adjacent test interval of size T is used for the out-of-sample evaluation of the model. Subsequently, both intervals are moved forward by F , and the process is repeated. This procedure allows for change-adaptive model selection yet avoids any overfitting problems.

IV. SYSTEMS UNDER STUDY

For the purpose of the study we collected traces from two classrooms with 16 machines each, thus totaling 32 machines.

Although the classrooms are primarily devoted to classes, during off-classes any student from the academic institution can access the machines for performing practical assignments, communicate through e-mail and browse the web. It is important to note that no policy related to the powering off of machines exists. In fact, users are advised but not obliged to power off machines when they stop work. On weekdays classrooms remain open 20 hours per day, closing from 4 am to 8 am. On Saturdays classrooms open at 8 am and close at 9 pm until the following Monday.

Trace A represents 39 consecutive days (from 25 November 2004 to 3 January 2005) and contains 27000 samples, while trace B corresponds to 17 consecutive days (from 31 January 2005 to 16 February 2005) having 12700 samples. Contrary to trace A that was collected on a period with classes (except for the last ten days which corresponded to Christmas holidays), trace B corresponds to an exam-period, without classes. Thus trace B depicts the off-classes activity of students, while trace A combines classes with off-classes activities.

While each sample contains a lot of machine metrics, we used only CPU idleness percentage, percentage of free virtual memory, machine availability (switched on or off), and user presence indicator (logged on or off). Only the last two metrics have been used as targets (other showed similar accuracy results).

V. EXPERIMENTS AND THEIR EVALUATION

A. Experimental setup

As the “row input data” in the attribute selection phase we have used all four metrics described above. However, in addition to these data we computed functions thereof, including simple moving averages of lengths from 5 to 60 samples, and calendar functions such as hour of the day, day of the week etc. After the final, second phase of the attribute selection we have taken the 10 most relevant attributes as input attributes for the prediction. We note that due to an error in a discretization filter of the Weka 3.4 library, the attribute selection for two machines in trace B could not be completed, and so we used only 30 machines for study in trace B.

Each of the two targets for each of traces A and B has been evaluated using five classification algorithms:

- Naive Bayes
- complement class Naive Bayes classifier
- John Platt's sequential minimal optimization algorithm for training a support vector classifier (SMO)
- k-nearest neighbors classifier (IBk)
- C4.5 decision tree classifier (J48).

For all algorithms the standard parameters provided in Weka 3.4 have been used. In nearly all cases, the SMO classifier produced the smallest mean squared error, and is thus discussed in the following.

The evaluation has been performed in walk-forward mode described above, with 1 week as the fitting time interval F , and 3 days as the test time interval. The prediction has been done for each of the samples with 30 minutes lead time (i.e. 30 minutes into the future). Other lead times of 15, 60 and 120 minutes have been also tested, and provided similar accuracy results.

As the evaluation criteria the mean squared error (mse) of the prediction has been used, i.e. the mean of squares of the differences between prediction and true value. We compare the mse value against the variance of the original signal value. Note that if the prediction values would be random, those two values are expected to be similar. Moreover, the mse emphasizes large deviations of the prediction, in a sense providing also a measure for the maximum encountered error.

B. Results and discussion

In almost all cases, the selected attributes stemmed from the same machine as the prediction target, i.e. no inter-machine correlations have been used in predictions. This does not imply non-existence of correlations between the machines - just their predictive significance was smaller than the target's own historical data. This phenomenon can be attributed to the fact that the machines have been used independently by the users, and also no general power on/off policy has been imposed.

Figure 2 shows the comparison of the mse and signal variance. The former is in all cases smaller than the variance, which points to high quality of the results. Except for the case “trace B/user presence”, the mse is almost below 0.1, while the variance is almost always above this value, partially soaring to 1.0. The relatively bad result for “trace B/user presence” can be explained by low variance values for many machines (those machines seem to have been not used at all) which cause the mse to appear relatively large. However, even in this scenario only for one machine the mse is higher than the variance (which is also the only case of prediction failure in the whole study).

The quantitative evidence how representative is our study has still to be delivered. This requires comparisons with results from similar and dissimilar environments. However, the following items point that our approach generalizes to other university and office desktop pools:

- the non-existence of a power off/on policy in our case makes environments with a policy even easier to predict
- the predictions worked for both the traces A and B despite of the quite distinctive usage characteristics
- the average resource consumption of the machines (around 4%) is similar to the findings from office environments [3].

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the results of the study on predicting machine availability and user presence in a pool of desktop computers. The results show that even in such a volatile environment robust prediction results can be achieved.

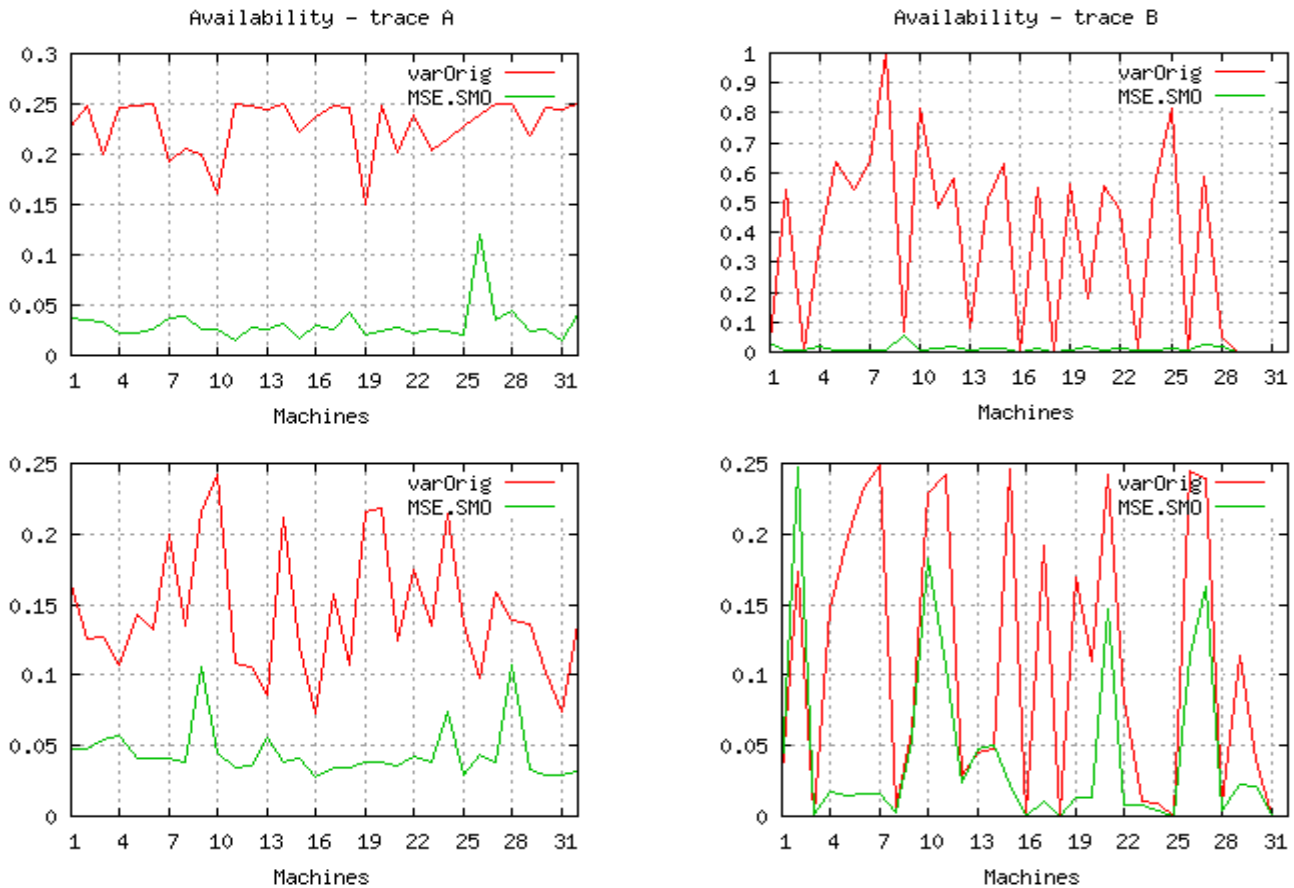


Figure 2: Mean squared error of the predictions vs. signal variance (left: trace A, right: trace B; upper row: availability, lower row: user presence)

This is particularly important as availability prediction is essential for many applications in this domain (such as desktop Grids and replica storage infrastructures). Also the management of such pools can greatly benefit from prediction, through better power management, availability scheduling, and faster intrusion detection, to name a few.

We have also described methods and frameworks for automated data collection and preprocessing, including automated selection of the predictive attributes. Our future work will focus on extending the OpenSeries framework for online (real time) prediction. Subsequent steps will target the above-described applications of prediction in desktop pools, including desktop Grids and early intrusion detection. Also evaluation of the techniques with larger trace data from different environments (office pools, file-sharing peers, business data centers) is planned.

ACKNOWLEDGMENT

Artur Andrzejak thanks Mehmet Ceyran and Ulf Hermann for contributions to the implementation work.

REFERENCES

- [1] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An experiment in public-resource computing", *Communications of the ACM*, 2002.
- [2] A. Andrzejak and M. Ceyran, "Characterizing and Predicting Resource Demand by Periodicity Mining", *Journal of Network and System Management*, vol. 13, no. 2, June 2005.
- [3] P. Domingues, P. Marques, and L. Silva, "Resource Usage of Windows Computer Laboratories", *Int. Conf. Parallel Processing (ICPP 2005), Workshop PEN-PCGCS, Oslo, Norway, 2005*.
- [4] J. R. Douceur and R. P. Wattenhofer, "Modeling Replica Placement in a Distributed File System: Narrowing the Gap between Analysis and Simulation", *Proc. 9th Annual European Symposium on Algorithms*, pp. 356-367, 2001.
- [5] A. Gupta, B. Lin, and P. A. Dinda, "Measuring and understanding user comfort with resource borrowing", *Proc. 13th IEEE Int. Symposium on High Performance Distributed Computing, Honolulu, Hawaii USA, 2004*.
- [6] M. A. Hall and L. A. Smith, "Practical feature subset selection for machine learning", *Proc. 21st Australasian Computer Science Conference, Perth, Australia*, pp. 181-191, 1998.
- [7] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien, "Characterizing and evaluating desktop grids: an empirical study", *Proc. 18th Int. Parallel and Distributed Processing Symposium, 2004*.
- [8] I. H. Witten and F. Eibe, *Data Mining - Practical machine learning tools with Java implementations*, Morgan Kaufmann Publishers, San Francisco, 2000.