Using Multi-System Monitoring Time Series to Predict Performance Events

Andreas Schörgenhumer[1], Mario Kahlhofer[1], Peter Chalupar[1],
Hanspeter Mössenböck[2], Paul Grünbacher[3]
{firstname.lastname}@jku.at
[1] Christian Doppler Laboratory MEVSS, Johannes Kepler University Linz, Austria
[2] Institute for System Software, Johannes Kepler University Linz, Austria
[3] Institute for Software Systems Engineering, Johannes Kepler University Linz, Austria

Abstract

The prediction of failures and other mission-critical events plays an important role in operating today's software systems and has drawn the attention of many researchers. Event prediction is particularly challenging if multiple systems are involved. In this paper, we thus present an event prediction model which utilizes time series monitoring data from multiple software systems to predict performance events. Our approach incorporates a comprehensive, multi-system data preprocessing framework for creating various feature vector sets, which we then use to train a random forest classifier to evaluate our multi-system event prediction. Our preliminary evaluation based on data from monitoring 250 systems over a period of 20 days shows promising results.

1 Introduction

Online failure prediction is an essential part of proactive fault management in systems [2]. In online failure prediction, dynamic data (e.g., monitoring time series data) of a system is used to predict certain events in the near future, so administrators can be notified in advance to take precautionary actions. A failure may not necessarily indicate a total system failure, but it can also be a violation of quality properties (e.g., performance anomalies), to which we refer simply as event prediction as a more general term.

Many researchers have addressed the issue of event prediction using different prediction approaches on various data sources. For instance, some utilized the information stored in continuous log files [1, 4, 6], others use system monitoring data like CPU or memory metrics [3, 7, 8]. All of this research yielded promising results, however, did not consider the case of analyzing data from multiple systems [9]. The main goal of extending event prediction to multiple systems is to learn from all systems and then make predictions for individual systems. This is especially interesting for small systems for which insufficient data is available to create accurate prediction models, either due to limited dynamic data or rare events. To the best of our knowledge, no previous work on this particular topic exists.

In this paper, we will investigate whether combining monitoring data and events from multiple systems is feasible and yields promising results. Our motivation is that low-level infrastructure measurements such as CPU, memory, disk or network metrics often indicate problems in one system, which may also be transferable to other systems. For example, high CPU loads or certain disk access patterns might lead to problems regardless of what the actual business logic of the inspected systems is. For this purpose, we use anonymized monitoring data from the infrastructure of an industry partner. This monitoring data comprises 34 different time series metrics for hosts, disks and network interfaces, based on which we create fully customizable feature vector sets to predict *service slowdown* performance events.

We claim the following contributions:

(i) We present a novel, highly customizable data framework for preprocessing multi-system monitoring data. The framework is an integral part of our event prediction approach.

(ii) We provide a preliminary evaluation of our event prediction on data of 250 monitored systems. We show that our multi-system event prediction yields promising results to be further pursued in future work.

2 Data Processing

Before we can predict events, we have to appropriately process the monitoring data which our industry partner provides.

2.1 Monitoring Data

We define a system as an independent composition of hardware and software components, operated by some service provider, e.g., a web shop running on a certain hardware setup.

Each monitored system can be represented as an entity-relationship diagram (ERD), which is shown in Figure 1. A host is a physical or virtual computing

Hosts	Disks	Network Interfaces
H-01: CPU idle %	D-01: Disk available	N-01: Bytes received
H-02: CPU system %	D-02: Disk used	N-02: Bytes sent
H-03: CPU load %	D-03: Disk read bytes	N-03: Received packets
H-04: Memory available %	D-04: Disk write bytes	N-04: Received dropped
H-05: Memory available	D-05: Disk read operations	N-05: Sent packets
H-06: Page Faults per second	D-06: Disk write operations	N-06: Sent dropped
H-07: CPU user %	D-07: Disk queue length	N-07: Received errors
H-08: CPU IO wait %	D-08: Disk util time	N-08: Sent errors
H-09: Memory used	D-09: Disk read time	N-09: Received utilization
H-10: SWAP available	D-10: Disk write time	N-10: Sent utilization
H-11: SWAP used	D-11: Disk inodes total	
	D-12: Disk inodes available %	
	D-13: Disk free space %	

Table 1: Infrastructure metrics for entity types. Corresponding units are %, bytes, milliseconds and counts.

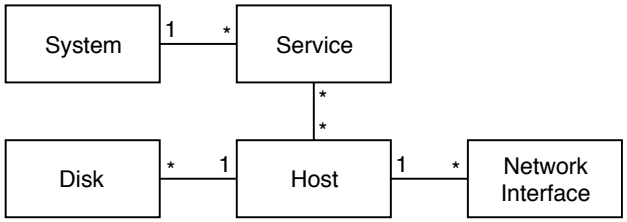

Figure 1: ERD of a monitored system.

unit to which zero or more disks and network interfaces can be connected. Time series metrics are collected for these three entities. Services represent the business logic that is carried out on one or more hosts, which, in turn, can host multiple services. The performance events which we want to predict occur at these services and are called *service slowdowns*. A service slowdown is created by the monitoring infrastructure of our industry partner if the average response time of a service exceeds a threshold compared to its baseline.

Table 1 lists the 34 different metrics which are collected at host level (H), disk level (D) and network interface level (N). All these measurements are time series recorded in 1-minute resolution.

2.2 Preprocessing Framework

We designed and implemented a comprehensive framework for preprocessing the time series data from multiple systems for later use with machine learning algorithms. Many parts and processing steps of the framework can be arbitrarily configured, which facilitates creating various labeled feature vector sets for evaluation (positive samples: event, negative samples: non-event).

We list some of the essential configuration options:

Metrics. We can define which metrics the feature vectors should consist of (cf. Table 1). This forms the basis for all following configuration settings. Example: {H-03, D-02}.

Observation windows. For each metric, we can create arbitrary observation windows (OW) that specify how many data points (= how many minutes) of the metric should be used, given a defined starting timestamp (e.g., the time of an event). Example: {H-03: 30}, start=x. This means that we want a 30-minute OW for metric H-03, ranging from $x - 30$ to x.

Aggregation functions. Our framework provides various common statistical functions (min, max, mean, median, etc.) which can be used to aggregate OWs if the user does not want to keep the raw time series values. Multiple aggregation functions can again be specified for each OW of a metric. Example: {H-03: 30 $\rightarrow$ [min, max]}. This means that the 30 minutes of data are aggregated into two values: minimum and maximum.

Combination functions. The available aggregation functions can also be used to combine data of multiple, similar entities. For example, a host might have two disks. If we selected a disk metric, we would get datasets for the first disk and additionally for the second disk. Since feature vectors must be identical in length, we have to combine the two datasets, e.g., via averaging.

The final output of our framework is a ready-to-use CSV-file containing the labeled feature vectors.

3 Evaluation

We evaluated our approach by training a random forest classifier on the binary target of a *service slowdown* event based on the monitoring data of 250 systems for a total of 20 days. Slowdown events were observed in all the systems. For training the random forest, we used the first 14-days of data and then evaluated the resulting models on the test set of the following 6 days of contiguous, unseen time series. For each of the 34 time series, data points prior to an event were aggregated per OW by four different functions: arithmetic mean, standard deviation, min and max, yielding a total of $34 \cdot 4 = 136$ feature vector entries

	Metric	Window Size [minutes] 5	10	15	30	60
non-event	Accuracy	**0.81**	0.80	0.79	0.75	0.74
	Recall	**0.81**	0.78	0.75	0.66	0.68
	Precision	**0.82**	**0.82**	0.81	0.80	0.77
	FPR	0.18	0.17	0.17	0.16	0.21
	F1 score	**0.81**	0.80	0.78	0.72	0.72
event	Accuracy	0.56	0.56	0.55	0.56	0.57
	Recall	0.48	0.51	0.44	0.49	0.47
	Precision	0.57	0.56	0.56	0.57	0.59
	FPR	0.37	0.39	0.35	0.37	0.33
	F1 score	0.52	0.53	0.49	0.53	0.52
all	Accuracy	0.75	0.75	0.72	0.74	0.72
	Recall	0.65	0.65	0.63	0.64	0.59
	Precision	0.81	0.81	0.78	0.79	0.80
	FPR	**0.15**	0.16	0.18	0.17	0.15
	F1 score	0.72	0.72	0.70	0.71	0.68

Table 2: Grouped test set results (bold = best).

per sample. In case of similar entities (e.g., multiple disks), the arithmetic mean was used as the combination function. Since this is only a preliminary evaluation to check whether our multi-system event prediction yields promising results in the first place, we decided to use a lead time [2] of 0, meaning that we predict events at the very moment and not into the future yet.

Negative samples were drawn in three ways: only from hosts where no events occurred *(non-event)*, only from hosts where events did occur *(event)* and a mixture of both *(all)*. We took samples from random timestamps that were at least 30 minutes away from event occurrences. The results for these three sampling options are displayed in Table 2, which shows commonly used evaluation metrics [5, pp. 79 and 127].

The preliminary results show that the *non-event* option for drawing negative samples yielded the best outcome, and the 5 minute observation window was generally preferable compared to larger window sizes.

4 Conclusion

In this paper, we briefly introduced our multi-system event prediction approach to predict *service slowdown* performance events. It operates on time series data of a multi-system monitoring environment. The main idea behind the multi-system setup is to enable prediction for systems where too little monitoring data is available or when trying to predict rare events, as we can utilize data from multiple systems and therefore build a large enough dataset for event prediction. Our approach includes a powerful data preprocessing framework which eases the creation of different feature vector sets needed for the evaluation.

Based on monitoring data of 250 systems over a period of 20 days, we performed a preliminary evaluation. We trained a random forest classifier on different datasets from the first 14 days and tested the prediction model on the following 6-days. Results with accuracies up to 81% indicate the applicability of our multi-system event prediction approach.

In future work, we will focus on an in-depth evaluation, comparing different classifiers as well as various data configurations using our framework. Furthermore, we are interested in predicting events for completely new systems based on historic training data from different systems.

Acknowledgements

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

[1] F. Salfner and S. Tschirpke. "Error Log Processing for Accurate Failure Prediction". In: *Proc. of the 1st USENIX Workshop on the Analysis of System Logs*. 2008.

[2] F. Salfner, M. Lenk, and M. Malek. "A Survey of Online Failure Prediction Methods". In: *ACM Comput. Surv.* 42.3 (2010).

[3] J. Alonso, L. A. Belanche Muñoz, and D. Avresky. "Predicting software anomalies using machine learning techniques". In: *Proc. of the Int'l. Symposium on Network Computing and Applications*. 2011.

[4] L. Yu et al. "Practical online failure prediction for Blue Gene/P: Period-based vs event-driven". In: *Proc. of the 41st Int'l. Conf. on Dependable Systems and Networks Workshops*. 2011.

[5] C. O'Neil and R. Schutt. *Doing Data Science*. Sebastopol, CA: O'Reilly Media Inc., 2014.

[6] T. Pitakrat et al. "A Framework for System Event Classification and Prediction by Means of Machine Learning". In: *Proc. of the 8th Int'l. Conf. on Performance Evaluation Methodologies and Tools*. 2014.

[7] X. Zhang et al. "TaskInsight: A Fine-Grained Performance Anomaly Detection and Problem Locating System". In: *Proc. of the 9th Int'l. Conf. on Cloud Computing*. 2016.

[8] T. Pitakrat et al. "Hora: Architecture-aware online failure prediction". In: *Journal of Systems and Software* 137 (2018).

[9] A. Schörgenhumer et al. "Using Crash Frequency Analysis to Identify Error-prone Software Technologies in Multi-System Monitoring". In: *18th IEEE International Conference on Software Quality, Reliability, and Security*. Lisbon, Portugal, 2018.