Time Series Analysis for Monitoring

Stefanie Deckers

February 8, 2016

Abstract

- First some fourier and wavelet approaches, just to figure out how data could be processed generally. What I did: Had a set of metrics, applied fourier or wavelet and tried to cluster. I just could evaluate ”manually” by having a look at the different clusters. At the end no approach was used because of two reasons: the data must be processed in real time. And it was decided to first focus single metrics.
- We needed a kind of anomaly / change detection. An example scenario is requests to a web service. If there are suddenly much less requests / second the user should be warned by the system. So a simple approach was applied by finding kind of confidence intervals in which the number of request / second should lie. If to much values are monitored that are below or above this interval, it is considered a change. This approach has the disadvantage that it does not take any seasonality into account.
- Moreover Recursive Least Squares regression was applied to disk data, to make better predictions. At this point predictions were just made for the next hour or something. Longterm predictions were not possible because it was not able to access historical data. Historical data here is data that is older than two hours. So RLS would be a good approach for long term predictions. The disadvantage was, that there is no possibility to take only a window of x hours / samples into account. So, if the disk is getting freed, the old values would still count to much and it would take a while until prediction would be reliable enough. It was tried whether the forgetting factor could be used, but that always worsens predictive results of the test dataset, because prediction is too sensitive to small temporary changes in the metric.
- It was make some effort to anomaly / change detection. The behaviour of a metric should be learned and taken into account when deciding whether there is a change / anomaly in the data.

  - gp - test set: cc blog requests. - first kernel that contains periodic component to find periodicity - works with right hyperparameters. but learning hyperparameters only work with good initialization. - initialization techniques were tried out (finding periodicity, standard deviaiton, ...) - it works for cc blog, but not for other data

(e.g. memory free metric) - then moved to weka because of final implementation - and tried to take weekday and second of day into account and use rbf kernel - works! - Problems to solve: take longterm trend into account, can we update gp on a real time basis (currently working on that), does it work without hyperparameter learning?
- maybe add correlation analysis - maybe write a short chapter about analysing issues and adjusting rules / knowledge

1 Introduction

This is a preliminary report for my internship project "Data Analysis for an intelligent APM tool". The main purposes is to give a rough description of the data that will be analysed and some goals that want to be achieved.

In section 1 a short outline about the background of the project is given. Section 2 and 3 the data and the analysis goals are described, respectivley.

2 Background

The internship is done at the start-up company Instana Gmbh, where they are developing an application performance monitoring (APM) tool. Their product, called Instana, is monitoring metrics of different systems like database systems, application servers or metrics on operation system level. A more detailed description of the monitored systems and data will be given in the next chapter.

The actual goal of the system is not just monitoring and presenting the metrics to the users, but to analyse them and make conclustion about the current system health. This is done by detecting issues in the metrics. If an issue is detected Instana makes suggestions on how the user may solve this issue. For example one issue is raised when a CPU steal time exeeds a certain threshold too often. This issue suggests the user to allocate more CPU to this virtual machine.

3 Data

In general the obtaned data are real time data streams of time series. For the last two minutes secondly data points are available, for older data the means of tumbling windows of 5 seconds are stored, further called rollups.

The data depends on the monitored metric. Following some of the monitored systems and their corresponding metrics are presented.

3.1 Host

The most basic system is the actual host system where other systems or applications are running on, i.e. the operation systems. The metrics obtained on this level are among others:

CPU metrics
There are some metrics indicating how much time the CPU spends on executing specific tasks, i.e.

- CPU User Time, indicating the amount of time the CPU spends on executing user processes (in %)
- CPU System Time, indicating the amount of time the CPU spends on executing system processes (in %)
- CPU Wait Time, indicating the amount of time the CPU spends on waiting for some other tasks to be finished (e.g. waiting for I/O operations) (in %)
- CPU Nice Time, indicating the amount of time the CPU spends on executing processes with higher niceness (i.e. lower priority) (in %)
- CPU Steal Time (for virtual machines), indicating the amount of time the virtual CPU spends waiting because of the host CPU serves other processes (in %)
- CPU Load Average over the last 1, 5 and 15 minutes is roughly speaking a measure of how many processes have to be wait to be executed. If the load is equal to the number of CPU cores, no process has to wait. The higher it is above the number of CPU cores the more processes need to wait. A load smaller than the number of CPU cores means that the CPU can still serve more processes without other process need to wait.

Main memory
The amount of main memory that is free (i.e. unused) is stored in bytes.

File system
Similar to main memory the amount of free disk space is monitored in bytes.

3.2 Cassandra

An example of a database system that can be monitored is Cassandra. Cassandra is a distributed database system for managing large amounts of data. Some of the metrics of interest are:

Requests
The count of read and write request data the system serves.

Latencies

The estimated read and write latencies, indicating how long it takes to execute a read or write request (in milliseconds). Because the exact latencies cannot be measured it is estimated using the number of read and write requests executed during the last second. The latencies are given are available as the mean, the 50th, 95th and 99th percentile.

3.3 Java Virtual Machine

A number of metrics related to the Java Virtual Machine is monitored, for example:

Threads

For each thread state (according to Thread.State) the number of threads are in that state is monitored.

Memory

The amount of memory that is used by the Java Virtual Machine(in bytes)

3.4 Others

The preceding sections just give a rough overview of the kind of data that is taken into account. There are many more metrics for the mentioned systems as well as more systems that are monitored, for example MongoDB, Node.js, Tomcat, DockerContainer and some more. It is beyond the scope of this report to describe all the available metrics, especially because some of them are quite specific. Since Instana is still in a start-up phase the list of monitored systems is still growing.

4 Knowledge

5 Goals

In general Instana shall not just monitor and present the available metrics, but assist the users with detecting possible problems and errors and make suggestions to fix them. Occuring problems and their corresponding fix suggestions are further called "issues". Issues also have a severity - a number between 0 and 10 as a rough measure on how much impact the issue has on the system health., where 0 is no issue at all and 10 is the worst state.

To detect issues there are several approaches. The simplest one is a knowledge-based approach, where domain knowledge - mainly in the form of threshold - is applied as rules. One rule for example is, that an issue is raised if the CPU load is larger than twice the number of CPU cores for more than 50% of the time during the last two minutes.

Another approach is to detect changes in the metrics. Assuming that the individual metrics show a specific behavior, it can be supposed that there might be an issue in the problem. An issue does not need to be a real problem in this case, but could indicate that something of interest happened. For example if there is a sudden increase of write or read requests to a database the user might like to be informed about that.

Moreover correlations or any kind of relationships between different metrics are of interest, especially in the case of an detected issues. By taking relationships into account, one might be able to detect the cause of a specific problem as well. For example a CPU load that is constantly too high may have different reasons. Maybe the CPU is generally too overloaded, meaning that there are too many processes running on it. Another reason may be some I/O issues; the CPU needs to wait too much for I/O tasks, such that there is too much latency when executing processes. If the metrics for CPU wait time and CPU user and system time are taken into account, there may be found a method to derive what kind of problem causes the high CPU load.

Further approaches generally take multiple metrics into account. As a simple example one might be able to predict the time a disk might be full depending on current and past behavior of the system. Currently this is done by linear regression, what is obviously not reliably method, because it is also sensitive to temporarily slopes.

When a model for multivariate metrics is found to make reliable predictions some supervised classfication may be applied. It is possible to simulate different kinds of problems by stressing the systems. For instance one can send many read or write requests to a database or write a large amount of data on the disk of the machine where the database is running. That simulates too different problems that can be labeled and used as a training set.

6 Applied techniques

During the internship several issues were addressed. In the following subsections an overview of the topics and the applied techniques is given.

6.1 Anomaly detection

One topic for monitoring is finding anomalies or changes in a single metric. If a metric starts to behave different than usual the user should be informed about that. An example of this case is the amount http requests to a server; if there are no requests monitored - or much less than usual, there might be a problem somewhere that should be investigated.

6.1.1 Confidence interval

The first approach to detect changes was a simple confidence interval based on percentiles of the data. For this purpose the the last two minutes of data is taken and divided into two windows. The first window is considered normal and functions as the reference data. The second window contains the data that is tested for "normality".

The 5th and the 95th percentile of the reference data is taken as a confidence interval. It is assumed that data of this metric normally lies within that interval. So the test data should mainly lie within this interval, otherwise a change in the recent window is assumed.

To decide whether there is a change or not a "normality index" was introduced.

- only tempory changes - no periodicity

simple appraoch with percentiles

twitter breakout and anomaly detection

two gp approaches (periodic and weekday attributes)

6.2 Linear regression

disk space problem. rls as solution for lack of historic data.

6.3 Adjusting Knowledge

explain what is meant by knowledge and give examples... to many false positive. analysing issues. and adjusted the rules.

6.4 Correlation

First approaches to find correlations between time series. Put aside, because change detection had higher priority.