Time series analysis for Application Performance Monitoring

Stefanie Deckers

February 7, 2016

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 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.

5 Current state

In the current state some simple knowledge is applied, including detecting a high load, for several CPU times. The rules are not very sophisticated, but are feasible to catch apparent problems.

For the change detection a simple algorithm was applied so far: Two windows

of data are taken into account. The first one serves as a reference window considered as normal data. The 5th and 95th percentiles of this data is the basis to decide whether there is a change on the second window: if too many values are below the 5th or above the 95th percentile, the second window is considered to contain a change. Some experiments were done to test the significance of the supposed change. Chi-Square test and a permutation test were applied, without satisfying results. So again a simple threshold was introduced to decide whether there is a change point. The whole setting is applied as a sliding windows, meaning that the two windows stay the same size but move to the right for every new data point.

This method is not satisfying for several reasons. One main problem is, that it just takes the last window as reference data regardless of whether this window can be considered as representative data of that metric or whether there already is a change or any kind of event within that window. Another issue is that this method does not take any characteristics of the metric like seasonality or specific peaks into account. It is just based on an overall range the data points lie within, what is not feasible to detect changes like missing or additional peaks or any kind of seasonality.

To face this issue a Gaussian Process is currently to be implemented, to detect changes. The aimed algorithm is based on the paper **gpcd** The idea is to find a kernel that is able to represent the metric and model the data as a Gaussian process. This Gaussian process is then used to predict the next value in the time series and compare it to the observed one. If an appropriate model for the data is found, it can be expanded to perform other tasks like classification to detect different problems.

In general he use of a gaussian process has several advantages. It is hoped that is possible to obtain good predictive results for the different metrics, since the single metrics behave very differently. Moreover some of the metrics are very noise or volatile, what hopefully can be handled by adding a sufficient noise term to the covariance matrix. Work on a gaussian process implementation is in progress. The first step will be to find an sufficient kernel and parameters and see how gaussian processes generally perform on the data.

6 Problems

There are some problems with the data and the general setting of Instana that makes it harder to model the data and get useful information out of it. This section gives an overview of issues that have to be taken into account.

One problem is the different nature of the data, as well for data of the same metric as for data of different metrics. For instance generally the disk-free metric (amount of free bytes on a disk) is very different from metrics like requests to a database. While disk-free metrics are very smooth and usually change slowly,

the request metric is very volatile. Another example are CPU metrics. They can even differ within the same kind of metric always depending on how the system is used. There may be a rather constant CPU usage or they may be periodically peaks. This makes it hard to find any correlations in the data. Some simple experiments were already done by applying spearman correlation to the read/write requests of the data and the load of the database host. But this simple approach did not lead to meaningful results.

The real-time nature of Instana restricts the possbile techniqes for several reasons. One issue is performance; to make any kind of predictions in real time algorithms must be found that are computationally not too expensive. The current structure of Instana intends secondly calculation to decide whether there is an issue or not. So computations for a new data point should not take longer than one second.

Moreover the data is processed as streaming data. For the choice of algorithms this means that techniques are seeked that do not need multiple iterations over the data. Historical data points are still available, but not as secondly data points anymore, but as the means of time tumbling windows of five seconds.

7 Discarded approaches

At the beginning of the internship some general experiments on time series representation were applied. Fourier or wavelet transforms were applied to use the resulting coefficients as data represention instead of the raw data points. The coefficients were used to cluster different metrics to get an idea of whether this data representation is sufficient for further analysis. But the frequency domain representation seemed to perform much worse than using the raw data points. Because of that and the fact that such transformation of the time series are probably not the best choice for streaming data and predictions in real-time, this appraoch was discarded for now.

8 Conclusion and future work

The main overall goal of Instana is to assist developers and IT operators with detecting problems or unusual behavior of the system. This detection should happen in real time, so any algorithms should mainly work on current streaming data.

There are mainly two different approaches for achieving this tasks. One appraoch is knowledge based, meaning that some simple rules are applied to specific metrics. For instance the rule that a CPU load should not extend twice the number of cores for a larger time period.

This approach is quite limited and specialized, so another machine learning based appraoch will be applied. The main idea is to find any abnormal behavior of single metric or a set of metrics. Moreover machine learning can also be used to implement classification, such that a specific behavior of some metrics may indicate a specific kind of problem.

So far there is just the knowledge based approach implemented for some of the metrics, especially the host metrics. For the machine learning task it is currently tried to model the data as a Gaussian process, mainly because of its great flexibility. This may the be used for change detection or classification tasks.

It was encountered that the general setting of Instana is challenging for machine learning tasks. This is because of the streaming data and real time requirements, so fast algorithms are needed that should avoid multiple data scans. A fourier or wavelet analysis is hard to apply to these requirements.