Automated Web Performance Analysis

Martin Pinzger (Mpinzger@gmail.com)
Department of Telecooperation
Johannes Kepler University Linz Austria

Abstract

Performance is a key feature in many systems nowadays. There are several tools on the market that ensure and test for adequate performance. They can be divided into simulation tools and monitoring tools. But only a few automatise and combine both approaches.

This paper describes a system capable of automatically creating a web performance simulation and conducting trend analysis of the system under test (SUT). To achieve this the system requires input information, like Monitoring Points and Static-Information about the SUT. The system monitors and analyses the SUT and based on this information generates a simulation model of the system. The simulation model is refined stepwise e.g. by adding or removing connections between the model components or adjusting the parameters until the aimed accuracy is achieved. With the help of the simulation model a prediction module creates an analysis of the SUT, and thereby can give as much information about the current state of the system and potential trends as possible. This predictive information can be used for pro-active server tuning or other performance optimisations.

The focus of my PhD thesis is on the adjustment and prediction part of the system described here. For all other parts, already existing tools and techniques will be used where possible. This initial paper outlines the complete system.

1. Introduction

Today, performance represents the key to successful applications, especially so in the field of web applications. Often performance aspects are already part of the development process of software [9], yet there are many projects where performance aspects have not been considered, and the number of realisable improvements are limited.

This paper addresses performance aspects, presenting an approach towards automatic online simulation model generation and a long term trend analysis system. The primary add-ons that have to be designed are an automation feature for the performance simulation and an automation feature for the performance prediction.

In this initial paper the complete system will be described, later work will focus on the adjusting and on the prediction parts of the system. The following research, development and testing is aimed at web applications, but the general ideas may also be adapted onto other areas.

While existing research deals with questions like How to measure performance? and How to build a fast and precise simulation?[11][12], this paper presents research focusing on an outline to answer the following questions: What methods can be used, developed and combined to automatise the process of performance measurement, simulation and prediction, increasing effectivity and accuracy? Are there any existing methods which can be applied to solve this problem?

2. Concept Orientation

The idea is to develop a system that is capable of automatically creating a web performance simulation and conducting a trend analysis of the system under test (SUT). The system has to provide three main functions: data collection, simulation and prediction. A potential architecture of these three components will be discussed in the next section.

- **Data Collection Component:** The component monitors the SUT, records relevant information and provides static information about the SUT. These pieces of information are then preprocessed and fed into the simulation and the prediction component.

- **Simulation Component:** The aim of the this component is to automatically generate and adjust a simulation model of the SUT. For this task the component has to combine the information provided by the data collection component and apply the extracted knowledge of the SUT to create the simulation model. Then based on the same procedure, the accuracy of the simulation model is improved in a self repeating process.
Prediction Component: This component uses the information, about the SUT, provided by the data collection component to accomplish its task. Based on the information about the SUT and by using the simulation component the prediction component generates possible scenarios and creates a long term trend analysis.

3. System Architecture

The major objective of this thesis is to provide a reference architecture for automated web performance simulation and trend analysis. Existing tools and methods should be integrated into this frame based on well defined interfaces among the components. The idea is to use existing tools and frameworks wherever possible and to concentrate on the process of interoperating the different technologies to create the system. As a result the research will focus on the one hand on the process and the interaction of the components and on the other hand on the new techniques required for simulation model generation and adjustment. The following sections will describe the system architecture, and refer to the technologies which will be used. The areas, where more research has yet to be done are also highlighted.

The system architecture is based on several components which are used at the same time but in a random order and it is important that specific components work at a specific minimal rate. So the idea is to have a primary management component that commands the other components. The management component also needs the ability to pause other components to ensure the specific rate for the components. Figure 1 provides a schematic of the system architecture, the dashed lines indicate that the components should be able to exchange data with each other without disrupting the communication with the management component.

4 Data Collection Component

The data collection component is made up of three elements which are the configuration component, the data recording component and the monitoring component. The configuration component provides access to static information about the SUT and the drafted system. The task of the data recording component is to store and preprocess the collected information about the SUT; key features here are: capability to serve several components at the same time (multi-threaded / multi-processor), to provide high performance access to the stored data and to incrementally add data. The monitoring component is dealt with in more detail in the following section.

4.1 Monitoring Component

Like any other system the drafted system is based on the input data provided, so even with the research and development focus located in another area, it is important to deal with monitoring. In general performance monitoring is divided into three types depending on their implementation [5][9]: hardware monitors, software monitors and hybrid or firmware monitors.

In this work the focus is on software monitoring tools, these have been divided into three different types based on there interaction with the environment:

- **Active Software Monitoring** gains a maximum of information at the cost of influencing the SUT. The primary utility used for active performance monitoring is code instrumentation. This can be done in several different ways. The obvious way is to edit the code by hand - but while this offers maximum flexibility it also requires a lot of time and knowledge about the SUT (what to monitor and how to test). Another possibility is to use automated tools for code instrumentation. These tools have different strategies of editing the code of the SUT, e.g. an annotation mechanism based on the JAVA byte code could be used. A third way of realising active performance monitoring is to use the built in functions of e.g. enterprise java beans applications or .NET applications to collect real-time runtime information. The main advantage of these methods is that no access to the source code is required [10].

- **Passive Software Monitoring**, gains less information than active software monitoring but has the advantage of no interference with the SUT. The basic way of implementing passive performance monitoring is to capture the input and output of the SUT. This could be realised by network sniffing respectively network monitoring. Two possible methods for implementing this should be mentioned: first, network monitoring tools with reporting functions, second, frameworks and libraries to implement network capturing tools. Examples for network monitoring tools are Nagios, Etheral.
Active and Passive Software Monitoring, is a combination of both monitoring approaches. Most systems have a built in logging mechanism, the idea is to use this mechanism online for monitoring purposes by sharing it via network with the system which tries to analyse the SUT.

The first realization of the drafted system will focus on code instrumentation, network sniffing libraries (JPCAP) and analysing standard log files like the Apache HTTP Server Log.

5 Simulation Component

The simulation component represents one of the two core components of the drafted system. This component has several functions to accomplish, which are quite demanding in respect to system resources and time. Therefore a key requirement for the so called online simulation is that the time required for the simulation must not be higher than the real response time of the system.

5.1 Model Generation Component

The model generation component has the task of initially generating the simulation model. There are three kinds of possible solutions for this, namely:

- Minimum complexity simulation model: The idea is to start with the simplest possible simulation model, in case of Queuing-Network-Models this would be a single class / single queue / single server model.

- Average complexity simulation model: A simulation model that aims at average complexity. This approach seems to be the best solution at first view, but there are several unanswered questions like: "What is exactly is average complexity?", "In which direction should the model be changed if it is not adequate?".

- Maximum complexity simulation model: A model that starts the system with a simulation model which attempts to include all possible combinations at the startup. E.g., a Queuing-Network-Model which is initialised with a multi class / multi queue / multi server model.

In a first realization of the automatic simulation model generation process the minimum complexity simulation model possibility will be realised. Based on the research plan, the first implementation of the drafted system will have no model generation component. The simulation model will be provided statically so that only the model comparison component and the model adjustment component are required.

5.2 Model Comparison Component

To be able to determine whether to run the model adjustment component or to keep in its current form, it is necessary to compare the results of the simulation model with the results of the SUT. For this the model comparison component uses the monitoring information provided by the data collection component as well as the results generated by the model simulation component which runs the simulation model. The idea is: When a request is sent to the SUT, it is also inserted into the simulation model. Then the response should be created by the SUT and the simulation model at the same time. If the discrepancy between responses is higher than the acceptable value, the simulation model needs to be adjusted.

5.3 Model Adjustment Component

Depending on the strategy of the model generation component there are different strategies for the model adjustment component as well. However in a general concept the basic operation for the model adjustment component could be summarised as follows: change a configuration value of the simulation model, add/remove a component to the simulation model and change a parameter of the model simulation component.

The configured value of a link in the simulation model can be changed using one of the following techniques:

- Stepwise: The basic idea is to increase or decrease the parameter value until the simulation model and the SUT correspond to each other. A delta which gives the size of a unit interval, by which the value is increased or decreased, has to be specified.

- Reference Table: The concept is that the system uses a provided set of values for the parameters of the simulation model. This set of values for the different parameters, could be a specific one for the concrete simulation model or a generic one which represents the common case.

- Random Estimation: A mathematical concept which could be freely defined should reduce the number of value changes which have to be run. For this the idea is to use methods like moving average, ARMA and ARIMA. New combined approaches to estimate the next value should be included here. As the system architecture provides all information it is well suited to automatic learning approaches. If the available time for computation is there, even a method which uses markov models could be integrated.

In a first realization of the model adjustment component only the operations for changing the configured value of a
link should be realised, to examine whether this operation would perhaps be sufficient.

5.4 Model Simulation Component

The model simulation component implements the execution environment for the simulation model. Allowing for the time consumption of the simulation model execution, the aim is to create a model simulation component which is independent from the basic simulation environment, so that the simulation environment can be replaced if the performance demands are not fulfilled. Possible options at the moment are: JSIM [1], Network Simulator [3] and a set of simulation frameworks mentioned in [4].

6 Prediction Component

The prediction component consists of four parts, these are: longterm analysis-, statistical-, scenario generation and execution- and reporting-component. The individual component’s responsibilities can be described as follows:

6.1 Longterm Analysis- and Statistical Component

The longterm analysis component and the statistical component are based on the information provided by the data collection component. This information is processed and analysed by the longterm analysis component, also the semantic of the provided input data are taken into account. The statistical component executes mathematical functions on the provided data. The component also should provide a resume option whenever possible, so as to be able to incrementally calculate the standard statistical data. Finally the results of the components are forwarded to the report component to be included into a report which is created by the prediction component.

6.2 Scenario Generation and Execution Component

The scenario generation and execution component uses the simulation model to generate and test possible events. Results provided by these experiments are then reported back to the data collection component so that the results can be included into the longterm analysis.

6.3 Reporting Component

The reporting component is intended as a base for the conversion of the data which should be shared with the environment of the drafted system. To accomplish this, the reporting component has to provide high flexibility and a multiple communication interfaces.

7 Management Component

The management component is responsible for the process chain of the outlined system. Based on this process chain the management component has to execute the different components of the drafted system. To ensure the performance requirements of the components, the management component needs the qualification and authorisation to pause, or even terminate specific components.

8 Conclusions

In this paper I have outlined the motivation behind my PhD thesis work and given a survey of research challenges and problems to be addressed. My work will focus on (a) developing a reference architecture supporting automated web performance analysis (as sketched in this paper), on (b) a prototypical implementation where I intend to integrate existing approaches and tools for subcomponents and (c) performing case studies illustrating the suggested approach.

References