INSTITUT FÜR INFORMATIK

Benchmarking the Performance of Application Monitoring Systems

Jan Waller

Bericht Nr. 1312
November 2013
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Benchmarking the Performance of Application Monitoring Systems

Jan Waller

Department of Computer Science
Kiel University, Kiel, Germany
jwa@informatik.uni-kiel.de

Abstract: Application-level monitoring of continuously operating software systems provides insights into their dynamic behavior helping to maintain their performance and availability at runtime. Such monitoring may cause a significant runtime overhead to the monitored system depending on the number and location of used instrumentation probes. In order to improve a system's instrumentation to reduce the caused monitoring overhead, it is necessary to know the performance impact of each probe.

In this paper, we present our MooBench approach to split the possible causes of monitoring overhead into three portions, and to quantify these portions of monitoring overhead with the help of benchmarks under controlled and repeatable conditions. To the best of our knowledge, most publications on monitoring frameworks provide none or only weak performance evaluations, making comparisons cumbersome. Our benchmark approach provides a basis for such comparisons.

1 Introduction

Modern software systems, especially continuously operating systems, have complex interactions within their internal components. In order to ensure the systems' performance and availability at runtime, it is necessary to monitor their internal behavior. Application-level monitoring frameworks, such as Kieker [vHWH12], can provide these required insights at the cost of additional performance overhead. This overhead is caused by the monitoring probes that instrument the monitored system, effectively executing additional monitoring code within the targeted system. Depending on the actual implementation of the monitoring framework, the used probes, and the workload of the monitored system, each execution of a monitored part of the software system incurs an additional performance overhead compared to the uninstrumented execution.

Detailed knowledge of the actual performance overhead, that is caused by each used probe at a specific location within the monitored software system, helps planning the instrumentation of this software system with acceptable performance overhead. Within our proposed MooBench approach, we split the possible causes of monitoring overhead into three portions. This split allows for a detailed comparison of different components of monitoring frameworks with each other. Furthermore, we propose a series of benchmarks to measure these portions of monitoring overhead under controlled and repeatable conditions.

The rest of this paper is structured as follows. In Section 2, we present our goals and research questions. In Sections 3 and 4, we propose our MooBench approach and its evaluations. Finally, we draw the conclusions in Section 5.

2 Goals and Research Questions

We envision a series of benchmarks to determine the performance of application-level monitoring frameworks. This section provides an overview of our goals and research questions.

G1: Causes of Monitoring Overhead In order to determine the performance of an application-level monitoring framework, we require a definition of monitoring performance. A possible definition for the performance is the change in the response time of a monitored method. This leads to our first research question: *Q1: What are the causes for observed changes in the response time of a monitored method?*

G2: Benchmarks to Measure the Monitoring Overhead Given our proposed causes of monitoring overhead, we have to determine the amount of monitoring overhead induced by each cause. A common solution to measure and compare performance in software engineering is the use of benchmarks. This leads to our second research question: *Q2: How to develop a benchmark to measure the causes of monitoring overhead?*

This research question leads to further subquestions, for instance:

Q2.1: What constitutes a good benchmark?

Q2.2: How to measure the monitoring overhead?

Q2.3: How to select benchmarking scenarios and workloads?

3 The MooBench Approach

Our MooBench approach to benchmark the performance of application monitoring systems proposes a split of the possible causes of monitoring overhead into three portions (**G1**). A simplified UML sequence diagram for monitoring a typical method call with the Kieker monitoring framework is presented in Figure 1. The portions of monitoring overhead (I, C_1, C_2, W_1, and W_2), as well as the time of executing the uninstrumented original method (T), are annotated in red. These portions correspond to three causes of monitoring overhead: the instrumentation of the monitored system (I), collecting data within the system ($C = C_1 + C_2$), and either writing the data into a monitoring log or transferring the data to an online analysis system ($W = W_1 + W_2$). Refer to [WH12, WH13] for a more detailed description of the identified portions of monitoring overhead.

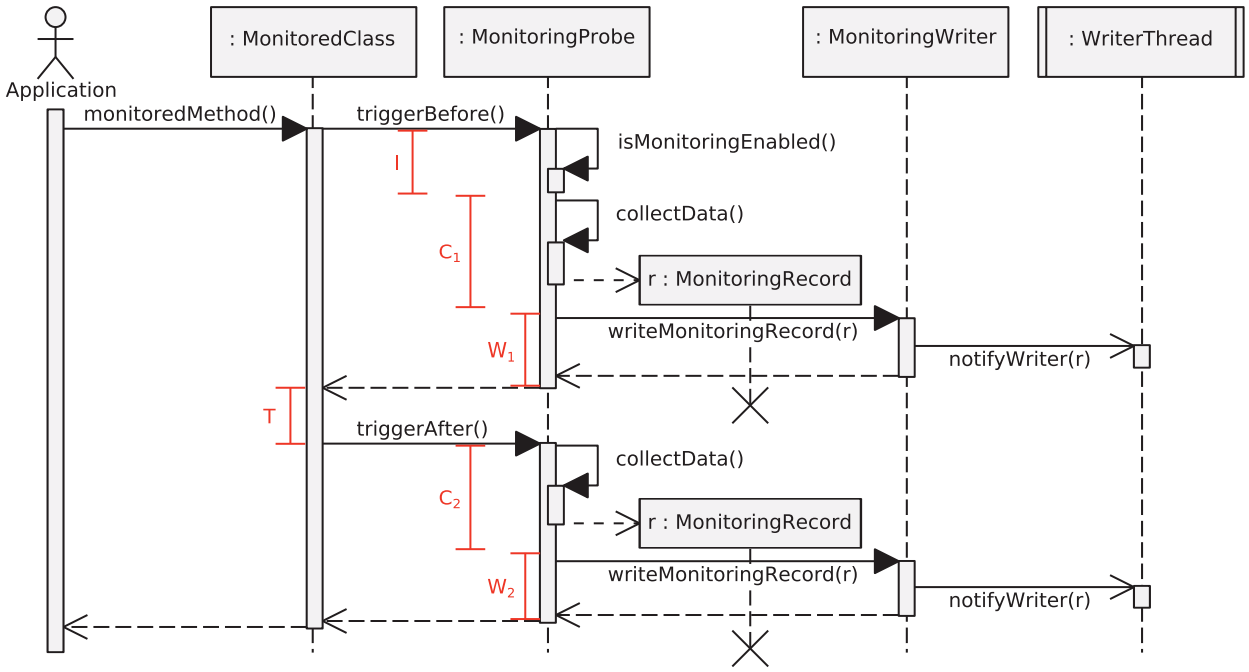

Figure 1: UML sequence diagram for method monitoring with the Kieker framework [WH13]

Figure 2: Benchmark engineering process [WH13]

In order to measure and quantify the portions of monitoring overhead in a monitoring framework, we propose the MooBench micro-benchmark (**G2**). It is designed in accordance with our benchmark engineering methodology, splitting the benchmark engineering process into three phases (see Figure 2). For each phase, a set of common guidelines is provided: to design and implement a benchmark, to execute the benchmark, and to finally analyze and present the results of the benchmark.

Our resulting MooBench micro-benchmark has been designed to measure the monitoring overhead of application monitoring frameworks. A brief description of the benchmark is included in [WH12]. Our benchmark engineering methodology is detailed in [WH13].

In addition to the use of our micro-benchmark, we propose the use of established macro-benchmarks, e. g., the SPECjbb2013 or SPECjvm2008 benchmarks. These benchmarks provide additional scenarios to our own micro-benchmark. On the other hand, these benchmarks are not focussed on benchmarking the monitoring overhead of single method executions. Thus, their results might become influenced by other parameters.

Finally, we propose a meta-monitoring approach. That is, monitoring the monitoring framework. Thus, we can use its performance monitoring capabilities to get a detailed description of the performance cost of monitoring a software system.

4 Performed and Planned Evaluations

Our proposed split into three portions of monitoring overhead (**G1**) has been evaluated in the context of the Kieker framework within several papers, e. g., [WH12, vHWH12, WH13]. Furthermore, we plan to verify these portions with lab experiments conducted with the help of further scientific and commercial application-level monitoring systems.

Our proposed micro-benchmark (**G2**) has already been used to evaluate the performance impact of several components of Kieker for several years. Furthermore, we employed the micro-benchmark in a structured performance engineering approach to enhance the monitoring performance of Kieker. Similar to our planned evaluation of the split into three portions of overhead, we plan to execute our micro-benchmark on further monitoring systems and to compare the monitoring overhead of these systems with each other.

Furthermore, we plan to validate the results of our micro-benchmarks by comparing them to the results of our performed macro-benchmarks and to the results of our meta-monitoring of the frameworks.

Finally, benchmarking is often considered to be a community effort [SEH03]. Thus, we provide our benchmarks as open-source software and invite the community to use our tools to verify our results and findings.[1]

5 Conclusions

We propose three typical causes of monitoring overhead in application-level monitoring systems and a series of benchmarks and measurements to quantify this overhead. In summary, we introduced our MooBench approach, its goals, and its research questions. Additionally, we sketched ideas for the planned and performed evaluations.

References

[SEH03] Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using Benchmarking to Advance Research: A Challenge to Software Engineering. In *Proc. of the 25th Int. Conf. on Software Engineering*, pages 74–83. IEEE Computer Society, 2003.

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proc. of the 3rd ACM/SPEC Int. Conf. on Performance Engineering*, pages 247–248. ACM, 2012.

[WH12] Jan Waller and Wilhelm Hasselbring. A Comparison of the Influence of Different Multi-Core Processors on the Runtime Overhead for Application-Level Monitoring. In *Multicore Software Engineering, Performance, and Tools*, pages 42–53. Springer, 2012.

[WH13] Jan Waller and Wilhelm Hasselbring. A Benchmark Engineering Methodology to Measure the Overhead of Application-Level Monitoring. In *Proc. of the Symp. on Software Performance — Joint Kieker/Palladio Days*, pages 1–10. CEUR Workshop Proc., 2013.

[1] `http://kieker-monitoring.net/overhead-evaluation/`