

Investigating the Warmup Behavior of Software Applications

Jóakim von Kistowski
joakim.vonkistowski@th-ab.de
Aschaffenburg UAS, Germany

Elijah Seyfarth
s210942@th-ab.de
Aschaffenburg UAS, Germany

Sebastian Frank
Independent Researcher, Germany

Abstract

Modern distributed applications are highly dynamic, starting and terminating different parts of the application at run-time. These actions create transient phases in which application performance and behavior may differ significantly from long-running, steady-state execution phases.

A transient effect commonly observed when starting executable units is the so-called *warmup* behavior. This usually manifests itself as a performance penalty at the start of an executable unit's run-time. The exact duration and impact of this warmup behavior can vary due to a variety of factors, such as the programming language, execution environment, frameworks/libraries, or underlying application data. Existing work, such as [4], focuses on specific aspects of warmup behavior and compares specific technologies. As a result, the overall effects have only been sparsely analyzed and classified.

State-of-the-art software performance measurement techniques, such as the SPEC Power and Performance Methodology [2], take care to avoid warmup effects, achieving stable and accurate results at the cost of ignoring the practical effects and implications that real-world warmup behavior might have. In addition, the effect of warmup behavior is often omitted from performance prediction models, again leaving end-users in the dark about this critical performance aspect of their applications.

In this experience talk, we present the results of a broad analysis of different warmup behaviors observed in a large set of experiments. We highlight the effects of both expected and unexpected factors that may influence the warmup behavior of an application. Our experiments range across different programming languages, execution environments, concurrency levels, and application types. Specifically, we analyze the warmup behavior of applications written in Python, Java, JavaScript (Node.js) and C#. For these languages, we analyze multiple execution environments (such as different JVMs) and multiple common frameworks. With the Java-ecosystem being commonly associated with complex warmup behavior, we perform a detailed investigation, showcasing

warmup-related aspects in Java applications. This investigation analyzes both warmup effects on synthetic micro-workloads, as well as warmup effects on applications using common frameworks, such as Spring-Boot, Quarkus, Javalin, pure Servlet containers, and more. We show how these underlying factors affect warmup behavior over time, and we generally characterize common forms of warmup behavior.

Finally, we present the do's and don'ts of warmup measurements. We show how different types of performance and load tests can be used to measure warmup behavior, and discuss why these measurements may lead to different conclusions. We address how the selection of open or closed workload types may influence the outcome of a warmup measurement, and we discuss the impact of high-concurrency load testing on warmup measurement results.

This experience talk is intended for researchers as well as practitioners. Researchers can use the experiences shared in this talk to improve the design of performance-related experiments by extending them to specifically cover warmup behavior. This should be especially useful for experiments on dynamic distributed applications, such as cloud-native software applications. After all, such applications feature lots of dynamic changes that trigger warmup behavior in their constituent parts during run-time. Alternatively, researchers can use the results of this talk to improve the accuracy of performance prediction models and simulators. Tools such as Palladio [1] and MiSim [3] can be enhanced by applying this knowledge about warmup behavior to improve the accuracy of their performance predictions in volatile and dynamic execution environments.

Practitioners, in turn, can use the results of this talk to improve their understanding of warmup behavior and its implications for their applications. This allows them to make better-informed technology choices for specific parts of their applications, and it allows them to improve their performance testing and benchmarking practices by taking into account the best practices regarding warmup experiments shared in this talk.

References

- [1] S. Becker, H. Koziolok, and R. Reussner. “The Palladio component model for model-driven performance prediction”. In: *Journal of Systems and Software* 82.1 (2009), pp. 3–22.
- [2] J. von Kistowski et al. “Measuring and rating the energy-efficiency of servers”. In: *Future Generation Computer Systems* 100 (2019), pp. 579–589.
- [3] S. Frank et al. “MiSim: A Simulator for Resilience Assessment of Microservice-Based Architectures”. In: *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*. 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS). Dec. 2022, pp. 1014–1025.
- [4] N. K. Ugwumba and P. S. Jaja. *Ahead-of-Time vs. Just-in-Time Compilation Trade-offs: Empirical Performance Studies*. Oct. 22, 2025. Pre-published.