Measuring Context Switches in Serverless Environments

Elaine Yao
elainedv111@gmail.com

Abstract

Todo

1 Introduction

In this work, we aim to answer the following research questions:

1.

2.

2 Background

2.1 Serverless environment

2.2 Context switches

3 Methodology

3.1 Challenges

Measuring the context switch time in serverless functions has several challenges:

C1 Characteristics of context switch in serverless environments Context switch is triggered differently in serverless functions compared to in traditional operating systems as it's scheduled by the cloud provider. And the service provider has the tendency to trigger more functions in the same hardware to get more profits. Thus, context switch may happen not only due to performance factors like multi-threading/processing, but also provider's profit considerations.

C2 Benchmark accuracy As no such benchmark is provided so far, we have to analyze the factors in our own benchmark that may lead to the potential variation to golden value, in order to reason about the accuracy of the measurement.

C3

3.2

To tackle *C1*, we design the following experiments to analyze the factors influencing context switch in serverless environments. We measure the execution time elapse between two adjacent lines of code repeatedly under different configurations in serverless functions. Shahrad *et al.* [6] shows that the execution time in serverless functions is influenced by function invocation frequency, memory size and function type. Thus these configurations will include the function invocation rate, memory allocated. Theoriotically, the measured time elapse should be 0 as there is not execution between these two lines. However, our initial experiments show that this is neither 0 nor a fixed value, indicating that interruption happens between these two lines and the time elapse changes with different configurations. Although the interruption is not limited to context switch but also page faults handling or container management, it can still provides insights in the influence of different configurations on context switch time in serverless environments.

3.3 Combine various benchmarks

To tackle *C2*, we first run our own designed benchmarks on local Linux system and compare the results with published benchmarks [1, 2, 4, 5], ensuring the correctness of the proposed ones. Then we deploy the new benchmarks in serverless functions and compare the results. If one of the benchmarks is consistently higher or lower than others, then we'll explore the reasons and this can help us improve our accuracy.

The previous benchmarks in context switching on Linux system mainly consider Our initial study to *C1* finds that context switch in serverless environments will also be influenced by invocation rate and memory allocation.

Currently, the new benchmarks are designed as the prior benchmarks with different configurations. The prior benchmarks we are choosing are:

1.

Benchmark	Pipe	Condition Var	lmbench	Pipe(python)
Time(/us)	1.96	2.29	1.78	33.6

Table 1: Context switch time by different benchmarks

Next, we'll implement these functions with different invocation rate and memory allocation setting in the cloud.

4 Experiments

4.1 Experiment setup

We implement the benchmarks on Google Cloud, which is a mainstream cloud platform to provide serverless functions. For local measurements, we use an Ubuntu 20.04-64bit machine with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz processor and 12GB RAM.

4.2 Factors impacting context switches in serverless environment

4.3 Measuring local context switches

All the previous benchmarks [1, 3–5] are implemented in C language. However, currently Google Cloud only supports functions written in Node.js, Go, Java, Python and Ruby. Consider that Python is one of the most popular languages, rewriting the prior benchmarks in Python and examining its correctness in local PC is important.

The context switch time measured by the first three benchmarks in local PC is usually among 2us from Table 1. However, when it's rewritten by Python, the measured time is 10 times larger than previous results. The reason might be that extra execution is added to the time measurement when translating the language. What's more, the C programs execute faster as compiled programs while Python executes slower due its interpreted programs. Therefore, these extra execution time is more obvious when using python. We aim to improve the Python version to make the result close to previous benchmarks.

4.4 Measuring context switches in the cloud

5 Discussions

5.1 Limitations of our Experiments

5.2 Future work

6 Related Work

References

[1] Measuring context switching and memory overheads for linux threads. https://eli.thegreenplace.net/2018/measuring-context-switching-and-memory-overheads-for-l Accessed: 2022-2-25.

[2] Francis M. David, Jeffrey C. Carlyle, and Roy H. Campbell. Context switch overheads for linux on arm platforms. In *Proceedings of the 2007 Workshop on Experimental Computer Science*, ExpCS '07, page 3–es, New York, NY, USA, 2007. Association for Computing Machinery.

[3] Chuanpeng Li, Chen Ding, and Kai Shen. Quantifying the cost of context switch. In *Proceedings of the 2007 Workshop on Experimental Computer Science*, ExpCS '07, page 2–es, New York, NY, USA, 2007. Association for Computing Machinery.

[4] Larry W. McVoy and Carl Staelin. lmbench: Portable tools for performance analysis. In *USENIX Annual Technical Conference*, 1996.

[5] John K. Ousterhout. Why aren't operating systems getting faster as fast as hardware? In *USENIX Summer*, 1990.

[6] Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. Architectural implications of function-as-a-service computing. *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019.