Task Report
For Google Trace Analysis

Yilong Zhao[a]

[a]*Shanghai Jiao Tong University*

February, 2023

Abstract

I implemented a toy tool to decode Google memtrace through **DynamoRio**'s API to analyze the characteristics and patterns of the thread switch. Based on the CDF images, I propose few key findings and give my **naive idea** about optimizing the performance caused by warm-up time. However, DynamoRio does not provide accurate records of switches, so I used an approximation and emailed the developer to confirm, which will be explained in detail later

1. Result

I provide the analysis results for $50,000,000$ instructions instead of $10,000,000$ instructions because I think more sample points will fit the distribution more accurately. The CDF images of the instructions executed between two switches are shown below.

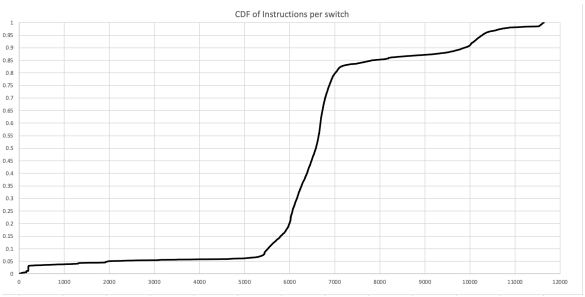

Figure 1: CDF of instructions per switch

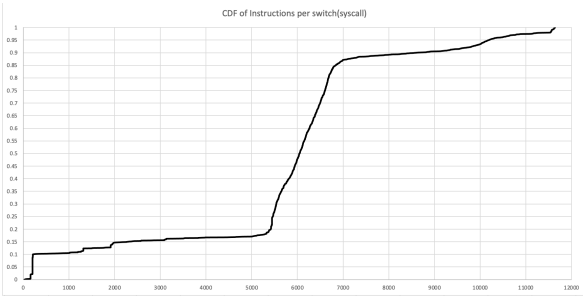

Figure 2: CDF of instructions per switch(syscall)

1.1. Observations

Among all kinds of switch, we can find that executed instructions is highly **concentrated**, with $6000-7000$ times accounting for 60% and $5000-8000$ times accounting for 80%. This may come from the characteristics of the thread code or the scheduler's time slice scheduling pattern.

Considering the switch caused by **syscall**, we find that in addition to the concentration at $5500-7000$, a noteworthy point is that $0-200$ executed instructions **occupies** 10%. The former point is consistent with the overall feature, while the latter feature probably comes from the code logic of calling syscall twice in succession, kind of **IO-bound task**, offering chance of optimization.

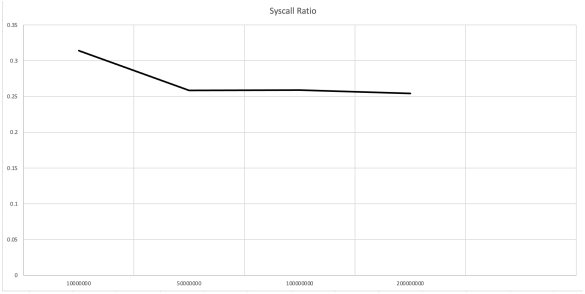

Figure 3: Ratios of the switch caused by syscall

We can see that the percentage of switch **caused by syscall** tends to be 25% when the number of instructions measured increases, indicating that the optimization towards syscall is feasible.

1.2. Idea

Because of the warm-up time of cache and predictors, the performance drops quickly when thread switch occurs. In fig. 2, We found that 10% of the switch caused by **syscall** occurred after just one hundred executed instructions, meaning that one thread was switched in, destroying the cache and predictors of the other threads, and then switched out without exploiting CPU's compute resource.

A naive idea is that if we can somehow, for example, compiler detection or programming language marking, make the OS **not change** the cache and predictors when switching a thread that only executes a few instructions in, then the performance of other threads will not be decreased.

2. Method

I use DynamoRio's interface to decode and read the trace, treating the instruction size of 2 as a possible syscall identifier. However, I didn't find the thread switch identifier in the single thread trace in the official documentation. By observing the pattern of the records, it is concluded that each thread switch is accompanied by a *CPU_ID* Marker. So I arbitrarily use this as the identifier of the thread switch. I then emailed[1] the developers and learned that DynamoRio's monitoring for threads scheduling did not meet the required granularity. Therefore my experimental results may not be accurate and future work will require more thread files to collect accurate data.

All codes and results can be found in my github repo[2].

3. Work

First I learned how the dynamic binary instrumentation tool works and understood the DynamoRio's API, then I tried to decode trace files using ZSIM as a reader. After going through "a lot of hard work" to set the environment, I found that the version of DynamoRio used in the trace file did not match the ZISM's one. I also read the "Modern Operating System" textbook to learn the basics of threading.

In the course of my work, I found myself weak in finding the right tools to solve problems, and my speed in reading documents still needs to be improved.

[1] See:https://groups.google.com/g/DynamoRIO-Users/c/Id9V6p7_02k

[2] See:https://github.com/happierpig/TraceToy