VisPerf: visualize and compare performance of stream processing applications executed on CPU

Claudio Scheer* Dalvan Griebler† Isabel Harb Manssour‡

Pontifical Catholic University of Rio Grande do Sul - PUCRS
Brazil

Abstract

This is my abstract.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

This is the introduction.

2 Perf

Perf is a profiling tool included in the Linux kernel. It can be used to capture CPU performance counters, tracepoints, kprobes[1], and uprobes[2] [1]. Perf also supports software events, such as page misses. In turn, tracepoints are placed in the source code to collect timestamps and stack traces. Perf has low overhead, as it is integrated into the kernel. Moreover, Perf supports counters in different architectures, which makes Perf a widely used tool.

There are several subcommands available in Perf. Below we list the main subcommands:

- `perf stat`: get event counts;
- `perf record`: record events for later analysis;
- `perf report`: report events recorded;
- `perf top`: see live event count;
- `perf list`: list available events for current architecture;

Discuss other subcommands. Performance counters are hardware registers that count events of the CPU, such as instructions executed and cache-misses.

2.1 Events

`perf list` shows events from several sources. Hardware events are provided by the processor and its PMUs (Performance Monitoring Unit), which may vary among different companies.

Software event

Hardware cache event

Next, we discuss what each counter used in our experiments captures.

- **cpu-cycles**: number of fetch-decode-execute cycles the CPU executed;
- **instructions**: number of instructions the CPU executed;
- **cache-misses**: number of cache-misses;
- **cache-references**: number of references found in cache;
- **L1-dcache-load-misses**: number of misses when loading data from L1 cache;
- **L1-dcache-loads**: number of data loads from L1 cache;
- **L1-dcache-stores**: number of data stores made in L1 cache;
- **L1-icache-load-misses**: number of misses when loading instructions from L1 cache;
- **L1-icache-loads**: number of instructions load from L1 cache;
- **L1-icache-stores**: number of instructions store made in L1 cache;
- **LLC-loads**: number of references loaded from LLC (Last Level Cache, usually L3);
- **LLC-load-misses**: number of references missed when loading from LLC;
- **LLC-stores**: number of references store made loading in LLC;
- **mem-stores**: number of stores made in the main memory;
- **mem-loads**: number of loads made from the main memory;

3 VisCPU

3.1 Data Pre-processing

Explain what we did.

4 Conclusion

Acknowledgments

Thanks!

References

[1] Linux Kernel Organization, Inc. *Perf Wiki*, May 2021.

*e-mail: claudio.scheer@edu.pucrs.br
†e-mail: dalvan.griebler@edu.pucrs.br
‡e-mail: isabel.manssour@pucrs.br

[1]`https://www.kernel.org/doc/html/latest/trace/kprobes.html`
[2]`https://www.kernel.org/doc/html/latest/trace/uprobetracer.html`