# 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] [2]. Perf also supports software events, such as page misses. In turn, tracepoints are placed in the source code to collect timestamps and stack traces and performance counters are hardware registers that count events of the CPU, such as instructions executed and cache-misses. Perf has low overhead, as it is integrated into the kernel. Moreover, Perf supports counters in different architectures. Therefore, it can be used to profile applications in different Intel architectures or even in AMD CPUs.

There are several subcommands available in Perf. Below we describe some of the subcommands:

- `perf stat`: gather performance counter statistics from a command;
- `perf record`: run a command and record its profile for later analysis. By default, profile is saved in a file named `perf.data`;
- `perf report`: read `perf.data` and display the profile recorded in this file;
- `perf script`: read `perf.data` and display the trace recorded in this file;
- `perf top`: display performance counter profile in real time;
- `perf list`: list events available to be captured in the current architecture;

*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`

`perf list` shows events from several sources, such as hardware and software. Hardware events are provided by the processor and its PMUs (Performance Monitoring Unit), which may vary among different companies. Next, we discuss the counters captured in our experiments [1].

- **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;
- **branch-misses**: number of mispredicted speculative branches;
- **branch-instructions**: number of correct speculative branches;
- **bus-cycles**: number of CPU cycles needed to fetch or write data to the main memory, for example;

For this paper, we profiled the person recognition application. The main goal of this application is to recognize is there is a person in a specific image, using OpenCV. To characterize a stream processing application, it is, we cannot predict when data will stop entering the application, we used a video as input.

Towards exploiting parallelism, we use the FastFlow library and compared two mapping policies. In summary, the mapping policy defines in which core threads launched by the application will be placed. We profiled the person recognition application using the default mapping policy os the Linux operating system and the mapping policy proposed by FastFlow.

## 3 VISPERF

To better visualize data captured from stream processing applications we propose VisPerf. The goal of this application is to

### 3.1 Data Pre-processing

Explain what we did.

## 4 CONCLUSION

## REFERENCES

[1] collectd Wiki. *Plugin:Intel PMU - collectd Wiki*, June 2021.
[2] Linux Kernel Organization, Inc. *Perf Wiki*, May 2021.