A Survey of State-of-the-Art NVIDIA GPU Profilers

Benjamin Walters
Dept. of Computer Science
Illinois Institute of Technology
bwalter4@hawk.iit.edu

Scott J. Krieder
Dept. of Computer Science
Illinois Institute of Technology
skrieder@iit.edu

Dr. Ioan Raicu
Dept. of Computer Science
Illinois Institute of Technology
iraicu@cs.iit.edu

High Parallelism

With the rapid growth of general purpose computing on GPUs (GPGPU), many applications are able to leverage the high parallelism of the GPU. However, since applications can run on hundreds of cores, it is difficult to visualize and analyze their performance. Therefore, developers need tools that allow them to do things such as collect metrics and capture traces of application runs.

Proposed Work

This work aims to analyze the features and usability of GPU-profiling tools. The focus of this work will be on tools that profile NVIDIA GPUs running CUDA. This work investigates the NVIDIA Visual Profiler (NVVP), NVIDIA CUDA Profiling Tools Interface (CUPTI), and Vampir. Visualization features that will be sought include application timelines and traces. Metrics that will be sought will include number of instructions executed and kernel efficiency.

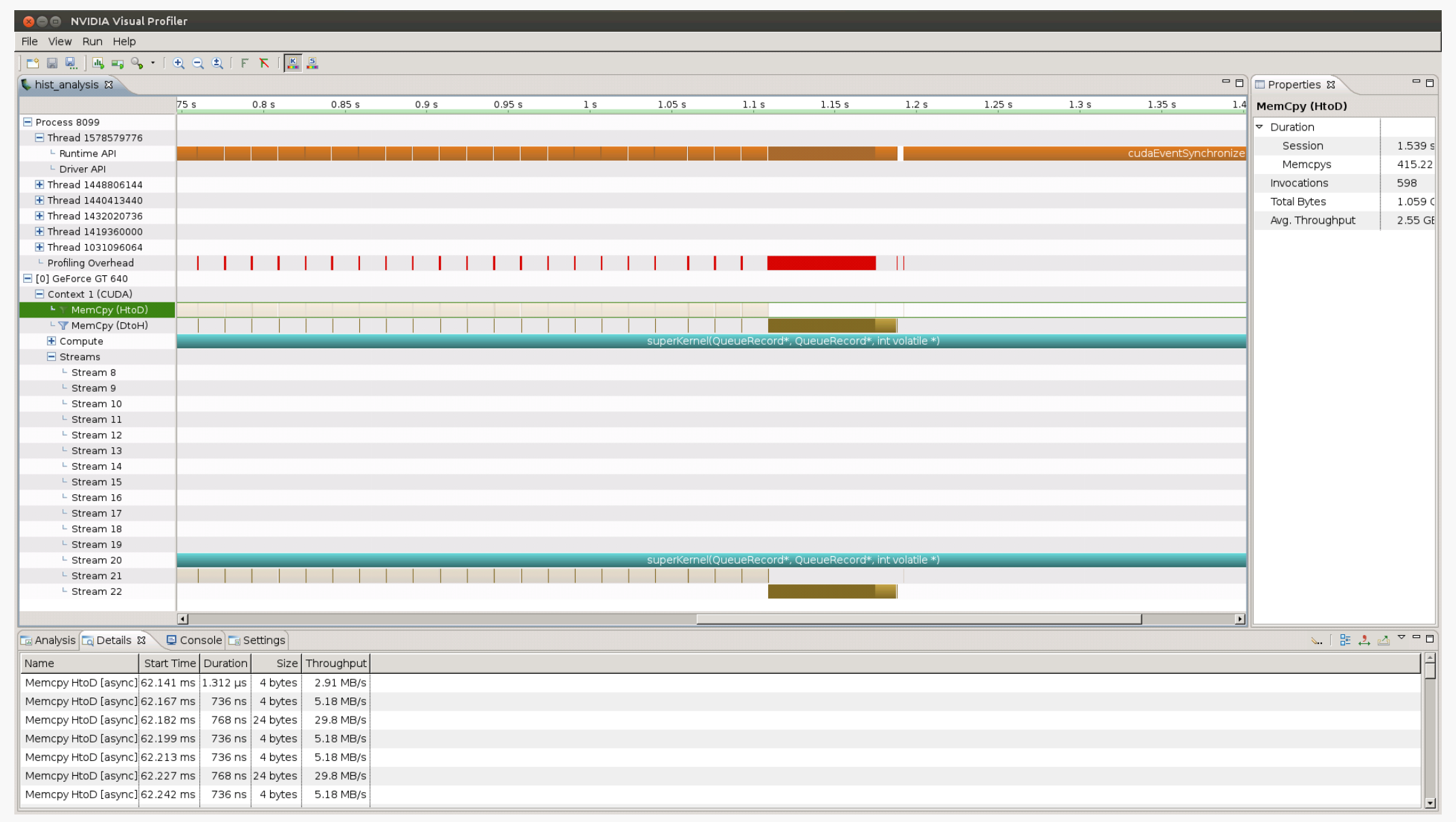
Overview of Profilers

NVIDIA Visual Profiler (NVVP) is a profiler with a graphical user interface. It is included in the CUDA Toolkit, and it does not require any code modification.

CUDA Profiling Tools Interface (CUPTI) is a C library that allows access to hardware counters of the GPU. It also allows the user to attach user-defined functions to CUDA API calls for more complicated profiling functionality.

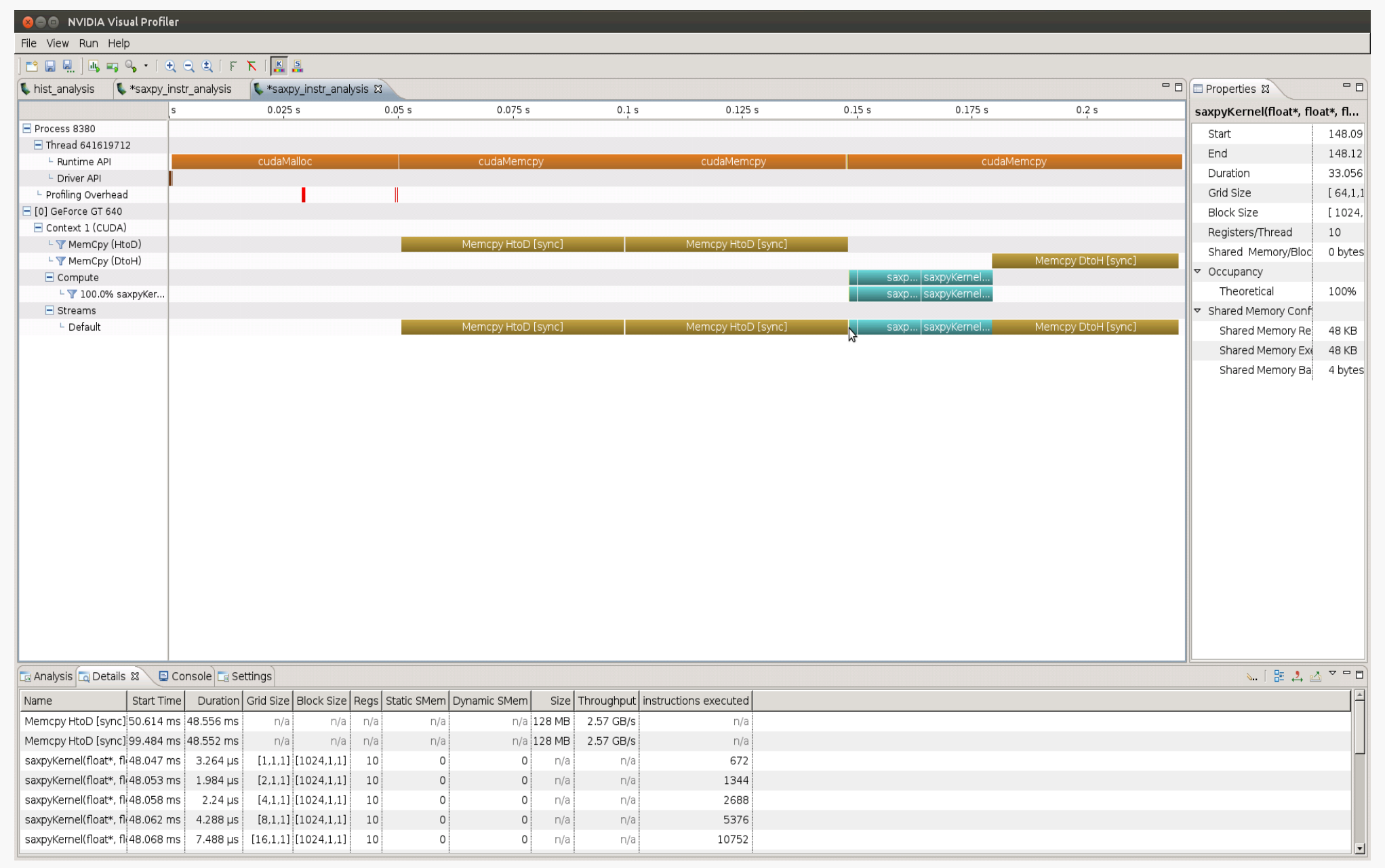
PAPI CUDA component is a C profiling library built on top of CUPTI. It provides the low level functionality of CUPTI with more built in features.

Visualization

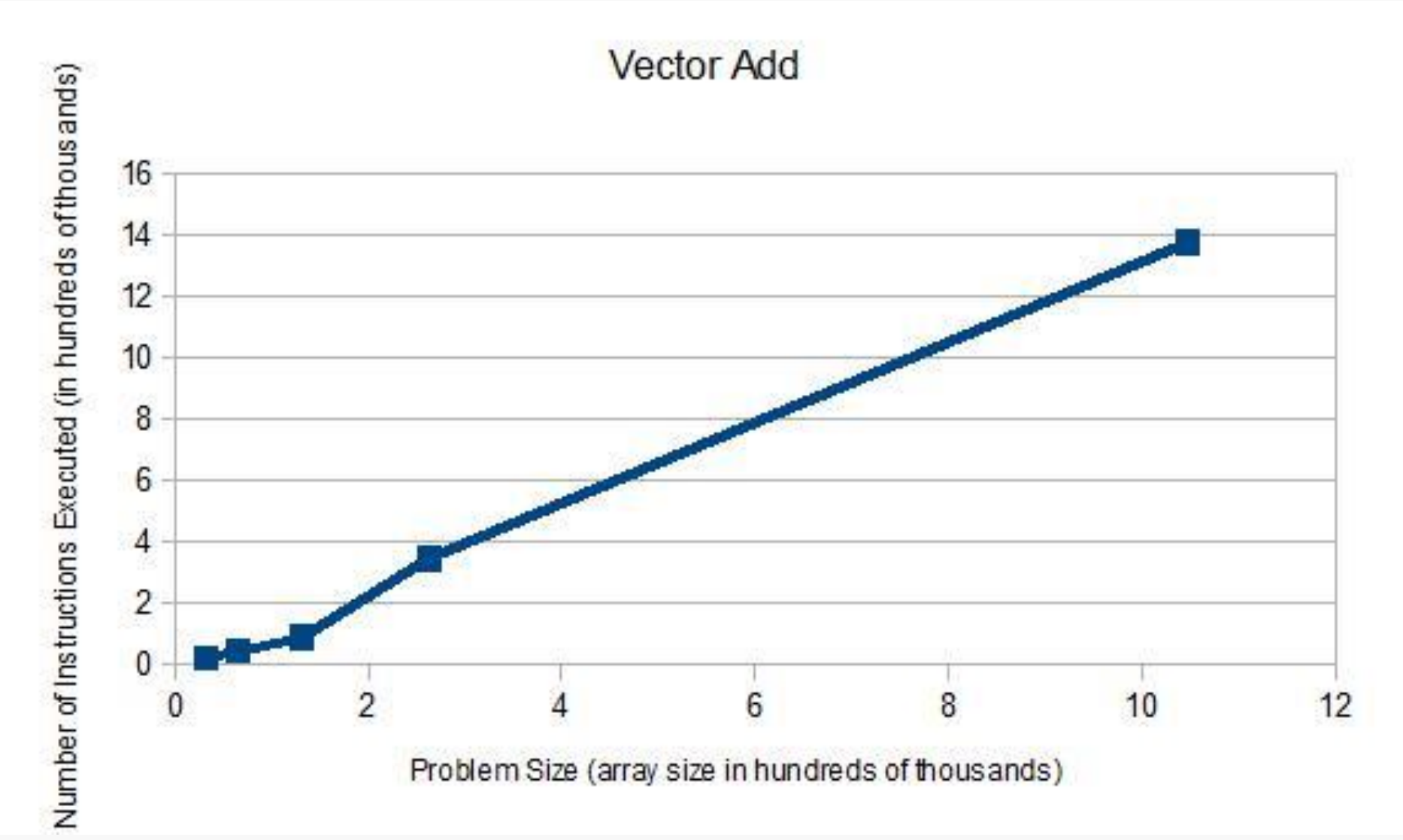


This figure shows a timeline for a GeMTC histogram application generated by NVVP. It clearly display asynchronous memory transfers at the top and the kernel execution in the middle. The bottom of the screen also shows data such as throughput for each memory transfer.

This figure shows a timeline for a Saxpy application that launches 15 saxpy kernels of varying sizes. This clearly illustrates the fact that although there are multiple kernel launches, data movement still takes the most time. The bottom also shows some metrics such as number of instructions executed per kernel.

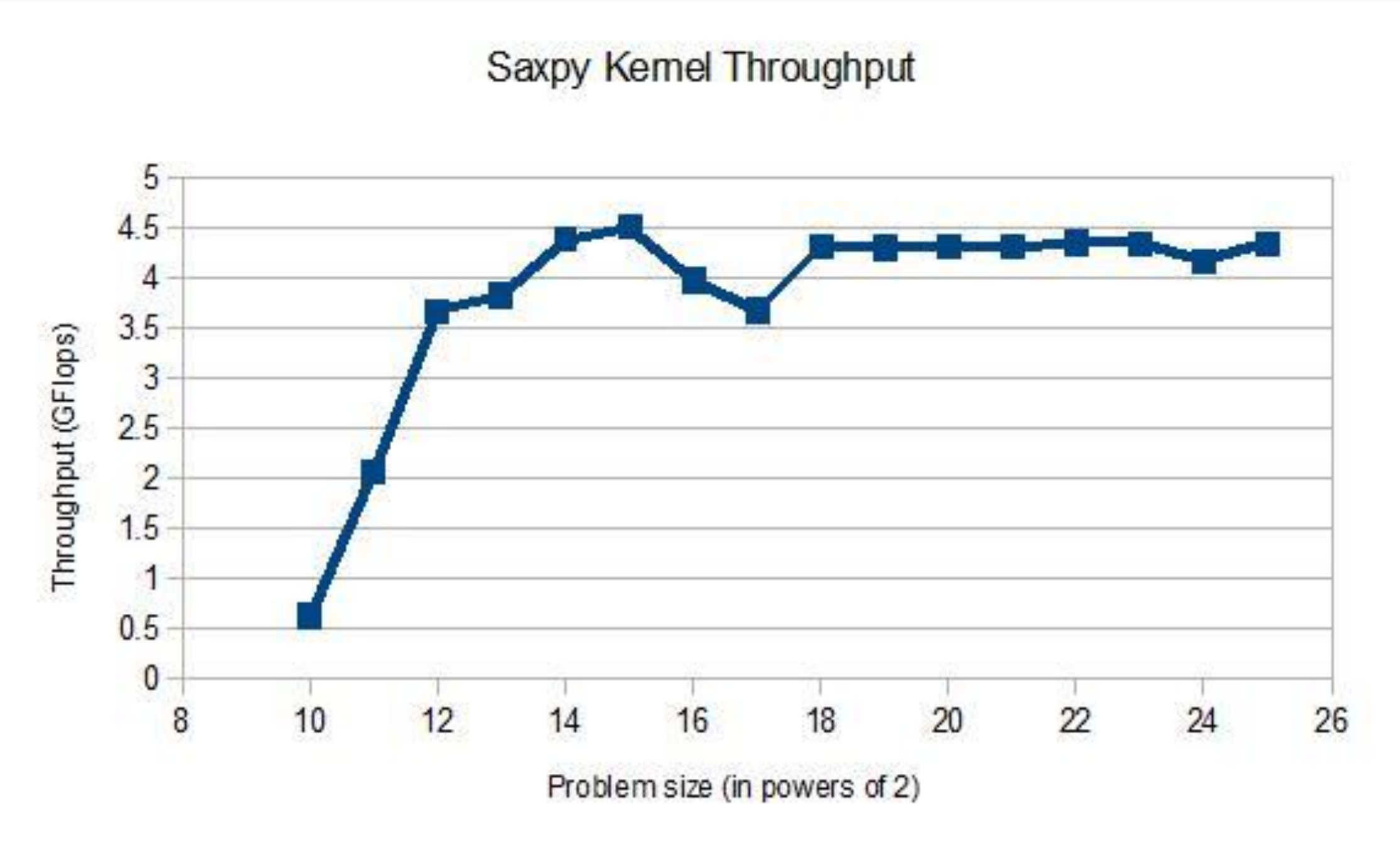


Metrics



This figure shows the instructions executed metric collected using PAPI CUDA for a Vector Addition kernel. The data was collected for problem sizes from 2^{15} to 2^{20}. This figure is an example of the data, other than visualization, that can be collected with NVVP.

This figure shows throughput calculated using the duration (collected in NVVP) and the flops (collected in CUPTI). This is an example of the type of metrics that can be collected in CUPTI and NVVP. NVVP can collect most of the same metrics much more easily.



Profiler Comparison

Profiler	Pro	Con
NVIDIA Visual Profiler	Ease of Use	Limited Functionality
CUPTI	Functionality	Difficult setup
PAPI	Functionality (slightly better)	Difficult setup

Conclusions

- NVIDIA Visual Profiler (NVVP) is very easy to use and fairly featureful. Only requires a compiled binary (no code modification)
- CUPTI is complicated to use, but it provides direct to access to all hardware counters.
- PAPI CUDA component is a slightly more featured version of CUPTI, but is more complicated to use.
- Overall, NVVP seems to be the best choice due to that fact that it easy to use and has most of the features that the others profilers have.

Future Work

Future work includes an in-depth study on GeMTC framework. This future work would leverage the tools surveyed in this work to evaluate the efficiency of GeMTC warp workers. Currently, it is unknown how much time is spent fetching applications versus actually running them. Other future work includes doing a similar survey for tools that visualize and evaluate applications on the Intel Xeon Phi coprocessor.