Ultra low latency trace offloading

Keywords : performance analysis, HPC, exascale

Context

High performance computing has become the cornerstone of many scientific fields ranging from medical research to climatology. Efficiently exploiting a supercomputer is difficult. Recent developments in the software and hardware stacks of supercomputers have made them difficult to operate. Due to the complexity of hardware architectures, numerous effects have a strong influence on performance (NUMA effects, cache effects, vectorization, etc.). In addition, supercomputers are now heterogeneous and applications now have to exploit several types of processing units (CPUs, GPUs, ...), increasing the number of component that may degrade performance.

Developpers thus rely on performance analysis tools to assist them in optimizing their application [1, 2, 3]. These tools collect performance data during the execution of the application and generate trace files that can be analyzed post-mortem. However, when running an application at a large scale (typically thousands of CPUs), each thread will generate performance data, and the resulting trace will be so large that it cannot be processed.

A prototype [4] trace format exploits the repetitive nature of most parallel applications [5][6] to perform on the fly data compression. It detects at runtime sequences of events that repeat and replaces them with meta-events. As a result, each thread of the application is described by a *grammar* (see Fig. 1) that depicts the sequences of events that occured. Moreover, the timestamps of each event, and the durations of each sequence are stored in separate files in order to analyze the performance of the application. This prototype can summarize the behavior of an application. It scales as the number of events per thread increases.

$$\begin{aligned} R &\mapsto AB^2A \\ A &\mapsto ab \\ B &\mapsto bc \end{aligned}$$

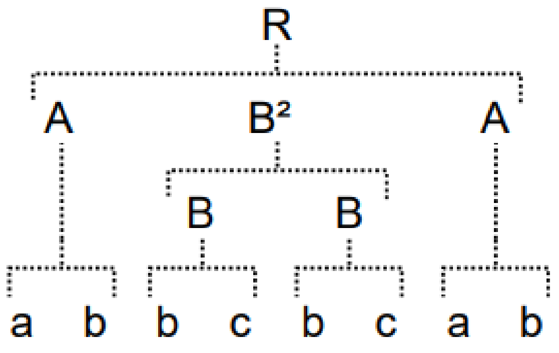

Fig. 1: Unfolding of a grammar representing the trace “$abbcbcab$”.

Research project

When collecting traces, it is important to avoid altering the application performance. The recording of events shall be as fast as possible. Typically, we target a recording overhead of around 100 ns at most. However, some features require processing time: detecting repetitive sequences of events, compressing timestamps, etc. These processing make it difficult to keep the overhead below 100 ns, or we have to use simple heuristics to reduce the processing cost.

But modern servers have losts of cores or hyperthreads (typically 32 cores), and applications don't use all of them. For example, HPC applications usually don't use hyperthreading, leaving almost half of the processing capacity unused.

The goal of the project is to use these unused processing capacities to offload time-consuming features of a tracing tool. When recording an event, an application thread could just copy a few bytes to a buffer. Then, an "offload" thread could process this data and perform time consuming computation (detecting patterns, compressing data, writing buffers to disk, etc.). This would limit the degradation of the application performance, while allowing to perform extensive event analysis computation.

The main steps of this project are:

- Study the HTF tracing library [4]
- Implement a low-latency producer-consumer system
- Evaluate the proposed system on various real life applications

Contact

François Trahay francois.trahay@telecom-sudparis.eu
Benagil group
Télécom SudParis

References

[1] Geimer, M., et al. "The Scalasca performance toolset architecture." Concurrency and computation: Practice and experience, 22(6), 702-719, 2010.
[2] F. Trahay, et al. "EZTrace: a generic framework for performance analysis" CCGrid 2011.
[3] Shende, S. S., & Malony, A. D. "The TAU parallel performance system." The International Journal of High Performance Computing Applications, 20(2), 287-311, 2006.
[4] https://github.com/trahay/Hierarchical-Trace-Format
[5] F. Trahay, et al. "Selecting points of interest in traces using patterns of events." PDP 2015.
[6] Colin, A., et al. "PYTHIA: an oracle to guide runtime system decisions." CLUSTER 2022.