Pricing Python Parallelism
Guided JIT compilation for Heterogeneous Architectures

Dejice Jacob

School of Computing Science, University of Glasgow

January 29, 2020

Abstract

Dynamic scripting languages, like Python, are increasingly used by end-user non-expert developers to write computationally intensive code. GPU accelerators can provide orders of magnitude speed-ups compared to interpreter execution for such code. Compute accelerators are increasingly commoditised and common-place. GPU accelerators are most commonly programmed by writing computational kernels in CUDA and OpenCL. These are dialects of C-like languages that specifically target the massively parallel architecture of these devices. Achieving the promised speed-ups requires a deep understanding of the parallelism model of a GPU accelerator.

ALPyNA [1, 2] is our loop parallelisation framework for Python in a heterogeneous environments. It performs classical dependence analysis and parallelisation of loop nests written in plain Python. However, unlike classical optimisation, ALPyNA stages dependence analysis between static and dynamic phases. Staging analysis in this manner provides a light-weight framework for JIT compilation at runtime.

ALPyNA uses Python's introspective capabilities to JIT compile loop nests derived from the actual dependences and loop domain sizes for each execution instance of a loop nest. This hybrid approach enables us to check assumptions at runtime and parallelise aggressively compared to a static compiler. The GPU code generated at runtime for each loop nest instance is tailored to the dependence structure that emerges at runtime. We evaluate performance results across 12 loop intensive benchmarks. However, in many cases, JIT compiling embarrassingly parallel benchmarks for the GPU does not yield a performance speed up. GPU performance is affected by thread scheduling costs, resource limitations on the GPU and data transfer overheads between the host and the GPU.

We introduce a lightweight cost model that adapts to ALPyNA's runtime analysis and custom GPU code generator to guide the selection of the appropriate device to target for JIT compilation in a heterogeneous environment. This analytical cost model is parameterised on the physical hardware characteristics of the CPU and the GPU. It also eliminates the need for extensive profiling requirements. We evaluate the accuracy of the cost model over 12 loop intensive benchmarks (e.g. Fig. 1).

References

[1] Jacob, D., and Singer, J. Alpyna: Acceleration of loops in Python for novel architectures. In *Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming* (2019), p. 2534.

[2] Jacob, D., Trinder, P., and Singer, J. Python programmers have GPUs too: Automatic Python loop parallelization with staged dependence analysis. In *Proceedings of the 15th ACM SIGPLAN International Symposium on Dynamic Languages* (2019), p. 4254.

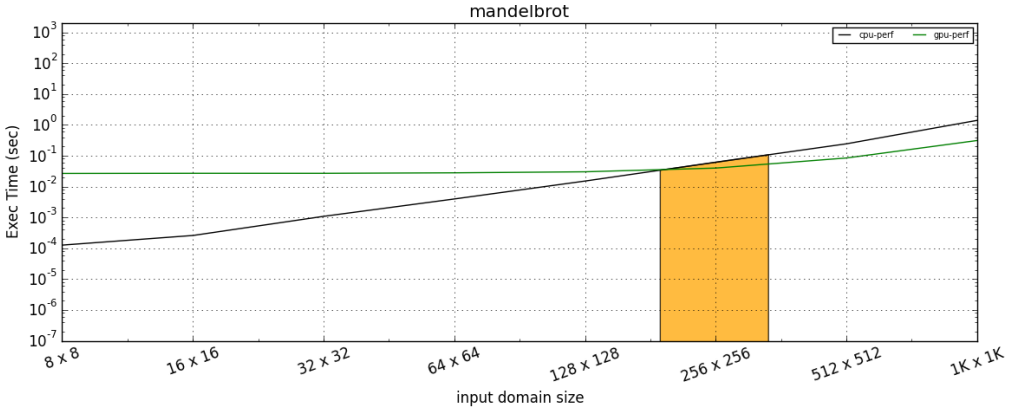

Figure 1: Comparison of execution time for different input sizes on two target JIT backends (CPU and GPU) for Mandelbrot. The ALPyNA cost model selects the optimal backend in all cases except for the range of domains highlighted by the shaded orange region.