

Generic Autotuning Technology for GPU Applications

7 – 11 April Lorentz Center@Oort

Graphics Processing Units (GPUs) are enabling and accelerating many applications in science and industry. The performance of GPU applications strongly depends on how the software has been optimized for the hardware. Developing highly-optimized GPU applications that can be auto-tuned involves creating many different implementations of the same program through parameterizations. Together these parameters create vast, non-convex, non-continuous design spaces that are infeasible to search by hand.

Generic auto-tuners aim to bring a universal solution, which can be used for different libraries, applications, as well as code produced by high-level programming languages or domain-specific languages. Several generic auto-tuners have arisen in recent years and are currently undergoing active development, each with their own merits and specific focus, seeking to provide a generalized solution to the auto-tuning problem.

There is an urgent need for this technology as GPU architectures are becoming increasingly heterogeneous, and therefore becoming even more difficult to optimize, and because the GPU market is rapidly diversifying after a long period of relative stability, further increasing the need for tools that can deliver performance portable code.

The aim of this workshop is to bootstrap international collaboration between research groups working on auto-tuning in different fields and create collaborations between research groups in auto-tuning and research groups in high-level programming languages and compilers.

Organization and Format

The workshop was proposed to take place in Snellius, but under Covid-19 measures only 16 participants could be present in person in Snellius. Fortunately, the Oort location became available to us due to another workshop being canceled. This meant we could use the Oort location with 25 participants present in person under 1.5 meter distancing rules. The workshop was organized as a hybrid workshop, mostly as a backup in case some participants could not travel at the last moment. In the end this meant we had 24 on-site participants and 2 participants joining remotely. Despite the relatively low number of participants joining online, this format turned out to work really well.

The workshop featured a number of presentations and plenary discussion sessions. On Tuesday we formed groups and for most of Wednesday and Thursday people worked on separate topics in four parallel working groups. On Friday, we came together again to discuss the progress made by the different groups.

Summary of the parallel working groups

T1 User Interfaces

T1 focused on open problems in autotuning, definitions and terminology, features comparison of current tools, and common file format for autotuning. The members of the group decided to look at open problems from the point of view of autotuning users, and not necessarily experts in the field, and identified the following challenges: (1) lack of clarity on the steps needed for autotuning, (2) dependence on a particular tuning framework, (3) minimum input necessary for the process, and (4) dependency between performance and workload.

On the terminology topic, T1 produced a document containing the terms that are used in our tools, focusing on what is important for the user, with the goal to reduce ambiguity and create a common vocabulary. The discussion on features comparison, focused mainly on the user facing APIs of Kernel Tuner, KTT, HyperMapper, and ytopt, resulted in the categorization of all features into four categories: (1) kernel specification, (2) search, (3) configuration space, and (4) general options. These categories have been used as the starting point of a common autotuning file format. The team is currently meeting every other week, working on the formalization of the first draft of the file format.

T2 Programmability

Specifying low-level, parameterized schedules, which serve as input for auto-tuning tools, is time consuming and requires expert knowledge a programmer might not possess. In WG T2, we propose therefore to facilitate this process by automatically generating these schedules via more abstract high-level schedules. Those high-level schedules can be simple instructions to optimize for spatial or temporal locality.

This can happen in multiple ways: (1) compiler of high-level parallel languages, such as CUDA-CHiLL, MDH, Accelerate, SAC, SkePU, Futhark or similar can generate these schedules, possibly with some input from the user to prune the search space. (2) generation via a DSL, such as Brick and ImageCL. (3) High-level code constructs may generate parameters feeding directly into the interfaces (e.g., RTE+RRTMG). In the proposed framework, expert programmers can still provide the hand-written low-level schedules, similar to the schedules in FireIron, Halide and TVM.

T3 Search methodology and metrics

Comparing the efficiency of autotuning space searchers is not trivial. A commonly used method in the field of searching/mathematical optimization is comparing the number of search steps (searched function evaluations). However, with autotuning, the cost of the evaluation highly depends on the search method. E.g., the surrogate model is evaluated faster than experiments made by searchers based on mathematical optimizations (but the surrogate model, on the other hand, can require re-building with new hardware or input). The cost of empirical evaluation steps also depends on how well the searcher biases search towards faster implementations and the tuned code's input size.

In T3, we systematically mapped how the results of searchers' comparison can be biased and designed a robust methodology, allowing fair and transparent comparison of the searchers. We defined which data has to be published with experimental results and proposed how to present them. The method will be published in a paper and demonstrated on several benchmarks using multiple autotuners. The team is currently meeting every other week, to collaborate on a paper about the methodology.

T4 FAIR data sharing

In T4, we have written a proposal for an online data sharing platform that would allow FAIR data sharing of auto-tuning benchmark data. After going through related initiatives, we have identified the gap in current state of the art, and defined the requirements of such a data sharing platform, in particular with the regard to make data FAIR. Building on the search space definitions that will be produced by the T1 group, and using the requirements provided by the T3 group, we have created a clear overview of the necessary information that should be recorded by authors of data sets when these are contributed to the online data sharing platform.

Scientific outcomes

In short, the following outcomes have been created or initiated at the workshop:

- Glossary / vocabulary of auto-tuning terminology
- Taxonomy of auto-tuner API features
- A common format for configuration spaces
- Formulation of High-Level Schedules
- A paper on a unified methodology for benchmarking search methods
- A proposal for FAIR sharing of auto-tuning data
- Several ideas for follow-up workshops
- Bootstrapped collaborations between different groups

Several of the parallel working groups have set up biweekly meetings and are coming together on a regular basis to continue discussions and work on the papers, software, and data specifications that were initiated at the workshop. Several participants have expressed interests to organize a follow-up meeting. These collaborations would not have been created so fruitfully without the Lorentz Center workshop. We as organizers are very happy with the impact the workshop has had on our work and collaborations.

Ben van Werkhoven (Amsterdam, The Netherlands)

Jiří Filipovic (Brno, Czech Republic)

Ari Rasch (Münster, Germany)

Gabriele Keller (Utrecht, The Netherlands)