An overview of GPU Computing Research in the Netherlands

Results from the NIRICT GPGPU Reconnaissance workshop

Marieke Huisman, *University of Twente*
Ana Lucia Varbanescu, *University of Amsterdam*
Ben van Werkhoven, *Netherlands eScience Center*
Anton Wijs, *Eindhoven University of Technology*

uni.lu UNIVERSITÄT LUXEMBURG

Map shows organizations that participated in the workshops by presenting their work

An overview of GPU Computing Research in the Netherlands

Results from the NIRICT GPGPU Reconnaissance workshop series

The goal of our *NIRICT GPGPU Reconnaissance* project was to understand the breadth and depth of the GPGPU research actively done in The Netherlands. Thus, we set out to provide all the stakeholders - beneficiaries and designers/developers alike - a platform to showcase their work. Based on these showcases, we can now sketch a landscape of the Dutch GPGPU research. This document presents this landscape, but it also goes beyond it by providing a vision on how GPGPU research can be strengthened in the near future. Ultimately, this vision should help the integration of GPGPU research in the national and international research roadmaps.

Motivation

Graphics processing units (GPUs) were initially designed to support computer graphics. Their specific architecture provides high processing power and massive parallelism, which allows for efficient solving of typical graphics-related tasks. However, it was soon realised that such architectures are also suitable for many other computational tasks, which led to the development of the area of GPGPU (General Purpose GPU) programming. As a result, GPUs are now used in many different fields, including numerical simulation, media processing, medical imaging, eye-tracking, genomics, fluid dynamics, and machine learning.

Complex GPU applications typically need to be programmed at a relatively low level, since they require expert knowledge on the specific hardware used. Given the importance, range and increasing complexity of GPGPU applications today, the main research challenge for GPGPU programming is that we urgently need systematic techniques that can optimise given GPGPU applications without requiring expert knowledge. This would allow GPGPU applications to be developed at a more convenient higher abstraction level. These techniques should offer a predictable performance increase, while preserving the original behaviour.

Research on performance and correctness of GPGPU applications in the Netherlands is spread around, without much collaboration. However, this is a fast growing area, with high potential for important results and high visibility. Therefore, we organised a series of workshops to establish closer connections between the different researchers working in this area, with the goal to strengthen Dutch research on performance and correctness of GPGPU applications.

This report is based on the outcomes of these workshops, providing an outlook on how to further develop research on GPGPU-related topics. as well as discussing how this research can

be added to the national and European research agendas. The programs for the different workshops are available at http://fmt.ewi.utwente.nl/NIRICT_GPGPU/events.html.

Geographic spread

Research in GPU Computing is carried out throughout the Netherlands. TU Eindhoven has a long history of research in verifying the correctness of software designs by means of model checking. In recent years, Anton Wijs is conducting research on using GPUs to speed up model checking, as well as using model checking to verify GPU programs. At the University of Twente, Marieke Huisman is leading a research group on the topic of software reliability. One of her research interests is the software verification of concurrent programs that include GPU code. At the University of Amsterdam, Ana Varbanescu is researching performance models and portability of heterogeneous computing applications. The Netherlands eScience Center is performing research in testing and auto-tuning GPU applications, which is then applied in the development of GPU applications in many domains, including radio astronomy, microscopy, digital forensics, and oceanography.

It is very good to see that many other researchers and commercial parties are involved in GPU Computing research. During the 4 meetings, over 40 different persons participated from more than 10 different organisations, such as the 4 Dutch Technical Universities (with multiple groups participating from each of these universities), the University of Amsterdam, VU Amsterdam, the Netherlands eScience Center, Astron, CWI, University of Leiden and companies such as TomTom, StreamComputing, and ORTEC Finance.

Kick-off Meeting

Funding for the reconnaissance action was provided by NIRICT, the Netherlands Institute for Research on ICT. Therefore, a first kick-off meeting was organised with key participants from the 3 technical universities involved in NIRICT at that time. All participants introduced themselves and their research, and a global discussion on the different aspects of GPGPU research was held. Based on this discussion, a division in three main areas of interest was identified: support systems for GPGPU programming, GPGPU applications, and analysis of GPU applications, and it was decided to organise follow-up workshops on each of these themes.

W1: Support systems for GPGPU programming

In the first of the three workshops, we have heard of several approaches focusing on supporting GPGPU designers and programmers to improve productivity and the performance of their code. We include here a very brief descriptions of the tools and solutions presented during the workshop.

Lessons Learned

A wide range of GPU programming and analysis tools are being researched in the Netherlands. The presentations covered source-to-source compilers, including Bones (van den Braak et al.) and MCL (Hijma et al.), heterogeneous execution frameworks, like HyGraph (Heldens et al.) and Risk-Neutral Modeling (de Geus), and performance analysis tools, Kernel Tuner (van Werkhoven), BlackForest (Madougou et al.), and Tuning Complexity (Sclocco et al.).

The presented systems can be classified in three (somewhat overlapping) classes: synthesis, tuning, and analysis tools. Synthesis tools are those that generate GPU code from higher-level representations, and primarily target a good trade-off between productivity of the front-end and high-performance of the back-end. Tuning tools are systems that (help) improve existing code with little intervention from the end-user, and primarily target performance. Finally, analysis tools provide means to better understand the performance of GPUs and GPU applications, and primarily target some form of predictability.

Despite the diversity of the topics included in this workshop, several directions of research have seen little to no representation. Our brainstorming session at the end of the workshop identified the following “absentees”: high-level and/or domain-specific programming models for GPUs, GPGPU simulators, benchmarks and microbenchmarks, and approaches, studies, or tools directly focusing on energy efficiency.

The way forward

Our immediate action plan is to investigate whether the research topics missing from our best-effort overview are indeed not represented in the Netherlands, or we have missed an important part of the community.

On the longer term, a thorough analysis is needed to assess which of these tools and systems could become flagship tools, and whether we can actively help in their promotion and adoption at both national and international level. The adoption of MCL by NLeSC is a good example of a success story, and we hope we can encourage both developers and users of such systems to join us.

Finally, we envision an active role for our community to foster collaboration, based on these systems, with industrial partners. A good example is the Risk-Neutral Modeling framework, which is based on a framework developed as part of a MSc thesis research.

W2: GPU Applications

At the GPU Applications workshop work was presented on applications from various scientific application domains, including microscopy, gene sequencing, astrophysics, linear algebra, tomography, visualization, and radio astronomy.

Lessons Learned

The discussion during the workshop was focusing on the lessons that can be learned from the discussed applications work. The most important step in creating GPU applications from existing software is to pick the right starting point. In general, we see that a lot of good software practices are thrown out of the window for the sake of high performance. However, we believe that good software development practices and high performance can certainly co-exist. Unnecessary and premature optimisation make it even harder to create a GPU implementation out of a sequential application.

Another critical point is to confirm that the existing code is actually capable of solving the problem at hand. Often, GPU applications are created to target a larger problem at a higher resolution, or a new, bigger, data set. Therefore, it is crucial to first assert that the prospected algorithm actually works at that larger scale. This may sound trivial, but experience teaches that this is not always the case.

The way forward

Applications remain a driving force behind the adoption of GPU computing at larger scale. As seen also during our workshop, a broad set of applications is already successfully exploiting the performance of GPUs.

In order to actively support the development of more and more success stories with GPUs and challenging applications, we believe we must encourage those interested, especially when coming from outside the computer science domain, to use our knowledge and expertise to their advantage.

Our idea therefore is to start an online community, with people and stories, where we can help out these potential application developers with on-line tutorials and materials. Moreover, given the significant teaching expertise available in this consortium, we consider organizing application development crash-courses in GPU processing for different audiences, ranging from students to professionals.

W3: GPUs and program analysis

The third workshop gave an overview of work on the analysis of GPU applications. We had two invited speakers: Alastair Donaldson (Imperial College London) who talked about how to expose errors related to weak memory in GPU applications, and Qixia Yuan (University of Luxembourg) who talked about how to use GPUs to accelerate steady-state computation of large probabilistic boolean networks.

Lessons Learned

In addition to the invited speakers, we had several presentations by Dutch researchers working on program analysis. The presentations could be divided into two groups: how to use program

analysis techniques to analyse GPU applications, and how to use the power of GPUs to analyse applications.

Several tools exist that allow one to analyse and further optimise GPU applications. They differ in the ease of use, and in the level of guarantee that they provide. For example, the Kernel Tuner provides a way to quickly and easily test GPU code, whereas the VerCors tool set requires a high number of annotations in the code, but can then produce a high level of assurance on the correctness of the application. When developing such tools and techniques, challenges are to integrate these into the GPU application development cycle smoothly, and to ensure scalability of the techniques.

At the same time, we see that GPUs are becoming effectively used for the analysis of other applications and systems. Adapting model checking techniques for GPUs results in substantial speed up, and we see that this will give a new push to the model checking community.

The way forward

We can see that the ongoing work in developing analysis tools that analyse GPU code or use GPUs for analysis of code in general differ in their target audience. Some tools have been developed as instruments of computer science research, while others are developed in a way that they are ready to be used in-production in GPU application development. To allow GPU application developers to take advantage of the advancements in GPU research, the tools that have been created for research can be developed further supported by tutorials and extensive documentation. Furthermore, tools that provide a high level of correctness assurance often require still a lot of work from the user. This process needs to be further automated, in order to increase its usability.

Landscape Summary and Roadmap

GPGPU research in the Netherlands is currently spread around, without too much collaboration. However, things are improving. Thanks to this series of workshops, we have established much closer connections and already some new collaborations have been initiated.

We can roughly group the challenges that GPGPU researchers in the Netherlands are working on into three main challenges:

1. The programmability problem
2. The performance optimisation problem
3. The application correctness and verification problem

These challenges are not likely to be solved by a single system and not for all application domains and scenarios. However, there are many research efforts that successfully demonstrate to tackle one of these challenges in certain situations. Therefore, we must take action to transfer these successes to the application developers, allowing them to use and build on this knowledge, experience, and expertise. This requires an investment in translating

research instruments into building production-ready and generic tools for software development, optimisation, and verification.

We have started to create a Dutch GPU community to foster more collaborations and share the significant expertise on GPGPU research with each other and with potential GPU application developers, which includes both students and professionals. We expect that this will result in more collaboration between researchers, and between researchers and application developers, such that researchers can learn about the problems actually encountered by application developers. We believe that this growing community will therefore accelerate the process to transfer the results from the GPGPU research community to the application developers.