Heterogeneous Computing

Nathanael Huber
01518376

Benjamin Walch
01518922

June 2, 2017

Abstract

Speed, flexibility and energy efficiency became important things (if not the most important things) in developing new computing components. However, with computers of today, it is not possible to satisfy all factors equally. If the main goal is speed, then you either have losses in flexibility or energy consumption. If the main goal is energy efficiency, there will be speed reductions. The same applies for flexibility.
Heterogeneous computer systems decrease or close the gap between these factors.

This paper gives an introduction about heterogeneous computing. In particular, we explain what it is and have a look at the hardware options of today. Challenges for both the software and the hardware are discussed, and in a short outlook we learn how heterogeneous computing will affect software development in the future.

1 Introduction

A heterogeneous system is a parallel system which consists of different computing nodes, each with its own advantages and drawbacks. In such a system, multiple nodes are combined to get all advantages of each. Therefore it's the opposite of a homogeneous system, consisting of similar cores (single-core systems are always homogeneous). Heterogeneous systems became more common at the beginning of parallel programming and will be replaced by heterogeneous systems in the future.

Zahran [1] distinguishes 3 types of heterogeneity:

The first type consists of cores with the same capabilities but different DVFS (dynamic voltage and frequency scaling). That means computing effects them differently, which changes their behavior.

The second type are Cores with different architectural capabilities. For example superscalar processors are of this type[1].

The third type of heterogeneity computing nodes contain cores with different execution models. Several different types of nodes exist here, which are explained later. Examples are GPU's and CPU's.

Processing units aren't the only systems which can be heterogeneous, another example are memory modules like caches (SRAM), volatile memory (DRAM) and nonvolatile memory (MRAM, STT-RAM, PCM, ReRAM).
In the following section, we will keep the focus on computing nodes with different execution models.

2 Computing Nodes

Bacon [3] mentions that there are two options where hardware can influence computing performance: There are GPPs (general-purpose processors) on one end of the spectrum and ASICs (application-specific integrated circuits) on the other.

Field Programmable Gate Arrays (FPGAs) is a technology which exists between those two extremes.

---

[1] Superscalar processors can execute multiple instructions per clock cycle, even with only a single core, this is called instruction-level parallelism[2]

2.1 Graphics Processing Unit

For Singh [4], a GPU is the most popular heterogeneous computing resource today. It provides a high aggregate memory bandwidth and has the ability to perform many more data-parallel operations than a conventional processor.

Expressed differently, the GPU operates using a single-instruction, multiple data execution model. A good example for this would be multiplying a constant with each element of a large matrix, where a GPU is the true winner compared to a CPU.

2.2 Field Programmable Gate Arrays

Marchal [5] defines FPGAs as an array of logic blocks (cells) placed in an infrastructure of interconnections, which can be programmed at three distinct levels.

In other words, FPGAs are small circuits which can be programmed to implement a certain function. The main advantages of this technology are, beside their outstanding performance[2], that these circuits are configurable, either once or multiple times, meaning that this units have a (limited) level of flexibility. It's fact that everything what can be done with software, can also be implemented with hardware. The software solution provides you with flexibility on the one hand, on the other hand a hardware solution is a lot faster. As a result, we can say that FPGAs decrease or close the gap in computing between flexibility and performance.

Consider a conventional GPP used to calculate some complex operations. While GPPs are highly programmable, they are often inefficient in terms of power and performance. FPGAs instead, can be configured or programmed at hardware-side for specific tasks, which means that they don't have as much flexibility, but provide a lot more efficiency.

2.3 Automata Processors

We know that FPGAs can be programmed and therefore it is obvious that other technologies were built on top of it. We are talking about an other model, the Automata Processor, which is built using FPGAs. An Automata Processor is a hardware-based accelerator of non-deterministic finite automata (NFA) [6]. A good example for such a non-deterministic finite automata is a regular expression, where a higher performance from an AP can be expected than using a traditional core or a GPU.

---

[2] in terms of execution time

3 Hardware Challenges

The main problem here is the bottleneck between the processing unit and the memory (as it can bee seen in figure 1), while this effects single-core systems too, this is especially significant within heterogeneous systems because of the memory they have to share. Doing so requires a memory hierarchy which reduces interference between the different cores, and deals efficiently with the different requirements of each [1].

This was caused, because processing units benefited the most of research done, in this area, and therefore followed Moore's Law[3] until a few years ago, while memory did not.

Other challenges are the interconnection between different cores and memory modules and the efficient distribution of the workload between the processors.

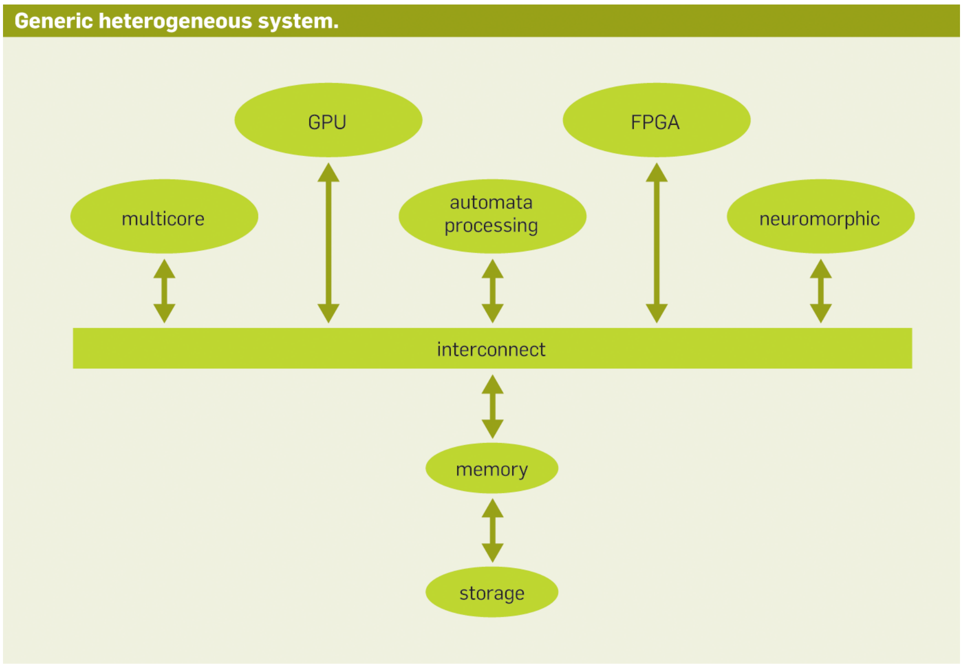

Figure 1: Generic heterogeneous system[1]

---

[3] Moore's Law states that the number of transistors in a dense integrated circuit doubles approximately every two years [7].

4 Software Challenges

Heterogeneous Computing, requires programmers to think about certain new aspects:

- Productivity: Applicators need to be split into multiple small threads (or processes), which should be assigned in consideration of advantages and drawbacks of cores. Functions have to adapt to the availability of Systems like CPU's, GPU's or FPGA's. All this decreases the performance of a programmer and could be solved without such disadvantage by a new programming model [1] [4].
- Scalability: Heterogeneous Systems need to be scalable, the performance of a application should increase if more cores are added. There is of course a limit as seen in figure 2 where the same application was tested on two different GPUs and a CPU. By increasing the cores at each test run, the CPU didn't improve after reaching a kernel size above 40, while the GPU's did.
- Reliability: If the hardware or system software does the error-handling, the programmer will be more productive [1], but the programmer could handle faults more specific and efficient.
- Portability: Applications should run on different heterogeneous systems, independent of the existence of certain components.
- Interconnection: How should the different cores and memory hierarchy modules be connected?

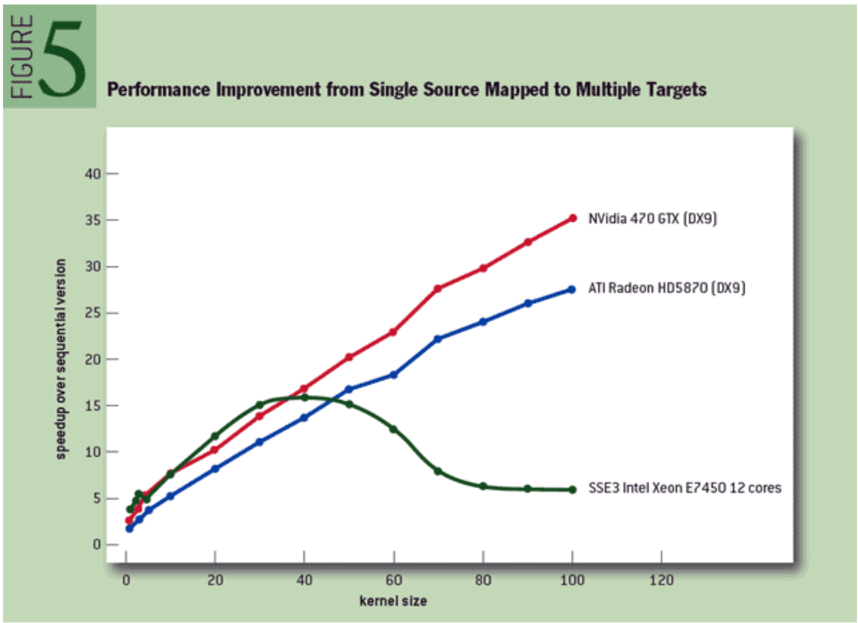

Figure 2: Performance Improvement from Single Source Mapped to Multiple Targets [4].

5 Outlook

To make the best use of the opportunities provided by heterogenous computing nodes will require to revisit the whole computing stack. Computation is now much cheaper than memory access [1]. Compilers will need to learn how to use heterogeneous nodes. A different programming language and a very different model of execution is needed in order to keep programming efficient, because current models focus mainly on CPU's. There are some options like OpenCL (Open Computing Language) for example which are heading in the right direction. The new model needs to efficiently map the same computation onto several different computing elements like GPU's or FPGA's. The Accelerator system already provides this and can be used in C, C++, or any language that supports a C calling interface, e.g. C#, F#, or Haskell [4]. It improves computation because the programmer needs no knowledge about the underlying architecture, the Accelerator system can adapt algorithms accordingly.

Heterogeneous computing is already there, and there is no question about that this technology will affect future hardware implementations for computers.

References

[1] M. Zahran, “Heterogenous Computing: Here to stay,” *Communications of the ACM*, vol. 60, no. 3, pp. 42–45, 2017.

[2] M. Johnson, *Superscalar microprocessor design.* Prentice Hall, 1991.

[3] D. F. Bacon, R. Rabbah, and S. Shukla, “FPGA Programming for the Masses,” *Communications of the ACM*, vol. 11, no. 2, 2013.

[4] S. Singh, “Computing without Processors,” *Communications of the ACM*, vol. 54, no. 8, 2011.

[5] P. Marchal, “Field Programmable Gate Arrays,” *Communications of the ACM*, vol. 42, no. 2, 1999.

[6] M. Nourian, X. Wang, X. Yu, W. chun Feng, and M. Becchi, “Demystifying automata processing: GPUs, FPGAs or Micron’s AP?,” in *Proceedings of the International Conference on Supercomputing*, p. 1, ACM New York, June 2017.

[7] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, Apr. 1965.