Unparalleled Parallelism?

CPU & GPU Architecture Trends and Their Implications for HPC Software

Laura Morgenstern
laura.morgenstern@informatik.tu-chemnitz.de
Chemnitz University of Technology
Chemnitz, Germany

Ivo Kabadshow
i.kabadshow@fz-juelich.de
Jülich Supercomputing Centre
Jülich, Germany

Matthias Werner
matthias.werner@informatik.tu-chemnitz.de
Chemnitz University of Technology
Chemnitz, Germany

ABSTRACT

The free lunch is over – again? In 2004, Herb Sutter observed the stagnation of clock frequencies and predicted hyperthreading and multicore capabilities as drivers for performance growth on CPUs. This prediction and the resulting advice to focus more on concurrency to achieve sustainable application performance, has become the daily reality of HPC software engineers. In this paper, we compare trends in the development of CPU and GPU architectures and examine their implications for the parallelization and portability of HPC software. The data analysis still reveals levelling clock frequencies but this time also for GPUs. Additionally, an increasing amount of hardware parallelism can be observed for both architectures.

CCS CONCEPTS

• **Computer systems organization → Parallel architectures**.

KEYWORDS

CPU, GPU, multicore architectures, GPGPU

1 INTRODUCTION

The goal of this work is to quantify trends in the architectural design of CPUs and GPUs and their implications for the parallelization and portability of HPC software. For this purpose, we consider the development of clock frequency, compute unit count and compute unit size for CPUs and GPUs over time. Based thereon, we describe how HPC software should reflect these developments. On a much smaller scale, we take first steps to apply the idea of [5] to GPUs.

FGBS '21, March 11–12, 2021, Wiesbaden, Germany

https://doi.org/https://doi.org/10.18420/fgbs2021f-02

2 PARALLEL ALGORITHM MODELS

To describe implications for HPC software, this work differentiates the parallel algorithm models data-parallelism and task-parallelism.

We refer to data-parallelism as the parallel execution of identical operations on multiple data elements. In the scope of this work, this refers to SIMD (Single Instruction, Multiple Data) vectorization on CPUs and SIMT (Single Instruction, Multiple Threads) parallelization on GPUs. Data-parallelism is commonly used to parallelize algorithms that can be decomposed into independent operations via uniform data partitioning.

Task parallelism, on the other hand, is typically applied to algorithms that exhibit more complex, irregular parallelism that cannot be fully exploited by applying regular data-parallelism. We refer to a *task* as a unit of work [1, p. 96]. Based thereon, task parallelism is the concurrent execution of tasks considering the dependencies between these tasks.

3 UNIFORM ARCHITECTURE MODEL

To compare architectural properties of CPUs and GPUs, a uniform view on both processor types is required. For this purpose we follow the platform model of OpenCL since it allows to describe an abstract architectural model that covers CPU-cores and SIMD-lanes as well as streaming multiprocessors and streaming processor cores.

In terms of the OpenCL platform model [2, p. 18] a heterogeneous compute node consists of a *host* processor that is connected to one or several compute *devices*. Each device is subdivided into *compute units* (CUs), which are further subdivided into *processing elements* (PEs). The mapping of CUs and PEs to actual hardware components is not determined by the OpenCL standard but specified by the software developer dependent on the parallelization scheme. For this work, we define a mapping that does not only cover the considered hardware properties of CPUs and GPUs but also supports considerations for data- and task-parallelism.

Table 1: Considered GPU and CPU Chips

ID	Year	Nvidia GPU Chips	AMD GPU Chips	Intel CPU Chips
1	2006	G80 (Tesla)		
2	2010	GF100 (Fermi)		X7560 (Nehalem)
3	2011			E7-8870 (Westmere)
4	2012		Tahiti (GCN 1)	E5-2687W (Sandy Bridge)
5	2013	GK180 (Kepler)	Hawaii (GCN 2)	
6	2014			E7-8890 v2 (Ivy Bridge)
7	2015	GM200 (Maxwell)	Fiji (GCN 3)	E7-8890 v3 (Haswell)
8	2016	GP100 (Pascal)	Ellesmere (GCN 4)	E5-2699A v4 (Broadwell)
9	2017	GV100 (Volta)		8180M (Skylake)
10	2018		Vega 20 (GCN 5.1)	
11	2020	GA100 (Ampere)	Arcturus (CDNA 1.0)	8380HL (Cooperlake)

Applying the terminology of the OpenCL platform model, we refer to CPU-cores as CUs. Further, we consider Nvidia's streaming multiprocessors or AMD's compute units as the GPU-equivalent of CPU-cores. Hence, we refer to those equally as CUs.

A CPU core exhibits SIMD units that execute an operation on multiple data elements simultaneously. For comparability of CPU and GPU architectures, we consider only single-precision floating point (FP32) SIMD operations in terms of MUL-, ADD- and FMA-units. Based thereon, each SIMD-lane is referred to as a single PE. Regarding GPUs, we refer to each of Nvidia's CUDA cores or AMD's stream processors as a single PE. These PEs are operated by multiple warp-lanes (Nvidia) or wavefront-lanes (AMD) in lockstep[1]. This is similar to the behaviour of CPU SIMD units. Hence, mapping processing elements to SIMD-lanes on CPUs and to CUDA cores/stream processors on GPUs allows for a reasonable comparison of both architectures.

4 METHODS

In order to quantify the development of hardware properties over the years, we considered Intel's high-end CPUs as well as Nvidia's and AMD's HPC GPUs since the advent of the first GPGPUs (General Purpose GPUs) in 2006 [4]. Table 1 provides an overview of the processors and microarchitectures that were used to retrieve the data for the charts in Figures 1, 2 and 3.

4.1 Selection of CPU Metrics

CPU data points cover only Intel server CPUs that are typically used in HPC-systems. Accordingly, hardware properties such as clock frequency and core count are retrieved from Intel's product specifications of the Intel Xeon Processors and Intel Xeon Scalable Processors [3].

[1]Except for Nvidia GPUs starting with Volta (compute capability ≥ 7.0).

The number of compute units *#CUs* corresponds to the core count. We do not consider Simultaneous Multi-Threading (SMT) since two SMT-threads share a SIMD unit and do not operate in parallel, but concurrently.

Given clock frequencies are base frequencies instead of max turbo frequencies. This allows for a reasonable comparison of past and present hardware and eliminates the influence of core utilization and SIMD usage on the clock frequency.

The number of processing elements *#PEs* corresponds to the number of SIMD-lanes, which is computed based on the corresponding FP32 SIMD width and the number of FP SIMD units.

4.2 Selection of GPU Metrics

A GPU chip is only included in the data set if it was used in dedicated HPC GPUs, i. e. Nvidia Data Center (former Tesla) series or AMD Radeon Instinct (former FirePro S) series. For each architecture, the chip exhibiting the highest CU count is considered. Data points refer to GPU chips, instead of specific GPU models, to exclude dual-GPU designs.

The number of compute units *#CUs* for all GPU chips is retrieved from the GPU Specs Database [6] and corresponds to the parameter *SM Count*.

The number of processing elements *#PEs* is determined from the ratio of *Shading Units* and *SM Count* as provided by the GPU Specs Database [6].

Similar to the corresponding CPU metric, given GPU clock frequencies are base frequencies instead of boost frequencies. The clock frequency refers always to the highest base clock frequency the GPU chip was operated at.

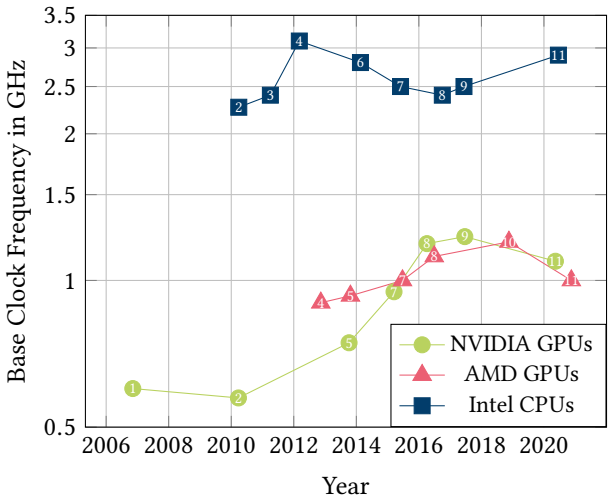

Figure 1: Development of CPU and GPU base clock frequencies over time

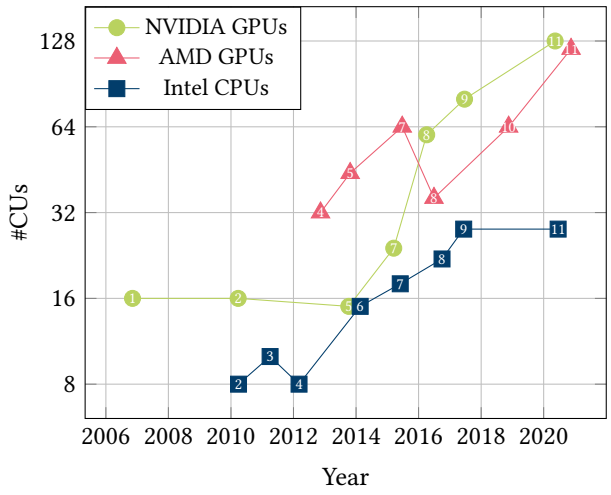

Figure 2: Development of CU count on CPUs and GPUs over time

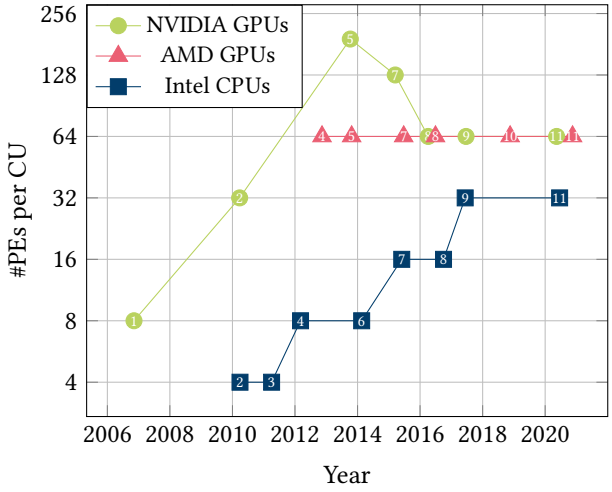

Figure 3: Development of CU size on CPUs and GPUs over time

5 CPU & GPU ARCHITECTURE TRENDS

5.1 Stagnating Clock Frequencies

The chart in Figure 1 depicts the development of base clock frequencies on CPUs and GPUs over time. The base clock frequencies of multicore CPUs stabilize at 2.3 to 3.1 GHz due to their thermal design power. Since 2016 a qualitatively similar stabilization at 1.0 to 1.2 GHz can be observed for GPUs.

5.2 More Compute Units

The chart in Figure 2 depicts the development of the number of compute units *#CUs* per processor since 2006. The chart shows that the amount of CUs increases for CPUs and GPUs regardless of vendors. To still increase hardware performance, the trend of stagnating clock frequencies on CPUs and GPUs is compensated by this effect of increasing CU counts.

5.3 Stagnating Compute Unit Size

The chart in Figure 3 shows the ratio of the number of processing elements per compute unit *#PEs/CU* on CPUs and GPUs over time. This ratio can be considered as the size of a CU, i. e. the amount of hardware parallelism that a CU provides. Over the years, the CU size is constant for AMD GPUs, while it increases for Nvidia GPUs till 2013. However, the Maxwell and Pascal architectures introduce a trend reversal that leads to a decrease in CU size and a levelling at 64 *#PEs/CU*; remarkably, at the same size as AMD CUs.

As can be seen from the chart, the size of CUs on CPUs is progressively increasing. Hence, it is gradually approaching the size of GPU CUs.

6 IMPLICATIONS FOR HPC SOFTWARE

For GPU programming, the increase in the CU count leads to the possibility to execute more independent compute kernels in parallel. This does not only support classical compute kernel overlapping, but also benefits task-parallel programming on GPUs. Accordingly, this effect leads to the extension of originally data-parallel programming models by task-parallel functionalities, e. g., in form of CUDA asynchronous task

graphs, dynamic parallelism or the execution of OpenMP tasks on GPUs.

The trend of stagnating CU sizes reflects the limited amount of inherent data-parallelism in most HPC algorithms. Hence, increasing the size of CUs further would not be effective to achieve reasonable scaling. Considering the origin of GPGPU programming in graphics processing, which is the paragon of data-parallelism, GPGPUs were designed for heavily data-parallel applications. Multicore CPUs, on the other hand, do not only provide data-parallelism through SIMD vectorization but also support the more flexible concept of multithreading, which is the basis for task-parallelism. The observed trends, however, indicate a gradual convergence of both architectures and accordingly lead to a more uniform description of parallelism on both architectures.

7 CONCLUSION

As shown in Section 5, CPU and GPU architectures are subject to similar hardware trends. These hardware trends reach from stagnating CU sizes, stagnating clock frequencies to increasing CU counts.

From the software perspective, the resulting challenge is to use this increasing amount of hardware parallelism through parallel programming technologies that allow for the further parallelization of algorithms beyond data-parallelism. This requires in particular the application of more complex parallelization schemes such as task-parallelism.

To support this endeavour, several programming technologies that aim to simplify parallel programming are under development. This includes in particular programming technologies for applications that are not purely data-parallel and accordingly rely on task-parallelism; this includes CUDA asynchronous task graphs as well as OpenMP tasks. Additionally, parallel programming models are extended to support portability between CPUs and GPUs; this includes frameworks such as OpenCL and OpenMP as well as libraries such as Nvidia's thrust and libcu++.

REFERENCES

[1] David Culler, Jaswinder Pal Singh, and Anoop Gupta. 1998. *Parallel Computer Architecture: A Hardware/Software Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[2] Khronos® OpenCL Working Group. December 18, 2020. *The OpenCL™ Specification.* https://www.khronos.org/registry/OpenCL/specs/3.0-unified/pdf/OpenCL_API.pdf

[3] Intel®. Accessed January 28, 2021. *Product Specifications: Intel® Xeon® Processors.* https://ark.intel.com/content/www/us/en/ark.html#@PanelLabel595

[4] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. 2008. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro* 28, 2 (2008), 39 – 55. http://doi.org/10.1109/MM.2008.31

[5] Herb Sutter. 2005. The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software. *Dr. Dobb's journal* 30, 3 (2005), 202–210.

[6] TechPowerUp. Accessed November 28, 2020. *GPU Specs Database.* https://www.techpowerup.com/gpu-specs/