General-Purpose Graphics Processing Units in Service-Oriented Architectures

María del Carmen Calatrava Moreno
E-Commerce Group
Vienna University of Technology
Vienna, Austria
Email: mc.calatrava.moreno@ec.tuwien.ac.at

Thomas Auzinger
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna, Austria
Email: thomas.auzinger@cg.tuwien.ac.at

***Abstract*—Over the last decades, graphics processing units have developed from special-purpose graphics accelerators to general-purpose massively parallel co-processors. In recent years they gained increased traction in high performance computing, as they provide superior computational performance in terms of runtime and energy consumption for a wide range of problems. In this survey, we review their employment in distributed computing for a broad range of application scenarios. Common characteristics and a classification of the most relevant use cases are described. Furthermore, we discuss possible future developments of the use of general purpose graphics processing units in the area of service-oriented architecture. The aim of this work is to inspire future research in this field and to give guidelines on when and how to incorporate this new hardware technology.**

I. Introduction

As a model for enterprise IT architecture, service-oriented architectures (SOAs) stayed relevant by adapting to several technological advances over the last years. Currently, Web 2.0 and cloud computing are the dominating trends that impact its future development [1]. This poses new parallelization challenges on unprecedented scales. An ever growing amount of users has to be served by increasingly parallel hardware architectures that have to cope with huge amounts of data [2].

On the hardware level, the performance growth of single-core microprocessors has become limited by both semiconductor scaling limits and the difficulty of increasing instruction-level parallelism even further. Thus, microprocessor manufacturers turn to multicore chip architectures to provide continued performance increases, even though software has to be explicitly designed to exploit these parallel computing capabilities.

Graphics processing units (GPUs) nicely fit this trend of increased parallelism, as they provide hundreds of processing elements together with high memory bandwidth. In the last two decades, they developed from single-purpose accelerators of 3D computer graphics to allow general-purpose computing on this hardware, which is generally referred to as GPGPU [3], [4]. For a wide range of application scenarios, they provide superior performance in terms of computational efficiency, runtime and power consumption (see Figure 1) and gain increasing traction in diverse computation scenarios such as high performance computing or mobile device architectures. The back-end empowerment provided by GPGPU is the driving force behind its application to service-oriented architectures. Optimized for throughput-computing of large data- or task parallel input, GPUs are well suited to accelerate business

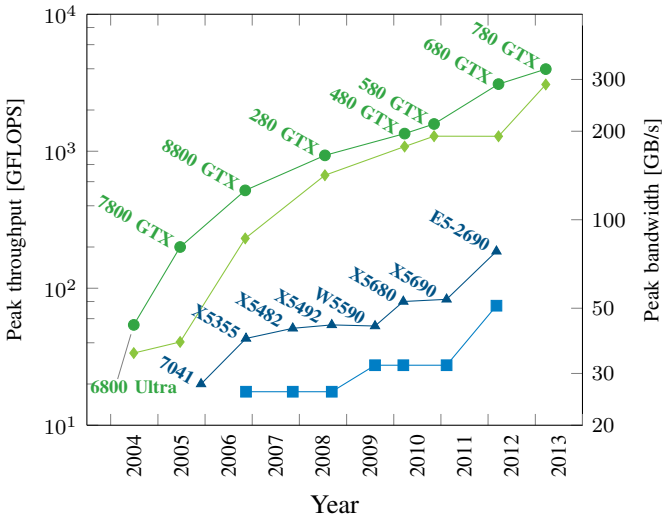

Fig. 1. **Hardware performance growth.** The rapid increase of computational performance of GPUs (—●—) has created a large gap in comparison to CPUs (—▲—). The theoretical peak throughput in terms of single precision floating point computations per second (FLOPS) is given according to the logarithmic scale on the left. In this figure, we provide the performance values for NVIDIA graphics cards and Intel processors of the last decade. Looking at the latest hardware generation, the gap stands at a factor of ~20 as a 780 GTX GPU provides nearly 4TFLOPS peak compute performance. A similar picture arises from the memory bandwidth values of the same GPU (—◆—) and CPU (—■—) models. The theoretical peak memory bandwidth of either global device memory or system memory is given by the logarithmic scale on the right in terms of GB per second. Both performance indicators show the excellent suitability of graphics hardware for data intensive parallel computations that become increasingly important in a wide range of business applications.

informatics applications such as batch processing or data mining, among many others.

In this survey we give an overview of this new exciting field by relating current research and use cases of GPGPU to SOA and by outlining future trends. Our aim is not only to give a comprehensive introduction to GPUs but to stimulate further research on the influences and possible applications of massively parallel hardware in SOA.

The paper is structured as follows: As we do not assume prior knowledge about GPUs, we give an introduction to both their hardware and software aspects in the forthcoming Section II. Related work is presented in Section III. After

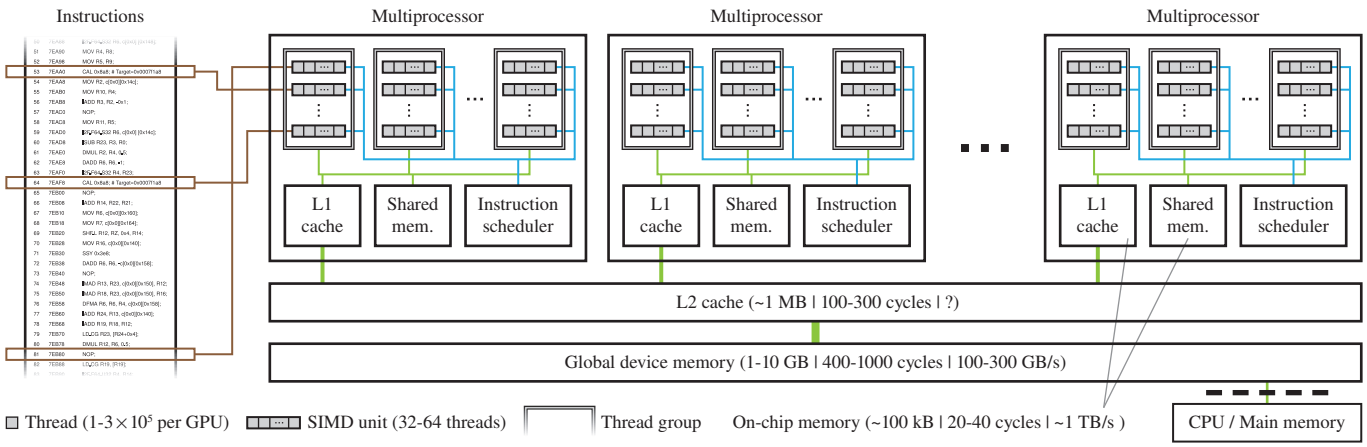

Fig. 2. **Hardware architecture of a GPU** (cf. Section II-B). The program instructions (left) are executed in parallel by the *SIMD units* of the device. The threads are furthermore assembled into programmer-specified *thread groups* which are assigned part of the fast on-chip memory of each multiprocessor. This allows efficient local data sharing between the SIMD units. Each multiprocessor provides an L1 cache and interfaces with the global device memory via an L2 cache. The specifications of each memory type for current GPU models are given as (size | latency | bandwidth). Note that this figure provides a high-level overview and does not cover all details of the hardware architecture.

summarizing current research and use cases in Section IV, Section V outlines future trends of GPUs in conjunction with SOA. We give a set of guidelines on GPU utilization and the conclusion in Section VI.

II. General-Purpose Graphics Processing Units

A. Overview

One of the characteristic hardware requirements for computer assisted image generation is the ability to quickly execute numerical operations on a large amount of independent picture elements (pixels). Already in the 1980s the first graphics coprocessors entered the market and quickly became commodity hardware that can be found in nearly all personal computers. In the 1990s the increasing popularity of real-time 3D graphics created a market for dedicated hardware accelerators and saw the founding of todays major graphics card manufacturers, NVIDIA and ATI Technologies (later acquired by Advanced Micro Devices). Starting as fixed-function devices, 3D graphics cards were soon augmented with programmable stages and in 1999 the term graphics processing unit (GPU) was popularized for the first graphics cards that supported all stages of the 3D graphics pipeline, such as polygon transformations, rasterization and surface shading. Graphics APIs, such as OpenGL and DirectX, appeared during the same time frame and provided the necessary tools to develop software for these new hardware capabilities.

Due to the high memory bandwidth to the internal image-array buffers, and due to a massively parallel hardware architecture, these devices were orders of magnitude faster for graphics operations than conventional CPUs. The trend to more flexible hardware continued and culminated in 2006 with the advent of fully programmable GPUs. While the first general-purpose computations on GPUs were mapped to the specific design of the graphics pipeline [5], [6], [7], this new capability brought the convenience of software development close to the levels found with CPUs.

Nowadays, two languages are used for the vast majority of GPGPU programming: the Open Computing Language (OpenCL) [8], [9] and the Compute Unified Device Architecture (CUDA) [10], [11]. Both are extensions to the programming language C bundled together with an API to interact with the hardware driver to issue program executions or memory transfers. OpenCL is designed as a cross-platform framework to support *heterogeneous (or hybrid) computing* on a wide range of CPUs and GPUs, and is managed by the Khronos Group consortium. CUDA is a proprietary language of NVIDIA tailored to their product range of GPUs.

Most graphics hardware can be found in one of three forms: Dedicated GPUs (1) with their own global memory typically interface with the CPU and system memory by means of expansions slots such as PCI Express. This type makes up the majority of the high-performance graphics card market on both personal and portable computers. Integrated graphics solutions (2) utilize system memory for their operation and are usually employed for either low-performance graphics or mobile devices. Motivated by the rise of GPGPU, a recent third form (3) has entered the market, which offers the computational power of graphics hardware with additional features, such as error-correcting memory, but without graphics related functions such as display outputs. This general-purpose GPU variant is commonly used in high performance computing and is the most relevant hardware for most of the applications that we describe in this paper.

B. Hardware Architecture

GPUs can be classified as *Single Program, Multiple Data* (SPMD) architectures according to an extended version of Flynn's taxonomy [12], [13] (see Figure 2). Each processing element (PE) executes the same program but has its own data and position in the instruction stream. Current models have several hundreds PEs that can be active at the same time. In each cycle, several schedulers issue instructions to a small subset of these. This allows a high utilization of the available computation units, since PEs, which execute high-latency memory accesses, can idle until these operations are completed. In the meantime, PEs, which are ready to perform

actual computations, take their place in a process that is commonly called *latency hiding*.

To maximize the number of PEs on a given GPU die area, computations are executed in a *Single Instruction, Multiple Data* (SIMD) fashion. Each SIMD unit consists of a fixed number of threads that have their own data but operate in lockstep. This means that all threads of a SIMD unit execute the same instruction unless a number of threads stays idle. This is the case if threads of the same SIMD unit follow different code paths, which could lead to significant performance penalties. However, the programmer does not need to be aware of this SIMD architecture to write a correct program. All threads of the GPU can be seen as independent entities that execute in non-specified order but can be orchestrated with synchronization operations.

A further architectural highlight of GPUs is their high memory bandwidth to global device memory. Being an order of magnitude larger as CPU memory bandwidth, it underlines the throughput-orientated design of graphics hardware. To further accelerate memory access operations, all processors of a GPU interface with the global memory via a common L2 cache and individual on-chip L1 caches. Furthermore, the threads (resp. SIMD units) of a device are divided into equally sized groups that reside on a single processor and share part of on-chip memory for fast data exchange and storage. In contrast to the large CPU caches, their GPU equivalents are not designed for temporal data reuse but mainly for an efficient distribution of the accessed cache lines to the threads of a given SIMD unit.

C. Software Architecture

Both current GPGPU programming frameworks – CUDA and OpenCL – are extensions to the programming language C and contain a compiler framework as well as APIs to interact with the graphics hardware. In comparison to traditional CPU programming, the hardware parallelism is directly exposed by additional keywords which provide access to thread and thread group indices. This allows the programmer to write a scalar program, called *kernel*, that is executed by the device threads in parallel. Both memory transfer to and from the GPU as well as kernel launches are generally issued by the CPU; very recent developments also enable this functionality for kernels. So far, no general translation method between CPU and GPU code exists and efficient programs have to be written specifically for the GPGPU architecture.

D. Comparison with CPUs

In this section we outline the main differences between CPUs and GPUs and their consequences. CPUs are designed to provide a fast response time to a single task and employ complex techniques such as branch prediction, out-of-order execution or super-scalar computation. Coupled with large on-die caches, these leave only space for a low amount of actual processing cores. GPUs trade single-thread performance for increased parallel processing using a hardware architecture as described in Section II-B.

Thus, a computational problem has to exhibit a high degree of data- or task parallelism to efficiently run on GPUs. Such applications, also referred to as *throughput computing*, are becoming increasingly relevant as many application domains are confronted with ever-growing amounts of data. A large body of scientific literature as been generated on the comparison of performance data from both CPUs and GPUs. Depending on the employed metric, large discrepancies can be observed.

Comparisons done by experts, who can ensure highly optimized code for both platforms, report a 2-10 times performance increase by employing GPUs for fundamental tasks such as dense and sparse linear algebra, fast Fourier transform, back propagation, k-means, etc. [14], [15], [16]. Programmers of real world applications report much larger performance increases by switching to GPUs [17], [18]. This can be explained by a comparison with not fully optimized CPU code, as such optimal programs can be considerably harder to obtain on the CPU than on the GPU [19]. For dedicated GPUs, the memory transfer time to and from the main system memory can be substantial [20] but will become less problematic as the last line of graphics hardware allows more autonomous GPU computing by launching kernels through kernel code.

In terms of energy efficiency, the most recent publication, which uses current hardware, reports GPUs as being ∼2-20 times more efficient than CPUs on characteristic workloads [21] or by grouping many small non-optimal workloads [22]. As a next step, current research explores optimal workload distribution on heterogeneous systems, which yields additional power savings in the 10-30% range [23], [24].

Both runtime and energy efficiency data show the superior performance of GPGPU for many data-intensive computing problems. We move away from the actual algorithms and in Section IV we show a wide variety of real-world applications where GPU computing is already used to great effect

III. Related Work

With the growing popularity of GPGPU, many are the scientific papers that explain the general operating mechanisms of graphics hardware for cloud and high performance computing [25], [26]. Others, instead, illustrate their utilization in a very specific context, such as pricing of securities estimation in financial analysis [27]. Such works generally show large performance benefits but tend to have either a very low level of abstraction focused on technical implementation details or are designed for a very specific application. Literature discussing the general application of GPGPUs to services is very scarce. An exception to this is the very recent work of Hu et al. [28], who presented a general scheme of a GPU-assisted cloud system comprised of three layers (cloud, server and GPUs) that maps tasks to hardware components.

With this paper we aim to fill the existing gap in the literature using the broader perspective of SOA. We do not restrict ourself to specific problems but give an overview of the multitude of existing application examples and the promising future employment of graphics hardware in SOAs. By classifying the use cases into the layer of reference architecture, we show which specific advantage of GPUs is most beneficial at each layer.

IV. Use Cases in Distributed Computing

In this section an overview of successful employments of GPUs in different application domains is given. Since a

TABLE I. CLASSIFICATION OF **CURRENT** GPGPU ACCORDING TO A SOA REFERENCE ARCHITECTURE [29].

Layer	Advantages	Techniques	Application Examples
Presentation	Advanced graphics	3D rendering	Local and remote rendering, 3D video
Service	Increased computational performance and/or power efficiency	(partially virtualized) IaaS	Amazon Elastic Compute Cloud, SoftLayer HPC servers
Service components	Efficient hardware utilization	Workload consolidation	Numerous small-scale simulations, file and networking systems, encryption
Operational systems	Increased computational performance and/or power efficiency	GPGPU programming or use of GPGPU libraries	Simulation in sciences and finance, data mining

complete listing of all GPU use cases exceeds the scope of this survey, we distill the most relevant examples into a SOA related classification. The organization is based on the layers of a reference architecture for SOA [29] and given in Table I. Our aim with this classification is to enable an overview of the interrelation of heterogeneous hardware and SOA and to efficiently communicate the potential benefits of using GPGPU in related fields.

We cover example applications for four layers, which are described in the following subsections. As the use of GPGPU pioneered in the field of high performance computing, most current examples can be found for the *operational systems* layers. As the professional use of general-purpose graphics hardware matures, we already see examples in the *service* and *service components layer*, while the *presentation* layer is still mainly influenced by graphics related processing.

A. Operational Systems Layer

The parallel nature and the large scale of many scientific computational problems have lead to an early adoption of graphics hardware. One of the earliest heavily publicized applications was the massively distributed biophysics simulation *Folding@Home* [30]. Processing during the idle time of personal computers of voluntary participants, the whole system reached a sustained performance level of five petaFLOPS already in 2008. The first GPU-enabled client software was released in 2006 and saw a 20-30 fold performance increase compared to the CPU version. Nowadays, GPUs are responsible for nearly 90% of the processed workload of Folding@Home while constituting ∼10% of the processors.

Computational simulation and analysis in the **natural sciences** was always at the forefront of high performance computing. Traditional supercomputers still play a major role, and the newest models, such as Oak Ridge's *Titan* or NCSA's *Blue Waters*, extensively use GPGPU hardware. Additionally, new trends such as grid or cloud computing emerged in the last decade. The *Worldwide LHC Compting Grid* [31], for example, comprises 170 computing facilities to process the approximately 25 PB of measurement data per year that are generated by the LHC particle collider of CERN.

Regarding the actual use cases, academic works and real-world applications on this topic exist for all major scientific fields. In biology, genomics research generates huge amounts of microarray data and SOA has been used to facilitate data access [32], to provide analysis platforms [33], and to automate analysis method choice [34]. Many core subtasks achieved major performance gains when being ported to the GPU, such as protein database search [35] or short read alignment [36]. In the vast field of physics, particle physics, for example, employs GPUs for supercomputing simulations [37] or real-time filtering of experimental data [38]. Further use cases can be found in astronomy [39], computational chemistry [40] or a SOA-based tsunami mechanics tool [41] in geophysics. GPU utilization in engineering is also widespread, e.g., for fluid [42] or electromagnetical and thermal simulation [43].

Capital markets are facing increasingly complex and global structured products. They require, among others, greater financial regulation as well as reliable internal risk management. The former empowers organizations to monitor activities and enforce actions (e.g., supervision of stock exchanges, listed companies, investment management, etc.). While the later involves operational and system contingency planning to respond to and mitigate damaging events. The abundance of market-data messages requires high performance **financial computing** to deliver real-time results, which is a major concern of the industry. Some of the most important computational tasks (i) are hedging strategies, which require high-frequency algorithmic trading to make gain opportunities; (ii) value at risk calculation, which copes with a large number of sophisticated assets to measure the risk of severe losses due to market events; (iii) and high dimensional option pricing.

To meet the demand of high volume data loads and/or real-time performance of the associated systems, GPGPU is employed. Example applications are option pricing [44], or value-at-risk estimations [45]. Industrial usage of GPUs can be found in software for trade management by *Murex* (www.murex.com) or for catastrophic risk evaluation by *RMS* (www.rms.com).

Typical **data mining** applications are confronted with a huge amount of information to process. The use of distributed computing is the dominating approach to enable the handling of the required compute and memory resources. Furthermore, most of the existing distributed data mining projects solve problems by composing a team of distributed specialized systems, each associated with a specific task (e.g., data cleansing, data pre-processing, data mining, etc.). SOA is therefore a logical fit for the overall system architecture [46].

Many of the subsystems exhibit a high degree of task- or data parallelism and the efficient mapping to parallel hardware architectures is an active field of research [47], [48]. Naturally, GPUs are well suited for these tasks and

their usefulness for clustering [49], graph traversals [50] and database query processing [51] is already established. Typical task sizes are usually orders of magnitude larger than the global memory of common graphics hardware, and an efficient orchestration of memory transfers between the different systems components is necessary for satisfactory overall performance. GPUMiner [52], for example, provides this functionality as I/O between storage, CPU and GPU is managed, as is CPU-GPU co-processing.

B. Service Components Layer

In the previous section, we provided examples for the use of GPGPU for a wide range of applications. Their realization was achieved by replacing a CPU-based program with either GPU or combined CPU-GPU computations. The achievable speed-up due to increased computational performance or memory bandwidth of the graphics hardware, depends strongly on the application at hand. In this section, we present examples of GPU utilization that achieve performance gains by combining a large amount of originally unsuited tasks. Small or sequential workloads do not fit the computation model of GPUs and cannot be accelerated by a change of the underlying hardware. Thus, a change in the operational systems layer will not provide speedups. In high-throughput computation, however, it is possible that many of these tasks can be combined into larger workloads and efficiently processed on the GPU. This approach is commonly called *workload consolidation* and constitutes the main topic of this section.

In different simulation scenarios, not one huge problem has to be solved but a multitude of small tasks. Examples of efficient parallel computations of such tasks on GPUs were given for quantum molecular dynamics simulations [53], where many small FFTs have to be solved, or for cellular automata [54], where many small graph problems arise.

The main body of this recent research field, can be found in the domain of data and network transfer. The parallel execution of a large number of parity checks in software RAIDs [55] is one example. The acceleration of encryption schemes such as SSL [56], [57] or AES [58] are results of recent efforts which triggered investigations into the security of GPUs. First vulnaribilites were already identified [59] and will influence future hardware designs.

C. Service Layer

The growing popularity of GPGPU made it commercially viable to provide on-demand access to graphics hardware in a cloud computing context (e.g., Amazon Elastic Compute Cloud, SoftLayer HPC servers). Following the *infrastructure as a service model*, these first generation services offer direct access to the hardware. Currently, no solution offers effective virtualization of GPU devices as they are transparently accessible per node.

D. Presentation Layer

In this layer, GPUs are mainly used for image generation and one exemplary architecture is remote rendering. The use of graphics hardware on the server side enables it to efficiently perform computations that are too demanding for the client. The limits on power consumption of mobile devices and their generally low computational performance make such an approach especially attractive for mobile 3D graphics [60], [61]. The output of server-side rendering is encoded and transmitted to the mobile device in the form of a video stream. However, constrained bandwidth and increased latency are a problem for interactive application that use this technology.

The increased computational performance of today's mobile devices, however, allows local small-scale 3D rendering which is the largest application of computer graphics when ranked by user count. Current research tries to extend these capabilities by performing energy-aware runtime decisions [62] or remote assistance by the architecture mentioned above [63].

V. Future Developments

After providing an overview of the current state-of-the-art in GPGPU in conjunction with SOA, we outline our predictions for future developments in this field. In this section, we again use the layering of a SOA reference architecture in Table II to provide a classification of the prospective influences of graphics hardware in the next years.

A. Operational Layer

The trend in parallel architectures to move away from symmetric multiprocessors and to employ heterogeneous system with different processors and memories will continue in the next years. This is a direct consequence of the lasting efforts to increase both computational power and energy efficiency while reaching the semiconductor limits of single thread performance. The efficient orchestration of diverse resources will be subject to intensive research in the coming years. For heterogeneous architectures, this is still mainly a manual task done by software engineers and usually tailored to a narrow range of hardware configurations. An automation of this process is highly desired and the first research efforts have already materialized. Learning the performance of certain tasks on different processors and memories enables the most efficient mapping of programs to free system resources at runtime [64]. Automated management of CPU-GPU data transfers for regular data layouts [65] and recursive pointer-based data structures [66] minimize waiting time for memory transactions. To avoid underutilization of hardware due to sub-optimal scheduling, automated workload balancing methods allow efficient job distribution on heterogeneous systems [67]. In the end, this will enable convenient application development by hiding the underlying complexities of the hardware while still guaranteeing adequate resource utilization. Note that the same holds for all future massively parallel devices, such as the Intel Many Integrated Core Architecture (MIC), which combines several dozens of CPUs into one multiprocessor.

B. Service Components Layer

We also expect an increased effort to enable workload consolidation on parallel hardware in the coming years. Many problems of throughput computing exhibit a large amount of small tasks that are processed similarly. While many areas of application can be identified (e.g., file systems, network transfers, database designs) and researched, an advanced topic is the optimal linkage of different components. Depending on the input data volume and the available system resources,

TABLE II. CLASSIFICATION OF **FUTURE** GPGPU ACCORDING TO A SOA REFERENCE ARCHITECTURE [29].

Layer	Advantages	Techniques	Application Examples
Presentation	Advanced graphics	3D rendering, computer vision	PCaaS, cloud gaming, augmented reality
Service	Increased computational performance and/or power efficiency	(fully virtualized) IaaS	NVIDIA GRID
Service components	Efficient hardware utilization	Workload consolidation, kernel linking	File systems, network transfers, databases, fully homomorphic encryption
Operational systems	Increased computational performance and/or power efficiency	Automated code generation on heterogeneous hardware	General-purpose throughput computing

the optimal component configuration might be different. First research efforts investigated the efficiencies of different GPU kernel fusions in respect to runtime performance [68] and energy efficiency [69], as well as the fair sharing of GPU resources among multiple tenant applications on the same system [70].

A further highlight will be the adoption of *Fully Homomorphic Encryption* (FHE) to remove one of the largest drawbacks of cloud computing. Even when using encrypted data and obfuscated programs for remote processing in the cloud, the data has to be encrypted at one point, which raises significant trust issues. FHE solves this issue by enabling encrypted programs to operate on encrypted data without the requirement of intermediate decryption [71]. Thus, even security sensitive computations can be moved to the cloud. A drawback of this method is its significant computational overhead due to the large amount of big integer multiplications. First experiments with GPU-aided acceleration methods [72] show promising results and we expect this research path to be continued in the next years.

C. Service Layer

We anticipate that the field of heterogeneous infrastructure as a service will mature in the near future by overcoming one of its main limitations [73]. Efficient sharing of underlying graphics hardware is an essential element of this task and will be enabled through full GPU virtualization. There are a multitude of approaches to achieve this, such as rCUDA [74], vCUDA [75] or gVirtus [76], among others. They all enable full access of the GPU from applications executed on top of the virtual machine but only gVirtus has so far succeeded in enabling efficient GPU sharing [77]. The first fully virtualized commercial product, the NVIDIA GRID [78], was recently presented but is restricted to the graphics capabilities of their GPUs. We present more information on it in the following section.

D. Presentation Layer

An example of new advances for this layer is the NVIDIA GRID [78] technology which is designed to deliver graphics-intensive processing to concurrent users for virtual desktops, visual computing and cloud gaming. The first use case enables a conceptual return to mainframe computing by removing the graphics bottleneck and constitutes a major step towards a *personal computer as a service*. Cloud gaming claims to revolutionize the multi-billion computer game industry by providing remote gaming to arbitrary devices [79]. However, it is heavily dependent on network performance and its actual potential is not fully explored yet.

Another major trend in this area is *augmented reality* where the a real-world camera stream is enhanced with virtual computer graphics. GPGPU is highly relevant in this area, as the processing not only includes graphics generation but additional massively parallel tasks such as geo-locating, multimedia retrieval and alignment with the camera stream has to be performed.

VI. CONCLUSION

In this survey, we presented a comprehensive introduction to the use of general-purpose graphics hardware in the field of SOA. After an description of the hardware and software architecture, we gave an outline of current use cases where the parallel processing capabilities of this new hardware architecture brought significant performance increase in terms of runtime and/or energy efficiency. We employed a classification along the layers of a SOA reference architecture to clearly present the benefits of GPUs at different system levels. The same was done for a selection of future developments which we expect to have major impact on the use of SOA.

As GPGPU is becoming a major technology, development for this hardware platform becomes increasingly convenient. However, we advise the use of GPU-assisted libraries for research in the near future or the cooperation with domain experts. Especially for improvements in the field of workload consolidation, a good knowledge of the underlying hardware is required to design well-performing algorithms.

With the future proliferation of massively parallel hardware we also expect a further increase of the relevance of SOA. As low-level software design, development and optimization have to respect vastly different processor architectures, an efficient decoupling is necessary to hide the associated complexities from high-level application developers. GPGPU can be seen as the forerunner of service-oriented general-purpose heterogeneous computing.

ACKNOWLEDGMENT

The authors would like to thank Christian Huemer and Amin Anjomshoaa for inspiring discussions. Thomas Auzinger is supported by the Austrian Science Fund (FWF) grant no. P23700-N23 (Modern Functional Analysis in Computer Graphics – MOFA).

REFERENCES

[1] G. Feuerlicht and S. Govardhan, "SOA: Trends and directions," *Systems Integration*, vol. 149, 2009.

[2] T. Hey, S. Tansley, and K. M. Tolle, Eds., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[3] D. Luebke, M. Harris, J. Krüger, T. Purcell, N. Govindaraju, I. Buck, C. Woolley, and A. Lefohn, "GPGPU: general purpose computation on graphics hardware," in *SIGGRAPH 2004 Course Notes*. ACM, 2004.

[4] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[5] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan, "Brook for GPUs: stream computing on graphics hardware," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 777–786, Aug. 2004.

[6] M. McCool and S. D. Toit, *Metaprogramming GPUs with Sh*. AK Peters Ltd, 2004.

[7] D. Tarditi, S. Puri, and J. Oglesby, "Accelerator: using data parallelism to program GPUs for general-purpose uses," in *Proc. of the 12th int. conf. on Architectural support for programming languages and operating systems*, ser. ASPLOS XII. ACM, 2006, pp. 325–335.

[8] Khronos Group. (2013, Aug.) OpenCL Standard. [Online]. Available: http://www.khronos.org/opencl/

[9] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science & Engineering*, vol. 12, no. 3, p. 66, 2010.

[10] NVIDIA. (2013, Aug.) CUDA. [Online]. Available: http://www.nvidia.com/cuda/

[11] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.

[12] M. Flynn, "Some computer organizations and their effectiveness," *IEEE Trans. Comput.*, vol. C-21, no. 9, pp. 948–960, 1972.

[13] F. Darema, D. George, V. Norton, and G. Pfister, "A single-program-multiple-data computational model for EPEX/FORTRAN," *Parallel Computing*, vol. 7, no. 1, pp. 11–24, 1988.

[14] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100x GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU," in *Proc. of the 37th annu. int. symp. on Computer architecture (ISCA)*. ACM, 2010, pp. 451–460.

[15] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli, "High performance discrete fourier transforms on graphics processors," in *Proc. of the 2008 ACM/IEEE conf. on Supercomputing*, ser. SC '08. IEEE Press, 2008, pp. 2:1–2:12.

[16] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *J. Parallel Distrib. Comput.*, vol. 68, no. 10, pp. 1370–1380, 2008.

[17] Q. Fang and D. A. Boas, "Monte carlo simulation of photon migration in 3d turbid media accelerated by graphics processing units," *Opt. Express*, vol. 17, no. 22, pp. 20 178–20 190, Oct 2009.

[18] P. Lu, H. Oki, C. Frey, G. Chamitoff, L. Chiao, E. Fincke, C. Foale, S. Magnus, J. McArthur, WilliamS., D. Tani, P. Whitson, J. Williams, W. Meyer, R. Sicker, B. Au, M. Christiansen, A. Schofield, and D. Weitz, "Orders-of-magnitude performance increases in GPU-accelerated correlation of images from the international space station," *J. of Real-Time Image Processing*, vol. 5, no. 3, pp. 179–193, 2010.

[19] N. Satish, C. Kim, J. Chhugani, H. Saito, R. Krishnaiyer, M. Smelyanskiy, M. Girkar, and P. Dubey, "Can traditional programming bridge the ninja performance gap for parallel computing applications?" in *Proc. of the 39th annu. Int. Symp. on Computer Architecture*, ser. ISCA '12. IEEE Computer Society, 2012, pp. 440–451.

[20] C. Gregg and K. Hazelwood, "Where is the data? why you cannot debate CPU vs. GPU performance without the answer," in *Performance Analysis of Systems and Software, 2011 IEEE Int. Symp. on*, ser. ISPASS '11, 2011, pp. 134–144.

[21] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a K20 GPU," in *Proc. of the 2nd Int. Workshop on Power-aware Algorithms, Systems, and Architectures*, ser. PASA '13, Oct. 2013.

[22] D. Li, S. Byna, and S. Chakradhar, "Energy-aware workload consolidation on GPU," in *Parallel Processing Workshops, 2011 40th Int. Conf. on*, ser. ICPPW '11, 2011, pp. 389–398.

[23] S. Hong and H. Kim, "An integrated GPU power and performance model," in *Proc. of the 37th annu. int. symp. on Computer architecture*, ser. ISCA '10. ACM, 2010, pp. 280–289.

[24] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "GreenGPU: A holistic approach to energy efficiency in GPU-CPU heterogeneous architectures," in *Parallel Processing, 2012 41st Int. Conf. on*, ser. ICPP '12, 2012, pp. 48–57.

[25] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. P. Walters, "Heterogeneous cloud computing," in *Cluster Computing, 2011 IEEE Int. Conf. on*, ser. CLUSTER '11. IEEE, 2011, pp. 378–385.

[26] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W.-m. Hwu, "GPU clusters for high-performance computing," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE Int. Conf. on*. IEEE, 2009, pp. 1–8.

[27] G.-H. King, Z.-Y. Cai, Y.-Y. Lu, J.-J. Wu, H.-P. Shih, and C.-R. Chang, "A high-performance multi-user service system for financial analytics based on web service and GPU computation," in *Parallel and Distributed Processing with Applications, 2010 Int. Symp. on*, ser. ISPA '10. IEEE, 2010, pp. 327–333.

[28] L. Hu, X. Che, and Z. Xie, "GPGPU cloud: A paradigm for general purpose computing," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 22–23, 2013.

[29] The Open Group, "SOA reference architecture," Dec. 2011.

[30] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, "Folding@home: Lessons from eight years of volunteer distributed computing," in *Proc. of the 2009 IEEE Int. Symp. on Parallel&Distributed Processing*, ser. IPDPS '09. IEEE Comput. Soc., 2009, pp. 1–8.

[31] I. Bird, "Computing for the large hadron collider," *annu. Review of Nuclear and Particle Science*, vol. 61, no. 1, pp. 99–118, 2011.

[32] D. Habich, S. Richly, W. Lehner, U. Assmann, M. Grasselt, A. Maier, and C. Pilarsky, "Data-aware soa for gene expression analysis processes," in *Proc. of the IEEE Congress on Services*, 2007.

[33] S. Richly, D. Habich, M. Thiele, S. Götz, and S. Hartung, "Supporting gene expression analysis processes by a service-oriented platform," in *Proc. of the IEEE Int. Conf. on Services Computing*, 2007.

[34] P. H.-M. Chang, V.-W. Soo, T.-Y. Chen, W.-S. Lai, S.-C. Su, and Y.-L. Huang, "Automating the determination of open reading frames in genomic sequences using the web service techniques - a case study using sars coronavirus," in *Proc. of the 4th IEEE Symp. on Bioinformatics and Bioengineering*, 2004.

[35] Y. Liu, B. Schmidt, and D. L. Maskell, "CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions," *BMC Res Notes*, vol. 3, p. 93, 2010.

[36] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, and T.-W. Lam, "SOAP3: ultra-fast GPU-based parallel alignment tool for short reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.

[37] M. Clark, R. Babich, K. Barros, R. Brower, and C. Rebbi, "Solving lattice QCD systems of equations using mixed precision solvers on GPUs," *Comput. Phys. Commun.*, vol. 181, no. 9, pp. 1517–1528, 2010.

[38] G. Collazuol, G. Lamanna, J. Pinzino, and M. Sozzi, "Fast online triggering in high-energy physics experiments using GPUs," *Nucl. Instrum. Meth. A*, vol. 662, no. 1, pp. 49–54, 2012.

[39] A. Hassan, C. J. Fluke, and D. G. Barnes, "Unleashing the power of distributed CPU/GPU architectures: Massive astronomical data analysis and visualization case study," *arXiv preprint arXiv:1111.6661*, 2011.

[40] A. Shkurti, M. Orsi, E. Macii, E. Ficarra, and A. Acquaviva, "Acceleration of coarse grain molecular dynamics on GPU architectures," *Journal of Computational Chemistry*, vol. 34, no. 10, pp. 803–818, 2013.

[41] A. Vazhenin, K. Hayashi, and A. Romanenko, "Service-oriented tsunami wave propagation modeling tools," in *Proc. of the 2012 Joint Int. Conf. on Human-Centered Computer Environments*. ACM, 2012, pp. 131–136.

[42] R. Di Lauro, F. Giannone, L. Ambrosio, and R. Montella, "Virtualizing general purpose GPUs for high performance cloud computing: an application to a fluid simulator," in *Parallel and Distributed Processing with Applications, 2012 IEEE 10th Int. Symp. on*, ser. ISPA '12. IEEE, 2012, pp. 863–864.

[43] C. Richter, S. Schops, and M. Clemens, "GPU acceleration of finite difference schemes used in coupled electromagnetic/thermal field simulations," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1649–1652, 2013.

[44] M. Benguigui and F. Baude, "Towards parallel and distributed computing on GPU for american basket option pricing," in *IEEE 4th Int. Conf. on Cloud Computing Technology and Science*, ser. CloudCom '12. IEEE, 2012, pp. 723–728.

[45] M. Dixon, J. Chong, and K. Keutzer, "Acceleration of market value-at-risk estimation," in *Proc. of the 2nd Workshop on High Performance Computational Finance*. ACM, 2009, p. 5.

[46] W. K. Cheung, X.-F. Zhang, H.-F. Wong, J. Liu, Z.-W. Luo, and F. C. Tong, "Service-oriented distributed data mining," *Internet Computing, IEEE*, vol. 10, no. 4, pp. 44–54, 2006.

[47] D. W. Cheung and Y. Xiao, "Effect of data distribution in parallel mining of associations," *Data Mining and Knowledge Discovery*, vol. 3, no. 3, pp. 291–314, 1999.

[48] E.-H. Han, G. Karypis, and V. Kumar, "Scalable parallel data mining for association rules," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 337–352, 2000.

[49] R. Wu, B. Zhang, and M. Hsu, "Clustering billions of data points using GPUs," in *Proc. of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. ACM, 2009, pp. 1–6.

[50] D. Merrill, M. Garland, and A. Grimshaw, "Scalable GPU graph traversal," in *Proc. of the 17th ACM SIGPLAN symp. on Principles and Practice of Parallel Programming*, ser. PPoPP '12. ACM, 2012, pp. 117–128.

[51] T. Kaldewey, G. Lohman, R. Mueller, and P. Volk, "GPU join processing revisited," in *Proc. of the 8th Int. Workshop on Data Management on New Hardware*, ser. DaMoN '12. ACM, 2012, pp. 55–62.

[52] W. Fang, K. K. Lau, M. Lu, X. Xiao, P. Yang Yang, B. He, Q. Luo, P. V. Sander, and K. Yand, "Parallel data mining on graphics processors," HKUST and Microsoft China, Tech. Rep., 2008. [Online]. Available: http://code.google.com/p/gpuminer/

[53] S. Mitra and A. Srinivasan, "Small discrete fourier transforms on GPUs," in *Proc. of the 2011 11th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, ser. CCGRID '11. IEEE Computer Society, 2011, pp. 33–42.

[54] D. Arendt and Y. Cao, "GPU acceleration of many independent mid-sized simulations on graphs," in *Proc. of the 4th Cellular Automata, Theory and Applications Workshop*, ser. *A-CSC '12, 2012.

[55] A. Khasymski, M. Rafique, A. Butt, S. Vazhkudai, and D. Nikolopoulos, "On the use of GPUs in realizing cost-effective distributed RAID," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2012 IEEE 20th Int. Symp. on*, ser. MASCOTS '12, 2012, pp. 469–478.

[56] K. Jang, S. Han, S. Han, S. Moon, and K. Park, "SSLShader: cheap SSL acceleration with commodity processors," in *Proc. of the 8th USENIX conf. on Networked systems design and implementation*, ser. NSDI'11. USENIX Association, 2011.

[57] J. Gilger, J. Barnickel, and U. Meyer, "GPU-acceleration of block ciphers in the OpenSSL cryptographic library," in *Proc. of the 15th int. conf. on Information Security*, ser. ISC'12. Springer-Verlag, 2012, pp. 338–353.

[58] G. Schönberger and J. Fuß, "GPU-assisted AES encryption using GCM," in *Proc. of the 12th IFIP TC 6/TC 11 int. conf. on Communications and multimedia security*, ser. CMS'11. Springer-Verlag, 2011, pp. 178–185.

[59] R. Di Pietro, F. Lombardi, and A. Villani, "CUDA Leaks: Information Leakage in GPU Architectures," *ArXiv e-prints*, May 2013.

[60] W. Yoo, S. Shi, W. J. Jeon, K. Nahrstedt, and R. H. Campbell, "Real-time parallel remote rendering for mobile devices using graphics processing units," in *Multimedia and Expo, 2010 IEEE Int. Conf. on*, ser. ICME '10. IEEE, 2010, pp. 902–907.

[61] M. E. Fathy, A. S. Hussein, S. H. Hamad, A. H. Abdelaziz, S. H. Abdelaziz, and H. El-Bery, "Parallel remote rendering of large 3D point-based models on mobile clients," in *Computational Intelligence, Modelling and Simulation, 2010 2nd Int. Conf. on*, ser. CIMSiM '10. IEEE, 2010, pp. 419–426.

[62] M. Hosseini, A. Fedorova, J. Peters, and S. Shirmohammadi, "Energy-aware adaptations in mobile 3d graphics," in *Proc. of the 20th ACM int. conf. on Multimedia*, ser. MM '12. ACM, 2012, pp. 1017–1020.

[63] S. Shi, K. Nahrstedt, and R. Campbell, "A real-time remote rendering system for interactive mobile graphics," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 8, no. 3s, pp. 46:1–46:20, Oct. 2012.

[64] P. M. Phothilimthana, J. Ansel, J. Ragan-Kelley, and S. Amarasinghe, "Portable performance on heterogeneous architectures," in *Proc. of the 18th int. conf. on Architectural support for programming languages and operating systems*, ser. ASPLOS '13. ACM, 2013, pp. 431–444.

[65] T. B. Jablin, P. Prabhu, J. A. Jablin, N. P. Johnson, S. R. Beard, and D. I. August, "Automatic CPU-GPU communication management and optimization," *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 142–151, 2011.

[66] T. B. Jablin, J. A. Jablin, P. Prabhu, F. Liu, and D. I. August, "Dynamically managed data for CPU-GPU architectures," in *Proc. of the 10th Int. Symp. on Code Generation and Optimization*. ACM, 2012, pp. 165–174.

[67] M. E. Belviranli, L. N. Bhuyan, and R. Gupta, "A dynamic self-scheduling scheme for heterogeneous multiprocessor architectures," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 4, pp. 57:1–57:20, Jan. 2013.

[68] S. Sarkar, S. Mitra, and A. Srinivasan, "Reuse and refactoring of GPU kernels to design complex applications," in *Proc. of the 2012 IEEE 10th Int. Symp. on Parallel and Distributed Processing with Applications*, ser. ISPA '12. IEEE Computer Society, 2012, pp. 134–141.

[69] G. Wang, Y. Lin, and W. Yi, "Kernel fusion: An effective method for better power efficiency on multithreaded GPU," in *Proc. of the 2010 IEEE/ACM Int. Conf. on Green Computing and Communications & Int. Conf. on Cyber, Physical and Social Computing*, ser. GREENCOM-CPSCOM '10. IEEE Computer Society, 2010, pp. 344–350.

[70] D. Sengupta, R. Belapure, and K. Schwan, "Multi-tenancy on GPGPU-based servers," in *Proc. of the 7th int. workshop on Virtualization technologies in distributed computing*, ser. VTDC '13. ACM, 2013, pp. 3–10.

[71] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of the 41st annu. ACM symp. on Theory of computing*. ACM, 2009, pp. 169–178.

[72] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Accelerating fully homomorphic encryption using GPU," in *High Performance Extreme Computing, 2012 IEEE Conf. on*, ser. HPEC '12. IEEE, pp. 1–5.

[73] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan, "Computational solutions to large-scale data management and analysis," *Nature Reviews Genetics*, vol. 11, no. 9, pp. 647–657, 2010.

[74] J. Duato, A. J. Pena, F. Silla, R. Mayo, and E. S. Quintana-Orti, "rCUDA: Reducing the number of GPU-based accelerators in high performance clusters," in *High Performance Computing and Simulation, 2010 Int. Conf. on*, ser. HPCS '10. IEEE, 2010, pp. 224–231.

[75] L. Shi, H. Chen, J. Sun, and K. Li, "vCUDA: GPU-accelerated high-performance computing in virtual machines," *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 804–816, 2012.

[76] G. Giunta, R. Montella, G. Agrillo, and G. Coviello, "A GPGPU transparent virtualization component for high performance computing clouds," in *Euro-Par 2010-Parallel Processing*. Springer, 2010, pp. 379–391.

[77] V. T. Ravi, M. Becchi, G. Agrawal, and S. Chakradhar, "Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework," in *Proc. of the 20th int. symp. on High performance distributed computing*. ACM, 2011, pp. 217–228.

[78] NVIDIA. (2013, Aug.) GRID. [Online]. Available: http://www.nvidia.com/object/nvidia-grid.html

[79] A. Ojala and P. Tyrvainen, "Developing cloud business models: A case study on cloud gaming," *IEEE Softw.*, vol. 28, no. 4, pp. 42–47, Jul. 2011.