Wolfgang Lehner, Annett Ungethüm, Dirk Habich

Diversity of Processing Units

Erstveröffentlichung in / First published in:

Datenbank-Spektrum. 2018. 18(1), S. 57–62. Springer. ISSN 1610-1995.

DOI: https://doi.org/10.1007/s13222-018-0276-y

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

Diversity of Processing Units

An Attempt to Classify the Plethora of Modern Processing Units

Wolfgang Lehner[1] · **Annett Ungethüm**[1] · **Dirk Habich**[1]

Published online: 2 February 2018

Abstract
Recent hardware developments are providing a plethora of alternatives to well-known general-purpose processing units. This development reaches into all major directions, i.e., into high-speed and low latency communications systems, novel memory components as well as a zoo of different processing units in addition to the traditional CPU-style processors. While all developments have great impact on the design of database systems, we will try—in the context of this *Kurz Erklärt*—to categorize recent advances in the context of processing units and comment on the impact on database systems.

Keywords Database Systems · Modern Hardware · Diversity of Compute Units

1 Introduction

The advances in the context of hardware are numerous and will have a significant impact on performance-critical or energy-efficient system designs. While this holds for all software systems, database systems may have a special status because of the variety of functional and non-functional requirements ranging from scalable and high performant query processing expectations to enterprise-scale qualities like robustness and resilience against all kinds of errors. Moreover, recent hardware advances are touching all aspects of data processing forcing the database community to take special care of opportunities to advance the state of the art. For example:

- Modern network technology, e.g., based on Infiniband with an expected bandwidth of up to 600 Gb/s [25] in combination with remote direct memory access (RDMA) requires to rethink traditional algorithms, like–for example–shown in [19, 2, 23].
- Memory technology is also advancing in many different directions to optimize for extremely low latency (like multi-channel DRAM (MCDRAM) on the same die as the computing unit), extreme capacity, or non-volatility without compromising the byte-addressability property (e.g., [22]).
- Finally, processing units have experienced the most visible change with developments reaching out into different directions, e.g., extremely high parallelism of general-purpose architectures, domain-specific extensions and–most notably–a combination of both.

Therefore, based on the VLDB 2017 keynote presentation in Munich [18], this *Kurz Erklärt* documents an attempt to characterize the already existing diversity in the context of individual processing units.

1.1 The Starting Point

The first wave of a disruption in the area of processing units was the switch from increasing to maintaining the frequency in combination with increasing the number of cores of a single CPU. Current general-purpose server processors like the Intel Xeon Processor E5-2699 v4 provide 22 cores resulting in a direct support of 44 threads using hyper-threading for a single socket. As Fig. 1 shows, this trend may continue by keeping the frequency at the current

✉ Wolfgang Lehner
wolfgang.lehner@tu-dresden.de

Annett Ungethüm
annett.ungethuem@tu-dresden.de

Dirk Habich
dirk.habich@tu-dresden.de

[1] Database Research Group, Technische Universität Dresden, Dresden, Germany

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

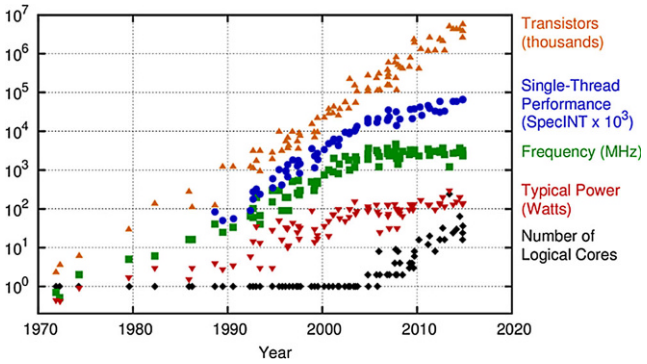

Fig. 1 Trend of the main processor characteristics (taken from: https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data)

level, but still increasing the number of cores on the same die using the growing number of transistors.

On the one hand, this already allows to build scale-up machines with ∼1 000 cores, e.g., HPE SGI UV 300 [24] including up to 48 TByte of DRAM. It goes without saying that such hardware infrastructures pose great challenges for database systems with respect to scalable data structures, design of centralized components like logging, or locking and latching. On the other hand, hardware and operating systems provide a rich API to control the hardware from an application perspective. For example, RAPL counters (on Intel platforms) allow to deploy detailed DFVS schemes (*Dynamic Frequency and Voltage Scaling*) for application-specific energy optimizations [17].

In parallel to general-purpose processing units, more specialized units like GPUs (more in this direction in Sect. 3) now provide tremendous computing capacities. For example, Fig. 2 illustrates the configuration of a high-performance computer listed within the Top 500[1] with a similar computing capacity as a state-of-the-art Nvidia Tesla V100[2] GPU (∼10T Euro). Within not even two decades, a large supercomputer weighting more than a 100 tons now fits on a single chip!

2 Development Directions of CPUs

The field of application for CPUs has expanded far beyond traditional servers and PCs. Emerging use-cases like for example augmented reality, edge-clouds, or machine learning have different priorities with respect to performance, energy consumption, and chip-size requirements. Due to this range of different requirements, different directions of development of CPUs emerged. Most relevant from a database system perspective is certainly the support of parallelism which might be considered for a single CPU at the core as well as at the instruction level.

At the core level, the number of cores per CPU varies heavily. While CPUs for the mobile or embedded application domain typically have between 2 and 8 cores, specialized server CPUs for HPC applications may have dozens of identical cores. For example, the Intel Xeon Phi KNL exhibits 72 HW cores resulting in 288 virtual cores by providing up to four hyper-threads per core. For the database system development, such infrastructures pose quite some challenges for the design of lock-free data structures or scalable scheduling algorithms.

Besides the instruction-level parallelism, there is also a growing potential of support for data parallelism. SIMD extensions (*Single Instruction Multiple Data*) allow to process multiple data items with a single instruction, which looks extremely promising for data-intensive applications like database systems. While almost all CPU architectures are providing SIMD extensions for many CPU generations known as SSE or AVX on the Intel and AMD cores as well as NEON for ARM cores, a significant step forward has been made in the recent past. On the one hand, larger registers can now be found on most modern CPUs. Having

[1] https://www.top500.org/featured/systems/asci-white-lawrence-livermore-national-laboratory/

[2] https://www.nvidia.com/en-us/data-center/tesla-v100/

Fig. 2 Comparison of Compute Power (Material provided by Norman May (SAP SE))

	Nvidia V100 (2017)	IBM ASCI White (2000)
Number of Processor Cores	3584	8192 (512 nodes x 16 IBM Power3)
Double-Precision Performance	7.5 TeraFLOPS	7.2 TeraFLOPS
NVIDIA NVLink™ v2 Interconnect Bandwidth	2x150 GB/s	N/A
PCIe x16 Interconnect Bandwidth	2x16 GB/s	N/A
Memory Capacity	16 GB	**6 TB DRAM** (Power 3 w/ **16 MB L2 cache**)
Max. overall data transfer speed	900 GB/s	?
Weight	450 gramm	106 tons
Energy consumption	**300W**	**3 MW**

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

initially started with 128 bit-wide registers, the current Intel AVX-512 extensions provide 512 bit-wide operations. On the other hand, the instruction set was extended to better support blending, comparing, and permuting operations in addition to conflict detection and support for floating-point arithmetic. While SIMD are heavily used in core database algorithms like scans [28] or compression algorithms [3] in column-stores, the design of efficient data structures fully leveraging the potential of SIMD extensions is still a challenging research task.

In addition to SIMD with the original motivation from image and text processing, recent developments also suggest to extend the instruction set of traditional CPU cores with a database-specific instruction set and corresponding additional elements like additional registers and larger memory interfaces. For example, a *dpCore* as part of a specialized *Data Processing Unit (DPU)* originally proposed within the Oracle Rapid project [1] offers single-cycle instructions like bit-vector load (BVLD), filter (FILT), and CRC32 hash-code generation to accelerate query operations like filters and joins. A similar approach is pursued by the Titan3D project [8] extending a general-purpose Xtensa LX5 core with instructions to support efficient bitwise decompression, hash code generation or efficient operators for bitmap index utilization.

3 Domain-specific Processing Units

With CPUs as general-purpose on the one side, domain-specific processing units may be considered the extreme on the other side of the diversity spectrum with GPUs probably serving as the most widely known representative of domain-specific processing units. GPUs are originally designed to process graphic pipelines including coordinate transformations and rasterization as well as hosting additional components, e.g., a display controller. While most of the elements were hard-wired and thus only a limited part of this functionality was programmable, the growing interest in GPUs has caused significant changes in the GPU-internal architectures. The graphic pipeline has become fully programmable, controlled by a user-defined sequence of so-called shaders allowing the GPU cores to be used for general-purpose computations. For example, Tesla GPUs of NVIDIA are now commonly used for HPC applications.

The term *GPGPUs* was coined to represent this development towards *General Purpose GPUs*. In order to support application development, different APIs like OpenCL or CUDA abstract the graphic pipeline by providing an abstract hardware model. Although significant support exists to ease the programming effort, it is still a demanding task to design efficient data-intensive algorithms fully exploiting the massive degree of parallelism provided by GPUs.

3.1 Acceleration Processing Units

GPUs or CPUs with application-specific instruction set extensions are representatives of attempts to support domain-specific requirements. Pushing the envelope by supporting specific application requirements exclusively leads into the field of domain-specific *acceleration processing units (ASICs)* usually deployed within a coprocessor model. While there exists–by design–a wide variety of different systems, ASICs for efficiently executing vector operations have a long tradition and are gaining–with Machine Learning as the driving application area–significant attention. For example, NEC provides a card-mounted vector processor SX-ACE [20] with a C++-based middleware to accelerate sparse matrix computations. Google recently announced a *Tensor Processing Unit (TPU)*, which aims at accelerating neuronal network computations [14]. Since the main task of a TPU is to multiply matrices, more than half of the chip is covered by the matrix multiplication unit including a 24MB scratchpad buffer. The remaining area is only used for the control, several interfaces, and other auxiliary components. The loss of general purpose flexibility and the focus on specialized application tasks results–as expected–in a significant performance gain of up to 30X compared to a corresponding implementation based on recent CPUs or GPUs [14].

While ASICs were developed to support compute-intensive tasks, ASICs also play a crucial role to support data-intensive tasks, which are directly relevant for efficient database systems. An example can be seen in the HARP ASIC to be found in systems of the SGI UltraViolet family (recently acquired by HPE). The HARP ASIC is responsible for connecting individual rack units (IRUs) to a cache-coherent NUMA system by maintaining cache coherency and providing a common address space. A HARP also hosts a *Global Reference Unit (GRU)* offering an API to offload memory operations within a NUMA-based system, e.g., functionality to asynchronously copy memory between processors and to accelerate atomic memory operations. As shown in [4] in greater detail, significant performance improvements can be achieved by using the specialized implementation of low-level memory management functions compared to variants offered by the operating system.

3.2 Reconfigurable Hardware

Domain-specific instruction set extensions or specific co-processors are usually developed to enhance the performance and to reduce power consumption for a specific task. However, they are hard-wired in a sense that once they are manufactured, they cannot be changed anymore. For situations with a higher degree of flexibility (e.g., changing application requirements) or for rapid-prototyping, recon-

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

figurable hardware especially Field Programmable Gate Arrays (FPGAs) lend themselves to providing a viable alternative to hard-wired ASICs. The core concept of FPGAs is based on an array of configurable logic blocks each implementing a binary function, which can be changed by modifying the truth table of the particular function. This table is written to the SRAM cells of the logic block and can be overwritten, thus yielding a reconfigurable processing unit. The number of these logic blocks as well as other specifications and the price vary heavily between different FPGAs. Simple development kits cost less than 100 € while more powerful boards cost up to 7 000 €. Besides differences in the interfaces, technology, and additional general-purpose cores, the applied FPGA chip is responsible for this huge price range. For example, expensive Intel/Altera boards use *Stratix FPGAs* with up to 5.5 million logic elements versus a *MAX 10* FPGA with only up to 50k elements on low-priced boards [5, 6]. Xilinx covers a similar range starting with the Spartan-6-boards in the entry-level segment and going up to the UltraScale+ for complex tasks. There are already some approaches to integrate high-end FPGAs into database systems. For example, [34] presents concepts and implementations for hardware acceleration for almost all important operators appearing in SQL queries. Here, an SQL query is analyzed and divided into single operators and modules, which are subsequently concatenated to a data path and loaded on the FPGA. A more general description of challenges for database developers working with FPGAs can be found in [26].

The ability to be reconfigurable obviously comes with some limitations compared to ASICs. First, the resources on an FPGA are limited, i.e., the specific task to be supported by an FPGA-based implementation has to be carefully selected; second, building takes some time. For example, the synthesis of an application for an FPGA can take between several minutes and multiple hours, depending on the complexity of the application and the available computing power. This is why the main part of debugging must be done in simulations. Finally, the size of an FPGA exceeds the size of most hard-wired solutions by several orders of magnitude. For comparison, a device package with the Xilinx Spartan-7 FPGA covers between 64 mm^2 and 729 mm^2 [31], while the Titan3D uses only 4.95 mm^2 [8].

4 Mixing it all together

Modern computing infrastructure are *rack-scale*, i.e., consist of multiple, usually heterogeneous components. Thus, different processing units can be combined in various ways at different levels in a more or less tightly coupled fashion.

4.1 Loosely-coupled Systems

At the one end, a rack may consist of multiple multi-socket machines either with or without common address space built up by homogenous or heterogeneous system components. In the smaller scale, individual machines may comprise one (or multiple) CPUs and one (or multiple) acceleration cards traditionally using a PCIe-based infrastructure for communication. A typical instance of such a pattern is reflected in the CPU-GPU combination with a CPU driving the machine and the GPU acting as a co-processor. Having multiple GPUs connected to each other (for example using NVLink v2 with 150GB/s) leads to GPGPU clusters usually used for high-performance computing scenarios, but also used for data processing tasks, e.g., as reported in [33].

Although, all domain-specific processing units are connected traditionally to the main board using the internal bus system, FPGA boards are also available in a variety of flavors, ranging to standalone boards (reachable via Ethernet) to tightly-coupled alternatives on the same board (see below). Accelerator cards can be integrated into an application by using for example the Coherent Accelerator Processor Interface (CAPI) originally developed by IBM[10]. Exploiting the interface for database systems (in the context of NoSQL systems) is, for example, reported in [11].

Specifically for database applications, the overarching question remains to decide where to place data and corresponding operators/operator fragments in such a loosely-coupled system with usually significant communication overhead to transfer intermediate result data. An approach to optimize query offloading from a host PC to an FPGA board is demonstrated in [29, 30]; a more abstract and cost-based solution for OpenCL compatible processing units is given in [15, 16].

4.2 Coupling on a Chip

The more interesting approach to couple individual processing units is putting multiple components on a single die and building complex *System on a Chip*-units (SoCs). As a starting point, regular CPUs consist–as already mentioned–of multiple homogenous cores with private and/or common caches as well as a communication infrastructure. The communication infrastructure (usually called the *Network-on-a-Chip* (NoC) is traditionally organized in a ring structure. Modern many-core systems like Intel's Xeon Scalable processors are moving towards a mesh network to reduce latency as well as increase the internal bandwidth. Starting from traditional CPUs with multiple homogenous cores, a huge variety of different heterogenous SoCs emerged within the recent past.

A first step can be seen in the big.LITTLE architecture, which features a combination of more powerful cores and

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

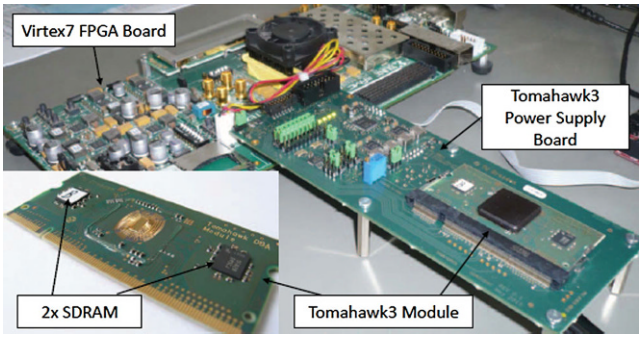

Fig. 3 Measurement setup of the Tomahawk3 (Figure taken from [7])

slower low-power cores, which can be found on low-power systems like the Ordoid [9]. This mix of binary-compatible cores on the same chip allows–depending on the current performance requirement–an energy-efficient scheduling of individual tasks sending unused cores into a sleep mode. An example of an energy-aware scheduling with respect to typical database workload patterns was demonstrated in [27].

Moving more towards heterogeneous systems, almost all combinations of CPU-style and domain-specific processing units can be found. The most used combinations (usually in every laptop) consists of a CPU-GPU combination. AMD coined the term *Accelerated Processing Unit* (APU) which even allows to access the same memory from CPU as well as GPU side. While the initial motivation stems from optimizing for space and thermal constraints, this architecture makes it attractive for off-loading specific operators without the need to explicitly ship data between host and acceleration card. The excessive use of GPUs for deep-learning applications has driven their recent enhancements with tensor processing units. One of the first GPUs featuring Tensor cores is the Tesla V100 with the GV100 chip, released mid 2017. In addition to the normal CUDA cores, the V100 includes also 640 Tensor Cores. This shows up to 12× times higher peak performance in deep learning compared to the predecessor P100[21].

Pairing CPUs with FPGA presents another extremely relevant combination for data-intensive applications. Interestingly, this combination can be seen from two perspectives. On the one hand, a traditional strong CPU can be *supported* by an FPGA. As an example, Intel recently announced to pair Xeon CPUs with FPGAs in the context of their *Hardware Accelerator Research Program* (HARP) [13]. On the other hand, FPGAs are extended with *traditional* general-purpose cores, usually based on the ARM architecture. Again, Intel offers (after the acquision of Altera) a wide range of FPGA/ARM core SoCs [12]. Also Xilinx provides FPGA boards in different performance- and cost-ranges with up to 8 ARM cores (2×4 big.LITTLE) and a large DRAM block [32].

As a final step, also *application-specific* extensions can be used to build *application-specific* SoCs. As reported in [1], a highly optimized DPU for database operators consists of 32dpCores (as outlined above) grouped into 4 clusters with a Power Management Unit, an ARM Cortex-A9 dual core processor and controllers for PCIe and DRAM access. Fabricated using a 40 nm process, the overall DPU results in a silicon area of 90.63 mm^2 and 540 millions transistors, of which 268 million transistors (roughly half) are used for low-latency scratchpad memory.

A similar approach is pursued within the Tomahawk project [7] to build an extremely energy-efficient SoC including database-optimized cores based on Titan3D (as outlined above). A Tomahawk3 (third generation) contains four cores with instruction set extensions plus a fifth core (core manager) to orchestrate the worker cores. An FPGA interface allows to provide the whole functionality of the programmable logic, but also to leverage the interfaces on the board, e.g., Ethernet and USB. Figure 3 shows a system setting with a Tomahawk3 mounted onto an FPGA board. The Tomahawk3 chip covers an area of 18 mm^2 and is produced in 28nm SLP CMOS technology. Two LPDDR2 interfaces allow a connection to two RAM modules[7]. As shown in Fig. 3, these RAM modules are on the same module as the Tomahawk3 minimizing the memory access time.

5 Summary

As shown, the development of processing units is astonishing! We currently see not only a growth of general-purpose cores within a single socket, but already a plethora of combinations with domain-specific processing units. Interestingly for the database community, hardware development is considering the shift from pure number-crunching (HPC-style) applications to data-crunching applications by providing hardware designs for efficient movement between the individual components within a chip but also beyond chip boundaries. However, leveraging this opportunity provided by the hardware does not come for free. The design of database systems requires a re-thinking on all levels – from the individual data structure via enhancements in query optimization to optimization of scheduling of different operators during runtime.

Acknowledgements This work is partly funded by the German Research Foundation (DFG) in the Collaborative Research Center 912 *Highly Adaptive Energy-Efficient Computing* and within the Cluster of Excellence *Center for Advancing Electronics Dresden* (Orchestration Path).

Final edited form was published in "Datenbank-Spektrum". 18 (1), S. 57–62. ISSN: 1610-1995.
https://doi.org/10.1007/s13222-018-0276-y

References

1. Agrawal SR, Idicula S, Raghavan A, Vlachos E, Govindaraju V, Varadarajan V, Balkesen C, Giannikis G, Roth C, Agarwal N, Sedlar E (2017) A many-core architecture for in-memory data processing. In: MICRO, pp 245–258
2. Barthels C, Loesing S, Alonso G, Kossmann D (2015) Rack-scale in-memory join processing using rdma. In: SIGMOD, pp 1463–1475
3. Damme P, Habich D, Hildebrandt J, Lehner W (2017) Lightweight data compression algorithms: an experimental survey (experiments and analyses). In: EDBT, pp 72–83
4. Dreseler M, Kissinger T, Dürken T, Lüubke E, Uflacker M, Habich D, Plattner H, Lehner W (2017) Hardware-accelerated memory operations on large-scale numa systems. In: ADMS@VLDB
5. FPGA I (2018a) Intel max 10 fpgas – overview. https://www.altera.com/products/fpga/max-series/max-10/overview.html. Accessed 5 Jan 2018
6. FPGA I (2018b) Intel stratix 10 fpgas – overview. https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html. Accessed 5 Jan 2018
7. Haas S, Arnold O, Nöthen B, Scholze S, Ellguth G, Dixius A, Höppner S, Schiefer S, Hartmann S, Henker S et al (2016) An mpsoc for energy-efficient database query processing. In: DAC, pp 1–6
8. Haas S, Scholze S, Höppner S, Ungethüm A, Mayr C, Schüffny R, Lehner W, Fettweis G (2017) Application-specific architectures for energy-efficient database query processing and optimization. Microprocess Microsyst 55:119–130
9. HardKernel (2018) Odroid-xu3. http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127. Accessed 10 Jan 2018
10. IBM (2014) Coherent accelerator processor interface (capi) for power8 systems, white paper. https://www-304.ibm.com/webapp/set2/sas/f/capi/CAPI_POWER8.pdf. Accessed 2 Jan 2018
11. IBM (2016) Data engine for nosql – power systems edition. http://www.ibm.biz/capiflash. Accessed 2 Jan 2018
12. Intel (2018a) Intel fpgas – socs – product overview. https://www.altera.com/products/soc/overview.html. Accessed 1 Jan 2018
13. Intel (2018b) Intel hardware accelerator research program. https://software.intel.com/en-us/hardware-accelerator-research-program. Accessed 5 Jan 2018
14. Jouppi NP et al (2017) In-datacenter performance analysis of a tensor processing unit. https://arxiv.org/pdf/1704.04760.pdf. Accessed 2018-01-03
15. Karnagel T, Habich D (2017) Heterogeneous placement optimization for database query processing. it Inf Technol 59(3):117
16. Karnagel T, Habich D, Lehner W (2017) Adaptive work placement for query processing on heterogeneous computing resources. Proceedings VLDB Endowment 10(7):733–744
17. Kissinger T, Habich D, Lehner W (2018) Adaptive energy-control for in-memory database systems. In: SIGMOD
18. Lehner W (2017) The data center under your desk – how disruptive is modern hardware for db system design? Proceedings VLDB Endowment 10(12):2018–2019
19. Li F, Das S, Syamala M, Narasayya VR (2016) Accelerating relational databases by leveraging remote memory and rdma. In: SIGMOD, pp 355–370
20. NEC (2017) Nec accelerates machine learning for vector computers. http://www.nec.com/en/press/201707/global_20170703_02.html. Accessed 4 Jan 2018
21. Nvidia (2017) Volta architecture whitepaper. http://www.nvidia.com/object/volta-architecture-whitepaper.html. Accessed 5 Jan 2018
22. Oukid I, Kettler R, Willhalm T (2017) Storage class memory and databases: opportunities and challenges. it Inf Technol 59(3):109
23. Salama A, Binnig C, Kraska T, Scherp A, Ziegler T (2017) Rethinking distributed query execution on high-speed networks. IEEE Data Eng Bull 40(1):27–37
24. SGI (2016) SGI UV 300H for SAP HANA. https://www.sgi.com/pdfs/4554.pdf. Accessed 2018-01-03
25. InfiniBand Trade Association (2015) Infiniband roadmap. http://www.infinibandta.org/content/pages.php?pg=technology_overview. Accessed 2018-01-03
26. Teubner J (2017) Fpgas for data processing: current state. it Inf Technol 59(3):125–131
27. Ungethüm A, Kissinger T, Mentzel W, Habich D, Lehner W (2016) Energy elasticity on heterogeneous hardware using adaptive resource reconfiguration LIVE. In: SIGMOD, pp 2173–2176
28. Willhalm T, Popovici N, Boshmaf Y, Plattner H, Zeier A, Schaffner J (2009) Simd-scan: ultra fast in-memory table scan using on-chip vector processing units. Proceedings VLDB Endowment 2(1):385–394
29. Woods L, Istvan Z, Alonso G (2013a) Hybrid fpga-accelerated sql query processing. In: FPL, pp 1–1
30. Woods L, Teubner J, Alonso G (2013b) Less watts, more performance: an intelligent storage engine for data appliances. In: SIGMOD, pp 1073–1076
31. Xilinx (2017) Xilinx data sheets – spartan-7 series. https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf. Accessed 1 Jan 2018
32. Xilinx (2018) Xilinx programmable devices – product overview. https://www.xilinx.com/products/silicon-devices/soc.html. Accessed 1 Jan 2018
33. Young J, Wu H, Yalamanchili S (2012) Satisfying data-intensive queries using GPU clusters. In: SC companion: high performance computing, networking storage and analysis, p 1314
34. Ziener D, Bauer F, Becher A, Dennl C, Meyer-Wegener K, Schürfeld U, Teich J, Vogt JS, Weber H (2016) Fpga-based dynamically reconfigurable sql query processing. ACM Trans Reconfigurable Technol Syst 9(4)