Implications of Emerging 3D GPU Architecture on the Scan Primitive

Jason Power
powerjg@cs.wisc.edu

Yinan Li
yinan@cs.wisc.edu

Mark D. Hill
markhill@cs.wisc.edu

Jignesh M. Patel
jignesh@cs.wisc.edu

David A. Wood
david@cs.wisc.edu

Department of Computer Sciences
University of Wisconsin–Madison

ABSTRACT

Analytic database workloads are growing in data size and query complexity. At the same time, computer architects are struggling to continue the meteoric increase in performance enabled by Moore's Law. We explore the impact of two emerging architectural trends which may help continue the Moore's Law performance trend for analytic database workloads, namely 3D die-stacking and tight accelerator-CPU integration, specifically GPUs. GPUs have evolved from fixed-function units, to programmable discrete chips, and now are integrated with CPUs in most manufactured chips. Past efforts to use GPUs for analytic query processing have not had widespread practical impact, but it is time to re-examine and re-optimize database algorithms for massively data-parallel architectures. We argue that high-throughput data-parallel accelerators are likely to play a big role in future systems as they can be easily exploited by database systems and are becoming ubiquitous. Using the simple scan primitive as an example, we create a starting point for this discussion. We project the performance of both CPUs and GPUs in emerging 3D systems and show that the high-throughput data-parallel architecture of GPUs is more efficient in these future systems. We show that if database designers embrace emerging 3D architectures, there is possibly an order of magnitude performance and energy efficiency gain.

1. INTRODUCTION

Due to recent technology trends including the continuation of Moore's law [16] and the breakdown of Dennard scaling [4], computing has become energy-limited. Although device manufacturers are continuing to add more transistors per chip (Moore's law), the threshold voltage of the transistors is not decreasing at the same relative rate (the breakdown in Dennard scaling) [4]. Unlike in the past, database designers cannot rely on computer architects to give increased performance for free; future devices with double performance will consume almost double the energy. Using ITRS projections, Esmaeilzadeh et al. found that by 2024 we can only expect a 7.9× average speedup compared to today's processors from traditional architectural optimizations, more than 24× less than if performance had continued to follow Moore's law [7].

However, there are two architectural trends that can help mitigate this bleak projection: 3D-die stacking and tightly-integrated accelerators. With 3D die-stacking, multiple silicon dies are stacked on top of one another (see Section 4 for more details). This allows tightly-integrating accelerators with traditional CPUs to become more economical and common. Additionally, 3D die-stacking enables very high-bandwidth memory systems, up to 1 TB/s in some projections [1], and decreases the energy for communication by a factor of 3× [3].

Computer architects are already creating tightly-integrated general-purpose commodity hardware accelerators. SIMD hardware (short vector units e.g., SSE and AVX) is one example of this trend. Now, like SIMD units, general-purpose graphics processing units (GPGPUs) are moving onto the CPU die and becoming first-order compute platforms. Today 99% of Intel's and 67% of AMD's desktop CPUs ship with an on-die GPU [18], and server chips with integrated GPUs have been announced [2]. In addition to tightly integrating GPUs and CPUs physically, new programming models, like heterogeneous system architecture (HSA) [19] enable these systems to be easily programmed.

SIMD units, GPGPUs, and other data-parallel architectures can have increased performance and energy efficiency compared to CPU platforms because they have lower overhead per processing element. For instance, in NVIDA's Kepler GPU architecture, 192 processing elements share a fetch, decode, and scheduling unit, compared to 8 processing elements sharing a front-end in Intel's Haswell architecture. CPUs can waste 90% of the instruction execution energy on the front-end pipeline [11], while GPU architecture amortizes this overhead across tens or hundreds of hardware threads. Additionally, through software-managed caches, large register files, data-caches optimized for throughput, and a high-degree of hardware multithreading, GPUs can effectively use hundreds of GB/s of main memory bandwidth, significantly improving performance and energy for bandwidth-bound applications. Current CPU memory systems are optimized for minimizing instruction latency with small buffers and large general-purpose caches. Thus, for data-parallel applications, CPUs are less efficient than optimized for high-throughput data-parallel hardware.

In this work, we focus on the scan operation. Scans are an important primitive and the workhorse in high-performance in-memory database systems like SAP HANA, Oracle Exalytics, IBM DB2 BLU and Facebook's Scuba. Scans are data-parallel operations, and a series of scan algorithms have been developed in the database community to exploit this parallelism using hardware artifacts such as the parallelism within regular ALU words (e.g. [14]), and SIMD to accelerate scans (e.g. [14, 21, 12, 5, 22]).

However, the parallelism from using CPU SIMD extensions is limited. SIMD hardware sits in an architecture design that is not optimized for high-throughput data processing. The latency-centric CPU memory hierarchy and execution model limit the effectiveness of the SIMD units. Using an architecture explicitly designed to efficiently execute data-parallel applications can significantly increase the efficiency of processing database scans.

Figure 1 shows the energy and performance tradeoff between CPU and GPU architectures. Even though discrete GPUs have more computation resources and higher bandwidth than CPUs, the performance improvement is overwhelmed by overheads due to copying data and operating system interaction. In fact, the discrete GPU results in higher response time (4×) and energy (6×) than a four core CPU; this behavior validates why GPUs have largely been ignored by database systems for scan processing today. However, we find that today's integrated GPUs mitigate many of these problems. But, they only provide a marginal benefit (17%) over the CPU since they are limited by the same memory interface. Thus, the benefit of integrated GPUs is also limited for database scan operations. However, as we project into the future and examine 3D die-stacked systems, GPUs, or other highly data-parallel architectures, can provide a large benefit over current CPU platforms (15×) and also 3D-stacked CPU systems (4×). Thus, the use of GPU-like high-throughput hardware can not be ignored by database designer in the near future—this is the central message of this paper. We provide initial evidence in support of this position by examining the implications for the database scan operator. (We acknowledge that future work is essential to expand this argument to other data processing operations).

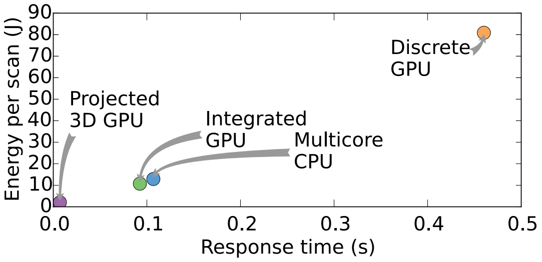


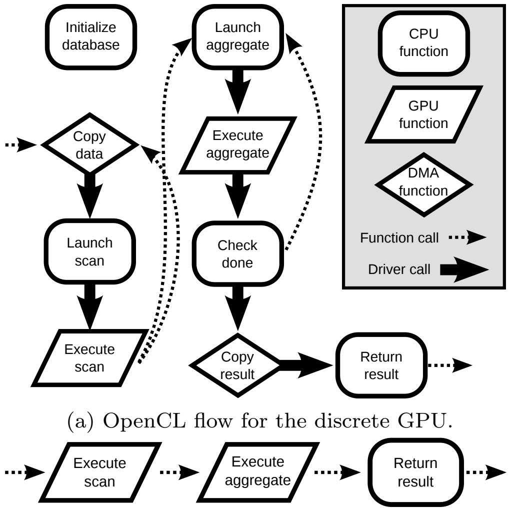
Figure 1: Performance and energy of scans on a multicore CPU, discrete and integrated GPUs, and a projection on a future high-bandwidth, highly data-parallel architecture. Lower and to the left is best.

2. GPGPU BACKGROUND

Graphics processing units (GPUs) have recently become more easily programmable, creating the general purpose GPU (GPGPU) computing landscape. This trend began with 1st-generation *discrete GPUs* that are connected to the system via the PCIe bus which has higher latency and lower bandwidth than main memory. Now, 2nd-generation *integrated GPUs* that share the same silicon chip are becoming mainstream. With emerging 3D die-stacking technology, we will soon see 3rd-generation *3D GPUs* that combine the high performance of discrete GPUs with the low latency and simple programming models of integrated GPUs.

There are three key differences between GPUs and CPUs that make GPUs, especially future GPUs, an important platform for database designers to consider. First, GPGPUs employ very wide data-parallel hardware. For instance, an AMD HD7970 can operate on 131,072 bits in parallel, compared to only 256–512 bits in modern CPU's SIMD hardware. Second, GPGPUs are programmed with SIMT (single-instruction multiple-thread) instead of SIMD (single-instruction multiple-data). The SIMT model simplifies programming the GPU's wide data-parallel hardware. For instance, SIMT allows arbitrary control flow between individual data-parallel lanes.

Finally, and importantly, GPU architecture can be more energy-efficient than CPU architecture for



(a) OpenCL flow for the discrete GPU.

(b) HSA flow for the integrated GPU.

Figure 2: Flowchart showing how the CPU and GPU interact in BitWarp.

certain workloads (e.g., database scans). Since many data-parallel lanes share a single front-end (instruction fetch, decode, etc.), this per-instruction energy overhead is amortized. On CPU architectures, the execution front-end and data movement consumes 20–40× more energy than the actual instruction [13]. Additionally, all of the parallelism is explicit for GPUs through the programming model, while CPUs require high energy hardware (like the re-order buffer and parallel instruction issue) to implicitly generate instruction-level parallelism, which wastes energy for data-parallel workloads.

As GPU hardware has become more closely integrated with CPU hardware, the GPU programming models have become more integrated with CPU programming models as well. In this paper, we use the heterogeneous system architecture (HSA) runtime to program the GPU. HSA presents the programmer with a coherent and unified view of memory and low-latency user-level API for using the GPU [19].

Past GPU APIs have significant overheads because they assume the discrete GPU model with high latency and low bandwidth communication between the CPU and GPU. These overheads can significantly affect application performance. We find that the discrete GPU performance is 16× slower than its potential because of these overheads.

Figure 2 shows the difference between the traditional GPU APIs (2a), and new programming models like HSA (2b). This figure shows the steps required to execute a full query on these two systems. The HSA flowchart elides both the operating system driver overheads and the data copies. Because of these changes, not only does the application perform better, but it is simpler to program as well.

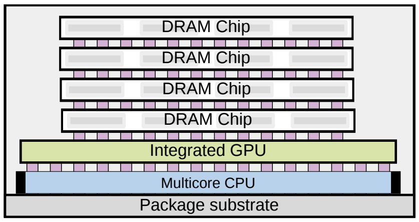


Figure 3: A potential future 3D architecture. All components shown would occupy a single socket.

3. BITWARP IMPLEMENTATION

To study scans on GPUs, we leverage previous work accelerating analytical query processing on the CPU: the BitWeaving scan algorithm [14]. BitWeaving outperformed state-of-the-art scan algorithms by leveraging intra-word parallelism on CPU hardware and is similar to some industry solutions. We have modified the BitWeaving algorithm to execute efficiently on the GPU. The main change in the algorithm is increasing the logical execution width to the size of the GPU's data-parallel units.

BitWeaving uses a coded columnar layout, packing multiple codes per word. This mechanism allows BitWeaving to leverage the 64-bit wide CPU word to execute the scan predicate on many column codes in a single cycle. In BitWarp, we leverage the entire 4096-bit lane width of the GPU. BitWeaving includes two different algorithms, horizontal and vertical, which refer to the way the codes are packed. In this work, we focus only on BitWeaving vertical, as it performs better than horizontal in all cases on the GPU. BitWeaving vertical packs the codes such that one bit of each code is in each consecutive word. The paper by Li and Patel details the algorithm and implementation of BitWeaving [14].

4. 3D ARCHITECTURE

A recent architectural trend is that "Die-stacking is happening" [3] due to advances in fabrication, cooling, and other technologies. There are many wide-ranging implications of die-stacking from device manufacturing to system design. Die-stacking has two important consequences for database designers: higher memory bandwidth and increased compute capability. Some project 1 TB/s of bandwidth, more than 40× the bandwidth available today for CPUs [1, 6, 17]! Die-stacking also moves memory closer to the computation resulting in lower energy: a 3× reduction for first-generation devices [3]. Also, die stacking allows multiple compute chips (e.g. CPU and GPU dies) to be packaged together.

Figure 3 shows a potential architecture which leverages 3D die-stacking. In 3D integration, separate silicon dies are stacked directly on top of one another and connected by through-silicon vias (TSVs). TSVs provide a high-bandwidth, low-latency, and low-energy interconnect. In the architecture in Figure 3, TSVs connect the CPU and GPU, yielding performance similar to an integrated die. Additionally, TSVs connect both the CPU and GPU to die-stacked DRAM. In such a system, we expect performance similar to a discrete GPU, without any overheads, and power similar to today's integrated chip. Figure 3 represents what is in one CPU socket in a motherboard in future systems. 2.5D integration (not shown) is similar to 3D integration, except chips are connected by TSVs in a silicon interposer instead of directly stacked. 2.5D die-stacking has similar characteristics to 3D stacking.

In addition to leveraging the high bandwidth of TSVs, 3D die-stacked architectures also allow CPU and GPU cores to use manufacturing processes customized specifically for CPU or GPU needs. For current CPU-only chips, hardware manufacturers optimize their CMOS manufacturing process to reduce latency, using faster, less energy-efficient transistors and thicker, wider wires that limit bandwidth in exchange for lower latency. For current GPU-only chips, hardware manufacturers optimize their CMOS processes to increase bandwidth (at the expense of latency), using slower, lower-leakage transistors and thinner, narrower wires that maximize intra-chip bandwidth, but also significantly increase latency. For current integrated architectures—where the CPU and GPU share the same silicon die—hardware manufacturers must strike a compromise between the incompatible, conflicting demands of CPUs and GPUs. 3D die-stacking allows the CPU and GPU to be manufactured on separate chips using appropriately optimized manufacturing processes, increasing the performance and energy-efficiency of both CPUs and GPUs.

Figure 3 shows one of many possible architectures which leverage 3D or 2.5D die-stacking to increase memory bandwidth and integrate a highly data-parallel architecture. There are many ways to architect the system to take advantage of these trends of increased bandwidth and compute capability. Determining the best system architecture for 3D-stacked systems is an interesting direction for future work.

5. METHODOLOGY

There are many possible designs for a highly data-parallel system. Some examples include re-architecting CPUs by increasing the vector SIMD width and changing their cache management policies, Intel's Xeon Phi processor which has 72 simple in-order cores with wide SIMD lanes [9], and GPU architecture which is an example of a highly data-parallel architecture that has already shown economic viability. Although there are many possible embodiments, we believe that any highly data-parallel architecture will share many characteristics with GPGPUs. These characteristics include:

- Large number of simultaneous threads to generate the memory-level parallelism needed to hide memory latency,
- Wide vector execution units that can issue vector memory accesses, and
- Many execution units to operate on data at memory-speed.

Additionally, it is likely that programming these devices will be similar to programming current GPGPUs. For instance, OpenCL is a flexible language which can execute on GPUs, CPUs, and other accelerators like the Xeon Phi. We focus on GPGPUs as an example data-parallel architecture that has already shown economic viability.

For a constant comparison point, we use AMD CPU (A10-7850K) and GPU (HD7970) platforms in our evaluation. We use a four core CPU at 3.7 GHz and two different GPUs, an integrated GPU that is on the same die as the CPU, and a discrete GPU connected to the CPU via the PCIe bus. The GPUs have 8 CUs at 720 MHz and 32 CUs at 1125 MHz, respectively. The theoretical memory bandwidth for the CPU-GPU chip is 21 GB/s, and the discrete GPU's memory bandwidth is 264 GB/s. We use a single-socket system in our evaluation, but integrated CPU-GPU chips should scale to multiple sockets similar to CPU-only chips.

We use the discrete GPU as a model to predict the performance of the future die-stacked system. According to the projection in [1, 6, 17], 264 GB/s is a reasonable assumption for the memory bandwidth in a first-generation die-stacked system, and the 32 CU chip in the discrete GPU will fit into a package like Figure 3.

6. RESULTS

To measure the efficacy of the GPU for the scan operation, we measured the performance, power, and energy consumed for the CPU and GPU hardware. Figure 4 shows the time per scan over a 1 billion entry column (about 1 GB of data).

The multicore CPU is not an efficient platform for performing the scan primitive. Figure 4 shows that the speedup of four cores over one core is only 60%. Scan is an embarrassingly parallel operation, so we expect almost perfect scaling from the scan algorithm. The reason the CPU does not scale linearly is that the CPU memory system is not designed to

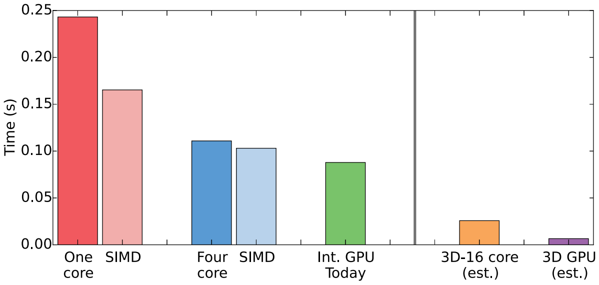


Figure 4: Performance of scan on 1 billion 10-bit codes. Averaged over 1000 scans.

support high bandwidth applications. CPU caches are built for low-latency, not high bandwidth.

The integrated GPU sees a small improvement over the multicore with SIMD, a 17% speedup. This improvement is because the integrated GPU's more efficient cache hierarchy. GPUs are designed for these types of high-bandwidth applications and have a separate memory controller which can exploit the memory-level parallelism of the application better than the CPU.

Figure 5 shows the power and energy consumed by the CPU and GPU when performing 1000 scans. The data was obtained by measuring the full system power; thus, it includes all of the system components (e.g., disk, motherboard, DRAM, etc.). The right side of the figure shows the total energy consumed (power integrated over time).

Figure 5 shows that even though the four core configuration and the integrated GPU take more power than the one core CPU, they execute much faster, resulting in lower overall energy. Additionally, the integrated GPU is more efficient than the four core CPU in power and performance.

In the future, it's likely that the GPU will become even more energy efficient compared to the CPU. Each GPU compute unit (CU) (similar to a CPU core) has lower power per performance than a CPU core [13]. Also, each GPU CU is smaller area per performance than CPU cores. Thus, there can be many more GPU CUs than CPU cores, exemplified by our test platform.

6.1 3D Architecture Performance

There are two characteristics of 3D architecture that significantly affects the performance of the scan primitive. First, by stacking a GPU die with the CPU die, each processor type is more highly optimized and can take more total area, thus increasing the overall compute capability of the system. Second, since the chip-memory interconnect uses TSVs, the memory bandwidth is orders of magnitude higher than current systems. These attributes together create a higher performance and more energy efficient system than today's architectures.

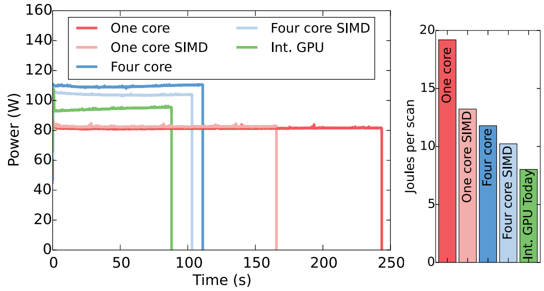


Figure 5: Power for 1000 scans and energy per scan.

The right side of Figure 4 shows the estimated performance of the CPU and the GPU in a 3D die-stacked system. To estimate the CPU's speedup, we assume that adding 4× more cores will result in a 4× speedup. This is a very aggressive estimate; the CPU is unlikely to get perfect speedup when adding more cores. There is only a 60% speedup from one to four cores; however, the added bandwidth will increase the CPU performance too. We err on the side of overestimating the CPU performance.

To estimate the 3D GPU's performance, we run the scan operation on a current discrete GPU with 32 CUs (4× more execution resources than the integrated GPU) with a bandwidth similar to future die-stacked systems [6]. We discard all of the overheads associated with using the discrete GPU, since on a die-stacked system, the overheads will be minimal, as we found with the integrated GPU.

Using this projection, we find that a die-stacked GPU system can provide 15.7× higher performance than today's non-stacked multicore CPU system. The 3D GPU is also 3.9× faster than the aggressive 3D CPU estimate.

In addition to the performance gains, 3D die-stacked DRAM enables lower energy per access by using smaller and shorter wires than off-chip DRAM. Coupled with the GPU's efficiency, it is likely that the energy benefit of this system is much higher than the performance benefit.

We predict that to take advantage of the increasing memory bandwidth from 3D die-stacking, database designers must embrace high-bandwidth, highly data-parallel architectures, and we have shown that the GPU is a good candidate. Looking forward, CPU architecture will continue to become more efficient, including for data-parallel workloads. However, GPUs will also increase in efficiency at a similar rate. We believe that as these trends come to fruition, it will become increasingly important for database designers to leverage high-bandwidth data-parallel architectures to keep pace with the highest possible performance.

7. CONCLUSIONS AND FUTURE WORK

Previous works have shown the huge potential of using GPUs for database operations (e.g., [8, 10, 15, 20]). However, many of these works have not included the large discrete GPU overheads when operating on large in-memory databases. We show current physically and logically integrated GPUs mitigate the problems with discrete GPUs and show a modest speedup and energy reduction over multicore CPUs for scan operations.

Looking forward, computer architects are pursuing many interesting avenues to increase the memory bandwidth significantly, such as 3D die-stacking. However, conventional multicore CPU architecture is not well suited to efficiently use this increased memory bandwidth. We advocate that to take advantage of these architectural trends, database designers should look to data-parallel accelerators, of which the GPU is one example. If database designers embrace these new architectures, there is possibly an order of magnitude performance and energy efficiency gain for scans. Examining these issues for a wider range of database workloads is an interesting direction for future work.

8. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (CCF-1218323, CNS-1302260, CCF-1438992, IIS-0963993, IIS-1110948, and IIS-1250886), the Anthony Klug NCR Fellowship, Cisco Systems Distinguished Graduate Fellowship, Google, and the University of Wisconsin (Kellett award and Named professorship to Hill). Hill and Wood have a significant financial interest in AMD and Google. Patel has a significant financial interest in Microsoft and Quickstep Technologies.

9. REFERENCES

[1] 3D-ICs. `http://www.jedec.org/category/technology-focus-area/3d-ics-0`, 2012. Accessed: 2014-6-24.

[2] AMD. AMD Opteron X2100 Series APU. `http://www.amd.com/en-us/products/server/x2100`. Accessed: 2014-5-28.

[3] B. Black. MICRO 46 Keynote: Die Stacking is Happening, 2013.

[4] M. Bohr. A 30 Year Retrospective on Dennard's MOSFET Scaling Paper. *IEEE Solid-State Circuits Newsletter*, 12(1):11–13, Jan. 2007.

[5] Z. Chen, J. Gehrke, and F. Korn. Query optimization in compressed database systems. In *SIGMOD Conference*, pages 271–282, 2001.

[6] H. Consortium. Hybrid Memory Cube Specification 1.0, 2013.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *ISCA Conference*, pages 365–376, 2011.

[8] N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, and D. Manocha. Fast computation of database operations using graphics processors. In *SIGMOD Conference*, page 215, 2004.

[9] R. Hazra. Accelerating insights in the technical computing transformation. In *Intl. Supercomputing Conf.*, 2014.

[10] B. He, K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander. Relational joins on graphics processors. In *SIGMOD Conference*, page 511, 2008.

[11] M. Horowitz. Computing's Energy Problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference*, pages 10–14, 2014.

[12] R. Johnson, V. Raman, R. Sidle, and G. Swart. Row-wise parallel predicate evaluation. *PVLDB*, 1(1):622–634, 2008.

[13] S. W. Keckler. Life after Dennard and How I Learned to Love the Picojoule. In *MICRO 44 Keynote*, 2011.

[14] Y. Li and J. M. Patel. BitWeaving: fast scans for main memory data processing. In *SIGMOD Conference*, pages 289–300, 2013.

[15] M. D. Lieberman, J. Sankaranarayanan, and H. Samet. A Fast Similarity Join Algorithm Using Graphics Processing Units. In *ICDE Conference*, pages 1111–1120, Apr. 2008.

[16] G. Moore. Cramming More Components Onto Integrated Circuits. *Electronics*, pages 114–117, Jan. 1965.

[17] J. T. Pawlowski. Hybrid Memory Cube (HMC). In *Hot Chips 23*, 2011.

[18] J. Peddie. Market Watch. Technical Report Q1'04, Jon Peddie Research, 2014.

[19] P. Rogers. Heterogeneous System Architecture Overview. In *Hot Chips 25*, 2013.

[20] N. Satish, C. Kim, J. Chhugani, A. D. Nguyen, V. W. Lee, D. Kim, and P. Dubey. Fast sort on cpus and gpus: a case for bandwidth oblivious SIMD sort. In *SIGMOD Conference*, pages 351–362, 2010.

[21] T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units. *PVLDB*, 2(1):385–394, 2009.

[22] J. Zhou and K. A. Ross. Implementing database operations using SIMD instructions. In *SIGMOD Conference*, pages 145–156, 2002.