Database Systems on GPUs

Other titles in Foundations and Trends® in Databases

Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases
Gerhard Weikum, Xin Luna Dong, Simon Razniewski and Fabian Suchanek
ISBN: 978-1-68083-836-7

Cloud Data Services: Workloads, Architectures and Multi-Tenancy
Vivek Narasayya and Surajit Chaudhuri
ISBN: 978-1-68083-774-2

Data Provenance
Boris Glavic
ISBN: 978-1-68083-828-2

FPGA-Accelerated Analytics: From Single Nodes to Clusters
Zsolt István, Kaan Kara and David Sidler
ISBN: 978-1-68083-734-6

Distributed Learning Systems with First-Order Methods
Ji Liu and Ce Zhango
ISBN: 978-1-68083-700-1

Database Systems on GPUs

Johns Paul
National University of Singapore

Shengliang Lu
National University of Singapore

Bingsheng He
National University of Singapore
hebs@comp.nus.edu.sg

the essence of knowledge
Boston — Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

J. Paul, S. Lu and B. He. *Database Systems on GPUs*. Foundations and Trends® in Databases, vol. 11, no. 1, pp. 1–108, 2021.

ISBN: 978-1-68083-849-7

Foundations and Trends® in Databases
Volume 11, Issue 1, 2021

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Databases publishes survey and tutorial articles in the following topics:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians

Foundations and Trends® in Databases, 2021, Volume 11, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Contents

Database Systems on GPUs

Johns Paul[1], Shengliang Lu[2] and Binsheng He[3]

[1]*National University of Singapore*

[2]*National University of Singapore*

[3]*National University of Singapore; hebs@comp.nus.edu.sg*

ABSTRACT

This article gives an overview of history and recent developments in database systems on graphics processing units (GPUs). GPU, which was originally designed as a co-processor for rendering and graphics, has become a powerful, programmable, many-core processor in the past decade. As the GPU achieves much higher computation power and memory bandwidth than the CPU, GPU accelerations become an effective means to improve the performance of main memory databases. Database systems on GPUs have their root designs on traditional database systems on the CPU, but many GPU-optimized system designs have been introduced, ranging from data layouts, operator design to query processing and query optimizations. Those designs can achieve significant performance improvements over the traditional designs. In this article, we start with introducing the background on GPU as a parallel architecture and the traditional parallel query processing in main memory databases. Next, we present the details of GPU-optimized system designs. We then survey a series of commercial and research systems, and outline the research trends.

We wrote this article as an introductory article in GPU-optimized database systems especially in online analytical

---

Johns Paul, Shengliang Lu and Binsheng He (2021), "Database Systems on GPUs", Foundations and Trends® in Databases: Vol. 11, No. 1, pp 1–108. DOI: 10.1561/1900000076.

processing (OLAP), which can be used as a short text for graduate level or a survey for researchers. We emphasize on the breadth and try to cover as many publications (such as those published in ACM/IEEE) as possible, with necessary details in some key GPU-optimized designs.

1

Introduction

The explosive growth in the amount of data that needs to be analyzed (Data, 2020) and the slow down of CPU performance growth over the last decade due to the breakdown of Moore's law (Michelogiannakis *et al.*, 2016) has pushed the usage of heterogeneous architectures to many data intensive applications. Among those heterogeneous architectures, graphic processing units (GPUs) have been successfully used to improve the performance of in-memory data analytics. While CPU performance growth has slowed down, GPU hardware has achieved over 45x increase in core count and over 23x increase in global memory bandwidth since 2008. Hence, GPUs have evolved as the hardware of choice for high performance database systems, especially for Online Analytical Processing (OLAP).

To demonstrate the attractiveness of GPU hardware for high performance data analytics, we compare the key hardware capabilities of modern CPUs against modern GPUs in Table 1.1. As shown in the table, a typical GPU offers over 6x higher global memory bandwidth and close to 200x more cores than CPUs. This use of higher bandwidth global memory and their massively parallel architecture design is ideal for OLAP systems that require the parallel processing of a larger number of data records. Therefore, we have witnessed significant efforts in the past decade to enable the use of GPUs in high performance data analytics operations (e.g., BreB *et al.*, 2018; Paul *et al.*, 2016; Chrysogelos *et al.*, 2019a; Funke and Teubner, 2020).

Table 1.1: Hardware comparison between GPUs and CPUs.

	Hardware Feature	GPU (NVIDIA Quadro V100)	GPU (AMD Radeon RX 6900 XT)	CPU (Xeon Gold 6230R)
Compute	Core Count	5,120	5,120	26
	Base Clock (GHz)	1.1	1.825	4.0
	Boost Clock (GHz)	1.62	2.25	4.0
Memory	Cache Size (MB)	6	128	35.75
	Memory Type	HBM 2.0	GDDR6	DDR4-2933
	Memory Size (GB)	32	16	1024
	Memory Bandwidth (GB/s)	870	512	140
Other	Hardware Generation	Volta	Radeon 6000 Series	Cascade Lake
	Price[1] (USD)	8,000	2,100	2,500
	TDP (W)	300	300	150

Despite the numerous hardware advantages of GPU hardware, designing and optimizing high performance database systems for GPUs is challenging due to the following reasons. First, due to their significant hardware differences compared to CPUs, database systems need to be re-designed from the ground up to ensure efficient use of GPU hardware, ranging from data layouts, operator design to query processing and query optimizations. Traditional designs are for a few cores, and on the CPU, we are facing thousands of cores. Second, the GPU hardware architecture has been evolving rapidly over the last decade. System designs have to catch up with the hardware evolution. This, combined with the lack of in-depth studies on the impact of different GPU architecture features, makes it difficult to design database systems that take advantage of the right GPU architecture features. Third, the relatively small global memory size of a single GPU often necessitates data access from remote CPUs or GPUs when processing large data sets. This makes it challenging to design efficient scalable GPU database systems due to the significantly higher cost of remote data accesses.

Numerous studies in the past decade have proposed GPU database systems (BreB *et al.*, 2018; Omnisci, 2009; Shanbhag *et al.*, 2020; Heimel *et al.*, 2013) that try to address the above inefficiencies. These studies focus on optimizing the performance of GPU database systems at two key levels: operator level and query level.

Operator level optimizations. Operator level optimizations attempt to improve the performance of individual database operators on single or multiple GPUs. In this regard, significant effort has been put into optimizing compute and memory intensive operators like join (Rui *et al.*, 2015; Kaldewey, n.d.; Sioulas *et al.*, n.d.), aggregation (Lauer *et al.*, 2010; Karnagel *et al.*, n.d.; Tomé *et al.*, n.d.; Cwi and Boncz, n.d.) and sort (Satish *et al.*, n.d.; Davidson *et al.*, 2012; Sintorn and Assarsson, n.d.; Merrill and Grimshaw, 2010). A variant of the join operation, *partitioned hash join*, has especially received a significant amount of attention from the research community (He *et al.*, 2008c; Rui *et al.*, 2015; Sioulas *et al.*, n.d.).

Existing studies on operator level optimizations focus on improving the execution efficiency of GPU database operators through more efficient parallel execution of GPU threads or by taking advantage of newer GPU architecture features. Despite the significant efforts from the research community, existing database operator implementations are often inefficient on GPUs due to their inability to make efficient use of the modern GPU hardware. For example, existing GPU hash join implementations fail to make efficient use of GPU hardware when joining skewed input relations on a single GPU or when joining large input relations on modern multi-GPU architectures.

Query level optimizations. Query level optimizations aim to improve the overall query execution performance on GPUs. Earlier studies focused on improving the overall query execution performance through data layout optimization (Ghodsnia, n.d.; Heimel *et al.*, 2013), better data placement (Gelado *et al.*, 2010; Becchi *et al.*, 2010; Govindaraju *et al.*, 2006a; Kaldewey, n.d.) and data compression (Fang *et al.*, 2010; Przymus and Kaczmarski, 2014b; Przymus and Kaczmarski, 2014a; He *et al.*, 2014; Funasaka *et al.*, 2016). More recent studies have focused on improving the query execution performance by minimizing the cost of moving the intermediate data between the relational operators/GPU kernels. To achieve this, techniques such as efficient pipelining of relational operators (Paul *et al.*, 2016; Funke *et al.*, 2018b), dynamic fusion of GPU kernels (Wu *et al.*, 2012) and just-in-time (JIT) code generation (Breß *et al.*, 2018; Chrysogelos *et al.*, 2019a; Funke and Teubner, 2020; Paul *et al.*, 2020) have been explored.

The rest of this article is organized as follows.

- **Chapter 2: Background.** We present the background and preliminary on GPU hardware architecture. We will first present the architectural evolution of GPUs as a many-core processor. We have witnessed quite some changes of GPU architectures for better performance and programmability for general-purpose computation. Next, we present the programming model and memory model, followed by the general programming and optimization guideline for efficient GPU programs.

- **Chapter 3: CPU Database.** For completeness and comparison purposes, we briefly present the architecture and design of main memory databases on the CPU.

- **Chapter 4: System Design on GPU.** We present the key design components of database systems from data storage layout, operator design and concurrency control. Those components have to be redesigned or revisited due to the architectural difference between the CPU and the GPU.

- **Chapter 5: Query Processing on GPUs.** We start with presenting the evolution of query processing on a single GPU, to adapt to the evolution of GPU architectures. The key issue is on the memory management due to the GPU memory limitations and resource utilization from query processing. Then, we present more recent advances in system design across multiple processors: (a) CPU+GPU, or (b) multiple GPUs. The communication and workload partitioning and scheduling are important issues for those scenarios.

- **Chapter 6: Systems.** We discuss an extensive list of highly influential database systems on GPUs in both academia and industry. The goal is to demonstrate how the technical concepts described in the earlier chapters are integrated by describing real database systems on GPUs.

- **Chapter 7: Future Directions.** We summarize this article and outline the current and future research directions. Database systems on the GPU are still a hot research topic, as new challenges rise from the hardware and new application requirements to database systems. We hope this article can trigger more research on this direction.

References

Abadi, D., S. Madden, and M. Ferreira. (2006a). "Integrating compression and execution in column-oriented database systems". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: ACM Press. 671–682. DOI: 10.1145/1142473.1142548.

Abadi, D., S. Madden, and M. Ferreira. (2006b). "Integrating compression and execution in column-oriented database systems". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 671–682. DOI: 10.1145/1142473.1142548.

Abadi, D. J., D. S. Myers, D. J. DeWitt, and S. R. Madden. (2007). "Materialization strategies in a column-oriented DBMS". In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 466–475.

Ailamaki, A., D. J. DeWitt, M. D. Hill, and M. Skounakis. (2001). "Weaving Relations for Cache Performance". In: *Proceedings of the 27th International Conference on Very Large Data Bases. VLDB '01*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 169–180.

Alcantara, D. A., V. Volkov, S. Sengupta, M. Mitzenmacher, J. D. Owens, and N. Amenta. (2012). "Building an efficient hash table on the GPU". In: *GPU Computing Gems Jade Edition*. Elsevier. 39–53.

AMD. "Welcome to AMD High-Performance Processors and Graphics". URL: https://www.amd.com/en (accessed on 08/19/2020).

Andrzejewski, W. and R. Wrembel. (2010). "GPU-WAH: Applying GPUs to compressing bitmap indexes with word aligned hybrid". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6262 LNCS. No. PART 2. Springer, Berlin, Heidelberg. 315–329. DOI: 10.1007/978-3-642-15251-1_26.

Anzt, H., T. Cojean, C. Yen-Chen, J. Dongarra, G. Flegar, P. Nayak, S. Tomov, Y. M. Tsai, and W. Wang. (2020). "Load-Balancing Sparse Matrix Vector Product Kernels on GPUs". *ACM Trans. Parallel Comput.* 7(1). DOI: 10.1145/3380930.

Apache Spark. "Apache Spark as a Compiler: Joining a Billion Rows per Second on a Laptop - The Databricks Blog". URL: https://databricks.com/blog/2016/05/23/apache-spark-as-a-compiler-joining-a-billion-rows-per-second-on-a-laptop.html (accessed on 08/19/2020).

Appuswamy, R., M. Karpathiotakis, D. Porobic, and A. Ailamaki. (2017). "The case for heterogeneous HTAP". In: *8th Biennial Conference on Innovative Data Systems Research*. No. CONF.

Arefyeva, I., D. Broneske, M. Pinnecke, M. Bhatnagar, and G. Saake. (2017). "Column vs. Row Stores for Data Manipulation in Hardware Oblivious CPU/GPU Database Systems." In: *Grundlagen von Datenbanken*. 24–29.

Babcock, C. (2016). "GPU Hardware Powers MapD Big Data Management". URL: https://www.informationweek.com/infrastructure/pc-and-servers/gpu-hardware-powers-mapd-big-data-management/d/d-id/1327755.

Bakkum, P. and S. Chakradhar. "Efficient Data Management for GPU Databases". *Tech. rep.*

Bakkum, P. and K. Skadron. (2010). "Accelerating SQL database operations on a GPU with CUDA". In: *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*. New York, New York, USA: ACM Press. 94–103. DOI: 10.1145/1735688.1735706.

Balkesen, C., G. Alonso, J. Teubner, and M. T. Özsu. (2013a). "Multi-core, main-memory joins: Sort vs. hash revisited". *Proceedings of the VLDB Endowment*. 7(1): 85–96.

Balkesen, C., J. Teubner, G. Alonso, and M. T. Özsu. (2013b). "Main-memory hash joins on multi-core CPUs: Tuning to the underlying hardware". In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE. 362–373.

Batcher, K. E. (1968). "Sorting networks and their applications". In: Association for Computing Machinery (ACM). 307. DOI: 10.1145/1468075.1468121.

Becchi, M., S. Byna, S. Cadambi, and S. Chakradhar. (2010). "Data-aware scheduling of legacy kernels on heterogeneous platforms with distributed memory". In: *Annual ACM Symposium on Parallelism in Algorithms and Architectures*. New York, New York, USA: ACM Press. 82–91. DOI: 10.1145/1810479.1810498.

Ben-Nun, T., M. Sutton, S. Pai, and K. Pingali. (2017). "Groute: An Asynchronous Multi-GPU Programming Model for Irregular Computations". *dl.acm.orgPaperpile*. DOI: 10.1145/3018743.3018756.

Binnig, C., S. Hildenbrand, and F. Färber. (2009). "Dictionary-based order-preserving string compression for main memory column stores". In: *SIGMOD-PODS'09 - Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems*. 283–295. DOI: 10.1145/1559845.1559877.

Blanas, S., Y. Li, and J. M. Patel. (2011). "Design and evaluation of main memory hash join algorithms for multi-core CPUs". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM. 37–48.

BlazingSQL. (2020). "BlazingSQL". URL: https://blazingsql.com/.

Böhm, C., R. Noll, C. Plant, and B. Wackersreuther. (2009). "Density-based clustering using graphics processors". In: *International Conference on Information and Knowledge Management, Proceedings*. 661–670. DOI: 10.1145/1645953.1646038.

Boncz, P., T. Grust, M. Van Keulen, S. Manegold, J. Rittinger, and J. Teubner. (2006). "MonetDB/XQuery: A fast XQuery processor powered by a relational engine". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 479–490. DOI: 10.1145/1142473.1142527.

Boncz, P. A., P. Boncz, M. Zukowski, and N. Nes. (2005). "MonetDB/X100: Hyper-Pipelining Query Execution Software Testpilot View project Using GPUs in analytic DBMSes View project MonetDB/X100: Hyper-Pipelining Query Execution". *Tech. rep.* URL: https://www.researchgate.net/publication/45338800.

Boncz, P. A., S. Manegold, M. L. Kersten, *et al.* (1999). "Database architecture optimized for the new bottleneck: Memory access". In: *VLDB*. Vol. 99. 54–65.

Boncz, P. A., M. L. Kersten, and S. Manegold. (2008). "Breaking the memory wall in MonetDB". *Communications of the ACM*. 51(12): 77–85. DOI: 10.1145/1409360.1409380.

BreB, S., B. Kocher, H. Funke, S. Zeuch, T. Rabl, and V. Markl. (2018). "Generating Custom Code for Efficient Query Execution on Heterogeneous Processors". *The VLDB Journal*. 27(6): 797–822. DOI: 10.1007/s00778-018-0512-y.

Breß, S., F. Beier, H. Rauhe, K. Sattler, E. S. .-. I. Systems, and undefined 2013. "Efficient co-processor utilization in database query processing". *ElsevierPaperpile*. URL: https://www.sciencedirect.com/science/article/pii/S0306437913000732.

Breß, S. (2014). "The Design and Implementation of CoGaDB: A Column-oriented GPU-accelerated DBMS". *Datenbank-Spektrum*. 14(3): 199–209. DOI: 10.1007/s13222-014-0164-z.

Breß, S., H. Funke, and J. Teubner. (2016). "Robust Query Processing in Co-Processor-accelerated Databases". *dl.acm.orgPaperpile*. 26-June-20(June): 1891–1906. DOI: 10.1145/2882903.2882936.

Breß, S., I. Geist, E. Schallehn, M. Mory, and G. Saake. (2012). "A framework for cost based optimization of hybrid CPU/GPU query plans in database systems * †". *Tech. rep.* No. 4. URL: https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-f650e8a0-6c37-486f-bfb1-929a5a74c2b9.

Breß, S. and G. Saake. (2013). “Why it is time for a HyPE: A hybrid query processing engine for efficient GPU coprocessing in DBMS”. *Proceedings of the VLDB Endowment*. 6(12): 1398–1403.

Brytlyt. (2020). “Brytlyt: GPU Accelerated Database for Speed of Thought Analytics”. URL: https://www.brytlyt.com/.

Cascarano, N., P. Rolando, F. Risso, and R. Sisto. (2010). “INFAnt: NFA pattern matching on GPGPU devices”. DOI: 10.1145/1880153.1880157.

Cederman, D. and P. Tsigas. (2009). “GPU-Quicksort: A practical quicksort algorithm for graphics processors”. *Journal of Experimental Algorithmics*. 14(4). URL: http://doi.acm.org/10.1145/1498698.1564500.

Chen, J. and X. Ran. (2019). “Deep Learning With Edge Computing: A Review”. *Proceedings of the IEEE*. DOI: 10.1109/JPROC.2019.2921977.

Chen, S., A. Ailamaki, P. B. Gibbons, and T. C. Mowry. (2007). “Improving hash join performance through prefetching”. *ACM Transactions on Database Systems (TODS)*. 32(3): 17.

Chhugani, J., A. D. Nguyen, V. W. Lee, W. Macy, M. Hagog, Y. K. Chen, A. Baransi, S. Kumar, and P. Dubey. (2008). “Efficient implementation of sorting on multi-core SIMD CPU architecture”. *Proceedings of the VLDB Endowment*. 1(2): 1313–1324. DOI: 10.14778/1454159.1454171.

Chrysogelos, P., M. Karpathiotakis, R. Appuswamy, and A. Ailamaki. (2019a). “HetExchange: Encapsulating heterogeneous CPU-GPU parallelism in JIT compiled engines”. *Proceedings of the VLDB Endowment*: 13. URL: http://infoscience.epfl.ch/record/262531.

Chrysogelos, P., P. Sioulas, and A. Ailamaki. (2019b). “Hardware-conscious Query Processing in GPU-accelerated Analytical Engines”. *Proceesings of the 9th Biennial Conference on Innovative Data Systems Research*: 9. URL: http://infoscience.epfl.ch/record/262529.

Cohen, J. and M. J. Molemaker. (2009). “A fast double precision CFD code using CUDA”. *Parallel Computational Fluid Dynamics: Recent Advances and Future Directions*: 414–429.

Crotty, A., A. Galakatos, K. Dursun, T. Kraska, U. Çetintemel, and S. B. Zdonik. (2015). "Tupleware:" Big" Data, Big Analytics, Small Clusters." In: *CIDR*.

CUDA. "CUDA Toolkit 6.5: NVIDIA Developer". URL: https://developer.nvidia.com/cuda-toolkit-65 (accessed on 08/21/2020).

Cui, B., B. C. Ooi, J. Su, and K. L. Tan. (2004). "Main memory indexing: The case for BD-tree". *IEEE Transactions on Knowledge and Data Engineering*. 16(7): 870–874. DOI: 10.1109/TKDE.2004.1318568.

Cui, B., B. C. Ooi, J. Su, and K.-L. Tan. (2003). "Contorting high dimensional data for efficient main memory KNN processing". In: *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD '03*. New York, New York, USA: ACM Press. 479. DOI: 10.1145/872757.872815.

Curino, C., E. Jones, Y. Zhang, and S. Madden. (2150). "Schism: a Workload-Driven Approach to Database Replication and Partitioning Publisher Very Large Data Base Endowment Inc. (VLDB Endowment) Terms of Use Creative Commons Attribution-Noncommercial-Share Alike 3.0 Schism: a Workload-Driven Approach to Database Replication and Partitioning". *Tech. rep.* URL: http://dl.acm.org/citation.cfm?id=1920853http://hdl.handle.net/1721.1/73347http://creativecommons.org/licenses/by-nc-sa/3.0/.

Cwi, T. G. and P. Boncz. "Exploring Query Execution Strategies for JIT, Vectorization and SIMD". *Tech. rep.* URL: https://t1mm3.github.io/assets/papers/adms17.pdf.

Dai, H., Z. Lin, C. Li, C. Zhao, F. Wang, N. Zheng, and H. Zhou. (2018). "Accelerate GPU Concurrent Kernel Execution by Mitigating Memory Pipeline Stalls". In: *Proceedings - International Symposium on High-Performance Computer Architecture*. Vol. 2018-February. IEEE Computer Society. 208–220. DOI: 10.1109/HPCA.2018.00027.

Data, B. "Big Data Market by Component (Solutions and Services), Function (Finance, Human resources, Marketing and Sales, Operations, and Others), Industry Vertical (IT & Telecom, BFSI, Retail, Healthcare, Government and Defense, Manufacturing, and Others), and Reg". URL: https://www.adroitmarketresearch.com/contacts/request-sample/1437 (accessed on 09/08/2020).

"Database Cloud: Oracle". URL: https://www.oracle.com/database/ (accessed on 08/19/2020).

Dathathri, R., G. Gill, L. Hoang, H.-V. Dang, A. Brooks, N. Dryden, M. Snir, and K. Pingali. (2018). "Gluon: A Communication-Optimizing Substrate for Distributed Heterogeneous Graph Analytics". *dl.acm.orgPaperpile*. 53(4): 752–768. DOI: 10.1145/3192366.3192404.

Davidson, A., D. Tarjan, M. Garland, and J. D. Owens. (2012). "Efficient parallel merge sort for fixed and variable length keys". In: *2012 Innovative Parallel Computing, InPar 2012*. DOI: 10.1109/InPar.2012.6339592.

"Db2 Database - Overview - Singapore : IBM". URL: https://www.ibm.com/sg-en/products/db2-database (accessed on 08/19/2020).

DGX. "DGX White Paper: NVIDIA". URL: https://www.nvidia.com/en-us/data-center/resources/dgx-1-system-architecture-whitepaper/ (accessed on 09/07/2020).

DGX-1. "NVIDIA DGX-1". https://www.nvidia.com/en-us/data-center/dgx-1/.

DGX-2. "NVIDIA DGX-2". https://www.nvidia.com/en-us/data-center/dgx-2/.

Djenouri, Y., D. Djenouri, A. Belhadi, and A. Cano. (2019). "Exploiting GPU and cluster parallelism in single scan frequent itemset mining". *Information Sciences*. 496: 363–377. DOI: https://doi.org/10.1016/j.ins.2018.07.020.

Dotzler, G., R. Veldema, and M. Klemm. (2010). *IWMSE: JCudaMP: OpenMP/Java on CUDA*.

Faerber, F., A. Kemper, P.-Å. Larson, J. Levandoski, T. Neumann, and A. Pavlo. (2017). "Main Memory Database Systems". *Foundations and Trends® in Databases*. 8(1-2): 1–130. DOI: 10.1561/1900000058.

Fan, Z., F. Qiu, A. Kaufman, and S. Yoakum-Stover. (2004). "GPU cluster for high performance computing". In: *Proceedings of the ACM/IEEE SC 2004 Conference: Bridging Communities*. Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/SC.2004.26.

Fang, R., B. He, M. Lu, K. Yang, N. K. Govindaraju, Q. Luo, and P. V. Sander. (2007). "GPUQP: Query Co-processing Using Graphics Processors". In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. SIGMOD '07.* New York, NY, USA: ACM. 1061–1063. DOI: 10.1145/1247480.1247606.

Fang, W., B. He, and Q. Luo. (2010). "Database compression on graphics processors". *Proceedings of the VLDB Endowment.* 3(1): 670–680. DOI: 10.14778/1920841.1920927.

FastData. (2020). "PlasmaENGINE". URL: https://fastdata.io/plasma-engine/.

Feng, Z. and E. Lo. (2015). "Accelerating aggregation using intra-cycle parallelism". In: *Proceedings - International Conference on Data Engineering.* Vol. 2015-May. IEEE Computer Society. 291–302. DOI: 10.1109/ICDE.2015.7113292.

Funasaka, S., K. Nakano, and Y. Ito. (2016). "Light loss-less data compression, with GPU implementation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 10048 LNCS. Springer Verlag. 281–294. DOI: 10.1007/978-3-319-49583-5_22.

Funke, H., S. Breß, S. Noll, V. Markl, and J. Teubner. (2018a). "Pipelined Query Processing in Coprocessor Environments". In: *Proceedings of the 2018 International Conference on Management of Data. SIGMOD '18.* New York, NY, USA: ACM. 1603–1618. DOI: 10.1145/3183713.3183734.

Funke, H., S. Breß, S. Noll, V. Markl, and J. Teubner. (2018b). "Pipelined query processing in coprocessor environments". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data.* New York, New York, USA: Association for Computing Machinery. 1603–1618. DOI: 10.1145/3183713.3183734.

Funke, H. and J. Teubner. (2020). "Data-Parallel Query Processing on Non-Uniform Data". *Proc. VLDB Endow.* 13(6): 884–897. DOI: 10.14778/3380750.3380758.

Garcia, P. and H. F. Korth. (2007). "Pipelined hash-join on multithreaded architectures". In: *Proceedings of the 3rd International Workshop on Data Management on New Hardware, DaMoN '07.* New York, New York, USA: ACM Press. 1. DOI: 10.1145/1363189.1363191.

Gelado, I., J. Cabezas, N. Navarro, J. E. Stone, S. Patel, and W. M. W. Hwu. (2010). "An asymmetric dstributed shared memory model for heterogeneous parallel systems". *ACM SIGPLAN Notices*. 45(3): 347–358. DOI: 10.1145/1735971.1736059.

Ghodsnia, P. "An in-GPU-memory column-oriented database for processing analytical workloads".

Giannikis, G., G. Alonso, and D. Kossmann. (2012). "Shareddb: Killing one thousand queries with one stone". *Proceedings of the VLDB Endowment*. 5(6): 526–537. DOI: 10.14778/2168651.2168654.

Goldstein, J., R. Ramakrishnan, and U. Shaft. (1998). "Compressing relations and indexes". In: *Proceedings 14th International Conference on Data Engineering*. IEEE. 370–379.

Gosink, L. J., K. Wu, E. W. Bethel, J. D. Owens, and K. I. Joy. (2009). "Data parallel bin-based indexing for answering queries on multi-core architectures". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5566 LNCS. 110–129. DOI: 10.1007/978-3-642-02279-1_9.

Govindaraju, N., J. Gray, R. Kumar, and D. Manocha. (2006a). "GPUTeraSort: High performance graphics co-processor sorting for large database management". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 325–336. DOI: 10.1145/1142473.1142511.

Govindaraju, N., J. Gray, R. Kumar, and D. Manocha. (2006b). "GPUTeraSort: High performance graphics co-processor sorting for large database management". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 325–336. DOI: 10.1145/1142473.1142511.

Govindaraju, N. K., B. Lloyd, W. Wang, M. Lin, and D. Manocha. (2005). "Fast computation of database operations using graphics processors". In: *ACM SIGGRAPH 2005 Courses, SIGGRAPH 2005*. New York, New York, USA: Association for Computing Machinery, Inc. 206. DOI: 10.1145/1198555.1198787.

GPUDirect. "GPUDirect: NVIDIA Developer". URL: https://developer.nvidia.com/gpudirect (accessed on 08/20/2020).

Graefe, G. "Sorting And Indexing With Partitioned B-Trees". *Tech. rep.* URL: http://www-db.cs.wisc.edu/cidr/cidr2003/program/p1.pdf.

Gregg, C. and K. Hazelwood. (2011). "Where is the data? Why you cannot debate CPU vs. GPU performance without the answer". In: *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE. 134–144.

Guo, C., H. Chen, F. Zhang, and C. Li. (2019). "Distributed join algorithms on multi-CPU clusters with GPUDirect RDMA". In: *ACM International Conference Proceeding Series*. Association for Computing Machinery. DOI: 10.1145/3337821.3337862.

Gupta, V., A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan. (2009). *GViM: GPU-accelerated Virtual Machines*.

Han, J., Z. Xie, and Y. Fu. "Join Index Hierarchy: An Indexing Structure for EEcient Navigation in Object-Oriented Databases". *Tech. rep.* URL: https://ieeexplore.ieee.org/abstract/document/761666/.

Harizopoulos, S., V. Liang, S. Madden, and D. J. Abadi. (2006). "Performance Tradeoffs in Read-Optimized Databases". *Tech. rep.* URL: http://nms.csail.mit.edu/%7B~%7Dstavros/pubs/vldb2006.pdf.

He, B., W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. (2008a). "Mars: A MapReduce Framework on graphics processors". In: *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 260–269.

He, B., W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. (2008b). "Mars: A MapReduce Framework on Graphics Processors". In: *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. PACT '08*. Toronto, Ontario, Canada: Association for Computing Machinery. 260–269. DOI: 10.1145/1454115.1454152.

He, B., M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. (2009). "Relational Query Coprocessing on Graphics Processors". *ACM Trans. Database Syst.* 34(4): 21:1–21:39. DOI: 10.1145/1620585.1620588.

He, B. and Q. Luo. (2008). "Cache-oblivious databases: Limitations and opportunities". *ACM Transactions on Database Systems (TODS)*. 33(2): 8.

He, B., K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander. (2008c). "Relational joins on graphics processors". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM. 511–524.

He, B. and J. X. Yu. (2011a). "High-Throughput Transaction Executions on Graphics Processors". *Proceedings of the VLDB Endowment*. 4(5).

He, B. and J. X. Yu. (2011b). "High-throughput transaction executions on graphics processors". *Proceedings of the VLDB Endowment*. 4(5): 314–325. DOI: 10.14778/1952376.1952381.

He, J., M. Lu, and B. He. (2013). "Revisiting co-processing for hash joins on the coupled CPU-GPU architecture". *Proceedings of the VLDB Endowment*. 6(10): 889–900.

He, J., S. Zhang, and B. He. (2014). "In-cache query co-processing on coupled CPU-GPU architectures". *Proceedings of the VLDB Endowment*. 8(4): 329–340.

Heimel, M., M. Saecker, H. Pirk, S. Manegold, and V.-h. Markl. (2013). "Hardware-oblivious parallelism for in-memory column-stores". *Proceedings of the VLDB Endowment*. 6(9): 709–720. DOI: 10.14778/2536360.2536370.

Huang, B., J. Gao, and X. L.X. (2009). "An empirically optimized radix sort for GPU". In: *Proceedings - 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2009*. 234–241. DOI: 10.1109/ISPA.2009.89.

Idreos, S., F. Groffen, N. Nes, S. Manegold, S. Mullender, and M. Kersten. (2012). "Monetdb: Two decades of research in column-oriented database". *IEEE Data Engineering Bulletin*.

Idreos, S., F. Groffen, N. Nes, S. Manegold, S. Mullender, and M. Kersten. (2016). "MonetDB: Two Decades of Research in Column-oriented Database Architectures". *Tech. rep.* URL: http://hdl.handle.net/11245/2.128089http://hdl.handle.net/11245/1.380005.

Ivanova, M. G., M. L. Kersten, N. J. Nes, and R. A. P. Gonçalves. (2010). "An architecture for recycling intermediates in a column-store". *ACM Trans. Datab. Syst. 35, 4, Article*. 24(4). DOI: 10.1145/1862919.1862921.

Jia, Z., Y. Kwon, G. Shipman LANL, P. M. Cormick LANL pat, lanl-gov Mattan Erez, A. Aiken, G. Shipman, P. M. Cormick, and M. Erez. (2017). "A Distributed Multi-GPU System for Fast Graph Processing". *dl.acm.orgPaperpile*. 11(3): 2977–310. DOI: 10.14778/3157794.3157799.

Kaldewey, T., J. Hagen, A. D. Blas, E. S. .-. F. U. W. On, and undefined 2009. "Parallel search on video cards". *usenix.orgPaperpile*. URL: https://www.usenix.org/event/hotpar09/tech/full%7B%5C_%7Dpapers/kaldeway/kaldeway%7B%5C_%7Dhtml.

Kaldewey, T. "GPU Join Processing Revisited".

Karnagel, T., R. Mueller, and G. M. Lohman. "Optimizing GPU-accelerated Group-By and Aggregation". *Tech. rep.* URL: http://www.adms-conf.org/2015/gpu-optimizer-camera-ready.pdf.

Karpathiotakis, M., I. Alagiannis, and A. Ailamaki. (2016). "Fast Queries over Heterogeneous Data Through Engine Customization". *Proc. VLDB Endow.* 9(12): 972–983. DOI: 10.14778/2994509.2994516.

Karpathiotakis, M., I. Alagiannis, T. Heinis, M. Branco, and A. Ailamaki. (2015). "Just-in-time data virtualization: Lightweight data management with ViDa". In: *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR)*. No. CONF.

Kato, S., K. Lakshmanan, R. Rajkumar, and Y. Ishikawa. (2011). "TimeGraph: GPU scheduling for real-time multi-tasking environments". In: *Proc. USENIX ATC*. 17–30.

Kaufmann, M. (2013). "Storing and processing temporal data in a main memory column store". *Proceedings of the VLDB Endowment*. 6(12): 1444–1449. DOI: 10.14778/2536274.2536333.

Kemper, A. and T. Neumann. (2011). "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots". In: *2011 IEEE 27th International Conference on Data Engineering*. IEEE. 195–206.

Kersten, T., V. Leis, A. Kemper, T. Neumann, A. Pavlo, and P. Boncz. (2018). "Everything you always wanted to know about compiled and vectorized queries but were afraid to ask". *Proceedings of the VLDB Endowment*. 11(13): 2209–2222. DOI: 10.14778/3275366.3284966.

Khorasani, F., K. Vora, R. Gupta, and L. N. Bhuyan. (2014). "CuSha: Vertex-Centric Graph Processing on GPUs". In: *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing. HPDC '14*. New York, NY, USA: Association for Computing Machinery. 239–252. DOI: 10.1145/2600212.2600227.

Kim, C., J. Chhugani, N. Satish, E. Sedlar, A. D. Nguyen, T. Kaldewey, V. W. Lee, S. A. Brandt, and P. Dubey. (2010). "FAST: Fast architecture sensitive tree search on modern CPUs and GPUs". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 339–350. DOI: 10.1145/1807167.1807206.

Kim, C., J. Chhugani, N. Satish, E. Sedlar, A. D. Nguyen, T. Kaldewey, V. W. Lee, S. A. Brandt, and P. Dubey. (2011). "Designing fast architecture-sensitive tree search on modern multicore/many-core processors". In: *ACM Transactions on Database Systems*. Vol. 36. No. 4. 1–34. DOI: 10.1145/2043652.2043655.

Kim, C., T. Kaldewey, V. W. Lee, E. Sedlar, A. D. Nguyen, N. Satish, J. Chhugani, A. Di Blas, and P. Dubey. (2009). "Sort vs. Hash revisited: fast join implementation on modern multi-core CPUs". *Proceedings of the VLDB Endowment*. 2(2): 1378–1389.

Kinetica. (2020). "The Kinetica Streaming Data warehouse". URL: https://www.kinetica.com/.

Kipfer, P., M. Segal, and R. Westermann. (2004). "UberFlow: A GPU-Based Particle Engine". *Tech. rep.* URL: https://dl.acm.org/doi/abs/10.1145/1058129.1058146?casa%7B%5C_%7Dtoken=zTU0Y3f9cBkAAAAA:jkhAdBijIq268cwTirtoKs0WfTOtJpUKb1g9TnR-ynYNSRn9yjMTUU1uO2WHJzeVJcQ1jwKKGb03GQ.

Al-Kiswany, S., A. Gharaibeh, E. Santos-Neto, G. Yuan, and M. Ripeanu. (2008). "StoreGPU: Exploiting graphics processing units to accelerate distributed storage systems". In: *Proceedings of the 17th International Symposium on High Performance Distributed Computing 2008, HPDC'08*. New York, New York, USA: ACM Press. 165–174. DOI: 10.1145/1383422.1383443.

Klonatos, Y., C. Koch, T. Rompf, and H. Chafi. (2014). "Building Efficient Query Engines in a High-level Language". *Proc. VLDB Endow.* 7(10): 853–864. DOI: 10.14778/2732951.2732959.

Koch, C., Y. Ahmad, O. Kennedy, M. Nikolic, A. Nötzli, D. Lupei, and A. Shaikhha. (2014). "DBToaster: higher-order delta processing for dynamic, frequently fresh views". *The VLDB Journal*. 23(2): 253–278. DOI: 10.1007/s00778-013-0348-4.

Kohn, A., V. Leis, and T. Neumann. (2019). "Making Compiling Query Engines Practical". *IEEE Transactions on Knowledge and Data Engineering*: 1. DOI: 10.1109/TKDE.2019.2905235.

Krikellas, K., S. D. Viglas, and M. Cintra. (2010). "Generating code for holistic query evaluation". In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE. 613–624.

Lang, H., T. Mühlbauer, F. Funke, P. A. Boncz, T. Neumann, and A. Kemper. (2016). "Data Blocks: Hybrid OLTP and OLAP on Compressed Storage Using Both Vectorization and Compilation". In: *Proceedings of the 2016 International Conference on Management of Data. SIGMOD '16*. New York, NY, USA: ACM. 311–326. DOI: 10.1145/2882903.2882925.

Larson, P., E. Hanson, S. P. .-. I. D. E. Bull., and undefined 2012. "Columnar Storage in SQL Server 2012." *academia.eduPaperpile*. URL: http://www.academia.edu/download/48908164/DE2012Q1.pdf%7B%5C#%7Dpage=17.

Larson, P. Å., C. Clinciu, E. N. Hanson, A. Oks, S. L. Price, S. Rangarajan, A. Surna, and Q. Zhou. (2011). "SQL server column store indexes". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1177–1184. DOI: 10.1145/1989323.1989448.

Lauer, T., A. Datta, Z. Khadikov, and C. Anselm. (2010). "Exploring graphics processing units as parallel coprocessors for online aggregation". In: *International Conference on Information and Knowledge Management, Proceedings*. New York, New York, USA: ACM Press. 77–84. DOI: 10.1145/1871940.1871958.

Lee, R., T. Luo, Y. Huai, F. Wang, Y. He, and X. Zhang. (2011). "Ysmart: Yet another sql-to-mapreduce translator". In: *2011 31st International Conference on Distributed Computing Systems*. IEEE. 25–36.

Lehman, T. J. and M. J. Carey. "A Study of Index Structures for Main Memory Database Management Systems". In: *Proceedings of the 12th International Conference on Very Large Data Bases*. URL: https://dl.acm.org/doi/10.5555/645913.671312.

Leis, V., P. Boncz, A. Kemper, and T. Neumann. (2014). "Morsel-driven parallelism: A NUMA-aware query evaluation framework for the many-core age". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: Association for Computing Machinery. 743–754. DOI: 10.1145/2588555.2610507.

Leis, V., A. Kemper, and T. Neumann. (2013). "The adaptive radix tree: ARTful indexing for main-memory databases". In: *Proceedings - International Conference on Data Engineering*. 38–49. DOI: 10.1109/ICDE.2013.6544812.

Leischner, N., V. Osipov, and P. Sanders. (2010). "GPU sample sort". In: *Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010*. DOI: 10.1109/IPDPS.2010.5470444.

Lemke, C., K. U. Sattler, F. Faerber, and A. Zeier. (2010). "Speeding up queries in column stores: A case for compression". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6263 LNCS. Springer, Berlin, Heidelberg. 117–129. DOI: 10.1007/978-3-642-15105-7_10.

Levandoski, J. J., D. B. Lomet, and S. Sengupta. (2013). "The Bw-tree: A B-tree for new hardware platforms". In: *Proceedings - International Conference on Data Engineering*. 302–313. DOI: 10.1109/ICDE.2013.6544834.

Li, A., S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker. (2019). "Evaluating modern GPU interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect". *IEEE Transactions on Parallel and Distributed Systems*. 31(1): 94–110.

Li, A., S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker. (2020). "Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect". *IEEE Transactions on Parallel and Distributed Systems*. 31(1): 94–110. DOI: 10.1109/TPDS.2019.2928289. arXiv: 1903.04611.

Li, A., S. L. Song, J. Chen, X. Liu, N. Tallent, and K. Barker. (2018). "Tartan: evaluating modern GPU interconnect via a multi-GPU benchmark suite". In: *2018 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE. 191–202.

Li, J., H. W. Tseng, C. Lin, Y. Papakonstantinou, and S. Swanson. (2015). "Hippogriff DB: Balancing I/O and GPU bandwidth in big data analytics". In: *Proceedings of the VLDB Endowment*. Vol. 9. No. 14. Association for Computing Machinery. 1647–1658. DOI: 10.14778/3007328.3007331.

Li, T., E. El-Araby, V. K. Narayana, and T. El-Ghazawi. (2011). "Evolutionary Design Space Exploration View project Hybrid Photonic-Plasmonic Non-blocking Broadband 5x5 Router for Optical Network View project GPU Resource Sharing and Virtualization on High Performance Computing Systems". *ieeexplore.ieee.orgPaperpile*. DOI: 10.1109/ICPP.2011.88.

Li, Y. and J. M. Patel. (2013). "BitWeaving: Fast scans for main memory data processing". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: ACM Press. 289–300. DOI: 10.1145/2463676.2465322.

Lin, C. H., S. Y. Tsai, C. H. Liu, S. C. Chang, and J. M. Shyu. (2010). "Accelerating string matching using multi-threaded algorithm on GPU". In: *GLOBECOM - IEEE Global Telecommunications Conference*. DOI: 10.1109/GLOCOM.2010.5683320.

Lutz, C., S. Breß, S. Zeuch, T. Rabl, and V. Markl. (2020). "Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery. 1633–1649. DOI: 10.1145/3318464.3389705.

Lynch, C. A. and E. B. Brownrigg. (1981). "Application of data compression to a large bibliographic data base".

Manca, E., A. Manconi, A. Orro, G. Armano, and L. Milanesi. (2016). "CUDA-quicksort: An improved GPU-based implementation of quicksort". *Concurrency Computation*. 28(1): 21–43. DOI: 10.1002/cpe.3611.

Manegold Peter Boncz Martin Kersten, S. L. (2002). "Generic Database Cost Models for Hierarchical Memory Systems". *Tech. rep.* URL: http://monetdb.cwi.nl.

Memcached. "memcached - a distributed memory object caching system". URL: https://memcached.org/ (accessed on 08/19/2020).

Menon, P., T. C. Mowry, and A. Pavlo. (2017). "Relaxed operator fusion for in-memory databases". *Proceedings of the VLDB Endowment*. 11(1): 1–13. DOI: 10.14778/3151113.3151114.

Merrill, D. G. and A. S. Grimshaw. (2010). "Revisiting sorting for GPGPU stream architectures". In: *Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT*. Vol. 2010. New York, New York, USA: Institute of Electrical and Electronics Engineers Inc. 545–546. DOI: 10.1145/1854273.1854344.

Michelogiannakis, G., J. Shalf, D. Donofrio, and J. Bachan. (2016). "Continuing the Scaling of Digital Computing Post Moore's Law".

Milvus. (2020). "Milvus". URL: https://milvus.io/.

Mittal, S. and J. S. Vetter. (2015). "A survey of CPU-GPU heterogeneous computing techniques". *ACM Computing Surveys*. 47(4). DOI: 10.1145/2788396.

Mühlbauer, T., W. Rödiger, R. Seilbeck, A. Reiser, A. Kemper, and T. Neumann. (2013). "Instant loading for main memory databases". *Proceedings of the VLDB Endowment*. 6(14): 1702–1713. DOI: 10.14778/2556549.2556555.

Müller, I., C. Ratsch, and F. Faerber. (2014). "Adaptive String Dictionary Compression in In-Memory Column-Store Database Systems". *researchgate.netPaperpile*. DOI: 10.5441/002/edbt.2014.27.

MySQL, A. B. (2007). "The world's most popular open source database". *MySQL AB*. URL: https://ci.nii.ac.jp/naid/10016599862.

Navarro, G. (2001). "A guided tour to approximate string matching". *ACM Computing Surveys*. 33(1): 31–88. DOI: 10.1145/375360.375365.

NCCL. "NVIDIA Collective Communication Library (NCCL) Documentation — NCCL 2.7.1 documentation". URL: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/index.html (accessed on 09/24/2020).

Neumann, T. (2011). "Efficiently Compiling Efficient Query Plans for Modern Hardware". *Proc. VLDB Endow.* 4(9): 539–550. DOI: 10.14778/2002938.2002940.

Neumann, T. and V. Leis. (2014). "Compiling Database Queries into Machine Code." *IEEE Data Eng. Bull.*

News, S. R. .-. I. and U. 2008Paperpile. "RAM is the new disk".

Nouri, Z. and Y. C. Tu. (2018). "GPU-based parallel indexing for concurrent spatial query processing". In: *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery. 1–12. DOI: 10.1145/3221269.3221296.

NVIDIA. "EGX A100 : Real Time AI Processing at the Edge: NVIDIA". URL: https://www.nvidia.com/en-sg/data-center/products/egx-a100/ (accessed on 09/09/2020).

NVIDIA. "Modern Data Centers to Accelerate HPC Workloads NVIDIA". URL: https://www.nvidia.com/en-us/data-center/ (accessed on 08/19/2020).

NVIDIA. "NVLink and NVSwitch: Advanced Multi-GPU Systems: NVIDIA". URL: https://www.nvidia.com/en-sg/data-center/nvlink/ (accessed on 09/09/2020).

NVIDIA CPU. "NVIDIA BlueField data processing unit (DPU)". URL: https://www.nvidia.com/en-sg/networking/products/data-processing-unit/.

NVlink. "NVLink & NVSwitch:Advanced Multi-GPU Systems: NVIDIA". URL: https://www.nvidia.com/en-sg/data-center/nvlink/ (accessed on 08/20/2020).

OmniSci. (2019). "https://www.omnisci.com/". In:

Omnisci. (2009). "Omnisci. Accelerated Analytics Platform". URL: https://www.omnisci.com/.

Ousterhout, J., P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman. (2010). "The case for RAMClouds". *ACM SIGOPS Operating Systems Review*. 43(4): 92–105. DOI: 10.1145/1713254.1713276.

Owens, J. D., M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. (2008). "GPU computing". *Proceedings of the IEEE*. 96(5): 879–899. DOI: 10.1109/JPROC.2008.917757.

Palkar, S., J. J. Thomas, A. Shanbhag, D. Narayanan, H. Pirk, M. Schwarzkopf, S. Amarasinghe, M. Zaharia, and S. InfoLab. (2017). "Weld: A common runtime for high performance data analytics". In: *Conference on Innovative Data Systems Research (CIDR)*.

Patel, J. M., H. Deshmukh, J. Zhu, N. Potti, Z. Zhang, M. Spehlmann, H. Memisoglu, and S. Saurabh. (2018). "Quickstep: A data platform based on the scaling-up approach". In: *Proceedings of the VLDB Endowment*. Vol. 11. No. 6. VLDB Endowment. 663–676. DOI: 10.14778/3184470.3184471.

Paul, J., B. He, and C. T. Lau. "Query Processing on OpenCL-based FPGAs: Challenges and Opportunities". *Tech. rep.*

Paul, J., B. He, S. Lu, and C. T. Lau. (2019). "Revisiting Hash Join on Graphics Processors: A Decade Later". In: *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE. 294–299.

Paul, J., B. He, S. Lu, and C. T. Lau. (2020). "Improving execution efficiency of just-in-time compilation based query processing on GPUs". In: vol. 14. No. 2. VLDB Endowment. 202–214.

Paul, J., J. He, and B. He. (2016). "GPL: A GPU-based Pipelined Query Processing Engine". *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*: 1935–1950. DOI: 10.1145/2882903.2915224.

Paul, J., S. Lu, B. He, and C. T. Lau. (2021). "MG-Join: A Scalable Join for Massively Parallel Multi-GPU Architectures". *Proceedings of SIGMOD*.

PCIe. "PCIe 4.0: Benchmark for Component Performance: AMD Partner Hub". URL: https://www.amd.com/en/partner/pcie-benchmark-for-component-performance (accessed on 09/09/2020).

Peters, H., O. Schulz-Hildebrandt, and N. Luttenberger. (2010). "Fast in-place sorting with CUDA based on bitonic sort". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6067 LNCS. No. PART 1. Springer, Berlin, Heidelberg. 403–410. DOI: 10.1007/978-3-642-14390-8_42.

Pinnecke, M., D. Broneske, G. C. Durand, and G. Saake. (2017). "Are Databases Fit for Hybrid Workloads on GPUs? A Storage Engine's Perspective". In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 1599–1606. DOI: 10.1109/ICDE.2017.237.

Pirk, H., S. Manegold, and M. Kersten. (2011). "Accelerating Foreign-Key Joins Using Asymmetric Memory Channels". In: *ADMS*.

Pirk, H., O. Moll, M. Zaharia, and S. Madden. (2016). "Voodoo-a vector algebra for portable database performance on modern hardware". *Proceedings of the VLDB Endowment*. 9(14): 1707–1718.

Plattner, H. (2009). "A common database approach for OLTP and OLAP using an in-memory column database". In: *SIGMOD-PODS'09 - Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems*. New York, New York, USA: Association for Computing Machinery (ACM). 1–2. DOI: 10.1145/1559845.1559846.

Polychroniou, O., A. Raghavan, and K. A. Rossy. (2015). "Rethinking SIMD vectorization for in-memory databases". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Vol. 2015-May. New York, New York, USA: Association for Computing Machinery. 1493–1508. DOI: 10.1145/2723372.2747645.

PostgreSQL. "PostgreSQL: The world's most advanced open source database". URL: https://www.postgresql.org/ (accessed on 08/19/2020).

POWER8. "POWER8 with NVIDIA NVLink Technology". URL: https://www-355.ibm.com/systems/power/openpower/tgcmDocumentRepository.xhtml?aliasId=POWER8%7B%5C_%7Dwith%7B%5C_%7DNVIDIA%7B%5C_%7DNVLink (accessed on 09/07/2020).

Press, G. A. .-. t. M. and undefined 1986. "A model of concurrent computation in distributed systems". *ci.nii.ac.jpPaperpile*. URL: https://ci.nii.ac.jp/naid/10000004223/.

Przymus, P. and K. Kaczmarski. (2014a). "Compression Planner for Time Series Database with GPU Support". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8920. Springer Verlag. 36–63. DOI: 10.1007/978-3-662-45761-0_2.

Przymus, P. and K. Kaczmarski. (2014b). "Dynamic compression strategy for time series database using GPU". In: *Advances in Intelligent Systems and Computing*. Vol. 241. Springer Verlag. 235–244. DOI: 10.1007/978-3-319-01863-8_26.

Purcell, T. J., C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. (2005). "Photon mapping on programmable graphics hardware". In: *ACM SIGGRAPH 2005 Courses, SIGGRAPH 2005*. Association for Computing Machinery, Inc. DOI: 10.1145/1198555.1198797.

QikkDB. (2020). "QikkDB". URL: https://github.com/qikkDB/qikkdb.

Raman, V., G. Attaluri, R. Barber, N. Chainani, D. Kalmuk, V. KulandaiSamy, J. Leenstra, S. Lightstone, S. Liu, G. M. Lohman, T. Malkemus, R. Mueller, I. Pandis, B. Schiefer, D. Sharpe, R. Sidle, A. Storm, and L. Zhang. (2013). "DB2 with BLU acceleration: So much more than just a column store". *Proceedings of the VLDB Endowment*. 6(11): 1080–1091. DOI: 10.14778/2536222.2536233.

Rao, J. and K. A. Ross. (1998). "Cache Conscious Indexing for Decision-Support in Main Memory". *Tech. rep.* URL: https://academiccommons.columbia.edu/doi/10.7916/D8T441ZB.

Rao, J. and K. A. Ross. (2000). "Making B+- trees cache conscious in main memory". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00*. New York, New York, USA: Association for Computing Machinery (ACM). 475–486. DOI: 10.1145/342009.335449.

Rao, J. and K. A. Ross. "Cache Conscious Indexing for Decision-Support in Main Memory". In: *Proceedings of the 25th International Conference on Very Large Data Bases*. URL: https://dl.acm.org/doi/10.5555/645925.671362 (accessed on 08/19/2020).

RAPIDS. (2020). "RAPIDS, Open GPU Data Science". URL: https://rapids.ai/.

Ravi, V. T., M. Becchi, G. Agrawal, and S. Chakradhar. (2011). "Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework". In: *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*. 217–228. DOI: 10.1145/1996130.1996160.

Raza, S. M. A., P. Chrysogelos, P. Sioulas, V. Indjic, A. C. Anadiotis, and A. Ailamaki. (2020). "GPU-accelerated data management under the test of time". In: *Online proceedings of the 10th Conference on Innovative Data Systems Research (CIDR)*. No. CONF.

Redis. "Redis". URL: https://redis.io/ (accessed on 08/19/2020).

Rödiger, W., T. Mühlbauer, P. Unterbrunner, A. Reiser, A. Kemper, and T. Neumann. "Locality-Sensitive Operators for Parallel Main-Memory Database Clusters". *Tech. rep.* URL: https://ieeexplore.ieee.org/abstract/document/6816684/.

Rossbach, C. J., J. Currey, M. Silberstein, B. Ray, and E. Witchel. (2011). "PTask: Operating system abstractions to manage GPUs as compute devices". In: *SOSP'11 - Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. 233–248. DOI: 10.1145/2043556.2043579.

Rozenberg CWI, E., A. ERozenberg, cwinl Peter Boncz CWI, and A. PBoncz. (2017). "Faster across the PCIe bus: A GPU library for lightweight decompression including support for patched compression schemes". *dl.acm.orgPaperpile*. May. DOI: 10.1145/3076113.3076122.

Rui, R., H. Li, and Y.-C. Tu. (2015). "Join algorithms on GPUs: A revisit after seven years". In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE. 2541–2550.

Rui, R. and Y.-C. Tu. (2017). "Fast equi-join algorithms on gpus: Design and implementation". In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. ACM. 17.

Rumble, S. M., A. Kejriwal, and J. Ousterhout. (2014). "Log-Structured Memory for DRAM-Based Storage". In: *Proceedings of the 12th USENIX Conference on File and Storage Technologies. FAST'14*. Santa Clara, CA: USENIX Association. 1–16.

Samuel, J., K. Ishizaki, and M. Kandasamy. (2016). "IBMSparkGPU.(2016)". *Retrieved January*. 9: 2018.

SAP HANA. "SAP HANA: In-Memory Database". URL: https://www.sap.com/sea/products/hana.html (accessed on 08/19/2020).

Satish, N., M. Harris, and M. Garland. "Designing Efficient Sorting Algorithms for Manycore GPUs". *Tech. rep.* URL: https://ieeexplore.ieee.org/abstract/document/5161005/.

Satish, N., C. Kim, J. Chhugani, A. D. Nguyen, V. W. Lee, D. Kim, and P. Dubey. (2010). "Fast sort on CPUs and GPUs: A case for bandwidth oblivious SIMD sort". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: ACM Press. 351–362. DOI: 10.1145/1807167.1807207.

Schaa, D. and D. Kaeli. (2009). "Exploring the multiple-GPU design space". In: *IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*. DOI: 10.1109/IPDPS.2009.5161068.

Schlegel, B., R. Gemulla, and W. Lehner. (2009). "k-ary search on modern processors". In: *Proceedings of the 5th International Workshop on Data Management on New Hardware, DaMoN 2009 in Conjunction with ACM SIGMOD/PODS Conference*. 52–60. DOI: 10.1145/1565694.1565705.

Schuh, S., X. Chen, and J. Dittrich. (2016). "An experimental comparison of thirteen relational equi-joins in main memory". In: *Proceedings of the 2016 International Conference on Management of Data*. ACM. 1961–1976.

Selinger, P. G., M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. (1979). "Access Path Selection in a Relational Database Management System". In: *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data. SIGMOD '79*. Boston, Massachusetts: Association for Computing Machinery. 23–34. DOI: 10.1145/582095.582099.

Sengupta, S., M. Harris, Y. Zhang, and J. D. Owens. (2007). "Scan Primitives for GPU Computing Publication Date Scan Primitives for GPU Computing". DOI: 10.2312/EGGH/EGGH07/097-106.

Severance, D. G. (1983). "A practitioner's guide to data base compression tutorial". *Information Systems*. 8(1): 51–62. DOI: 10.1016/0306-4379(83)90030-3.

Shaikhha, A., Y. Klonatos, L. Parreaux, L. Brown, M. Dashti, and C. Koch. (2016). "How to Architect a Query Compiler". In: *Proceedings of the 2016 International Conference on Management of Data. SIGMOD '16*. New York, NY, USA: ACM. 1907–1922. DOI: 10.1145/2882903.2915244.

Shanbhag, A., S. Madden, and X. Yu. (2020). "A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery. 1617–1632. DOI: 10.1145/3318464.3380595.

Shen, J., Z. Wang, D. Wang, J. Shi, and S. Chen. (2019). "Introducing AresDB: Uber's GPU-Powered Open Source, Real-time Analytics Engine". URL: https://eng.uber.com/aresdb/.

Sikka, V., F. Färber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhövd. (2012). "Efficient transaction processing in SAP HANA database - The end of a column store myth". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 731–741. DOI: 10.1145/2213836.2213946.

Singh, D. P., I. Joshi, and J. Choudhary. (2018). "Survey of GPU Based Sorting Algorithms". *International Journal of Parallel Programming*. 46(6): 1017–1034. DOI: 10.1007/s10766-017-0502-5.

Singh, S. and R. Nasre. (2019). "Optimizing Graph Processing on GPUs Using Approximate Computing: Poster". In: *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming. PPoPP '19*. New York, NY, USA: Association for Computing Machinery. 395–396. DOI: 10.1145/3293883.3295736.

Sintorn, E. and U. Assarsson. "Fast Parallel GPU-Sorting Using a Hybrid Algorithm". *Tech. rep.* URL: https://www.sciencedirect.com/science/article/pii/S0743731508001196.

Sioulas, P., P. Chrysogelos, M. Karpathiotakis, R. Appuswamy, and A. Ailamaki. "Hardware-conscious Hash-Joins on GPUs". URL: http://infoscience.epfl.ch/record/262530/files/sioulas%7B%5C_%7Dicde2019%7B%5C_%7Dgpujoin.pdf.

Sitaridi, E. A. and K. A. Ross. (2016). "GPU-accelerated string matching for database applications". *VLDB Journal*. 25(5): 719–740. DOI: 10.1007/s00778-015-0409-y.

Slee, M., A. Agarwal, and M. Kwiatkowski. (2007). "Thrift: Scalable cross-language services implementation". *Facebook White Paper*. 5(8).

splunk. (2018). "The Data-to-Everything Platform". URL: https://www.splunk.com.

SQream. (2019). "Reference architecture for 50-100 CONCURRENT users - SQream DB Reference Architectures".

Station, D. "NVIDIA DGX Station". https://www.nvidia.com/en-us/data-center/dgx-station/.

Stonebraker, M., D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'neil, P. O'neil, A. Rasin, N. Tran, and S. Zdonik. "C-Store: A Column-oriented DBMS". *Tech. rep.* DOI: 10.5555/1083592.1083658.

PG-Storm. (2020). "PG-Strom Manual". URL: https://heterodb.github.io/pg-strom/.

Tahboub, R. Y., G. M. Essertel, and T. Rompf. (2018). "How to Architect a Query Compiler, Revisited". In: *Proceedings of the 2018 International Conference on Management of Data. SIGMOD '18*. New York, NY, USA: ACM. 307–322. DOI: 10.1145/3183713.3196893.

Teubner, J. and R. Mueller. (2011). "How soccer players would do stream joins". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: ACM Press. 625–636. DOI: 10.1145/1989323.1989389.

"The AMD FX APU | AMD". URL: https://www.amd.com/en/products/fx-processors-laptop (accessed on 09/24/2020).

Thrust. "Thrust: NVIDIA Developer". URL: https://developer.nvidia.com/thrust (accessed on 08/21/2020).

Tomé, D., T. Tomé, T. Gubner, M. Raasveldt, E. Rozenberg, and P. Boncz. "Optimizing Group-By And Aggregation using GPU-CPU Co-Processing". *Tech. rep.*

Valentin, G., M. Zuliani, D. C. Zilio, G. Lohman, and A. Skelley. "DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes". *Tech. rep.* URL: https://ieeexplore.ieee.org/abstract/document/839397/.

Volk, P. B., D. Habich, and W. Lehner. "GPU-Based Speculative Query Processing for Database Operations". *Tech. rep.* URL: http://www.vldbarc.org/archives/workshop/2010/proceedings/files/vldb%7B%5C_%7D2010%7B%5C_%7Dworkshop/ADMS%7B%5C_%7D2010/adms10-volk.pdf.

Wanderman-Milne, S. and N. Li. (2014). "Runtime Code Generation in Cloudera Impala." *IEEE Data Eng. Bull.* 37(1): 31–37.

Wang, G. and G. Zhou. (2012). "GPU-based aggregation of on-line analytical processing". In: *Communications in Computer and Information Science*. Vol. 288 CCIS. No. PART 1. Springer, Berlin, Heidelberg. 234–245. DOI: 10.1007/978-3-642-31965-5_28.

Wang, K., K. Zhangz, Y. Yuan, S. Ma, R. Lee, X. Ding, and X. Zhang. (2014). "Concurrent analytical query processing with GPUs". *Proceedings of the VLDB Endowment*. 7(11): 1011–1022. DOI: 10.14778/2732967.2732976.

Wang, L., M. Huang, and T. El-Ghazawi. (2011). "Exploiting concurrent kernel execution on graphic processing units". In: *Proceedings of the 2011 International Conference on High Performance Computing and Simulation, HPCS 2011*. 24–32. DOI: 10.1109/HPCSim.2011.5999803.

Willhalm, T., I. Oukid, I. M. Uller, and F. Faerber. "Vectorizing Database Column Scans with Complex Predicates". *Tech. rep.*

Willhalm, T., N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. (2009a). "SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units". *Proceedings of the VLDB Endowment*. 2(1): 385–394. DOI: 10.14778/1687627.1687671.

Willhalm, T., N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. (2009b). "SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units". *Proceedings of the VLDB Endowment*. 2(1): 385–394. DOI: 10.14778/1687627.1687671.

Williams, H. E. and J. Zobel. (1999). "Compressing integers for fast file access". *Computer Journal*. 42(3): 193–201. DOI: 10.1093/comjnl/42.3.193.

Wu, H., G. Diamos, S. Cadambi, and S. Yalamanchili. (2012). "Kernel weaver: Automatically fusing database primitives for efficient GPU computation". In: *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE. 107–118.

Wu, H., F. Drive, G. Diamos, S. Baxter, M. Garland, T. Sheard, M. Aref, and S. Yalamanchili. (2014). "Red Fox: An Execution Environment for Relational Query Processing on GPUs". *Proceeding of theInternational Symposium on Code Generation and Optimization (CGO)*: 44:44–44:54. URL: http://doi.acm.org/10.1145/2544137.2544166.

Yan, Z., Y. Lin, L. Peng, and W. Zhang. (2019). "Harmonia: A high throughput B+tree for GPUs". In: *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP*. New York, NY, USA: Association for Computing Machinery. 133–144. DOI: 10.1145/3293883.3295704.

Yuan, Y., R. Lee, and X. Zhang. (2013). "The Yin and Yang of processing data warehousing queries on GPU devices". *Proceedings of the VLDB Endowment*. 6(10): 817–828.

Zaharia, M., M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. J. Franklin, and S. Shenker. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". *Tech. rep.* URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

Zhang, E. Z., Y. Jiang, Z. Guo, and X. Shen. (2010). "Streamlining GPU Applications on the Fly: Thread Divergence Elimination through Runtime Thread-Data Remapping". In: *Proceedings of the 24th ACM International Conference on Supercomputing. ICS '10*. New York, NY, USA: Association for Computing Machinery. 115–126. DOI: 10.1145/1810085.1810104.

Zhang, K., J. Hu, B. He, and B. Hua. (2017). "DIDO: Dynamic Pipelines for In-Memory Key-Value Stores on Coupled CPU-GPU Architectures". DOI: 10.1109/ICDE.2017.120.

Zhang, S., J. He, B. He, and M. Lu. (2013). "OmniDB: Towards portable and efficient query processing on parallel CPU/GPU architectures". *Proceedings of the VLDB Endowment*. 6(12): 1374–1377. DOI: 10.14778/2536274.2536319.

Zhong, J. and B. He. (2014). "Kernelet: High-throughput GPU kernel executions with dynamic slicing and scheduling". *IEEE Transactions on Parallel and Distributed Systems*. 25(6): 1522–1532. DOI: 10.1109/TPDS.2013.257. arXiv: 1303.5164.

Zilliz. (2020). "Zilliz builds open source data science software for the era of AI and IoT". URL: https://zilliz.com/.

Zu, Y., M. Yang, Z. Xu, L. Wang, X. Tian, K. Peng, and Q. Dong. (2012). "GPU-based NFA implementation for memory effcient high speed regular expression matching". In: *ACM SIGPLAN Notices*. Vol. 47. No. 8. 129–139. DOI: 10.1145/2370036.2145833.

Zukowski, M., S. Héman, N. Nes, and P. Boncz. "Super-Scalar RAM-CPU Cache Compression". *Tech. rep.* URL: https://ieeexplore.ieee.org/abstract/document/1617427/.

Zukowski, M., M. Van De Wiel, and P. Boncz. (2012). "Vectorwise: A vectorized analytical DBMS". In: *Proceedings - International Conference on Data Engineering*. 1349–1350. DOI: 10.1109/ICDE.2012.148.