DASlab: The Data Systems Laboratory at Harvard SEAS

Stratos Idreos

Harvard University
http://daslab.seas.harvard.edu

ABSTRACT

DASlab is a new laboratory at the Harvard School of Engineering and Applied Sciences (SEAS). The lab was formed in January 2014 when Stratos Idreos joined Harvard SEAS. DASlab curently consists of 3 PhD students, 1 postdoctoral researcher and 9 undergraduate researchers while it is set to double its graduate student population in the next one to two years. The lab is part of a growing community of systems and computer science researchers at Harvard; computer science faculty is scheduled to grow by 50% in the next few years.

The main focus of DASlab is on designing data systems that (a) make it easy to extract knowledge out of increasingly diverse and growing data sets and (b) can stand the test of time.

1. INTRODUCTION

At DASlab our long-term goal is to assist in minimizing the time it takes to turn data into knowledge by designing and building novel data systems, tailored for the new and ever-evolving challenges of a data-driven world.

Today knowledge hides in plain sight. It is quite likely that we already have all the raw data to discover solutions for many of the worlds big scientific challenges and we just do not have the proper systems (i.e., algorithms and data structures) to analyze the existing data or we simply do not know what to look for. As the size of the data we collect grows, data systems have become one of the most fundamental bottlenecks for data analytics; storing, accessing and analyzing raw data has to happen before any meaningful observations can take place and as it stands today we cannot even move, let alone store, access and analyze all our data.

In this document, we provide a brief overview of some of the research projects that are currently under way at DASlab and discuss our long term goal and motivation. Our research is driven by two fundamental goals outlined below.

Everyone should be a data-scientist. Knowledge is power and should be accessible to everyone. The more data we collect, though, the harder it becomes to make sense of the data. Today analyzing data requires expertise and resources that very few individuals hold. What if we made it equally easy for data scientists, data systems experts, astronomers, biologists, statisticians, as well as the general public to extract knowledge out of data via auto-tuning, interactive and intuitive data systems?

We should not have to design new data systems every decade. Computer science is a fast moving field. Every few years the environment changes dramatically with new hardware and new application requirements (meaning new data formats, models, etc.). What if we did not have to completely and manually redesign our data systems, algorithms and data structures every decade? What if we spend this energy designing more powerful ways to analyze data with systems that can automatically evolve to match the new environment?

The two high level goals above are strongly interconnected and they both lead to a vast array of fundamental computer science challenges. Our focus is on designing big data systems that are easy to use, i.e., with as few knobs as possible and with as little human input as possible; systems that just work, no matter the underlying hardware properties. Our vision is that data systems should be intuitive to use and interactive, guiding the user towards the interesting patterns in the data, doing just enough to generate the maximum insight within a limited time and resources budget. Indexing, tuning, optimization and other complex technical features should all be actions that are continuously active but always hidden from the users.

2. RESEARCH DIRECTIONS

In this section, we briefly describe some of the ongoing research projects at DASlab, focusing on discussing motivation and goals.

2.1 Self-designing Data Systems

Different applications and different hardware require different system designs to achieve optimal performance (i.e., throughput, query-latency, energy-performance etc.). Yet, so far, all data system architectures are static, operating within a single and narrowly defined design space (NoSQL, NewSQL, SQL, column-stores, row-stores, etc.) and hardware profile. Historically, a new system architecture requires at least a decade to reach a stable design. However, as we go deeper into the big data era, hardware and applications evolve continuously, leaving data-driven applications locked with sub-optimal systems. The more our businesses, sciences and every day life become data-driven, the more this becomes a fundamental shortcoming.

In this project, we ask the following question: *How many aspects of data system design can be abstracted and automated?* The end goal is that such systems take the "shape" of the data and queries with minimum human intervention during the design phase. This leads to systems that are fully tailored for a specific scenario and hardware profile, yet they are fast to design and implement. The ideal end result is fully self-designed and self-implemented systems.

We are currently experimenting with several ideas and designs about how data systems can self-design some of their most critical components. For example, inspired by the theory of evolution, a self-designing system may deploy multiple competing solutions down at the low level of its architecture such as using various combinations of data layouts, access methods and execution strategies. Then "the fittest design wins" and becomes the dominant architecture until the environment (workload and hardware) changes again. As new data and queries come in, a system evolves such that its architecture matches the properties of the incoming workload. Other research directions include the adoption of ideas from machine learning to make system design decisions, the use of advanced statistic models and what if analysis, as well as applying these concepts in combination.

Essentially, there are two distinct opportunities with this line of work. The first one targets system design: Given a set of data, queries and hardware, return a data system. The second one targets tuning of an existing system in an environment with diverse workload and hardware properties. In such cases, the complexity of the system tuning options makes it impractical to manually tune as the environment often changes over time as well.

As a more concrete example to what a self-designed system can be useful for, imagine the scenario where a research lab or a company has multiple different workloads that require very different system architectures while new requirements appear frequently. With self-designed data systems the data scientists would only need to point the system to the different datasets and it would take the appropriate shape for each one of them.

What does the appropriate shape mean? Query languages, data models, data layouts, physical layout, query processing operators are only a few of the design dimensions that a self-designed data system needs to decide upon. In our research, we started working throughout the whole stack by keeping the least common denominator so as the resulting system is generic and expandable enough. We consider the relational data model as well as RDF and other hierarchical data models, and we currently investigate the physical storage of such data, including the traditional row-store and column-store layouts as well as hybrid layouts.

This line of research creates numerous opportunities to bootstrap new applications, to automatically create systems that are tailored for specific scenarios, to minimize system footprint and automatically adapt to new hardware.

2.2 Queriosity: Auto-exploration

Data exploration is the natural paradigm for extracting knowledge out of data. It involves a series of steps such as: take a look at the data, try out a few models to see if they fit, look for outliers, learn from this experience and data seen so far, zoom into or out of this data set and repeat until satisfied with the knowledge gained. The output of each step is the input of the next one. Yet modern data systems are not designed with data exploration in mind.

Modern data systems are based on strict forms of interaction and they are designed for expert users who know enough both about the application domain and about how to set-up, tune and interact with data systems. This state of affairs restricts significantly the range of people who can actually explore data as well as the time it takes to extract knowledge out of growing data sets.

In this project, we ask the following question: *What if we had systems that can automatically explore rich data sets and report back interesting facts?* We are designing an autonomous "data robot", Queriosity, a portmanteau of *query* and *curiosity*.

Queriosity's goal is to explore data sets and figure out interesting patterns. It continuously learns about the fastest way to explore a given data set,

observes how interesting its findings are for the user and adjusts its strategy accordingly. As a result, with Queriosity the demanding task of data exploration is reduced to just the following two steps on the part of the user:

Here is a data set I want to explore;
Show me something interesting.

No more input from the user is needed other than confirming that a given insight is useful or not. Queriosity finds new insights as well as new relevant data sets automatically.

The design of an autonomous data exploration system that learns continuously presents various challenges on the conceptual as well as the system level. On the conceptual ground, for instance, we need to answer questions such as the following: What statistical properties of a dataset mirror user's interest? How can these indicators be determined without a priori information of either the dataset, the domain or both? What does it mean for an exploratory system to learn from experience? On the other hand, building the first autonomous data exploration system we are confronted with system level challenges such as the following: How to design a system that remains interactive while exploring and learning from potentially Petabytes of data? What form should the eventual system take i.e., collaborative, stand-alone or a hybrid? How does such a system integrate with myriads of existing data systems and data representations? How can it be designed to remain relevant decades from now?

With the prevalence of paradigms such as data-intensive science, Internet of things and information governance, we envision Queriosity finding application in virtually all domains as a *personal data scientist* that assists, data scientists in businesses and scientific research as well as people in every day life that try to make sense of the data around us.

2.3 Interactive and Visual Analytics

Why should researchers and data scientists have to learn complex languages and interfaces to use a data system? Why should they wait for several hours or days to analyze big piles of data if they only care about a small part? A data scientist should be able to simply point to the data and start extracting knowledge immediately.

The fundamental problem we are addressing in this project is that it is not easy anymore to extract knowledge out of data. Modern data scientists are confronted with complex systems and tools that lead to the following problems: (a) they are slow as they are designed with the traditional notion of processing all data to always give complete answers, (b) they rely on complex interfaces and languages which are meant only for experts of a specific system category and (c) they require expertise in terms of system tuning. The side effect is that data analysis becomes slow and requires expertise which becomes harder and harder for a single person to acquire. This directly translates to a significant financial overhead for businesses and scientific research labs. For example, a delay in the acquisition of knowledge implies lost business opportunities, while the need for more expertise implies that more personnel is required.

In this project we enable data analytics via intuitive touch interfaces and gestures and work towards a new class of data systems that are designed from the ground up to be interactive and tailored for data exploration. A data scientist can see, touch and explore data directly on a tablet device and can fully drive low-level query processing actions and complex analytics via gestures. The new paradigm is that the system continuously adjusts its storage and data access patterns to match the exploration path, it accesses only as much data as needed to instantly provide enough visual feedback to the data scientist, introducing a radically new data processing paradigm that is tailored for interactive exploration. Instead of having to learn complex languages and interfaces, data scientists interact with the data system via intuitive gestures. Instead of having to wait to set up and tune the system and its low level knobs, the system automatically adjusts to the tasks a data scientist performs. Instead of having to wait long stretches of time to get a complete answer over a big data set, the system gives a quick answer back and data exploration takes place as a continuous interaction between the data system and the data scientist which step by step leads to the interesting part in the data.

For example, in our current working prototype a data scientist can work directly over an iPad tablet using touch gestures. Data is visualized in the form of various geometric shapes (e.g., rectangles) and a data scientist can declare various queries/actions such as scans and aggregations and can start touching the data with slide and zoom-in/out gestures as opposed to using complex query languages. As data is being touched, the system plots interactive graphs to communicate the observed data patterns only for the actions performed and only for the data touched. As the data scientist adjusts the gesture characteristics and area of the data touched, the graphs get increasingly more complete and the data patterns become more and more apparent. The system is interactive enough such that every sin-

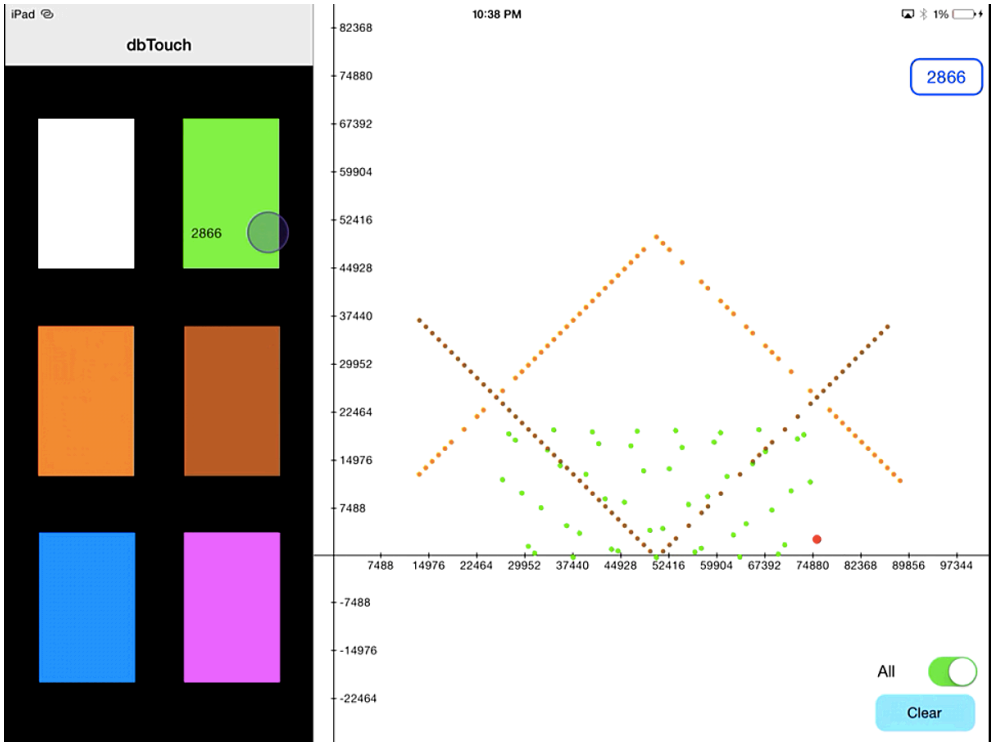

Figure 1: Interactive gesture-based exploration with dbTouch [18,26].

gle touch instantly translates to a visual change in the interactive graphs and data scientists can immediately use this information as feedback on which parts of the data they should focus next. Figure 1 depicts a screen-shot of our current prototype; the grey circle shows where a finger touches a data object which results in data plotted on the right hand side. With a few simple slide gestures we can get a quick understanding about the data distribution in the data set represented by the rectangle objects. The x-axis in the plot represents row-ids while the y-axis represents data values.

We are currently exploring similar ideas for data analytics via 3D gestures and in virtual reality environments. The common challenge is the design of data systems kernels that are interactive at their core regardless of the data sizes as opposed to state of the art systems that are monolithic.

2.4 Indexing in Modern Data Systems

Modern data systems do not rely on indexing in the same way past systems did. At a first glance this is for a good reason. For example, modern systems that target analytical workloads can perform very efficient scans using technologies such as column-storage which allows for reading fewer data items than a traditional row-store. In addition, relying on fixed width and dense columns allows for scans with tight for-loops which give excellent opportunities for prefetching and eliminate if statements and branch misses from the critical path. Similarly, exploiting SIMD and modern multi-core CPUs, as well as working directly over compressed data, allow for extremely fast scans. Another significant trend is that with larger memory sizes, data systems applications will store all hot data in memory, removing the major bottleneck of reading data from disk. At the same time, advanced query processing techniques allow for scans to answer multiple queries at the same time in environments where concurrent query requests are the norm; essentially we can answer N queries with the cost of a single scan.

In light of all these developments here we ask the question: *Are secondary indexes still useful today and if so in what form?* In our research we consider multiple variations of secondary indexes, from full indexes like a B$^+$-Tree to partial indexes like zonemaps. We analyze modern data systems using both models and experimentation. Our initial findings suggest that the decision of index use requires not only the traditional parameter of query selectivity but also other system conditions. This is especially true in light of a new trend in modern analytical workloads: increasing query concurrency.

We also find that designing strict index structures sacrifices performance when the workload changes. This leads to the investigation of dynamic and morphable data structures that can balance read, write and space amplification on-the-fly.

2.5 Hardware Software Co-design

Typically, software is being designed based on, apart from the workload, the available hardware. In this line of research, we investigate opportunities to reverse this relationship. The driving force of hardware development has been the applications, hence, in a perfect world hardware should be co-designed with the software. As a concept this is not a new idea; database machines have been studied extensively in the past and have been revisited in recent years. There were good reasons to abandon such ideas in the past especially given how fast commodity hardware was evolving and scaling but there are also exciting new opportunities today. Rather than proposing a full hardware redesign, in this project we investigate more hybrid solutions where the architecture of a database system remains generic enough and can be assisted by strategically placed accelerators that exploit game-changing modern hardware properties. Below we describe two of the directions we are pursuing.

Near Memory Processing. As the cost of accessing main memory becomes increasingly expensive and the memory size becomes larger, the main bottleneck of any data system is caused by fetching data to the processor. Here we ask the question: *Which parts of the processing can be offloaded in a generic enough way to functional units near memory or disk?* We design near-data query operators like select and project and we currently investigate the design of other operators, like hashing, sorting, and aggregations. Envisioning a complete architecture for near memory processing hardware is not as straightforward as pushing all actions close to the data. For example, while it is easy to see that it makes sense to push all selections, when it comes to more complex operators such as joins the discussion becomes more interesting; data after a join may be bigger than before, and so we may end up pushing more data up the memory hierarchy.

The Relational Disk. One of the most challenging questions in data management recently has been the bridging of the requirements of transactional and analytical workloads. There have been many proposals on the data systems layer, while, arguably, it has proven to be very difficult to provide a system capable of doing both analytical and transactional processing equally efficiently. Here we ask the question: *What is needed from the hardware in order to build a data system capable of bridging the analytical and transactional processing?* A disk-subsystem capable of delivering column-store performance when accessing a single column, yet supporting row-wise updates is the ultimate goal. Such a storage subsystem which we call *The Relational Disk* requires either a radically new hardware design assuming, instead of generic file structure, relational file organization, or enough add-on functionality (accelerators) to present the "illusion" of optimal row-wise and column-wise accesses.

3. INSPIRATION AND PAST WORK

For our work at DASlab we draw inspiration from several lines of work in the DB community. Most prominently we align with other exciting work on database architectures, adaptive systems and data exploration. Our own past work lies on a breadth of topics and is joint work with several labs, more frequently with colleagues from the database groups at CWI, EPFL, HP Labs, Microsoft Research, IBM Research, Google, Paris Descartes University, and NUS. The big chunk of our past work has focused on a series of topics on how to minimize the cost of bootstrapping database systems. Below we briefly point to some of these research projects.

With our work on adaptive indexing we have studied the design of modern data systems where indexing requires zero human intervention and tuning. Indexes are created adaptively and incrementally as a side-effect of query processing [10, 15, 17, 9, 16, 7, 20, 32, 35, 31, 8]. Building on top of such adaptive ideas our work on column/row hybrids proposes a system with adaptive storage components [4] that can choose the optimal layout on-the-fly.

Similarly, our work on adaptive loading presents the design of systems that do not require data loading up front. Instead, such systems can work directly on top of raw data, while matching the performance of traditional systems [12, 2, 3].

Our work on touch-based interactive systems introduced the idea of systems that are interactive enough to immediately react on gestures that represent query actions over big data sets [18, 26].

Our past work also includes work on core architecture topics such as column-store architectures [1, 5, 13], exploiting compressed indexes [14], designing column-stores that support stream processing [25, 29, 30], effectively utilizing compression during join processing [24] as well as proposing statistics oblivious access paths [6].

Furthermore, our past work has also focused on distributed query processing, for several different data models, e.g., relational [22, 19], information retrieval [33, 34] and RDF [28, 27].

Finally, much of the above work can be captured under the umbrella of data exploration techniques in an effort to design database kernels that support data exploration at their core [11, 23, 21].

4. ACKNOWLEDGMENTS

Our work is currently funded by the Harvard School of Engineering and Applied Sciences, the US National Science Foundation, the Swiss National Science Foundation, Facebook, and Google.

5. REFERENCES

[1] D. Abadi, P. A. Boncz, S. Harizopoulos, S. Idreos, and S. Madden. The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases*, 5(3):197–280, 2013.

[2] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: efficient query execution on raw data files. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 241–252, 2012.

[3] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: efficient query execution on raw data files. In *Communications of the ACM, Research Highlights*, 2015.

[4] I. Alagiannis, S. Idreos, and A. Ailamaki. H2O: a hands-free adaptive store. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1103–1114, 2014.

[5] R. Barber, P. Bendel, M. Czech, O. Draese, F. Ho, N. Hrle, S. Idreos, M.-S. Kim, O. Koeth, J.-G. Lee, T. T. Li, G. M. Lohman, K. Morfonios, R. Müller, K. Murthy, I. Pandis, L. Qiao, V. Raman, R. Sidle, K. Stolze, and S. Szabo. Business Analytics in (a) Blink. *IEEE Data Eng. Bull.*, 35(1):9–14, 2012.

[6] R. Borovica, S. Idreos, A. Ailamaki, M. Zukowski, and C. Fraser. Smooth scan: Statistics-oblivious access paths. In *Proceedings of the International Conference on Data Endineering (ICDE)*, 2015.

[7] G. Graefe, F. Halim, S. Idreos, H. Kuno, and S. Manegold. Concurrency Control for Adaptive Indexing. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(7):656–667, 2012.

[8] G. Graefe, S. Idreos, H. Kuno, and S. Manegold. Benchmarking adaptive indexing. In *Proceedings of the TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC)*, pages 169–184, 2010.

[9] F. Halim, S. Idreos, P. Karras, and R. H. C. Yap. Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-Memory Column-Stores. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(6):502–513, 2012.

[10] S. Idreos. Database Cracking: Towards Auto-tuning Database Kernels. *CWI, PhD Thesis*, 2010.

[11] S. Idreos. *Big Data Exploration*. Taylor and Francis, 2013.

[12] S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki. Here are my Data Files. Here are my Queries. Where are my Results? In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2011.

[13] S. Idreos, F. Groffen, N. Nes, S. Manegold, S. Mullender, and M. L. Kersten. MonetDB: Two Decades of Research in Column-oriented Database Architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.

[14] S. Idreos, R. Kaushik, V. R. Narasayya, and R. Ramamurthy. Estimating the compression fraction of an index using sampling. In *Proceedings of the International Conference on Data Endineering (ICDE)*, pages 441–444, 2010.

[15] S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2007.

[16] S. Idreos, M. L. Kersten, and S. Manegold. Updating a cracked database. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 413–424, 2007.

[17] S. Idreos, M. L. Kersten, and S. Manegold. Self-organizing tuple reconstruction in column stores. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 297–308, 2009.

[18] S. Idreos and E. Liarou. dbTouch: Analytics at your Fingertips. In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[19] S. Idreos, E. Liarou, and M. Koubarakis. Continuous multi-way joins over distributed hash tables. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 594–605, 2008.

[20] S. Idreos, S. Manegold, H. Kuno, and G. Graefe. Merging What's Cracked, Cracking What's Merged: Adaptive Indexing in Main-Memory Column-Stores. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 4(9):585–597, 2011.

[21] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of Data Exploration Techniques. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2015.

[22] S. Idreos, C. Tryfonopoulos, and M. Koubarakis. Distributed evaluation of continuous equi-join queries over large structured overlay networks. In *Proceedings of the International Conference on Data Endineering (ICDE)*, page 43, 2006.

[23] M. Kersten, S. Idreos, S. Manegold, and E. Liarou. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 4(12):1474–1477, 2011.

[24] J. Lee, G. K. Attaluri, R. Barber, N. Chainani, O. Draese, F. Ho, S. Idreos, M. Kim, S. Lightstone, G. M. Lohman, K. Morfonios, K. Murthy, I. Pandis, L. Qiao, V. Raman, V. K. Samy, R. Sidle, K. Stolze, and L. Zhang. Joins on encoded and partitioned data. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 7(13):1355–1366.

[25] E. Liarou, R. Goncalves, and S. Idreos. Exploiting the power of relational databases for efficient stream processing. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 323–334, 2009.

[26] E. Liarou and S. Idreos. dbTouch in Action: Database kernels for touch-based data exploration. In *Proceedings of the International Conference on Data Endineering (ICDE)*, pages 1262–1265, 2014.

[27] E. Liarou, S. Idreos, and M. Koubarakis. Evaluating conjunctive triple pattern queries over large structured overlay networks. In *International Semantic Web Conference*, pages 399–413, 2006.

[28] E. Liarou, S. Idreos, and M. Koubarakis. Continuous rdf query processing over dhts. In *ISWC/ASWC*, pages 324–339, 2007.

[29] E. Liarou, S. Idreos, S. Manegold, and M. L. Kersten. Monetdb/datacell: Online analytics in a streaming column-store. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(12):1910–1913, 2012.

[30] E. Liarou, S. Idreos, S. Manegold, and M. L. Kersten. Enhanced stream processing in a DBMS kernel. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 501–512, 2013.

[31] E. Petraki, S. Idreos, and S. Manegold. Holistic indexing in main-memory column-stores. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2015.

[32] H. Pirk, E. Petraki, S. Idreos, S. Manegold, and M. L. Kersten. Database cracking: fancy scan, not poor man's sort! In *Proceedings of the International Workshop on Data Management on New Hardware (DAMON)*, 2014.

[33] C. Tryfonopoulos, S. Idreos, and M. Koubarakis. Publish/subscribe functionality in ir environments using structured overlay networks. In *ACM SIGIR Special Interest Group on Information Retrieval*, pages 322–329, 2005.

[34] C. Tryfonopoulos, S. Idreos, M. Koubarakis, and P. Raftopoulou. Distributed large-scale information filtering. *T. Large-Scale Data- and Knowledge-Centered Systems*, 13:91–122, 2014.

[35] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1555–1566, 2014.