Big Data Research: Will Industry Solve all the Problems?

Magdalena Balazinska
Dept. of Computer Science & Engineering
University of Washington
magda@cs.washington.edu

Abstract

The need for effective tools for big data data management and analytics continues to grow. While the ecosystem of tools is expanding many research problems remain open: they include challenges around efficient processing, flexible analytics, ease of use, and operation as a service. Many new systems and much innovation, however, come from industry (or from academic projects that quickly became big players in industry). An important question for our community is whether industry will solve all the problems or whether there is a place for academic research in big data and what is that place. In this paper, we address this question by looking back at our research on the Nuage, CQMS, Myria, and Data Pricing projects, and the SciDB collaboration.

1. INTRODUCTION

Over the past ten years, the need to manage big data has become an increasingly pressing problem in industry. As a result, the landscape of big data management systems has grown at a rapid pace. The Hadoop stack [3] is the prime example with innovations at the storage (HDFS), execution (Hadoop MapReduce), resource management (Hadoop YARN), and declarative query layers (*e.g.*, Apache Hive [4]). Many other systems are also under active use and development including Cloudera Impala [9], Amazon Redshift [2], Apache Storm [6], and many others. With so much innovation happening in industry, a natural question for the academic community is whether industry will solve all the big data problems or whether there is a place for academic research in big data management. The clear answer, of course, is that academic research has an important role to play. In fact, some of the more transformative systems in use in industry today (*e.g.*, Spark [5] and GraphLab [12]) originated in academia. The more accurate question to ask is thus how best to contribute to the vibrant big data management area. In this paper, we reflect on big data management research by looking back at example innovations from our Nuage [26], CQMS [10], Myria [25], and Data Pricing [11] projects, and the SciDB [28, 29] collaboration.

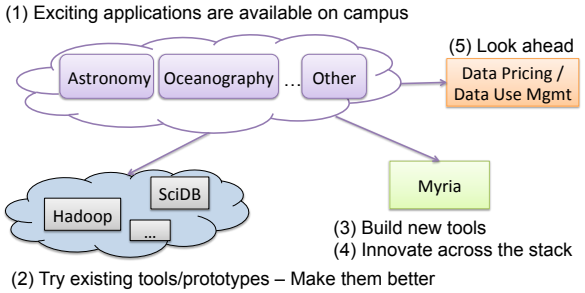

Figure 1: Academia can contribute in many ways to research in big data management and analytics.

2. FOUNDATION IN SCIENTIFIC DATA AND APPLICATIONS

Similar to industry, science is increasingly becoming data-driven [13]. From small research labs to large communities [21,30], scientists have access to more data than ever before. As academic researchers on a university campus, we have access to these exciting domain science datasets and problems. In our big data research in the database group at the University of Washington, we are tapping into the real data and workloads from science collaborators on campus. These real applications enable us to understand some of the challenges related to big data management. The applications that we have studied include the analysis of telescope images [31], N-body simulation data analysis [19, 20], processing of bibliometrics data, social networking data analysis, and, most recently, natural language processing and biology data analysis. We focus on the first two applications as more detailed illustrative examples.

The goal of the N-body application [19, 20] is to study how structure forms and evolves in the universe. The approach is to use simulations that typically start shortly after the Big Bang and run the full lifespan of the universe, approximately 14 billion years. The output of the simulation and thus the input to the analysis takes the form of a single table. The universe is represented with a set of particles. Each row in the table captures the state of one particle at one timestep during the simulation. This dataset while simple exercises data management systems in interesting ways. First the input data is growing at a fast pace. Simulations are increasingly longer, large-volume, and fine-grained with growing numbers of particles. Just a few

Proceedings of the VLDB Endowment, Vol. 8, No. 12

years ago, the output of a simulation was tens of GB in size. Last year, it was a few TB in size. This year, it is hundreds of TB in size. Second, the analysis ranges from simple to complex: select, project, join, and aggregate queries are useful for basic data exploration [19]. But the important analysis requires data clustering (with arbitrarily large clusters) [18] to identify galaxies at each timestep and the study of the evolution of these galaxies over time [20].

In the telescope image data analysis application [31], the goal is to extract celestial objects (*e.g.*, galaxies) from telescope images such as those collected by the the Sloan Digital Sky Survey (SDSS [30]) or the upcoming Large Synoptic Survey Telescope (LSST [21]). The input data takes the form of a time-series of 2D images at different locations in the sky. Because some sources are too faint to appear in a single image, the approach is to co-add a stack of images before performing the source extraction. There are several interesting challenges with this application: First, the original data lies on a sphere rather than a flat 2D surface. Second, the co-addition requires an iterative data cleaning process to remove outliers. Finally, the data is large in the order of tens of TB for the SDSS and tens of TB per night for the LSST.

With the above examples, we found that the straightforward application of existing data management systems could already yield some improvements in productivity and efficiency of scientific data analysis through declarative querying and the seamless manipulation of datasets too large to fit in memory [19]. At the same time, it unveiled interesting limitations in existing technologies that lead to innovations as we describe next. Other applications will have other interesting requirements for data management systems and will push their boundaries in different ways.

3. EXTENDING EXISTING TOOLS

Early in our Nuage [26] project, which focused on big data management in the context of scientific applications, we applied single-node relational database management systems (RDBMSs) and also Dryad [14] and Hadoop [3] to the applications described above. We found that it was far from trivial to use these systems in a way that resulted in high performance. As a concrete example, we applied Dryad (and later Hadoop) to the data clustering step in the N-body application [18]. While Dryad and Hadoop made it easy to express the computation, both lead to problems with uneven load distribution, also called skew. This observation lead us to perform a more in-depth study of skew in MapReduce applications, thus generalizing the problem beyond the original use-case. We then built the SkewReduce and SkewTune [17] systems, which both extended Hadoop to better manage skewed data either statically or dynamically (please find the full list of publications on our project website [26]). Both systems are available as open source, have been downloaded by many groups, and can be accessed through the Nuage project website. In that same line of work, motivated by scientific applications and limitations we were observing in existing systems, we developed a progress indicator for MapReduce workflows [24], extended Hadoop with efficient support for iterative analytics [7], developed a fault-tolerance optimizer for parallel, shared-nothing systems [34], and evaluated the benefits of lazy evaluation in MapReduce computations. In all cases, we were inspired by concrete problems found in academia and developed general-purpose solutions by extending existing tools.

Through the SciDB collaboration [28, 29] whose goal has been to build a parallel array processing system, we experimented with applying the SciDB engine to scientific problems. Because SciDB is an array engine, the telescope image analysis application was an obvious fit. We used this application and other workloads to motivate and evaluate a new storage manager for parallel array engines [32]: We evaluated the performance of different types of array partitioning and on-disk organization methods and also different techniques for storing multiple versions of an array in support of efficient time-travel operations. Beyond the storage layer, we developed new query execution methods, especially for parallel array processing with overlap and for iterative array processing [31]. Finally, we studied how to handle spherical rather than Cartesian data and developed the AscotDB system for telescope image analysis on top of our extended version of SciDB. The full list of papers appears on our project website [29].

Lessons Learned: These first experiences with big data research made it clear to us that we had access to exciting scientific applications on campus. Starting with these real applications and applying existing tools to these applications enabled us to identify specific limitations of current tools, improve and extend these tools, and give back to the community through open source software and publications. We found this approach to be the fastest method to produce publishable results and get attention. Most importantly, the new problems that we identified in domain sciences in academia lead to general research and solutions that are broadly applicable. As a result, we found that users beyond academia were experiencing similar problems in many cases and appreciated our contributions.

4. BUILDING NEW TOOLS

In academia, we may not have access to large engineering teams, but we can still invest in building new data management systems from scratch rather than remain constrained by existing tools. Building a new tool opens the door to being more transformative and more creative compared with using existing engines. Furthermore, in academia, when building a new system, we can focus on the novel research problems, the new techniques, without worrying that the system be feature complete. At the same time, building a system sufficiently full-featured to support real users enables the research to remain focused on real problems.

In that spirit, over the past few years, our group (in collaboration with members of the eScience Institute) has moved beyond Hadoop and SciDB and built our own, parallel data management system called Myria [25]. Myria combines state-of-the-art methods for parallel data management together with new techniques and algorithms. We use the system for research in efficient query processing, operation as a cloud service, and usability. We are also operating Myria as a service for users on the University of Washington campus.

Efficient Query Processing: Myria's query execution layer is called MyriaX. It is a relational, parallel, shared-nothing engine. Similar to other engines, it comprises a coordinator node and a set of worker nodes. As in HadoopDB [1], datasets ingested into Myria are sharded into PostgreSQL databases local to each node. MyriaX can read from other sources including HDFS and Amazon S3, but it uses PostgreSQL as internal

storage to leverage the tool's binary storage and data indexing capabilities. During query processing, once data is read out of PostgreSQL, MyriaX processes it entirely in memory using its own, often novel, operators. The engine streams data directly from one operator to the next without going to disk and it can scale out elastically as needed. Finally, MyriaX query plans can have loops to support iterative computations. Based on this state-of-the-art foundation, we innovate by developing new query processing algorithms.

In a recent project, we focused on efficient and fault-tolerant *iterative* query processing [35]. We were unhappy with existing methods for iterative computations because they all had important restrictions: Some were limited to synchronous processing with synchronization barriers at the end of each iteration. Others did support asynchronous processing but were specialized for graphs. Others yet were more general but were exposing a low-level interface where users needed to build query plans manually. To address these limitations, using our Myria system, we developed a new approach to iterative query processing. With our approach, users express declarative queries (*i.e.*, Datalog programs with what we define as bag-monotonic aggregates or equivalent MyriaL scripts) that get automatically compiled into parallel, shared-nothing plans with loops. These plans are evaluated incrementally, either asynchronously or synchronously, and with various types of prioritized processing and failure handling methods. Because we implemented the approach in our Myria system, we were able to leverage our previous design choices, such as ensuring the system never imposes unnecessary synchronization barriers. We were also able to extend the system as we chose and make our new features directly available to our users. By testing the approach on a variety of real applications, including the N-body application described above, we found that different iterative execution models yield the fastest query runtimes for different applications in practice. Asynchronous processing is not always the best approach and, when it is, processing priorities can dramatically affect query execution times.

In a second project, we focused on the theoretical foundations of parallel query processing. Going back to theory and foundations is an important component of academic research. In this project, we considered recent theoretical results on join processing in MapReduce systems and new developments in single-node multiway join processing. Using our MyriaX execution engine, we studied the systems aspects of these novel algorithms, the benefits of their joint use during parallel query evaluation, and developed algorithms to make their use practical in arbitrarily-sized clusters [8].

For both projects, the resulting innovations have been integrated into the Myria stack and either are already available to our users or are in the process of becoming available.

Operation as a Service: We are operating Myria as a cloud service using the database group's private cluster. Users can access the service directly from their browsers using either our generic interface or special-purpose interfaces. One example of a specialized service on top of Myria is an application that we developed for the analysis of galactic merger trees in N-body simulations [20]. Users can also write Python (or other) scripts to talk to the Myria stack.

By operating Myria as a cloud service, we are taking the opportunity to re-think the interface and guarantees that cloud data services should offer. With existing cloud services, users either pay by the gigabyte processed (Google BigQuery) or by instance-hour (Amazon Web Services). In Myria, we take a completely different approach. Users show us the schema and size of their data. We generate for them, what we call, *Personalized Service Level Agreements (PSLAs)* [27]. These PSLAs show different service levels, each with a fixed hourly price. At each service level, the user is shown the expected query runtimes for different patterns of queries on their data. Under the covers, we launch Myria clusters in the cloud and elastically re-size these clusters to meet the performance advertised in the PSLAs. Our new interface holds the promise to dramatically simplify the usability of cloud services. It also opens interesting research questions including how best to guarantee query runtimes in a cloud service? How to explain query performance without teaching users about query plans? How to help users write queries in a way that leads to better performance? We are actively pursuing these questions.

Usability: Finally, in the context of the Myria project but also in our earlier CQMS project [10], we studied problems related to database usability. Increasingly many data scientists and data enthusiasts need to interact directly with big data management and analytics systems: How can we make these users more productive? Through this research thread, we developed new techniques to help users articulate SQL queries. We built the SnipSuggest system [16] that provides contextual SQL autocomplete recommendations based on similar queries written by other users. We also developed techniques that let users ask declarative queries about the performance of their analysis in the PerfXPlain system [15]. Most recently, we developed data cleaning techniques that maximize the quality of a user's visualization while minimizing the user's data cleaning efforts.

Lessons Learned: Overall, we find that building a new database system is a significant investment but it opens the door to greater freedom to try new designs and new approaches. These new methods may not entirely work at first, which is fine in an academic setting, but they drive interesting research and eventually become practical or give rise to new ideas. Interestingly, in academia, building a new system means focusing on the new features and new research rather than completeness of features. At the same time, however, putting the engine in the hands of real users helps to unveil the critical challenges to address next. It also makes it easier to identify research problems at different levels in the stack.

5. LOOKING IN NEW PLACES

A final exciting aspect of academic research in big data is that we can take the time to look in places where others are not looking yet. For example, when analyzing big data, a feature that has become prominent is that users often analyze data provided by other users or organizations (*e.g.*, SDSS data or MIMIC-II [23]). In cases where a user analyzes external data, the data can come at a cost. The Azure Marketplace [22] is an example web service where users can buy or sell data. Some datasets are free, but most datasets cost money. Today, industry uses simple data pricing mechanisms. Some users complain about the limited pricing capabilities (*e.g.*, when the data changes, users often have no easy way to purchase only updates) but the problem is not yet sufficiently widespread to draw attention. In academia, however, we have the luxury to look into problems that are looming on the horizon. In this case, an interesting research question is how best to price relational (or other) data.

The goal is to enable buyers to purchase exactly the data that they need without requiring in-person negotiations, which don't scale. In a recent project, we developed new methods for data pricing (see publications on our project website [11]), where the key idea is to price *queries* (or views) rather than individual tuples because queries better capture the information content of the data and enable the enforcement of important properties such as arbitrage-free pricing.

Datasets from third-party providers are also typically accompanied by terms of use. For example, the MIMIC-II data, which is free to use for research, requires that users take an online course to learn what they can and cannot do with the data. Through an informal survey of 13 data providers, we found that all datasets (or APIs to access them) came with terms of use, averaging 8 pages in length. Reading, understanding, and respecting such agreements is tedious and error-prone even for honest users. Today, the process is manual. While industry is tolerating the current manual data use management process, as the use of third-party data grows, such a process will eventually become untenable. To address this challenge, we developed new techniques for automatically enforcing data use policies with low overhead [33]: We developed a model for specifying data use policies in SQL over the data's usage log and developed a new method with several optimizations for efficiently enforcing the policies. The resulting system is available as open source and can be accessed through our project website.

Lessons Learned: Many big data management challenges go beyond performance, usability, and service operation. Some of these problems, including pricing and data use management are not receiving as much attention as they should because they have not yet become critical problems. We anticipate that they will grow in importance. In academia, we can look many years in the future, study what we expect to become the important challenges, and develop principled solutions and system prototypes for addressing them.

6. CONCLUSION

The landscape of big data research is rich. Academia has many ways to contribute to this exciting field by applying and improving existing tools, developing new and different tools, considering challenges across the stack from usability to service operation, and by looking at problems years in the future. For a long time, data management has been a crucial tenet of industry and sciences. Trends all point to its continued growth.

Acknowledgments: I would like to thank all members of the UW database group, our UW eScience Institute and domain science collaborators, and our collaborators beyond UW. I would also like to thank our sponsors: the National Science Foundation, the Intel Science and Technology Center for Big Data, Amazon, Microsoft, Google, EMC, HP, and Yahoo.

7. REFERENCES

[1] A. Abouzeid et al. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *PVLDB*, 2(1):922–933, Aug. 2009.

[2] Amazon Redshift. `http://aws.amazon.com/redshift/`.

[3] Apache Hadoop. `http://hadoop.apache.org/`.

[4] Apache Hive. `https://hive.apache.org/`.

[5] Spark: Lightening-Fast Cluster Computing. `https://spark.apache.org/`.

[6] Apache Storm. `https://storm.apache.org/`.

[7] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst. HaLoop: Efficient iterative data processing on large clusters. *Proc. of the VLDB Endowment*, 3(1):285–296, 2010.

[8] S. Chu, M. Balazinska, and D. Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *Proc. of the SIGMOD Conf.*, pages 63–78, 2015.

[9] Impala. `http://impala.io/`.

[10] Collaborative query management. `http://cqms.cs.washington.edu/`.

[11] Data Eco$y$tem. `http://cloud-data-pricing.cs.washington.edu/`.

[12] Dato: Create intelligence. `https://dato.com/`.

[13] T. Hey, S. Tansley, and K. Tolle. The fourth paradigm: Data-intensive scientific discovery, 2009.

[14] M. Isard et al. Dryad: Distributed data-parallel programs from sequential building blocks. In *Proc. of EuroSys Conf.*, pages 59–72, 2007.

[15] N. Khoussainova, M. Balazinska, and D. Suciu. PerfXPlain: Debugging MapReduce job performance. *Proc. of the VLDB Endowment*, 5(7):598–609, 2012.

[16] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. SnipSuggest: Context-aware autocompletion for SQL. *Proc. of the VLDB Endowment*, 4(1):22–33, 2010.

[17] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. SkewTune: Mitigating skew in MapReduce applications. In *Proc. of the SIGMOD Conf.*, 2012.

[18] Y. Kwon et al. Scalable clustering algorithm for N-body simulations in a shared-nothing cluster. In *SSDBM*, 2010.

[19] S. Loebman et al. Analyzing massive astrophysical datasets: Can Pig/Hadoop or a relational DBMS help? In *Proc of IASDS'09*.

[20] S. Loebman et al. Big-data management use-case: A cloud service for creating and analyzing galactic merger trees. In *Proc. of DanaC 2014*, pages 1–4, 2014.

[21] Large Synoptic Survey Telescope. `http://www.lsst.org/`.

[22] Microsoft Azure Marketplace. `http://azure.microsoft.com/en-us/marketplace/`.

[23] MIMIC II Databases. `http://physionet.org/mimic2`.

[24] K. Morton, M. Balazinska, and D. Grossman. ParaTimer: A progress indicator for MapReduce DAGs. In *Proc. of the SIGMOD Conf.*, June 2010.

[25] Myria: Big Data management as a Cloud service. `http://myria.cs.washington.edu/`.

[26] Nuage: Scientific Data Management in the Cloud. `http://nuage.cs.washington.edu/`.

[27] J. Ortiz, V. T. de Almeida, and M. Balazinska. Changing the face of database services with personalized service level agreements. In *Seventh CIDR Conf.*, 2015.

[28] SciDB. `http://www.scidb.org/`.

[29] UW SciDB Branch. `http://scidb.cs.washington.edu/`.

[30] Sloan Digital Sky Survey. `http://cas.sdss.org`.

[31] E. Soroush, M. Balazinska, K. S. Krughoff, and A. J. Connolly. Efficient iterative processing in the SciDB parallel array engine. In *SSDBM*, 2015.

[32] E. Soroush, M. Balazinska, and D. Wang. ArrayStore: A storage manager for complex parallel array processing. In *Proc. of the SIGMOD Conf.*, pages 253–264, June 2011.

[33] P. Upadhyaya, M. Balazinska, and D. Suciu. Automatic enforcement of data use policies with DataLawyer. In *Proc. of SIGMOD Conf.*, pages 213–225, 2015.

[34] P. Upadhyaya, Y. Kwon, and M. Balazinska. A latency and fault-tolerance optimizer for online parallel query plans. In *Proc. of the SIGMOD Conf.*, pages 241–252, June 2011.

[35] J. Wang, M. Balazinska, and D. Halperin. Asynchronous and fault-tolerant recursive datalog evaluation in shared-nothing engines. *PVLDB*, 8(12), 2015.