Research Statement

Jennie Duggan
jennie@csail.mit.edu

The management and processing of big data affects nearly everyone from businesses to individuals. This data deluge presents many new research challenges and opportunities. Data-driven applications—including personalized medicine, adaptive public transportation, and credit card fraud detection—are being made tractable by the availability of abundant, high-quality data. On the other hand, storing and querying this diverse, ever-growing data at scale necessitates completely new approaches to data management.

In my research, I *design and build systems for large-scale data analytics.* My emphasis is on creating systems that are performant and predictable. To provide performance, I develop novel parallel data processing techniques [11] and for predictability, I apply statistical machine learning to model expected workload performance [3, 5, 7, 8]. My approach frequently optimizes core database internals, such as storage management [1, 6, 10] and query execution planning [4], improving their use of system hardware.

Summary of Research

Optimizing the Performance of Database Workloads

My dissertation addressed the prediction of performance for queries simultaneously executing on the same hardware platform. By studying the relationship between databases and their hardware, I created a framework that estimated the duration of individual queries before they started executing. My work developed abstractions to quantify how multiple interleaved queries affect one another's rate of progress. This forecasting is complicated by the interactions among queries as they share resources (i.e., memory and I/O). The system used statistical machine learning to approximate this contention. Such predictions are useful for a wide variety of applications, including reduction of the completion time of large query batches, better provisioning and query-to-server assignments for cloud databases, improved query progress indicators, and concurrency-aware query optimizers.

I first researched this problem for static workloads with well-defined queries [3] and progressed on to ad-hoc workloads [5]. The latter had a two phase approach; it first estimated the resource requirements of the query under prediction. The predictor then forecasted how heavily the other concurrent queries used shared hardware resources by analyzing their data access patterns. Its predictions came within 25% of the correct value on average on our PostgreSQL/TPC-DS testbed.

After that, I extended my work to portable database performance prediction, wherein a model forecasts the throughput of a query workload on a new hardware platform upon which it has not executed in the past [7]. This work makes it possible for database administrators to dynamically provision the right level of infrastructure for a given database schema and collection of queries executed. The framework first identifies example workloads that have similar performance to the target one on configurations where both have executed. The predictor then uses the recorded performance of relevant examples on the new platform to make its estimate.

Encoded Storage for Array Databases

During my graduate studies, I also began work on SciDB, an array-centric, open-source data management system [9, 1]. In this platform, all data objects are arrays, and their storage is optimized for analytical, spatial queries, such as k-nearest neighbors or downsampling an image. My work optimized the use of compression on multidimensional arrays to reduce their I/O requirements. I applied several compressors to the vertically partitioned, multidimensional chunks, which are SciDB's unit of storage and memory management. My framework selected the compression scheme once per attribute based on a randomized sample of its value distribution.

I also researched a vectorized execution model for sparse arrays in SciDB. This storage representation simplified the query's execution, reduced its I/O consumption, and made more efficient use of the CPU cache. Prior to my work, all sparse arrays in this database stored the positions of their cells as a list of coordinates. My approach carefully composed compression schemes such that the database executed its queries directly on compressed cell positions. This vectorized representation is available in the main release of SciDB.

Data Placement for Elastic Databases

In my post-doctoral studies, I first focused on assigning data to nodes in elastic databases. Such systems execute on a changing set of hardware resources dynamically provisioned to serve their storage and processing demands. Several factors influence the sharding of an elastic database. As the cluster expands and contracts, data placement should be incremental, minimizing the cost of each reconfiguration. The matching will be skew-aware, balancing the database's work to maximize its throughput. Lastly, partitioning is workload-sensitive, respecting data dependencies and cognizant of how queries access the underlying data sources.

In [6], I researched this challenge in SciDB. This platform has a no-overwrite data model, i.e., users only delete data when available space is exhausted. In this setting, storage space is the limiting factor. My work optimizes the partitioning of large arrays for incremental cluster expansion. I both extended well-known partitioning algorithms and created new ones. At each scale out operation, the partitioner receives n new nodes and reassigns data to balance the load. This partitioning is made scalable by an iterative approach to each expansion, in which the partitioner first identifies the most overburdened node and splits it. It repeats the process with each of the new additions, thus catering to skew and simplifying the reorganization.

I am also working on sharding for H-Store, a main memory, parallel, transactional database [10]. In this platform, CPU is the scarce resource, and thus redistributing the data means assigning an equal number of transactions per node rather than balancing storage. Our approach minimizes the cost of elasticity for a live system executing transactions by assigning each tuple to one of two storage tiers. This technique first identifies and isolates hot spots, distributing them over the cluster as its top level of data placement. The remaining tuples are handled separately using standard hash or range partitioning. In cases of more broadly distributed skew, our system uses statistical techniques to classify the data access patterns and selects an appropriate repartitioning algorithm, which adaptively adjusts the sharding.

Query Optimization for Array Databases

I am presently studying the optimization of skewed, massively parallelized joins for SciDB [4, 11]. Array databases with a shared-nothing architecture are often network bound for join operations because aligning the data is far more expensive than comparing the individual cells. To this end, I proposed a novel n-way shuffle algorithm, which plans the join operation in two discrete steps: data alignment and join execution. The shuffle is so named because it coordinates the exchange of data between n cluster nodes such that it minimizes the network bandwidth used. Its data alignment phase brings sparse portions of the array to their denser counterparts, allowing the join execution to begin more quickly. This technique resulted in a 3X speedup in our experiments on skewed data. The optimization easily generalizes to relational joins.

Future Research

I have investigated several projects that I believe have strong potential for future research. I would enjoy pursuing the first direction at a new university or research institution over the next 3–5 years. The second is a longer-term endeavor positioned for collaboration between academia and industry.

Query Optimization for Complex Analytics

Query optimizers generate execution plans consisting of directed acyclic graphs. The planner takes in a declarative query and translates it into a set of well-known operators, such as sorts and aggregates. This operator-level planning is not well-equipped for complex analytics, especially on the commonplace shared-nothing architecture. For example, it does not take into account skew in the distribution of queried data amongst nodes.

The language of complex analytics is different from relational querying and warrants a rethinking of how the optimizer plans and executes its work. Such queries often consist of operators that are (a) blocking and (b) not commutative. For example, if a user is executing a matrix multiplication on a transposed array, the order of these operations cannot be reversed without producing erroneous output. This appears to limit the benefits of conventional query optimization, such as the reordering the operators. It does, however, create new opportunities for skew-aware query planning. Although there has been some work on optimizing joins for skewed data distributions, in the general case there has been little research that deviates from the "bring query processing to the data" paradigm. In cases of severe skew, when most or all of the data selected on a cluster resides on very

few nodes, it may be more cost-effective to redistribute the data to achieve greater parallelism. This is especially relevant for math-intensive operations including Fast Fourier transform and singular value decomposition.

This fine-grained assignment of "slices" of each query operator is complicated in a shared-nothing architecture because the allocation is made at two levels: by host and its CPU cores. Careful consideration must be paid to have sufficient parallelism for fast query execution without exceeding the available memory on each node. This problem is especially challenging for clusters comprised of diverse, heterogeneous nodes.

Optimizing intra-operator parallelism for complex analytics at scale is too complex for conventional dynamic programming strategies. More sophisticated searches, such as simulated annealing and evolutionary algorithms, may result in faster query execution. Managing the trade-off between improved query plans and lower optimization time is also relevant to this research.

Personal Use of Big Data

Presently most data-driven applications, including epidemic tracking, supply chain management, and advertisement matching on web sites, are domain-specific and used by skilled experts only. There has been little work done to bring the advancements of this new "big data" tool set to everyday life. It is easy to imagine how many users would benefit from generic data-centric tools, which may guide personal decision making, aid in the collection and management of information, and be used to visualize topics of interest.

Data-driven decision making (DDDM) is the first application I will investigate. Everyone makes decisions in domains where they are not experts, and generalized DDDM tools will empower people to make more informed choices in their personal and public lives. Users may ask questions such as: "If I seek a degree at University X, what is my expected return on investment?" or "I am planning to vacation in Jamaica this summer; are there likely to be more sunny days in July or August?"

Presently, most questions of this kind are answered using heuristics, back-of-the-envelope calculations, or single-use applications such as a mortgage calculator. By tapping into real data and integrating the user's information with third-party sources, we can do better. For example, a driver who wants to forecast his car's depreciation rate will get a more accurate estimate by taking into account his commute, driving record, and typical auto maintenance schedule.

All of these questions are context-sensitive and rely on extrapolating from examples of similar circumstances. In [2], I proposed a query model for answering such questions. This vision paper outlines how users would supply a frame of reference for their decisions, identify relevant external data, and determine a decision model for their queries. This work poses several novel challenges in privacy preservation, data integration, query language development, and managing the distributed execution of decision queries.

Bringing scalable analytics to data mining and visualization is also of keen interest to me. Users are presently also awash in information that either affects or interests them. People regularly click through end user license agreements (EULAs) with hardly a glance at their contents and no intuition of how they compare to other contracts. The same is true for bills as they pass through Congress, mortgages, credit card agreements, and privacy policies, and countless other jargon-heavy documents.

Most data with which users interact is lengthy and unstructured. Finding the right techniques to automate document visualization presents several open research challenges, including the scalable parsing and classification of data and pairing it with the most intuitive representation–whether it is a graph, hierarchical tree, or other tool. I am particularly interested quantifying the contents of documents so that users can more directly gain insights from the underlying data.

A visualization I would like to pursue is a "nutritional facts" label for documents. It is analogous to the ones found on food containers. A person who seeks to understand a federal budget may bring up a tool that starts with the top-level view, containing only a list of each department and its percentage of the budget. This user could then drill down to understand the programs being funded by each organization, using pie charts and other appropriate figures at each step, and continue until they reach the individual items in the document. Such a generic document tool could train models for Internet privacy policies, prescribing data on pharmaceuticals, and a myriad of other domain-specific documents, enabling people to interact more directly with their data.

Both of these topics would benefit from an interdisciplinary approach, and I am eager to work with potential collaborators, especially in visualization and statistical machine learning. My background in scalable analytics is readily applicable to these increasingly data-rich domains.

References

[1] P. Cudre-Mauroux, H. Kimura, K. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik. A Demonstration of SciDB: A Science-Oriented DBMS. In *Proc. VLDB Endow.*, 2(2), pages 1534-1537, 2009.

[2] J. Duggan. The Case for Personal Data-Driven Decision Making. To appear in VLDB 2014.

[3] J. Duggan, U. Çetintemel, O. Papaemmanouil, and E. Upfal. Performance Prediction for Concurrent Database Workloads. In *SIGMOD*, pages 337-348, 2011.

[4] J. Duggan, O. Papaemmanouil, L. Battle, and M. Stonebraker. Skew-Aware Join Execution for Scientific Databases. Under preparation.

[5] J. Duggan, O. Papaemmanouil, U. Çetintemel, and E. Upfal. Contender: A Resource Modeling Approach for Concurrent Query Performance Prediction. In *EDBT* 2014.

[6] J. Duggan and M. Stonebraker. Incremental Elasticity for Array Databases. To appear in SIGMOD 2014.

[7] J. Duggan, Y. Chi, H. Hacigümüs, S. Zhu, and U. Çetintemel. Packing Light: Portable Workload Performance Prediction for the Cloud. In *ICDE Workshops*, pages 258-265, 2012.

[8] J. Rogers, O. Papaemmanouil, and U.Çetintemel. A Generic Auto-Provisioning Framework for Cloud Databases. In *ICDE Workshops*, pages 63-68, 2010.

[9] J. Rogers, R. Simakov, E. Soroush, P. Velikhov, M. Balazinska, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, S. Zdonik, A. Smirnov, K. Knizhnik, and P. Brown. Overview of SciDB: Large Scale Array Storage, Processing, and Analysis. In *SIGMOD*, pages 963-968, 2010.

[10] M. Serafini, E. Mansour, J. Duggan, R. Taft, A. Elmore, A. Aboulnaga, A. Pavlo, and M. Stonebraker. Incremental Data Placement for Main Memory Database Systems. Under preparation.

[11] M. Stonebraker, J. Duggan, L. Battle, and O. Papaemmanouil. SciDB DMBS Research at M.I.T. IEEE Data Eng. Bull. 36(4): 21-30 (2013).