RESEARCH STATEMENT

Raul Castro Fernandez (raulcf@csail.mit.edu)

Data has the potential to substantially improve many areas of our lives, from advancing scientific research to making people more productive to understanding society better. To achieve these goals we must exploit all data available to us: the hard truth of the era of big data is that current abstractions, algorithms and systems use only a small fraction of it. Uncovering the larger fraction that remains hidden is hard because most data is sequestered in silos. Unlike searching the web, there is not currently a solution for finding datasets across the many data sources that organizations use to store their data. As a consequence, teams know about a few datasets they own but miss the many opportunities to combine those datasets with others that will lead to more insights. When analysts need to answer questions requiring data from other parts of the organization, they spend more time finding relevant datasets than solving the actual problems at hand. Without efficient ways of discovering relevant data, the real value remains untapped and that means we are missing data's potential transformative effect.

I build high-performance and scalable systems to discover, prepare, and process data. For example, to find and combine datasets among thousands of databases and files, I have built Aurum, a data discovery system which summarizes large volumes of data into efficient data structures to scale, and implements a novel discovery interface to help analysts answer many different discovery questions. To understand how to combine structured data in database tables with unstructured data such as text from emails and PDF files, I am building Termite. Termite is a system that learns a vector space from both structured and unstructured data in a way that relates otherwise hard to relate data sources. Finally, I have designed and built systems to process large scale data with programming interfaces that allow users to write sophisticated algorithms, from machine learning applications to streaming and monitoring applications.

In my research, I like to identify problems that, if solved, would make some real user's life easier. Grounding the initial research questions in problems faced by real users helps me motivate the importance of the problem, and maximize the impact by making sure the final system solves the initial problem well. I enjoy bringing into practice techniques from other fields such as statistics and machine learning when they are appropriate, but I always strive for the simplest solution which solves the problem at hand; building systems is hard, and unnecessarily complex solutions often remain unimplemented. A key component of my research is to build practical systems to test hypotheses both empirically, in the lab, as well as in the wild through deployments alongside collaborators. As an example, Aurum is used by 5 companies and Termite is in the onboarding process with 2 others.

Current Research

Data Discovery

Modern organizations possess vast amounts of data, which is stored in many different data sources such as relational databases and data lakes. All this data has been produced by different teams and processes over time, so it is extremely heterogeneous. Today's analysts want to answer more and more sophisticated questions, many of which require accessing sources that span multiple systems and databases. For example, consider the problem of finding all personally identifiable information across an organization, which is necessary for companies that operate in Europe due to the new GDPR laws. Or suppose an analyst wants to combine some customer data from the product team with sales data from the marketing team to predict future sales of a new product. Today, solving these use cases is a time-consuming process that involves manually locating the data, cleaning it, and normalizing values so datasets can be combined together and used to solve the original problem. Each specific use case demands an immense investment of resources to create datasets that are not often reusable later on. A consequence of the heterogeneous and large volumes of data and the cost of preparing it is that most data remains in silos and is therefore inaccessible to analysts. More often than not, analysts spend more time finding relevant data than answering the question at hand: they suffer a *data discovery* problem.

I have built Aurum [1], a system to solve data discovery use cases such as the ones described above. The initial research questions I wanted to answer were: i) is it possible to abstract away the meaning of "relevance" so as to support multiple different discovery use cases? ii) if so, is it possible to build a practical system despite the volume and heterogeneity of existing data? To address the first question, I observed that for most discovery use cases relevancy can be defined in terms of relationships. For example, a dataset is relevant to a *duplicate-search* application if there is a similarity relationship between that dataset and its duplicates. Every discovery case can be expressed in terms of some arbitrary combination of different relationships, so in order to support many discovery use cases, it must be possible to use many types of relationships between datasets. Aurum supports many varied discovery use cases because it finds relationships of many types between the often hundreds of thousands of data sources it ingests. It then represents these relationships in a hypergraph, which users query with a "discovery API". Instead of building a system with a predefined concept of relevance (which would need to be re-engineered when the use case changes), I built Aurum so that users can define what is relevant for them. By letting users express their varied discovery queries, Aurum addresses many discovery use cases and can quickly adapt to new ones.

The second question is about scalability. To find all relationships between the many data sources involves $O(n^2)$ comparisons for each relationship of interest. Each comparison requires expensive access to the original data as well as verifying if the relationship exists. I adapted sketching techniques such as MinHash and LSH to compute similarity between data sources to find duplicates and complementary data. I also devised a technique to compute containment using LSH [12], which previously had been proven to be a hard problem. With this technique, Aurum can find primary-key/foreign-key relationships, which are necessary to combine datasets with each other. Sometimes data contains ambiguous terms, and that makes deciding what data is relevant hard. I adapted probabilistic models of language, e.g., word embeddings, so they could be used by Aurum to surface semantic relationships [2]. And for scenarios that require identifying highly structured data—data that follows a strict pattern such as dates, ISBN numbers, etc.—I proposed a technique to learn a regex-like structure in linear time [4].

By understanding multiple relationships between data and allowing users to query it efficiently, Aurum is the first data discovery system which supports solving a varied set of scenarios such as those outlined above. Despite being a research prototype, Aurum is deployed and used across more than 5 companies that span domains such as pharmaceuticals, banking, telecommunications, chemistry and computer storage.

Stateful Data Processing

Sometimes, analysts have access to the data they need to answer a question or probe a pattern. However, the large volume of data means they cannot execute their programs on their laptop due to lack of space and compute power, and adapting the programs to run in clusters requires expertise and time. In addition, the programs analysts want to run are increasingly more sophisticated. For example, they want to train machine learning models, build recommendation systems and perform all sorts of complex statistical analysis. Furthermore, many of these programs make explicit use of state, and platforms for cluster computing such as MapReduce and Spark are stateless. Stateless designs are good because they make it easier to exploit parallelism while tolerating common machine failures, but they can only support a restricted set of algorithms efficiently.

I proposed *stateful data processing*, a model that allows users to write sophisticated queries in a language similar to those they use on a daily basis and with explicit access to state, while still leveraging the parallelism and fault tolerance of clusters. I implemented the stateful data processing model in a new system, SEEP [10], which makes the following key contributions: i) Usability. SEEP shows that it is possible to use the control dataflow graph of imperative programming languages—which users find easy to use—to extract a *Stateful Dataflow Graph* (SDG), a dataflow model that represents access to state [9]. ii) Expressiveness. SEEP processes both data that is streaming in real time with low latency as well as large static data with high throughput. The key insight is that it is possible to grow a stream processing system to support batch processing by integrating a scheduler, but it is much harder to make a batch processing system process data with low-latency, as required by streaming applications. iii) Efficiency. Being able to write stateful code and execute it in hundreds of machines is only good if the system can leverage the compute resources available efficiently and tolerate failures. In SEEP, state is treated as a first class citizen in the system—rather than as a hack in application-code that users need to deal with. This enables scaling to hundreds of machines and tolerating failures efficiently.

SEEP became the base for many other research projects. We adapted it in Ako [6] to support the efficient execution of machine learning workloads such as kernel density estimation, logistic regression and various deep networks. We used it as the initial basis of SABER [7], a high-performance stream processing system that processes data at line-rate on a 10Gbps NIC. Ideas from SEEP have been used in Samza, a stream processing system used by LinkedIn among others; we described some of the use cases in Liquid [8].
Beyond SEEP. I am interested in understanding how different users use data and in building better systems to support those scenarios. One example is Quill [5], a system I helped build in collaboration with MSR. Quill is a distributed data processing system which runs entirely in the cloud. Users of Quill interact with the system through API calls in their favorite programming language. All the system aspects, from deployment to monitoring to provisioning machines, etc. are completely transparent and handled automatically by the system. Another example is Metadataflows [3], a new dataflow model which is optimized for executing entire families of dataflows which correspond to different configurations of the same job (think of finding the right parameter configuration for a machine learning workload). The gist of the system is that it understands the data access patterns, and by integrating resource provisioning with scheduling, it can skip computation when possible. Therefore, it helps accelerate query answering.

Future Research

My research goal is to build algorithms, abstractions and systems to facilitate data discovery, preparation and processing. I now describe a research direction I plan to pursue short-term as well as my research vision.

The Fabric of Data

In addition to the thousands of relational systems that organizations use to store their data, they also keep valuable information in emails, Slack conversations, PDFs and other unstructured data sources. All this data, which represents a large percentage of the data within organizations, often remains hidden and unexploited. When unstructured data must be used, different research communities have focused on first extracting some structure from the unstructured sources and then integrating that newly structured data in the databases. As data formats and use cases are always changing and evolving, these imperfect solutions must be continuously tuned for each specific use case. Most of the value of data remains untapped because of the high cost of tuning the above pipeline.

In a new line of work, called *fabric of data*, I want to build an intermediate representation for data which skips the above extract-and-integrate pipeline. The intermediate representation is a vector space in which points represent words, sentences and paragraphs from unstructured data, as well as rows, cells and columns from relational tables. Crucially, the distance between points in the vector space indicates the strength of some relationship of interest between the data those vectors represent. For example, by pointing to a row in a relational table that contains an empty cell, we can pull all the PDFs that are related to the row, which may help with finding the missing value, or help by uncovering valuable complementary information. The key challenge of this project is in building a vector space with the desired properties.

I have built an initial prototype, Termite [11], which combines data from multiple webpages, DBPedia and relational databases, and permits searching for relevant data by performing algebraic operations on the vector space. The road ahead is full of interesting research questions such as what is the best way of building the vector space and what kind of applications can be efficiently supported on top of it. The potential impact of this project is in increasing our understanding of data by enabling us to access it through a common data representation without having to parse and process the multiple heterogeneous systems and formats in which it is originally stored.

Research Vision: Data as an Asset

Most data is hard to use because it is difficult to understand how it came to be and how it has been used before. Without that context, data preparation is difficult. As a consequence, analysts spend large amounts of time understanding how to clean and transform it, as well as whether it must be combined with other data before it can be used. Without that context, it does not matter how many data privacy laws and regulations are put in place, it will be hard to enforce them because it will remain difficult to understand the many ways in which data is being used.

Until today, we have relied on external systems to keep track of metadata that can help with using the data. This approach has not succeeded for at least two reasons. First, because it depends on external systems having the ability to extract and manage the metadata, but without precise knowledge of what metadata is necessary, this is hard to implement. Second, because it depends on data users taking the initiative to document and annotate datasets as they are used. But users don't have incentives to do this: they want to get their job done, and not spend time documenting or explaining how they transformed a dataset to produce a report.

I propose an alternative philosophy on how to maintain and use data. In this vision, each dataset is seen as an asset, a first-class-citizen similar to applications and workers. Each dataset is capable of explaining its own purpose, because it has access to its entire life history. More in detail, these *existential datasets* I envision, contain the actual data in addition to all the queries that have been executed against them. Read-only queries are useful because they serve as documentation to people who want to use the data. Write-queries, such as cleaning, transformation operations and updates, are useful for people who need to prepare the data. In addition to its own context, existential data keeps track of its own indexes and statistics to enable fast query execution. I believe existential datasets justify the dissassembling of today's monolithic databases into distributed components which take responsibility for different aspects of data management. Data processing systems, for example, can use existential datasets to enable applications that we do not know how to solve today:

Data Sharing It is frustrating to know the precise dataset one needs to solve a challenging problem but be blocked because the owner is unsure whether sharing is "safe". Because anonymizing data is difficult, it does not guarantee the absence of data leaks, and the responsibility of potential problems falls in the data owner, the default solution is to not share data at all. This leads to the consequent loss of value. Even in cases in which a team effort is put into sharing a dataset safely, the many rounds of anonymization and transformations that the data suffered and its lack of context means that it is difficult to prepare it to use in downstream tasks.

Context is fundamental to understanding how to use data. With existential datasets, it is possible to envision cryptographic schemes in which both data and context are only visible to people and systems with the right privileges and hidden to the rest. It is therefore possible to restrict the kind of operations that can be run on the dataset, by allowing only certain types of transformations to its provenance, for example.
Data Usability Data systems can reason about the metadata contained in existential datasets to answer difficult questions that cannot be solved today without investing large time-consuming human efforts. When an analyst accesses a dataset for the first time, the system can recommend other datasets that are often jointly accessed, as well as queries that are often executed against this dataset. When a data preparator must transform the data for an application, end systems can explain how other analysts have transformed data in the past, and even provide the code for doing so. Finally, end systems can give a detailed view of the provenance of the datasets, because they have access to the provenance of each individual dataset. This is useful to make data use (and misuse) more transparent.
Data Market Existential datasets must be seen as an asset that can be placed in a market that owns it and can reason holistically about issues such as sharing, quality and insurance. In such a market, data can be traded by data publishers, subscribers and stewards—people who own the data now, people who want to use the data, and people who know how to clean it and prepare it but do not need to use it. A market can help set the right incentives to keep datasets with the highest quality and usability for the interested parties.

Summary

It is the most exciting time to work on data because data will be at the core of the next technological advances. My goal is to advance the frontier of what can be done with data, by understanding the most pressing challenges across disciplines, building systems to solve them, and testing systems both in the lab and in the wild alongside collaborators.

References

[1] *Aurum: A Data Discovery System*, Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, Michael Stonebraker. ICDE'18

[2] *Seeping Semantics: Linking Datasets using Word Embeddings for Data Discovery*, Raul Castro Fernandez, Essam Mansour, Abdulhakim Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, Nan Tang. ICDE'18

[3] *Meta-Dataflows: Efficient Exploratory Dataflow Jobs*, Raul Castro Fernandez, William Culhane, Pijika Watcharapichat, Matthias Weidlich, Victoria Lopez Morales, Peter Pietzuch. SIGMOD'18

[4] *Extracting Syntactical Patterns from Databases*, Andrew Ilyas, Joana M. F. da Trindade, Raul Castro Fernandez, Samuel Madden. ICDE'18

[5] *Quill: Efficient, Transferable, and Rich Analytics at Scale*, Badrish Chandramouli, Raul Castro Fernandez, Jonathan Goldstein, Ahmed Eldawy, Abdul Quamar. VLDB'17

[6] *Ako: Decentralised Deep Learning with Partial Gradient Exchange*, Pijika Watcharapichat, Victoria Lopez Morales, Raul Castro Fernandez, Peter Pietzuch. SOCC'16

[7] *SABER: Window-Based Hybrid Stream Processing for Heterogeneous Architectures*, Alexandros Koliousis, Matthias Weidlich, Raul Castro Fernandez, Paolo Costa, Alexander Wolf, Peter Pietzuch. SIGMOD'16

[8] *Liquid: Unifying Nearline and Offline Big Data Integration*, Raul Castro Fernandez, Peter Pietzuch, Joel Koshy, Jay Kreps, Dong Lin, Neha Narkhede, Jun Rao, Chris Riccomini, Guozhang Wang. CIDR'15

[9] *Making State Explicit for Imperative Big Data Processing*, Raul Castro Fernandez, Matteo Migliavacca, Evangelia Kalyvianaki and Peter Pietzuch. USENIX ATC'14

[10] *Integrating Scale Out and Fault Tolerance in Stream Processing using Operator State Management*, Raul Castro Fernandez, Matteo Migliavacca, Evangelia Kalyvianaki and Peter Pietzuch. SIGMOD'13

[11] *Termite: A System for Tunneling Through Heterogeneous Data*, Raul Castro Fernandez, Samuel Madden. Under submission.

[12] *Lazo: A Cardinality-Based Method for Coupled Estimation of Jaccard Similarity and Containment*, Raul Castro Fernandez, Jisoo Min, Demitri Nava, Sam Madden. Under Submission