Research Statement

Yanif Ahmad, Computer Science Department, Cornell University.

Research Methodology

The body of work I have pursued falls under the umbrella topic of data stream processing, in centralized and distributed fashion. Data stream processing addresses the question of how to perform long-running relational database style processing on high-volume, sequentially observed data. Data stream processing has been one of the standard bearers for an argument against the architectural design strategy of major database vendors, against "one size fits all" [15]. The argument claims that commercial database systems have been found lacking in many aspects because they have been designed with overly general and abstract foundations, resulting in mismatches between application requirements and the features and performance provided by data management tools. Such issues are not limited to streaming applications – scientists find they are lacking the right data models and programming primitives, and the ability to incorporate domain expertise, while large web companies and financial institutions find data management tools fail to live up to the promises of declarative languages, scalability and robustness when query optimizers cannot transform declarative queries to match the performance of hand-written code, and parallel and distributed databases often adopt overly restrictive consistency models, that cripple their ability to scale.

Stream processing systems overcome these application mismatches with a from-scratch redesign of the core query processing engine and database architecture. This research statement outlines my work on data stream processing, and how my experiences have shaped the principles and methodology of my research approach: a vertical approach to build high-performance, usable tools, by questioning classical design principles and reflecting on hardware and software trends, and by being user-facing. My goals are to build effective tools by through two guidelines, picking problems where applications cannot be suitably addressed by existing DBMS, and providing an immediately effective solution. The first point reflects my desire to build data management tools, as opposed to databases, and requires rethinking computational models, declarative languages and system architectures. The second states that performance and scalability, and ease of use, cannot be dealt with separately – effective tools must provide both. I have found the approach of designing systems from the ground up, from semantics and data and query representations, to application-level requirements and usage models, a key enabler in building tools that provide both game-changing performance and functionality. Equally as important, I find that building systems provides many natural opportunities to involve collaborators, including students at all levels, and external researchers and industrial contacts as both users and for further research.

Data Stream and Continuous Query Processing

Data stream processing research enjoyed a revival in the first half of this decade, with much of the work focused on system design, and temporal semantics aspects of such tools, before progressing to algorithmic issues and the distributed setting, and commercialization of the research tools. The motivating use cases for stream processing included financial applications, network monitoring, sensor data processing and environmental monitoring, and massively multiplayer games [6]. The fundamental challenges and novelties of data stream processing [5] include the focus on continuous queries – queries whose answers must be computed given a data stream, an infinite sequence of tuples, as input. Continuous queries are the persistent entities in stream processing engines, as opposed to the data in classical database engines. Handling such queries requires bounding the data processed, typically through time or frequency windows, such as the last 10 minutes of data, or 100 tuples. Furthermore the high-throughput and low-latency requirements of stream applications motivated the use of one-pass, on-the-fly, main-memory database engines. Initial work on prototypes identified the need for handling high-volume inputs as a key requirement, inspiring the core directions of my dissertation work, that of model-based, and distributed, stream processing in SAND, Borealis and Pulse [3, 1, 11, 2]. My latest endeavors in the DBToaster compiler [9] [DBToaster] have begun to challenge the design principles of these early stream processors, arguing for general incremental processing of relational algebra, and for compilation of such incrementally processed queries to low-level code to aggressively execute queries.

Compiling continuous queries: rethinking incremental query processing.

The DBToaster project [9] investigates the design of query processors for continuous queries, rethinking the role of query compilation given the long-running static nature of such queries. The focus here is on aggressively performing incremental processing of general relational queries. In contrast, existing stream processors fully re-evaluate interpreted query plans, and

DBMS perform an extremely limited form of incremental evaluation with interpreted view maintenance query plans. DBToaster incrementally maintains query results on arbitrary sequences of updates for SQL queries, the same data and query model as views in today's DBMS, but compiles the entire query directly to C++ code. This data stream model enables DBToaster to be used for a wide range of applications, for example automated trading on order book data, as found in the majority of electronic exchanges such as NASDAQ, when compared to more restrictive stream models in existing stream processors.

DBToaster uses a novel recursive query compilation technique to produce tuple processing functions for each input relation – functions that both compute query results and maintain auxiliary data structures to capture query state for work performed on tuples from other relations. DBToaster is based on a simplified form of the domain relational calculus known as the map calculus, wherein calculus formulae can be easily represented as map data structures. DBToaster exploits updates to individual relations to simplify calculus formulae, factoring aggregates and eliminating variables. The maps we maintain during query execution are chosen during the compilation process. DBToaster produces extremely lightweight code consisting of simple arithmetic expressions on map entries, resulting in game-changing query processing that provides at least one to two orders of magnitudes speedup over state-of-the-art view maintenance algorithms and stream processing engines in both algorithmic trading and online data warehousing scenarios. DBToaster illustrates how far we have yet to go, even with novel architectures such as stream processing – there is still much room for understanding the foundations of, and being creative in data management systems.

Model-based stream processing: data management without the data.

While the database community has contributed the relational model and declarative querying to stream processing, data streams themselves have long been acquired and analyzed in the physical sciences, engineering, and industry, for example in atmospheric sciences, telecommunications, and military applications. Much of the work here involves numerical analysis of the data stream, including both mathematical modelling of the data stream, validating the model, and evaluating the model for interpolation, extrapolation and optimization (extremal points). In the broader context, models represent a vital challenge for database research, databases are poor at incorporating domain-specific knowledge, unless such knowledge can be directly represented as tuples.

My dissertation presents Pulse [4, 11, 2], a framework that looks at bridging the gap between the relational and non-relational functionality for (mathematical) model-driven processing, using models for compression and approximation, as well as for predictive querying. Pulse uses piecewise polynomials to represent streaming data, and processes queries directly on polynomial segments by making use of novel query processing techniques, where queries are transformed into systems of equations and solved as the processing primitive. Pulse uses discrete input tuples, such as sensor readings, to fit piecewise polynomials on-the-fly, processes queries, and then supports discretization of query results to enable interpolation and extrapolation of results. Pulse can process the full positive relational algebra queries in addition to standard aggregates such as sum, average, min and max. Pulse tackles the semantic challenges in representing data in symbolic polynomial form, particularly the inherent discrepancies and errors between polynomial representations and the raw data itself. Pulse provides users the ability to request error bounds on query results, that is to ask for results from processing polynomials to be within a bounded distance from processing discrete tuples, whenever tuples are available. Considering errors has revealed a large design space for handling noise and outliers while providing error guarantees.

Pulse's processing strategy of solving equation systems is one of its core novelties and raises many questions how to best design an entire database system around such a query processor . In my dissertation, I implemented Pulse on top of the Borealis stream processing engine, and presented an initial solution to the broader challenge of designing architectural abstractions for model-based stream processing and error management. My dissertation also investigated statistics estimation and query optimization for equation solving. Classical databases use histograms to capture statistics on data distributions to guide a query optimizer's decisions, however, histograms cannot trivially be applied to polynomials. Pulse's statistics estimator uses multidimensional histograms on a parameter space of polynomial segments, and defines geometry operations on histogram bins (hypercubes) to perform selectivity estimation. Pulse's optimizer uses these selectivity estimates in a novel cost model that jointly addresses operator reordering, multi-query optimization, as well as the potential for inaccurate statistics. By considering both reordering and multi-query optimization, Pulse is able to exploit a wide range of plan configurations when faced with complex selectivity scenarios.

Scaling up and out: handling the data deluge.

In addition to designing novel query processors from both the performance and functionality perspectives, I have a long-standing interest in the question of scalable query processing in the distributed setting, including both cluster and cloud environments, as well as the wide-area context. The major themes I have worked on for scalable processing can be broadly viewed as: i) infrastructure and mechanisms [1, 10]; ii) parallelization, distribution and replication [Cumulus] [7]; iii) deployment and in-network processing [3, 8]; iv) network programmability and extensibility [13, 14].

Borealis: building core distributed stream processing mechanisms [1]. Borealis is to this date, the only open source distributed stream processing system available, and has been downloaded by more than 200 users. My role in this project included designing

the core operator migration and query adaptation mechanisms for use in various optimizers such as a distributed load balancer, and load shedder, in addition to designing the distributed system infrastructure layer (such as a distributed catalog) around the Aurora stream processing engine, and the in-memory table operators to support data sharing between operators and handle persistent state. My role as one of the primary architects of the system culminated in being the lead developer for our award-winning demonstration on the use of stream processing in a multiplayer game, where we used Borealis to maintain a consistent, persistent game world. This project provided one of the primary experiences in my research career, that of working in a multi-institution collaboration with students from MIT and Brandeis, and developing a codebase as a group. It indicated the immense benefits of student-driven mentoring, especially in terms of exchanging development experiences, can improve the overall level of knowledge in the group.

SenseWeb: communication-efficient spatio-temporal indexing for a sensor web portal. During my internship at Microsoft Research, I collaborated with Suman Nath on the Senseweb project which looked at query processing techniques for a sensor web portal [10]. A sensor web portal consists of community generated sensor data, and acts as a rendezvous point for sensor network administrators to publish sensor streams without having to open up their sensornets for external access. A sensor portal operates in an on-demand manner, where sensornets are probed for data as queries arrive into the system. We developed COLR-Tree, an abstraction layer for a sensor portal that jointly addresses data collection and query processing challenges. COLR-Tree supports spatio-temporal aggregate queries and is extremely efficient in terms of communication overhead through its ability to couple an index structure with a window-based cache for aggressive reuse of sensor readings, and to leverage a sampling technique to probe subsets of sensors, that provides guarantees on both the number of sensors probed and the sensing workload uniformity.

Cumulus: massively parallelizable incremental processing for online main-memory data warehousing. In joint work with Oliver Kennedy and Christoph Koch, the Cumulus system presents a massively parallel query processor for online main-memory analytical query processing in a cloud computing environment, such as Amazon's EC2 service. Cumulus leverages the structure of the code produced for incremental computation from the DBToaster project, which, for non-nested queries, consists of simple arithmetic expressions of map lookups and map updates, potentially with loops. Each expression can be thought of as a message, leading to our representation of queries as a message passing program to perform incremental view maintenance of SQL aggregate queries. Furthermore, map datastructures have been studied extensively in the cloud computing community – they are simply the key-value stores used in many large-scale web companies today. Thus Cumulus presents a SQL aggregate query processor based on a simpler lightweight key-value store than classical parallel databases, and provides a highly scalable system in terms of bandwidth, processor and memory utilization.

SAND: intelligent deployment and scaling for wide-area stream processing. The SAND project [3] investigated an operator placement algorithm for query plans, asking the question of how to best perform in-network processing in a wide-area network to minimize both network utilization and query performance. The project included the development of a distributed middleware layer and protocol to facilitate black-box placement provided simple selectivity statistics from an underlying query processor. The project used an early prototype of Borealis to deploy network monitoring queries on PlanetLab [8], demonstrating the gains of in-network processing on overlays.

XPORT: extensible publish-subscribe systems. In the XPORT project [13, 14], I was part of a team with Olga Papaemmanouil, Uğur Çetintemel and John Jannotti, that proposed a publish-subscribe system whose key focus was on providing extensible dissemination tree construction algorithms, in terms of the optimization objective of the resulting overlay. XPORT supports a novel, customizable two-layer aggregation model over network nodes and paths to define this objective, and is capable of specifying many common objective functions such as overlays that minimize the maximum path latency on any root-to-leaf path, or maximize the average bandwidth available between any parent and child in the dissemination tree. XPORT was also deployed and evaluated on the PlanetLab wide-area testbed.

Research Agenda

My research approach involves drawing inspiration from application domains that are ill-suited by existing data management tools, and while rooted in database research, branches into both declarative language design and distributed systems to solve practical challenges faced in building effective systems. I plan to pursue both *core streaming applications* in automated trading and scientific computing, where my work has revealed many pressing issues to be solved, and *untapped applications* in services science and sustainable computing, where I hope to identify abstractions common to core applications and adapt streaming techniques to suit the context. These topics are clearly cross-cutting, and it is my firm belief this research will immensely benefit from collaborations within computer science, from programming languages, distributed systems and sensor networks, and externally, for example with mathematical finance, operations research, and the many disciplines related to sustainability. I believe the diversity of topics I have pursued in query processing foundations, declarative language design and systems building makes me a strong candidate to cover these challenges and bridge research areas, and that in particular, my drive to build tools will provide a tangible, demonstrable focal point for collaborators.

Financial and scientific computing.

Despite making initial inroads into addressing the needs of financial and scientific computing, stream processors are lacking in their abilities to perform and reason about mathematical computing, and to be used in existing tools for these applications.

Model expressiveness. My work on the Pulse framework has only scratched the surface of designing a model-based stream processing engine [4]. While there is undoubtedly additional functionality and expressiveness that could be considered in terms of queries, the key challenge I intend to investigate in the near-term is model expressiveness, to understand the classes of models that can be represented and queried, such as differential equations, time series and frequency models as found in signal processing, in addition to the underlying system abstractions that enable relational query processing directly on these models. Investigating declarative specification of such models can be directly applied to finance applications, for example the famous option pricing model, the Black-Scholes model, is a second-order differential equation, while stock prices and volatility are often represented with autoregressive models such as ARIMA and GARCH. Another key question is the issue of model interchange, that is how do we process queries over multiple models simultaneously, for example processing a join where one input is represented as a polynomial, and the other as a time series?

Combining symbolic and numerical data processing. The work on Pulse has also highlighted the concept that today's database systems adopt a finite, discrete computational model, despite the existence of promising work such as constraint databases [12] which are capable of representing infinite sets, and producing query results without any need for discretization of the data. Coupled with related work on computer algebra systems, I plan to investigate the viability of symbolic mathematical and query processing, with the objective of taking a practical, application-driven approach to problem, as motivated by finance and scientific applications, where such symbolic processing is abound. In particular, symbolic techniques trade off restrictions in assumptions for computational performance, and one significant challenge would be to design numerical processors capable of exploiting such restrictions, and interacting with and conveying the resulting semantics to users. Additionally it remains unclear how to best utilize both symbolic arithmetic simplifications with relational calculus or algebra simplifications. Furthermore the DBToaster project has illustrated the benefits of performing extremely aggressive incremental processing, and the same concept of incremental processing occurs frequently in many numerical recipes such as with matrix operations, particularly sparse matrices, where the sizes of matrix problems often encourages reuse of partial prior results wherever possible, or for using coarse-grained approximations as seeders for fine-grained computation.

Embedded stream processing. These core applications also exhibit the property that data management tools need to be both integrated into external tools, and leverage highly tuned algorithms and functionality present in external tools. The latter has been well-studied in supporting user-defined functions and building extensible databases, however, the former challenge is not well understood in the context of stream processing. Classical databases are most commonly accessed through OBDC or JDBC interfaces, but we are also witnessing the rise of embedded language usage through precompilers and language extensions such as LINQ, and embedded SQL. However, stream processing employs push-based continuous queries as opposed to pull-based one-time queries, and thus far, have required full control over the event processing loop. It is clearly undesirable to develop entire financial and scientific applications within a data stream processor, for example high-performance messaging functionality or data visualization, but there has been little investigation into how to best embed streams into host languages. The natural approach appears to be to use asynchronous programming techniques such as callbacks or futures. These challenges are exacerbated when factoring in concurrency aspects to both the host language and the underlying query processing.

Services science and sustainable computing.

In planning a longer-term agenda and vision, my primary desire is to find challenging topics with significant potential for impact and outreach. I view the question of how to be able to draw future generations of researchers into computer science as an extremely important issue, and consider the following applications as ideal sources of problems, and motivators for conveying the importance and essence of data management.

Combining data processing and (combinatorial) search. Many large-scale industrial organizations have become increasingly interested in developing principled methods of providing and consuming services in today's economy given the service sector is fast replacing agricultural or manufacturing sectors. Services science is often used as an umbrella for a range of disciplines, including computer science and operations research. Indeed the field of combinatorial optimization has long-studied the intersections between the two areas, with declarative mathematical and constraint programming languages such as ILOG's CPLEX and OPL widely used as optimization-based decision support systems. These systems are often loosely coupled with database tools, where the queries on the relational database system are used to determine parameters to the optimization problem. I believe there are a number of opportunities to integrate the functionality provided by the separate systems, essentially combining both tuple-oriented processing, and search over variable domains for values satisfying a query, as illustrated by related work on constraint databases and to a lesser degree by Pulse. For example, optimization problems are often involve a minimum or maximum value of an objective function defined on constants and variables, subject to constraints. Query processing would simply enumerate the objective function, and find the minimal value from a finite input database, while optimization-based approaches prune

variable domains for the answer. Alternative computational strategies could mix and match approaches, partially using the finite input database when the problem structure does not admit rapid pruning. I plan to study novel processing strategies in online optimization problems, where parameters change rapidly, for example as with vehicle routing for transportation fleets or auctions for advertising.

Computational tools for everyday sustainability. Sustainable *computing* thus far has primarily referred to the body of work investigating the energy footprint and lifetimes of our computing platforms, ranging from data center efficiency, to the wireless sensor networks monitoring our environments, and for example their radio communication efficiency. I envision a broader use for the term, one describing the design of a computational platform underpinning a sustainable, low carbon economy – that is a platform consisting of a multitude of diverse data sources for physical measurement, the computation of carbon emissions from measurements, prior to analysis across a large number of geographical, and categorical factors, and the use of provenance to effect a carbon tax on the cause of the emissions. A simple example of a query is to answer "what is the carbon footprint of manufacturing, delivering and selling a can of Coke?". Answering this requires information regarding the manufacturing process, as well as the supply-chain and logistics behind the product, in addition to models of the carbon emissions associated with each physical action. While the realization of any systems to achieve this may be a long way off, my goal here is to motivate research into practical systems facing data integration, numerical and symbolic computation, and continuous, fine-grained, analytical processing. I believe such an application may represent the dominant use-case of large-scale numerical stream processing to come, and will undoubtedly require collaborations across both industry and other scientific disciplines.

References

[1] D. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. The Design of the Borealis Stream Processing Engine. In *Proc. of the Second Biennial Conference on Innovative Data Systems Research (CIDR'05)*, Jan. 2005.

[2] Y. Ahmad. *Pulse: Database Support for Efficient Query Processing Of Temporal Polynomial Models*. PhD thesis, Brown University, Department of Computer Science, 2009.

[3] Y. Ahmad and U. Çetintemel. Network-Aware Query Processing for Distributed Stream-Based Applications. In *Proc. of the 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 456–467, Sept. 2004.

[4] Y. Ahmad and U. Çetintemel. Declarative temporal data models for sensor-driven query processing. In *Proc. of the Fourth International Workshop on Data Management for Sensor Networks (DMSN'07)*, Sept. 2007.

[5] Y. Ahmad and U. Çetintemel. Data stream management architectures and prototypes. In *Encyclopedia of Database Systems*, pages 639–643. 2009.

[6] Y. Ahmad and U. Çetintemel. Streaming applications. In *Encyclopedia of Database Systems*, pages 2847–2848. 2009.

[7] Y. Ahmad, U. Çetintemel, J. Jannotti, and A. Zgolinski. Locality Aware Networked Join Evaluation. In *Proc. of the 1st International Workshop on Networking Meets Databases (NetDB'05)*, Apr. 2005.

[8] Y. Ahmad, U. Çetintemel, J. Jannotti, A. Zgolinski, and S. B. Zdonik. Network awareness in internet-scale stream processing. *IEEE Data Eng. Bull.*, 28(1):63–69, 2005.

[9] Y. Ahmad and C. Koch. DBToaster: A SQL compiler for high-performance delta processing in main-memory databases. *PVLDB*, 2(2):1566–1569, 2009.

[10] Y. Ahmad and S. Nath. COLR-Tree: Communication efficient spatio-temporal index for a sensor data web portal. In *Proc. of the 24th International Conference on Data Engineering (ICDE'08)*, pages 784–793, Apr. 2008.

[11] Y. Ahmad, O. Papaemmanouil, U. Çetintemel, and J. Rogers. Simultaneous equation systems for query processing on continuous-time data streams. In *Proc. of the 24th International Conference on Data Engineering (ICDE'08)*, pages 666–675, Apr. 2008.

[12] G. M. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases*. Springer, 2000.

[13] O. Papaemmanouil, Y. Ahmad, U. Çetintemel, J. Jannotti, and Y. Yildirim. Extensible optimization in overlay dissemination trees. In *Proc. of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 611–622, June 2006.

[14] O. Papaemmanouil, Y. Ahmad, U. Çetintemel, J. Jannotti, and Y. Yildirim. XPORT: extensible profile-driven overlay routing trees. In *Proc. of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 769–771, June 2006.

[15] M. Stonebraker and U. Çetintemel. "One Size Fits All": An idea whose time has come and gone. http://www.cs.brown.edu/ ugur/fits_all.pdf (keynote). In *ICDE*, pages 2–11, 2005.