# Toward Simulation-Based Optimization in Data Stream Management Systems

Christoph Heinz, Jürgen Krämer, Tobias Riemenschneider, Bernhard Seeger
Department of Mathematics and Computer Science
University of Marburg, Germany
{heinzch, kraemerj, tobys, seeger}@mathematik.uni-marburg.de

***Abstract*— Our demonstration introduces a novel system architecture which massively facilitates optimization in *data stream management systems* (DSMS). The basic idea is to decouple optimization from the operative system by means of a secondary optimization system, which bears the burden of determining new query plans. Within the secondary system, which typically runs on a separate machine, we utilize suitable statistical models of the original data streams to simulate them. As the simulation can run at much faster rates, we are able to examine and assess new query plans in a shorter period of time without running the risk of deteriorating the original plan; we only migrate practically approved plans into the operative system. In our demonstration, we will present our prototypical implementation of this optimization architecture. We will demonstrate the interaction between primary and secondary system as well as the key features of the whole optimization process.**

## I. Introduction

Market analysis, production control as well as traffic management are only but a few examples of *continuous query* applications, which are characterized by the volatility of the processed streams and the long-running nature of the corresponding queries. Due to the growing importance of these applications, the need for dedicated DSMS has come to the fore in recent years. Among the components of a DSMS, the query optimizer has turned out to be a vital one. Its main objective is to determine high quality query plans that keep pace with changes in stream or system conditions. However, the huge number of possible query plans combined with changing stream characteristics severely impedes achieving this objective, e. g., outdated statistics may lead to an even poorer plan. For that reason, a suitable cost model which assesses the query plans suggested during plan enumeration is crucial. However, only a few cost models have been proposed yet in the context of continuous queries [1], [2]. Furthermore, none of them covers the complete set of operations required in streaming environments. Generally, it is difficult to develop cost models that capture arbitrary stream characteristics, e.g. temporal correlations, with sufficient accuracy. Additionally, estimations based on complex cost models often require expensive calculations.

On account of these deficiencies, we pursue a novel approach that substitutes traditional cost-based optimization for *simulation-based optimization*. More precisely, we simulate the evaluation of a query plan with synthetic data streams generated from statistical models and simply measure the runtime costs, rather than estimating them by means of a cost model. For the simulation, we utilize lightweight statistical models of the original streams, which keep pace with the current stream characteristics. As the simulation can run at much higher rates, we can significantly accelerate the enumeration of possible plans. It is worth mentioning that the simulation and the plan generation can either run on the same machine as the operative system or on a separate one. If it runs on the same machine, a sufficient amount of system resources must be reserved for the simulation and optimization process. For the sake of simplicity, we distinguish between a primary system, which runs the current query plan, and a secondary system, which runs the simulation and determines a new query plan.

## II. System Overview

We have implemented this novel optimizer architecture on top of our data stream infrastructure PIPES [3], [4], which is an integral part of our Java library XXL [5]. For illustration purposes, Figure 1 provides an overview of the general architecture, its essential modules, and their interaction.

### A. Primary System

The key component of the primary system is the **query executor**. It manages the entirety of continuous queries in the running system, i. e., it receives new queries, combines them with already registered queries, and removes outdated queries. An important aspect in this context is the underlying *continuous query algebra* (CQ algebra), which defines the semantics of the available stream operators. By means of a sound CQ algebra, queries can be posed and combined in a single query graph. PIPES provides an extensive operator algebra with a precisely defined semantics [6], which relies on stream elements in the form of a tuple augmented with temporal information. To pose a query, we provide an SQL parser that translates queries expressed in an SQL dialect, SQL-1992 with a subset of the window constructs known from SQL:2003, into a logical plan.

The **query monitor** continuously observes the query graph and collects metadata describing the involved queries. Static metadata comprises general information about the query graph like schemas, whereas dynamic metadata represents varying runtime statistics, e. g., input and output rates or operator memory allocation. As the computational cost for monitoring metadata of a query graph is not negligible, we monitor them

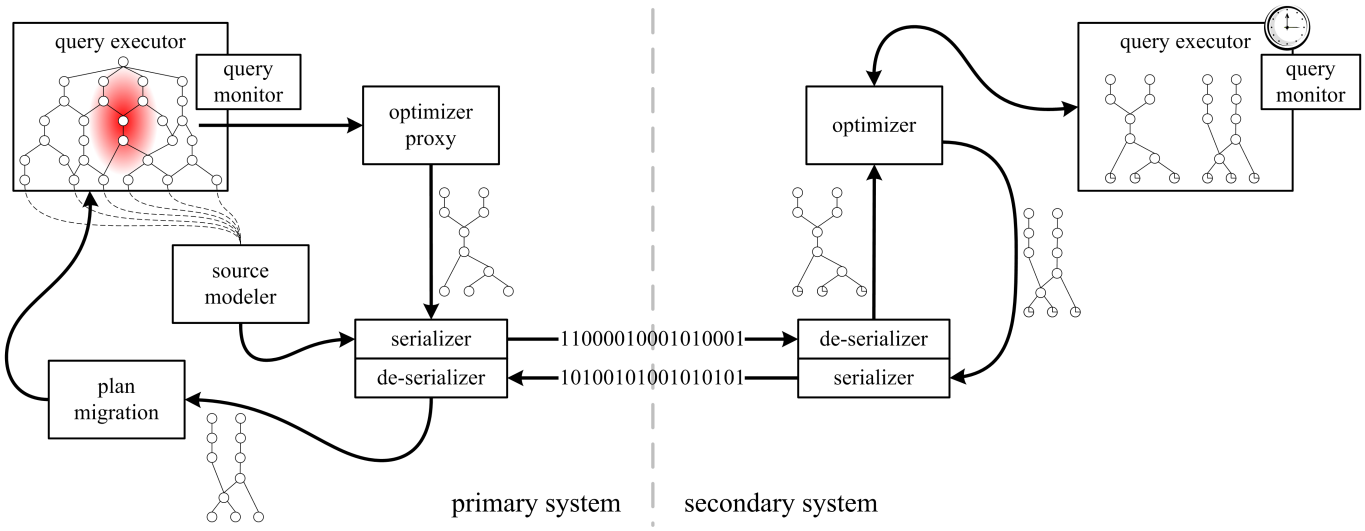


Fig. 1. Architectural overview and module interaction

on demand instead of monitoring all available metadata for every operator in the graph. More precisely, our query monitor relies on a publish/subscribe mechanism to collect specific metadata of an operator only when necessary [7].

At the heart of our query simulation approach is a novel DSMS module termed **source modeller**. During runtime, this module continuously maintains a meaningful statistical model of each data source in the query graph at low computational cost. By means of these models, the stream of each data source can be simulated at an arbitrary rate. The statistical model of a data source consists of the value distribution of the stream elements and the distribution of the inter-arrival time between successive stream elements. More precisely, we estimate the associated *probability density functions* (pdf) which uniquely describe the distributions. In [8], [9], we presented two sophisticated approaches, one based on kernels and the other one on wavelets, for continuously estimating the pdf of a data stream in a nonparametric manner, given only limited resources. In case the data exhibits temporal dependencies or cyclic variations like temperature throughout a day, the source modeller aims to capture them automatically and learns a separate model for each cycle, e. g., for each hour of the day. By observing the developing of the pdf, we can also detect changes in the underlying distribution, which indicate changing stream characteristics.

The **query optimizer** employed in the primary system is only a service proxy for the optimizer of the secondary system. It is responsible for determining *when to optimize*, e. g., in case the query executor receives new queries or the query monitor reports significant changes in the observed metadata. In this case, the optimizer proxy utilizes the **serializer** component to transfer the corresponding query subgraph along with the current statistical models for optimization purposes to the secondary system.

The **plan migration** module located on the primary system is triggered by the secondary system. When it receives an optimized query plan from the **de-serializer**, it has to *migrate* the currently executed plan into the new, optimized one. This *dynamic plan migration* [10], [11] determines the point in time, when the new plan produces exactly the same results as the old one. Then, the new plan completely substitutes the old one, which can be removed afterward. During substitution, the plan migration module has to prevent the production of duplicates and the loss of intermediate results. For that purpose, it determines a suitable *split time*. The old plan produces the query results before and the new plan after the split time. The migration process is finished if all input streams have reached the split time [10].

### B. Secondary System

The secondary system serves as the optimization platform for the primary one. The core of the secondary system consists of the **query optimizer** which receives a query plan along with statistical models of the plan's data sources from the **de-serializer**. It generates a set of new, equivalent query plans by applying transformation rules from the temporal relational algebra, which we adapted to our stream algebra [12], [6]. The rule-based query optimizer used in our implementation divides its transformation rules into groups according to common optimization objectives. During plan enumeration, these groups are systematically applied to the query in order to prevent cyclic application of opposed rules. To the best of our knowledge, such an extensive and operational optimization framework has not been demonstrated in the streaming context before.

Thereafter, the optimizer passes each of these equivalent query plans to the **query executor**, which runs them at accelerated speed with simulated streams generated from the corresponding statistical models. More precisely, we use the estimated pdf of each data source to generate synthetic stream elements. For that purpose, we utilize *rejection sampling*, a technique to draw elements from a distribution described by its pdf. Recall that we maintain two statistical models of each data

source, one for the values and one for the inter-arrival time. In order to generate a synthetic stream element, we simply have to draw an element from the value distribution and set its timestamp as the timestamp of its successor plus the new inter-arrival time drawn from the time distribution.

As the simulation does not have to wait for the arrival of real stream elements, we can assess the efficiency of a query plan within a short period of time. During simulation, the **query monitor** of the secondary system continuously collects relevant runtime statistics. The optimizer utilizes these statistics to determine the new optimized query plan, which is finally transmitted to the primary system.

### C. Additional Benefits

The general architecture to be demonstrated can easily be adapted to tackle other system-related tasks in a DSMS. For example, it can serve as a testbed for different scheduling or resource allocation strategies. Furthermore, the robustness of a DSMS and its registered queries can be tested on the secondary system without endangering the stability of the primary one. Last but not least, the separation of our DSMS into a primary and a secondary system and the option to run them on different machines prepares the ground for adaptive, distributed DSMS.

## III. Demonstration Proposal

The objective of this demonstration is twofold. First, we want to demonstrate our novel optimization architecture, its components, and their interaction. Second, we want to convey a general impression of our stream processing infrastructure PIPES and its key features.

### A. SQL Parser

The SQL parser provides an intuitive entry point to the functionality of a DSMS. By using a declarative programming language, users only have to specify the desired results instead of implementing how the result has to be calculated. Therefore, naïve users as well as experts are able to pose queries to the DSMS without having specific knowledge about implementation details. In our demonstration, we will introduce our SQL parser as an integral part of the common graphical user interface (GUI) shown in Figure 2. We will also illustrate why our query language is compatible with SQL:2003.

### B. Metadata Monitoring

Dynamic metadata plays an important role in our optimization architecture. On the one hand, the primary system utilizes dynamic metadata such as current input and output rates or operator memory allocation for determining *when to re-optimize* and *which subquery to re-optimize*. On the other hand, the secondary system utilizes dynamic metadata to determine the measured cost of a simulated query. Our sophisticated publish/subscribe mechanism for dynamic metadata, which is part of our demonstration, renders these tasks possible. We will present a GUI (Figure 2) that allows a user to flexibly compose and manage dynamic metadata at runtime.

### C. Statistical Modeling

Our approach to simulation-based optimization uses statistical models as major ingredient. In order to enable the simulation of data sources on the secondary system, their associated statistical models have to be maintained and forwarded from the primary to the secondary system. Furthermore, the models can be used to detect changes in the characteristics of a data source, which is crucial for identifying re-optimization potential. For that reason, we will make use of our GUI (Figure 2) to demonstrate our statistical models and their continuous, incremental updates at runtime.

### D. Query Optimization

During plan enumeration, the query optimizer selects the best plan from a set of semantically equivalent query plans. Recall that the optimizer does not make use of a cost model, but evaluates plans according to their efficiency measured during simulation. In our demonstration, we will present the rule-based query optimizer used in PIPES. We will point out why the well-known transformation rules from conventional and temporal databases equally apply to our stream algebra. We will also illustrate the query translation process from SQL to a logical query plan and the subsequent use of heuristics to determine the final physical plan. Additionally, we will explain how our optimizer summarizes query plans with common subexpressions to reap the benefits of subquery sharing.

### E. Plan Migration

The plan migration module is indispensable for the optimization of continuous queries because it replaces the current query plan with the optimized plan; thereby, it prevents the generation of duplicates and the loss of intermediate results. This migration process will be part of our demonstration. We will (i) show the interaction of the plan migration module and the query executor and (ii) demonstrate the proper migration from the currently running plan to the optimized plan inside the primary system.

### F. System Interaction

A major part of our demonstration will be the presentation of our novel optimization architecture, in particular the interaction of primary and secondary system. Our demonstration will comprise (i) the collection of secondary metadata, (ii) the maintenance of statistical models in the primary system, (iii) the transfer of queries to optimize and the associated statistical models, (iv) the simulation of query evaluation and cost measurement in the secondary system, and (v) the transfer of the optimized plan back to the primary system.

### G. Demonstration Data Sets

Our demonstration relies on synthetic as well as real-world data streams. As real-world data, we use sensor readings obtained from a sensor network installed in the Intel Berkeley Research lab [13]. The synthetic data streams simulate specific stream characteristics of practical relevance like bursts and changes in stream rates.

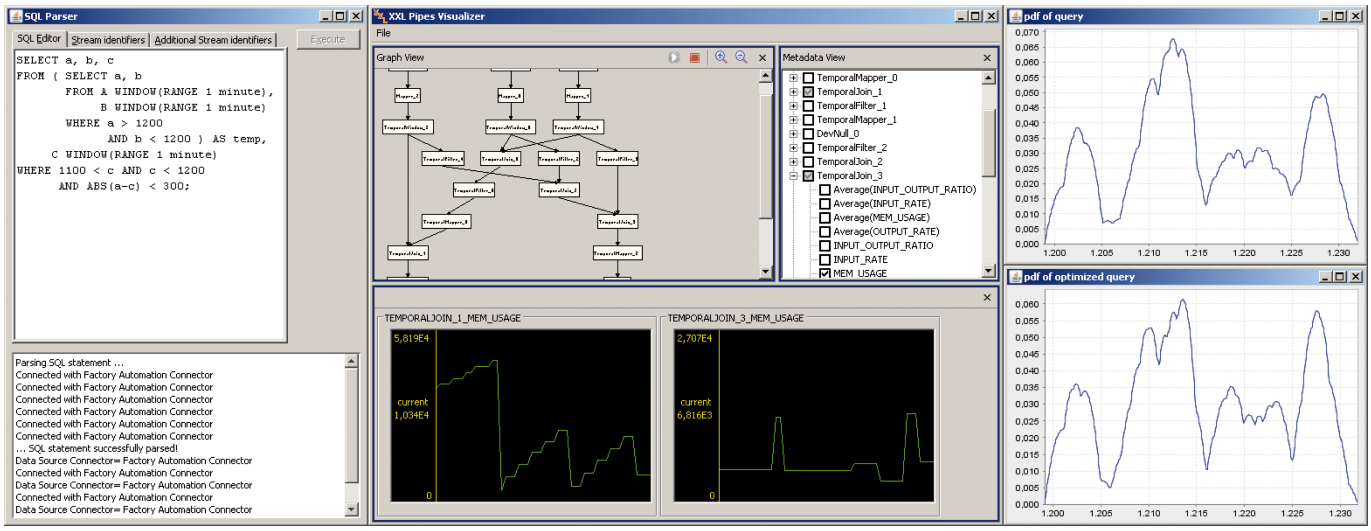


Fig. 2. GUI with SQL parser, query executor, metadata monitor, and statistical models

## IV. Conclusions

Our demonstration introduces a first approach toward simulation-based optimization in a DSMS. At the heart of this approach is the usage of sophisticated statistical models to simulate data streams in a secondary system. This system founds its decision for the best query plan on the performance achieved by candidate query plans obtained from simulations. Since we can accelerate the arrival rate in the simulation by orders of magnitude, we are able to predict the efficiency of a query plan extremely fast. Hence, we do not need an explicit cost model, which is typically difficult to determine in the data stream scenario due to the volatility of the stream characteristics. In our demonstration, we will present our prototypical implementation of this novel optimizer architecture, including (i) an architectural overview, (ii) component interaction, (iii) our statistical modeling framework, and (iv) the optimization process involving metadata measurement and plan migration.

In general, this optimization architecture poses a plethora of exciting research questions, which we are currently investigating. For example, which amounts of resources does the secondary system need? How can we use the statistical models to determine when to re-optimize? Which other system-related tasks in a DSMS can be similarly addressed? However, the discussion of these questions is beyond the scope of this demonstration proposal. The demonstration is intended to convey a first notion of the manifold benefits of our approach.

## Acknowledgements

This work has been supported by the German Research Society (DFG) under grant no. SE 553/4-3.

## References

[1] S. D. Viglas and J. F. Naughton, "Rate-based Query Optimization for Streaming Information Sources," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM Press, 2002, pp. 37–48.

[2] M. Cammert, J. Krämer, B. Seeger, and S. Vaupel, "A Cost-based Approach to Adaptive Resource Management in Data Stream Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, 2008.

[3] J. Krämer and B. Seeger, "PIPES – A Public Infrastructure for Processing and Exploring Streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM Press, 2004, pp. 925–926.

[4] M. Cammert, C. Heinz, J. Krämer, T. Riemenschneider, M. Schwarzkopf, B. Seeger, and A. Zeiss, "Stream Processing in Production-to-Business Software," in *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society, 2006, p. 168.

[5] J. Bercken, B. Blohsfeld, J.-P. Dittrich, J. Krämer, T. Schäfer, M. Schneider, and B. Seeger, "XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries," in *Proceedings of the International Conference on Very Large Data Bases*. Morgan Kaufmann, 2001, pp. 39–48.

[6] J. Krämer and B. Seeger, "A Temporal Foundation for Continuous Queries over Data Streams," in *Proceedings of the International Conference on Management of Data*. Computer Society of India, 2005, pp. 70–82.

[7] M. Cammert, J. Krämer, and B. Seeger, "Dynamic Metadata Management for Scalable Stream Processing Systems," in *Proceedings of the International Workshop on Scalable Stream Processing Systems*. IEEE Computer Society, 2007.

[8] C. Heinz and B. Seeger, "Towards Kernel Density Estimation over Streaming Data," in *Proceedings of the International Conference on Management of Data*. Computer Society of India, 2006.

[9] ——, "Adaptive Wavelet Density Estimators over Data Streams," in *Proceedings of the International Conference on Scientific and Statistical Database Management*. IEEE Computer Society, 2007, p. 35.

[10] J. Krämer, Y. Yang, M. Cammert, B. Seeger, and D. Papadias, "Dynamic Plan Migration for Snapshot-Equivalent Continuous Queries in Data Stream Systems," in *Proceedings of the International Conference on Extending Database Technology*, ser. Lecture Notes in Computer Science, vol. 4254. Springer-Verlag Heidelberg, 2006, pp. 497–516.

[11] Y. Yang, J. Krämer, D. Papadias, and B. Seeger, "HybMig: A Hybrid Approach to Dynamic Plan Migration for Continuous Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 398–411, 2007.

[12] G. Slivinskas, C. S. Jensen, and R. T. Snodgrass, "A Foundation for Conventional and Temporal Query Optimization Addressing Duplicates and Ordering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 21–49, 2001.

[13] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux. (2004) Intel Lab Data. data.txt.gz. [Online]. Available: http://berkeley.intel-research.net/labdata/