PIPES – A Public Infrastructure for Processing and Exploring Streams

Jürgen Krämer
Department of Mathematics and Computer Science
Philipps University Marburg
kraemerj@mathematik.uni-marburg.de

Bernhard Seeger
Department of Mathematics and Computer Science
Philipps University Marburg
seeger@mathematik.uni-marburg.de

Abstract

PIPES is a flexible and extensible infrastructure providing fundamental building blocks to implement a data stream management system (DSMS). It is seamlessly integrated into the Java library XXL [1, 2, 3] for advanced query processing and extends XXL's scope towards continuous data-driven query processing over autonomous data sources.

A Brief Summary of PIPES

A library approach, such as XXL and its integral part PIPES, seems to be adequate in the context of stream processing since it provides a variety of transparent and exchangeable building blocks. Based on these components, we are able to create a fully functional DSMS tailored to a specific application scenario, while ensuring flexibility as well as extensibility. Furthermore, we intend to analyze and compare the performance of such prototypically generated systems to determine which combination of building blocks is most suitable for a given application scenario. Contrary to existing work, we do not aim for building just another monolithic DSMS. Instead, we are firmly convinced that it is almost impossible to develop a general, performant as well as *flexible* DSMS that tries to cope with all issues arising in data stream management. Moreover, the flexibility as it is provided by PIPES is of utmost importance for an ad-hoc integration of streams and persistent databases.

The infrastructure PIPES is composed of several interacting frameworks as illustrated in Figure 1. The core framework allows to construct directed acyclic query graphs based on a publish-subscribe mechanism integrated into the graph nodes. We distinguish between three different types of graph nodes: A *source* transfers its elements to a set of subscribed sinks. A *sink* can subscribe and unsubscribe to multiple sources, respectively. During its subscription, it processes all incoming elements delivered by its sources. All operators satisfy the interface *pipe* that combines the functionality of a sink and a source. Hence, a pipe processes the incoming elements and transfers its outgoing elements to all subscribed sinks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD 2004 June 13-18, 2004, Paris, France.
Copyright 2004 ACM 1-58113-859-8/04/06 . . . $5.00.

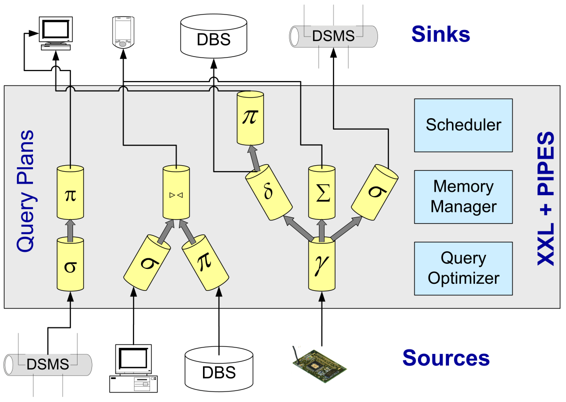

Figure 1: An architectural overview

A powerful and generic *operator algebra* provides the basis for the query construction framework. It relies on a well-defined, deterministic temporal semantics over time intervals that guarantees snapshot-equivalence. As a result, PIPES covers the functionality of the Continuous Query Language (CQL). Our operator algebra supports all operations known from the extended relational algebra, except sorting, whereas the operators are independent from concrete data types due to a parameterization with user-defined functions and predicates. Their data-driven implementation relies on time-based sliding windows and an incremental evaluation to ensure a non-blocking behavior. The current state of an operator is maintained by employing so-called SweepAreas, highly dynamic data structures with efficient insertion, retrieval, and reorganization capabilities managing the content of a sliding window. By adjusting the window size, the resource requirements of an operator can be controlled at runtime.

In order to build a DSMS, PIPES also contains separate frameworks that establish a basis for other essential runtime components, namely the scheduler, the memory manager, and the query optimizer. These components are controlled by flexible strategies which typically incorporate operator metadata maintained at runtime, e. g. mean stream rates or operator selectivities. Therefore, each node in a query graph continuously collects and updates relevant metadata information that is maintained in an online manner.

The *scheduling framework* provided by PIPES consists of a 3-layer architecture. In the lowest layer, multiple successive nodes in a query graph can be combined as a virtual node with an inherently push-based processing. This novel approach substantially reduces the communication overhead since it does not require any inter-operator queues. The middle layer allows to schedule virtual nodes within a single thread by decoupling successive virtual nodes with the help of an operator that buffers elements. In the top layer, all threads of the middle layer are scheduled. This hybrid approach benefits from neither being restricted to a single thread for scheduling nor assigning a separate thread per operator.
Due to dynamic system load caused by varying stream rates and frequent query graph modifications, scalability may suffer from limited memory resources. To overcome this problem, the *memory manager* offers a flexible framework for an adaptive memory management. It globally assigns and redistributes the available memory among stateful operators at runtime. Furthermore, it takes approximate query answers into account by applying user-defined load-shedding techniques whenever an operator exceeds its memory consumption limit.
The *query optimizer* is based on the semantics of our temporal operator algebra that retains most traditional transformation rules. Therefore, we are currently developing a rule-based query optimizer that is able to integrate a new continuous query into the running query graph, while sharing preferably large subgraphs. This is achieved by heuristically choosing from a set of snapshot-equivalent query plans the best one according to a cost model.

Benefit from XXL and its Library Approach

Since monolithic systems are too inflexible to adapt fast enough to the specialized query processing needs of new applications, we have focused on the development of a flexible and extensible software library for query processing ensuring the following qualities and advantages:

- *Applicability* Our library represents a toolkit consisting of intuitive and ready-to-use building blocks. These form a foundation for new query processing algorithms and even complete applications, while default implementations and simple use-cases facilitate the development tremendously. Moreover, single components are still usable and applicable with little knowledge about the entire library.

- *Reusability* The building blocks of our library are comprehensible and highly reusable because they are based on generic and abstract code derived from design patterns. XXL contains a variety of ready-to-use components, such as queues, heaps, index structures, and a rich cursor algebra. These components have contributed to accelerate the development of PIPES.

- *Extensibility* Due to the extensibility resulting from the library approach, components and state-of-the-art algorithms can easily be enhanced and simplify the deployment of specific applications. Since XXL seamlessly extends Sun's SDK and is freely available under the terms of the GNU LGPL, the worldwide number of XXL users is constantly growing.

- *Connectivity* In recent years the scope of XXL has continuously been widened. Hence, it forms a profitable and expanding platform for PIPES that provides powerful mechanisms to connect operators to databases, raw disks, files and even remote data sources.

- *Comparability* PIPES is an ideal testbed for algorithmic comparisons due to its library design. As a result of its research-oriented nature, a lot of advanced stream processing algorithms have already been implemented and can be used for experimental evaluation. Currently this is a unique feature of PIPES in comparison to related stream processing approaches.

Demonstration

Our demonstration will place emphasis on the advantages of PIPES resulting from its library approach. We will point out its broad applicability and easy connectivity considering two real application scenarios as examples, namely traffic management and online auctions. Both represent a foundation for the development of benchmarks in stream processing. We will show how to express and execute continuous queries in PIPES and provide an insight into the temporal operator algebra, the publish-subscribe mechanism, as well as the runtime system. At this, we will make use of two visualization tools. The first tool is a graphical user interface that offers to build complex query plans by a visual style of programming, whereas the second tool continuously monitors metadata maintained in the query graph nodes. This is especially helpful to illustrate the varying stream rates and operator characteristics in the application scenarios, for instance the effect of fluctuating stream rates on internal buffers. In order to show the advantage of code reusability, we will present data flow translation operators. These gracefully combine demand-driven and data-driven query processing over relations and streams by establishing a bridge between XXL's cursor algebra and PIPES. We finally demonstrate how PIPES suits for algorithmic comparisons by exemplarily introducing its sophisticated join and scheduling frameworks, which are powerful enough to handle state-of-the-art join and scheduling techniques.

Acknowledgments

This project has been supported by the German Research Society (DFG) under grant no. SE 553/4-1. In addition, we want to thank all recent and former members of our database research group for their comprehensive contributions.

REFERENCES

[1] The XXL / PIPES Project. http://www.mathematik.uni-marburg.de/DBS/xxl/.

[2] 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 *Proc. of the Conf. on Very Large Databases (VLDB)*, pages 39–48, 2001.

[3] M. Cammert, C. Heinz, J. Krämer, M. Schneider, and B. Seeger. A Status Report on XXL - a Software Infrastructure for Efficient Query Processing. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 26(2):12–18, 2003.