Nile: A Query Processing Engine for Data Streams*

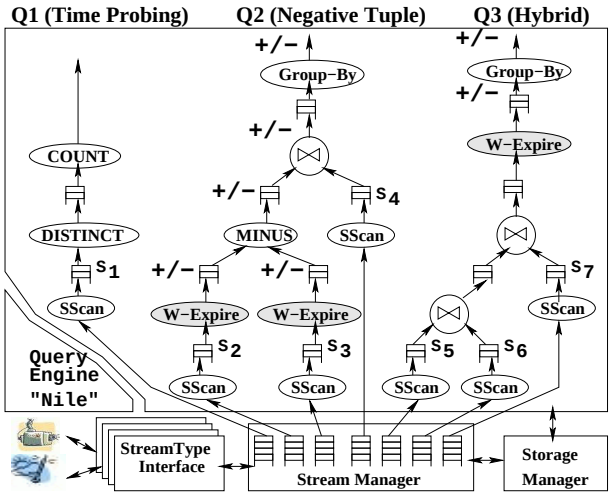
M. A. Hammad, M. F. Mokbel, M. H. Ali, W. G. Aref, A. C. Catlin, A. K. Elmagarmid,
M. Eltabakh, M. G. Elfeky, T. M. Ghanem, R. Gwadera, I. F. Ilyas, M. Marzouk, X. Xiong
Purdue University, West Lafayette, IN., USA

1. Introduction

This demonstration presents the design of "STEAM", Purdue Boiler Makers' stream database system that allows for the processing of continuous and snap-shot queries over data streams. Specifically, the demonstration will focus on the query processing engine, "Nile". Nile extends the query processor engine of an object-relational database management system, PREDATOR [4], to process continuous queries over data streams. Nile supports extended SQL operators that handle sliding-window execution as an approach to restrict the size of the stored state in operators such as join. More specifically, Nile supports the following features: (1) Efficient and correct pipelined execution of sliding window queries over multiple data streams. The correct execution is enforced by two novel pipelined scheduling approaches: the Time Probing approach and the Negative Tuple approach [2]. (2) Scalability in terms of the number of queries [3] and the number of data streams. (3) Access control to accept/register new continuous queries and new streams. (4) Providing guarantees for Quality of Service and Quality of Answers. (5) Online stream summary manager. (6) Integrating online data mining tools in query processing over data streams. (7) Approximate window join processing and joining in a network of data streams.

2. Stream Query Processing

The main components of the Nile stream query processor are shown in the figure. Source streams are modeled via a stream data type, *StreamType*. Nile communicates with source streams (e.g., remote locations over the network or sensor devices) through a *Stream Manager*. The Stream Manager registers new stream-access requests and uses the StreamType interface to retrieve data from the registered streams. Nile uses *Stream Scan (SScan)* to communicate with the stream manager. Nile uses several approaches [2] to schedule the pipelined execution of sliding window operators. The first approach (Time Probing) uses window-based operators, the second approach (Negative Tuple) uses a special operator, *W-Expire*, to emulate tuple addition to and subtraction from the window content. In addition, the Negative Tuple approach alleviates the operators from checking the window constraint. The third approach (Hybrid) approach uses a mix of the first two approaches. Nile introduces several window operators, besides window join [1], such as sliding window DISTINCT, Aggregate, and Set operations. In addition, Nile implements each operator as a separate preemptive system-scheduled thread. The operators communicate with each other through a network of FIFO queues.



3. Description of the Demo

In this demonstration we show the flexibility of the Nile system to define new streaming sources and execute sliding window queries. We present applications on real data sets that include retail transactions from Wal*Mart stores, video data streams, and spatio-temporal data.

4. Acknowledgment

We would like to thank Michael Franklin for his collaboration in developing many of the ideas in Nile.

References

[1] M. A. Hammad, W. G. Aref, and A. K. Elmagarmid. Stream window join: Tracking moving objects in sensor-network databases. In *SSDBM*, 2003.

[2] M. A. Hammad, W. G. Aref, M. J. Franklin, and et al. Effi cient execution of sliding-window queries over data streams. In *Purdue University CSD TR #03-035, 2003.*

[3] M. A. Hammad, M. J. Franklin, W. G. Aref, and et al. Scheduling for shared window joins over data streams. In *VLDB*, 2003.

[4] P. Seshadri. Predator: A resource for database research. *SIGMOD Record*, 27(1):16–20, 1998.

* This work was supported in part by the National Science Foundation under Grants IIS-0093116, EIA-9972883, IIS-9974255, IIS-0209120, and EIA-9983249.