TelegraphCQ: Continuous Dataflow Processing*

Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin
Joseph M. Hellerstein, Wei Hong†, Sailesh Krishnamurthy, Samuel R. Madden
Fred Reiss, Mehul A. Shah

University of California, Berkeley, CA 94720
†Intel Research Laboratory, Berkeley,CA 94704

http://telegraph.cs.berkeley.edu

1. INTRODUCTION

At Berkeley, we are developing TelegraphCQ [1, 2], a dataflow system for processing continuous queries over data streams. TelegraphCQ is based on a novel, highly-adaptive architecture supporting dynamic query workloads in volatile data streaming environments. In this demonstration we show our current version of TelegraphCQ, which we implemented by leveraging the code base of the open source PostgreSQL database system. Although TelegraphCQ differs significantly from a traditional database system, we found that a significant portion of the PostgreSQL code was easily reusable. We also found the extensibility features of PostgreSQL very useful, particularly its rich data types and the ability to load user-developed functions.

Challenges: As discussed in [1], sharing and adaptivity are our main techniques for implementing a continuous query system. Doing this in the codebase of a conventional database posed a number of challenges:

- **Adaptivity:** While the TelegraphCQ query executor leverages a lot of existing PostgreSQL functionality, it has a very different adaptive architecture.
- **Shared processing:** TelegraphCQ *shares* the execution of multiple queries causing a significant change to the process-per-connection model of PostgreSQL.
- **Data ingress:** Shared stream processing results in a new non-blocking architecture for wrappers.

Figure 1 shows an overview of TelegraphCQ. A client establishes a connection with a TelegraphCQ Front End (FE) process, just as in PostgreSQL. Queries are sent through the connection where they are pre-planned by the FE and sent to the TelegraphCQ Back End (BE) process. The BE *dynamically* folds them into the running query, and sends

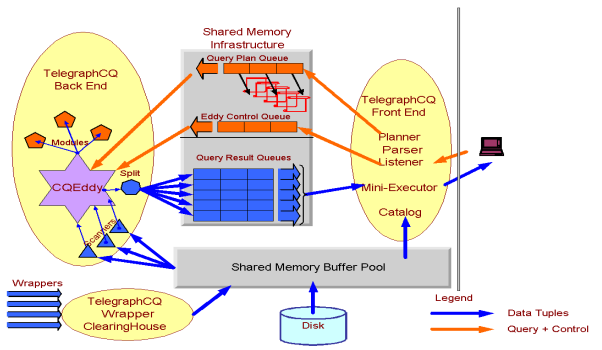

Figure 1: TelegraphCQ Architecture

results back to the FE through the results queue. All user-defined wrappers are hosted in the TelegraphCQ Wrapper ClearingHouse process which sends newly formed tuples to the BE through the standard PostgreSQL shared memory bufferpool.

Status: In its current state, the TelegraphCQ DDL supports the creation of archived and unarchived streams that are fed with external sources using stream- and source- specific *wrapper* functions. Continuous queries can be registered that work over these streams. The queries can involve individual streams, joins over multiple streams and joins over streams and tables. The DML for these queries includes sliding window syntax that supports grouped aggregate operations over these sliding windows.

In this demonstration we use two applications to show how TelegraphCQ's adaptive architecture enables the dynamic introduction of new queries into a running system, as well as how it reacts to volatility in the input data streams. First, we integrate TelegraphCQ with a sensor network to show how the system adapts to abrupt changes in the content and volume of data streams. Next we use a simulation of freeway traffic sensors to show TelegraphCQ scales to large numbers of input streams and queries.

*This work has been supported by NSF awards IIS-0208588, IIS-0205647, IIS-0086057 and SI0122599 as well as funding from Microsoft, IBM, Intel and the UC MICRO program.

SIGMOD 2003, June 9-12, 2003, San Diego, CA.

2. REFERENCES

[1] Chandrasekaran, S., *et al.*. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proceedings of CIDR Conference* (2003).

[2] Krishnamurthy, S., *et al.*. TelegraphCQ: An architectural status report. *IEEE Data Engineering Bulletin 26*, 1 (March 2003).