Streaming Database Architecture

TelegraphCQ

Introduction

- Streaming data – hot new topic
- Needs to be handled differently by something other than a traditional query processor
- TelegraphCQ – “a system for continuous dataflow processing”
- Made to handle many streams of continuous queries and large amounts of variable data streams

TelegraphCQ

- Data sources continually send data to named streams in TelegraphCQ
- Each stream is associated with a wrapper used to understand the source data
- Connections can be made to TelegraphCQ by users and applications to issue continuous queries over these streams

Architecture Overview

- Uses PostgreSQL to support TelegraphCQ features
- Back End – dedicated process that executes continuous queries - Eddies
- Front End – per connection process that fields queries and returns results from/to the client
- Wrapper ClearingHouse – process that ensures query execution is not impeded by entering data operations

Back End Overview

- Main Feature: Eddies
- Tuples are dynamically ordered by routing policy
- done and ready bits track the path of each tuple
- New queries melded with already running queries in modules
- completionMask for each query determines if a tuple can be the output
- Scanners scan the source - single scanner per module so tuples are never copied
- Tuples flow through all the modules and across all the queries
Front End Overview

- Listener – receives commands from client and chooses where to execute it within the Eddy control queue
- Parser – parses queries
- Planner – updates the query plan queue that is to be utilized within the Eddy
- Mini-Executor – dequeues results from the Query Result queue formed by the Eddy and sends it back to the client

Wrapper Overview

- Push and Pull sources through Fjord implementation
- Pull sources – traditional database systems
- Push sources – connections can be initiated by TelegraphCQ or by data source
- Fjords implement push and pull queues – push queue input operator has to reply to demands by output operator
- Push queue input operator is non-blocking

Summary

- Very good system but is a work in progress
- Adaptivity and storage system problems
- Future holds promise

*Sources from: http://citeseer.nj.nec.com/599185.html
“TelegraphCQ: An Architectural Status Report,” Krishnamurthy, Cooper, Deshpande, Franklin, Hellerstein, Hong, Madden, Reiss, Shah.