2006

Integrating database and data stream systems

Rutul Mashruwala

Integrating Database and Data Stream Systems

Master's Project Proposal

By

Rutul Mashruwala

Department of Computer Science

Rochester Institute Of Technology, NY

Advisor: Prof. Raj

Reader: Prof. Bischof

Table of Content

1 **Abstract**

Currently we have a traditional database system, which stores the data, and it is persistent and we query it assuming data is there. There is active research going on Data Stream System. Data Streams are nothing but the stream. To the programmers world streams are not a new concept, but database world do think. Languages like c/c++ and java have already concept of streams. My hypothesis is that DBMS and DSMS can be combined under one name Data Management System. As the streams are not persistent the data coming through the streams can get lost. As we already have one persistent data storage management system (the traditional DBMS), combining DBMS and DSMS will make the streams persistent.

Instead of giving a client separate look of DBMS and DSMS I would like the client to see only Data Management System by building a middleware. The client will access the data through middleware. I will be using ACF to test my application. Take an example of Kodak stocks. All the history of price is stored in DBMS and new price changes are coming through the streams. Now when client want to study the price graph of Kodak shares it needs to consider all data, the history and the new streaming data. When client talks with the middleware, the middleware will provide the client all information by combining the history and the new streaming data.

2 Introduction

Data Streams

There is ongoing research on developing efficient Data Stream Management Systems that is a system for processing continuous queries over multiple continuous data streams and stored relations. Applications like network traffic, sensor produces continuous data and in very high volume. The Data stream management system (DSMS) should be efficient enough to run continuous queries on these high volume and continuously changing data. As the queries might take a long time to run, designing of DSMS should also consider the time to run these long queries and generating approximate result. A separate language has been developed similar to SQL to write queries in DSMS. This language is called Continuous Query Language (CQL) for the STREAM project of Stanford University.

Active Collection Framework

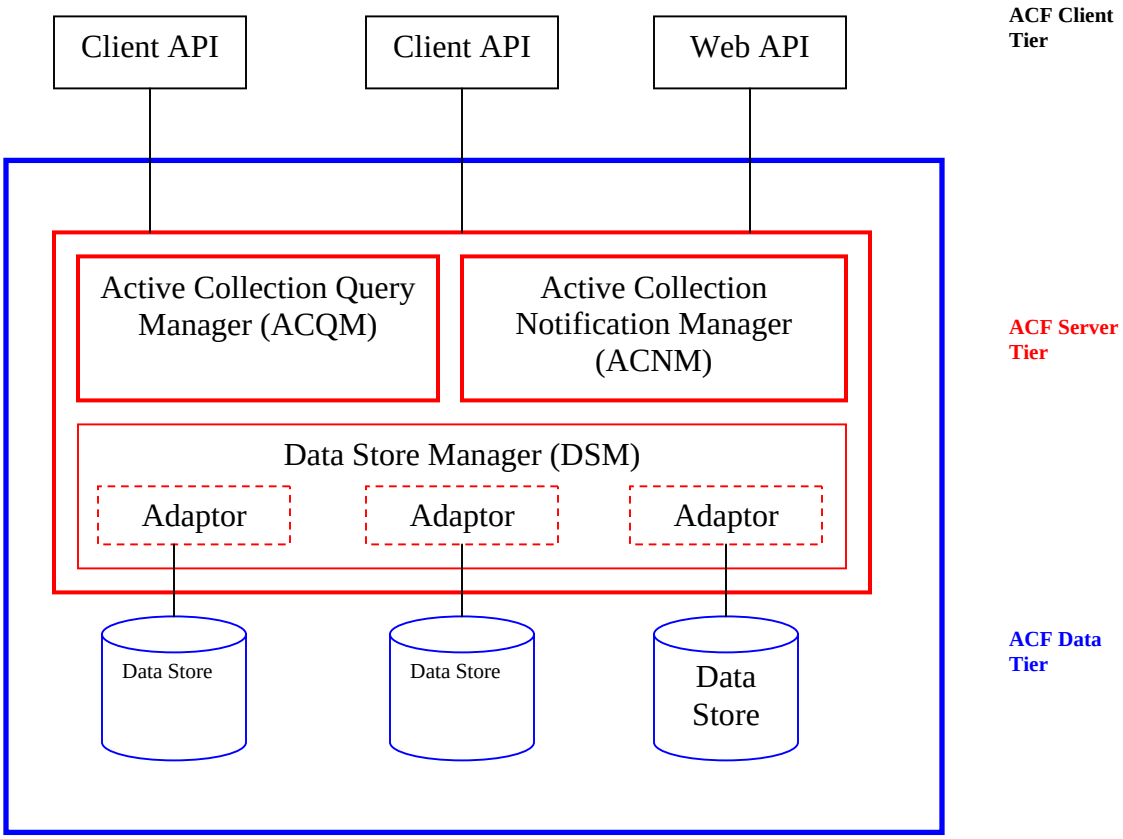

Figure 1: Logical Architecture of ACF (from [1])

Active Collection Framework is a framework that provides data as well as data changes to applications. These data changes provided by Active Collection Framework are near-real-time data. The main components of the ACF are ACF Object and Active Collection. Each ACF Object represents corresponding tuple in a relation. The fields of a tuple represent instance variables of ACF Object. Active Collection is collection of such ACF Objects fulfilling certain predicates. Client request the server to create an Active Collection. The Active Collection, collection of ACF Objects, is created based on the predicates supplied by the client. Then server puts the ACF Objects satisfying the predicates into Active Collection. And return the active collection back to the client. Server needs to keep monitoring the changes in data. Whenever there is a change ACF Server updates ACF Objects. It checks the predicates against each ACF Object, if the updates object still satisfy the predicate it updates object in Active Collection, if Object no longer satisfy the predicate ACF Server removes the object from Active Collection and the new objects satisfying the predicate are being added to Active Collection by server. And this updated Active Collection is sent to client to notify the client about changes it is registered for.

The changing of the data can be coming trough the streams. If ACF is using streams as a means of communication it can also use DSMS for the changing data. For simplicity, this project will simulate the stream data. DSMS does not only deal with streams but it also deals with the stored relations.

3 Architecture

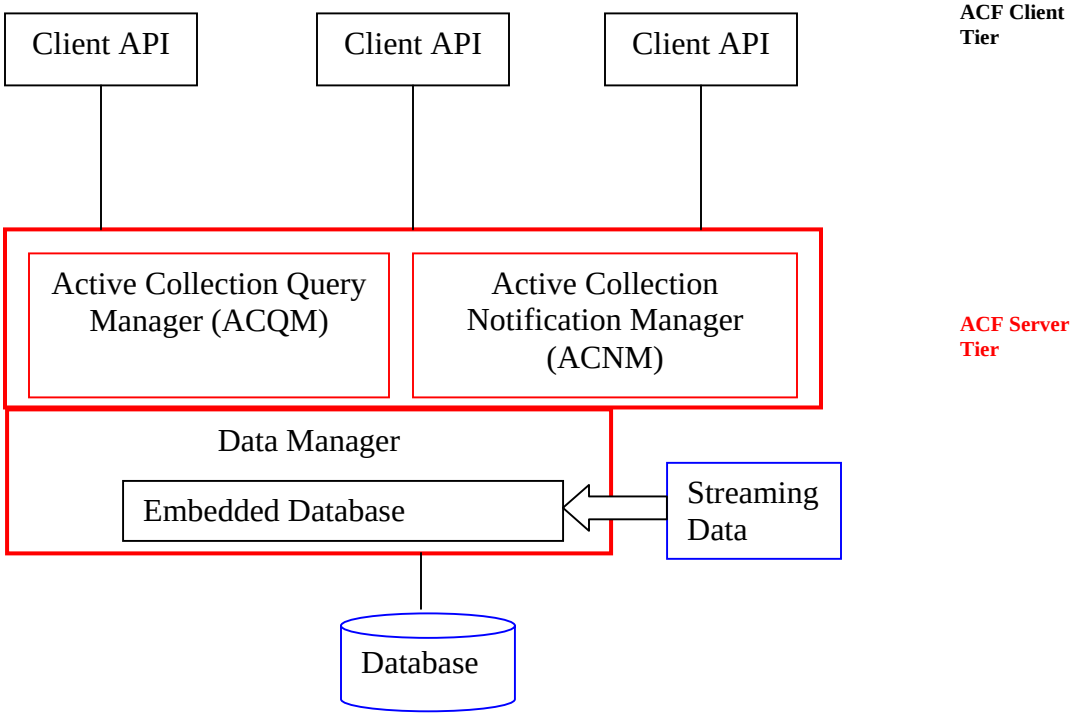

Figure 2: Architecture of Database and Data Stream with ACF

The goal here is to make the data access transparent to the client. As shown in the figure, Data Manager is the added layer to existing ACF architecture that makes this possible. This layer is responsible to hide difference between Database and Data Streams to the client. As this project will be using Java, an object-oriented language, the "data" on the stream will be in form of "objects". And in order to proper communicate, these objects should be self-describing. That is these objects should have some protocol buried in them that makes server understand what to do with incoming stream data. The client will request for creating Active collection and it will also perform the CRUD operations.

The Data Manager will have embedded database. Embedded database is a main memory database and they are faster compare to traditional disk based database. This embedded database will have history of the data as well as the data coming from the streams. By writing the stream data into embedded database we can achieve faster response time, as there is no disk IO is involved. The streaming data will be also written to disk database but after the client is notified about the changes if required. Client always query to these embedded database. When an update comes from client, the updates need to go to disk database before client is notified about successful updates.

The core component in Active Collection Framework is the Notification Manager. The server should able to generate notification to client whenever there is a change in the data that affects the Objects satisfying predicates given by the client. As this project focuses

on integrating Database and Data streams using ACF, the notification manager now also have to consider the data coming through the streams besides the data that is stored in database. And generate the notification based on data from both places and the predicates.

The communication between client and server will take place through streams. The “object” that comes through the stream by client is passes to Query Manager. The client has to include the port number it is listening on for the updated Active Collection. The Server is also responsible for sending the Active Collection, which is provided by Notification Manager, to client. Thus, client does not have to request for the updated collection. It is responsibility of ACF Server to provide the updated data. This eliminates any polling by the client and implements PUSH technology.

If you implement triggers in database to monitor the data changes, you need something to monitor the changes in database. In this project, instead of implementing triggers, Query Manager will write the changes in database and will notify Notification manager that there are some data changes. Notification Manager will read the changes, update Active collections for the client and will send the Active Collection to client via Communication Layer.

4 Functional specifications

The project will provide a transparent data system, which is integration of Database and Data Streams. As ACF is used for testing, ACF Server with basic functionality and a small application will be developed which will work as client.

At the start up of the application history from the disk database will be loaded into embedded main memory database. And the two databases will be kept in sync for the subsequent data changes. ***ACFServer*** will wait for client request. Upon receiving a request from client, creates a new thread of ***RequestHandler*** for further processing. This scheme will allow server to handle requests from multiple clients simultaneously.

RequestHandler will read the "object" provided by ***ACFServer*** and parse it. Based on the information hidden in the object ***RequestHandler*** will take further actions. The possible operations can be Insert, Delete, Update, Retrieve, Create Active Collection or Destroy Active Colletion. It passes on the information to ***QueryManager*** and waits for response about transaction (success or error) from ***QueryManager***. Once, getting the response back it will notify the client with result (data or success or error if any). It also registers client with the server if not yet register and has asked for Creating Active Collection. Registration is based on IP Address and port number the client is listening on.

QueryManager first checks for the validity of supplied parameters based on operation requested. If validated takes following action based on operation:
For Insert, Delete and Update passes on the data to ***DatabaseManager*** to update embedded and disk base database and returns the result to ***RequestHandler***. It will also notify ***NotificationManager*** that there is some change in database. Notification Manager will read from embedded database.
For Retrieve operations, ***QueryManager*** will send RecordSet that it got from ***DatabaseManager*** to ***RequestHandler***.
For Creating Active Collection, ***QueryManager*** will make an entry in NotificationCollections will supplied predicates and client info.

NotificationManager will request ***DatabaseManager*** for updated data when it receives a signal from ***QueryManager***. ***DatabaseManager*** will fulfill this request by reading the data from embedded database. Upon receiving data from ***DatabaseManger*** it will update all ActiveCollections and notify the client that is register for it. The client will get updated Collection only if there is change in the Collection.

Stream Data Simulation

Just to differentiate stream data with the crud operations, Server will listen for Stream data on a separate port. Client will have the control to generate streaming data. Client can control the time interval at which data is generated.

5 Schedule

- Server side design and implementation 3 Weeks
- Client (Application) design and implementation 3 Weeks
- Project Report 2 Weeks

6 Deliverables

- Project Report
  - User guide included in project report
- Server code
- Client Code
- Project Presentation

7 Reference:

1. “Experience with the Active Collections Framework” by Rajendra K. Raj, Rochester Institute of Technology. August 2003
2. STREAM: The Stanford Stream Data Manager (short overview paper) *IEEE Data Engineering Bulletin, Vol. 26 No. 1, March 2003*
3. “A Programming Framework for Using Data Stream (and Database) Systems” by Rajendra K. Raj, Rochester Institute of Technology
4. www.SQLite.org
5. www.sleepycat.com