

Using Stream Semantics for Continuous Queries in Media Stream Processors

Amarnath Gupta

San Diego Supercomputer Center
University of California San Diego
gupta@sdsc.edu

Bin Liu Pilho Kim Ramesh Jan

Electrical and Computer Engineering
Georgia Institute of Technology
{bliu, phkim, jain}@ece.gatech.edu

Introduction

Over the past few years, processing continuous queries over one or more unbounded streams has become a major research area in data management. A number of research groups are developing data stream processing systems for a wide variety of problem domains including network data management, traffic monitoring, business data analysis, environmental sensor networks and immersive environments (see [1] for a repository of application domains and example queries). Our motivation arises from an immersive system application where we need to continually evaluate queries over *media* and *feature streams*. A *media stream* is usually the output of a sensor device such as video, audio or motion sensor that produces a continuous or discrete signal, but typically cannot be directly used by a data stream processor. Instead, it needs to be post-processed by one or more transformers to produce *feature streams* which are data streams correlated to the media stream temporally and in terms of content. Very often, we would want to evaluate a query over one or more feature streams, but would want to output a part of the media stream.

Our Approach

The basic premise of our approach is that not all streams are created equal – they differ in content, streaming properties, operations permitted on them, and how they would be used in the application. Furthermore, in the case of media and feature streams, explicit *inter-stream constraints* exist and can be exploited in the evaluation of continuous queries in the spirit of semantic query optimization. We express these properties using a media stream declaration language MSDL. In the demonstration, we present IMMERSI-MEET, an application built around an immersive environment. The IMMERSI-MEET system distinguishes between *continuous streams*, where values of different types come at a specified data rate, and *discrete streams* where sources push values intermittently. In MSDL, any dependence declaration must have at least one dependency specifying predicate in the body. We use statements like

`punctuates(discrete_stream, continuous_stream)` to denote inter-stream dependency. Similarly, dependency predicate `start_synchronized(discrete_stream, discrete_stream)` denotes that the start of the stream elements are synchronized, but not their ends.

As stream declarations are registered, the stream constraints are interpreted to construct a set of *evaluation directives*. For example, the skip keyword for the stream *S* produces a directive `S.dropElement(<fraction>)` whereby a few frames of the media may be skipped without much effect. We call this a directive because the stream query engine may choose not to exercise the directive.

The Demonstration

The immersive environment of the IMMERSI-MEET environment is much richer in the variety of media streams. It is built around a meeting room with a number of cameras and microphones. There is a designated camera and microphone for the primary speaker. Every other participant has a dedicated microphone and shares some cameras. During demonstration, the meeting room will be “simulated” by first pre-recording the different channels of the meeting on a number of different computers. These computers will then send a set of media and feature streams to a processing computer that runs the media stream processor. We will show an interface to register streams and queries. Further, we will present a “debugger’s console” which would show the operators used for a query and the states of the queues as the execution occurs. The queries we would show would include both the notification case, where the user gets connected to the meeting when a specified event occurs, and the “log upon event occurrence” case, where the output is a media record composed by projecting portions of the media that satisfy the query and other attributed computed from feature streams or from static relations.

References

- [1] The Stanford STREAM Group. Stream Query Repository. web site at <http://www-db.stanford.edu/stream/sqr/>.