DataCell: Exploiting the Power of Relational Databases for Efficient Stream Processing

by Erietta Liarou and Martin Kersten

Designed for complex event processing, DataCell is a research prototype database system in the area of sensor stream systems. Under development at CWI, it belongs to the MonetDB database system family. CWI researchers innovatively built a stream engine directly on top of a database kernel, thus exploiting and merging technologies from the stream world and the rich area of database literature. The results are very promising.

Rather than simply transmitting the raw measured data, current state-of-the-art sensors are capable of a limited amount of processing. This feature has many positive effects, such as keeping the network usage and costs as low as possible. However, this is not enough to replace the role of well-equipped nodes that gather streaming sensor data from multiple sources and which account for the biggest share of the processing cost. These nodes should be able to perform complex query processing on large amounts of incoming data, meeting strict real-time deadlines even in periods when the frequency of incoming data explodes.

Our work focuses on this part of the sensor research. We are designing and developing a system called the DataCell, which is capable of efficiently collecting and processing high volumes of stream data. We are currently studying the DataCell over the stream application scenario of an ambient home setting. The DataCell is positioned as a data refinery cell that acts as an easily programmable data hub in a multi-network environment. Its task is to collect, filter and aggregate information from different sources to enable complex decision making, such as control of the lighting based on audio/video presentations. The challenge in an ambient environment is to hide the computer from the casual user, even while it is actively steering the environment. An example query in the ambient scenario could be, tune the television to my favourite show when I sit on the couch; ie depending on the weight measured by a sensor in the seat, and the time of day, different TV shows will appear on the screen.

In stream applications, we need mechanisms to support long-standing queries over data that is continuously updated from the environment. This requirement is

significantly different from what happens in a traditional database system, where data are stored in static tables and users fire one-time queries to be evaluated over the existing data. Given this critical difference, the pioneering architects of the data stream management system naturally considered existing database architectures inadequate to achieve the desired performance: instead they designed new architectures from scratch.

However, working from scratch makes it difficult to exploit the existing knowledge and techniques of relational databases. This disadvantage became more pronounced as the stream applications demanded more functionality. In DataCell therefore, we started at the other end of the spectrum, building an efficient data stream management system on top of an extensible database kernel. With careful design, this allows us to reuse the sophisticated algorithms and techniques of traditional databases. We can provide support for any kind of complex functionality without having to reinvent solutions and algorithms for problems and cases for which a rich database literature already exists. Furthermore, it allows for more flexible and efficient query processing by allowing batch processing of stream tuples, as well as non-consecutive processing by selectively picking the tuples to process.

The idea is that when stream tuples arrive in the system, they are immediately stored in (appended to) a new kind of table called a basket. By collecting tuples into baskets, we can evaluate the continuous queries (which are already submitted to the system and are waiting for future incoming data) over related baskets as if they were normal one-time queries. This allows us to reuse any kind of algorithm and optimization designed for a modern database system. Each query has at least one input and one output basket. It continuously reads data from the input baskets, processes this data and creates a result which it then places in its output baskets. Once a tuple has been seen by all relevant queries, it is dropped from its basket.

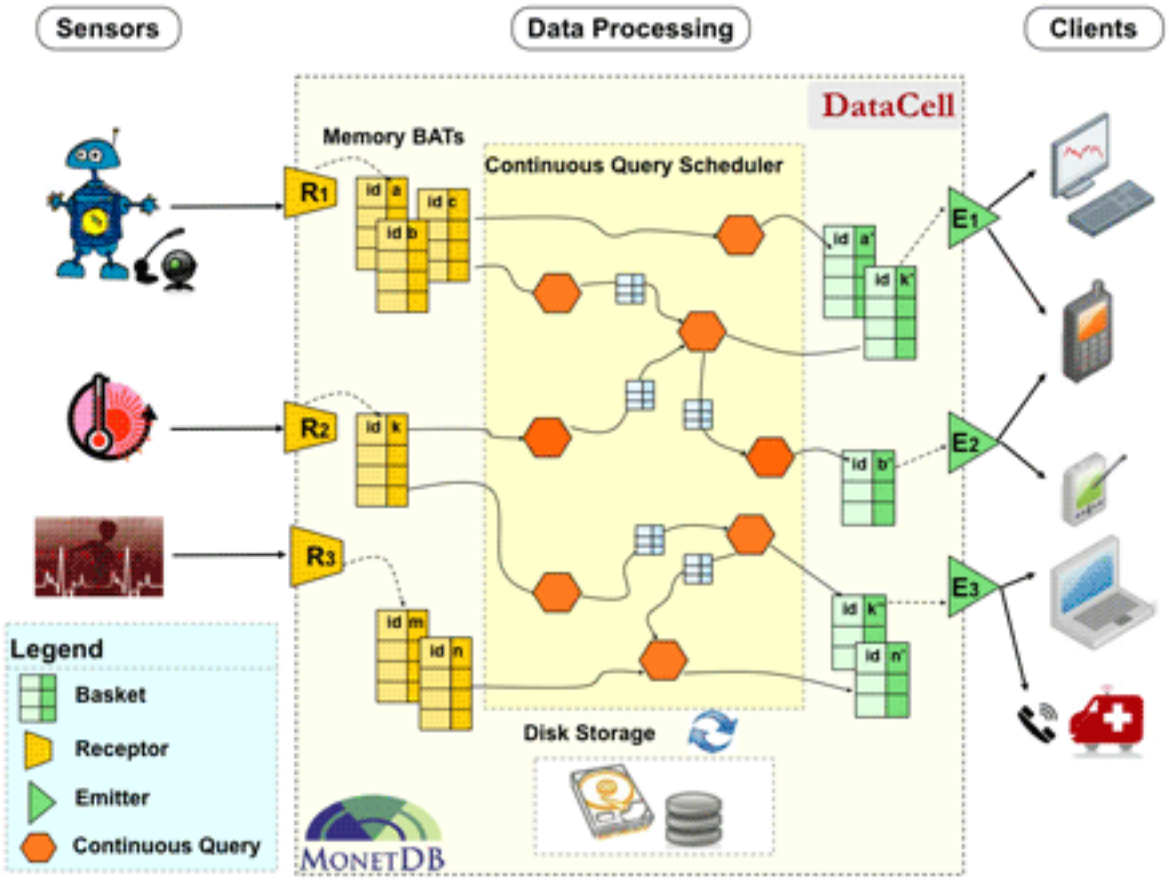

Figure 1: The DataCell in the ambient scenario.

This description of the process is somewhat simplified, since this process allows the exploration of quite flexible strategies. For example, the same tuple may be thrown into multiple baskets where multiple queries are waiting, query plans may be split into parts, and baskets may be shared between similar operators (or groups of operators) of different queries, allowing results to be reused.

The periphery of a sensor stream engine is formed by adapters, eg software components to interact with devices, RSS feeds and SOAP Web services. The communication protocols range from simple messages to complex XML documents transported using either UDP or TCP/IP. The adapters for the DataCell consist of receptors and emitters. A receptor is a separate thread that continuously picks up incoming events from a communication channel and forwards them to the DataCell kernel for processing. Likewise, an emitter is a separate thread that picks up events prepared by the DataCell kernel and delivers them to interested clients, ie those that have subscribed to a query result.

We designed and developed the DataCell at CWI in Amsterdam, funded by the BRICKS project. It is implemented on top of the MonetDB, an open-source column-oriented database system. Currently it is a research prototype and the goal is to be able to disseminate the DataCell soon as part of MonetDB.

Link:
http://monetdb.cwi.nl/

Please contact:
Erietta Liarou
CWI, The Netherlands
Tel: +31 20 59 24 127
E-mail: erietta@cwi.nl

Martin Kersten
CWI, The Netherlands
Tel: +31 20 59 24 066
E-mail: mk@cwi.nl