Fjording the Stream: An Architecture for Queries over Streaming Sensor Data

Samel Madden, Michael J. Franklin
University of California, Berkeley

Presented by: Tai Hu

Terminology
- Fjords – Framework in Java for Operators on Remote Data Streams
- Operator – a set of standard database modules
- Tuple – A collection of data which understood by operator
- Window – A time interval

Outline
- Introduction – why do we need a new architecture
- Background – Brief introduction for sensor environment and requirement for sensor query processing
- Solution – How Fjord works
- Experiment (will be skipped)
- Conclusion

Motivation
- If industry visionaries are correct, our lives will soon be full of sensors
- Sensors will be connected by wireless networks, each monitoring and collecting data about the environment at large
- Need a new architecture for queries over Streaming data

Why Do We Need a New Architecture?
- Properties of sensor
  - Unreliable network connection
  - Typically sample periodically and push immediately
  - No record of historical information
  - Power constraint (Most of them operate on the battery power)
  - Produce very large amount of data in real time

Standard Database vs. Sensor based Data Source
- Difference between Standard Database Source and Sensor Based Data Source
  - Sensors typically deliver data in streams. They produce data continuously, often at well defined time intervals without having been explicitly asked for that data
  - Sensors are fundamentally different from the over-engineered data sources typical in a business DBMS.
Proposed Solution to Problem

- An enhanced query plan data structure called Fjords
  - Allow users to pose queries that combine streaming, push based sensor source with traditional pull-based sources
  - Non-blocking and windowed operators
- A power-sensitive Fjords operators called sensor proxies
  - A mediators between the query processing environment and physical sensors

Difference to Other Works

- Providing the underlying systems architecture for sensor data management
- Enable other applications to be built on the top of this framework

Sensor Environment

- Physical sensor
  - A sensor consists of a remote measurement device that provides data at regular interval
  - A sensor may have some limited processing ability or may simply output a raw streams of measurements
  - No parsing on the queries or keeping track of which clients need to receive samples from them

Sensor Environment - Continue

- Sensor’s proxy
  - Run on a fixed, powered, and well connected server with abundant disk and memory resource
  - An interface between physical sensor and the rest of the query processor
  - One machine is the proxy for many sensors.
  - Responsible for packaging samples as tuples and routing those tuples to user queries as needed

Sensor Environment – Continue

- Sensors do not participate in query processing
- Proxy can adjust their sample rate or ask them to perform simple aggregation before relaying data.
Next-Generation Traffic Sensor
- Tiny – Only 10 cm³ now. The ultimate goal is 1 mm³
- 8-bit microprocessors with small amounts of RAM running from 1 to 16MHz
- Transmitting at tens of kilobits per second with a range of few hundred feet.
- Capable of on board computation

Requirements for Sensor Query Processing
- Properties must be taken into account when designing the low level infrastructure
  - Limitations of Sensors
  - Streaming Data
  - Processing Multiple Queries

Limitations of Sensors
- Limited Resource, such as, battery capacity, communication bandwidth, and CPU cycles
- Data transmission is much more expensive than CPU cycles

Streaming Data
- Continuous, never ending streams of data
- Blocking operator cannot be used. (i.e. sorts, aggregates and join)
- Pull-based iterator model does not map well onto sensor streams
- Operators must process data only when sensors make it available

Solution
- Fjords: Generalized Query Plans for Streams
  - Operators and Queues
  - Flexible Data Flow
  - State Based Execution Model
  - Sensor Sensitive Operators
  - Sensor Proxy
  - Multiple Queries in a Single Fjord

Processing Multiple Queries
- In many sensor scenarios, multiple users pose similar queries over the same data streams
- A query processing system should be able to dynamically adjust sensor’s sample interval and delivery rate
Fjords: Generalized Query Plans for Streams
- A generalization of traditional approaches to query plans
- Support for integrating streaming data that is pushed into system with disk-based data which is pulled by traditional operators

How a Fjords Works
- Each machine runs a single controller in its own thread
- Operators are instantiated by controller
- Operators are connected by queues

How A Fjords works - Continue

Operators and Queues
- Operators form the core computational unit of Fjords
- Each operator has a set of input queue and output queue
- Operator read tuple from input queue in any order and output any number of tuple to its output queue
- A dataflow architecture

Operators and Queues - Continue
- Responsible for routing data from one operator to another
- No transformation on the data
- Could connect either local operator or remote operator
- It is possible for a queue to fill

Flexible Data Flow
- Key advantage: Using a mixture of push and pull connections between operators
- Push and pull are implemented by the queue
- Push queue relies on its input operator to put data into it
- Pull queue actively requests that input operator produce data in response to a get call on the part of the output operator
State Based Execution Model

- Programming model for operators is based upon state machine
  - Each operator in the query plan represents a state in a transition diagram
  - A current state $s$ and some set of inputs $I$ causes the operator to transition to a new state (or remain in the same state)
    $$ o \leftarrow \text{transition}[s, i] $$
  - Implicit in this model of state programming is that operators do not block

Advantage of a State Machine Model

- Reduces the number of threads
  - Threads are very expensive on certain operating system
  - Lack of control on thread
  - Fjord scheduler mechanism enables dynamical control on all operators

Sensor Sensitive Operators

- Some traditional operator cannot be applied to the streaming data
  - Incrementally or periodically output the results (Non-blocking)
  - Window based operator (Blocking)

Sensor Proxy

- Shield the sensor from having to deliver data to hundreds of interested end-users
  - Adjust the sample rate of the sensor based on user demand
  - Direct the sensor to aggregate samples in predefined ways or to download a completely new program into the sensor
  - Limit the number of copies of sensor tuples flowing through the query processor to just one per sample

Multiple Queries in a Single Fjord

- Instantiate streaming scan operators with multiple outputs that allocate only a single copy of every streaming tuple
- New queries over the same streaming source are folded into an existing Fjord rather than being placed in a separated Fjord
### Conclusion

- Fjords architecture combines proxies, non-blocking operators and conventional query plans.
- Sensor proxies serve as intermediaries between sensors and query plans, using sensors to facilitate query processing while being sensitive to their power, processor, and communications limits.