StreamFS: Metadata A First Class Citizen

Jorge Ortiz and David Culler
Computer Science Division
University of California, Berkeley
{jortiz,david.su,culler}@cs.berkeley.edu

1 Introduction

A recent article [1] highlights some of the fundamental challenges in dealing with sensor data. These fundamental issues have led to various research efforts [6, 8],almost exclusively dealing with efficient querying and data quality. Sensor metadata is often treated as a second-class citizen. However, we contend that metadata management just is as fundamental. Sen- sors are embedded in the environment and their placement, categorization, and other metadata is as important as the data being collected from them. As such, the operations and queries we perform should always consider the associated metadata. If changes in the metadata imply changes in placement, calibration, or other deployment state, the system should account for these changes and resolve them when possible or inform the user.

Any sensing environment collecting data about the physical environment must formulate a solution for managing the metadata. As the deployment grows, so does the management complexity. File systems have been used for organizing a wide range of data and we believe that this abstraction can serve to alleviate the complexity of organizing, locating, and tracking application metadata in sensor deployments. In addition, the file system abstraction can simplify sharing data across de- ployments by canonizing the hierarchical organization of the data; similar to the the File System Hierarchy standard [1] used in Linux-based systems.

Much of the deployment metadata can be decomposed hierarchically and when coupled with symbolic linking it can capture the interrelationships between multiple deployment contexts. For example, when a power meter is attached to an outlet, it is bound to the device is it attached to, bound to the location in which it is placed, and bound to a load element in the electrical load tree. All three describe where the sensor is placed and are valid ways of referring to that sensor and its data. Furthermore, that sensor denotes a point of interaction between each context. File system constructs naturally accommodate multiple namespaces by partitioning each namespace into separate directories and linking files symbolically between them. In addition, we can version the hierarchy to track context changes and consider them when queries are performed by the user.

Sensor placement can vary over time and this change in context affects how the data is interpreted. Therefore, the metadata should always be considered when data is retrieved. Queries on the data should run in the right context and that context should be tracked as it changes throughout the lifetime of the deployment.

- Snap shot rebuilding of hierarchy and associated metadata.
- Query of timeseries data stream w.r.t. metadata associations.

2 Metadata

In sensor deployments, metadata is as important as the data being collected. Without the metadata, the data is effectively useless. Therefore it is very important to not only track changes in the metadata, but take them into account when the user is querying the data. The metadata must be a first-class citizen in the system.

Like the timeseries data collected from the sensors, metadata should also be maintained as a timeseries and used in conjunction with timeseries queries performed on the data. In querying deployment data, the user has two options: 1) she queries for some information from the logical context, which may include a changing set of sensor items over the time interval of interest or 2) she queries a specific sensor item, whose logical context may change during the interval of the query. In either case if the change is not noted and accounted for, the data can be return incorrect results.

Lets consider a concrete example where temperature sensors are deployed in a room. For the sake of demonstration, lets assume that the temperature in that room is being taken by a mobile temperature sensor on cell phones of the occupants. For now we can assume that context is accurately maintained as occupants with this

Buildsys'12, November 6, 2012, Toronto, ON, Canada.

application enter and leave the room. If the user queries for the average temperature in that room throughout the day the room for all temperature sensor readings taken and take the average. It is important to only consider the temperature readings taken when the cell phone was inside that room. The user, however, should not be concerned with how this accounted for, it should occur automatically at query runtime.

Using the same scenario, lets imagine that the user wishes to know the average temperature reading taken by her cell phone throughout the day. In order to assure that the data is correctly interpreted, the query should return several averages; one for each setting as well as the aggregate average. Moreover, if the sensor was removed from the system before the upper-bound of the time interval constraint, the user should be alerted and the missing readings should not be counted in the average. These mechanisms should exist to assure the integrity of the interpretation of the data. Misinterpretation of the data can lead to gross discrepancies in the conclusions that are drawn from the data analysis and in the context of sensor deployments, where the sensor context is not always static, the analysis is even more prone to error without such treatment of the metadata.

3 References

[1] M. Balazinska, A. Deshpande, M. J. Franklin, P. B. Gibbons, J. Gray, M. Hansen, M. Liebhold, S. Nath, A. Szalay, and V. Tao. Data management in the worldwide sensor web. *IEEE Pervasive Computing*, 6(2):30–40, Apr. 2007.