SmartCIS: Integrating Digital and Physical Environments

Mengmeng Liu Svilen R. Mihaylov Zhuowei Bao Marie Jacob
Zachary G. Ives Boon Thau Loo Sudipto Guha
Computer and Information Science Department, University of Pennsylvania, Philadelphia, PA, U.S.A.
{mengmeng,svilen,zhuowei,majacob,zives,boonloo,sudipto}@cis.upenn.edu

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems and Software—*distributed systems,query processing*; H.2.5 [**Database Management**]: Heterogeneous Databases—*data translation*

General Terms

Experimentation, Languages, Performance

Keywords

sensor, stream, data integration, intelligent building

1. INTRODUCTION

As networked sensors continue to grow in sophistication and decrease in cost, we are seeing a new class of applications: those that combine data from the digital world with sensor readings, to facilitate environments that intelligently manage resources and assist humans. Examples include intelligent power grids [15], smart hospitals [14], home health monitors, energy efficient data centers, and building visitor guides.

In all of these applications, there is a need to bring together disparate data from databases (e.g., site information, patient treatments, maps) with data from the Web (e.g., weather forecasts, calendars), from streaming data sources (e.g., resource consumption within a server), and from sensors embedded within an environment (e.g., generator temperature, RFID readings, energy levels) — in order to support decision making by high-level application logic. Today this sort of data integration, if done at all, is performed by a proprietary software stack over fixed devices.

In order for intelligent environments to reach their full potential, what is necessary is an *extensible*, *multi-purpose* data acquisition and integration substrate through which the application can acquire data — without having to be coded with special support for new device types, new network types, and new datatypes. Over the past 20 years, the database community has developed a wealth of techniques for performing data integration through queries, views, and related formalisms [10]. Likewise, declarative queries have already been shown to be useful beyond databases, with extensions for distributed data stream management [1, 2, 3, 8] and sensor networks [4, 5, 12]. The key question is how to develop a unified declarative query and integration substrate, which supports a multitude of stream and static data sources on heterogeneous, possibly unreliable networks. Computation should be expressed in a single query language and "pushed" to where it is most appropriate, taking into account capabilities, battery life, and network bandwidth.

Copyright is held by the author/owner(s).
SIGMOD'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
ACM 978-1-60558-551-2/09/06.

The ASPEN (Abstraction-based Sensor Programming ENvironment) project tackles these issues, focusing on extending formalisms of data integration — schema mappings, views, queries — to the distributed stream world. We are developing (1) new query processing algorithms suitable for integrating highly distributed stream data sources, both in low-power sensor devices [13] and more traditional PCs and servers [11], (2) query optimization techniques for federations of stream processors specialized for sensor, wide area, and LAN settings, and (3) new datatypes, query extensions, and data description language abstractions for environmental monitoring and for routing information to users. In support of smart environments, we seek a *single data access layer* for integrating sensor, stream, and database data, regardless of origins. This single programming interface over heterogeneous sensors and stream sources distinguishes us from from other sensor systems [6, 9, 12].

To evaluate our work, we have been developing a showcase application: instrumenting our Computer and Information Science buildings, labs, and data centers with devices and user interfaces to improve energy efficiency, guide visitors to their desired destinations, and find free desks and laboratories. Our application, SmartCIS (Smart Computer and Information Science Building), forms the centerpiece of our demonstration. SmartCIS consists of a suite of sensor devices deployed throughout a portion of Penn's Moore building (which holds most of our laboratories), a set of "soft sensors" (monitors of logical state) running on computers, and a graphical interface and control logic. Functionality is enabled via the ASPEN data acquisition and integration substrate.

2. DEVELOPING A SMART BUILDING

One of the most compelling emerging applications of sensors is intelligent building environments: they promise to make the experience of visiting a large building or a hospital less disorienting, to make buildings or large datacenters more energy-efficient, to help occupants remember to take their medications or make it to a next meeting. A distinguishing feature of such environments, when compared with other sensor network applications, is a need to bring together database data with streaming data from the Web or Internet and streaming data from sensor devices. The task of designing a smart building can be separated into three tiers: data acquisition and integration, control logic, and a user-interface view (analogous to model-view-controller architectures).

As a testbed and showcase of intelligent environments and our ASPEN substrate, we have been developing the SmartCIS application, which monitors occupancy and locations, machine activity, and machine physical state. It also incorporates data from databases and the Web. We target two tasks: monitoring the space that people (students) use in order to guide them to destinations, and monitoring machines in order to facilitate adaptive power management or to detect failures. SmartCIS has the following capabilities:

Room monitoring. Laboratories and offices can be monitored for temperature and light on/off status. This can be used, e.g., to determine whether a laboratory is open or a room is occupied.

Machine-state monitoring. Servers and workstations run software that monitors machine activity: jobs executing, users logged in, CPU utilization, memory, number of requests being handled in a Web server application. This helps determine machine usage, including patterns *across* machines.

Workstation monitoring. Servers and workstations are plugged into power distribution units (PDUs) with Web interfaces showing current power consumption. A "wrapper" periodically (every 10s) extracts this value and sends it along a data stream. IRIS or iMote2 sensors mounted on each machine monitor its temperature. At each desk, the light-level sensor on a similar "mote" is used to detect if someone is seated in the chair.

Detection of occupants. "Mote" sensors are embedded in the hallways at major intersection points, and every 100 feet. These sensors listen for a "beacon" transmission from an active RFID device (also a mote) carried by an occupant and determine where that person is positioned in the building.

Databases and Web sources. We incorporate database information specifying the coordinates on the map of each RFID detector (the motes have no built-in positioning capability), a list of machine configurations and locations in each laboratory, and a table of "routing points" describing possible path segments and distances in the building in order to suggest routes to resources.

Based on these inputs, SmartCIS supports a variety of queries. We can trigger alarm notifications if machines exceed a temperature or load factor. We can monitor the total resources used (energy, memory, CPU) by any user or application, even across machines. We can find available machines in the laboratories, even by capability. We can determine where a visitor is located. Finally, we can do path routing in the buildings, in order to guide the visitor to a destination. Our graphical displays are located on laptops with wireless access, which may be virtually "mapped" to positions in the building. At each display, the user may select a query by adjusting controls, e.g., double-click on a machine or laboratory to monitor, or via a combo box choose a resource to locate.

3. SMARTCIS-ASPEN ARCHITECTURE

The SmartCIS system architecture consists of three major components: a graphical interface for authoring queries and returning results, which is deployed on machines in the environment; the ASPEN data integration and acquisition substrate, which includes two query runtime systems (one that enables certain computations to be "pushed" to sensor devices, and one that does distributed stream processing over PC-style servers and workstations) plus a federated query optimizer; and wrappers and interfaces over the actual sensors, databases, and machines. (See Figure 1.) Components of the ASPEN substrate are highlighted in boldface. (Ultimately ASPEN will also include support for schema mappings and query reformulation, but for SmartCIS these components are not necessary.)

Most of the research innovations are in the ASPEN modules. ASPEN takes a query (Stream SQL with extensions for devices and for routing query output to displays) and invokes a federated query optimizer that partitions it into two portions (as in Fig. 1): a subquery that will be "pushed" out to the sensor network and sensor devices, and the remaining computations that will be executed on our distributed stream engine for servers and workstations.

The distributed sensor engine, whose core features were described in [13], is novel in supporting not only aggregation and selection queries over sensor devices, but *in-network* joins between devices. This is useful in SmartCIS, for instance, when we return machine temperature data for workstations that are in use. We detect that a workstation is being used by checking for a low light-level at the adjacent chair. Hence, the most efficient query strategy is to perform a proximity-based join between temperature and light-level sensors (with a threshold applied on the light level), and to only route temperature data across the sensor network if the light threshold is met. Our sensor engine's query optimizer decides, on a sensor-by-sensor basis, where to perform the join computation.

Our distributed stream engine, described in [11], supports not only basic Stream SQL queries over windowed data, but also *transitive closure queries* that enable computation of neighborhoods and paths. The stream engine performs most of the query processing within SmartCIS, bringing together stream data, database data, and data output by the subqueries sent to the sensor engine. It is also responsible for computing suggested routes for building occupants to get to their destination — this can be done in real-time based on the occupant's current position and information about the topology of the buildings (the routing points described previously).

The federated query optimizer supports multiple underlying heterogeneous distributed query engines. Somewhat along the lines of the model established in the Garlic system [7], the federated optimizer enumerates all possible plans, and partitions these plans among the different query engines. It invokes the optimizer for each query engine over its assigned partition, and determines (1) whether this is a query plan the engine can actually execute, and (2) what the cost of the query partition would be. The novelty in ASPEN is that the cost models of the different sub-optimizers may return different cost parameters: the sensor optimizer attempts to minimize message traffic, whereas the stream optimizer attempts to minimize latency to answers. The federated optimizer must convert everything to one model, in part by making use of catalog information about the sensor network diameter, sampling rates, etc.

4. SMARTCIS DEMONSTRATION

The demonstration will combine real, live components in Penn's buildings (primarily the Moore building with our computing labs) with remote laptops on-site at the conference. See Figure 2 for a screenshot of our graphical interface and deployment.

The physical configuration of the SmartCIS devices and monitoring software at Penn are as described in Section 2. We will use laptops to display data and pose queries from the perspective of different locations within the building. Local motes will be logically mapped to RFID sensors in the hallways of the building (i.e., their data will be used in lieu of the data from the actual device). An additional mote device will perform as an active RFID beacon (transmitting a signal with low power), and as visitor approaches one of the local motes, this simulates moving in the building. The visitor will then request a set of desired features for a free machine (e.g., Fedora, Word, etc.). The SmartCIS application will plot on the GUI a route to such a machine in the laboratories.

Additionally, a visitor may click on different graphical elements of the GUI to see real-time information for the conditions in the demo area, e.g., machine state or laboratory status. Finally, we will include real-time information about the actual computations being performed: the query plan and its partitioning across subsystems and devices; wireless signal strength; remaining battery life; etc.

This demonstration will show the effectiveness of our ASPEN system adn technologies in a real application, while simultaneously highlighting the potential of intelligent buildings.

5. REFERENCES

[1] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB J.*, 15(2), 2006.

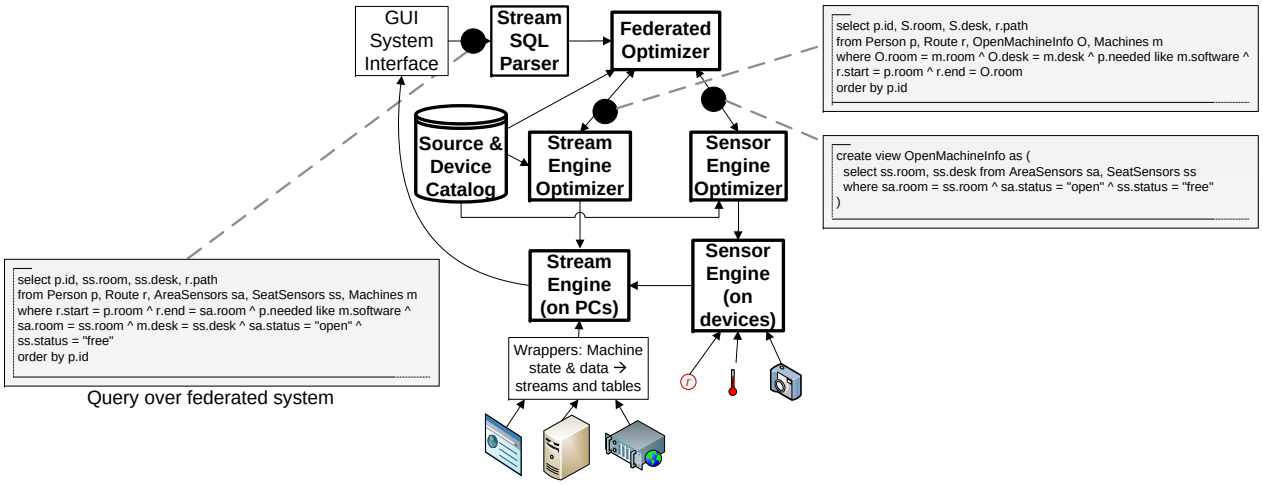

Figure 1: Architecture of SmartCIS, including ASPEN components in bold.

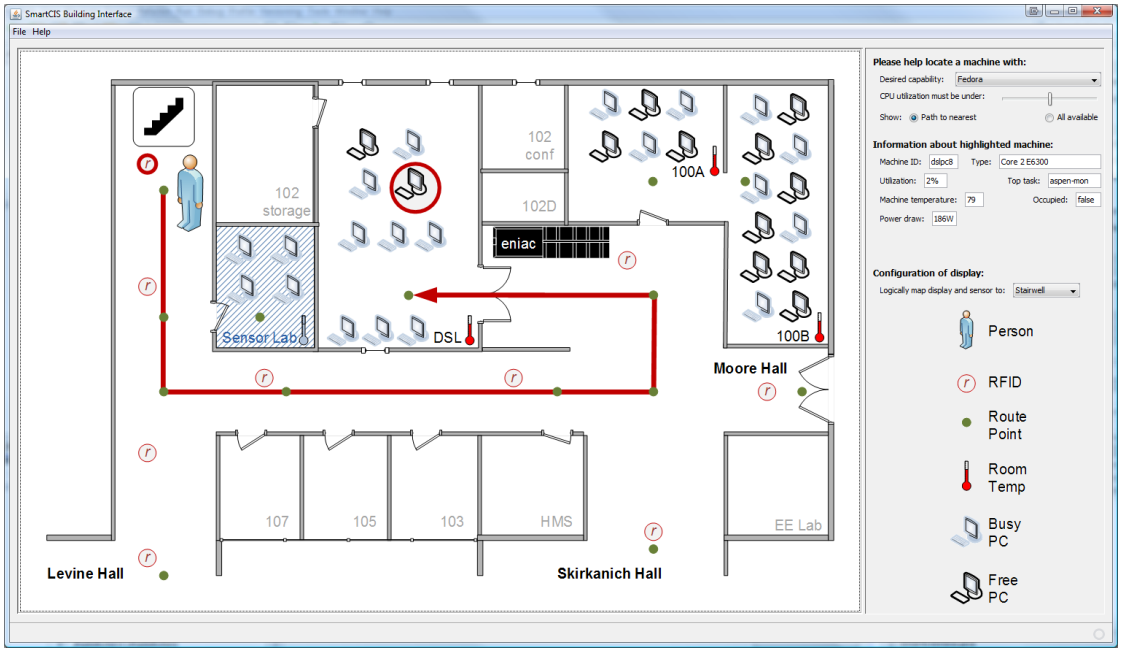

Figure 2: Screenshot of GUI showing building layout, open and closed (shaded with dashed lines) labs, free and unavailable machines, and a path to and details about the nearest machine with Fedora Linux.

[2] M. Balazinska, H. Balakrishnan, S. Madden, and M. Stonebraker. Fault-tolerance in the Borealis distributed stream processing system. *ACM Trans. Database Syst.*, 33(1), 2008.

[3] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[4] A. J. Demers, J. Gehrke, R. Rajaraman, A. Trigoni, and Y. Yao. The Cougar project: a work-in-progress report. *SIGMOD Record*, 32(3), 2003.

[5] A. Deshpande and S. Madden. MauveDB: Supporting model-based user views in database systems. In *SIGMOD*, 2006.

[6] M. J. Franklin, S. R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. W. 0002, O. Cooper, A. Edakkunni, and W. Hong. Design considerations for high fan-in systems: The hifi approach. In *CIDR*, 2005.

[7] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *VLDB*, 1997.

[8] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Quering the Internet with PIER. In *VLDB*, 2003.

[9] N. Khoussainova, E. Welbourne, M. Balazinska, G. Borriello, G. Cole, J. Letchner, Y. Li, C. Ré, D. Suciu, and J. Walke. A demonstration of cascadia through a digital diary application. In *SIGMOD*, New York, NY, USA, 2008.

[10] M. Lenzerini. Tutorial - data integration: A theoretical perspective. In *PODS*, 2002.

[11] M. Liu, W. Zhao, N. Taylor, Z. Ives, and B. T. Loo. Maintaining recursive stream views with provenance. In *ICDE*, 2009.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Design of an acquisitional query processor for sensor networks. In *SIGMOD*, 2003.

[13] S. R. Mihaylov, M. Jacob, Z. G. Ives, and S. Guha. A substrate for in-network sensor data integration. In *DMSN*, August 2008.

[14] J. V. Sutherland, W.-J. van den Heuvel, T. Ganous, M. M. Burton, and A. Kumar. *Future of Intelligent and Extelligent Health Environment*, volume 118/2005, pages 278–312. IOS Press, 2005.

[15] J. Taft. The intelligent power grid. *Innovating for Transformation: The Energy and Utilities Project*, 6:74–76, 2006. Available from www.utilitiesproject.com.