Title
An integrated cyberinfrastructure for real-time data acquisition and decision making in smart buildings and coral reef monitoring

Permalink
https://escholarship.org/uc/item/51x363f1

Author
Shin, Peter Hongsuck

Publication Date
2010

Peer reviewed|Thesis/dissertation
An integrated cyberinfrastructure for real-time data acquisition and decision making in smart buildings and coral reef monitoring

A thesis submitted in partial satisfaction of the requirements for the degree
Master of Science

in

Computer Science

by

Peter Hongsuck Shin

Committee in charge:

Professor Paul Linden, Chair
Professor Ryan Kastner
Professor Jan Kleissel
Professor Tajana Rosing

2010
The thesis of Peter Hongsuck Shin is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

Chair

University of California, San Diego

2010
DEDICATION

I dedicate this work to my family and mentors.
# TABLE OF CONTENTS

Signature Page ................................................................. iii

Dedication ................................................................. iv

Table of Contents ......................................................... v

List of Figures ............................................................. vii

Acknowledgements ........................................................ ix

Abstract of the Thesis .................................................. x

Chapter 1 Introduction .................................................. 1

Chapter 2 Background .................................................... 3
  2.1 Smart Building ....................................................... 3
  2.2 Review of Smart Building in the Industries ................. 4
    2.2.1 EnergyWise by Cisco ........................................ 4
    2.2.2 Startup Companies .......................................... 5
    2.2.3 Sensor Networks ............................................ 6
    2.2.4 Stream Data Processing .................................... 6

Chapter 3 System Requirements .......................................... 8

Chapter 4 System Architecture .......................................... 11
  4.1 Sensor deployment sites ......................................... 14
  4.2 Data Center (Building Control & UCSD) ..................... 15
    4.2.1 Downlink Node ............................................. 15
    4.2.2 Event Detection Module .................................. 16
    4.2.3 Real-Time Data Server: .................................. 18
    4.2.4 Archival Database: ....................................... 18
    4.2.5 Simulation Models for Smart Building: ............... 19
    4.2.6 Building Control Logic for Smart Building: ....... 19
    4.2.7 OPtiPortal: ................................................ 19
  4.3 Client Interface for UCSB and Building Operator: ....... 20

Chapter 5 Implementation ................................................ 21
  5.1 Technology choices ............................................... 21
    5.1.1 Programming Language .................................... 21
    5.1.2 Streaming data middleware ................................ 22
    5.1.3 Complex Event Processing engine - ESPER .......... 25
  5.2 Integration of the DataTurbine and ESPER .................. 26
LIST OF FIGURES

Figure 4.1: This figure illustrates the smart building control system. The system is composed of three major components: sensor deployment site, building control site, and the building operator. They can be abstracted to be sensor deployment site, data center site, and the user interface site. ........................................... 12

Figure 4.2: This figure illustrates the coral reef monitoring system named the Digital Moorea system. The system is similarly composed of three major components: sensor deployment site in Moorea, Data Center site in UCSD, and the client scientists in UCSB. . . 13

Figure 5.1: This Figure shows how the DT server aggregates, synchronizes and distributes the data. ................................. 23

Figure 5.2: This figure represents another key functionality of the DT. The DT server acts as the in-network ring buffer. When the DT server receives the data, the DT server keeps the data in the ring buffer until the size of the data in the buffer exceeds the size of the max size of the buffer. Then, the oldest data gets overwritten with the most recent data. The DT clients can open the connection to the DT server and fetch the streaming data in the specified time window. Since the most recent data is accessed most frequently, the ring buffer has the most recent data in the physical memory and the older data in the hard disk. 24

Figure 5.3: This figure represents how the integration software handles the data from the DT and sends them to the ESPER in a synchronized integration mode. In this example, the DT contains two channels with two data values in each channel. This figure shows the processing order of the data values. .................. 28

Figure 5.4: The three plots represent the solar irradiance activities. On the top left plot, the blue line indicates the optimal solar irradiance while the red one indicates the observed solar irradiance. While the top right one represents the ratio between the observed value and the optimal value, the bottom plot represents the difference between the two values. ............................ 31

Figure 6.1: This figure shows the simulated humidity reading and the alerts from the event detection module. The plot on the left displays the humidity readings, and the plot on the right hand shows the alerts when the air gets humid inside while the outside air is dry. 33
Figure 6.2: This figure shows the detection of absence of the data on the left plot, and the humidity channel on the right. The humidity channel did not receive any data for thirty seconds at times although it was expected to have a new reading every ten seconds.

Figure 6.3: This figure represents the reading from the depth gauge during the tsunami in Moorea.

Figure 6.4: This figure shows both the depth change as well as the tsunami alarm. The sudden change in the depth triggers the alarm on the right graph.
ACKNOWLEDGEMENTS

First, I appreciate love and support from my parents and my brother. I also deeply appreciate Tony Fountain who gave me the opportunities to work on these topics. I thank all my advisors, especially Paul Linden, for being patient with me and giving me encouragement.

This work is a culmination of collaborative efforts. The idea of integrating the ESPER into the overall system came from Sameer Tilak, and Heejin Choi helped implementing the integration pieces. I appreciate Tiffany Ford for many edits. Any unknown errors in this script are all my fault, but I give full credits to my collaborators.

This work is supported by a grant from the Gordon and Betty Moore Foundation, also NSF OCI Award 0722067, and NSF MCR LTER Award OCE 04-17412.
ABSTRACT OF THE THESIS

An integrated cyberinfrastructure for real-time data acquisition and decision making in smart buildings and coral reef monitoring

by

Peter Hongsuck Shin

Master of Science in Computer Science

University of California, San Diego, 2010

Professor Paul Linden, Chair

Buildings and environmental monitoring systems sense their surrounding conditions using various types of sensors. They process information and react to their environments in real time. The integrated cyberinfrastructure (CI) for the real-time data acquisition, real-time data processing and decision making is proposed, analyzed, implemented, and tested in this thesis. The open architecture solution is presented and implemented with open source software packages. Then, the CI is applied in two settings, the simulation and the real world. The smart building application uses the simulation environment, and the coral reef monitoring system uses the sensors deployed in the Pacific Ocean. This research is based on studies performed in Moorea. The use case scenarios and the configuration for detecting and responding to such scenarios are described for both applications.
Chapter 1

Introduction

Modern campus buildings consume large amounts of power in order to keep internal climates healthy and comfortable. Often these systems are inefficient in their power utilization; they ignore environmental conditions, variations of building usage, and available resources. Paul Linden Ph.D comments on this problem: The American school buildings are too cold in the summer and too hot in the winter. The following research provides an introduction to the intelligent sensor based control issues.

The current research proposes a real-time sense-and-control system that offers opportunities for implementing optimization policies. Such systems integrate data from environmental sensors and the buildings infrastructure, e.g., power generators and air conditioners, to make real-time decisions to balance system requirements and resources.

During the past few years in the industry, more and more companies have started incorporating these information technologies into their buildings. They provide a way to monitor the current energy usage. In addition to monitoring the energy usage, companies provide proprietary solutions to control and manage the buildings. The real-time sense-and-control system proposed and implemented in this thesis is novel to this research field: it provides an open architecture solution with the open source software components to integrate both proprietary and open software and hardware. The integrated cyberinfrastructure (CI) provides real-time solutions for the data acquisition, the event detection, and the building controls.
Although the current system lacks the implementation of the building controls due to non-technical reasons, the rest of the system is implemented and tested in this thesis. Since the generic CI does not restrict its usage in smart building only, an additional application of the coral reef monitoring system is also discussed.

The thesis is composed in the following sequence. First section provides background; it discusses the trends in the smart building industries and the real-time data management issues. In the System Requirements section, the integrated CI that can be used for smart building, more broadly, for the sense and control system is analyzed. The System Architecture section describes the three-tiered architecture that will satisfy the requirements previously described and proposes open architecture that is technology agnostic. Two system architecture figures are presented one for the smart building application and another one for the coral reef monitoring application. The two resembling systems are based on the same integrated CI. Then, in the Implementation section, the chosen technologies are explained, meeting the requirements of the system and fitting in the architecture. Furthermore, the Implementation section includes the description of the novel software components that integrate the technologies. The Application section lists several use case scenarios in simulated setting for the smart building application and the real world setting for the coral reef monitoring system. To conclude, the summary of the thesis is presented in the Conclusion section.
Chapter 2

Background

2.1 Smart Building

Modern buildings consume large amount of electricity. According to the study The Advanced Sensors and Controls for Building Applications: Market Assessment and Potential R&D Pathways [BHH+05]:

Commercial buildings are a significant and growing consumer of Americas energy resources. Americas 4.7 million commercial buildings span a great variety of functions, sizes, operating schedules and types, from large 24/7 hospitals to small retail stores. Providing the necessary energy services in these buildings (lighting, comfort, fresh air, cooking, and power for computers and other equipment) required 17.4 quadrillion Btu (quads) in 2002, 18% of the Nations annual energy use (DOE 2004a). Commercial buildings also constitute the most electric-intensive sector in the country; 76% of their energy services are provided by electricity, and they consume 35% of the Nations total electricity.

In addition, buildings account for 38% of carbon dioxide emissions in the United States, and the amount of energy consumed by the buildings is expected to grow[BHH+05].

The report also describes the reasons for a low market penetration of the smart building technology solutions in the past. The building owners face high installation cost, and the saved energy cost is usually not a big selling point to the renters who often ignore the energy efficiency of the buildings in selecting their offices.
Recently, however, with the rising costs of energy and CO2 levels, there has been huge shifts in society’s interest in green energy, green technologies, and smart buildings [BHH+05], [KPY09], [Fai09], [Ene09]. At the time of my Masters thesis proposal in early 2008, the commercial industry did not advocate smart building solutions. Since then, the commercial companies started developing their solutions. Of particular interest are the solutions being developed by two companies: Cisco and Agilewave. After explaining my approach to these problem, I explain how my solution compares with the companies.

2.2 Review of Smart Building in the Industries

Here is the excerpt from David Fisher on Building Automation and Control protocol (BACnet) [BAC09]: Around 1984 various people in the industry began voicing their concerns about the burgeoning use of networking of building automation system components using proprietary (non-standardized and often secret) communications methods. The resulting controversy prompted ASHRAE to form SPC-135P, a committee with the express charter to investigate and develop a new standard to address these issues. Through a consensus process involving nearly every major vendor of controls in North America, as well as academics, end users, consulting engineers and government interests, the BACnet standard was born after nearly nine years of effort.

An BACnet is the standard aimed addressing the needs of all building systems, and not just HVAC. The whole concept of BACnet is the facilitation of interoperability between different building systems types as well as different manufacturers.

With the burgeoning sensors and IP-based networks with green energy initiatives, there have been several new efforts and startup companies in recent years.

2.2.1 EnergyWise by Cisco

In 2009, Cisco launched a program called EnergyWise to solve and capitalize on this problem, by making its networking infrastructure into a platform for energy
Cisco EnergyWise is an energy management architecture that the Cisco company proposed and are in the rapid process of developing it. EnergyWise architecture enables measuring and controlling the power usage of various networked devices through the network. Cisco is following through the three phase development plan for hardware and software infrastructure. During the first phase (February, 2009), Cisco developed a control layer support in their switches to manage the energy consumption of IP devices such as phones, video surveillance cameras and wireless access points. In the next phase (Summer 2009), Cisco developed the support layer for personal computers (PCs), laptops and printers. In the final phase (early 2010), Cisco EnergyWise will manage the building system assets such as heating, ventilation and air conditioning (HVAC), elevators, lights, employee badge access systems, fire alarm systems and security systems. Their management system focuses on how much energy the device is currently consuming. The amount of power is regulated accordingly, changing the state of the power to on, off or standby.

Similar to my work, EnergyWise will allow companies to create event-based policies for energy reduction (i.e. turn off all lights in datacenter or hotel room unless someone swipes in - and turn off when they swipe out). It will also allow for control of the energy utilization of everything from wireless access points, all the way up to building’s air conditioning systems. Policies can be grouped by tags, so you can control entire buildings campuses or geographies.

Cisco also announced the acquisition of Richards-Zeta Building Intelligence Inc. to obtain access to the intelligent middleware to provide interoperability and integration between building infrastructure, IT applications and Cisco EnergyWise. This device is BACNet protocol compliant [Ene09].

2.2.2 Startup Companies

There are two startup companies that focus on installing sensors and controlling the HVAC and other electronic equipments through a network. The new companies include Agilewave [Agi09] and Mango [Man09]. The approach taken
by AgileWave is proprietary methods without utilizing the open protocol. They provide both software and hardware. In contrast to the closed, proprietary system, Mango published their system software to a completely open source approach. Mango focuses more on integrating various hardware pieces and customizing them for the clients needs. They implemented and published an open source version of BACnet protocol.

2.2.3 Sensor Networks

Sensor network was first started for military application in 1984 for battlefield surveillance [ECPS02]. Since then, with the advancement of computers and networks, the sensors and networking equipments have become widely available, inexpensive and miniaturized. These sensors have built-in networking capabilities or are connected to the network interface hardware.

2.2.4 Stream Data Processing

The networked sensors produce real-time, continuous data streams that are aligned in time with either explicit timestamps or implicit timestamps according to their arrival time. The time sequential ordering of the items does not change in other words, the timestamps from the sensors monotonically increase. Often, storing the entire history locally might not be possible or necessary. Furthermore, the real-time applications using the sensor net often employ the queries that are constantly running. They incrementally return new results as new data arrive [GO03]. The traditional relational database applications do not meet these requirements. The other stream data processing applications also include internet traffic analysis program and stock market analysis programs. Recently, the new stream query languages have developed based on object-oriented language, procedural language, or relation-based language [GO03].

With the advancement of the sensor network and other stream processing technologies, the design of the overall system to support the data streams becomes an important issue. The system needs to integrate various types of data streams
in a modular and scalable way. These requirements are described in the following.
Chapter 3

System Requirements

The real-time data acquisition and stream data processing is intended to serve a broad range of streaming data and event processing applications. Processing the real-time information from the sensors to optimize the building control needs both real-time data acquisition and a stream data processing system. Most of these sensor-based applications have a common set of requirements:

**Heterogeneous sensors and instruments:** The system must support the integration of numerous instruments from a variety of vendors. Because all the data is time-based observational data, the synchronization and integration across instruments usually forms the basis for processing information. The sensors include both commercial and experimental instrument packages. The system has the capability of supporting both binary data objects such as video camera images as well as the numeric data from various sensors.

**Incorporating external information:** Often the information from the sensors is not sufficient to make optimal decisions. For example, if the weather forecasts hot temperature during the summer season, the system will chill down the building during the night. If there is a particularly hot day, the system will create cooler temperatures in contrast with the other cooler days. On the other hand, incorporating the pricing information of the electricity from the energy companies might lead to different cooling strategies. Furthermore, incorporating the external models such as physics-based simulation models
may benefit the building operations. The system should support the interface with such external information in synchronized manner.

**Real-time processing:** In order to respond in a timely manner, the system must support acquisition and processing of observations in real-time. The temporal scale of processing should be adaptable to varying application demands and system characteristics. In other words, the system requires a certain degree of flexibility so that it can be adjusted and fine tuned as needed.

**Automated event detection:** Monitoring the continuous data streams and responding to them are labor intensive. For example, the system needs to automatically detect and respond to events associated with environmental phenomena and the systems overall operational state in a timely manner. Responding to the detected events may not be automated. For instance, after detecting the failure of sensors, a repairman must go out to replace or repair the sensor.

**Feedback of the processed information into the system:** The detected events or the processed data should be able to flow back into the system for further processing and logging. This processed information should be seamlessly integrated just like external information.

**Interoperability with other systems:** The system must be able to support the sharing of resources. Starting from the BACnet enabled devices, the adoption of the standard based approach allows modular and interoperable system. For the UC San Diego campus, the building control system for each building needs to communicate with central command or with each other to inform the environmental conditions. By adopting the standard based approach, it allows flexible configuration of the overall system.

**System security as a separate layer:** The security of the network devices and the building controller is critically important from an operational perspective. An ad-hoc way of building the security into the system often leads to vulnerability. The streaming data middleware and the event processing
engine are designed so that they cover two different aspects of the system. Their design must separate the security layer for themselves and for the integration of the two. Furthermore, the implementation of each layer must consider the security layer as well. In this study, we chose technologies that deal with security at a different layer. This separation of the security layer is important because it leads to a clean design of the overall system. NASA and others used a mixture of hardware and software solutions as a security layer apart from the data transport or data processing layer [FTH+09]. ESPER technology also has similar way of securing the system at a different level [ESP09]. The possible limitations of combining the two technologies are described in the Implementation section.
Chapter 4

System Architecture

Following the modular design principle, the overall system architecture is divided into three major parts. First, the sensor deployment sites are described for both the smart building application and for the MCR field station. This is followed by the description of cyberinfrastructure system of the Data Center. Currently for the MCR, such architecture is functional and is in semi-production mode while for the smart building, the logistical challenges limit the real world practice. Lastly, various client applications are described.
Figure 4.1: This figure illustrates the smart building control system. The system is composed of three major components: sensor deployment site, building control site, and the building operator. They can be abstracted to be sensor deployment site, data center site, and the user interface site.
Figure 4.2: This figure illustrates the coral reef monitoring system named the Digital Moorea system. The system is similarly composed of three major components: sensor deployment site in Moorea, Data Center site in UCSD, and the client scientists in UCSB.
4.1 Sensor deployment sites

The Figure 4.1 and Figure 4.2 show the two deployment sites one in a building, and the other at the Gump Research station for the MCR project. Both applications contain two major components: a suite of sensors and an Uplink Node that aggregates and sends the data streams to the Data Center.

For the smart building application, the sensors are expected to be deployed both indoors and outdoors as shown in the Figure 4.1. For UCSD classroom buildings, the temperature, humidity, carbon dioxide and motion sensors are expected to be deployed uniformly in each classroom while the office rooms will have their own temperature and humidity sensors. The sensors will stream the observed values into the Uplink Node which could be deployed on each floor of the office building or one for the entire building. The Uplink Node transmits the observed data to the Data Center. In addition to the sensors inside the building, the outdoor conditions are critical in taking advantage of the natural source of heating and cooling.

In the DEMROES project [DEM09], temperature, humidity, wind direction and speed, solar irradiance sensors are mounted on the roof top of the buildings across the UCSD campus. It is powered by a combination of battery power and solar power, and the solar panel conditions are also monitored. The data logger, CR1000, from a Campbell Scientific company collects the observed information and communicates via wireless modem to the server.

Similar to the smart building application, the coral reef monitoring project employs two major components at the field station: a suite of sensors and the Uplink Node. The station employs three major types of the instruments: a weather station, a IP-based video camera, and the Conductivity Temperature Depth (CTD) gauge. The weather station has the same data logger, CR1000, which the DT interfaces with in the DEMOROES project. The CTD is deployed with a long wire about 30 meters away from the station at the 3.5 meter water depth. The IP-based video camera is installed on the roof of the station about ten meters away from the weather station at a height of 4 meters.
4.2 Data Center (Building Control & UCSD)

The key design concepts for the Data Center cyberinfrastructure are modularity, scalability, and open architecture. These design concepts are applied to the cyberinfrastructure at the Data Center. The architecture of the system closely follows the commercial enterprise computing models. Most of the specific software packages chosen as the components in the system are adapted from the commercial, yet open source software communities. Open architecture allows swapping the technology without affecting the other components. Therefore, this system should be viewed as more of an evolving system with new technologies.

Before describing each component in the system, it is necessary to describe the virtual computing platform first. In our applications, the system does not have the constant load. When the event of the interests is detected, the compute intensive model can be started, and the demand for computing power could be increased for a certain period of time. Another demand might be from the users perspective. Querying the database or the system periodically may increase the demand on the physical computing resources. As the demand of the system grows, the system can migrate from the low resource (CPU, memory, or hard disk space) node to the high resource node. The cloud computing provides the cost effective, on-demanding computing resources. For this project, we employed the Xen hypervisor as the virtual machine software. On the virtual platform, the Redhat Linux is chosen as the software Operating System (OS).

Both applications share the four major components of the system: Downlink Node, event detection module, real-time data server and archival database. Each component has different functionality in the open-architecture design concept. Each component needs appropriate adaption of both hardware and software. They are described below.

4.2.1 Downlink Node

The Downlink Node communicates with the Uplink Node mainly to stream in the sensor data. In addition to this purpose, the system needs to monitor
the health of the Uplink Node and the Downlink Node. Hence, the Downlink Node should support the maintenance and operational aspect of the system. The Downlink Node contains hardware pieces that cover the networking aspect and the computing aspect. From the system engineering point of view, the Downlink Node is more stable and more compute intensive than the Uplink Node.

Here is an example of the hardware pieces in the MCR project. The Downlink Node employs a universal power supply, a router, a special networking device that converts the Ethernet protocol back to a RS-232 Serial protocol for the networking aspect. As a computing machine, the Campbell Scientific data logger (CR1000) software requires a Microsoft Windows XP based machine to communicate. Hence, the Downlink Node contains the Windows based PC.

The specific software components of the Downlink Node vary according to the hardware associated with the Uplink Node and the sensors. In other words, given the specific hardware, the drivers will need different software platform (e.g. Windows vs. Linux). From the system engineering perspective, the streaming data from the Downlink Node should a middleware. The streaming data middleware allows adding an abstract layer to integrate these different and possibly proprietary software components.

4.2.2 Event Detection Module

Making decisions with the streaming data requires a module that processes the streaming data in real-time. The event processing engine and the sensor network applications started with military applications. They needed the environmental information to make a real-time decision. Recently, the financial market and business industries started adopting the event driven architecture as well [ESP09], [Li05]. Specifically, the Event Streaming Processing (ESP) and Complex Event Processing (CEP) engines are developed to process the data in real-time. A timely response to the events is critical, and the traditional database was not designed for this. The specific software package chosen for this task is described in the Implementation section.

The event detection module serves two major purposes in the real-time
decision making system. It detects the environmental phenomena and monitors the state of the system. The following examples for the applications motivate the need for the event detection module.

In smart building application, taking advantage of environmental information from the inside and the outside of the building may lead to reducing the overall energy consumption. For example, when it is hot and humid in the rooms and the outside air is dry and cool, the HVAC system can intake the outside air without using the air conditioners. Furthermore, if the weather forecast predicts it will be very hot on the next day, the building can be cooled more than the usual preceding night.

The coral reef monitoring project has three examples for detecting the environmental phenomena. The scientists want to detect the coral bleaching conditions, which are correlated to the rapid rise in temperature during periods of intense sunlight. They also want to study the effects of the freshwater runoff from the pineapple fields. The freshwater from the rain and the salinity of the seawater indicates the proper time to go out and take the sample from the seawater. Lastly, the environmental phenomena characterize the periodicities in temperature caused by internal waves. In this case, the temperature change measured at different depth and locations over a specific period indicates the internal waves.

For both applications, the operational manager needs to check the sensor failure and sensor calibration error events. As the number of sensors grows, this task prohibits manual checking and calibration. The event detection module can automatically detect these scenarios. The real-time detection of thermostat failures and other sensor calibration errors may lead to rapid response in maintaining a healthy system for both building control application and coral reef monitoring science project.

When these events occur, the system assists making decisions or triggers automatic responses. It often requires more computational power. The on-demand cloud computing platform satiates the need for varying computation power. The system adopts the ESP and CEP concepts from the industry. The specific details of the packages are mentioned in the Implementation section.
4.2.3 Real-Time Data Server:

The real-time data and events from the Downlink Node and from the event processing module need to be distributed to the users and other applications. The Data Center provides the publish-and-subscribe service for them. As the number of users and applications fluctuates, the computational demand for the real-time data server changes. Balancing the load of the components is critical in order to scale up the overall system. By decoupling the load from the users and the applications, the Downlink Node focuses on stability and reliability of the incoming data streams. Running on the cloud computing platform, the real-time data server adapts to the changing demands. The real-time data server serves the mirrored data from the Downlink Node and the event streams.

4.2.4 Archival Database:

From the system architecture point of view, the archival database provides two functionalities. It supports complex query over the entire data. Furthermore, because it can transform or summarize the data flexibly, it also plays an important role in sharing the data in a standardized way. The database provides the Structure Query Language (SQL) or sometimes Online Analytical Processing (OLAP). Although the real-time data server may contain all of the data, it does not provide various ways of aggregating, summarizing, and analyzing the data. For smart buildings, the database queries enable summarizing the energy usage in quarterly or seasonal periods of time very easily. In studying coral reefs, the environmental phenomena can be summarized using time window, the data values or combination of both. The database query provides ways to transform the data. When the building users or the scientists want to share the data or summary of the data, the database plays a big role in transforming the data. The coral reef scientists often publish their data using netCDF or the OpeNDAP server [Ope09], and by using the DB, it is easy to translate the data to their format.
4.2.5 Simulation Models for Smart Building:

The simulation models provide additional information in assessing the current situation or predicting the future. The observational information may feed into the simulation model, and the model may output the extra information back into the system. Or, the model may not need the observational information. In the DEMROES project, the optimal solar irradiance is calculated to predict the appearance of clouds. The calculation only needs the spatio-temporal information. From the system perspective, each simulation model differs in its computational requirement as well as in the required response time. With the current architecture, the cloud computing platform provides the scalability and the flexibility in computing power that the simulation requires. The streaming data middleware architecture also enables scaling up the process because it can buffer large amount of the streaming data. By employing the data streaming middleware, it can handle bursts of data streams in both sampling rates as well as the number of the data streams. The simulation model can potentially create these bursts.

4.2.6 Building Control Logic for Smart Building:

This component is a mix of the traditional building control system such as Johnson Control system, and the interface to the system such as Mediator from Cisco. This component used to be a proprietary component. However, as the number of sensors and their vendors grows, the market is shifting the attention to the standard protocol which is mentioned in the II. Background Section. From the system architecture point of view, the standard protocol is the way it interacts with other components.

4.2.7 OPtiPortal:

Visualizing plays a key role in understanding the environment using the sensors. The command center often has large number of tiled displays. Visualizing and manipulating a large number of data streams becomes a difficult issue. At the CalIT2 in UCSD, the OPtiPortal provides a solution to visualizing large number
of data streams [Opt09]. The visualization system is also a tiered architecture that utilizes the streaming data middleware. In the real-time decision making system, the OPtiPortal is viewed as a visualization engine and the interface.

4.3 Client Interface for UCSB and Building Operator:

For both smart buildings and the coral reef monitoring system, client applications need to provide functionalities to view the data, interact with the observed data, and summarize the historical data. The energy usage monitor and the data viewer provide a way to monitor the streaming data. The building control and the scientific models and analysis use the streaming data as input to the processes. The DB queries allow various ways to manipulate the historical data. The tiered approach using service as a way to interface with each component is being explored in our applications. Such services include serves multiple clients.
Chapter 5

Implementation

In the previous chapter, the architecture was proposed. Following the design principles, the appropriate technologies for each piece are identified in this chapter. The system is composed of the modular, free and open source packages that enable a few proprietary components to be integrated into the generic cyberinfrastructure framework.

5.1 Technology choices

In order to implement the most generic cyberinfrastructure, a platform independent programming language is necessary, along with an open source data stream middleware and stream processing engines.

5.1.1 Programming Language

JAVA and Jython are chosen for the programming language. JAVA: JAVA is a free, open-source, object-oriented language that is platform independent. The components in this system may run on physically different platforms. In order to maximize the compatibility and portability, JAVA was chosen.

JYTHON: In addition, JYTHON is used. Jython is a JAVA implementation of Python. Jython has several advantages over JAVA. Because of Python syntax, the coding time can be reduced with less number of lines. Furthermore, because it
is based on JAVA, all the JAVA libraries and other JAVA packages can be accessed. In addition, for array manipulations, object type is explicitly defined, which keeps the runtime performance of the programs written in Jython similar to the ones written in JAVA. The disadvantage of using Jython is the startup time of about two seconds. The requirements of our system indicate a continuously running program that will handle various error cases; they will not need the fast startup time or the frequent restarts.

5.1.2 Streaming data middleware

DataTurbine: Ring-Buffered Network Bus (RBNB) DataTurbine (DT) was developed and owned by Creare Inc. in the 1980s. With various observing system initiatives, the executives at Creare have release the DataTurbine into open source software product after collaborating with San Diego Supercomputer Center (SDSC) at UCSD for several years.

The software product provides the basis for developing robust streaming data middleware. As shown in the diagram below, the DT has two main functionalities. (1) The DT aggregates and synchronizes across various types of incoming data streams, and acts as a network bus. (2) The DT stores the streaming data in the ring buffer, which allows the historical data to be re-streamed into various applications.

Figure 5.1 shows how the DT server aggregates, synchronizes and distributes the data. In order to insert the data into the DT server, the user must develop data source clients. Every data point has a timestamp and a data field. The DT server synchronizes the incoming data streams from the different data sources according to their timestamps. The DT supports both the numerical type such as an integer and a floating point, and the binary object type such as an image or binary data. Once the DT server receives the data, the data streams can be directed in multiple ways. Thus, the DT server acts as a multiplexer.

As a common sense, the time only moves forward. In the observing system, the value of the Unix timestamp from the sensors (the number of seconds since January 1st, 1970) only increases as time goes by. The DT assumes and expects
the timestamp to be monotonically increasing. With this assumption, the DT can move the data streams at low latency, and can synchronize across all the data streams efficiently. Low latency and efficient synchronization are the key elements in building scalable observing system. The DT scales up to kHz range, and can handle indefinite number of data streams. These are both limited by the hardware constraints rather than the software aspect.

Having the in-network buffer enhances overall robustness. In real world applications, more than one DT servers are connected to each other to stream the data. For example, there could be two DT servers: one server near the sensors and the other server at the central command location. The low-powered computers deployed near the sensors in the building or in the wilderness might have less computing power and memory. However, because of its proximity to the sensors, the likelihood of having network problems in between them is low. With a reliable
Figure 5.2: This figure represents another key functionality of the DT. The DT server acts as the in-network ring buffer. When the DT server receives the data, the DT server keeps the data in the ring buffer until the size of the data in the buffer exceeds the size of the max size of the buffer. Then, the oldest data gets overwritten with the most recent data. The DT clients can open the connection to the DT server and fetch the streaming data in the specified time window. Since the most recent data is accessed most frequently, the ring buffer has the most recent data in the physical memory and the older data in the hard disk.

In addition, the tiered architecture with ring-buffer embedded in different places allows load-balancing and ensures network efficiency. Extending from the prior example, if the applications require lots of clients to receive the data streams, the low-powered computer near the sensors should not be the one serving up many requests from the clients. A DT server with high network bandwidth and a big memory at the command center should distribute the data. By creating the network of DT servers with appropriate hardware resources at the different locations, the load balancing can be achieved. In addition, if a certain portion of the network...
has a limited bandwidth, the DT servers can minimize the network traffic on the limited bandwidth line by serving the data through high bandwidth lines.

5.1.3 Complex Event Processing engine - ESPER

The streaming data synchronized through the DT server needs to be processed in order to help make decisions for the overall system. As described in the requirement section, the system must respond to the environmental conditions in real time. The complex event processing engine plays a critical role in processing the information in a timely manner and creating a decision support system.

According to the ESPER website, an event is defined as an immutable record of a past occurrence of an action or state change [ESP09]. Relating to the DT, the event is one observation with one timestamp. A stream of events is a sequence of events, ordered in time. The streams of events are provided through the streaming data channels from the DataTurbine. The event processing engine transforms, filters, detects patterns and derives other events from the incoming event streams. The detected events can flow back into the DataTurbine as the streaming data. The term complex in complex event processing engine refers to the interaction and the integration of more than one event stream.

Understanding the streaming data comes from analyzing the temporal sequence aspect of the data and correlating the streaming data across the channels. ESPER is designed to detect the temporal trend of the sequence of the data over a sliding window, within one channel or across multiple channels. It is also designed to process the streaming data in real-time, which leads to reacting to the detection with minimum latency.

ESPER provides a Event Processing Language (EPL) which allows expressing rich event conditions, correlation, possibly spanning time windows, thus minimizing the development effort required to set up a system that can react to complex situations. [ESP09]. ESPER is written in JAVA, and it can be embedded with J2EE, Plain Old JAVA Object (POJO), or Enterprise System Bus (ESB).
5.2 Integration of the DataTurbine and ESPER

Most of the functionalities of the DT and of the ESPER compliment each other. Both technologies handle the streaming data, according to the timestamp of the data. Both technologies assume that the streaming data within one channel arrives in order - the timestamp is monotonically increasing. In addition, both offer two options to control the time: the internal timer based on the Java Virtual Machine (JVM) and the external timer specified by the application. Through the control of the timer and the timestamp, the streaming data with the timestamps can be synchronized and passed on to each other.

In order to integrate the two software components, the control of the timer must be explained first. When the data is inserted into the DT server, the user may supply timestamp from the sensor, another external device, or the system wall clock from the Java Virtual Machine (JVM). Similarly, the ESPER also puts the timestamp on the data either using the system clock or by a user specification. In addition to the internal timer, ESPER introduces the idea of time resolution in order to support the time window based queries. Philosophically, the time flows continuously, but in the computer system, it needs to be discretized. In plain English, it can be thought of as a clock tick. The period of one tick can be specified in the ESPER. The ESPER sets the temporal granularity to 100 millisecond by default; in other words, the time proceeds 100 milliseconds at a time.

If the small latency between the two technologies can be ignored, the streaming data can be passed onto the ESPER engine directly using the internal timer of the JVM that ESPER is running on. In this case, the timestamp supplied from the DT is ignored. Only the data values from the DT will be used. Using this approach, the rate of the streaming data can be measured and used. The application for this mode includes monitoring and detecting the overall system health monitoring. For example, if the outside temperature reading has not been received for more than thirty minutes while other sensors have been sending the data, the sensor might have failed. Or, if all the sensors have not been sending over in the data, the network might be down.

The other way to integrate the two is to use the timestamp values supplied
by the DT as shown in Figure 5.3. In this integration approach, the user takes control of the current time of the ESPER engine. The user constantly supplies the current time to the ESPER engine when streaming in the data. Instead of having a fixed time resolution, the user specifies the current time each time.

Incrementing the time according to the timestamps received implies that the integration component will synchronize the time across all the DT channels that are used in ESPER engine. In other words, if one of the channels did not receive the data, the time cannot proceed. The below Figure 5.3 illustrates how the current time gets set according to the timestamps received. There are two channels from the DT, and they are sampled at a different time. The ESPER increments the time by ordering all the channels. It only intakes the streaming data up to the earliest point between the two channels. This approach buffers the data and synchronizes across the channels. This buffering causes two potential drawbacks. In terms of performance, the sorting time across the channels take time. However, since all the data within one channel are sorted, finding the minimum timestamp value can be computed using constant time $O(n)$. The other issue is the size of the buffer. The worst case scenarios is when there are large number of sensors with a high sampling rate where one channel stopped streaming in the data. This situation will eventually create the out of memory error. The user of this program must be aware of this issue, and configure the program appropriately.
Figure 5.3: This figure represents how the integration software handles the data from the DT and sends them to the ESPER in a synchronized integration mode. In this example, the DT contains two channels with two data values in each channel. This figure shows the processing order of the data values.
5.3 Connecting various sensors to DT

It is a challenging task to interface with many different sensors. From the
data receiver point of view, sensors vary in their data format and their commu-
ication styles. One of the goals of the streaming data middleware is to handle and
coordinate the different types of the data from the different types of the sensors.

First, the format of the data can vary from a numerical data representing
a humidity reading to a binary object representing a scene. The DT streaming
middleware supports all the common types (int, float and binary object type)
of the data. For example, in the MCR project, both the binary image objects
from the IP-based video camera and the numerical data from the weather station
and the Conductivity Temperature Depth (CTD) sensor are inserted into the DT.
Similarly, in the DEMROES (Decision Making using Real-time Observations for
Environmental Sustainability) project, the numerical data are streamed into the
system.

Next, in terms of how the sensor samples and communicates the data, the
sensors can be classified as dumb sensors or as smart sensors. The smart sensors
usually have bidirectional communication capability, supporting the users com-
mands through a known protocol. For the MCR project, the Seacat Conductivity
Temperature and Depth (CTD) sensor (Seacat CTD 16plus) from the Seabird
company communicates using RS-232 serial protocol. A device driver is written in
Jython to communicate using the serial protocol specification. The dumb sensors
usually do not provide any digital communication methods, and they often have
an analog voltage response given the input power. Often the temperature sensors
in the building are this way. In order to bring the information online, the dumb
sensors are usually connected to a data logger which converts the analog voltage
into a digital value. The data loggers often support the digital communication
through a serial or the Ethernet interface. Both the DEMROES and the MCR
projects deployed a Campbell Datalogger, which shared a common driver written
in Jython.
5.4 Model integration using MATLAB

External programs such as physics based simulation models might provide useful information for making appropriate decisions. The DT framework provides easy integration with the information other than the observed data through the timestamp information. For example, the DEMROES project adopted a computational model from NASA on computing the optimal solar irradiance of the given location and the time. The DEMROES project explores the possibility of building a solar production forecast system. The production of the solar power is quickly disrupted when the clouds cover the sky and when the solar panels do not receive the irradiance. If the solar power were to be an integral part of the current electrical grid system, a sudden drop in the production of the solar power must be countered with other means of the electricity production. If the drop in production can be predicted, then the grid system can be notified, and take the immediate action of increasing the production of other means of electricity [DM07]. One way to predict the coming clouds is to compare the optimal solar irradiance and the current solar irradiance. Prior to the shadow on the solar panels, the solar production often exceeds the optimal value because the reflection from the clouds adds on to the current irradiance level. This scenario requires fusing the data between the current observation and the computed numerical values through the timestamps.

The Figure 5.4 shows the interaction of the DT with the external program. When the observed solar irradiance data is streamed into the DT, the Matlab program subscribes to the data through the DT to Matlab interface. Matching the timestamp of the observation, the program computes the optimal irradiance at the current time, and computes the ratio and the difference between the observed value and the optimal value. All the computed values are streamed back into the DT. For example, the blue line indicates the optimal value while the red line indicates the observed values on the top left plot.
Figure 5.4: The three plots represent the solar irradiance activities. On the top left plot, the blue line indicates the optimal solar irradiance while the red one indicates the observed solar irradiance. While the top right one represents the ratio between the observed value and the optimal value, the bottom plot represents the difference between the two values.
Chapter 6

Applications

6.1 Smart Building

In this section, we illustrate how the real-time event detection from the streaming data may lead to saving energy in the buildings. Because applying the automated control in the real buildings entails various technical and non-technical issues, the researchers simulated the data streams from the sensors and other external information. The system uses the simulation to detect the events. Simulating all possible cases is redundant. Instead, we categorize the types of the event detection queries and show the detailed example for each type. From the system point of view, however, the queries can be classified as either an environmental phenomena type of queries or a system state of health monitoring type of queries. The event detection module is designed to handle both types of usages. The queries that detect the environmental phenomena for smart buildings are listed first. For all the experiments of the smart building applications, Matlab programs are created to generate the streaming data. The first type of the queries detects an event based on the specific threshold over a specific period. This example illustrates the use case. The temperature and the humidity in the classrooms are not closely monitored in the school buildings - periodic heating and cooling from the central system often lead to uncomfortable classrooms for the students and the inefficient usage of energy. Maintaining the proper temperature and the humidity of the rooms can be done using real-time sensing information. A concept is simple. It is
similar to opening the window to let the air out. If the outside air is cool, and the classrooms are hot, the HVAC system can intake the air outside without using the air conditioning system. The Figure 6.1 illustrates such conditions. When these conditions are detected, the central system will use the fan to intake the outside air.

Figure 6.1: This figure shows the simulated humidity reading and the alerts from the event detection module. The plot on the left displays the humidity readings, and the plot on the right hand shows the alerts when the air gets humid inside while the outside air is dry.

The following ESPER query detects such events.

```sql
insert into humInEvent select inside_humidity > 55 as status from Inside_Humidity.win:time(20 seconds)
insert into dryOutEvent select 51 > outside_humidity as status from Outside_Humidity.win:time(20 seconds)
insert into humInDryOutEvent select 1 as status from pattern [ every ( humInEvent(status=true) and dryOutEvent(status=true))]
insert into humInDryOutEvent select 0 as status from pattern [ every
```
(humInEvent(status=false) or dryOutEvent(status=false))]

More generally, the query depicts a simple case of detecting an event using a threshold value and a time window, as mentioned. This type of query can be applied to many settings. For example, if it is sunny outside and the room has windows with shades that can be controlled, the shades can change the shades to let the sunlight come into the rooms. If it is sunny outside and the room is hot in the summer time, the system can be used to block out the sun. This can be done using the solar irradiance sensor on the building and the temperature sensors in the rooms. The system can also utilize the external information other than the sensor data. For example, the energy price fluctuates throughout the day. The system can process the energy price in real-time to decide if the price is very high and respond by adjusting the target temperature (thermostat) to be a couple of degrees higher than the normal setting when cooling the building. The real-time pricing from the energy companies is often available on the web. Similarly, based on the next day weather forecast information during the summer season, the system may decide how much to cool down the building. Most of these buildings use the air conditioner to cool the building at night during the summer when the demand and the price of the electricity is low. The system can detect that it will be hot and sunny the next day and cool down the temperature a few degrees more than the regular temperature. If it will be cool the next day, the system can set the temperature to be a few degrees higher than a regular day. From the systems point of view, the above cases all respond to environmental phenomena in order to save energy. The event detection can be also applied to quality assurance and quality control (QA/QC) settings. ESPER also allows the users to compute the statistical operation on one or more channels to create an event. For example, the system can detect the sensor failure or the calibration error if the multiple sensors deployed at the same site do not have the highly correlated observations. Although this scenario will fail if all the sensors drift all to the same value at the same time, this scenario has the merit of providing the quality assurance to the observed data. The example of the using the correlation is described in the next section when detecting the low
correlation of the temperature and the conductivity. In the system level, the event
detection modules can check the current health of the system. The system may
check the absence of the expected data over a period. The absence of the streaming
data implies some type of failure - the network, the power or the sensor failures.
Lack of sensor information would lead to wasting energy. Therefore, mitigating
the effects through real-time detection and responding quickly to it would lead to
energy efficiency.

In this example, the simulation program generates a humidity reading every
ten seconds, and occasionally holds out generating the data for thirty seconds. The
Figure 6.2 shows the detected absence of the data for those periods. Unlike the
threshold examples, for this case, the integration between the DT and the ESPER
must use the wall clock to see if the data has arrived according to the wall clock
time.

![Figure 6.2: This figure shows the detection of absence of the data on the left plot,
and the humidity channel on the right. The humidity channel did not receive any
data for thirty seconds at times although it was expected to have a new reading
every ten seconds.](image)

The following query detects the failure of receiving the data for a specified
period of time.
Observing the patterns in the system metric streams can aid in monitoring the system state of health events. Events can indicate (1) catastrophic failures, (e.g., loss of a network link), or (2) calibration problems, (e.g., sensor or clock drift), or (3) trends that threaten the quality of service (QoS) agreements, e.g., buffer saturation, server loads, or transmission delays. Similar to the environmental events, the responses to state of health events can be simple. For example, send a technician to fix the sensor, or develop sophisticated load balancing through the migration of virtual machine in a cloud.

6.2 Coral reef monitoring

The Coral reef monitoring system also needs real-time event detection. The oceanographers want real-time event detection to help their science. As before, the events of interests can be classified as either environmental phenomena, or system state of health conditions.

Here is the description of the coral reef monitoring system, the Digital Moorea system [FTS+09]:

The Moorea Coral Reef (MCR) Long Term Ecological Research [MCR09] site is the only member of the U.S. LTER network [LTE09] focused on coral reefs. The MCR site is the coral reef complex surrounding the island of Moorea, French Polynesia. Moorea is a small, triangular volcanic island 20 km west of Tahiti in the Society Islands. An offshore barrier reef forms a system of shallow (mean depth 5-7 m), narrow (0.8-1.5 km wide) lagoons around the 60 km perimeter. All major coral reef types (e.g., fringing reef, lagoon patch reefs, back reef, bar-
rrier reef and fore reef) are easily accessible for study. Field research is conducted at the UC Berkeley Gump Research Station [Gum09].

Sensor deployments in the lagoons and on the fore reef surrounding Moorea began in 2005. Currently there are six instrumented sites (two per side of the island) and a seventh site located in one of the deep-water bays on the islands north shore. All data collection and processing is performed by the sensors and data are stored using internal memory. SCUBA divers retrieve the sensors every few months to download data, replace batteries, clean housings, and then redeploy on the reef. These sensor deployments provide the foundation for building the Digital Moorea real-time sensing system.

One of the sites in Moorea near the Gump Station started streaming the real-time data in May 2009. The site contains a CTD, a weather station and a IP-enabled video camera. The detailed description of the deployment is described in [FTS09].

Observing patterns in the environmental sensor streams can lead to detecting environmental events. We collaborate with the MCR scientists to create these three event detection queries 1) to detect the nutrient runoff, 2) to check the quality of the salinity observations, and 3) to detect the tsunami. They are currently deployed, and they have been continuously monitoring the data streams over six months.

When it rains in Moorea, the freshwater runoff from agricultural fields, especially pineapple farms, introduces foreign nutrients into the salt water around Moorea. The scientists need to collect the salt water samples in the bay when the fresh water runoff occurs. The event detection system alerts the biologists to find the proper time to go out and sample near the bay. The system can detect the freshwater runoff based on the temporal correlations between salinity and rainfall measurements. The scientists proposed the two ways to detect the low salinity and high rain.

6.2.1 Detecting the freshwater runoff condition

The freshwater runoff can be detected if it has been raining for awhile and the salinity of the salt water drops low. The scientist proposed detecting the
following conditions:

- Salinity is less than 35.5.
- The rate of the rain is greater than 3 mm/5 min, and such condition is sustained for one hour.

The following query captures such logic:

```sql
insert into LowSalinityEvent select 36.5 > salinity as status from Salinity
// LowSalinityEvent indicates the result of the salinity threshold comparison.
insert into HighRainEvent select sum(rain) > 3 as status from Rain
  .win:time(5 min)
// HighRainEvent computes the sum of the rain over the five minute period.
insert into LowSalinityHighRainEvent select lse.status AND hre.status as status from LowSalinityEvent .std:lastevent() as lse,
                                  HighRainEvent .std:lastevent() as hre
// LowSalinityHighRainEvent indicates when both values are true
insert into LowSalinityHighRainForHourEvent select 1 as status from pattern [ every LowSalinityHighRainEvent (status=true) -> timer:
                            interval(1 hour) and not LowSalinityHighRainEvent (status=false) ]
// LowSalinityHighRainForHourEvent becomes 1 if the low salinity and high rain condition defined before has not been false for one hour.
insert into LowSalinityHighRainForHourEvent select 0 as status from pattern [ every LowSalinityHighRainForHourEvent (status=1) -> timer:
                            interval(1 hour) and LowSalinityHighRainEvent (status=false) ]
// If the low salinity and high rain condition did not hold true for an hour, insert 0.
```

In this ESPER query, the keyword pattern is introduced to match the temporal sequence of events. The arrow indicates "followed by" that the event prior to the arrow is followed by the event after the arrow. The keyword every indicates a match in the query conditions on each LowSalinityHighRainEvent.
6.2.2 QA/QC on Salinity

The salinity values are not directly measured using the sensors. The sensors observe pressure, temperature and the depth. Based on those three observations, the salinity is computed. The oceanographers would like to check it in real-time to see if the derived value is within the reasonable range. In this query, the statistical property, a correlation, is computed by measuring the across the two data streams. Here is the property of the query:

- Compute the moving correlation between the temperature and the conductivity.
- Use the threshold criterion for correlation.

```sql
insert into CTLowCorrelation select 0.5 > correlation as status from AverageConductivityTemperature.stat:correl(conductivity, temperature)
// The .stat:correl notation computes the correlation between the two data streams.
insert into QAWarning select 1 as status from pattern [every (CTLowCorrelation(status=false) -> CTLowCorrelation(status=true))]
// The above query sends a warning if the result of the threshold changes from false to true.
insert into QAWarning select 0 as status from pattern [every (CTLowCorrelation(status=true) -> CTLowCorrelation(status=false))]
// This query is the same as above except that the condition changes from true to false.
```

6.2.3 Tsunami detection

In this example, the scientists wanted to detect the tsunami events using the CTD sensor. Although the sampling frequency is only at 1 sample per 2 minutes, the change of the depth was significant enough so that the tsunami events were detected at the correct times. The threshold values are chosen based on the historical data from the deployment. French Polynesia and Samoan islands
regions experience tsunami waves which are mainly caused by earthquakes. On the September 29, 2009, an 8.3-magnitude earthquake struck the Samoan Islands at 7:48 a.m. (French Polynesian time). As a result of the earthquake, five small waves between 25 -70 centimeters (9.9-27.6 inches) were measured off Papeete between 11:10 a.m and noon Tuesday according to the Tahiti Presse [Tah09]. The coral reef monitoring system, namely Digital Moorea system, also captured this tsunami in Moorea, one of the French Polynesian islands. The CTD sensor is deployed in the water about 30 meters away from the Gump lab located in the Cooks Bay, in approximately 3.5 meters depth. The earthquake resulted in unusually big fluctuations in pressure measured by the CTD. Typically, pressure changes due to surface waves were less than 0.05 decibar (1 decibar ≈ 1 meter) before arrival of the tsunami. However, the CTD captured a pressure change over 0.1 decibar for twenty minutes beginning 11:09 local time. Tsunamis typically last periods of 20-30 min, consistent with these observations. Fluctuations due to the tsunami are superimposed on those due to wind waves in the graph below.

The following ESPER query captured the tsunami event.

- Calculate the moving standard deviation on the streaming data received in last ten minutes.

- If the standard deviation is higher than 0.035 which is nine times the average standard deviation, flag it as an unusual fluctuation.

```
insert into PressureEvent select stddev (Pressure) as psdev from Pressure
insert into PressureHighFluctuationEvent select 1 as status from PressureEvent where psdev >= 0.035
insert into PressureHighFluctuationEvent select 0 as status from PressureEvent where 0.035 > psdev
```
Figure 6.3: This figure represents the reading from the depth gauge during the tsunami in Moorea.
Figure 6.4: This figure shows both the depth change as well as the tsunami alarm. The sudden change in the depth triggers the alarm on the right graph.
Chapter 7

Conclusion

The proposed and developed cyberinfrastructure may save energy usage in buildings. The cyberinfrastructure is designed to handle the real-time data acquisition and the real-time data processing in a generic way. The coral reef environmental monitoring system also needs similar functionalities, and it is shown here as another of these applications.

Most buildings currently do not employ intelligent controllers to save energy. In the last few years, several companies have started working on applying information technology to monitor and optimize the energy usages. Cisco is adding capabilities to monitor the usage of the electronic devices while a couple of startup companies are trying to integrate the sensors and building control system together. In terms of real world application, it is still at the inception stage.

The proposed system approaches the problem in a generic way, and the requirements of such system are analyzed first. Meeting the requirements, the proposed open architecture and its components are explained in detail. The instantiation of the open architecture incorporates the open source technology choices, and the integration among the pieces. The implementation is explained in detail, and the source codes are published in the public repository [DT10].

Although controlling the HVAC system and other building control system in the real world did not happen for various reasons other than the technical reasons, the simulation model and the results demonstrate how and what it can do to save the energy. As a real world application, the cyberinfrastructure for the
coral reef monitoring system is described, and it has been in the operation since summer of 2009. The sensors and the network equipments industries are growing. They will be more reliable, less expensive, and easier to communicate than now. The cyberinfrastructure for the real-time data acquisition and real-time processing system provides a generic way of composing the appropriate sensors and creating the solutions. The open architecture leaves enough open space to interface with the future standardization of the protocols and other interfaces.
Bibliography


