A Scalable Cloud-based Cyberinfrastructure Platform for Bridge Monitoring

Seongwoon Jeong[a*], Rui Hou[b], Jerome P. Lynch[b], Hoon Sohn[c], Kincho H. Law[a]

[a]Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA; [b]Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, USA; [c]Department of Civil and Environmental Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

*email: swjeong3@stanford.edu

A Scalable Cloud-based Cyberinfrastructure Platform for Bridge Monitoring

Cloud computing is a computing paradigm wherein computing resources, such as servers, storage and applications, can be provisioned and accessed in real time via advanced communication networks. In the era of Internet of Things (IoT) and big data, cloud computing has been widely developed in many industrial applications involving large volume of data. Long term deployment of a structural health monitoring (SHM) system would incur significant amount of data of different types. This paper presents a cloud-based cyberinfrastructure platform designed for bridge monitoring applications. A cloud-based platform comprises of virtual machines, distributed database and web servers. Distributed database built on a peer-to-peer architecture enables scalable and fault-tolerant data management on a cloud computing environment. Platform-neutral web services are designed in compliant with the REpresentational State Transfer (REST) design, and enable easy access to the cloud resources and SHM data via a standard web protocol. For data interoperability, a bridge information model for bridge monitoring applications is adopted. The scalable cloud-based platform is demonstrated for the monitoring of bridges along the I-275 corridor in the State of Michigan. The results show that the cloud-based cyberinfrastructure platform facilitates storage, retrieval and utilization of sensor data and bridge information for various bridge monitoring applications.

Keywords: cloud computing, bridge monitoring, bridge information modelling, interoperability, scalability, web service

1. Introduction

Advances in sensor and communication network technologies have led to increasing deployment of sensors for structural health monitoring (SHM) of civil infrastructures (Lynch and Loh 2006, Zhou and Yi 2013). Data collected from SHM systems can be useful in many different contexts, from short-term anomaly detection to long-term management of infrastructures (Lim et al. 2014, Cross et al. 2013, Dervilis et al. 2015). However, the lack of easy access, sharing and utilization of data hinders the potential

use of collected data. The volume and the variety of sensor data make data management an important task for long term structural health monitoring. A data management framework that can support long-term data archiving and effective data access is one indispensable component of data-intensive SHM systems (Law *et al*. 2014). This paper describes the design and implementation of a cloud-based cyberinfrastructure platform to cope with the big data issues in SHM systems.

Few attentions have been paid on data management in structural health monitoring research. Early SHM systems typically collect and store measurement data in files on local computers (Brownjohn *et al*. 1995, Farrar *et al*. 2000, Wong *et al*. 2000). File-based systems does not directly support queries, which often makes data access a tedious task. The use of relational database management systems (RDBMSs) as a centralized data storage for SHM systems has been reported (Li *et al*. 2006, Fraser *et al*. 2009, Koo *et al*. 2011, Smarsly *et al*. 2012). RDBMSs support structured query language (SQL) so that data residing in a database can be retrieved using query statements. However, research studies have suggested that current RDBMSs, which were architected decades ago when the characteristics of hardware and data processing requirements were very different, are not effective in meeting the data needs of today's applications which often involve text, time-series, image and video data (Stonebraker *et al*. 2007, Agrawal *et al*. 2011). Furthermore, a cloud computing platform provides a scalable computing infrastructure in the form of multiple commodity machines. To realize scalable data management, a database management system (DBMS) that can effectively "glue" the distributed commodity machines is needed. To overcome the performance and scalability issues of RDBMSs, NoSQL (Not-only-SQL) databases have been proposed. NoSQL databases have become desirable alternatives over the RDBMS for cloud-based data management (Grolinger *et al*. 2013). Structural health

monitoring is a data intensive application that shares the same burdens (Jeong *et al*. 2016a). This study extends the authors' previous work on NoSQL-based SHM framework (Jeong *et al*. 2016a) to enable scalable and interoperable SHM data management on a cloud computing environment. Specifically, a NoSQL database that enables dynamic scaling of the system and distributed data management is employed.

Cloud computing has been widely employed in many large scale industrial applications particularly in the era of Internet of Things (IoT) and big data. Advances in cloud computing provide highly scalable and accessible computing environment, as well as cost-effectiveness (Zhang *et al*. 2010, Zaslavsky *et al*. 2013, Gillis 2015, Deckler 2016). Many state-of-the-art data management platforms take advantage of cloud computing to allow communication and data sharing among physical systems, sensors, software applications and users. While there are several IoT platforms offered by major cloud vendors (Strukhoff 2017), most of commercial platforms tend not to design for specific applications. Generally, there is no one-size-fit-all platform: a data management platform needs to be tailored to meet domain-specific application requirements. The development of cloud-based frameworks has been reported in many science and engineering domains, including smart city (Lea and Blackstock 2014), smart home (Ye and Huang 2011), healthcare (Lin *et al*. 2015), robotics (Arumugam *et al*. 2010), manufacturing (Xu 2012) and construction (Das *et al*. 2015). A number of efforts have also been reported on utilizing cloud computing in infrastructure monitoring. Liao *et al*. (2014) described a cloud-based wireless sensor network framework to process sensor data collected from infrastructure and environmental monitoring. Zhang *et al*. (2016) described a cyberinfrastructure platform, called SenStore, that supports cloud-based data interrogation for infrastructure health management. Alampalli *et al*. (2016) utilized cloud platform to process data collected

for SHM of railroad bridges. While these efforts have focused on hosting SHM data and applications on a cloud platform, they do not address the performance and scalability issues for data management, data utilization, data interoperability and information sharing. Following our preliminary studies on the use of cloud computing for SHM systems (Jeong *et al*. 2016b, Jeong *et al*. 2016c), this paper describes in details a highly scalable and interoperable cloud-based data management system for bridge monitoring.

To take advantage of cloud computing, the software framework should be designed with consideration of the useful features provided by cloud services (e.g., dynamic provisioning, distributed computing and on-demand commodity hardware), as well as domain-specific application requirements (e.g., information model, application, interface, etc.). In contrast to traditional proprietary servers, the real value of cloud computing relies upon interoperability among systems and engineering services (Law *et al*. 2016). For service interoperability, engineering services on cloud platforms need to be exposed via standard interfaces. There are two main web service paradigms: namely, Service-Oriented Architecture (SOA) and Resource-Oriented Architecture (ROA). SOA is built upon standard web service protocols, such as Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Business Process Execution Language (BPEL), etc.. While SOA's reliability and message-level security benefit enterprise-level applications, the complexity of the protocols makes them less attractive for basic, *ad hoc* integration of services (Pautasso *et al*. 2008). ROA, on the other hand, is based on the *de facto* Representational State Transfer (REST) (Fielding 2000). REST has become a preferable approach because of its simple and lightweight architecture, easy accessibility and scalability (Zhao and Doshi 2009, Mulligan and Gracanin 2009, Belqasmi *et al.* 2012). In this study, RESTful web services (i.e., web services based on REST) are developed to provide standard interfaces.

Data interoperability requires information models that describe relevant bridge information in a platform-neutral language. One of the most notable efforts for establishing a bridge information modeling (BrIM) schema is the OpenBrIM standards (Chen and Shirolé 2013, Bartholomew *et al*. 2015) supported by the US Federal Highway Administration (FHWA). While the current OpenBrIM standards focus on the geometric representation of bridge structures, the authors of this paper have proposed a BrIM schema for bridge monitoring applications by extending the OpenBrIM standards (Jeong *et al*. 2017). In this work, the cloud-based cyberinfrastructure platform adopts the extended BrIM schema for data interoperability.

In this paper, a scalable cloud-based cyberinfrastructure platform for managing, sharing and utilizing sensor data and bridge information is presented. The cloud-based platform comprises of the virtual machines (VMs), distributed database, web servers, applications and user interfaces. For scalable, flexible, fault-tolerant and high-performing data management, an open source distributed NoSQL database is employed. NoSQL database ensures data consistency, supports partitioning and replication, and allows queries over decentralized data storages across multiple VMs. For data interoperability, database schema is designed based on the BrIM schema for bridge monitoring applications. For service interoperability, RESTful web services are implemented on web servers. To demonstrate the utilization of the cloud-based cyberinfrastructure platform, the prototype platform is implemented and validated for the monitoring of bridges along the I-275 corridor in the State of Michigan.

2. Cloud computing environment

Cloud computing, as defined by the National Institute of Standards and Technology (NIST), is a “model for enabling convenient, on-demand network access to a shared

pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell and Grance 2011). Cloud computing can reduce the cost and lessen the burdens on the deployment, operation, maintenance and management of data centers. Using cloud computing, a SHM system can be easily and quickly scaled up and down on demand with optimal usages of computing and storage resources.

Cloud computing services are typically categorized into three service models (Mell and Grance 2011): (1) Software as a Service (SaaS) that provides applications and web services to end users, (2) Platform as a Service (PaaS) that provides runtime and database supports, and (3) Infrastructure as a Service (IaaS) that provides the basic computing utilities including network, processor and storage. In this work, as depicted in Figure 1, the cloud-based cyberinfrastructure platform serves as PaaS and SaaS that employ computing infrastructures and platforms (i.e., IaaS and PaaS) to provide data management and application services.

Figure 1. A model of cloud computing for SHM

IaaS utilities are typically offered in the form of VMs. Here, a VM is a virtualized computing system that abstracts the underlying physical computer architecture and offers the same functionalities of a physical computer (Smith and Nair 2005). A VM can be configured and created in minutes and be managed through cloud interfaces offered by a cloud vendor. For example, Figure 2(a) shows the web interface of the Microsoft Azure cloud platform[1], namely the Azure portal, that shows of a VM's

[1] https://azure.microsoft.com/

information such as name, status, operating system (OS) and size. Once created, a VM can be accessed via standard network protocols, such as Secure Shell (SSH) and Secure Copy Protocol (SCP). Figure 2(b) shows the shell interface of a VM on Azure cloud platform accessed via the SSH protocol. Similar to using a remote physical server, a VM can be used to deploy computing platforms and applications. The proposed cyberinfrastructure platform utilized VMs to deploy computing components, such as distributed database, web servers and applications.

The IaaS utilities can be scaled both vertically (i.e., increasing capability of a VM) and horizontally (i.e., adding new VMs) on demand. While vertical scalability is limited to the maximum capability of a single VM, the horizontal scalability is nearly unlimited since cloud vendors allow adding as many VMs as needed. Figure 2(c), for example, shows that multiple VMs are deployed as needed on the Azure cloud platform. To take advantage of the scalability of an IaaS utility and thus enable scalable SHM data management, the cloud-based cyberinfrastructure platform should be designed to run on a distributed computing environment such that new VMs can be dynamically added on demand. For example, as will be discussed later, the proposed cyberinfrastructure platform adopts a NoSQL database which can be effectively executed on multiple VMs to offer a large scale distributed data store.

Figure 2. Virtual machine created on Microsoft Azure cloud platform: (a) Web-based cloud interface, (b) Shell interface, (c) List of virtual machines deployed on Azure cloud platform

3. A cloud-based cyberinfrastructure platform for bridge monitoring

Today's bridge monitoring and management tasks involve various types of data collected from different sources including, for examples, a SHM system, engineering

analysis and design, traffic data, maintenance and inspection, etc.. With effective integration and utilization of data, actionable insights for bridge management and maintenance can be derived (Cross et al. 2013, Zhang *et al*. 2016, Hou et al. 2017). Careful design of a SHM data management platform is essential to facilitate the utilization of SHM data.

In this study, a bridge monitoring scenario that involves diverse types of data, including information about sensors, sensor measurement data, video image data, bridge geometries and engineering models, is considered. Sensor networks instrumented on a bridge collect structural data (e.g., acceleration and strain) and environmental data (e.g., temperature, wind speed and wind direction). Video cameras mounted near the bridge collect video images of traffic passing through the bridge. For the utilization of collected sensor data, sensor information (e.g., sensor ID, sensor type, sensor location, etc.) is also managed. In addition, bridge geometric models and engineering models are involved to perform structural analysis and to understand the behavior of the bridge structure.

A cloud-based cyberinfrastructure platform is designed to manage these diverse types of data and to enable data retrieval by a variety of client systems (e.g., data analysis modules, engineering analysis tools and end-user interfaces). The platform is designed with an emphasis on (1) a scalable database design to handle voluminous and heterogeneous SHM data, (2) information model to capture SHM-related data and to facilitate data interoperability, and (3) interoperable web services to enable easy access to SHM data and to facilitate SHM application developments. Figure 3 shows the conceptual framework of the cloud-based cyberinfrastructure platform which includes the following major components:

- *A distributed NoSQL database* is deployed to manage sensor data and bridge information over multiple cloud VMs (i.e., VMs deployed on cloud platforms). Specifically, a highly scalable NoSQL database, namely Apache Cassandra database[2], is employed to guarantee partitioning tolerance and database availability.
- *A BrIM schema* (Jeong *et al*. 2017) is employed to support data interoperability among applications. The BrIM schema includes data entities for the description of bridge geometric models, engineering models and sensor information.
- *Web servers* are employed to host RESTful web services that expose resources to clients via Hypertext Transfer Protocol (HTTP). Web services for storing and retrieving bridge monitoring data are implemented.

Figure 3. Conceptual framework of cloud-based cyberinfrastructure platform

3.1 A scalable distributed database

There are a variety of readily available NoSQL database systems, each with their own features. Selection of an appropriate NoSQL database for a specific purpose is critical for efficient data management. The cloud-based cyberinfrastructure platform aims for high scalability to handle voluminous SHM data on a distributed cloud computing environment. Given the requirement, Apache Cassandra database, which is one of the most widely used NoSQL databases, is selected (Jeong *et al*. 2016a). The Cassandra database is built upon the peer-to-peer (P2P) architecture which is a preferable approach for a highly available and scalable distributed database (Stonebraker *et al*. 2007). Figure

[2] http://cassandra.apache.org/

4, as an example, shows the status table of a Cassandra cluster consisting of five nodes (i.e., database instances). The status table shows the current states of the nodes and their topology information such as the data center ID (e.g., DC1) and the rack IDs (e.g., RAC1, RAC2, etc.). The Cassandra database is particularly suitable for large scale data management because the number of nodes can be easily modified without causing operational downtime and the database performance is linearly scaled as new nodes are added to an existing Cassandra cluster (Hewitt 2010). Another advantage of using the Cassandra database for SHM applications is its query performance for time-series data (Le *et al*. 2014). The variety of data types (e.g., number, array, dictionary, binary data, etc.) and flexible data schema supported by Cassandra database can also be an advantage for SHM data management which involves sensor data and video image data.

Figure 4. A Cassandra cluster instance

To maintain the consistency of the database and to process requests in a decentralized manner, nodes in a Cassandra cluster communicate among one another according to a "ring" topology as shown in Figure 5. Data is replicated and distributed over multiple nodes to ensure high availability and fault-tolerance, as well as to maintain efficient reading and writing performances. Figure 5, for instance, illustrates how SHM data is stored in a Cassandra cluster. In this example, the incoming data *R* has two rows *r1* and *r2* which could be sensor measurement data collected by different sensors. The replication factor (i.e., the number of replicas in a cluster) is two. Any node (say, node *N5* in the example) can accept the write request. The incoming sensor data is partitioned into two pieces and then copied twice over the nodes. Since the sensor data is replicated over the cluster, writing and reading the data can still be

performed even when a node is down, as long as other nodes remain available for processing the requests.

Figure 5. Ring topology of Cassandra database

3.2 A data schema definition of bridge information model

3.2.1. Bridge information model

A consistent and unified information model is needed to manage the heterogeneous information involved in bridge monitoring and to enable data interoperability among the applications. The proposed platform adopts a BrIM framework for bridge monitoring application (Jeong *et al*. 2017). This BrIM framework extends the OpenBrIM standards (Chen and Shirolé 2013, Bartholomew *et al*. 2015) by introducing additional data entity definitions for representing sensor information and finite element (FE) models. Specifically, the BrIM framework drew upon the data entities of SensorML (an open standard for sensor description (Open Geospatial Consortium 2014)) and CSI Bridge (a structural engineering software tool[3]).

The base schema (i.e., the data schema of OpenBrIM standards) represents a bridge using hierarchical objects and their parameters (Bartholomew *et al*. 2015). Each object describes a physical or conceptual entity (e.g., beam, column, group and project) while each parameter describes an attribute (e.g., width, height and length) of an object (or refers to another object). To encode bridge information in an object-oriented fashion, OpenBrIM standards use the ParamML, an Extensible Markup Language (XML)-based syntax (ParamML 2017). Figure 6(a), for example, shows the data

[3] https://www.csiamerica.com/products/csibridge

schema of a “Shape” object defined in the base schema. The data schema is displayed using an XML schema definition (XSD) diagram, which can be interpreted as follows:

- Each box refers to an XML component, such as element, attribute and complex types.
- The abbreviated letters in the inner shadowed boxes refer to the XML component types. For example, “CT”, “A” and “E” refer to “xs:complexType”, “xs:attribute” and “xs:element”, respectively (W3C 2014).
- The attribute “T” represents the type of an object.
- The attribute “N” represents the name of a parameter.
- The xs:element “O” and “P” represent BrIM object and parameter, respectively.
- The string on the left-side of the “:” represents the name of an XML component, while the string on the right-side of the colon represents the data type or an extension base. For example, “Shape: Object” means that the XML component has name “Shape” and its extension base is “Object”.
- The numbers on the left side of an XML component represents the possible number of the components. For example “0..*” means equal or greater than zero.

As an example, the XSD diagram in Figure 6(a) defines that (1) a “Shape” object is extended from the “Object” object; (2) a “Shape” object can have any number of child objects each of which has either “Point”, “Shape” or “Circle” type; and (3) a “Shape” object can have any number of parameters having name “Material”.

The base schema is extended with additional data entities for the representation of FE models and sensors. Figure 6(b) shows the extended data schema of “FELine” object as a representative example of FE model objects. The original definition of “FELine” includes parameters “Node1”, “Node2” and “Section” to describe two ends of

the line element and its section information. In addition, new data entities FELineMesh, FELineRelease and FELineSection are defined to describe mesh information, member-end release information and standard section shapes, respectively. Figure 6(c), as another example, shows the newly defined data schema "StrainGauge" as a representative example of sensor object. The data schema of "StrainGauge" object includes data entities for describing the "Input" (data that will be processed by the sensor), "Output" (processed data), "Parameters" (values needed for processing data) and "DataLink" (link to the sensor data repository).

Figure 6. Data schema definition in BrIM for bridge monitoring applications: (a) Shape, (b) FELine, (c) StrainGauge entity types

3.2.2 Database schema definition

An extensible database schema is needed to effectively manage the complex SHM data and bridge information. RDBMSs are not designed to manage hierarchical objects because of their inflexible data structure, Cassandra database, on the other hand, offers more flexible data structure that can elegantly handle complex data (Hewitt 2010). This section describes Cassandra database schema definitions for managing complex data involved in SHM systems.

The Cassandra database is built upon a column-oriented data model consisting of "keyspace," "column family," "row" and "column," which are analogous to "database," "table," "tuple" and "attribute" of relational database, respectively. To manage bridge information, the database schema follows closely the BrIM schema for bridge monitoring applications. Figure 7 shows data mapping between the BrIM schema of the "FELine" object and the corresponding column family schema "FELine". The database schema contains the data entities of "FELine", as well as "child" and "parent"

entities to record the hierarchical relation between the objects. As such, bridge information stored in the column-oriented database can be mapped to hierarchical BrIM objects.

Figure 7. Data mapping between BrIM schema "FELine" and corresponding Cassandra column family.

Figure 8 shows examples of the rows of the column-oriented database for storing BrIM objects where a single object is stored in each row. Each row has a mandatory partition key (e.g., "shp001" of the first row in Figure 8(a)). A row has columns for storing attributes and parameters, as well as the list of child and parent objects. Since the Cassandra database supports collection types, any number of child objects can be recorded in the child column. In Figure 8(a), for example, the child column of the shape object contains the ID and types of child objects (i.e., ["pt001": "Point", "pt002": "Point", …]). One issue in managing hierarchical object data is that each object may have different sets of attributes. This data irregularity can be efficiently handled by the Cassandra database with its flexible data structure. Specifically, the Cassandra database allows rows in the same column family to contain different sets of columns. For instance, Figure 8(b) shows that the two rows in the column family "FELine" have different column sets: the first row has the "FELineRelease" column, while the second one does not. In fact, BrIM objects with the same type often have different sets of attributes and child objects. As such, the flexible data structure of Cassandra database is suitable to handle the heterogeneous BrIM object entities without enforcing every row to have the same set of columns.

Figure 8. Database schema for BrIM objects: (a) Rows storing Shape object and its child Point objects, (b) Rows storing heterogeneous FELine objects

In addition to BrIM objects, SHM systems collect a large volume of time-series sensor data and traffic video images. Since SHM applications often utilize continuous time-series data collected within a certain period, efficient range query performance for time-series data is needed. Cassandra database has better range query performance comparing to RDBMSs because of the clustering and dynamic column features (Le *et al*. 2014). Cassandra data schema for SHM time-series sensor data is defined as follows. As shown in Figure 9(a), data collected from a sensor is stored in a single row in a sorted order by assigning the timestamp (e.g., "2014-08-02T00:00:08") of data as a clustering key. Since sensor data collected from SHM systems usually has very high sampling rate with same interval period between data points, it is redundant to record timestamp for every data point. Instead, the proposed data schema encodes sensor data as a numeric array type that stores data collected during a specified time period (e.g., 1 second) in a sorted order. The timestamp records can be regenerated, if needed, based on the sampling rate. When new data is collected, the incoming data is stored to the same row by dynamically adding new columns at the end of the row. This data schema improves the range query performance by enforcing the consecutive sensor data to be stored in a contiguous physical disk location in the same node (Le *et al*. 2014). To prevent a single row from becoming too lengthy, part of the timestamp (e.g., year) is added to the row key so that the data from a sensor can be partitioned to several rows based on a specific time period (e.g., year) of data acquisition.

Similarly, Figure 9(b) shows that sequential image files collected from a traffic video camera are stored in a row by assigning timestamp (e.g., "2016-08-23T10:02:08") as a clustering key. In addition, part of the timestamp (e.g., year and month) is added to the row key to partition image data to several rows based on the year and month of its acquisition. Each image file is encoded in a binary large object (BLOB) data (e.g.,

"/9j/4AAQSkZj … H//Z") and stored in a single column. The BLOB data can be converted back to the original image file using imaging libraries, such as Python Imaging Library[4].

Figure 9. Database schema for time-series data: (a) Row storing sensor measurement data, (b) Row storing traffic video image data

Data stored in the Cassandra database can be retrieved using a SQL-like query language, namely Cassandra Query Language (CQL). For example, Figure 10 shows a SELECT query statement that specifies sensor ID ("TRB_u07_ch0"), year ("2014") and time range (from "2014-08-02T00:00:00" to "2014-08-02T01:00:00") to retrieve acceleration data. The query result is returned in a tabular structure. Since Cassandra database offers application programming interfaces (APIs) for various programming languages, platforms and applications can remotely access the Cassandra database and query the data using CQL.

Figure 10. Select query for sensor data retrieval and query result in a tabular format

3.3 Web servers

The resources residing in the cloud-based cyberinfrastructure platform needs to be exposed via standardized interfaces to facilitate sharing, integration and potential utilization of the data from various applications and devices. To this end, the design and implementation of web services for the cyberinfrastructure platform are presented.

[4] http://www.pythonware.com/products/pil/

3.3.1 Web services

Web services, as defined by W3C, is a "software system designed to support interoperable machine-to-machine interaction over a network" (Haas and Brown 2004). Web services enable sharing of data and integration of applications over the network. Since the cyberinfrastructure platform needs to support utilization of data from various devices (e.g., cloud, local computer, micro computer and mobile devices) and platforms (e.g., different operating systems), it is important to employ a widely-adopted web service protocol. Furthermore, the cyberinfrastructure platform needs to support conditional queries involved in SHM applications (e.g., range query for time-series data). To meet these requirements, the cyberinfrastructure platform employs RESTful web services which have fast performance and high scalability (Mulligan and Gracanin 2009). It should be noted that there are other lightweight communication protocols, such as MQTT (MQ Telemetry Transport) protocol[5]. While such protocols have advantages on real-time communication via publish/subscribe messaging, our design stresses on the query capabilities that efficiently handle typical data utilization patterns commonly seen in SHM applications (e.g., range query for time-series data). RESTful web services are described using five constraints (Guinard *et al*. 2010):

- *Resource identification.* Resources are identified by uniform resource identifiers (URI).
- *Uniform interface*. Resources can be accessed via the HTTP.

[5] http://mqtt.org/

- *Self-descriptive messages.* Resources are represented using standardized formats, such as Hypertext Markup Language (HTML), XML and JavaScript Object Notation (JSON).
- *Hypermedia as the engine of application state.* Resources contain links by which clients can interact with web services.
- *Stateless interactions*. Requests contain all the required information for web services to process the requests.

Table 1 summarizes the web services currently implemented for the cloud-based cyberinfrastructure platform to manage sensor data and bridge information. Here, the HTTP method GET is used to retrieve data from the specified URIs, while the HTTP method POST is used to submit data to the specified URIs (Fielding *et al*. 1999). The following briefly describes two examples to illustrate the basic process for implementation of RESTful web services.

Table 1. RESTful web services currently implemented on the cyberinfrastructure platform

Figure 11, for example, shows the "Sensor data store" service which accepts POST requests with URI "/sensordata" and processes a request in four steps:

(1) The client sends a POST request to the web server with URI (/sensordata), protocol (HTTP/1.1), host (<ws_address>.cloudapp.azure.com), content type (application/json) and JSON-encoded content including sensor_id, event_time and data.
(2) The web server sends an INSERT query to the database by parsing the request.
(3) The database informs the web server that the INSERT query has been processed.

(4) The web server returns a response to the client with a status code 200, which means "OK" in the HTTP (Fielding *et al*. 1999). (Other status codes are returned if an error is encountered in the retrieval process.)

As another example, Figure 12 describes the "Sensor data retrieval" service which accepts GET requests with URI "/sensordata/{sensorID}". Again, the service processes a request in four basic steps:

(1) The client sends a GET request to the web server with URI "/sensordata/TRB_u131_ch0", query parameters (e.g., "event_time_begin" and "event_time_end"), protocol (HTTP/1.1) and host (<ws_address>.cloudapp.azure.com).
(2) The web server sends a SELECT query corresponding to the query parameters to the database.
(3) The database returns query result to the web server.
(4) The web server returns a response enclosing JSON-encoded query result and the status code 200 to the client.

Figure 11. Web service example: sensor data store service

Figure 12. Web service example: sensor data retrieval service

3.3.2 Web service composition

This section describes how web services offered by the cyberinfrastructure platform can be used for developing and integrating SHM applications. Web service composition refers to the process of combining different web services to provide a new service that carries out composite functions (Sheng *et al*. 2014). Standardized web services can be

efficiently composed. Different methods have been suggested for composition of RESTful web services (Pautasso 2008, Rosenberg *et al.* 2008, Zhao and Doshi 2009, Pautasso 2009). For visual demonstration purpose, this study adopts Pautasso (2009)'s approach that uses JOpera[6], a visual composition language. JOpera describes control flow and data flow between programs using a graphical model (Pautasso and Alonso 2005). Each node of the graph represents either a program, an input of a program, or an output of a program, while each edge of the graph represents a control flow or data flow between nodes. Each program performs a function, such as web service invocation, script execution and HTML document creation. The following briefly describes two examples to illustrate the composition of RESTful web services implemented on the cyberinfrastructure platform.

Figure 13(a) shows the JOpera data flow of the first demonstrative application named "DataRetrievalByLocation". This application composes two web services "Sensor list retrieval" and "Sensor data retrieval" in order to retrieve sensor data measured at a specified location by a specified type of sensor. Here, the hollow arrows describe the input and the output flow of each program, while the solid arrows describe the data flow and control flow between programs.

As described in Figure 13(a), the application consists of three programs, namely "SensorListRetrieval", "SearchByLocation" and "SensorDataRetrieval", and processes a request in five steps:

(1) The application accepts input arguments including target time period ("start_time", "end_time"), sensor type ("sensor_type") and local coordinate ("loc_X" and "loc_Y") from a client.

[6] http://www.jopera.org/

(2) The “start_time”, “end_time” and “sensor_type” are passed to the program “SensorListRetrieval” as input parameters. The program invokes the “Sensor list retrieval” service with the input parameters, and then returns an output parameter “SYS.page” enclosing the retrieved sensor list.

(3) The “loc_X” and “loc_Y” and the “SYS.page” from the previous step are passed to the program “SearchByLocation” as input parameters. The program searches a sensor corresponding to the “loc_X” and “loc_Y” from the sensor list, and then returns “sensor_id” of the searched sensor.

(4) The “start_time”, “end_time” and the “sensor_id” from the previous step are passed to the program “SensorDataRetrieval” as input parameters. The program invokes the “Sensor data retrieval” service with the input parameters, and then returns an output parameter “SYS.page” enclosing the retrieved sensor data.

(5) Finally, the “SYS.page” from the previous step is passed to the application’s output “sensor_data” which is returned to the client.

Figure 13(b) shows the retrieved sensor data when the application is executed with the input arguments “2014-08-01T00:00:00”, “2014-08-10T00:00:00”, “Accelerometer”, “102” and “-50”, which correspond to “start_time”, “end_time”, “sensor_type”, “loc_X” and “loc_Y”, respectively.

Figure 13. A composite application DataRetrievalByLocation: (a) Data flow, (b) Execution example

Figure 14(a) shows the data flow of the second application named “SensorInfoOnMap” that composes an internal web service “Sensor information retrieval” and an external service “Google Map API”. Given a sensor’s ID, the application shows sensor information at the location of the sensor on the map. As shown

in Figure 14(a), the application consists of three programs, including "SensorInfoRetrieval", "SensorInfoParser" and "MapHandler", and processes a request in five steps:

(1) The application accepts an input argument "sensor_id" from a client.
(2) The "sensor_id" is passed to the "SensorInfoRetrieval" as an input parameter. The program invokes the "Sensor information retrieval" service with the input parameter, and then returns an output parameter "SYS.page" enclosing the sensor information.
(3) The output parameter of the previous step (i.e., "SYS.page") is passed to the "SensorInfoParser" as an input parameter. The program parses the sensor information and returns extracted data entities including sensor's ID, coordinate, type, position and description.
(4) The data entities from the previous step are passed to the "MapHandler". The program returns an HTML document that displays extracted data entities on the Google Map by using the Google Map JavaScript API[7].
(5) Finally, the HTML document from the previous step is passed to the application's output that can be visualized by a web browser.

Figure 14(b) shows the result of the "SensorInfoOnMap" application with an input argument "TRB_u07_ch0", where the sensor information and the location marker are displayed on the Google map.

Figure 14. A composite application SensorInfoOnMap: (a) Data flow, (b) Execution example

[7] https://developers.google.com/maps/documentation/javascript/

3.3.3 Web server

A web server is a computer system that processes clients' requests and returns responses to the corresponding clients. The cloud-based cyberinfrastructure platform deploys web servers to host the RESTful web services. Given the high volume and high velocity of SHM data, a web server that can process intensive input/output (I/O) requests is needed. The proposed cyberinfrastructure platform employs Node.js[8], a server-side JavaScript runtime environment. Based on the non-blocking I/O feature, Node.js is suitable for intensive I/O processing (Lei *et al.* 2014, Chaniotis *et al*. 2015).

While a single web server can serve multiple web services for all the clients, the web server would become a bottleneck of the cyberinfrastructure platform due to the limited capabilities of a single machine. To improve the scalability of the web server, the cyberinfrastructure platform deploys duplicated web servers that process requests in a distributed manner. Figure 15 depicts a demonstrative situation in which two separate sensor networks are transmitting sensor data to a cyberinfrastructure platform simultaneously and, at the same time, a local computer attempts to retrieve sensor data from the cyberinfrastructure. Three duplicated web servers offering the same web services are implemented on three cloud VMs. The web servers balance loads as follows:

(1) The local computer sends a GET request with URI "/serverIP" to any of the web servers (e.g., "Web server 2" in this example) in order to find the least busy web server.

[8] https://nodejs.org/en/

(2) The “Web server 2” communicates with the other web servers to check if the other servers are up and, if so, to count the number of current connections of each server.

(3) The “Web server 2” sends a response enclosing the IP address of the web server having least connection (e.g., “Web server 3” in this example) to the local computer.

(4) Given the IP address, the local computer starts to invoke the “Sensor data retrieval” service via the “Web server 3”.

Figure 15. Duplicated web servers

4. Implementation

This section describes the implementation of the prototype cloud-based cyberinfrastructure platform for real bridge monitoring systems. For demonstration purposes, bridge monitoring applications (e.g., automated data store application, automated data retrieval and analysis application and user interfaces) built upon the cloud-based cyberinfrastructure platform are presented.

The research team at the University of Michigan has implemented bridge monitoring systems on the Telegraph Road Bridge (TRB) and the Newburg Road Bridge (NRB). Both are steel girder bridges located along the I-275 corridor between Romulus, Michigan and Toledo, Ohio (see Figure 16). The TRB has been monitored with a wireless sensor network that consists of sixty sensors including accelerometers, strain gauges and thermistors since 2011 (O’Connor *et al*. 2012, O’Connor *et al*. 2017). The NRB has been monitored by a wireless sensor network that consists of twelve sensors including accelerometers, strain gauges and thermistors since late 2016.

Specifically, these bridge monitoring systems are instrumented with Narada wireless sensor nodes (Swartz *et al*. 2005) for the wireless data transmission from the sensors to the onsite computers, as well as for the time synchronization. Data transmitted from the Narada wireless sensor nodes is stored temporarily to the local file system of the onsite computers. The sensor networks of the bridges acquire data for a one-minute duration every two-hours. The sampling rate of the accelerometers is 200 Hz, while the sampling rate of strain gauges and thermistors is 100 Hz. Sensor information, as well as the geometric and engineering models of the bridges (which are created based on the 2-dimensional drawings of the bridges), are also recorded and stored. In addition, the monitoring systems collect traffic video images from the traffic monitoring system of the Michigan Department of Transportation (MDOT)[9]. These diverse types of data are collected and managed using the cloud-based cyberinfrastructure platform.

Figure 16. Bridges on the I-275 corridor installed with bridge monitoring systems: (a) Telegraph Road Bridge (TRB), (b) Newburg Road Bridge (NRB)

A prototype cloud-based cyberinfrastructure platform is built upon eight Ubuntu Linux (16.04 LTS) VMs on the Microsoft Azure cloud platform. As summarized in Table 2, five VMs (No.1 – No.5) are used to build a multi-node Cassandra database (Version 3.4), while three VMs (No.6 – No.8) are used to construct three duplicated web servers using Node.js (Version 4.2.6). The web servers are connected to the database nodes using the "DataStax Node.js Driver for Apache Cassandra" (DataStax 2017).

[9] http://mdotnetpublic.state.mi.us/drive/

Table 2. Specification and role of virtual machines provisioned on the Microsoft Azure cloud platform

4.1 Automated data store and retrieval

Two automated data store applications are developed using the cyberinfrastructure's web services to archive sensor data and video image data, respectively. Figure 17 shows the workflow of the first application that runs on onsite computers and transmits sensor data from an onsite computer to the cloud-based cyberinfrastructure platform. When new data is transmitted from the sensor network to the onsite computer, the application records the list of the new data files and labels them as "un-transmitted". For an "un-transmitted" file, the application parses the raw data file encoded in DAT file format (see Figure 18(a)) into a JSON format (see Figure 18(b)) that the "Sensor data store" service of the cyberinfrastructure can read. The application then invokes the "Sensor data store" service with the parsed sensor data. If the service returns a status code 200, the application changes the label of the data file to "transmitted". Otherwise, the application retries invoking the "Sensor data store" service up to N times (i.e., a predefined maximum number of retries) to ensure the transmission is done properly. This parsing and storing process is repeated until there are no "un-transmitted" data files in the list. Since the application keeps track of data transmission status of data files, data losses due to an unstable network connection can be minimized.

Figure 17. Workflow: application for data store automation

Figure 18. Sensor data file: (a) Raw data file, (b) Parsed sensor data

Figure 19 shows the workflow of the second application for collecting traffic video images from an external data source (i.e., MDOT's traffic monitoring system) and

the manner by which images are archived along with the camera ID and timestamp. The application first accesses the data source to locate the dynamic Uniform Resource Locators (URLs) of video image files. Once the URLs are found, the application fetches the video image files and converts them to the BLOB format. The application then transmits the BLOB data to the cyberinfrastructure by invoking the "Traffic image store" service. The application service repeats this process every two seconds corresponding to the time interval between new images in the MDOT's traffic monitoring system.

Figure 19. Workflow: traffic video image collecting application

Data stored in the cloud-based cyberinfrastructure platform can be easily retrieved using data retrieval services. For example, Figure 20(a) shows a request for the "Sensor data retrieval" service with query conditions including sensor ID ("TRB_u131_ch0"), event_time_begin ("2016-09-01T12:02:00.000z") and event_time_end ("2016-09-12T12:02:10.000z"). Similarly, Figure 20(b) shows a request for the "Traffic image retrieval" service with query conditions including camera_ID ("Telegraph2"), event_time_begin ("2016-08-18T18:01:00.000z") and event_time_end ("2016-08-18T18:01:20.000z").

Figure 20. Data retrieval using web services: (a) Sensor data retrieved by invoking the sensor data retrieval service, (b) Traffic images retrieved by invoking the traffic image retrieval service

4.2 Data integration and utilization

The advantages of the cloud-based cyberinfrastructure platform is its ability to support easy access, integration and utilization of SHM data. This section presents a

demonstrative example application developed to extract patterns from heterogeneous data (e.g., structural sensing data and environmental data) to find the relationships between the modal frequencies derived from sensor data with temperature measurements. The cyberinfrastructure platform enables machine-to-machine communication so that the workflow can be fully automated with applications written using programming scripts, for example, Python. As shown in the conceptual workflow described in Figure 21, the application accesses the cyberinfrastructure platform via web services, retrieve acceleration data and temperature data, and performs analyses. The application service comprises of the following steps:

(1) the application reads the input arguments “StartTime” and “EndTime” specifying the target time period which typically includes multiple data acquisition events.

(2) the application service retrieves the accelerometer list for a data acquisition event by submitting request as a web service to the “Sensor list retrieval service” as shown in Figure 22(a).

(3) the application service retrieves the acceleration data collected from the data acquisition event for each accelerometer in the list by submitting a request (as another web service) to the “Sensor data retrieval” service, as shown in Figure 22(b).

(4) the application executes the subspace identification module service (Overschee 2012) to compute the modal frequency from the retrieved acceleration data.

(5) the application service retrieves the thermistor list for the data acquisition event by submitting a request (as a web service) to the “Sensor list retrieval service” as shown in Figure 22(c).

(6) the application service retrieves temperature data collected from the data acquisition event by submitting a request to the "Sensor data retrieval" service, as shown in Figure 22(d).

(7) Executing Gaussian Process Regression (GPR) service to find the general pattern on the variation between the fundamental modal frequency and temperature measurements.

The application service repeats step 2 to step 6 by moving on to the next data acquisition event until reaching the EndTime of the targeted period. Once modal frequencies and temperature measurements that share the same timestamps are collected, the application service proceeds to step 7 to perform regression analysis.

Figure 23(a) shows the history of the first modal frequency over the duration from June 2013 to August 2015. The modal frequencies vary with a seasonal trend and tend to increase during the winter and decrease during the summer. As shown in Figure 23(b), the GPR analysis result shows a nearly bilinear relation between the modal frequency and temperature. This example application shows that the cyberinfrastructure platform can be used to automate repetitive tasks involving multiple SHM data sets.

Figure 21. A workflow for relating structural behaviour with temperature data

Figure 22. HTTP requests and corresponding CQL queries for sensor list and data retrieval: (a) HTTP request for accelerometer list retrieval, (b) HTTP request for acceleration data retrieval (c) HTTP request for thermistor list retrieval, (d) HTTP request for temperature data retrieval

Figure 23. Patterns of modal frequency of the Telegraph Road Bridge: (a) History of first modal frequency (from August 2013 to August 2015) (b) Gaussian process regression showing the confidence interval of modal frequency according to temperature changes

4.3 Web and mobile interface

To facilitate ubiquitous access to the bridge monitoring information, preliminary web and mobile user interfaces are developed based on the cloud-based cyberinfrastructure platform. A web interface is an interactive program that reads user inputs via a web browser (e.g., Google Chrome), invokes the requested web services and returns a web page displayed on a web browser. The preliminary web interface supports the retrieval of sensor list, sensor data, traffic video images and bridge models. The sensor information retrieval interface (Figure 24(a)) allows users to retrieve the sensor list with query parameters (e.g., "Sensor ID", "Sensor type", "Install before" or "Removed after") by invoking the "Sensor list retrieval" service. The sensor data retrieval interface (Figure 24(b)) accepts query parameters (e.g., "Sensor ID, "Begin timestamp" and "End timestamp") and returns corresponding sensor data by invoking the "Sensor data retrieval" service. Similarly, the traffic video image retrieval interface (Figure 24(c)) accepts query parameters (e.g., "Camera ID", "Begin timestamp" and "End timestamp") and returns corresponding traffic video images by invoking the "Traffic image retrieval" service. The bridge model retrieval interface (Figure 24(d)) allows users to download bridge models by invoking either the "Geometric model retrieval" or "Engineering model retrieval" service. Through this interface, users can specify the "Bridge name" and model type (e.g., "GeometricModel", "FEModel (xml)" and "FEModel (xlsx)"). The bridge models downloaded can be regenerated by proper software tools, such as the OpenBrIM viewer[10] for geometric models (Figure 25(a)) and CSI Bridge for engineering models (Figure 25(b)).

[10] https://openbrim.org/

Figure 24. Prototype web-based user interface: (a) Sensor information retrieval, (b) Sensor data retrieval, (c) Traffic image retrieval, (d) Bridge model retrieval

Figure 25. Telegraph road bridge model downloaded from the web-based user interface: (a) Geometric model visualized by OpenBrIM viewer, (b) Engineering model visualized by CSI Bridge

A mobile user interface is also developed based on the cyberinfrastructure platform. The mobile user interface reads user inputs via the mobile devices, invokes the web services, and displays the retrieved information on the mobile devices. The preliminary mobile user interface is built upon iOS operating system and supports the retrieval of sensor list, sensor information and sensor data. For example, Figure 26(a) shows the sensor list view that reads user's search keyword (e.g., "Accelerometer") and retrieves sensor list by invoking the "Sensor list retrieval" service. Figure 26(b) shows the sensor map view that displays the retrieved sensor list on a map view. Figure 26(c) shows the sensor detail view that displays brief information about a sensor along with its sensor data by invoking the "Sensor data retrieval" service. Figure 26(d) shows the sensor information view that displays a sensor's detailed information by invoking the "Sensor information retrieval" service.

Figure 26. Prototype mobile interface: (a) Sensor list view, (b) Sensor map view, (c) Sensor detail view, (d) Sensor information view

5. Summary and conclusion

Cloud computing has become a popular computing paradigm that offers many benefits such as cost-effectiveness, scalability and accessibility for developing large scale software application. In this study, cloud computing technology is explored to develop a

cloud-based cyberinfrastructure platform for SHM systems. Specifically, the cloud-based platform is demonstrated to have the ability to handle a wide variety of data including sensor data, sensor information, bridge geometry, engineering models and traffic video images.

For the scalable data management on a distributed cloud computing environment, Apache Cassandra database is employed. The Cassandra database enables high availability, partitioning-tolerance and scalability, which are the most important features for the management of large scale SHM data sets. For data interoperability, a database schema is defined based on a BrIM framework which is intended for SHM applications. Specifically, the BrIM framework extends the OpenBrIM standards by adding new data entities for the representation of engineering models and sensor information. For service interoperability, RESTful web services are implemented to support store and retrieval of sensor data and bridge information. For the scalability of web services, the cloud-based platform employs duplicated web servers that handle requests in a distributed manner.

The prototype cloud-based cyberinfrastructure platform has been implemented for the bridge monitoring systems along the I-275 corridor between Romulus, Michigan and Toledo, Ohio. Eight VMs are created on the Microsoft Azure cloud platform, where five VMs serve as the distributed database and three VMs serve as the web servers. Several demonstrative applications built upon the cloud-based platform are developed. For example, automated data store applications are developed to transmit sensor data and traffic video image to the cyberinfrastructure platform. Furthermore, a demonstrative application that calculates the modal frequency history is developed to show that the cloud-based platform is useful for complex queries, iterative jobs and batch operations. Finally, preliminary web and mobile user interfaces are developed to

facilitate ubiquitous access to the bridge monitoring data residing in the cloud-based cyberinfrastructure platform.

In the current design and development of cloud-based cyberinfrastructure platform for SHM systems, this paper mainly focuses on the cloud-based data management with emphasis on scalability and interoperability. One of the important aspects that could be explored in the future studies is meaningful analyses of large scale heterogeneous data set. The authors' current work attempts to develop data analysis applications that extract meaningful information from the large data sets collected from sensors to help bridge managers evaluate bridge conditions. In addition, cloud security is another important topic that was not discussed in this paper. For secure data management, the proper use of cryptography and authentication schemes is indispensable. Another concern regarding the cloud security is that bridge managers may not desire to store their sensitive data on public cloud platforms. To address such issues, the authors' current work attempts to develop a hybrid cloud computing framework in which the sensitive data sets are stored in private data centers, while the public cloud service is used to guarantee high scalability and availability of a system.

Acknowledgement

This research is supported by a Grant No. 13SCIPA01 from Smart Civil Infrastructure Research Program funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korea government and Korea Agency for Infrastructure Technology Advancement (KAIA). The research is also partially supported by a collaborative project funded by the US National Science Foundation (Grant No. ECCS-1446330 to Stanford University and Grant No. CMMI-1362513 and ECCS-1446521 to the University of Michigan). The authors thank the Michigan Department of Transportation (MDOT) for access to the

Telegraph Road Bridge, Newburg Road Bridge and for offering support during installation of the wireless monitoring system. The in-kind support by Computers and Structures, Inc. for providing the CSI Bridge software to the research team at Stanford University is gratefully appreciated. Any opinions, findings, conclusions or recommendations expressed in this paper are solely those of the authors and do not necessarily reflect the views of NSF, MOLIT, MDOT, KAIA or any other organizations and collaborators.

Reference

Agrawal, D., Das, S. and Abbadi, A. E. (2011). "Big data and cloud computing: current state and future opportunities," *In Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11)*, New York, NY, USA, pp. 530-533.

Alampalli, S., Alampalli, S. and Ettouney, M. (2016). "Big data and high-performance analytics in structural health monitoring for bridge management," *Proceedings of the SPIE Smart Structures/NDE Conference*, Las Vegas, NV, USA, March 20-24, art no. 980315.

Arumugam, R., Enti, V. R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F. F., Kumar, A. S., Meng, K. D. and Kit, G. W. (2010). "DAvinCi: A cloud computing framework for service robots," *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010, pp. 3084-3089.

Bartholomew, M., Blasen, B. and Koc, A (2015). "Bridge Information Modeling (BrIM) Using Open Parametric Objects," Report No. FHWA-HI F -16-010, Federal Highway Administration.

Belqasmi, F., Singh, J., Melhem, S. Y. B. and Glitho, R.H. (2012). "Soap-based vs. restful web services: A case study for multimedia conferencing," *IEEE internet computing*, 16(4), pp.54-63.

Brownjohn, J.M.W., Zasso, A., Stephen, G.A. and Severn, R.T. (1995). "Analysis of experimental data from wind-induced response of a long span bridge," *Journal of wind engineering and industrial aerodynamics*, 54, pp.13-24.

Chaniotis, I. K., Kyriakou, K. I. D. and Tselikas, N. D. (2015). "Is Node.js a viable option for building modern web applications? A performance evaluation study," *Computing*, 97(10), pp. 1023–1044.

Chen, S. S. and Shirolé, A. M. (2013). "Implementation Roadmap for Bridge Information Modeling (BrIM) Data Exchange Protocols," Federal Highway Administration.

Cross, E. J., Koo, K. Y., Brownjohn, J. M. W. and Worden, K. (2013). "Long-term monitoring and data analysis of the Tamar Bridge," *Mechanical Systems and Signal Processing*, 35(1), pp. 16-34.

Das, M., Cheng, J. C. P. and Kumar, S. S. (2015). "Social BIMCloud: a distributed cloud-based BIM platform for object-based lifecycle information exchange," *Visualization in Engineering*, 3(8), pp. 1-20.

DataStax. (2017). "DataStax Node.js Driver for Apache Cassandra," [programming library], Retrieved from: https://github.com/datastax/nodejs-driver (accessed on 30 January 2017).

Deckler G. (2016), "Cloud vs. On-Premises – Hard Dollar Costs," [online article], Retrieved from: https://www.linkedin.com/pulse/cloud-vs-on-premises-hard-dollar-costs-greg-deckler/?trk=pulse_spock-articles (accessed on 5 December 2017).

Dervilis, N., Worden, K. and Cross, E. J. (2015). "On robust regression analysis as a means of exploring environmental and operational conditions for SHM data," *Journal of Sound and Vibration*, 347, pp. 279-296.

Farrar, C.R., Cornwell, P.J., Doebling, S.W. and Prime, M.B. (2000). "Structural health monitoring studies of the Alamosa Canyon and I-40 bridges," No. LA-13635-MS. Los Alamos National Lab., NM (US).

Fielding, R. T. (2000). "Architectural styles and the design of network-based software architectures," *Doctoral dissertation*, University of California, Irvine.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. (1999). "Hypertext transfer protocol--HTTP/1.1" (No. RFC 2616).

Fraser, M., Elgamal, A., He, X. and Conte, J. P. (2009). "Sensor network for structural health monitoring of a highway bridge," *Journal of Computing in Civil Engineering*, 24(1), pp.11-24.

Gillis, T. (2015), "Cost Wars: Data Center vs. Public Cloud," [online article], Retrieved from: https://www.forbes.com/sites/tomgillis/2015/09/02/cost-wars-data-center-vs-public-cloud/#60c91b11923f (accessed on 5 December 2017).

Grolinger, K., Higashino, W. A., Tiwari, A. and Capretz, M. A. (2013). "Data management in cloud environments: NoSQL and NewSQL data stores," *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1), pp. 22–41.

Guinard, D., Trifa, V. and Wilde, E. (2010). "A resource oriented architecture for the Web of Things," *2010 Internet of Things (IOT)*, Tokyo, 2010, pp. 1-8.

Haas, H. and Brown, A. (2004). "Web Services Glossary," *W3C Working Group Note*, [online article], Retrieved from: https://www.w3.org/TR/ws-gloss/ (accessed on 30 January 2017).

Hewitt, E. (2010). "Cassandra: the definitive guide," O'Reilly Media, Inc.

Jeong, S., Zhang, Y., O'Connor, S. M., Lynch, J. P., Sohn, H. and Law, K. H. (2016a). "A NoSQL Data Management Infrastructure for Bridge Monitoring," *Smart Structures and Systems*, 17(4), pp. 669-690.

Jeong, S., Zhang, Y., Hou, R., Lynch, J. P., Sohn, H. and Law, K. H. (2016b). "A Cloud based Information Repository for Bridge Monitoring Applications," *Proceedings of the SPIE Smart Structures/NDE Conference*, Las Vegas, NV, USA, March 20-24, art no. 980313.

Jeong, S., Hou, R., Lynch, J. P., Sohn, H. and Law, K. H. (2016c). "Cloud-based cyber infrastructure for bridge monitoring," *Proceedings of the 14th International Symposium on Structural Engineering (ISSE-14)*, Beijing, China, Oct.12-15, 2016.

Jeong, S., Hou, R., Lynch, J. P., Sohn, H. and Law, K. H. (2017). "An information modelling framework for bridge monitoring," *Advances in engineering software* (accepted).

Koo, K. Y., Battista, N. D. and Brownjohn, J. M. W. (2011). "SHM data management system using MySQL database with MATLAB and web interfaces," *In 5th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-5)*, Cancún, México, pp. 589-596.

Law, K. H., Smarsly, K. and Wang, Y. (2014). "Sensor data management technologies for infrastructure asset management," *In Sensor Technologies for Civil Infrastructures: Applications in Structural Health Monitoring*, (Eds., Wang,

M.L., Lynch, J.P. and Sohn, H.), Woodhead Publishing, Cambridge, UK, 2(1), 3-32.

Law, K. H., Cheng, J. C. P., Fruchter, R. and Sriram, R. D. (2016). "Engineering Applications of the Cloud," *In Encyclopedia of Cloud Computing*, (eds S. Murugesan and I. Bojanova), John Wiley & Sons, Ltd, Chichester, UK.

Le, T. D., Kim, S. H., Nguyen, M. H., Kim, D., Shin, S. Y., Lee, K. E. and da Rosa Righi, R. (2014) "EPC information services with No-SQL datastore for the Internet of Things," *2014 IEEE International Conference on RFID (IEEE RFID)*, Orlando, FL, 2014, pp. 47-54.

Lea, R. and Blackstock, M. (2014). "City Hub: A Cloud-Based IoT Platform for Smart Cities," *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, Singapore, pp. 799-804.

Lei, K., Ma, Y., and Tan, Z. (2014). "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js," *Proceedings of 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE)*, pp. 661–668.

Li, H., Ou, J., Zhao, X., Zhou, W., Li, H., Zhou, Z. and Yang, Y. (2006). "Structural health monitoring system for the Shandong Binzhou Yellow River highway bridge," *Computer-Aided Civil and Infrastructure Engineering*, 21(4), pp.306-317.

Liao, Y., Mollineaux, M., Hsu, R., Bartlett, R., Singla, A., Raja, A., Bajwa, R. and Rajagopal, R. (2014). "Snowfort: An open source wireless sensor network for data analytics in infrastructure and environmental monitoring," *in IEEE Sensors Journal*, 14(12), pp. 4253-4263.

Lim, H. J., Sohn, H. and Liu, P. (2014). "Binding conditions for nonlinear ultrasonic generation unifying wave propagation and vibration," *Applied Physics Letters*, 104(21), 214103.

Lin, W., Dou, W., Zhou, Z. and Liu, C. (2015). "A cloud-based framework for Home-diagnosis service over big medical data," *Journal of Systems and Software*, 102, pp.192-206.

Lynch, J. P. and Loh, K. J. (2006). "A summary review of wireless sensors and sensor networks for structural health monitoring," *Shock and Vibration Digest*, 38(2), pp. 91-130.

Mell, P. and Grance, T. (2011). "The NIST definition of cloud computing - Recommendations of the National Institute of Standards and Technology," *NIST Special Publication 800-145*, Computer Science Division, Information Technology Laboratory, National Institute of Standards.

Mulligan, G. and Gracanin, D. (2009). "A comparison of SOAP and REST implementations of a service based interaction independence middleware framework," *Proceedings of the 2009 Winter Simulation Conference (WSC)*, Austin, TX, 2009, pp. 1423-1432.

O'Connor, S. M., Lynch, J. P., Ettouney, M., vander Linden, G. and Alampalli, S. (2012). "Cyber-enabled decision making system for bridge management using wireless monitoring systems: Telegraph Road Bridge demonstration project," *In Structural Materials Technology 2012*, pp. 177-184.

O'Connor, S. M., Zhang, Y., Lynch, J. P., Ettouney, M. M. and Jansson, P. O. (2017). "Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics," *Structure and Infrastructure Engineering*, 13(5), pp.604-624.

Overschee, P.V. (2002), "Subspace Identification for Linear Systems," [Matlab package]. Retrieved from: http://www.mathworks.com/matlabcentral/fileexchange/2290-subspace-identification-for-linear-systems (accessed on 27 June 2017).

Open Geospatial Consortium (2014). "OGC® SensorML: Model and XML Encoding Standard" [online article], Retrieved from: http://www.opengeospatial.org/standards/sensorml (accessed on 1 June 2017)

ParamML, "ParamML Author's Guide," [Online article], Retrieved from: https://sites.google.com/a/redeqn.com/paramml-author-s-guide/ (accessed on 1 June 2017)

Pautasso, C. (2008). "BPEL for REST," In International Conference on Business Process Management, pp. 278–293.

Pautasso, C. and Alonso, G. (2005). "The JOpera visual composition language," *Journal of Visual Languages and Computing*, 16(1-2), 119–152.

Pautasso, C., Zimmermann, O. and Leymann, F. (2008). "Restful web services vs. "Big" web services: making the right architectural decision," *In Proceedings of the 17th international conference on World Wide Web*, pp. 805-814. Beijing, China, April 21 - 25, 2008.

Pautasso, C. (2009). “Composing RESTful Services with JOpera,” *Proceeding of International Conference on Software*, Zurich, Switzerland, pp 142–159.

Rosenberg, F., Curbera, F., Duftler, M. J. and Khalaf, R. (2008). “Composing RESTful Services and Collaborative Workflows: A Lightweight Approach,” *In IEEE Internet Computing*, 12(5), pp. 24–31.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S. and Xu, X. (2014). “Web services composition: A decade’s overview,” *Information Sciences*, 280, pp. 218-238.

Smarsly, K., Law, K. H. and Hartmann, D. (2012). “Multiagent-based collaborative framework for a self-managing structural health monitoring system,” *Journal of computing in civil engineering*, 26(1), pp.76-89.

Smith, J. E. and Nair, R. (2005). “The architecture of virtual machines,” *Computer*, 38(5), pp. 32-38.

Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N. and Helland, P. (2007). “The end of an architectural era (it's time for a complete rewrite),” *Proceedings of the 33rd International Conference on Very large data bases*, pp. 1150–1160.

Strukhoff, R. (2017), “Adopting an IoT Platform: Things to Know and Pitfalls to Avoid,” [online article], Retrieved from: https://www.altoros.com/blog/adopting-an-iot-platform-things-to-know-and-pitfalls-to-avoid/ (accessed on 5 December 2017).

Swartz, R., Jung, D., Lynch, J. P., Wang, Y., Shi, D., and Flynn, M. P. (2005). “Design of a wireless sensor for scalable distributed in-network computation in a structural health monitoring system.” *5th International Workshop on Structural Health Monitoring (IWSHM)*, Stanford, CA.

W3C. (2014). “XML Schema,” [Online article], Retrieved from: https://www.w3.org/2001/XMLSchema (accessed on 1 June 2017)

Wong, K.Y., Lau, C.K. and Flint, A.R. (2000). “Planning and implementation of the structural health monitoring system for cable-supported bridges in Hong Kong,” *Proceedings of the SPIE 3995*, Nondestructive Evaluation of Highways, Utilities, and Pipelines IV.

Xu, X. (2012). “From cloud computing to cloud manufacturing,” *Robotics and computer-integrated manufacturing*, 28(1), pp.75-86.

Ye, X. and Huang, J. (2011). "A framework for cloud-based smart home," *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Harbin, 2011, pp. 894-897.

Zaslavsky, A., Perera, C. and Georgakopoulos, D. (2013). "Sensing as a service and big data," *In International Conference on Advances in Cloud Computing* (ACC-2012), Bangalore, India, July 2012, pp. 21–29

Zhang, Q., Cheng, L. and Boutaba, R. (2010). "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, 1(1), 7–18.

Zhang, Y., O'Connor, S. M., van der Linden, G., Prakash, A. and Lynch, J. P. (2016). "SenStore: A Scalable Cyberinfrastructure Platform for Implementation of Data-to-Decision Frameworks for Infrastructure Health Management," Journal of Computing in Civil Engineering, 04016012

Zhou, G. D. and Yi, T. H. (2013). "Recent developments on wireless sensor networks technology for bridge health monitoring," *Mathematical Problems in Engineering*, art no. 947867.

Zhao, H. and Doshi, P. (2009). "Towards automated restful web service composition," *2009 IEEE International Conference on Web Services*, Los Angeles, CA, 2009, pp. 189-196.

Table 1. RESTful web services currently implemented on the cyberinfrastructure platform

Service	Method	URI	Parameter
Sensor data store	POST	/sensordata	-
Sensor data retrieval	GET	/sensordata/{sensorID}	event_time_begin, event_time_end
Traffic image store	POST	/imagedata	-
Traffic image retrieval	GET	/imagedata/{cameraID}	event_time_begin, event_time_end
Sensor information store	POST	/sensor	
Sensor list retrieval	GET	/sensor	sensor_type, install, remove
Sensor information retrieval	GET	/sensor/{sensorID}	install, remove
Geometric model store	POST	/geometricmodel	
Geometric model retrieval	GET	/geometricmodel/{BridgeID}	
Engineering model store	POST	/femodel	
Engineering model retrieval	GET	/femodel/{BridgeID}	file_format (xml or xlsx)

Table 2. Specification and role of virtual machines provisioned on the Microsoft Azure cloud platform

No.	Spec	Role
1,2,3,4,5	Azure Standard_A2m_v2 (2 cores, 16 GB memory)	Database node
6,7,8	Azure Standard DS2 v2 (2 cores, 7 GB memory)	Web server

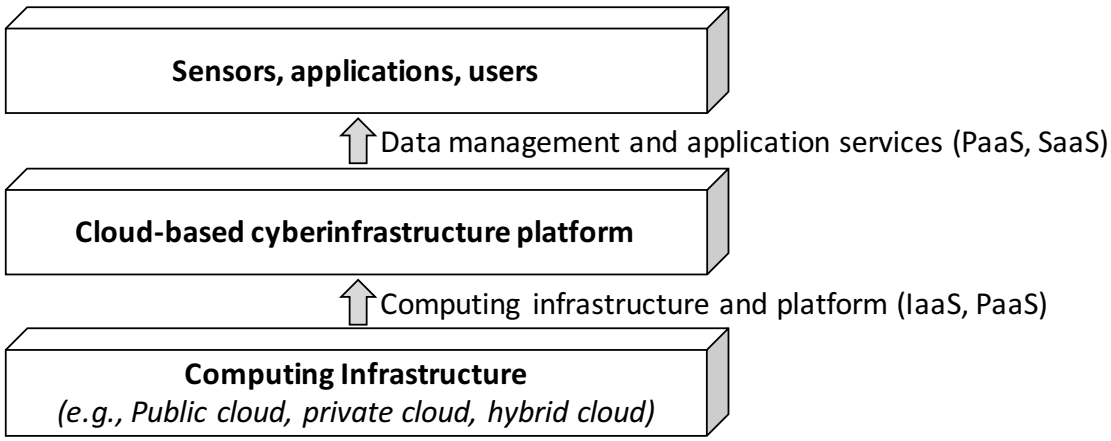
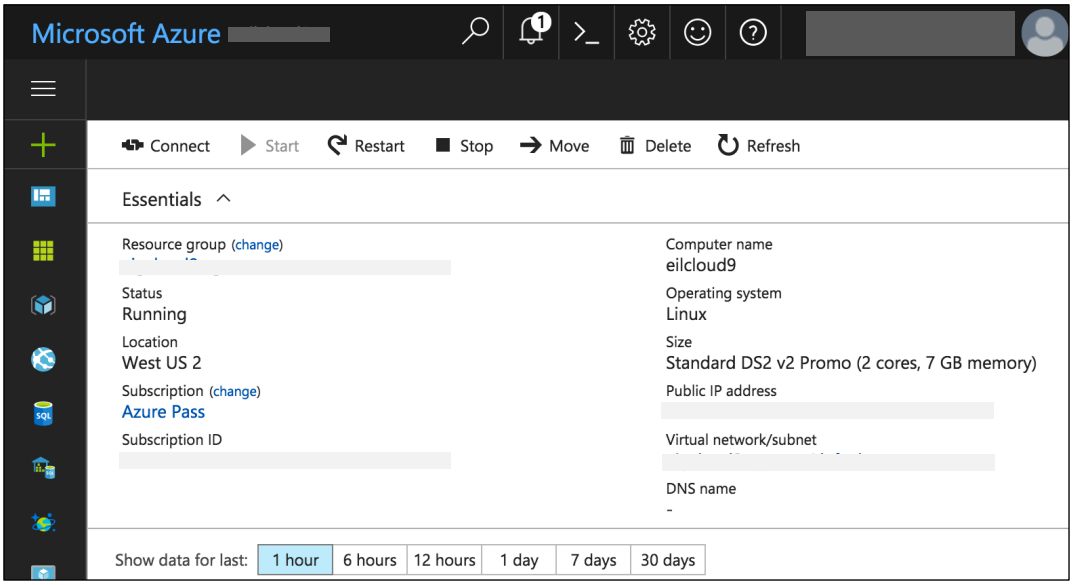
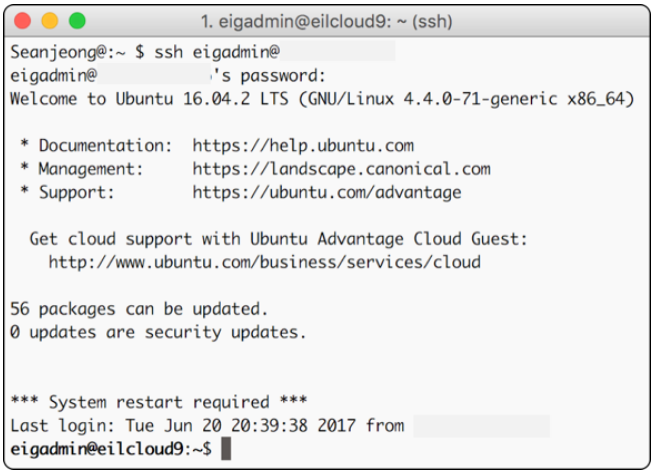


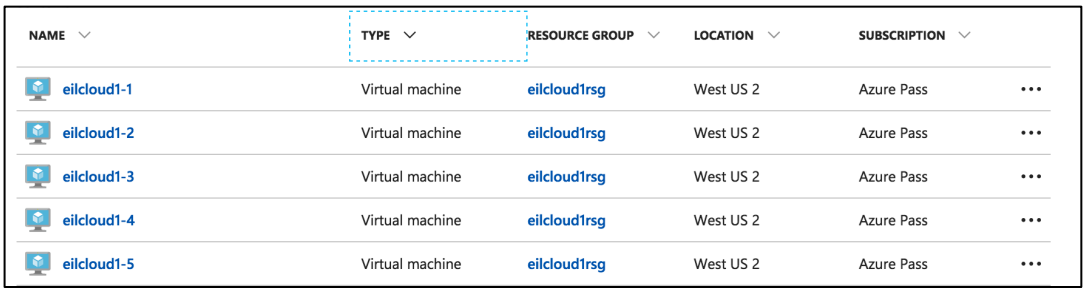
Figure 1. A model of cloud computing for SHM



(a) Web-based cloud interface



(b) Shell interface



(c) List of virtual machines deployed on Azure cloud platform

Figure 2. Virtual machine created on Microsoft Azure cloud platform

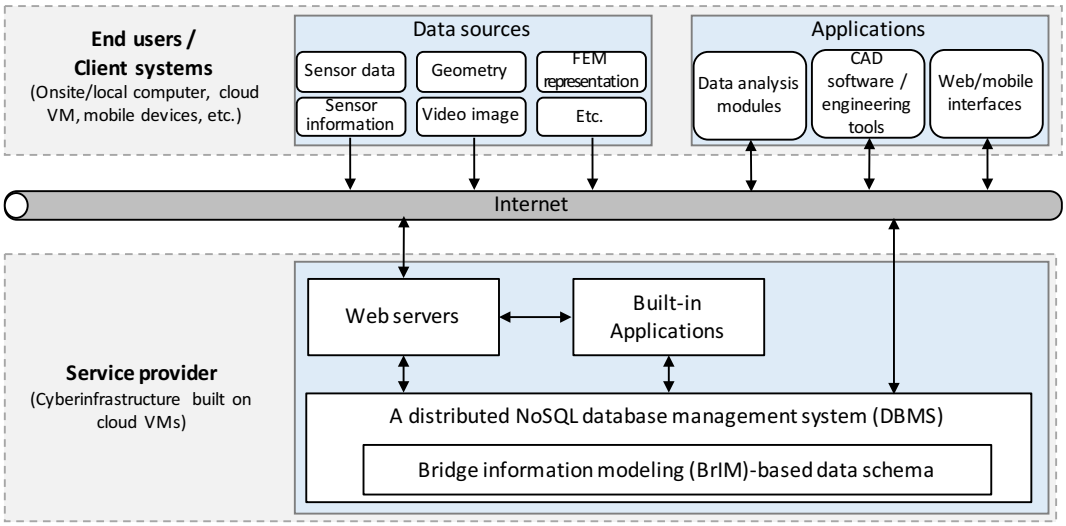


Figure 3. Conceptual framework of cloud-based cyberinfrastructure platform

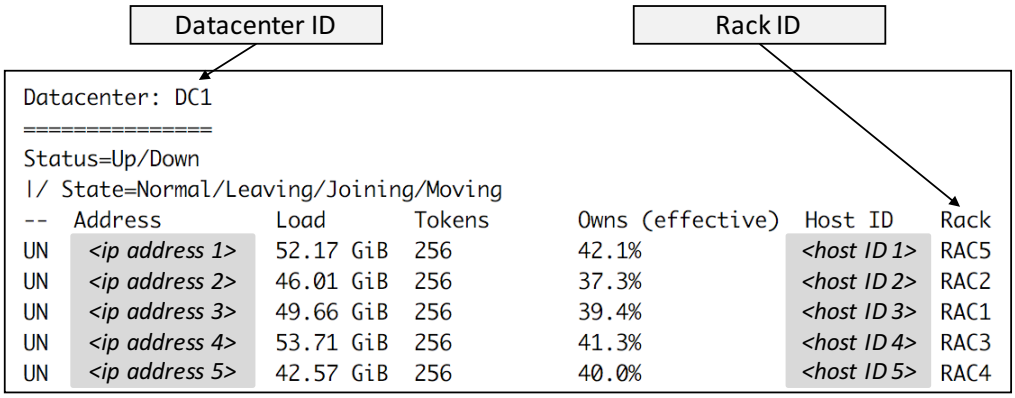


Figure 4. A Cassandra cluster instance

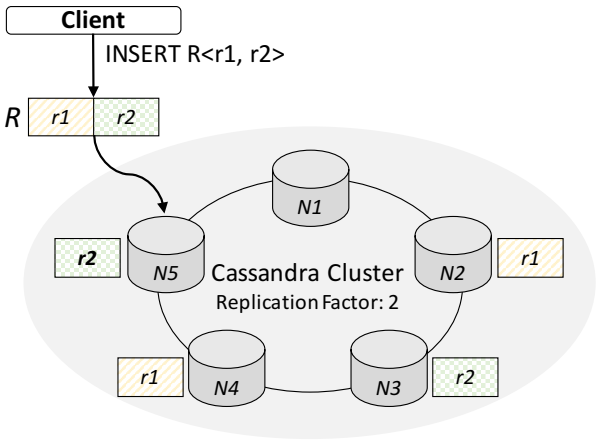
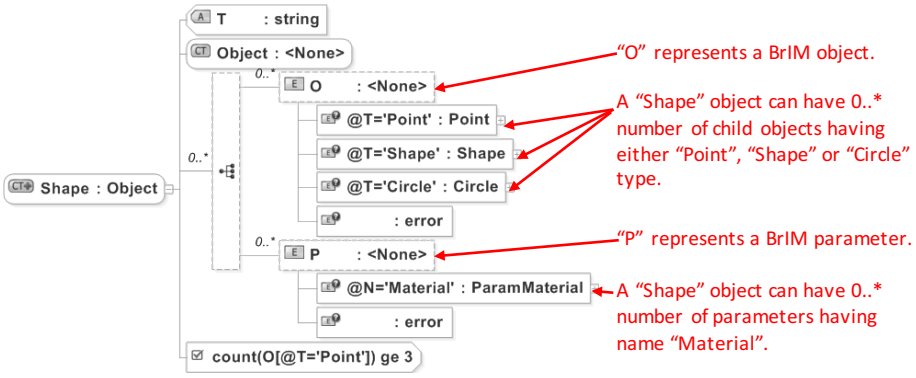
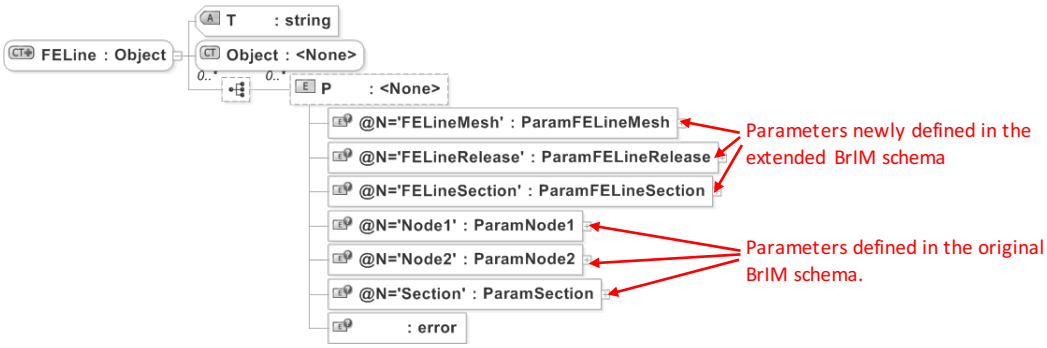


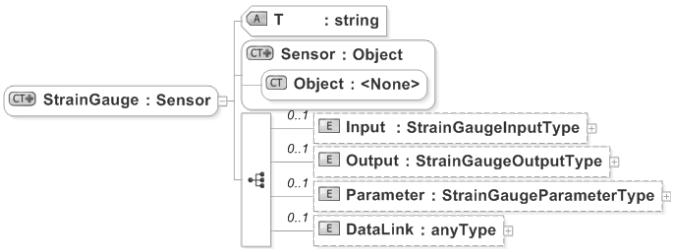
Figure 5. Ring topology of Cassandra database



(a) Entity: Shape



(b) Entity: FELine



(c) Entity: StrainGauge

Figure 6. Data schema definition in BrIM for bridge monitoring applications: (a) Shape, (b) FELine, (c) StrainGauge entity types

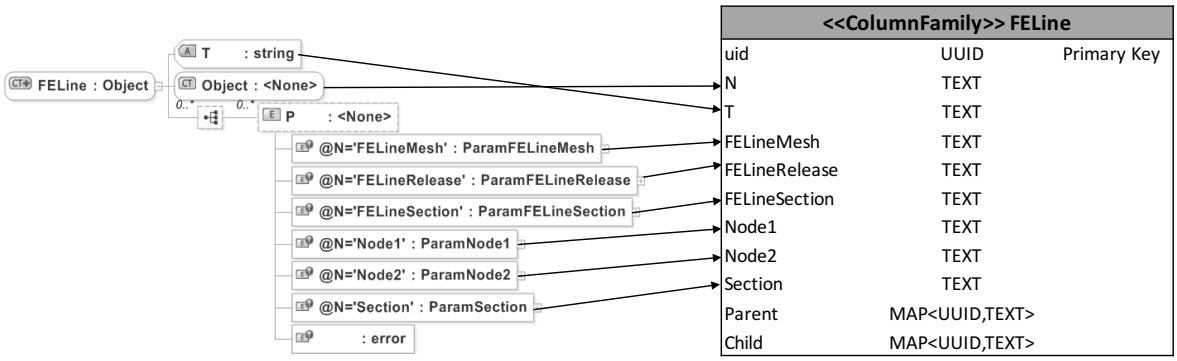


Figure 7. Data mapping between BrIM schema "FELine" and corresponding Cassandra column family.

shp001	T	Material	Child
	”Shape"	“Concrete”	["pt001”: “Point”, "pt002”: “Point”, …]

pt001	T	X	Y	Parent
	"Point"	"-10"	"-10"	[“shp001”, “Shape”]

pt002	T	X	Y	Parent
	"Point"	"-10"	"10"	[“shp001”, “Shape”]

(a) Rows storing Shape object and its child Point objects

FELine001	T	FELineRelease	FELineSection	Node1	Node2
	”FELine"	“FELineReleaseType1”	“Steel I-Beam type1”	”Node090”	”Node091”

FELine002	T	FELineSection	Node1	Node2
	”FELine"	“Steel I-Beam type1”	”Node091”	”Node092”

(b) Rows storing heterogeneous FELine objects

Figure 8. Database schema for BrIM objects

Incoming measurement data

u07ch0 | 2014	2014-08-02T00:00:08	2014-08-01T00:00:09	2014-08-01T00:00:10
	Array[32792.0, 32776.0, 32803.0, ...]	Array[32849.0, 32849.0, 32867.0, ...]	Array[32851.0, 32863.0, 32842.0 ...]

(a) Row storing sensor measurement data

Incoming image data

TRB_01 | 201608	2016-08-23T10:02:08	2016-08-23T10:02:13	2016-08-23T10:02:19
	BLOB(/9j/4AAQSkZJ ... H//Z)	BLOB(/9j/4AAQSkZJ ... /9k=)	BLOB(/9j/4AAQSkZJ ... Af/Z)

(b) Row storing traffic video image data

Figure 9. Database schema for time-series data

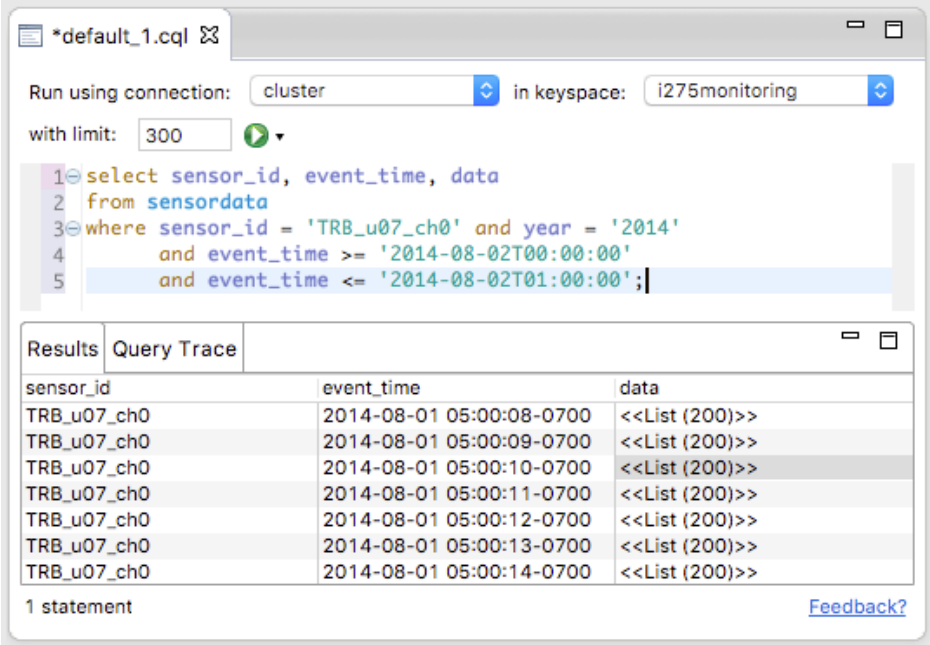


Figure 10. Select query for sensor data retrieval and query result in a tabular format

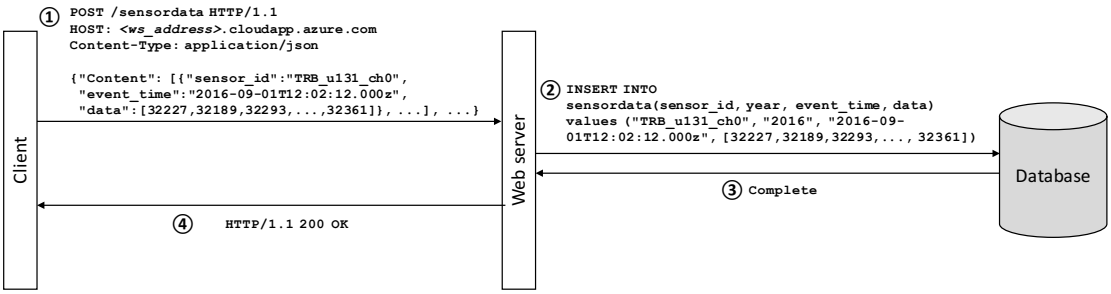


Figure 11. Web service example: sensor data store service

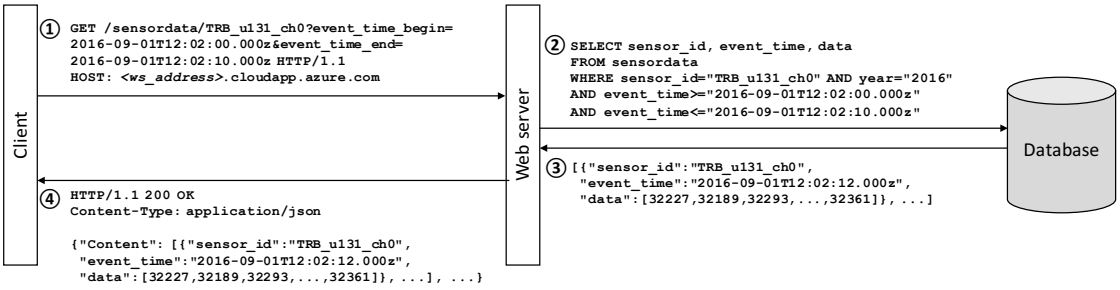
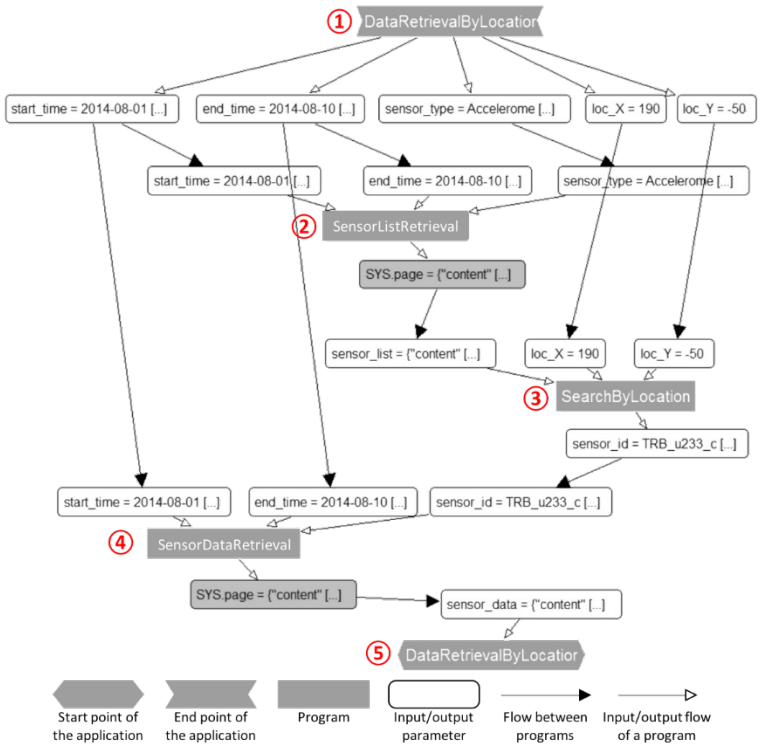
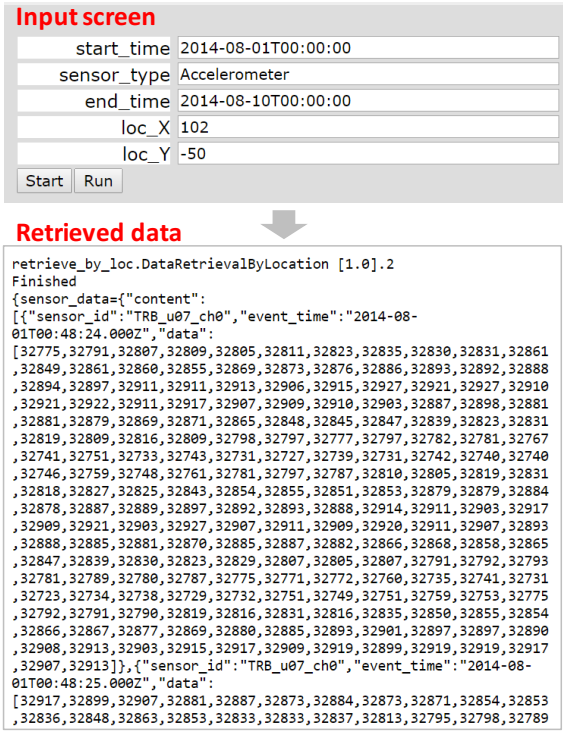


Figure 12. Web service example: sensor data retrieval service

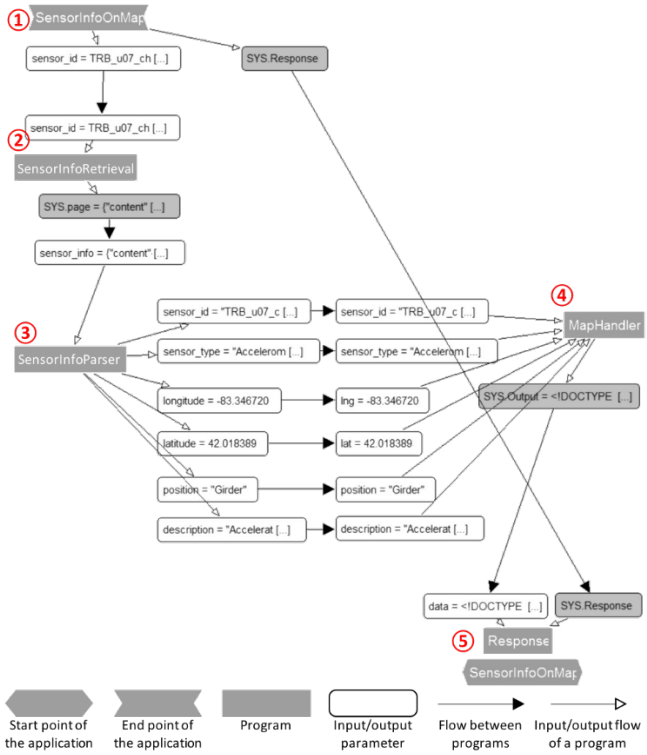


(a) Data flow

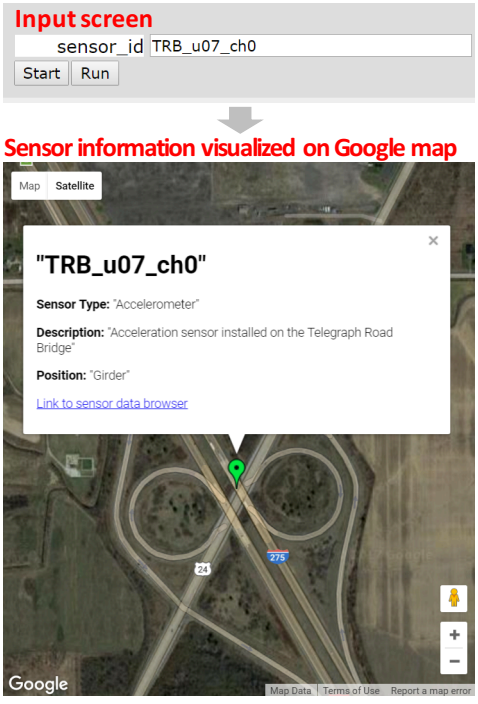


(b) Execution example

Figure 13. A composite application DataRetrievalByLocation



(a) Data flow



(b) Execution example

Figure 14. A composite application SensorInfoOnMap

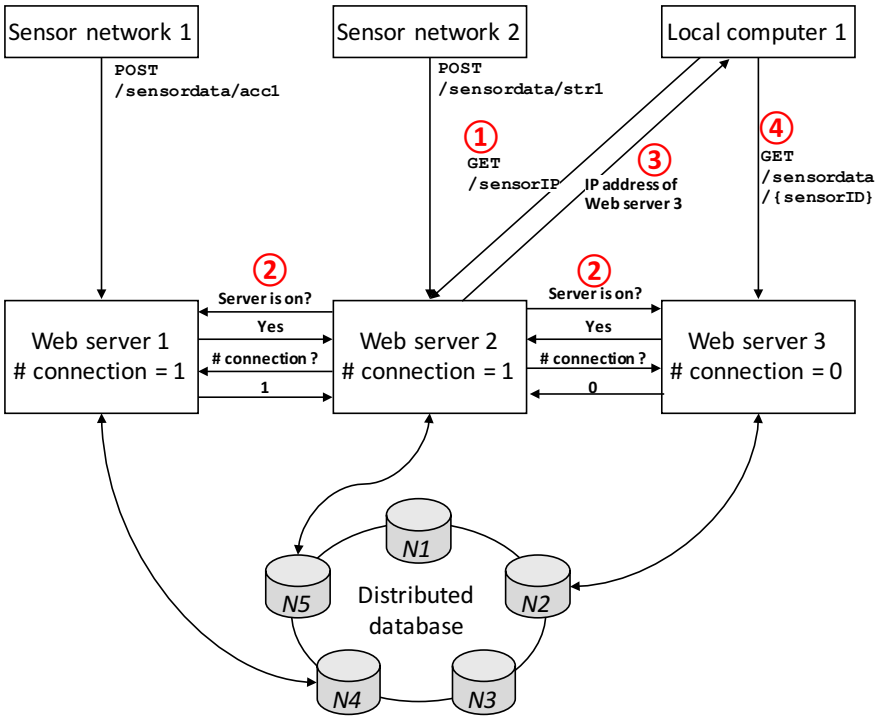


Figure 15. Duplicated web servers

(a) Telegraph Road Bridge (TRB)

(b) Newburg Road Bridge (NRB)

Figure 16. Bridges on the I-275 corridor installed with bridge monitoring systems

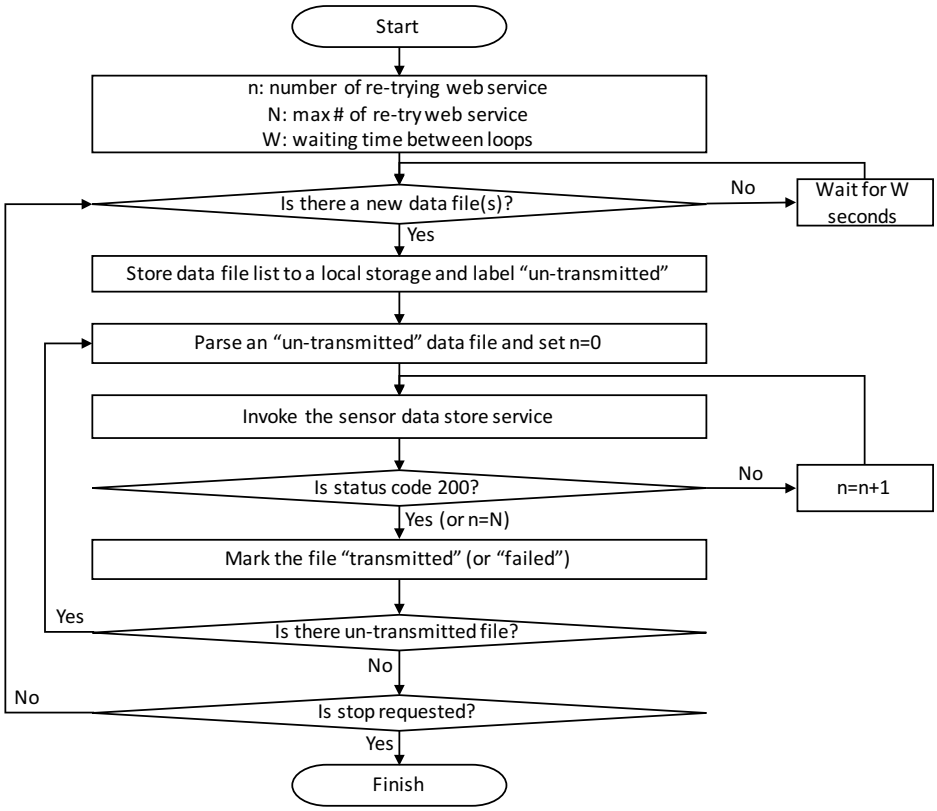
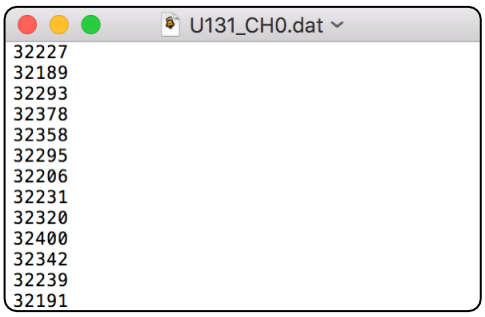
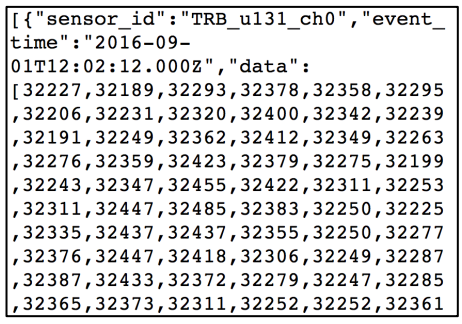


Figure 17. Workflow: application for data store automation



(a) Raw data file



(b) Parsed sensor data

Figure 18. Sensor data file

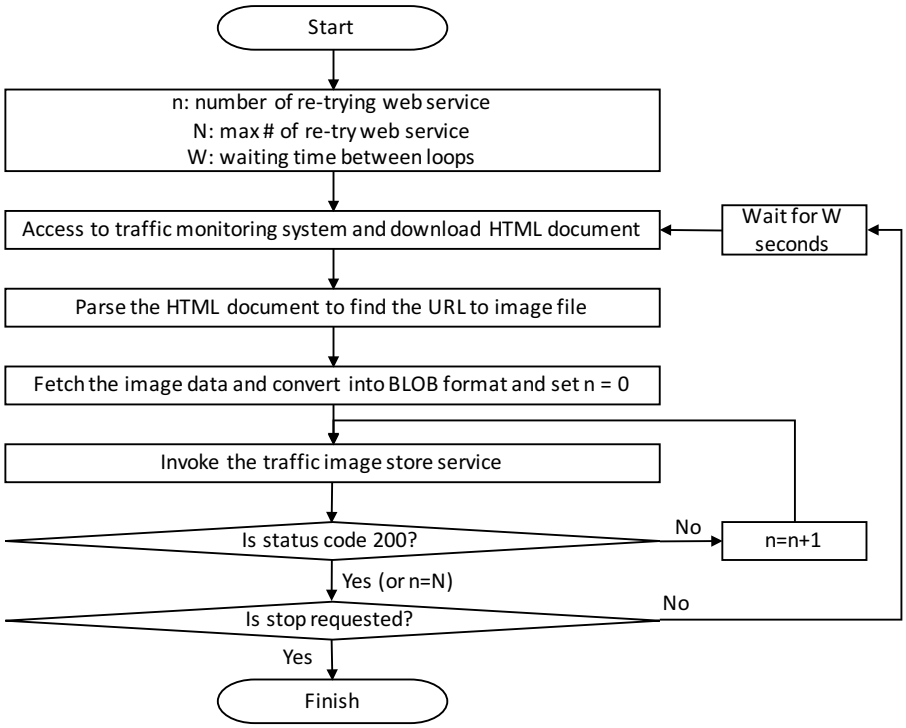
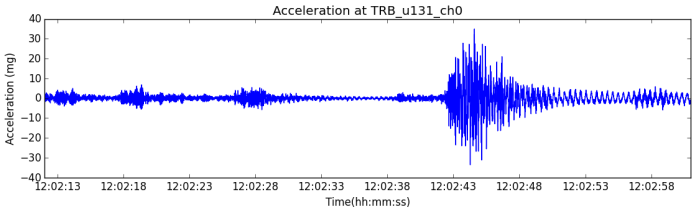


Figure 19. Workflow: traffic video image collecting application

```
GET /sensordata/TRB_u131_ch0?
event_time_begin=2016-09-
01T12:02:00.000z
&event_time_end=2016-09-
01T12:02:10.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
```



(a) Sensor data retrieved by invoking the sensor data retrieval service

```
GET /imagedata/Telegraph2?
event_time_begin=2016-08-
18T18:01:00.000z
&event_time_end=2016-08-
18T18:01:20.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
```

(b) Traffic images retrieved by invoking the traffic image retrieval service

Figure 20. Data retrieval using web services

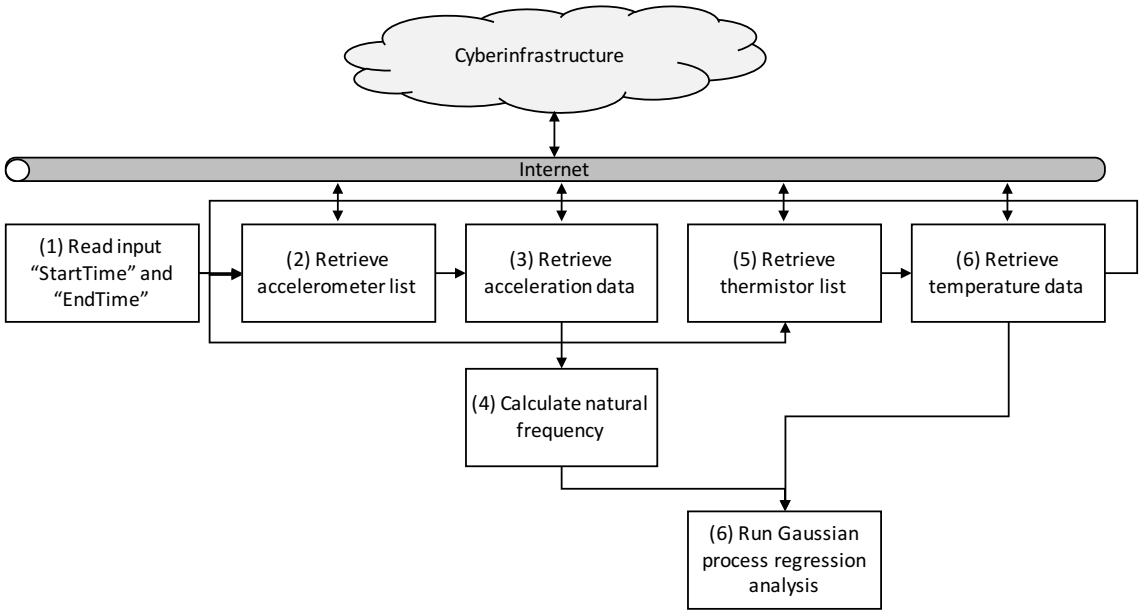


Figure 21. A workflow for relating structural behaviour with temperature data

```
GET /sensor?sensorType=Accelerometer&
install=2014-08-01T00:00:00.000z&
remove=2014-08-01T02:02:00.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
              <-- Rest is omitted -->
```

(a) HTTP request for accelerometer list retrieval

```
GET /sensordata/TRB_u07_ch0?
event_time_begin=2014-08-01T00:00:00.000z&
event_time_end=2014-08-01T02:02:00.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
              <-- Rest is omitted -->
```

(b) HTTP request for acceleration data retrieval

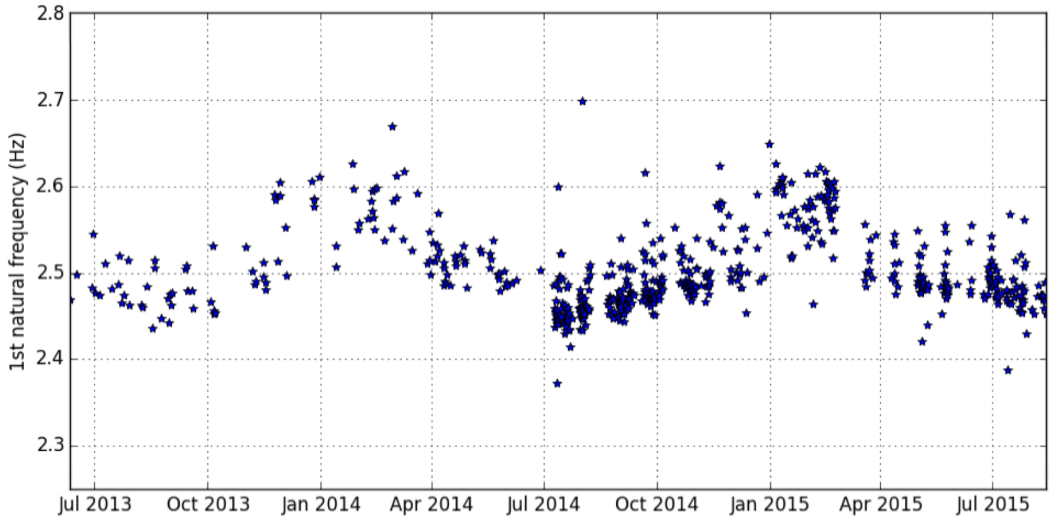
```
GET /sensor?sensorType=Thermistor&
install=2014-08-01T00:00:00.000z&
remove=2014-08-01T02:02:00.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
              <-- Rest is omitted -->
```

(c) HTTP request for thermistor list retrieval

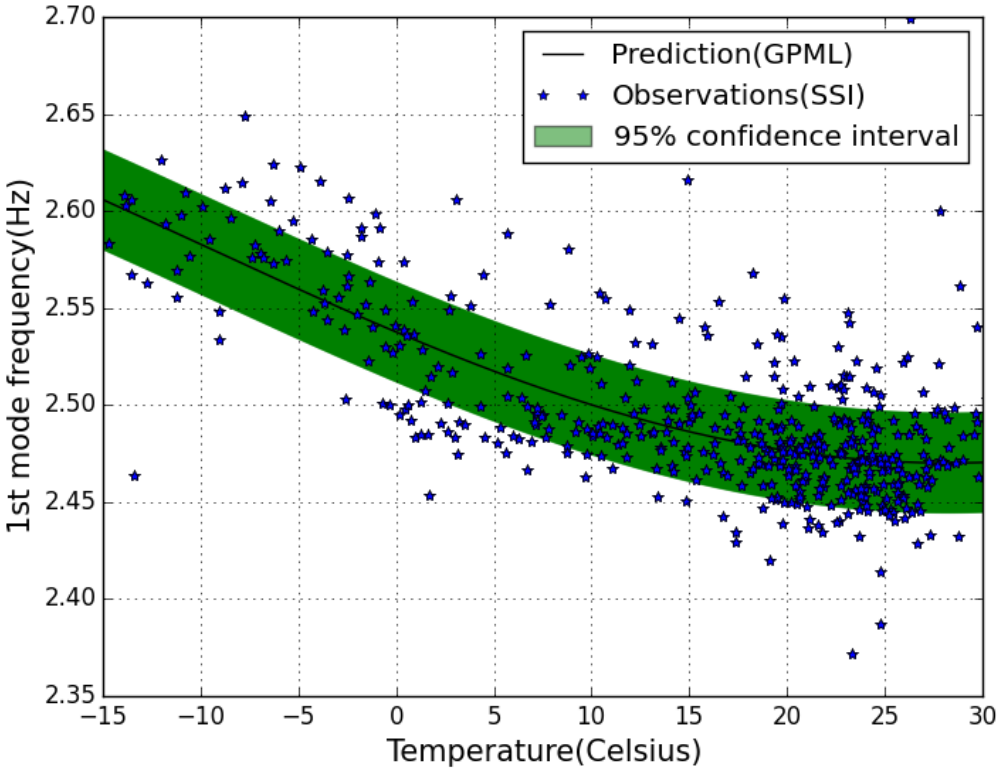
```
GET /sensordata/TRB_u45_ch0?
event_time_begin=2014-08-01T00:00:00.000z&
event_time_end=2014-08-01T02:02:00.000z HTTP/1.1
HOST: <ws_address>.cloudapp.azure.com
              <-- Rest is omitted -->
```

(d) HTTP request for temperature data retrieval

Figure 22. HTTP requests and corresponding CQL queries for sensor list and data retrieval

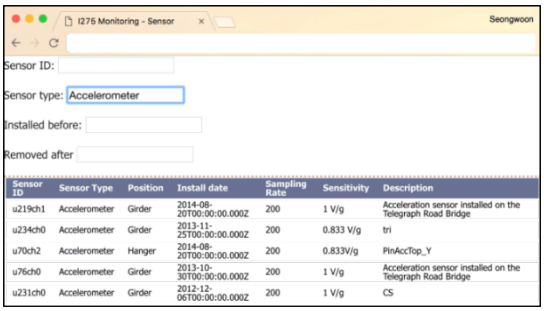


(a) History of first modal frequency (from August 2013 to August 2015)

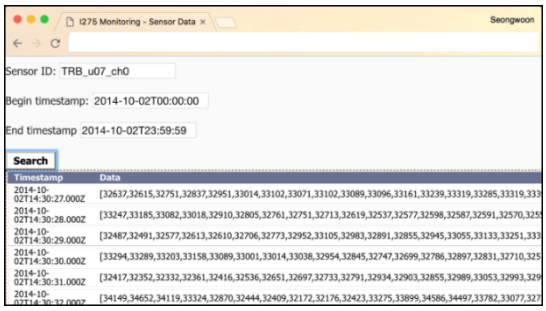


(b) Gaussian process regression showing the confidence interval of modal frequency according to temperature changes

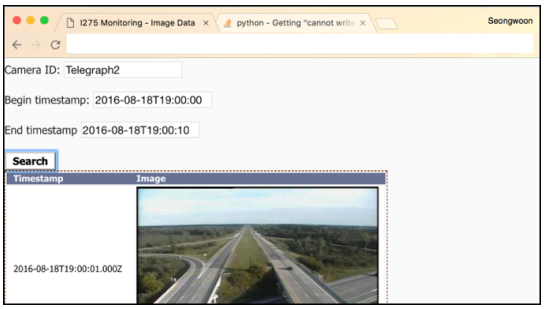
Figure 23. Patterns of modal frequency of the Telegraph Road Bridge



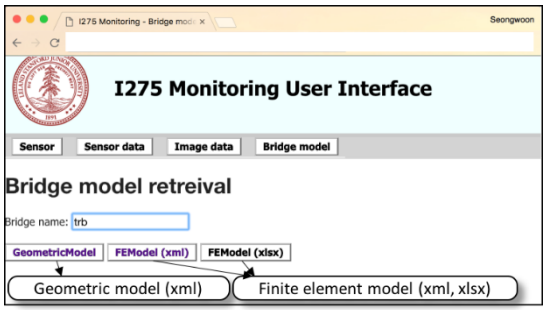
(a) Sensor information retrieval



(b) Sensor data retrieval

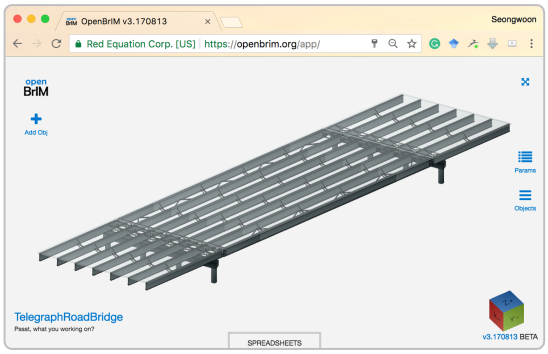


(c) Traffic image retrieval

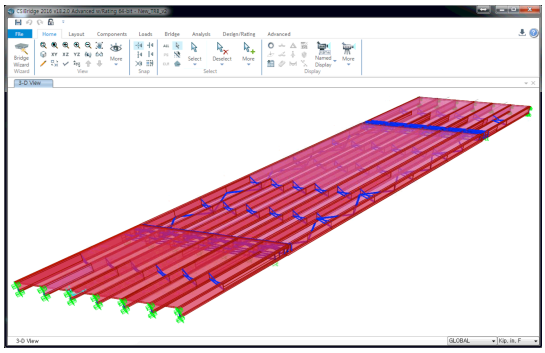


(d) Bridge model retrieval

Figure 24. Prototype web-based user interface

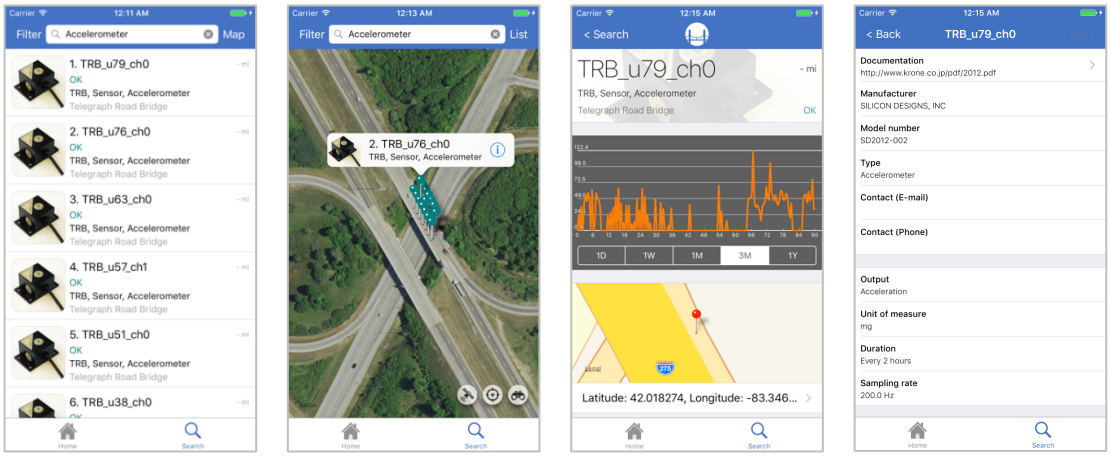


(a) Geometric model visualized by OpenBrIM viewer



(b) Engineering model visualized by CSI Bridge

Figure 25. Telegraph road bridge model downloaded from the web-based user interface



(a) Sensor list view (b) Sensor map view (c) Sensor detail view (d) Sensor information view

Figure 26. Prototype mobile interface