CLOUD-BASED CYBER INFRASTRUCTURE FOR BRIDGE MONITORING

Seongwoon Jeong[1], Rui Hou[2], Jerome P. Lynch[2], Hoon Sohn[3], and Kincho H. Law[1]
[1] Dept. of Civil & Environmental Eng., Stanford University, Stanford, CA, USA 94305
[2] Dept. of Civil & Environmental Eng., University of Michigan, Ann Arbor, MI, USA, 48109
[3] Dept. of Civil & Environmental Eng., KAIST, Daejeon 305-701, Republic of Korea

Abstract:
This paper describes a cloud-based cyber infrastructure for the management of information involved in bridge monitoring applications. Recent years have seen an emergence and increasing use of sensor technologies for bridge monitoring. In addition to the measurement data from the sensors, bridge monitoring and management systems require many different types of information including bridge geometries, engineering analytical models and the metadata about the sensors. In this study, we employ cloud computing services and have implemented a cloud-based cyber infrastructure platform to effectively manage the information involved in bridge monitoring applications. Furthermore, NoSQL database systems are deployed for scalable and flexible data storage. To facilitate interoperability, a bridge information model is designed based on open standards and bridge engineering analysis and design tools. For demonstration purposes, the model of the Telegraph Road Bridge located in Michigan and the sensor data collected on the bridge are employed to illustrate the use of the cyber infrastructure system for bridge monitoring.
Keywords: Cloud computing, cyber infrastructure, bridge monitoring, NoSQL database

1 INTRODUCTION

Cloud computing, where application services are delivered over advanced communication networks, has emerged as a new computing paradigm that promises to have significant impact to engineering practices. The cloud computing model enables ubiquitous and convenient access of information and applications. Focusing on bridge monitoring and management applications, this paper discusses the development of a cloud-based cyber infrastructure that provides ubiquitous access of information and supports interoperability among engineering services.

In recent years, there has been an emergence and increasing use of sensor and sensor network technologies for bridge monitoring. As sensing and communication technologies continue to advance and their costs continue to decrease, the deployment of sensors and sensor networks for bridge monitoring will continue to grow. While most current research efforts deal with continuing development of advanced sensing devices and instrumentation strategies to afford better assessment of structural conditions, relatively few efforts have been reported on the data issues involved in bridge monitoring and management (Law *et al*. 2014). The amount of data to be collected will continue to grow as bridges are being instrumented with dense networks of sensors, some of them with very high sampling rates. In addition to the voluminous measurement data collected from the sensors, many different types of information, including bridge geometry and engineering models, are needed for bridge management. In this study, we focus on the development of a flexible and scalable data management infrastructure that, on one hand, can potentially handle the large volume of data sets and that, on the other hand, can support easy access by engineering applications and a wide variety of devices.

Traditionally, relational database management systems (RDBMSs) have become a *de facto* approach for storing data in businesses as well as in engineering applications, including bridge monitoring (McNeill, 2009). However, recent studies have shown that RDBMSs have fundamental limitations in terms of flexibility and scalability and are not efficient in dealing with time series data streams (Han *et al*., 2011). For instance, sensor measurement data are typically organized rigidly in tabular form or

stored separately in files and they are not explicitly integrated with engineering entities.

NoSQL database management systems have been shown capable of overcoming some of the limitations of RDBMSs (Han *et al*., 2011). In addition, bridge information models can be easily mapped into NoSQL data schemas, and thus facilitate information interoperability and access by engineering application services. Furthermore, recent emergence of cloud computing technologies provides a cost-effective and scalable platform to handle large volumes of data and information (Marston *et al*., 2011).

In this paper, a cloud-based cyber infrastructure for management of the information in bridge monitoring applications is presented. Section 2 describes the infrastructure framework including cloud computing services and NoSQL database technologies. Data schemas for representing sensor data and other information pertinent to bridge monitoring are discussed in Section 3. In Section 4, the utilization of the cyber-infrastructure framework is demonstrated using the sensor data and the bridge model of the Telegraph Road Bridge (located in Monroe, Michigan).

2 CLOUD-BASED CYBER INFRASTRUCTURE FRAMEWORK

Broadly speaking, a bridge monitoring system typically involves several computing components: onsite computer (or DAQ system), main server and local computer (Law *et al*., 2014). Connecting to the sensor network, the onsite computer (or DAQ system) receives and temporarily stores the sensor data, and sends them to the main server. The main server stores the sensor data permanently along with other information, such as the bridge model and the metadata about the sensors. Users' local computers, or mobile devices, periodically retrieve sensor data and other relevant bridge information from the main server (or occasionally directly from the onsite computer) and performs data analysis. This section describes the development of a main server by deploying NoSQL data management technologies and cloud computing services.

2.1 Cloud computing service

Cloud computing can be broadly defined as a utility over a network model that enables efficient access (e.g., via Internet connection) to cost-effective computing resources, such as servers, storages, applications and services (Law *et al*., 2016). According to the US National Institute of Standards and Technology (NIST), cloud computing services can be broadly categorized into three basic models, namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) (Mell and Grance, 2011). As described by Remde (2011) and illustrated in Figure 1, the service models are defined by computing resources provided by the service vendors.

The deployment of cloud services depends on the provisioning of the cloud infrastructure whether the infrastructure is publicly available, privately owned by an organization, shared by a community of users, or a hybrid combination of the former three provisioning types (Mell and Grance, 2011). Using a public cloud service model can have many benefits, particularly for research purposes (Marston *et al*., 2011). First, public cloud services alleviate the need to purchase or to manage server machines. Furthermore, users can use an optimal amount of computing resource based on demand, thereby minimizing expenses on unnecessary resources. Last, but not least, the computing resources can be flexibly scaled according to the amount of sensor data that need to be stored.

In this work, we employ the IaaS service model implemented on a public cloud. The hardware and O/S are provided by the cloud service vendor, while the database systems and applications are self-incorporated to the service. The cloud service vendor employed herein is Microsoft Azure, which is a widely used cloud service. To implement the cloud-based platform, we create a virtual machine (Linux Ubuntu 14.04 LTS, Standard A1 Tier: 1 core, 1.75GB memory) on Azure.

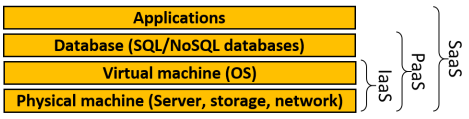


Figure 1. Cloud computing service models

2.2 NoSQL database system

Because of their scalability and flexibility compared to relational database management systems (RDBMSs), NoSQL database systems have been applied to handle "big data" applications (Han *et al*., 2011). There exist many

NoSQL database systems each having their own features and performance advantages. In this study, Apache Cassandra is selected as the persistent store for the bridge monitoring data. Cassandra is known to be effective for handling extremely large amounts of data. For example, for scaling with computing resources, Cassandra exhibits linear performance as new computing nodes are added to the existing system (Han *et al*., 2011). Moreover, the database system can easily be scaled outwardly from an existing system since Cassandra can automatically rebalance the allocation of data when additional computing nodes are added (Hewitt, 2013). Lastly, the master-less architecture of Cassandra prevents single point of failure and thus ensures high availability as a database system (Hewitt, 2013).

2.3 Cyber infrastructure framework

Based on the cloud computing service model and the NoSQL database system, a cyber infrastructure framework for bridge monitoring applications is designed. Figure 2 shows the overall configuration of the cloud-based cyber infrastructure framework. A set of virtual machines provided by Microsoft Azure serves as the main server for the bridge monitoring system. On the cloud computing platform, Apache Cassandra is installed to manage the sensor data along with other pertinent information including the bridge geometry and engineering model. Due to the scalability of Azure and Cassandra, the number of virtual machines can be easily adjusted according to the demand.

Python scripts are developed for the onsite computer to enable seamless data flow from the sensor network to the cloud-based main server. When measurement data are received from sensor network by the onsite computer, the scripts automatically parse the data and store them locally in MongoDB (a document-oriented NoSQL database system) for temporary storage of the data on the onsite computer. When the Cassandra database system is ready, the scripts also parse and send the data to the main server. Application Program Interfaces (API) (such as PyMongo and Cassandra driver) are utilized to connect the DAQ file on the system with MongoDB on the onsite computer and the Cassandra on the cloud server.

The local computers (such as desktop or laptop computers of system end-users) serve as the computational platform of the system. In the current implementation, the analysis modules are installed on local computers and are separated from the main server so that the main server can guarantee consistent performance as a data management platform. APIs (such as Matlab Engine (for Matlab) and rpy2 (for R)) have been deployed to retrieve the data from the main server to the analysis modules on the local computers.

Additionally, a web server module has been developed based on the HTTP protocol to handle users' request to the main server. The web server module includes functions for converting a user's request into a Cassandra query and returning the query results to the user's device (such as a web browser on the mobile device).

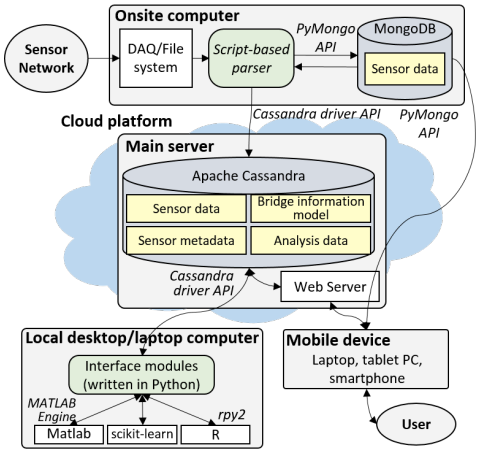


Figure 2. Cloud and NoSQL based data management framework for bridge monitoring

3 DATA SCHEMAS AND DATA MANAGEMENT

3.1 Data schema for bridge information

In order to support a broad spectrum of potential usages of the data for bridge monitoring applications, the data schema needs to include information such as bridge geometry, finite element model parameters, and metadata about the sensors (e.g., type, sensitivities, locations, orientations). The information in the main server needs to be properly organized as much as possible and should adhere to open standards for interoperability purposes. The bridge

information model (BrIM) developed in this work is built upon the OpenBrIM standards, which is an open standard established as an integrated bridge structure model throughout the project lifecycle (Chen, 2013). The bridge data stored in the main server can be mapped to the OpenBrIM schema and then be utilized by applications supporting OpenBrIM.

While the current OpenBrIM standards capture the geometry of a bridge, the standards do not have any entities for describing sensors and lacks many data entities that are necessary to represent an engineering analysis model. The BrIM model thus enriches the OpenBrIM standards with additional user-defined objects and parameters. For example, to include metadata for the sensors, BrIM adopts SensorML which is a widely used standard for sensor description (Open Geospatial Consortium, 2014). Also introduced are the data entities derived from CSI Bridge (a software tool for bridge modelling and analysis) to support an analysis model. Figure 3 shows the BrIM schema developed in this study.

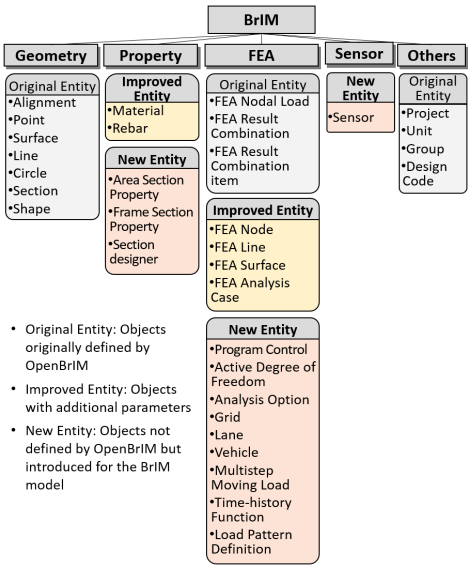


Figure 3. BrIM model for bridge monitoring

Figure 4 shows an example of a BrIM document and the corresponding Cassandra database instance. The BrIM document consists of a hierarchical set of objects and their parameters where an object represents a physical or a conceptual element of a bridge while the parameters are the attributes of each object. In the Cassandra database, each row stores the information of an object, its parameters as well as the object hierarchy by including the information about the parent and the children entities of the object.

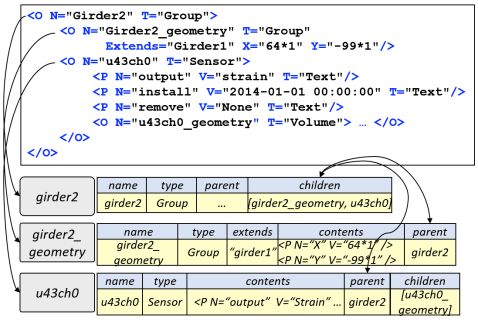
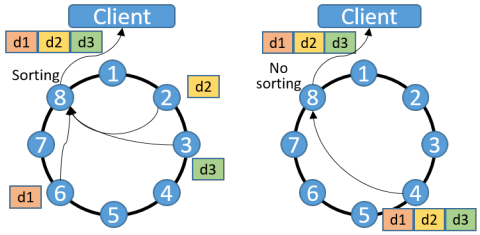


Figure 4. Example of a BrIM document and the corresponding Cassandra database instance

3.2 Sensor data management

A very large portion of the data in a structural monitoring system is the time-series measurement data collected from the sensors. The time-series data need to be stored contiguously in a sequentially sorted order to minimize disk seek time. For distributed storage systems, the partitioning feature of Cassandra may result in distribution of the time-series data to many different computing nodes, causing excessive disk seek time. Figure 5(a) shows an example where the time-series data d:<d1, d2, d3> is distributed over multiple Cassandra database nodes. In this case, a database query not only needs to retrieve the data from all three nodes, but also needs to sort the data according to their timestamps.



(a) Time-series data distributed over multiple nodes

(b) Time-series data that stored in the same node in sorted order

Figure 5. Time-series data stored in a distributed system

To reduce data access time, we employ a strategy for handling time-series data in Cassandra as shown in Figure 6 (McFadin, 2015). First, the partitioning key is defined as the combination of sensor ID and date (such as year and month of the data acquisition), so that the time-series data measured by a single sensor in a specific time span (such as a month) are stored in a single node. Second, the clustering key is defined to be the timestamp when the data is measured, so that the time-series data can be written to the disk in a sorted order. Figure 5(b) shows the example that time-series data d:<d1, d2, d3> are stored in a single node in sorted order.

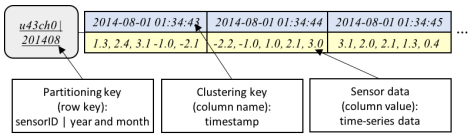


Figure 6. Data schema for time-series data

4 IMPLEMENTATION

For demonstration purpose, we use the bridge model of the Telegraph Road Bridge (TRB) located in Monroe, Michigan and the sensor data collected on that bridge to illustrate the cloud-based cyber infrastructure system (Zhang *et al*. 2016). The sensor data are collected by 14 accelerometers, 40 strain gauges, and 6 thermistors and are acquired between August 2014 and February 2015. For this example, the finite element model is created using CSI Bridge (version 2015).

We first create a virtual machine on the Microsoft Azure cloud platform and set up the Apache Cassandra database along with the defined data schema. For illustration purposes, we use a laptop computer as the onsite computer. The Python scripts on the onsite computer start to execute and to store the sensor data temporarily in MongoDB which is installed on the onsite computer. Furthermore, sensor data is also parsed and sent to the main server which is on Azure. In addition to the sensor data, the finite element model, bridge geometry and sensor metadata are also stored in the Cassandra database.

The data stored in the Cassandra database can be retrieved using the Cassandra Driver API. The API provides a SQL-like query language that supports SELECT, FROM and WHERE statements. For example, Figure 7 shows an example of the query statement for retrieving sensor data and the corresponding query result plotted using the matplotlib (Python library for plotting). Based on the Cassandra Driver API, users can also build query scripts to retrieve the FE model of the bridge structure. Figure 8 shows the retrieved FE model of the TRB and the modal analysis results.

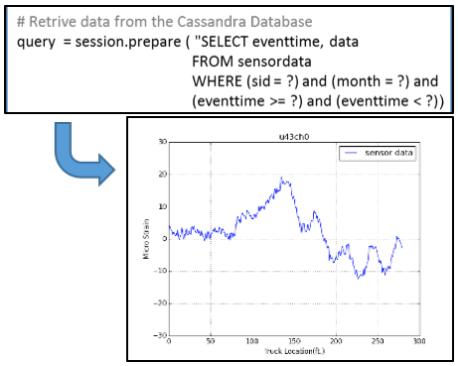


Figure 7. Query statement to retrieve sensor data and corresponding query result plotted using matplotlib

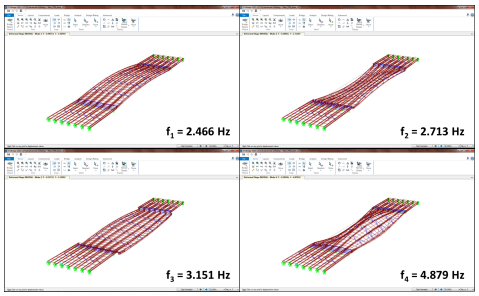


Figure 8. Finite element model retrieved from the main server and modal analysis results

5. SUMMARY AND CONCLUSION

In this study, a cloud-based cyber infrastructure framework for bridge monitoring application is discussed. The main server of the framework utilizes a cloud platform service and a NoSQL database system to enable a cost-effective, scalable and flexible data management platform. APIs and scripts are implemented to allow seamless data flow among the onsite computer, the main server and the tools residing on a local

computer. For time-series sensor data and the complex bridge information, data schemas for SHM are defined. The developed framework and data schema are tested using the data collected from the TRB. While the implementation presented employs the public cloud service platform, the methodology is general and can be deployed in other public or private cloud services and web service environment. Implementation results show that the cyber infrastructure framework can elegantly manage the bridge monitoring data and allow users to easily retrieve data for data analysis.

ACKNOWLEDGEMENTS

This research is supported by a Grant No. 13SCIPA01 from Smart Civil Infrastructure Research Program funded by the Ministry of Land, Infrastructure and Transport (MOLIT) of the Korea government and the Korea Agency for Infrastructure Technology Advancement (KAIA). The research is also partially supported by the US National Science Foundation (NSF), Grant No. ECCS-1446330 to Stanford University and Grant No. CMMI-1362513 and ECCS- 1446521 to the University of Michigan. The authors would like to thank the Michigan Department of Transportation (MDOT) for access to the Telegraph Road Bridge and for offering support during the installation of the wireless monitoring system. 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, KAIA, MDOT or any other organizations and collaborators.

REFERENCES

CHEN S S. 2013. Bridge Data Protocols for Interoperability Local Failure Bridge Data Protocols for Interoperability and Life Cycle Management. [Online article], Retrieved from: http://iug.buildingsmart.org/resources/itm-and-iug-meetings-2013-munich/infra-room/bridge-data-protocols-for-interoperability-and-life-cycle-management (accessed on Mar 2016)

HAN J, HAIHONG E, LE G, DU J. 2011. Survey on NoSQL database[C]. Proceedings of ICPCA 2011, 363-366.

HEWITT E. 2010. Cassandra: the definitive guide. O'Reilly Media, Inc., [Online Book], Retrieved from: http://proquest.safaribooksonline.com/9781449399764 (accessed on Mar 2016)

LAW K H, CHENG J C P, FRUCHTER R, SRIRAM R. 2016. Cloud applications in engineering[M]. In Encyclopedia of Cloud Computing, MURUGESAN S, BOJANOVA I. (eds.), Wiley. (in press).

LAW K H, SMARSLY K, WANG Y. 2014. Sensor data management technologies for infrastructure asset management[M]. In Sensor Technologies for Civil Infrastructures: Applications in Structural Health Monitoring, WANG M L, LYNCH J P, SOHN H. (eds.), Woodhead Publishing, Cambridge, UK, 2(1): 3-32.

MARSTON S, LI Z, BANDYOPADHYAY S, ZHANG J, GHALSASI A. 2011. Cloud computing—The business perspective[J]. Decision support systems, 51(1):176-189.

McFADIN P. 2015. Getting Started with Time Series Data Modeling. [Online article], Retrieved from: https://academy.datastax.com/demos/getting-started-time-series-data-modeling (accessed on Mar 2016)

MCNEILL D K. 2009. Data management and signal processing for structural health monitoring of civil infrastructure systems[M]. In Structural Health Monitoring of Civil Infrastructure Systems, V.M. KARBHARI, F. ANSARI (eds.), CRC Press, Boca Raton, FL, 283-304.

MELL P, GRANCE T. 2011. The NIST definition of cloud computing - Recommendations of the National Institute of Standards and Technology[M]. NIST Special Publication 800-145, Computer Science Division, Information Technology Laboratory, National Institute of Standards and Technology.

OPEN GEOSPATIAL CONSORTIUM. 2014. Sensor Model Language (SensorML) [Online article], Retrieved from: http://www.opengeospatial.org/standards/sensorml (accessed on Mar 2016)

REMDE K. 2011. SaaS, PaaS, and IaaS.. Oh my!. [Online article], Retrieved from: https://blogs.technet.microsoft.com/kevinremde/2011/04/03/saas-paas-and-iaas-oh-my-cloudy-april-part-3/ (accessed on Mar 2016)

ZHANG Y, O'CONNOR S M, VAN DER LINDEN G, PRAKASH A, LYNCH J P. 2016. SenStore: A Scalable Cyberinfrastructure Platform for Implementation of Data-to-Decision Frameworks for Infrastructure Health Management[J]. Journal of Computing in Civil Engineering, 10.1061/(ASCE)CP. 1943-5487.0000560, 04016012.