BIM AND SENSOR BASED DATA MANAGEMENT SYSTEM FOR CONSTRUCTION SAFETY MONITORING

ABSTRACT

Purpose: This research investigates the integration of real-time monitoring of thermal conditions within confined work environments through wireless sensor network (WSN) technology when integrated with Building Information Modelling (BIM). A prototype system entitled Confined Space Monitoring System or CoSMoS (which provides an opportunity to incorporate sensor data for improved visualization through new add-ins to BIM software) was then developed.

Design/ methodology/ approach: An empirical study was undertaken to compare and contrast between the performance (over a time series) of various database models to find a back-end database storage configuration that best suits the needs of CoSMoS.

Findings: Fusing BIM data with information streams derived from wireless sensors challenges traditional approaches to data management. These challenges encountered in the prototype system are reported upon and include issues such as hardware/ software selection and optimisation. Consequently, various database models are explored and tested to find a database storage that best suits the specific needs of this BIM-wireless sensor technology integration.

Originality/ value: This work represents the first tranche of research that seeks to deliver a fully integrated and advanced digital built environment solution for automating the management of health and safety issues on construction sites.

KEYWORDS

Building information modelling, digital built environment, health and safety management, sensor data, data management

INTRODUCTION

According to U.S. Bureau of Labor Statistics (BLS), a total of 796 fatal injuries were recorded in the U.S. construction industry in 2013 from which 14% of fatalities were due to the exposure to hazardous environment and 2% were caused by fire and explosions (BLS, 2013). One major operational area of safety concern on construction sites is workers operating in confined spaces. Occupational Safety and Health Administration (OSHA) has defined a confined space as: *"any space that has limited means for entry or exit and that is primarily not designed for continuous worker occupancy"* (OSHA, 2014). OSHA (2015) estimates that there are 53 worker deaths and injuries, 4,900 lost workday cases and 5,700 non-lost time accident annually. Lack of oxygen in confined spaces is the main cause of accidents faced by construction workers (IACS, 2007). OSHA suggests that in confined spaces oxygen content should be continuously monitored both during and prior to starting work. If the oxygen content in air drops by 19.5% by volume or increases from 22% by volume then entry into a confined space should be avoided (OSHA, 2011). Consequently, effective 'real-time' monitoring of oxygen and temperature levels in a confined space environment is needed to improve worker safety.

The built environment is increasingly becoming digitized and advanced smart city technologies may present new opportunities to more effectively monitor environmental conditions and for this research, hazardous parameters (Pärn *et al.,* 2017). BIM is one of several prominent digital technologies that has gained extensive practitioner and academic interest within the Architectural, Engineering, Construction and Owner-operated (AECO) sector. BIM provides a multi-collaborative platform for sharing both semantic and geometric information throughout the project management team during the asset's entire lifecycle (Vanlande *et al.,* 2008). Similarly, wireless sensor network (WSN) technology has gained prominence within the management of built environmental assets post construction (Li and Becerik-Gerber, 2011). A WSN incorporates a wireless network of spatially distributed autonomous devices or sensors (often called nodes) to provide real time monitoring of physical or environmental conditions (such as temperature, sound and pressure) within built environment systems (Cook and Das, 2004). These nodes are often low cost/ complexity and relay information either to a 'sink' for local usage or a 'gateway' for remote user access via networks connected to the internet (Zhu et al., 2010). Attempts to integrate BIM with different sensing technologies have already been made to specifically improve: environmental monitoring (Piza, *et al.,* 2005); building performance (Guinard *et al.,* 2009; Katranuschkov *et al.,* 2010; Attar *et al.,* 2011; Setayeshgar *et al.,* 2013); facility management (Cahill *et al.,* 2012; Ozturk *et al.,* 2012); and health and safety (H&S) management (Shiau and Chang, 2012; Guven *et al.*, 2012).

These aforementioned studies provide motivation to develop a BIM-Sensor based solution to improve H&S of construction workers particularly when working within confined spaces. The research specifically explored the integration of BIM and WSN with the objective of monitoring environmental conditions within confined spaces - an often ignored area where the physical condition and safety of operators is at high risk. A prototype system entitled Confined Space Monitoring System or CoSMoS was then developed and evaluated (Riaz *et al.,* 2014). This paper reports upon the latest developments of this on-going research work and commensurate improvements to the prototype system which was transformed into a superior solution by addressing the inherent limitations identified during the study. The paper also highlights challenges faced due to collection of real time sensor data and resulting data management issues. To improve the performance of CoSMoS, the work sought to determine the most appropriate choice of database system that fits the application's requirements. Consequently, various database models are explored and tested to find a database storage that best suits the needs of CoSMoS.

COSMOS BACKGROUND: DATA MANAGEMENT CHALLENGE

The prototype CoSMoS (Riaz *et al.,* 2014) was developed as a proof of concept where the development environment comprised of: Crossbow's TolesB mote (Wireless Sensors); Autodesk Revit™ Architecture 2013 (BIM Software); Visual Studio.Net (Software Development Environment); and SQL Server (Database Management System). CoSMoS data aggregation, storage and management framework (refer to Figure 1) highlights the back-end sensor application that was programmed to: read TelosB gateway mote; convert acquired raw sensor values in a

human understandable format; and push the data to a data storage layer. The self-updating BIM model in CoSMoS depends upon a database link between Revit™ external application and a database system. Therefore, overall application data management performance not only dealt with centralized sensor data storage but also included BIM integration and visualization.

<Insert Figure 1 about here>

The data storage layer stores sensor values with 'SensorID', 'Sensor type', and 'Timestamp' in a SQL Server (DBMS) and relational format. Storing each sensing mote's unique identification (ID) is critical to later correlate the acquired sensor values to the BIM model of a confined space. Once huge sensor data is collected from multiple motes, the next challenge posed relates to efficiently managing and retrieving relevant data from a database. The developed prototype system uses an SQL server which is a traditional method to store data, based on the Relational Database Management Systems (RDBMS). Recent research highlights that sharp a growth of data makes traditional RDBMSs inadequate to handle huge volume and heterogeneity of sensor data (Mai *et al.,* 2014). However, there the literature offers no consensus on which type of database performs best for real time sensor applications.

TRADITIONAL RELATIONAL DATABASES IN SENSOR DATA CONTEXT

Existing literature on using traditional relational databases to store dynamically changing sensor data illustrates that adding more capacity to such a database requires greater processing and

storage power on a single machine (Shah *et al.,* 2009). The literature also reveals several key challenges, namely:

- *rigid*: RDBMS provides a well-defined structure which is required for transactional data but is too rigid for sensor data (van der Veen *et al.,* 2012);
- *expensive*: with dynamically changing data it is too expensive to maintain indexes on time (Pungilă *et al.,* 2009, Shah *et al.,* 2009); and
- *less scalable*: relational databases are built for consistency and availability so there is less tolerance for network partitions making it difficult to scale horizontally (van der Veen *et al.,* 2012).

The extant literature also highlights some characteristics of a good sensor database which include the need to be: energy efficient; scalable; self-organizing; and robust against node failures and topology changes (Elnahrawy, 2003). Moreover, the underlying system should benefit from general characteristics of sensor data, such as their spatio temporal nature. Consequently, literature highlights the use of different types of time series software that benefit from the temporal nature of sensor data by having indexes on time. For example, time series DataBlade is used as a potential solution to performance degradation that result from reasoning over a huge volume of data (Shah *et al.,* 2009).

Database Types:

To overcome these challenges posed by RDBMS, several types of databases have been proposed such as NoSQL and Time Series Databases (TSDB) (Mai *et al.,* 2014; Dix, 2016). RDBMS (or SQL databases) store data in database objects called tables. A table is a collection of related data entries and consists of columns and rows (Harrington, 2010). A NoSQL (often interpreted as 'Not only SQL') database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases. For example, column stores, document stores, key-value stores are some of the examples of the data models which come under NoSQL (Nayak *et al.,* 2013). A TSDB on the other hand, is a software system that is optimized for handling time series data, arrays of numbers indexed by time (a date-time or a date-time range) (Fu, 2011).

Literature highlights StoneDB (a NoSQL database) as a possible solution to storing sensor data (Elnahrawy, 2003). Hypertable is also underlined as a potential solution which can overcome the limitations of scalability posed by traditional RDBMS however it is limited by the fact that it can store only strings (Sikkens, 2010). Existing research compares databases such as MySQL, MongoDB, CouchDB and Redis on scalar sensor data. Results show that NoSQL databases, particularly MongoDB, performed better in write intensive systems with the use of bulk inserts, followed by MySQL, CouchDB, and Redis (Mai *et al.,* 2014). Table 1 gives a snapshot of databases reviewed within the extant literature. It also documents the category and advantages of each database.

<Insert Table 1 about here>

DATABASE CHOICES FOR COSMOS:

To improve the performance of CoSMoS and following a careful review of literature, it was decided to evaluate MySQL as a representative of relational databases, MongoDB as a representative of NoSQL, MongoDB with time series concept to test the benefits of having indexes on time and finally Informix, Influx and Kairos databases as representative of the time series group.

MySQL is the world's most popular open source relational database and can cost-effectively help to deliver high performance, scalable database applications (Converse, 2004). Here data is stored in tables consisting of rows and columns. Since the existing prototype system for CoSMoS already uses MySQL, it seemed a viable choice as a representation of traditional RDBMS. MongoDB is an open-source document database that provides high performance, high availability and automatic scaling (Chodorow, 2003). A record in MongoDB is a document, which is a data structure composed of field and value pairs. The extant literature revels that MongoDB has good performance for write intensive applications (Mai *et al.*, 2014) and because CoSMoS is write intensive, MongoDB appeared to be a good choice. Using MongoDB as a Time Series concept, multiple readings are stored into a single document - this further improves the efficiency of the schema as repeating data structures can be isolated (Membrey *et al.*, 2010). MongoDB with a time series concept for storing sensor data has not been discussed extensively

within extant literature and hence, was chosen to be explored further in this research. Hypertable also made a viable candidate however, when millions of rows of integers are required to be stored into string conversion it can get rather expensive in terms of data processing and hence is eliminated from further consideration.

Informix, Influx and Kairos belong to the relatively new Time Series group of databases and are not discussed thoroughly in the literature (which predominantly discusses Time Series data blade in general only) (Ahsan and Vijay, 2014). Hence, in this research the three mentioned Time Series databases were benchmarked for sensor readings. IBM® Informix® Time Series software is a built-in feature of Informix that greatly expands database functionality by adding sophisticated support for managing time series (time-stamped) data (IBM, 2015). Influx is a Time Series database that can perform standard functions such as min, max, sum, count, mean, median and percentiles. Moreover, it uses a SQL-like query language designed for working with time series and analytics (Influx, 2015). Finally, Kairos is a Time Series database that can be run using a back-end store of Apache Cassandra, Apache HBase, or H2 (Bader *et al.,* 2017). It supports millisecond granularity when used with Apache Cassandra and allows to group and aggregate data in flexible ways.

TEST APPROACH

The main goal was to devise a standard means of testing the performance of writes and reads across different databases, with the table structures resembling that being used by the CoSMoS

prototype. To achieve this objective, two core areas of development required greater consideration, namely: i) sensor network setup and deployment; and ii) software and database development environment – each of these areas is now discussed in some further detail.

Sensor network setup and deployment

The sensing network store routing tables and execute routing protocols to route sensor readings from one mote to another. Choosing the correct network architecture for routing sensor readings is important for the network to be scalable and reliable. In doing so, the architecture must keep the sensor network active and working effectively. Sensor network architecture for CoSMoS has been designed for single-hop communications, having single aggregator mote and multiple sensing motes. Sensing motes have been programmed using a specialized programming language NesC on a TinyOS platform. For sensor data aggregation, a TinyOS utility named 'BaseStation' has been run for the operation of aggregator motes. BaseStation works as a bridge between the BIM server serial port and radio network. When it receives a data packet from the serial port, it transmits it on the radio channel; when it receives data packets over the radio, it transmits it to the serial port of BIM server. Because TinyOS has a tool chain for generating and sending packets to a mote over a serial port, using a BaseStation allows PC tools to communicate directly with mote networks (Levis and Gay 2009). To establish communication with a mote over the server's serial port, MOTECOM environment is set to get reading from the aggregator mote running BaseStation utility. Whereas, PORT depends on the platform and where the mote has been plugged in. For Linux/UNIX machines, it is /dev/ttyUSBn for a serial-over-USB port as

mentioned below. A built-in 'SerialForwarder' program is also used to open a data packet source and allow CoSMoS connect to it over a TCP/IP stream in order to use sensor data source.

Export MOTECOM = serial@/dev/ttyUSB0: telosb

java net.tinyos.sf.SerialForwarder -comm serial@/dev/ttyUSB0:telosb

Once motes have been programmed and deployed in a building, motes will initialize and implement operations such as neighbor discovery, data sensing, sensor data processing and sensor data transmissions.

Software and database development environment

For the prototype system, two temperature sensing motes are placed in two different locations of the building under development. Aggregator motes are programmed to aggregate the sensor values received from other sensing motes and forward them along with its own sensed value to a serial port. Aggregator motes are connected to a BIM server using a standard Universal Serial Bus (USB) interface. The back-end sensor application is written in Microsoft Visual Studio 2012 (Software Development Environment) running on a Revit^{TM} server to read the aggregator mote. Sensor application is programmed to: read TelosB gateway mote; convert acquired raw sensor values in a human understandable format; and store sensor values with 'SensorID', 'Sensor type', and 'Timestamp' in Comma Separated Values (CSV) file. Storing each sensing mote's unique identification (ID) is critical to later correlating the acquired sensor values for the confined space

within the BIM model. A data connection is established between Revit™ Architecture software and Database Management System (DBMS) using a Revit™ application programming interface API. The primary objective is to select the most appropriate DBMS for CoSMoS, so that database performance can be optimised. Consequently, various database models are explored and tested to find a database storage that best suits the needs of CoSMoS. For consistency all tests conducted were run on the same machine with hardware and software configurations as reported upon in Table 2.

<Insert Table 2 about here>

Figure 2 shows the CoSMoS database schema for BIM and sensor data storage. Here SensingMote and AggregatorMote are the categories of the basic sensor mote that have functions to acquire data through WSNGateway, save and forward sensor readings to a DataServer. The application for H&S manager retrieves data through Revit™ Server using an external interface that can be viewed (via a graphical user interface (GUI)) on a DesktopClient or MobileApp. This database design provides a detailed view of how BIM and sensor data is linked to give a real-time visualization for monitoring confined spaces. To test the performance of different databases, a homogeneous data schema for execution is maintained for consistency.

<Insert Figure 2 about here>

Subsequent steps involve creating a table with a timestamp, sensor reading and roomID, followed by population of sensor data using multiple inserts. A detailed algorithm for multiple insert/ select is illustrated in Figure 3. It should be noted from the algorithm that the values are averaged for two rounds of Insert/ Select Operations so as to remove any effects of sudden CPU surge due to other background OS processes like paging.

<Insert Figure 3 about here>

1. RESULTS AND ANALYSIS

Table 3 lists the average performances of read and write operations across the evaluated databases. A PERL script was written to measure the time for these operations.

<Insert Table 3 about here>

Testing initially started with the relational database MySQL where the average time measured for read operation was 250 ms whereas, for write it was 100 ms. However, relational databases are renowned to start degrading with increasing data. When data volume is a key consideration and an application is expected to receive several millions to billions of entries per day, NoSQL and time series storage are considered suitable.

The NoSQL data model with MongoDB (as its representative test subject) was then tested. MongoDB is well known for its good performance of read/ write operations because unlike a relational database that stores data in structured rows, MongoDB is free of any rigid structure and stores data in documents. Therefore, MongoDB with high insert rates is preferred to increase query performance and flexibility particularly when handling sensor data and, geometric and semantic BIM information. Table 3 confirms that MongoDB gives better performance for both operations (when compared to other viable alternatives) - read and write times being 2ms and 50ms respectively. However, this performance may degrade due to numerous document scanning when queries attempt to perform aggregation of sensor readings to create sensor data visualizations on BIM software. To improve performance of such aggregated queries, a nested json (java script object notation) approach is tested using MongoDB as a Time Series data store. Furthermore, due to the flexible structure of MongoDB's json document, aggregated average, minimum and maximum values for each document can also be recorded, thus saving redundant aggregation computes. This implementation has already been successfully used by the MongoDB software development team for performance monitoring applications where five billion entries are carried out per day across 10 servers. Table 3 reveals that MongoDB ‘as a time series’ concept takes 127 ms for inserts and 46 ms for selects and is thus much faster than MySQL but slower in comparison to MongoDB ‘as a regular document store’ concept. This can be accepted as the use of a time series concept comes in handy when aggregates such as average or max/min of readings are to be performed. Hence, it is a fair compromise and well suited for the CoSMoS application under consideration.

Evaluation

The evaluation commenced by comparing the relational database MySQL and Informix (used as relational database without the Time Series concept). Table 3 reveals that Informix (as a relational database) outperformed MySQL in read operation but was slower in write operation when averaged over 50 different batch tests. Informix was then tested as a time series database (refer to Table 3) to reveal that the average time for select/ read operation in Informix (when used with the time-series concepts), shoots from 5ms to 7000ms; this is 30 times slower than that of MySQL. In order to better understand this observed slow performance of Informix for read operation, experts on the Informix developer forum and dbforums.com were contacted and based on received suggestions, the code was changed to insert all the records under the same ID column thus allowing all sensor values to fall under the same record. However, this further degraded the performance from 7000ms read time to average time of around 14690ms.

Considering the lack of proper documentation for Informix and its poor results on read operations, other open source Time Series data store alternatives were tested (such as Influx and Kairos). As mentioned earlier, Kairos is a time series database, which is primarily written for Apache Cassandra (as a free and open source distributed database management system) but works on Apache HBase as well (as a free open source, non-relational distributed database written in Java). Apache Cassandra was chosen as the underlying database for testing as it is fastest performer amongst the supported databases for Kairos (Apache HBase is known to be slower than Kairos) (Goldschmidt, 2014). Table 3 reveals that Kairos gave better performance

when compared with other Time Series databases such as Informix and Influx. However, MongoDB (as a time series concept) outperforms all other Time Series data stores.

The performance for read and write operations are also depicted using graphs, presented in Figures 4 and 5 respectively. Figure 4 confirms that MongoDB as a time series concept outperforms all other tested time series databases. For write operations, Informix used as a relational database and MySQL exhibits almost similar performance (refer to Figure 5). However, performance degrades for Informix when used with the time series blade and becomes worse when values are inserted into the same bucket (refer to Figure 5). Moreover, Kairos performed better than Informix (with time series) and Influx. Finally, Figure 6 confirms that MongoDB has the best performance in the time series category.

<Insert Figures 4 and 5 about here>

LIMITATIONS AND FUTURE WORK

To ensure consistency and derive standard results, all the above tests were run on a 64-bit Windows machine. However, performance may vary from machine to machine and better or faster hardware can give improved results. Even though consistent efforts were made to ensure the comprehensiveness of the test results, results for Informix may not reflect the best performance due to lack of comprehensive documentation for the subject. In addition to the tested databases, other time series databases which are gaining popularity (such as Graphite and

OpenTS) should be examined. However, other time series databases require a Linux based test environment and are unavailable for more commonly used platforms like Windows.

Furthermore, environmental monitoring wireless sensing motes have serious resource constraints (Madden *et al.,* 2002; Zhao *et al.,* 2003). In particular, they have limited: communication bandwidth (1-100 Kbps); storage; transmission data rate; processing capabilities; and battery life, which may cause motes to operate for a restricted number of hours. For example, the wireless sensing mote used for this research has an 8MHz processor, 10 KB programming memory and 250 Kbps data rate only. These mote limitations require special network management algorithms for sensor data streams that can explicitly incorporate these resource constraints, for example, incorporating an idle mote to sleep mode for energy as well as data efficient mechanisms.

For this research, the performance of various databases in the presence of dynamically changing sensor data was assessed in terms of identifying a best fit for the CoSMoS application. These tests, were performed locally on a single machine but future work should assess how performance of the databases differ for: a distributed network where data is stored at different locations; platform virtualization for multitasking; and when placed on a cloud based (vis-a-vis local personal computer (PC)) platform to reduce overhead costs and improved accessibility. Since time series data is being used, it would be useful to apply knowledge discovery techniques to find useful patterns to predict anomalies. For CoSMoS, where sensors are constantly monitoring oxygen levels and temperature in confined spaces, detecting patterns in sensor data will help to

collect data that propagates useful information which in turn can generate new knowledge (such as, anomaly detection when sensor reading goes beyond the predicted pattern). Smoothing, curve fitting, linear regression and autocorrelation are several techniques mentioned within extant literature that can be used to identify patterns in time series data (Harrell, 2013). These standard statistical techniques are however limited in terms of their ability to generate intelligence and new knowledge – hence, future developments should seek to implement machine learning (Hana and Golparvar-Fardb, 2016) (including computational intelligent algorithms such as generic algorithms (Kumar and Cheng, 2015) to optimise data driven predictions and decision making (Catbas and Malekzadeh, 2017). Future work for CoSMoS development will explore the inherent value of these scientifically advanced techniques for finding patterns and knowledge in time series data that will invariably augment decision support and so engender higher levels of safety conformance.

CONCLUSION

The digital built environment and the advanced technological solutions that deliver such, afford huge potential to continuously monitor environmental and hazardous conditions that impact upon human health and safety. Realising the true potential of these technological solutions, will require greater far integration and coalescence between them – the primary objective of CoSMoS was not only to show improvement in system performance *per* se but also illustrate the potential of digital integration to solve real-life construction problems.

With regards to performance and depending on the nature of data being stored, the overall performance of a database system will vary across different database models and applications. This research illustrates that the choice of database depends not only upon system architecture but also on its ability to: communicate at very high data exchange rates; insert performance at high data rate; and hold extremely large data amounts. To ensure optimal performance of any sensor based application (including CoSMoS), an optimised database model must be chosen that best suits the data needs of the application. CoSMoS deals with a continuous stream of time series sensor data and therefore required a database that accounts for the nature of this data and scales well with an increase in data volume. Traditional RDBMS or SQL databases do not fulfill this requirement owing to their rigid structure, hence other alternatives like NoSQL and time series databases were explored which allow database scaling to as many servers as required by distributing content among them. After a careful review SQL database (MySQL) was tested against NoSQL (MongoDB) and time series databases (Informix, Influx, Kairos, MongoDB with time series concept). Tests were designed to measure performance of read and write operations across databases populated with CoSMoS sensor readings. Results revealed that MongoDB performed the best both in terms of read and write operations amongst all the databases. However, given the temporal nature of CoSMoS data and taking into consideration the case where aggregation is required, MongoDB used as a time series store was considered as the best choice for the application.

Future work is however required to include: developing of a more sophisticated pattern analysis criteria to support decision support and generate new knowledge from the data and information accrued within CoSMoS; and extending the capabilities of CoSMoS to cover a wider range of environmental, health and safety issues. Such developments will contribute towards creating a fully autonomous and intelligent digital built environment.

ACKNOWLEDGEMENTS

The authors wish to thank Srilakshmi Chintala and Muhammad Arslan for their various contributions to this research work.

REFERENCES

Ahsan, K. and Vijay, P. (2014) Temporal Databases: Information Systems. Bloomington: Booktango.

Attar R., Hailemariam E., Breslav S., Khan A., and Kurtenbach G. (2011) Sensor-enabled Cubicles for Occupant-centric Capture of Building Performance Data. ASHRAE Annual Conference pp. 1-8. Available via: http://www.autodeskresearch.org/pdf/6774_Final.pdf (Accessed: January 2, 2016).

Bader, A., Kopp, O. and Falkenthal, M. (2017) Survey and Comparison of Open Source Time Series Databases. Available via: ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/INPROC-2017-06/INPROC-2017-06.pdf (Accessed: February 25, 2017).

Bureau of Labor Statistics (BLS) (2013) Census of Fatal Occupational Injuries (CFOI) - Current and Revised Data. Available via: http://stats.bls.gov/iif/oshcfoi1.htm, (Accessed: December 5, 2015).

Cahill B., Menzel, K. and Flynn D. (2012) BIM as a Centre Piece for Optimised Building Operation. Available via: http://zuse.ucc.ie/~brian/publications/2012/BIM_centre_piece_OBO_ECPPM2012.pdf (Accessed December 21, 2015).

Catbas, F.N. and Malekzadeh, M. (2017) A Machine Learning-Based Algorithm for Processing Massive Data Collected from the Mechanical Components of Movable Bridges, *Automation in Construction*, Vol 72, pp. 269–278.

Chodorow K. (2013) MongoDB: The Definitive Guide, Powerful and Scalable Data Storage. 2nd Edition. Sebastopol: O'Reilly Media, Inc.

Converse T., Park J. and Morgan C. (2004) PHP5 and MySQL bible, Vol. 147, London: John Wiley & Sons.

Cook D. and Das S. (2004) Smart Environments: Technology, Protocols and Applications, London: John Wiley & Sons.

Diao Y., Ganesan D., Mathur G. and Shenoy P.J. (2007) Rethinking Data Management for Storage-centric Sensor Networks, Conference on Innovative Data Systems Research (CIDR), Vol. 7, pp. 22-31. Available via: http://web.mit.edu/tibbetts/Public/CIDR_2007_Proceedings/papers/cidr07p03.pdf (Accessed: July 5, 2016).

Dix P. (2016) Why Time-Series Matters For Metrics Real-Time and Sensor Data. Available via: https://www.influxdata.com/resources/why-time-series-matters-for-metrics-real-time-and-sensor-data/ (Accessed: November 2, 2016).

Elnahrawy E. (2003) Research Directions in Sensor Data Streams: Solutions and Challenges, Available via: http://eolo.cps.unizar.es/docencia/doctorado/Articulos/DataStreams/Research%20Directions%20in%20Sensor%20Data%20Streams.%20Solutions%20and%20Challenges.pdf (Accessed: January 2, 2016).

Fu T.C. (2011) A Review on Time Series Data Mining. *Engineering Applications of Artificial Intelligence*, Vol. 24 No.1, pp.164-181.

Golab L. and Özsu M.T. (2003) Issues in Data Stream Management, ACM Sigmod Record, Vol. 32, No.2, pp.5-14.

Goldschmidt T., Jansen A., Koziolek H., Doppelhamer J. and Breivold H.P.(2014) Scalability and Robustness of Time-Series Databases for Cloud-Native Monitoring of Industrial Processes, IEEE 7th International Conference on Cloud Computing (CLOUD), pp. 602-609. Available via: http://www.koziolek.de/docs/Goldschmidt2014-IEEE-CLOUD-preprint.pdf (Accessed: July 5, 2016).

Guinard A., McGibney A. and Pesch D. (2009) A Wireless Sensor Network Design Tool to Support Building Energy Management. 1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp. 25-30.

Guven G., Ergen E., Erberik M., Kurc O. and Birgönül M. (2012) Providing Guidance for Evacuation During an Emergency Based on a Real-time Damage and Vulnerability Assessment of Facilities, *Computing in Civil Engineering* pp. 586-593. Available via: https://www.e-education.psu.edu/geog588/sites/www.e-education.psu.edu.geog588/files/file/Guven_etal_2012.pdf (Accessed: August 13, 2016).

Hana, K.K. and Golparvar-Fardb, M. (2016) Appearance-based Material Classification for Monitoring of Operation-level Construction Progress Using 4D BIM and Site Photologs, *Automation in Construction*, Vol. 53, pp. 44–57.

Harrell F.E. (2013) Regression Modelling Strategies: with Applications to Linear Models, Logistic Regression, and Survival Analysis. New York: Springer Science & Business Media.

Harrington J. L. (2010) Introduction to SQL, Morgan Kaufmann Series in Data Management Systems, Boston, USA: Morgan Kaufmann.

IACS, Confined Space Safe Practice (rev.2), Available via: http://www.iacs.org.uk/document/public/Publications/Guidelines_and_recommendations/PDF/REC_72_pdf212.pdf/ 2007, (Accessed: December, 5, 2015).

IBM. (2015) IBM Informix, Available via: http://www-01.ibm.com/software/data/informix/timeseries/ (Accessed: December 5, 2015).

Influx. (2015) The Influx Data Platform, Available via: https://influxdb.com, (Accessed: December 5, 2015).

Katranuschkov P., Weise M., Windisch R., Fuchs S. and Scherer R. J. (2010) BIM-based Generation of Multi-model Views, Available via: http://www.hesmos.eu/plaintext/downloads/paper_114_final.pdf (Accessed: March 3, 2016).

Kumar, S.S. and Cheng, J.C.P. (2015) A BIM-Based Automated Site Layout Planning Framework for Congested Construction Sites, *Automation in Construction*, Vol. 59, pp. 24–37.

Levis P. and Gay D. (2009) TinyOS Programming, Cambridge, UK: Cambridge University Press.

Li, N. and Becerik-Gerber, B. (2011) Performance-based evaluation of RFID-based indoor location sensing solutions for the built environment. *Advanced Engineering Informatics*. Vol 25, Issue 3, pp. 535–546

Madden S., Franklin M. J. and Hellerstein J. M. (2002) TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks, Proceedings of 5th Annual Symposium on operating Systems Design

and Implementation (OSDI). Available via: http://db.lcs.mit.edu/madden/html/madden_tag.pdf (Accessed: March 03, 2016).

Mai P.T.A., Nurminen J.K. and Di Francesco M. (2014) Cloud Databases for Internet-of-Things Data, IEEE International Conference on Cyber, Physical and Social Computing 2014 (CPSCom 2014) and IEEE International Conference on Green Computing and Communications 2014 (GreenCom 2014), IEEE, Taipei, pp. 117-124. Available via: https://pdfs.semanticscholar.org/dc28/d92ac4e2dd3c9a77abfce80594a424797c1f.pdf. (Accessed: January 29, 2016).

Membrey P., Plugge E. and Hawkins D. (2010) The Definitive Guide to MongoDB: the noSQL Database for Cloud and Desktop Computing, New York: Apress.

Nayak A., Poriya A. and Poojary D. (2013) Type of NOSQL Databases and its Comparison with Relational Databases, *International Journal of Applied Information Systems*, Vol. 5 No. 4, pp. 16-19.

OSHA (2014) What are Confined Spaces? Available via: https://www.osha.gov/SLTC/confinedspaces/, (Accessed: December 3, 2016).

Ozturk Z., Arayici Y., and Coates S. P. (2012) Post Occupancy Evaluation (POE) in Residential Buildings Utilizing BIM and Sensing Devices: Salford Energy House Example, Available via: http://usir.salford.ac.uk/20697/1/105_Ozturk.pdf (Accessed: March 13, 2016).

Pärn, E.A., Edwards, D.J. and Sing. M.C.P. (2017) The Building Information Modelling Trajectory in Facilities Management: A Review, *Automation in Construction*, Vol. 75, pp 45-55.

Piza H.I., Ramos F.F. and Zuniga F. (2005) Virtual Sensors for Dynamic Virtual Environments. 1st IEEE Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing, pp. 177-180.

Pungilă C., Fortiş T.F. and Aritoni O. (2009) Benchmarking Database Systems for the Requirements of Sensor Readings, IETE Technical Review, Vol. 26, No. 5, pp.342-349.

Riaz Z., Arslan M., Kiani A.K. and Azhar S. (2014) CoSMoS: A BIM and Wireless Sensor Based Integrated Solution for Worker Safety in Confined Spaces, *Automation in Construction*, Vol. 45, pp. 96-106.

Setayeshgar S., Hammad A., Vahdatikhaki F., and Zhang C. (2013) Real Time Safety Risk Analysis of Construction Projects using BIM and RTLS, Available via: http://www.iaarc.org/publications/fulltext/isarc2013Paper224.pdf (Accessed: December 21, 2015).

Shah N., Tsai C.F. and Chao K.M. (2009) Monitoring Appliances Sensor Data in Home Environment: Issues and Challenges, Proceeding of the IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria, pp. 439-444. Available via: https://pdfs.semanticscholar.org/5714/e46af2e0090d57cd5a5e10d7543e97c39a79.pdf (Accessed: July 21, 2016).

Shiau Y. C. and Chang C. T. (2012) Establishment of Fire Control Management System in Building Information Modelling Environment, Available via: http://onlinepresent.org/proceedings/vol5_2012/11.pdf (Accessed: October 21, 2015)

Sikkens B. (2010). The Storage and Retrieval of Sensor Data and its Annotations, Available via: http://essay.utwente.nl/59484/1/scriptie_B_Sikkens.pdf (Accessed: January 4, 2016).

United States Department of Labor, Occupational Safety and Health Standards, Available via: http://www.osha.gov/pls/oshaweb/owadisp.show_document?p_table=standards&p_id=9797/ 2013 (Accessed December 5, 2015).

United States Department of Labor, OSHA-Regulations (Standards-29CFR), Available via: http://www.osha.gov/ (Accessed: November 25, 2015).

Van der Veen J.S. van der Waaij B. and Meijer R.J. (2012) Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual, Proceeding of the 5th International Conference on Cloud Computing (CLOUD), IEEE, Hawaii, USA, pp.431-438. Available via: https://www.ceid.upatras.gr/webpages/faculty/vasilis/Courses/SpatialTemporalDM/Papers/SQLorNoSQL2012.pdf (Accessed: September 22, 2015)

Vanlande R., Nicolle C., and Cruz C. (2008) IFC and Building Lifecycle Management. *Automation in Construction*, Vol.18, No.1, pp.70-78.

Zhao J., Govindan R. and Estrin D. (2003) Computing Aggregates for Monitoring WSN, 1st IEEE International Workshop on Sensor Network Protocols and Applications. Available via: http://www.cs.cornell.edu/~destrin/resources/conferences/2003may-Zhao-Estrin-Computing.pdf (Accessed: October 26, 2016)

Zhu Q., Wang R, Chen Q, Liu Y., and Qin W. (2010) IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. IEEE/IFIP 8th International Conference on Embedded and

Ubiquitous Computing (EUC), 11-13 Dec. 2010. Available via: http://ieeexplore.ieee.org/abstract/document/5703542/ (Accessed: September 20, 2015)

Figure 1 – CoSMoS Data Framework

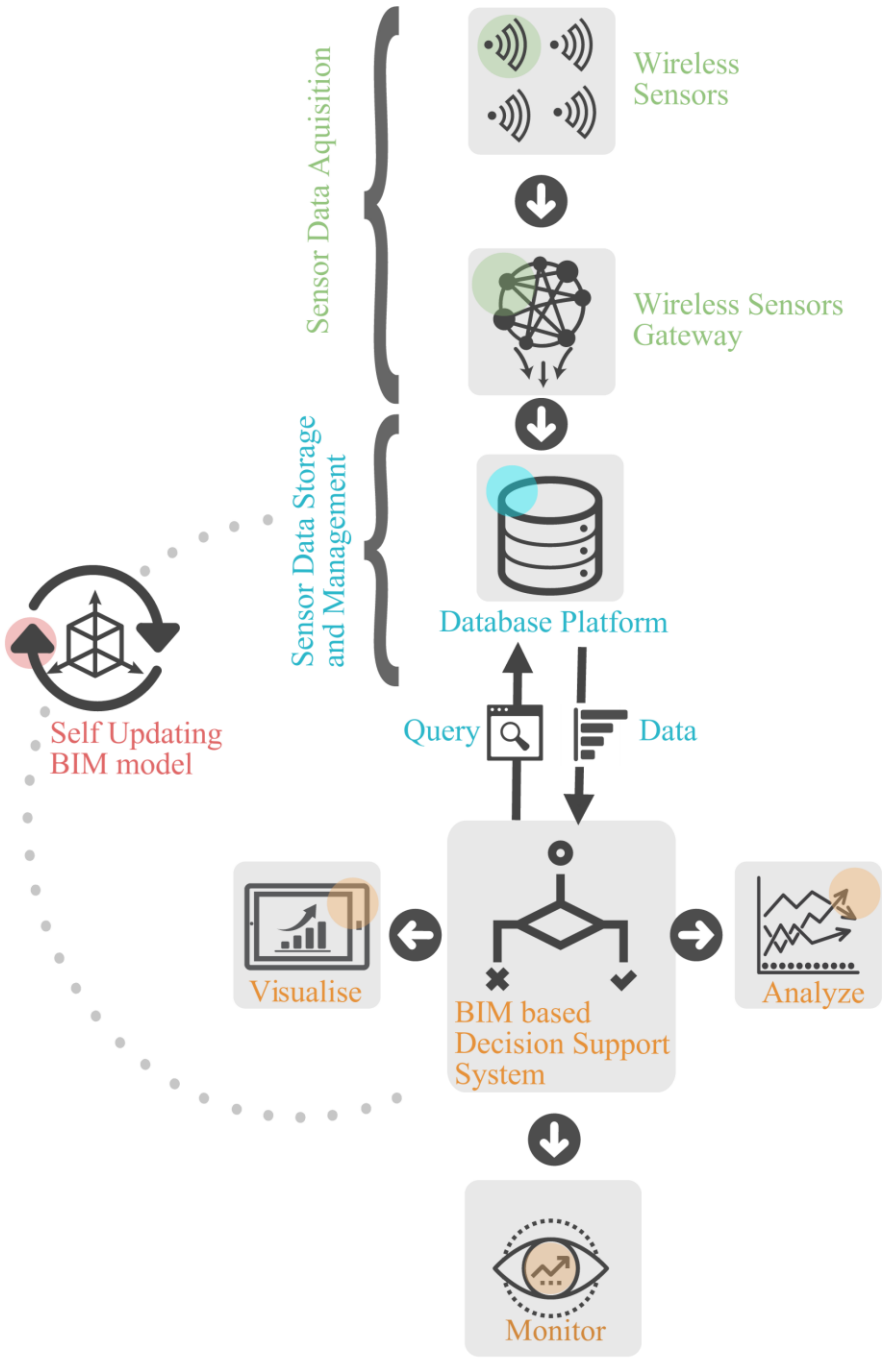

Table 1 - Snapshot of the Various Databases Reviewed in Literature

1.1.1 Database	1.1.2 Type	1.1.3 Advantages	1.1.4 Comments	1.1.5 Reference
1.1.6 MySQL	1.1.7 SQL	1.1.8 Consistency and availability.	1.1.9 Does not scale well	1.1.10 (van der Veen *et al.*, 2012)
1.1.11 Oracle Berkeley DB	1.1.12 SQL	Enables prediction of patterns consistency and availability.	1.1.13 Open source embedded database engine, offered as a library that directly links into applications.	1.1.14 Pungilă *et al.*, 2009
1.1.15 Hypertable	1.1.16 NoSQL	1.1.17 Supports applications requiring maximum performance, scalability, reliability resistant to component failures.	1.1.18 Distributed data storage system 1.1.19 stores only strings.	1.1.20 (Sikkens, 2010)
1.1.21 Cassandra	1.1.22 NoSQL	1.1.23 Can cope with very large amounts of data spread out across many commodity servers.	1.1.24 Structured key-value store using the mechanism of eventual consistency 1.1.25 relaxes either consistency or availability.	1.1.26 (Mai *et al.*, 2014)
1.1.27 MongoDB	1.1.28 NoSQL	1.1.29 Includes a powerful query language that allows for regular expressions and Javascript functions to be passed in as checks for matching keys and values.	Provides a key-value store that manages collections of BSON (binary JSON) document relaxes either consistency or availability.	1.1.30 (Mai *et al.*, 2014)
1.1.31 PostgreSQL	1.1.32 SQL	1.1.33 ACID (atomicity, consistency, isolation, durability) compliant and is fully transactional.	1.1.34 Traditional open source SQL database 1.1.35 does not scale well.	1.1.36 (Pungilă *et al.*, 2009)
1.1.37 Informix	1.1.38 Time Series	1.1.39 Consolidates and organizes time-stamped data much more efficiently than traditional, relational databases.	1.1.40 IBM® Informix® TimeSeries software is a built-in feature of Informix that greatly expands database functionality by adding sophisticated support for managing time series (time-stamped) data.	1.1.41 (Pungilă *et al.*, 2009, IBM, 2015)

1.1.42 Influx	1.1.43 Time Series	1.1.44 Has no external dependencies.	1.1.45 Designed to track data from tens of thousands of sensors rates of once a second or more. Using the Influx JavaScript library one can build custom sensor analytics.	1.1.46 (Influx, 2015)
1.1.47 Kairos	1.1.48 Time Series	1.1.49 KairosDB supports millisecond granularity when used with Cassandra.	1.1.50 KairosDB can be run using a back-end store of Cassandra, HBase, or H2.	1.1.51 (Goldschmidt, 2014)

Table 2 - Hardware and Software Configurations of Test Machine

Criteria	**Specification**
CPU Type	AMD A6-6310 Quad-Core APU processor 2.40GHz
System memory	8GB RAM
Hard Drive	1 TB
Operating System	Windows 7, 64 bit, TinyOS 2.1.2 packages
BIM software	Autodesk Revit™ Architecture 2013

Figure 2 - CoSMoS Database Schema

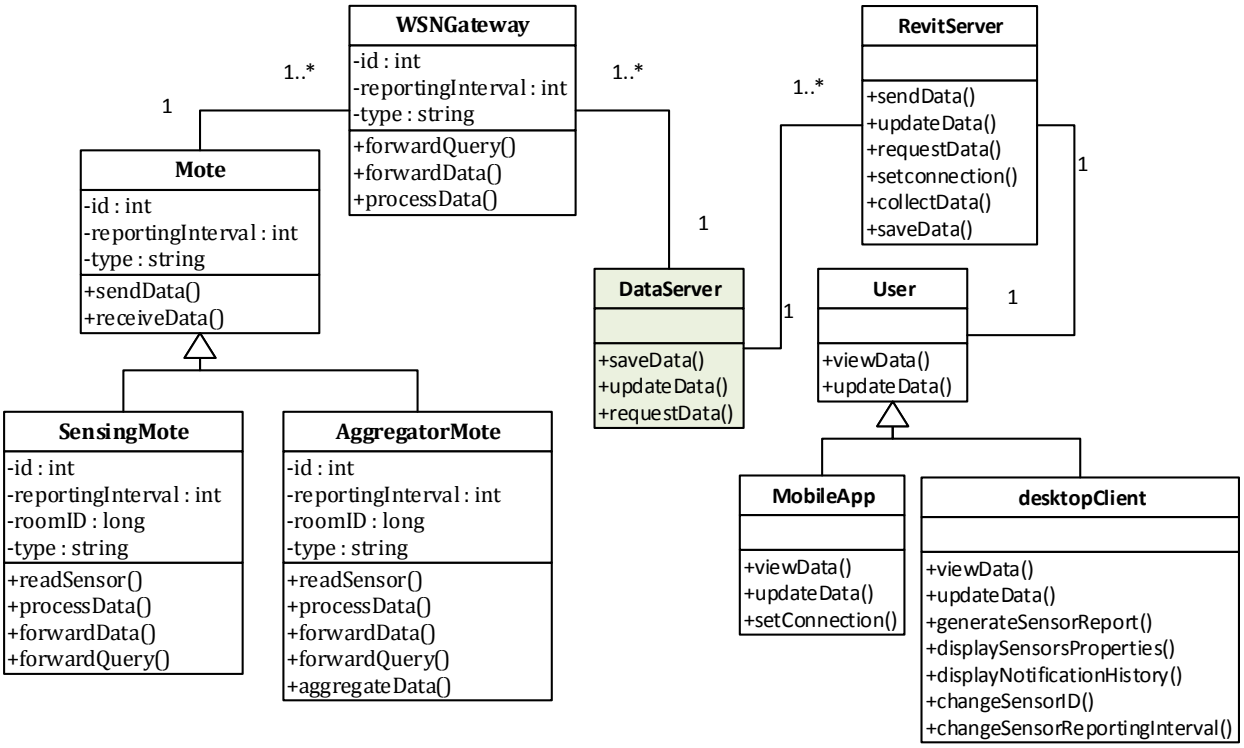

Figure 3 - Algorithm for Benchmarking Performance of Database for Insert/Select Operations

```
Establish connection to database
While batchSize <= 500
        Start timer
                totalTime = 0;
        For each batch size measure time over 2 rounds
                        startTimeForBatch=currentSystemTime;
                        Perform Insert/Select
                        endTimeForBatch = currentSystemTime -
                        startTimeForBatch;
                        totalTime = totalTime +
                        endTimeForBatch;
        Calculate the average time taken for the 2 rounds
                totalTime = totalTime/2;
        Record batchSize and time taken
        Increment batchSize by 10
                batchSize = batchSize+10
Calculate average time across all batches
```

Table 3 - Average Performance of Reads and Writes Across Multiple Databases.

Database(Test Info)	1.1.52 **Select**	1.1.53 **Insert**
1.1.54 MySQL	1.1.55 250 ms	1.1.56 100 ms
1.1.57 MongoDB as Document / key-value store	1.1.58 2 ms	1.1.59 50 ms
1.1.60 MongoDB as Time Series Store	1.1.61 46 ms	1.1.62 127 ms
1.1.63 Informix	1.1.64 5 ms	1.1.65 250 ms
1.1.66 Informix with Time Series data blade	1.1.67 7000 ms	1.1.68 250 ms
1.1.69 Influx	1.1.70 ~100,000 ms	1.1.71 750 ms
1.1.72 Kairos	1.1.73 385 ms	1.1.74 285 ms

Figure 4 - Graph Depicting Performance of Reads Across Multiple DB's

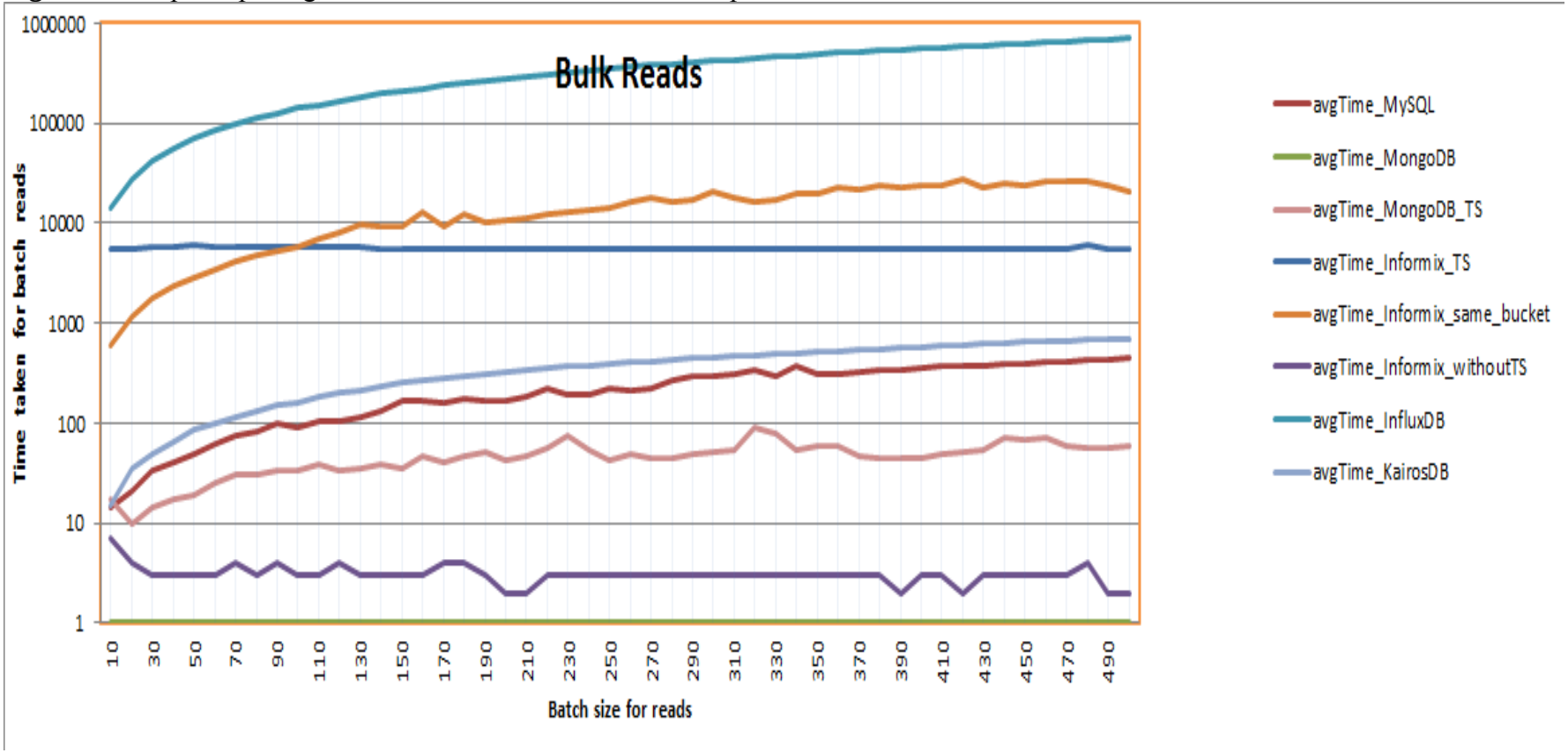

Figure 5 - Graph Depicting Performance of Writes Across Multiple DB's