Web user interface based on OGC standards for sensor cloud using big data

Vijayasherly Velayutham* and
Srimathi Chandrasekaran

School of Computer Science and Engineering,
VIT University,
Vellore, India
Email: vvijayasherly@vit.ac.in
Email: csrimathi@vit.ac.in
*Corresponding author

Subaji Mohan

Center for International and Industry Interaction,
VIT University,
Vellore, India
Email: msubaji@vit.ac.in

Abstract: Sensor cloud is the combination of wireless sensor networks and Cloud computing which performs real-time data acquisition and processing in a distributed environment. The infrastructure enables collection, processing, sharing, visualisation, archival and searching of large amounts of sensor data. It makes it possible for sharing computing resources and data, on an unprecedented scale, among an infinite number of geographically distributed groups. The paper provides an implementation of efficient web based technology for processing large, heterogeneous set of data based on Open Geospatial Consortium Standards. The proposed interface uses the sensor cloud middleware and wireless sensors for distributed applications that process large amounts of data. Sensor cloud middleware in the proposed work uses Open Stack and Hadoop framework along with utilities like generic modelling environment, HBase and Zabbix. The user interface with features such as viewing sensor data, scheduling sensors for data collection and alerting users have also been discussed.

Keywords: sensor cloud; middleware; Open Stack; Hadoop; big data; sensor web; user interface.

Reference to this paper should be made as follows: Velayutham, V., Chandrasekaran, S. and Mohan, S. (2018) 'Web user interface based on OGC standards for sensor cloud using big data', *Int. J. Communication Networks and Distributed Systems*, Vol. 20, No. 4, pp.389–412.

Biographical notes: Vijayasherly Velayutham is currently working as an Assistant Professor (Selection Grade) in the School of Computer Science and Engineering, VIT, Vellore. She completed her MTech in Computer Science and Engineering at the VIT University. Her research interests include distributed systems, cloud computing and internet programming.

Srimathi Chandrasekaran is currently working as an Associate Professor in School of Computer Science and Engineering, VIT, Vellore. She received her PhD in the field of fault tolerance in computational grid. Her research interests include cloud computing, big data analytics and agent-based computing. She has also completed a funding project sanctioned by a government agency in India.

Subaji Mohan is currently working as a Professor and Deputy Director, Center for International and Industry Interaction, VIT University, Vellore. He received his PhD at the Kookmin University, South Korea on 'MDA Based Workflow Meta-Modelling of CRM System with Integrated Business Process'. His research interests include data mining, business process management using big data.

1 Introduction

A sensor cloud provides a platform for large-scale sensor data with features such as collection, processing, sharing, accessing, storing and visualisation. Sensor cloud setup comprises of sensor nodes for environmental monitoring used for collecting temperature, light, and humidity. Managing the sensor cloud requires an efficient middleware capable of maintaining the efficiency of the sensor cloud. In general, a prominent sensor cloud middleware that fulfils the needs of the sensor cloud is needed. Apache's Hadoop framework provides data collection and storage using Hadoop distributed file system (HDFS), parallel processing, data sharing among cluster nodes and visualisation of data and results. Hadoop big data applications are run on Open Stack (http://www.openstack.org/software/start/). Multiple Hadoop cluster environments with differences in software and hardware stack may be required in business units leading to operational complexities. This operational challenge is alleviated by Open Stack.

The architecture for the sensor cloud middleware needs to be implemented as per the requirements. A proper modelling tool is required to view the workflow of the sensor cloud framework beforehand of the middleware implementation. Generic modelling environment (GME) serves as a suitable modelling tool to generate workflow of the sensor cloud framework. Monitoring the sensor cloud nodes for issues such as hardware failures, power failures, data corruption can pave way to rectifying these errors before it becomes a serious issue. Zabbix plays the part of monitoring the cluster nodes in case of issues.

Along with performance analysis, tuning and monitoring, the sensor cloud middleware requires an easy-to-use interface for the users. This can be achieved by providing the users with Java web services based on open geospatial consortium (OGC) standards. Accessing sensor data through web services not only provides faster response, but also hides the underlying layers, network communication details, and wireless sensor hardware.

OGCs sensor web enablement (SWE) provides many specifications for web services, of which sensor observation service (SOS) and sensor planning service (SPS) with web notification service (WNS) and sensor alert service (SAS) were used for connecting the user interface with the middleware.

The user interface provides features such as accessing the historical sensor data, scheduling a sensor node for data fetching and alerting users for a specific condition of a sensor node. In the proposed work, SOS standard serves as a base for observing historical sensor data by the users, SPS is used for tasking the wireless sensors, WNS is used for sending notifications to the users, and SAS for creating triggers for sensor data based on user request. Alerting the user is done by combining the functions of both SAS and WNS standards.

1.1 Contributions

The contributions of the proposed sensor cloud middleware framework are:

1 Monitoring and collecting real-time environmental data from deployed wireless sensors and modelling of the sensor cloud framework workflow.

2 Implementing the Hadoop framework on Open Stack as sensor cloud middleware to support parallel processing and storing big data.

3 Designing a web user interface using Java web services based on OGC standards. This paper discusses the sensor cloud framework architecture and its components in Section 3. Discussion about the back-end and front-end utilities of the sensor cloud framework is done in detail under Section 4. A brief discussion of the proposed methodology is done in Section 5. Section 6 concludes the paper.

2 Related work

Kyzirakos et al. (2014) proposed a service with a web application for real-time wildfire monitoring using the geospatial data and satellite images. The architecture proposed consisted of stRDF, a modelling extension of World Wide Web Consortium (W3C) standard resource description framework (RDF) which represented real-time changes in geospatial data and stSPARQL, an extension for W3Cs SPARQL (SPARQL protocol and RDF query language) query language which was used to query and update stRDF data. The stRDF used OG Cs well-known text (WKT) and Geography Markup Language (GML) to represent dynamically altering geospatial data. The proposed work uses the environmental data such as temperature and humidity instead of geospatial data, and implements OGC standards such as SOS, SPS, WNS and SAS instead of geospatial related OGC standards.

Misra et al. (2015) proposed a QoS-aware sensor allocation algorithm (Q-SAA) that uses an auction-based mechanism to find the optimal solution for allocation of a subset of available sensors to achieve efficient target tracking in a sensor cloud environment. In the sensor cloud architecture, the problem of resource allocation in a target tracking scenario has been addressed.

Madria et al. (2013) proposed a cloud of virtual sensors (VSs) built on top of physical sensors, and it provisions VSs to the user on the basis of applications' demands. It enables better sensor management capability. The users can use and control their view of wireless sensor networks (WSNs). Further data captured by WSNs can be shared among multiple users, which reduces the overall cost of data collection for both the system and user. Although the work is web enabled it does not employ OGC standards.

Chatterjee et al. (2015a) proposed a novel dynamic and optimal pricing scheme for provisioning Sensors-as-a-Service (Se-aaS) within the sensor cloud infrastructure. The work presents a new cloud pricing model for heterogeneous service oriented architecture of Se-aaS that comprises pricing attributed to hardware (pH) and pricing attributed to infrastructure (pI). The problem of pricing the physical sensor nodes substantiating variable demand and utility of the end-users is addressed by pH. The pricing incurred due to the virtualisation of resources is addressed by pI.

Vitolo et al. (2015) proposed software named environmental virtual observatory pilot (EVOp) which provided services based on representational state transfer (REST) architecture. OGCs web processing service (WPS) was used to deploy environmental models and to communicate between client and server using HTTP GET requests and eXtensible Markup Language (XML) responses. EVOp provided a web interface for flood monitoring using the geospatial data collected in real-time. The proposed work varies from discussed work in terms of OGC standards used (proposed work primarily implements SOS and SPS standards) and the data type collected (proposed work collects and stores non-spatial environmental data).

Chatterjee et al. (2014) proposed techniques for evaluating rescue teams in major natural disasters. The evacuation system assesses the details about the rescue troops to be deployed, determines rescue routes and evacuation strategies based on the criticality of damage in the affected regions. The cloud servers feed the data collected from the affected area into the federated cloud environment. A provision for proactive pre-deployment of rescue workers in zones where the disaster and its consequent effects are predicted to spread is being analysed. This creates an opportunity for evacuating people from the suspected regions.

Jirka et al. (2013) proposed web service client based on SWEs SOS for alert notification subscriptions. The sensor web client provides a feature to analyse and compare data on different time series and also a risk monitoring facility to send notifications during critical situations in real-time.

Misra and Chatterjee (2014) proposed a cloud-assisted WBAN-based architecture for aggregating data from LDPUs of the patients and an algorithm, named OCA, for channelising data through dynamic gateway allocation. The health-criticality is a metric of the transmitted packet and this enables the medical teams to develop an elementary idea of each patient from the data packet itself. Cloud gateways are used to channelise data from the mobile monitoring nodes.

Chatterjee et al. (2015c) proposed a novel algorithm named Metric of Misbehaviour (MoM) based of theory of social choice. The work focusses on the quantification of node misbehaviour in WSNs. According to the algorithm, the misbehaving nodes are the voting alternatives and the normally behaving nodes are the voters. Based on majority ranking of social choice, MoM is obtained for every alternative in a fair manner. Hence a metric has been is derived to judge the severity of turbulence that a misbehaving node might cause to the entire WSN. Moulik et al. (2015) proposed a framework, Smart-Evac that solves the issues faced by emergency management systems during an evacuation after a severe disaster. The system aims at disaster risk reduction by employing a ubiquitous cloud-based health monitoring system that can handle big data via WBANs. Thus injured victims get immediate assistance from a cloud-based ambulatory medical service, which can further decide if a victim has to be admitted in hospital or not.

Bisoi et al. (2013) developed iSENSE, a WSN testbed with a simple web interface which included features such as a query engine for viewing historical sensor data and

download facility to gather the queried sensor data for future reference. Wireless sensors were deployed to collect environmental parameters such as temperature, humidity, and light. The collected sensor data was stored in MySQL database and processed using PHP. The proposed work deals with HBase, a big data utility, for storing environmental data and provides access to these data through Java web services.

Chatterjee et al. (2014) proposed a dynamic mapping algorithm (SDMA) based on the theory of social choice for tracking multiple targets using the sensor cloud infrastructure. Privacy and correctness of sensed information about the targets is maintained by correctly mapping sensors to targets, in presence of overlapping coverage. The proposed algorithm, S-DMA, ensures the best possible allocation of sensors to targets.

Chatterjee and Misra (2015) proposed a work on optimal formation of VSs within a sensor cloud infrastructure. The proposed algorithms focusses on efficient virtualisation of the physical sensor nodes and optimal composition of VSs – within the same geographic region bearing homogeneous sensing hardware (CoV-I) and spanning across multiple regions with heterogeneous sensing hardware (CoV-II). It has been proved that CoVs enhance the resource utilisation to a great extent, compared to the existing techniques of maximal allocation of the physical sensor nodes.

Shelestov et al. (2013) implemented a geoportal for agricultural monitoring based on OGC standards which provided interoperability with other monitoring and decision-making support systems. Quantum GIS (QGIS) was used to handle geospatial data in the form of separate layers of the geoportal monitoring system which made accessing the geospatial data quicker.

Tian and Huang (2012) proposed a geospatial web service based on OGC standards and a query mechanism for semantic reasoning and simple spatial query. OGC standards such as Web Map Service (WMS), Web Feature Service (WFS) and Web Coverage Service (WCS) were used, whereas in the proposed work standards such as SOS, SPS, SAS and WNS were used.

Kumar et al. (2015) proposed at a novel approach to analyse the coalition game among the connected vehicles in IoV environment that includes objects such as vehicles, sensors and actuators, with respect to cooperative and non-cooperative behaviour of the players.

Misra et al. (2014a) proposed a theoretical model of virtualisation for a sensor cloud environment with an extensive discussion on mapping an application to its physical resources and the procedure for virtualisation of the resources. The theoretical modelling of sensor cloud can be used for building models in order to solve different problems to be encountered in using sensor cloud.

Chatterjee et al. (2015b) proposed a work on dynamic scheduling of data centres (DCs) in a sensor cloud infrastructure, given a particular application, and a set of geographically scattered DCs considering the QoS of the application. The QoS has been quantified by migration cost within the DCs, the delivery cost to the application from the scheduled DC, and the overall service delay of provisioning Se-aaS.

Ding et al. (2013) proposed a novel mechanism, known as sea-cloud-based data management (SeaCloudDM) that provided satisfactory performances in managing and querying massive sensor sampling data. The mechanism reduced and handled the data need to be managed in the cloud environment and heterogeneous sensor sampling data in a uniformed manner respectively. This method is purely based on data storage and

processing mechanism but does not provide a proper user interface to access these data as done in the proposed work.

Chatterjee et al. (2016b) proposed an algorithm – QoS based automated selection of cloud service provider (CSP) (QASeC) that enables selection among multiple CSPs through a process of judgment even without possessing a definite perception or awareness of the cloud services. CSPs are parameterised by few QoS parameters and based on the parameters, a QoS utility metric is formulated for every gateway of each CSP. The proposed algorithm achieves a quantification of the QoS that can be provisioned by every CSP thereby obtaining the CSP with the highest QoS for a naive end-user in an IoT scenario.

Indriya testbed, the sensor network testbed of the school of computing, at the National University of Singapore provided research possibilities (http://www.indriya.comp.nus.edu.sg/motelab/html/index.php) in sensor network programming environments, communication protocols, system design and applications. It is a sensor network testbed based on Hadoop cloud computing framework that stores large-scale of sensor data using HBase. It also provided a permanent framework for development and testing of sensor network protocols and applications. Registered users may interact with the testbed through a web-based interface based on Harvard's Motelab's interface. Indriya testbed does not include a metamodelling phase when compared to the proposed work.

3 Proposed sensor cloud architecture

The purpose of our sensor cloud is to monitor variations in humidity, temperature and moisture over time. The data gathered will be used for agriculture purpose, ground water resource and ecology department.

The approach uses a scalable, hierarchical and collaborative architecture. Hybrid, large-scale distributed sensor nodes are coupled by a sensor cloud middleware system. Sensor web interface proposed will be user-friendly interface and flexible integrated frame work for sensor cloud which has a good performance and does not consume much time for installation at the client side.

The procedure to implement sensor cloud based on the architecture described below has the following major steps:

1 Establishing the connection between sensor motes in an agriculture area through the base station to the Moteview.

2 On the Moteview, a service that collects extracts and formats the sensor data to useful information from raw data.

3 Refined data is loaded and published to a sensor cloud through the middleware.

4 SWE is used to access real-time sensor data through the sensor cloud and the client program subscribes to the specific sensor cloud services.

5 The web interface for accessing sensor data with scheduling and alert facilities to users who are interested.

6 The information is parsed and visualised in a defined collaborative session client for sharing.

The implemented architecture of the proposed sensor cloud framework consists of back-end with layers such as sensor layer and middleware layer followed by front-end with layers such as web service layer and application layer. The detailed discussion on these back-end and front-end implementation are provided in Section 4.

Figure 1 Sensor cloud framework architecture (see online version for colours)

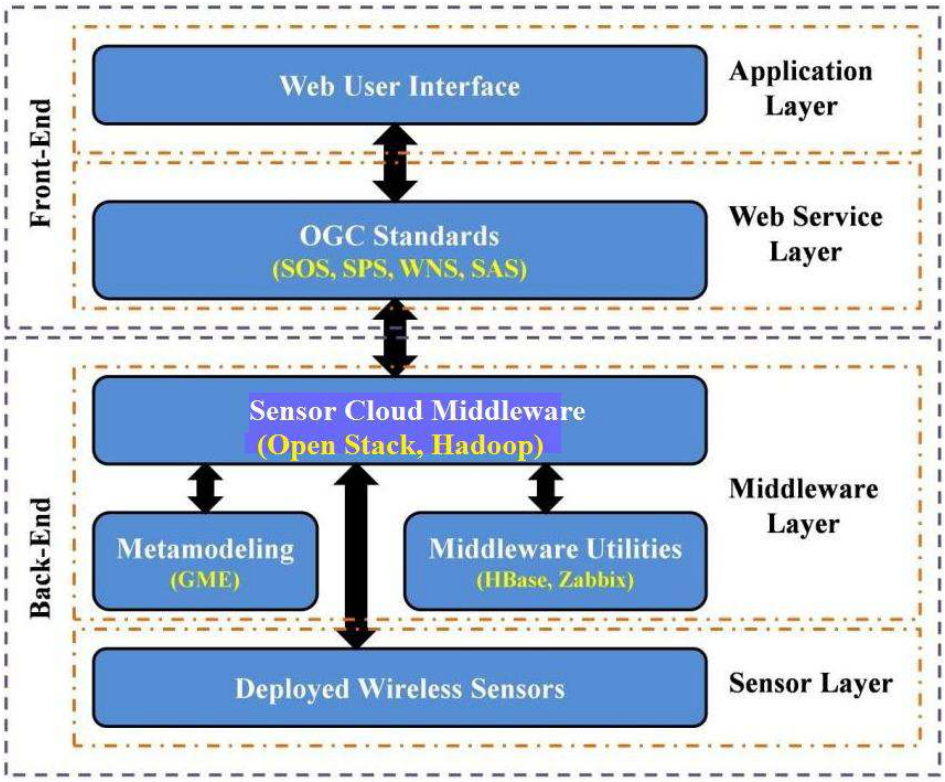


Figure 1 shows the sensor cloud framework architecture of the proposed work with components used in each layer for both back-end and front-end. Sensor layer deals with wireless sensors which collects environmental data. The middleware layer consists of Hadoop framework implemented on Open Stack as sensor cloud middleware with metamodelling tool, for designing a workflow model for the sensor cloud framework, HBase for big data management and Zabbix for monitoring sensor cloud cluster nodes.

Web service layer comprises of a collection of web services based on OGC standards such as SOS, SPS, WNS and SAS. This layer connects the middleware layer to the application layer. The application layer provides users with web services for interacting with the deployed wireless sensors through sensor cloud middleware.

The steps given below explain the interaction between the sensor cloud middleware and the SWE service based on OGC standards:

1 Using the OGC interface, the user subscribes to the necessary services in the cloud computing utility based on their need.

2 The meta-data generation service is called by the sensor cloud middleware to generate the modelling and meta-data for each request like date of submission, name of the machine submitted from, and the user's username on the submission machine, types of user (premium, regular or new).

3 Repeat the Step 2 for multiple jobs and generate the meta-data information for all the users and update the same in the information service and catalogue module of the sensor cloud middleware.

4 Based on the information from the previous step for each user and using scheduling policy necessary data is retrieved from the back end.

5 The information service is been updated or sorted according to the OGC standards.

6 Steps 2 to 5 will be repeated for the resources of the sensor cloud. The request of the user will be submitted to the cloud through the middleware and the results are sent back to the user in a standard form.

7 The logging and book keeping service of the sensor cloud maintains all the necessary information about the user to improve the performance of the system and can be used for prediction analysis.

8 At the end of the service, the user of the system will be kept in track about various services she/he was using and alert services that are sent to the specific user.

4 Sensor cloud framework utilities

Sensor cloud framework architecture consists of two main sections, backend with sensor and middleware layers followed by front-end with web service and application layer. The utilities used in each section are discussed in the following sub-sections.

4.1 Back-end utilities

The back-end comprises of two layers, namely sensor layer, consisting of geographically deployed wireless sensors collecting environmental data and middleware layer, composed of metamodelling, sensor cloud middleware and its utilities. Figure 2 shows the utilities used in the backend of the proposed work.

4.1.1 Sensor layer

Sensor layer consists of two types of wireless sensors for collecting environmental data such as temperature and humidity. The types of sensors used are micaz motes namely MDA100CB for collecting temperature values and MDA300CA for collecting humidity values as depicted in Figure 3. The collected values are stored in MySQL which are later transferred to big data utility for storage and processing.

Figure 2 Overview of back-end utilities (see online version for colours)

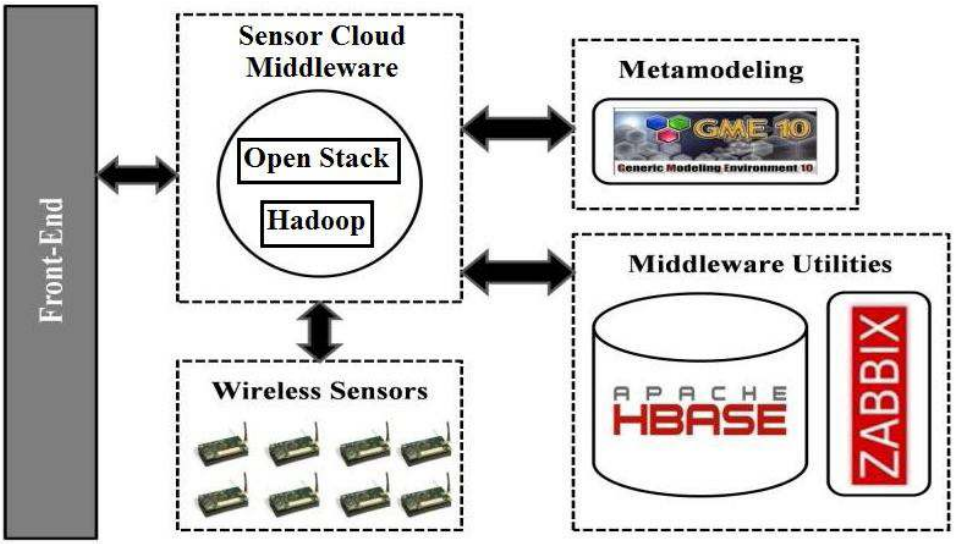
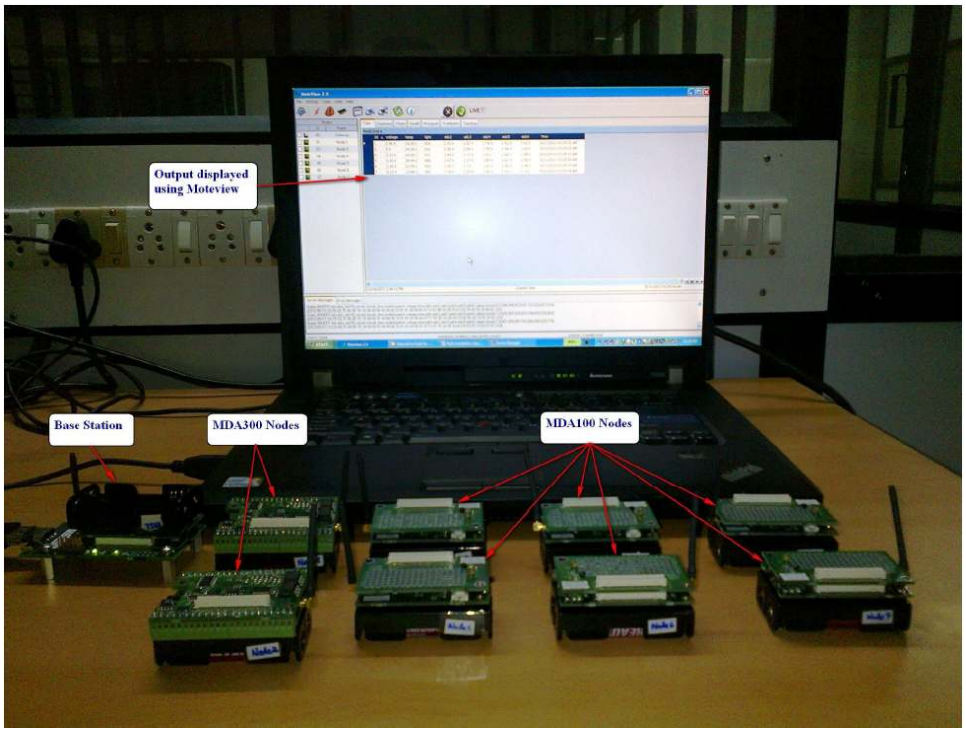


Figure 3 Sensor setup used for designing the OGC interface (see online version for colours)



4.1.2 Middleware layer

The middleware layer consists of components, namely metamodelling with GME as a metamodelling tool, Sensor cloud middleware with Hadoop and Open Stack framework, Middleware utilities with HBase as a big data store and Zabbix as a middleware monitoring tool.

4.1.2.1 Metamodelling

Metamodelling is an abstraction of a model highlighting the model properties and it depends on model driven architecture (MDA) (Liu and Wang, 2011). A domain specific modelling tool such as GME was used for representing the abstract of a model. GME tool is used to generate an abstract workflow model of the sensor cloud middleware. The reusability of the generated workflow model plays a prominent role in choosing this tool. Metamodelling technique acts as a modelling tool that provides a workflow model for the sensor cloud middleware beforehand implementation of the middleware.

4.1.2.2 Sensor cloud middleware

Sensor cloud middleware acts as an intermediary between sensor cloud and applications for the users. The core component of the sensor cloud middleware proposed is the Hadoop framework deployed on Open Stack cloud. This integration enables parallel processing of sensor data irrespective of the differences in software and hardware stack used by multiple Hadoop clusters. Hadoop framework and the context-aware collaboration framework were integrated which effectively processed massive sensor events and semantically analysed the big data within the cluster environment as discussed by Park et al. (2014).

Hadoop is an open source parallel processing framework for storing, processing and running distributed applications on large amounts of data (Apache Hadoop, http://www.hadoop.apache.org/docs/stable/index.html).

MapReduce programming model of the Hadoop framework provides a distributed file system and a framework for the analysis and transformation of bulk amount of data sets. In Wang et al. (2013b), detailed description about the data flow of a MapReduce job was provided. MapReduce framework harnesses the power of distributed computing without the complexities such as fault tolerance and data loss. Hadoop serves as a tool for partitioning and computation of data in parallel and in location monitoring using rack awareness features as discussed by Fox et al. (2008).

The Hadoop cluster consists of one active master node, a passive master node and three slave nodes. The active master node consists of an active Namenode, a JobTracker, a Datanode and a TaskTracker. The passive master node consists of a passive Namenode, a JobTracker, a Datanode and a TaskTracker all in passive state. The passive master node and active master node shares a common folder over the network, which stores snapshots of the active Namenode's status. This cluster setup avoids single point failure, which results when only one Namenode exists. Such robustness feature of the Hadoop cluster minimises the Namenode downtime and loss of data.

Each slave node consists of a Datanode and a TaskTracker. In Phadke et al. (2010), discussion about the characterisation provides the ability for unsupervised learning of individual Hadoop jobs has been done. In Wang et al. (2013a), an alternate design and

implementation of Hadoop termed as GHadoop was proposed. According to Wang et al. (2013a), G-Hadoop can schedule data processing tasks across nodes of multiple clusters, whereas normal Hadoop can schedule data processing tasks only within the nodes of a single cluster. Hadoop is scalable in terms of adding new nodes to the existing Hadoop cluster without damaging the data within the cluster. Hadoop not only supports scalability but also allows addition and maintenance of heterogeneous nodes. Open Stack is the software platform for cloud computing. Multiple heterogeneous Hadoop clusters can be run on Open Stack. The three features such as robustness, scalability and heterogeneity of the Hadoop framework and Open Stack serves as the main reasons for choosing it as sensor cloud middleware in the proposed work.

4.1.2.3 Middleware utilities

Middleware utilities used in the proposed work are HBase and Zabbix. HBase plays a huge role in the form of sensor cloud middleware utility as the growing sensor data needs to be effectively managed. Discussion about the research opportunities, challenges, and issues in big data management were done in Chen and Zhang (2014) and Hashem et al. (2014). Architecture, mainly for managing large-scale sensor data was proposed by Hohwald et al. (2010).

HBase is an open-source, column-oriented and distributed data store (Apache HBase, http://www.hbase.apache.org/book/architecture.html). In Attebury et al. (2009), HBase was described as a data store with bi table-like capabilities on top of Hadoop and HDFS. In Ku (2011), HBase is termed as a column-oriented database structure that enables the storage of data column by column. Sqoop is a tool particularly designed for bulk data transferring between Hadoop and relational databases. Sqoop is used to import the streaming (Apache Sqoop, http://www.sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html). Real-time wireless sensor data collected and stored in MySQL to HBase. The operations are much simplified that significantly reduces the cost of the ownership of Petabyte scale data storage over alternative solutions.

Querying an HBase table is difficult for a new user. In order to overcome this issue, a data query mechanism similar to SQL called HiveQL is used on top of HBase (Apache Hive, http://www.hive.apache.org/index.html). This is accomplished by integrating Hive and HBase by creating an external table in Hive for an HBase table as discussed in Jing-min and Guo-Hui (2010). Hive's external table keeps track of HBase insertion and deletion of rows or columns. A similar approach to manage massive sensor datasets using Hadoop was discussed by Bao et al. (2012). The main reason for opting HBase is that by the integration of HBase and Hive, querying process is made a lot easier.

A disaster management monitoring system for wireless sensors has been discussed in Cen et al. (2011). Proposed work used Zabbix, an enterprise-class open source distributed monitoring tool for monitoring Hadoop cluster nodes. The Zabbix monitoring tool can monitor the health and integrity of servers, and also has the ability to alert users through email using a flexible notification mechanism for any type of custom trigger events. Zabbix monitors the Hadoop cluster node for status. When the trigger is activated, an alert is sent to the admin so that necessary actions can be taken to avoid loss of data. The ability of Zabbix to monitor each cluster node and alert the administrator in case of node failure, proved to be a valid reason choosing it in the proposed work.

4.2 Front-end utilities

The front-end comprises of two layers, namely web service and application layer. The web service layer provides services based on the standards and specifications defined by OGC in order to integrate sensor cloud middleware with web user interface. The application layer consists of user interface for the sensor cloud middleware which eases the workload of users by providing it as web services. Figure 4 shows the components of the front-end in the proposed work.

Figure 4 Overview of front-end utilities (see online version for colours)

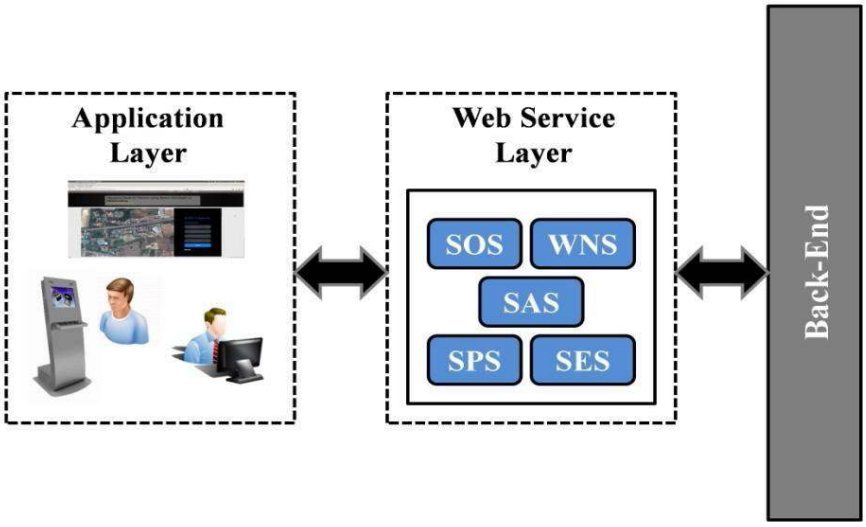
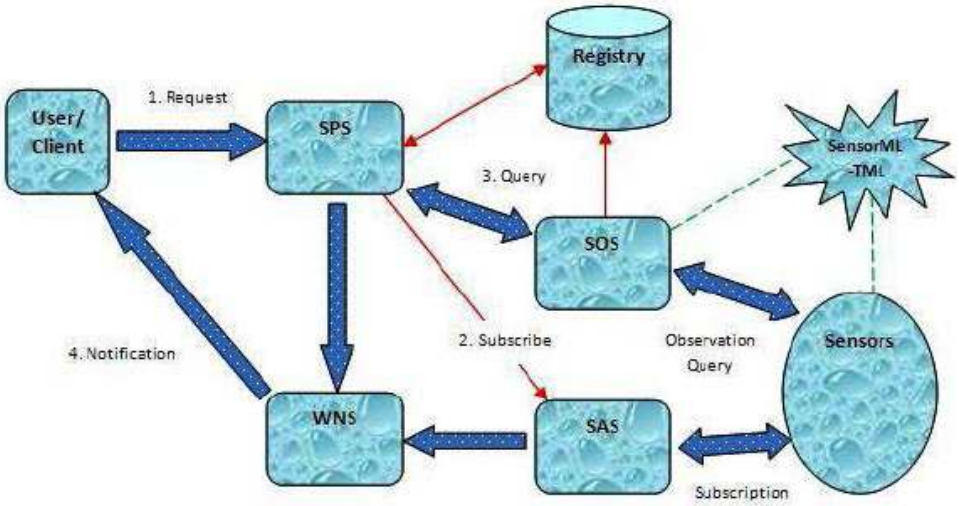


Figure 5 SWE service interactions based on OGC standards (see online version for colours)



4.2.1 Web service layer

The web service layer acts as a communication layer between the middleware layer and the application layer. The web service layer consists of a user interface using Java web services based on OGC standards. OGC standards provide open standard protocols for interfaces, encodings, schemas and architectures (http://www.opengeospatial.org/standards/is). OGC defines a suite of standard encodings and web services known as SWE. SWE provides features such as discovery of sensors, their processes and observations followed by sensor tasking. Additional features such as accessing observations and observation streams along with publish-subscribe capabilities for alert system as shown in Figure 5. SWE specifications enable implementation of interoperable and scalable service oriented networks of heterogeneous sensor systems and client applications and it is further classified as encoding and web service specifications.

(SWE, http://www.opengeospatial.org/projects/groups/sensorwebdwg) SWE encoding specifications are:

- Sensor Model Language (SensorML), a set of Standard models and XML schema for describing the processes within sensor and observation processing systems.
- Observations and Measurements (O&M), a set of general models and XML encodings for observations and measurements.
- Transducer Markup Language (TML), a set of conceptual model and XML schemas for depicting transducers and sustaining real-time streaming of data, to and from sensor systems.

SWE web service specifications are:

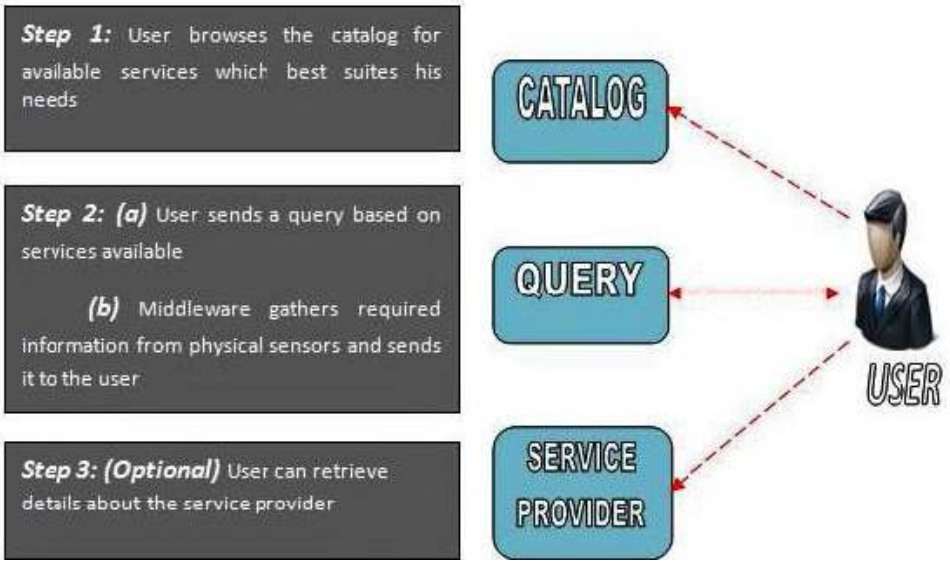
- SOS, a web service interface description for requesting observations from sensor networks and observation repositories. SOS is the intermediary between a client and an observation repository (SOS, http://www.opengeospatial.org/standards/sos).
- SPS, an open interface for a web service by which a client can determine the feasibility of collecting data from one or more sensors or models as well as submit collection requests (SPS, http://www.opengeospatial.org/standards/sps).
- SAS, a standard web interface for publishing and subscribing to sensor alerts (SAS, http://www.portal.opengeospatial.org/files/?artifact_id=15588).
- WNS, a web service interface definition for the transmission of asynchronous messages between SWE services. WNS is an intermediary for transmission of alert messages between SAS and the client (http://www.portal.opengeospatial.org/files/?artifact_id=18776).

The proposed work implements the SWE web service specifications such as SOS, SPS, WNS and SAS. Janowicz et al. (2013) proposed a transparent and RESTful SOS proxy that provided clients with sensor data on request without any modifications to existing services. A crisis management system based on OGCs SOS for accessing sensor data and SES for filtering sensor data was proposed by Jirka et al. (2013). De Longueville et al. (2010) described how the components of SWE standards such as SOS, SAS, SES and SPS were used to provide users with services.

4.2.2 Application layer

In order to utilise full potentiality of the sensor cloud middleware, a well designed user interface is needed. The user interface must be easy to use and provide fast results based on the user request. To attain such user interface, users must be provided with services to fulfil need for fast responses as shown in Figure 6. Such a web user interface that manages middleware framework through services was discussed in Le-Phuoc et al. (2012) and Tracey and Sreenan (2013).

Figure 6 User interaction using OGC (see online version for colours)



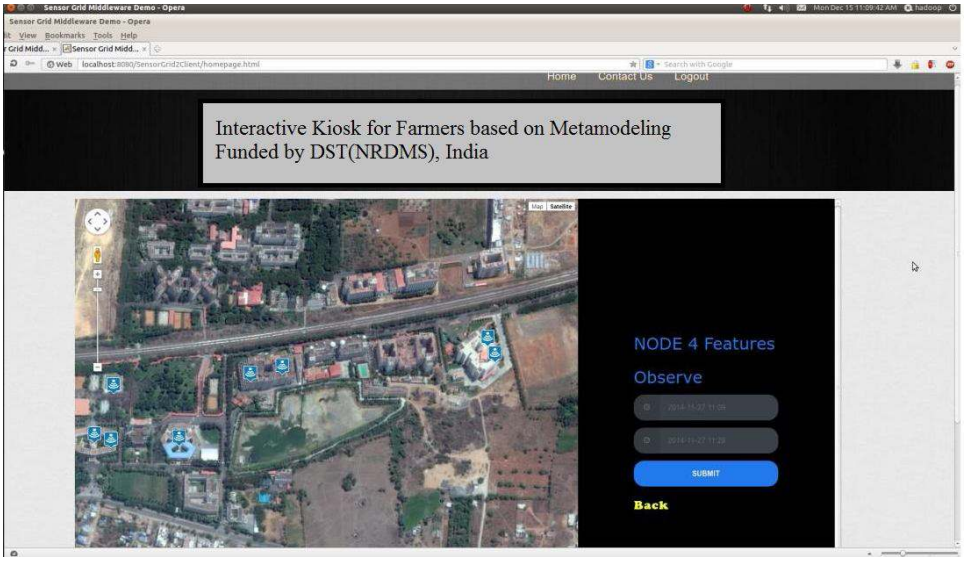
The standards such as XML, Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) were used to create and manage web services. XML provides a communication gateway between client-side applications and the server-side resources and services. SOAP is a communication protocol used to send requests and receive a response in XML format. WSDL is a combination of SOAP and XML for describing and providing web services to the clients through internet. WSDL description file provides a series of web service attributes such as input/output messages, operation method, and port type and so on as described by Yang et al. (2012). The application layer provides the users with web interfaces developed using Java. The three types of services availed by the application layer are information retrieval, scheduling and warning services.

4.2.2.1 Information retrieval service

The service is based on OGCs SOS standard with feature to observe historical sensor data. Figure 7 shows the snapshot of the user interface with information retrieval service and observation features. The historical sensor data is stored in HBase data store. As

discussed in the middleware layer, Hive is integrated with HBase to provide SQL-like query support. The observation feature requires two parameters such as starting time and ending time of the observation to be viewed. Starting time is the time at which the fetching of the observation data is starts while the ending time is the time at which fetching of the observation data stops. On submission of the observation request, the web service designed in Java queries the HBase data store through Hive JDBC connection and provides the response in both XML and HTML formats.

Figure 7 User interface with observation feature (see online version for colours)



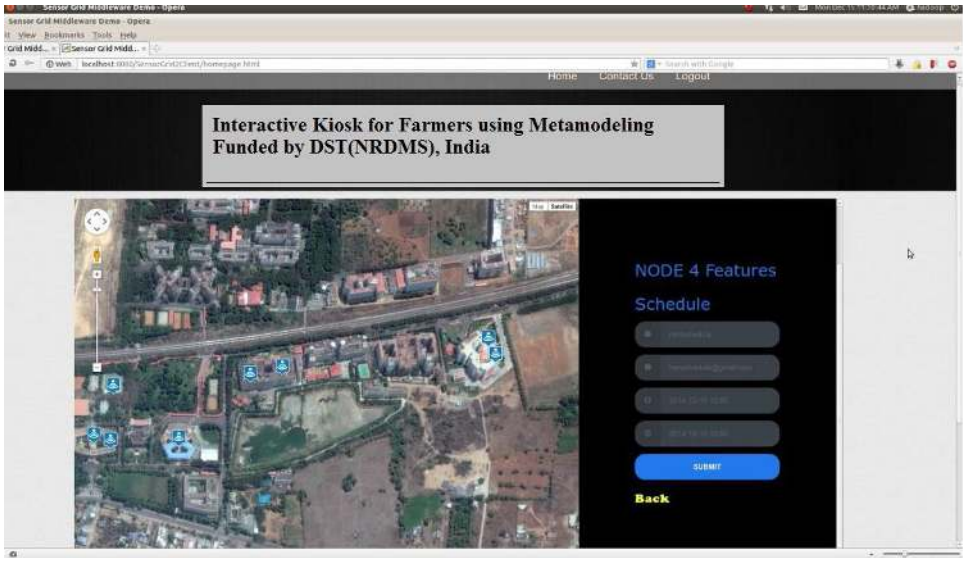
The server-side XML request sent will contain the node type, node id, starting time and ending time from which the historical sensor data is to be observed. Sample server-side XML request will look as given below:

```
<?xml version = "1.0" encoding = "UTF-8"?
<GetObservation
xmlns = "http://www.opengis.net/sos/1.0">
...
service = "SOS" version = "1.0.0">
<nodeType> MDA100CB </nodeType>
<nodeID> 4 </nodeID>
> <startTime> 2014-11-27
11:00 </startTime>
<endTime> 2014-11-27
11:30 </endTime>
</GetObservation>
```

4.2.2.2 Scheduling service

The scheduling service is based on OGCs SPS and WNS standards. This service is used to schedule the sensors to collect data during a specific time period and send back those data to the users through email. SPS performs the task of scheduling the wireless sensors to collect data and the WNS is used for notifying the users with scheduling results. The real-time wireless sensor data collected are stored in HBase data store. Figure 8 shows the snapshot of the user interface with schedule feature which requires user parameters such as scheduling name, email, starting time and ending time of scheduling.

Figure 8 User interface with scheduling feature (see online version for colours)



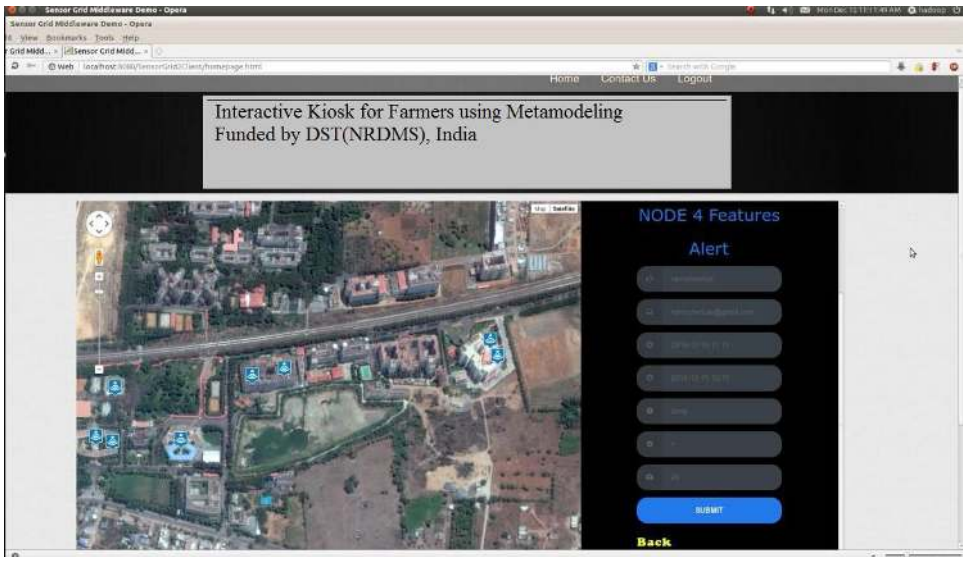
The scheduling name is a custom name to be provided by the user for the scheduling request. Email is the user email through which scheduling results will be provided. Starting and ending time is the time at which scheduling starts and ends respectively. Since the sensor data is stored in HBase in real-time, the Java web service sends the scheduling request to collect sensor data from HBase itself and provides the response in comma separated values (CSV) format through email.

4.2.2.3 Warning service

The warning service is based on SAS and WNS specification which provides users with the facility to perform custom trigger for a sensor that sends alert to user through email when it meets a specific criteria provided by the user. SAS is used to define filters for custom alerts, whereas WNS is used to send notifications to a user when alert trigger is activated. The snapshot of user interface with alert feature is shown in Figure 9 which

needs user inputs such as alert name, email, starting time, ending time, parameter to be monitored, user condition, and value.

Figure 9 User interface with alerting feature (see online version for colours)

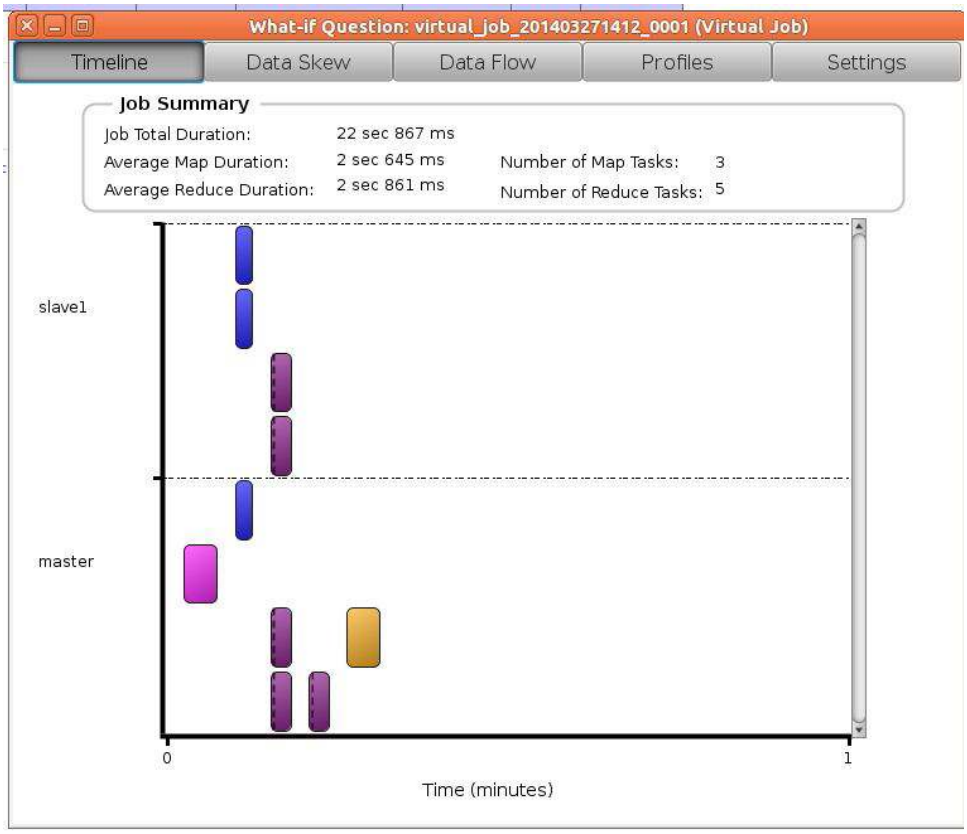


Alert name is the name provided by the user to alert and email is the id provided by the user to receive alerts. Parameter is the values to be monitored based on user request, namely temperature and humidity. Condition and value are the user-defined custom condition and value of alerts respectively. The user input data is processed by the SAS standard to create a trigger based on the input. When the trigger is activated, the details are sent to user's email using WNS standard.

5 Result and discussions

In general, the proposed methodology can be classified into back-end and front-end utilities. The back-end comprises of metamodelling, sensor cloud middleware and the middleware utilities such as sensor data management, performance analysis, performance tuning (starfish's what-if engine), and monitoring too (Zabbix). The front-end concentrates on the web user interface developed using Java web services based on OGC standards. Starfish, an open source project, is a tool that performs auto-tuning for the configurations in Hadoop based on the load, data, cluster, etc., and provides the best performance (http://www.cs.duke.edu/starfish/). It enables Hadoop users and applications to achieve good performance automatically through three levels of tuning with a what-if engine, optimiser, scheduler and a data manager. Figures 10 and 11 depicts the starfish architecture with levels of tuning.

Figure 10 Starfish (what-if engine) before tuning (see online version for colours)



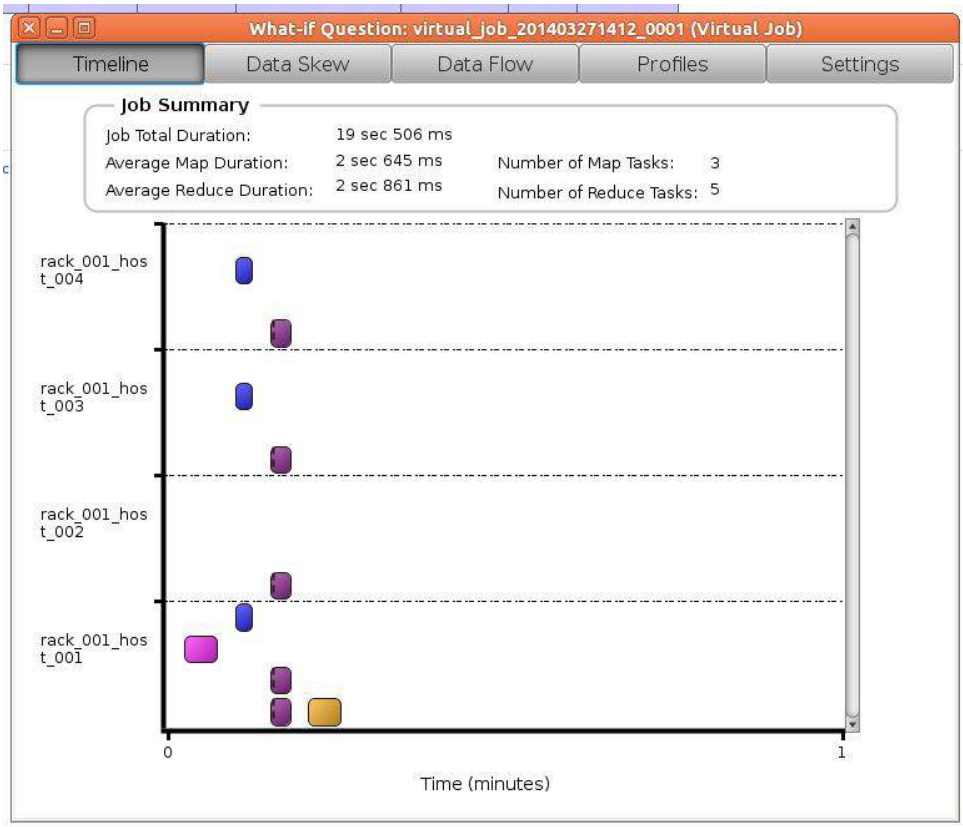
Metamodelling using GME serves as a modelling tool with reusability feature that avoids redesigning of the sensor cloud framework workflow from scratch in case if modifications are needed. The sensor cloud middleware enables parallel processing using the map/reduce programming and maintaining a large number of clusters with features such as scalability, fault tolerance and heterogeneity.

Sensor data management is a sensor cloud middleware utility for storing and maintaining big data. In Hadoop framework, HBase is used as a big data utility for managing bulk amount of streaming sensor data. Effective monitoring tool, Zabbix, as shown in Figure 12, monitors cluster health and a prominent performance analyser for the middleware to provide in-depth details on map/reduce jobs. The user interface provides users with features in the form of web services and is based on specifications defined by OGCs SWE. The specifications used in the proposed work are SOS, WNS and sensor event service which enables features such as information retrieval, scheduling and warning respectively.

The information retrieval feature has been performed using the SOS by querying the historical sensor data stored in HBase. Scheduling feature provided a way to schedule the sensor for a particular time to collect specific data and notify the users when scheduling is completed. This feature utilised SPS for tasking sensors and WNS for notifying the

users. Warning feature sends an alert to users through the contact email provided by the user when a user defined condition for the alert is met. This feature was achieved by combining the WNS for sending notifications and the SAS for defining custom filtering for alert triggers.

Figure 11 Starfish (what-if engine) after tuning (see online version for colours)



To make the comparison of the proposed work more useful and detailed, improved percentage efficiency for all iterations were performed. The influence of starfish tuning, played a crucial role in making the proposed methodology more efficient. It was further observed that from the monitoring tool that the total running time of a query is indirectly proportional to total physical memory used (i.e.,) the total running time of a query is less when it utilises more physical memory within the cluster. Total running time of a query is directly proportional to the time spent in mapping phase (i.e.,) when time spent in mapping by a query is less, the total running time of a query is also less. Total running time of a query is directly proportional to total CPU time spent (i.e.,) when the total CPU time spent by a query is less, the total running time of a query is also less.

The infrastructure was capable of providing adaptive level of efficiency, tolerance in a network of unreliable, heterogeneous computation nodes.

Figure 12 Zabbix monitoring tool (see online version for colours)

6 Conclusions

The proposed methodology demonstrated the effectiveness of both the back-end and front-end utilities used for environmental monitoring. The metamodelling tool made redesigning the sensor cloud framework workflow easier without starting the modelling process from scratch. The sensor cloud middleware proved to be an efficient and feasible tool to manage the clusters with ease. The features such as scalability, robustness, heterogeneity which overcome issues such as hardware failure, single point of failure and fault tolerance proves to be the key role for choosing the proposed middleware. Processing and managing big data was handled by HBase along with tools such as Hive and Sqoop.

The cluster was monitored using Zabbix tool which also provided alerts to administrators through email in case of downtime of any cluster nodes and processes. The user interface provided users with features in the form of web services developed using Java and was based on OGC standards such as SOS, SPS, WNS and SAS. Future work can be concentrated on implementing more features based on other OGC standards for the web user interface and analysing the applications of our proposed middleware.

Acknowledgements

This work is a part of a funded project sponsored by DST (NRDMS), New Delhi, India.

References

Apache Hadoop [online] http://www.hadoop.apache.org/docs/stable/index.html (accessed 10 March 2015).

Apache HBase [online] http://www.hbase.apache.org/book/architecture.html (accessed 10 March 2015).

Apache Hive [online] http://www.hive.apache.org/index.html (accessed 10 March 2015).

Apache Sqoop [online] http://www.sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html (accessed 10 March 2015).

Attebury, G., Baranovski, A., Bloom, K. and Bockelman, B. (2009) 'Hadoop distributed file system for the grid', *NSS/MIC 2009: Proceedings of IEEE Nuclear Science Symposium Conference Record*, October-November, pp.1056–1061.

Bao, Y., Ren, L., Zhang, L., Zhang, X. and Luo, Y. (2012) 'Massive sensor data management framework in cloud manufacturing based on Hadoop', in *INDIN 2012, Proceedings of 10th IEEE International Conference on Industrial Informatics*, July, pp.397–401.

Bisoi, S., Bhunia, S.S., Roy, S. and Mukherjee, N. (2013) 'iSENSE: intelligent sensor monitoring services with integrated WSN Testbed', *Procedia Technology*, Vol. 10, pp.564–571, DOI: 10.1016/j.protcy.2013.12.396.

Cen, J., Yu, T., Li, Z., Jin, S. and Liu, S. (2011) 'Developing a disaster surveillance system based on wireless sensor network and cloud platform', in *ICCTA 2011, Proceedings of the IET International Conference on Communication Technology and Application*, October, pp.757–761.

Chatterjee, S. and Misra, S. (2015) 'Optimal composition of a virtual sensor for efficient virtualization within sensor-cloud', *Proceedings of IEEE ICC, the IEEE International Conference on Communications*, 8–12 June, London, UK.

Chatterjee, S. and Misra, S. (2016a) 'Adaptive data caching for provisioning sensors-as-a-service', *Proceedings of IEEE BlackSeaCom*, 6–9 June, Varna, Bulgaria.

Chatterjee, S. and Misra, S. (2016b) 'QoS estimation and Selection of CSP in oligopoly environment for internet of things', *Proceedings of IEEE WCNC 2016*, 3–6 April, Doha, Qatar.

Chatterjee, S., Ladia, R. and Misra, S. (2015a) 'Dynamic optimal pricing for heterogeneous service-oriented architecture of sensor-cloud infrastructure', *IEEE Transactions on Services Computing*, Vol. 10, No. 2, pp.203–216, DOI: 10.1109/TSC.2015.2453958.

Chatterjee, S., Misra, S. and Khan, S.U. (2015b) 'Optimal data center scheduling for quality of service management in sensor-cloud', *IEEE Transactions on Cloud Computing*, No. 99, pp.1, DOI: 10.1109/TCC.2015.2487973.

Chatterjee, S., Sarkar, S. and Misra, S. (2015c) 'Quantification of node misbehavior in wireless sensor networks: a social choice-based approach', *Proceedings of IEEE ICC 2015 – 4th IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2015)*, 8–12 June, London, UK.

Chatterjee, S., Sarkar, S. and Misra, S. (2014) 'Evacuation and emergency management using a federated cloud', *IEEE Cloud Computing*, Vol. 1, No. 4, pp.68–76.

Chen, C.L.P. and Zhang, C-Y. (2014) 'Data-intensive applications, challenges, techniques and technologies: a survey on big data', *Information Science*, Vol. 275, pp.314–347, DOI: 10.1016/j.ins.2014.01.015.

De Longueville, B.. Annoni, A., Schade, S., Ostlaender, N. and Whitmore, C. (2010) 'Digital Earth's nervous system for crisis events: real-time sensor web enablement of volunteered geographic information', *International Journal of Digital Earth*, Vol. 3, No. 3, pp.242–259, DOI: 10.1080/17538947.2010.484869.

Ding, Z., Xu, J. and Yang, Q. (2013) 'SeaCloudDM: a database cluster framework for managing and querying massive heterogeneous sensor sampling data', *The Journal of Supercomputing*, Vol. 66, No. 3, pp.1260–1284, DOI: 10.1007/s11227-012-0762-1.

Fox, G., Ho, A., Wang, R., Chu, E. and Kwan, I. (2008) 'A collaborative sensor grids framework', in *CTS 2008, Proceedings of IEEE International Symposium on Collaborative Technologies and Systems (CTS)*, May, pp.29–38.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U. (2014) 'The rise of big data on cloud computing: review and open research issues', *Information Systems*, Vol. 47, pp.98–115, DOI: 10.1016/j.is.2014.07.006.

Hohwald, H., Frias-Martinez, E. and Oliver, N. (2010) 'ARBUD: an architecture for building pervasive user models from massive sensor datasets', in *PUMP 2010, Proceedings of Pervasive User Modeling and Personalization, Indriya* [online] http://www.indriya.comp.nus.edu.sg/motelab/html/index.php (accessed 10 March 2015).

Janowicz, K., Broring, A., Stasch, C., Schade, S., Everding, T. and Llaves, A. (2013) 'A RESTful proxy and data model for linked sensor data', *International Journal of Digital Earth*, Vol. 6, No. 3, pp.233–254, DOI: 10.1080/17538947.2011.614698.

Jing-Min, L. and Guo-Hui, H. (2010) 'Research of distributed database system based on Hadoop', in *ICISE 2010, Proceedings of IEEE International Conference on Information Science and Engineering*, December, pp.1417–1420.

Jirka, S., Nust, D. and Proß, B. (2013) 'Sensor web and web processing standards for crisis management', in *ISCRM 2013, Proceedings of 10th International Conference on Information Systems for Crisis Response and Management (ISCRM) Conference*, May.

Ku, W-Y. (2011) 'The cloud-based sensor data warehouse', in *ISGC 2011 & OGF 31, Proceedings of the International Symposium on Grids and Clouds and the Open Grid Forum*, WNS [online] http://www.portal.opengeospatial.org/files/?artifact_id=18776 (accessed 10 March 2015).

Kumar, N., Misra, S., Rodrigues, J.J.P.C. and Obaidat, M.S. (2015) 'Coalition games for spatio-temporal big data in internet of vehicles environment: a comparative analysis', *IEEE Internet of Things Journal*, January, OGC, DOI: 10.1109/JIOT.2015.2388588 [online] http://www.opengeospatial.org/standards/is (accessed 10 March 2015).

Kyzirakos, K., Karpathiotakis, M., Garbis, G., Nikolaou, C., Bereta, K., Papoutsis, I., Herekakis, T., Michail, D., Koubarakis, M. and Kontoes, C. (2014) 'Wildfire monitoring using satellite images, ontologies and linked geospatial data', *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 24, pp.18–26, DOI:10.1016/j.websem.2013.12.002.

Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X. and Hauswirth, M. (2012) 'A middleware framework for scalable management of linked streams', *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 16, pp.42–51.

Liu, Y. and Wang, Y. (2011) 'A study of Metamodeling based on MDA', in *ICCRD 2011, Proceedings of IEEE 3rd International Conference on Computer Research and Development*, March, pp.171–173.

Madria, S., Kumar, V. and Dalvi, R. (2013) 'Sensor cloud: a cloud of virtual sensors', in *Software, IEEE*, 12 November, Vol. 31, No. 2, pp.70–77, ISSN: 0740-7459, DOI: 10.1109/MS.2013.141.

Misra, S. and Chatterjee, S. (2014) 'Social choice considerations in cloud-assisted WBAN architecture for post-disaster healthcare: data aggregation and channelization', *Information Sciences (Elsevier)*, Vol. 284, pp.95–117.

Misra, S., Chatterjee, S. and Obaidat, M.S. (2014a) 'On theoretical modeling of sensor-cloud: a paradigm shift from wireless sensor network', *IEEE Systems Journal*, Vol. 11, No: 2, pp.1084–1093, DOI: 10.1109/JSYST.2014.2362617.

Misra, S., Singh, A., Chatterjee, S. and Obaidat, M.S. (2014b) 'Mils-cloud: a sensor-cloud based architecture for the integration of military tri-services operations and decision making', *IEEE Systems Journal*, Vol. 10, No. 2, pp.628 – 636, DOI: 10.1109/JSYST.2014.2316013.

Misra, S., Singh, A., Chatterjee, S. and Mandal, A.K. (2015) 'QoS-aware sensor allocation for target tracking in sensor-cloud', *Ad Hoc Networks (Elsevier)*, October, Vol. 33, pp.140–153.

Moulik, S., Misra, S. and Obaidat, M.S. (2015) 'SMART-EVAC: a big data-based decision making system for emergency evacuation', *IEEE Cloud Computing*, Vol: 2, No. 3, pp.58–65 (Manuscript ID: CCM-2014-12-0063; accepted on 26 April 2015).

Open Stack [online] http://www.openstack.org/software/start/ (accessed 10 March 2015).

Park, K., Kim, Y. and Chang, J. (2014) 'Semantic reasoning with contextual ontologies on sensor cloud environment', *International Journal of Distributed Sensor Networks*, Vol. 2014, Article ID 693957, p.13, DOI:10.1155/2014/693957.

Phadke, S., Aggarwal, S. and Bhandarkar, M. (2010) 'Characterization of Hadoop jobs using unsupervised learning', in *CloudCom 2010, Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science*, November-December, pp.748–753.

SAS [online] http://www.portal.opengeospatial.org/files/?artifact_id=15588 (accessed 10 March 2015).

Shelestov, A.Y., Kravchenko, A.N., Skakun, S.V., Voloshin, S.V. and Kussul, N.N. (2013) 'Geospatial information system for agricultural monitoring', *Cybernetics and Systems Analysis*, Vol. 49, No. 1, pp.124–132.

SOS [online] http://www.opengeospatial.org/standards/sos (accessed 10 March 2015).

SPS [online] http://www.opengeospatial.org/standards/sps (accessed 10 March 2015).

SWE [online] http://www.opengeospatial.org/projects/groups/sensorwebdwg (accessed 10 March 2015).

Tian, Y. and Huang, M. (2012) 'Enhance discovery and retrieval of geospatial data using SOA and semantic web technologies', *Expert Systems with Applications*, Vol. 39, No. 16, pp.12522–12535, DOI:10.1016/j.eswa.2012.04.061.

Tracey, D. and Sreenan, C. (2013) 'A holistic architecture for the internet of things, sensing services and big data', in *CCGRID 2013, Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp.546–553.

Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C.J.A. and Buytaert, W. (2015) 'Web technologies for environmental big data', *Environmental Modelling & Software*, Vol. 63, pp.185–198, DOI: 10.1016/j.envsoft.2014.10.007.

Wang, L., Tao, J., Ranjan, R., Marten, H., Streit, A., Chen, J. and Chen, D. (2013a) 'G-Hadoop: MapReduce across distributed data centers for data-intensive computing', *Future Generation Computer Systems*, Vol. 29, No. 3, pp.739–750, DOI:10.1016/j.future.2012.09.001.

Wang, P., Wang, J., Chen, Y. and Ni, G. (2013b) 'Rapid processing of remote sensing images based on cloud computing', *Future Generation Computer Systems*, Vol. 29, No. 8, pp.1963–1968, DOI: 10.1016/j.future.2013.05.002.

Yang, C., Chen, N. and Di, L. (2012) 'RESTFul based heterogeneous Geo processing workflow interoperation for sensor web service', *Computers & Geosciences*, Vol. 47, pp.102–110, DOI: 10.1016/j.cageo.2011.11.010.