2013

Knowledge as a Service Framework for Disaster Data Management

Katarina Grolinger
Western University, kgroling@uwo.ca

Emna Mezghani
emezghan@laas.fr

Miriam AM Capretz
Western University

Ernesto Exposito
eexposit@laas.fr

Knowledge as a Service Framework for Disaster Data Management

Katarina Grolinger, Miriam A.M. Capretz
Department of Electrical and Computer Engineering,
Faculty of Engineering
Western University
London, ON, Canada N6A 5B9
{kgroling, mcapretz}@uwo.ca

Emna Mezghani, Ernesto Exposito
CNRS; LAAS; 7 av. du Colonel Roche, F-31400
Toulouse, FRANCE
Université de Toulouse; UPS, INSA, INP, ISAE;
LAAS; F-31400 Toulouse, France
{emezghan, eexposit@}laas.fr

Abstract— **Each year, a number of natural disasters strike across the globe, killing hundreds and causing billions of dollars in property and infrastructure damage. Minimizing the impact of disasters is imperative in today's society. As the capabilities of software and hardware evolve, so does the role of information and communication technology in disaster mitigation, preparation, response, and recovery. A large quantity of disaster-related data is available, including response plans, records of previous incidents, simulation data, social media data, and Web sites. However, current data management solutions offer few or no integration capabilities. Moreover, recent advances in cloud computing, big data, and NoSQL open the door for new solutions in disaster data management. In this paper, a Knowledge as a Service (KaaS) framework is proposed for disaster cloud data management (Disaster-CDM), with the objectives of 1) storing large amounts of disaster-related data from diverse sources, 2) facilitating search, and 3) supporting their interoperability and integration. Data are stored in a cloud environment using a combination of relational and NoSQL databases. The case study presented in this paper illustrates the use of Disaster-CDM on an example of simulation models.**

Keywords: disaster management; cloud computing; NoSQL; Knowledge as a Service (KaaS); big data.

I. INTRODUCTION

Each year, a number of natural disasters strike across the globe, killing hundreds and causing billions of dollars in property and infrastructure damage. Extreme weather events have been predicted by climate scientists and have been attributed to global warming. As the number of such events increases, minimizing the impact of disasters becomes imperative in today's society.

The role of information and communication technology in disaster management has been evolving. Large quantities of disaster-related data are being generated. Behaviour of critical infrastructures is being explored through simulation, response plans are being created by government agencies and individual organizations, sensory systems are providing potentially relevant information, and social media (Twitter, Facebook) have been flooded with disaster information [1]. Current data storage systems are disparate and provide few or no integration capabilities. To make the most of available information, a reliable and scalable storage system supported by information sharing, reuse, integration, and analysis is needed.

In other domains, recent advances in cloud computing, big data, and NoSQL have been changing how data are captured, stored, and analyzed. NoSQL solutions have been especially popular in Web applications [2], including Facebook, Twitter, and Google. However, the use of cloud and NoSQL solutions in disaster management has been sparse.

A solution to store disaster related data in a cloud environment can provide the following benefits to disaster management [3]:

- *High availability*. Within the cloud environment, data are automatically replicated, often across large geographic distances. If a region is affected by a disaster and a local data centre fails, the system remains available because it can switch to another data centre.
- *Scalability and elasticity*. The amount of disaster-related data is immense, and a cloud solution could adapt storage resources based on real time needs and priorities. Data can be automatically redistributed to take advantage of heterogeneous servers.
- *There is no need for a large initial investment*. The system can start small and be expanded by adding heterogeneous nodes as needed.

Moreover, NoSQL databases have a number of characteristics that can benefit disaster management, including [4]:

- *Flexible data structure*. Disaster data are extremely diverse, and therefore it would be almost impossible to store them in a predetermined data structure.
- *Great horizontal scalability*. NoSQL databases were designed for a cloud environment, and therefore they scale easily over a large number of commodity servers.
- *Performance*. For simple read/write operations, NoSQL databases can provide excellent performance.

Consequently, this paper proposes a Knowledge as a Service (KaaS) framework for disaster cloud data management (Disaster-CDM). KaaS [20] aims to generate from stored data in a cloud environment, knowledge such as advice or response to meet organizational needs. Therefore, Disaster-CDM has the following objectives of 1) Storing large amounts of disaster-related data from diverse sources, 2) Facilitating search of disaster data, and 3) Supporting their interoperability and integration.

K. Grolinger, E. Mezghani, M.A.M. Capretz, E. Exposito, Knowledge as a Service Framework for Disaster Data Management, The 22nd IEEE WETICE conference, pp. 313-318, 2013, DOI: 10.1109/WETICE.2013.48

The storage of large amounts of heterogeneous data will be achieved by using relational and NoSQL databases in the cloud environment. Knowledge delivery architecture will use semantic integration to facilitate search and interoperability.

The remainder of the paper is organized as follows: Section II reviews related work; the proposed Disaster-CDM is portrayed in Section III, while Section IV depicts a case study. Finally, conclusions and future work are presented in Section V.

II. Related Work

Research in disaster management involves many fields, including health science, environmental science, computer science, and a number of engineering disciplines. Crisis informatics [5,6], the area of research concerned with the role of information and technology in disaster management, has been attracting increased research attention recently.

Hristidis *et al.* [1] surveyed data management and analysis in the disaster domain. The main focus of their survey was on data analysis techniques without the storage aspect. In contrast, in Disaster-CDM, storage and analysis are considered as integral parts. Hristidis *et al.* identified the following data analysis technologies as relevant in disaster data management: information extraction, information retrieval, information filtering, data mining, and decision support. Similarly, Disaster-CDM uses a number of technologies from information extraction and retrieval. The survey reveals that the majority of research has focused on a very narrow area of disaster management, for example, a specific disaster event such as an earthquake or a flood, or specific disaster-related activities such as communication among actors, estimating disaster damage, and use of mobile devices. Hristidis *et al.* recognized the need for flexible and customizable disaster-management solutions that could be applied in different disaster situations. Disaster-CDM aims to provide such a solution using cloud and NoSQL approaches.

Silva *et al.* [7] aimed to integrate diverse, distributed information sources by bringing them into a standardized and exchangeable common data format. Their approach focused on data available on public Web sites. Data were first extracted from different source Web sites and stored in a relational database. Next, the data were transformed into Linked Open Data (LOD) and published. In contrast to their work which addressed data available on public Web sites, the proposed Disaster-CDM can accommodate various information sources. In addition, Disaster-CDM is designed for high availability and large amounts of data.

Palen *et al.* [5] presented a vision of technology-supported public participation during disaster events. They focused on the role of the public in disasters and how information and communication technology can transform that role. Similarly to [1], they recognized information integration as a core concern in crisis informatics.

Anderson and Schram [8], like Palen *et al.* [5], studied the role of public and social media in disaster events. They proposed a crisis-informatics data analytic infrastructure for the collection, analysis, and storage of information from Twitter. The main objective of their work was the support of other crisis information research by extracting disaster-related tweets from Twitter and storing them in a database. In their initial study [8], data were stored in a relational database, specifically MySQL. Later, after encountering scalability challenges, they transitioned to a hybrid architecture that incorporates relational and NoSQL databases [6]. Similarly, Disaster-CDM also uses a combination of relational and NoSQL databases. However, a combination of several NoSQL databases has been used to address the storage requirements of diverse data. Specifically, Disaster-CDM allows choice of storage solutions to suit data structures and access patterns. For example, ontologies are stored in a graph database because ontologies closely resemble a graph structure.

To provide disaster knowledge services, Disaster-CDM uses a KaaS approach. Within KaaS, a knowledge provider answers requests presented by knowledge consumers through knowledge services [9]. This approach has been used in various domains including disaster management [10, 11, 21]. Lino et al. [21] discuss KaaS for emergency response in natural disasters like tsunami and earthquakes, which is restricted to Interactive Digital TV. In Disaster-CDM, KaaS is suitable for accommodating structured and unstructured data stored in relational and NoSQL databases.

III. Disaster-CDM Architecture

The heterogeneity of the data involved in disaster-related activities is one of the main challenges in providing a comprehensive solution that could be used by various stakeholders in diverse disaster situations. Disaster-CDM addresses this challenge using the KaaS approach; disaster-related knowledge is provided as a knowledge service.

The Disaster-CDM architecture is illustrated in Fig. 1. It consists of two parts: knowledge acquisition and knowledge delivery services. Knowledge acquisition is responsible for acquiring knowledge from diverse sources, processing it to add structure to unstructured or semi-structured data, and storing it in databases. Data are stored in a cloud environment, specifically in a variety of relational and NoSQL databases. The second part, knowledge delivery services, is responsible for integrating information from different databases and delivering knowledge to consumers.

It was decided to process the information and to store the processed, enriched data because the time required for queries and information integration would be compatible with near real-time requirements of emergency situations. Therefore, this will allow shorter response time to the queries than performing the processing "on the fly".

The main reasons for choosing NoSQL databases in a cloud environment are 1) high availability ensures continuity during disasters, and 2) excellent scalability over a large number of commodity servers.

The following two sections describe the two main parts of Disaster-CDM: knowledge acquisition and knowledge delivery.

A. Knowledge acquisition

The Disaster-CDM knowledge acquisition function obtains data from heterogeneous data sources, processes them, and stores them in the cloud environment.

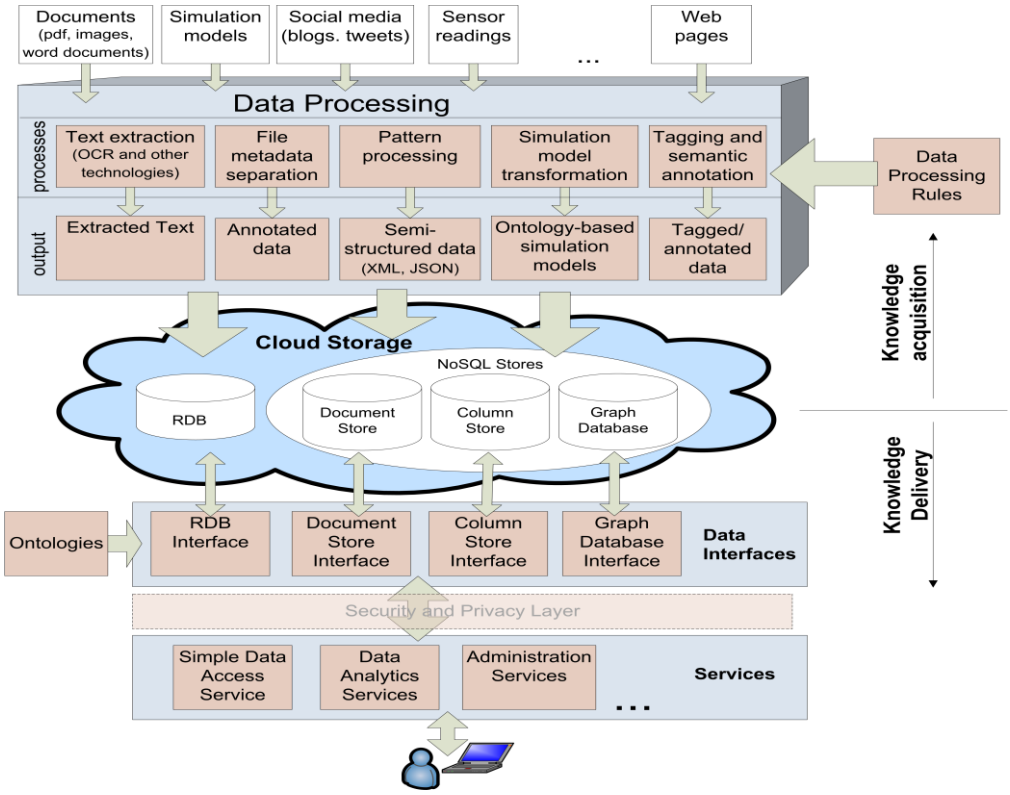

Figure 1. Disaster-CDM architecture.

1) Heterogeneous Data Sources

A few examples of information related to disasters are disaster plans, incident reports, situation reports, social media, simulation models including infrastructure and health-care simulation. As for representation formats, examples include MS Word, pdf, xml, a variety of image formats (jpeg, png, tiff), and simulation model formats specific to simulation packages. Data representation is important because it determines the methods that can be used to add structure to unstructured or semi-structured data.

From our experience working with local disaster-management agencies, the majority of information is stored in unformatted documents, primarily pdf and MS Word files. This agrees with the work of Hristidis *et al.* [1], who reported that most information is in pdf and MS Word files.

2) Data processing

Because the input data are so diverse, they cannot be processed using a single approach. Therefore, the processing is driven by the input data and by *data processing rules*, as illustrated in Fig. 1. *Data processing rules* specify what processes are to be applied to which input data and in which order. For example, a pdf incident report must go through *file metadata separation*, *text extraction*, and *pattern processing*.

The main processes with their associated outputs are included in Fig. 1. However, Disaster-CDM can be easily expanded to include new data processing methods.

Text Extraction from Images recognizes the text in an image and separates it [12]. This step prepares images and pdf files for other processing steps such as tagging. Text extraction is especially important in the case of diagrams such as flowcharts or event-driven process chains because these documents contain large amounts of text that can be used for tagging. MS Word documents also go through text extraction because they often contain images that may include relevant information.

File Metadata Separation makes use of file and directory attributes, including file name, creation date, last modified date, owner, and access permissions. File names themselves carry important information about content because they are typically chosen with the aim of describing the content. They are processed to separate the words contained in the file name. Keywords can also be used in the file name as a descriptive metadata. The creation date and last modified date can assist in distinguishing newer and potentially more relevant information from older and possibly outdated information. The file directory structure contains additional information about file content such as file structure (table of content) because directories are used to organize files. Directories can be seen as a categorization and therefore should be included in metadata separation.

Pattern Processing makes use of existing patterns within documents to extract the desired structure. Hristidis *et al.* [1] observed that most available information is stored in unstructured documents, but that "typically the same

organization follows a similar format for all its reports" [1]. Therefore, it is feasible to use patterns for information extraction. However, the number of organizations involved in disaster management is large, which may result in a large number of patterns. This represents a challenge because the patterns need to be identified before pattern processing can be applied. Another challenge is with new sources where people must be involved in identifying patterns.

Simulation Model Transformation is the process of converting simulation models into a representation which enables model queries and integration with other disaster-related data. Simulation is considered especially important for this project because it involves various domains which are crucial to disaster management. Therefore, simulation-related activities are described here in more detail.

Although the act of simulation is not domain-specific, simulation packages are usually application-oriented, meaning that they are designed for simulation experiments in a specific domain. These packages use different modeling approaches and domain-specific vocabularies and store simulation models in simulation-engine-specific file formats. Such simulation models cannot be directly queried, nor can they be integrated with other disaster-related data.

To extract as much information as possible from simulation model files, an ontology-based representation of simulation models has been used [13]. Unlike text-processing approaches, an ontology-based representation makes it possible to 1) address simulator-specific terminology, 2) remain schema-independent because ontologies do not have predefined schemata, and 3) focus on entities and their relations.

To transform simulator model files to their corresponding ontology-based models, the process described in [13] has been used. This process requires two simulator-specific entities to be created for each simulator: a simulator ontology, which identifies the simulator's building blocks [14]; and a simulation model reader [13], which is responsible for reading the simulator model files. Once those two entities have been created for a specific simulator, any model represented in that format can be transformed to its ontology-based representation. In the next step, these ontology-based simulation models will be stored in the cloud database.

Tagging and semantic annotation. Tagging is the process of attaching keywords or terms to a piece of information with the objective of assisting in classification, identification, or search [15]. Semantic annotations additionally specify how entities are related. In disaster-management data tagging, both manual and automated tagging are needed. Automated tagging applies various natural language processing (NLP) and soft computing techniques to add tags automatically to pieces of information. Because disaster data are immensely diverse, it might not be feasible to tag all content automatically. Images are examples of data which may require computationally expensive tagging. Therefore, manual tagging is used to supplement the automated approach. Tagging will be explored first, and semantic annotations will be addressed in later stages of the project.

3) Data Storage in the Cloud Environment

Relational databases (RDBs) are traditional data-storage systems designed for structured data. They have been used for decades due to their reliability, consistency, and query capabilities through a standard SQL language. However, they do not gracefully meet mass data needs. In other words, RDBs exhibit horizontal scalability challenges, big data inefficiencies, and limited availability [16].

In this context, the next generation of databases, namely NoSQL databases, have been designed for a distributed environment [3]. They are mainly dedicated to projects that are distributed, that involve large amounts of data, or that must scale. In the case of simple operations, NoSQL databases improve performance relative to traditional RDBs.

Disaster-CDM, as illustrated in Fig. 1, uses both relational and NoSQL databases. Even though there are four main categories of NoSQL databases [4], Disaster-CDM uses only three. The following discussion introduces all four categories and explains the choice of not including key-value databases in Disaster-CDM architecture.

Key-value databases are used for fast and simple operations. They have the simplest data model: they provide a simple mapping from the key to the corresponding value. When using a key-value database, relations between data are handled at the application level. This data model greatly restricts integration capabilities, and therefore it is not included in Disaster-CDM.

Document databases offer a flexible data model with query possibilities. They focus on optimized storage and access for semi-structured documents as opposed to rows or records. Document databases are considered an evolution of key-value databases because they include all the benefits of the key-value databases while adding query capabilities.

Column-family databases are suitable for very large datasets which have to be scaled at larger size. On the surface, they are similar to relational databases, but the difference lies on how they store the data. A relational database stores data by rows, while a column-family database stores them by columns. Each column is identified by a key value (row) and can be extended with arbitrary values. Column-family databases provide query capabilities.

Graph databases are specialized for efficient management of heavily linked data. Applications based on data with many relationships are well suited for graph databases because the cost of intensive operations like recursive "join" operation can be replaced by efficient graph traversals [4].

Despite the advantages of NoSQL databases, Disaster-CDM also accommodates relational databases. RDBs are still an appropriate solution for many applications because of their characteristics such as ACID transactions, their status as an established technology, and their advanced query capabilities. Moreover, existing data in relational databases do not need to be migrated. However, integration among relational and NoSQL databases is a challenge. Part of this challenge is the fact that NoSQL databases do not support a standard query language.

B. Knowledge delivery

The Disaster-CDM knowledge delivery service will answer information requests submitted by service consumers by integrating information stored in the cloud. As presented in Fig. 1, data access is mainly composed of three parts:

- *Ontologies*: These provide an overall view of the local ontologies representing each database independently of its category. Ontologies represent the mapping between heterogeneous sources which is needed to unify query capabilities.
- *Data interfaces:* After querying the ontology, it is necessary to access the data. Data interfaces enable translation of the generic query into a specific language that corresponds to the underlying database system. Thus, the data stored in heterogeneous sources can be accessed, analyzed, and administered.
- *Services:* This is the access layer for users. It provides services independently of how the data are stored. Thus, users are unaware of the storage architecture and are provided with a unified view of the data.

Security and privacy are outside the scope of this work; however, they are included in Fig. 1 to underline their importance.

IV. Case Study

The use of the proposed Disaster-CDM will be demonstrated here on an example including simulation models. This choice is motivated by the importance of simulation in a number of domains crucial for disaster management. Moreover, simulation models contain complex information that should be preserved in the transformation process. The complexity and heterogeneity of simulation models represents a challenge for storage and integration.

Specifically, the I2Sim [17] model, which was developed for the investigation of infrastructure interdependencies, is used. I2Sim is an interdependency simulator built upon MATLAB's Simulink engine. Simulink provides block libraries which can be customized to conform to a specific simulation domain. Complex models are managed by dividing models into hierarchies of sub-models. Accordingly, I2Sim builds upon Simulink by customizing Simulink blocks and providing entities specific to infrastructure interdependency simulation.

The I2Sim simulator model used in this case study was developed to investigate infrastructure interdependencies in an incident on the Western University campus. The model involves a number of infrastructures, including electricity, water, and steam distribution. It is complex and consists of several hierarchy levels. These hierarchy levels hide complexity and aid in model creation and management; however, they pose a challenge for model checking. To verify that the model conforms to a specific requirement, the user typically must open and review each sub-model in the hierarchy. Storing ontology-base representations in a database provides querying abilities and thus facilitates model checking. The simulation models are first processed to convert them into ontology-based models; then they are saved in a cloud database.

A. Simulation model processing

As described in Section III.A.2, simulation models are processed by transforming each simulator-specific proprietary model to its corresponding ontology-based model. In the case of I2Sim, the simulator model is stored in a Simulink-style .mdl file. The transformation of this .mdl file to an ontology-based model has been described in [13].

The transformation approach [13] enables the representation of ontology-based models in different ontology languages. In this case study, OWL [18] was used because it is the W3C-recommended ontology language.

B. Storing simulation models

Disaster-CDM is designed to enable the choice of a storage solution that corresponds to the data requirements in terms of data structure as well as access patterns.

After the simulation models have been transformed into ontologies, they are represented in OWL, which is characterized by a formal semantic and an abstract ontology structure that can be perceived as a graph.

Graph databases use graph structures with nodes, edges, and properties to represent and store data. They are optimized for efficient management and storage of graph-like data. Consequently, because ontologies can be perceived as graphs, it is apparent that graph databases are a good choice for storing ontologies as well as ontology-based simulation models. Another characteristic that makes a graph database a good choice is its query capabilities. Graph database implementations typically offer query capabilities using specialized graph query languages. Specifically, this case study uses the Neo4j graph database [19]. Neo4j can be queried using Cypher, a property graph query language developed by Neo4j; using Gremlin, a graph traversal language; or even using the RDF query language, SPARQL.

The processing stage creates ontology-based representations of simulation models in the OWL language. Next, these ontologies are loaded into Neo4j. Because Neo4j is a graph database and OWL ontologies are forms of graphs, loading ontologies into the database proved to be straightforward.

After ontologies are stored in the database, they can be queried, what makes it possible to perform model checking that otherwise would be done manually. This can be illustrated using an example involving channels and production cells. In simple terms, an I2Sim *production cell* is an entity that transforms inputs into outputs, while a *channel* transports entities. It was necessary to find which *production cells* were directly connected by *channels*. To do this directly on the I2Sim model, the user needs to check each *channel* to determine whether it directly connects to *production cells*. The hierarchy of sub-models makes this task especially challenging because each sub-model needs to be checked as well.

However, Disaster-CDM provides querying for model checking. In the following query, SPARQL is used to list all *production cells* that are directly connected by a *channel*, together with the sub-models to which they belong:

```
SELECT ?cell1Name ?cell2Name ?subModel1 ?subModel2
WHERE {?cell1 a i2sim:production_cell.
?cell2 a i2sim:production_cell.
?cell1 simmodel:Name ?cell1Name.
?cell2 simmodel:Name ?cell2Name.
?channel simupper:hasStartNode ?cell1.
?channel simupper:hasEndNode ?cell2.
?cell1 i2sim:parentSystem ?subModel1.
?cell1 i2sim:parentSystem ?subModel2.}
```

Instead of each sub-model having to be checked manually, this query provides a unified view of all *production cells*, in all sub-models, that are directly connected by *channels*. The first few rows of the results of this query are displayed in Table I. Note that the names of the connected *production cells* in the first two rows are identical. However, the first row indicates a connection in the *steam_house-boiler_3* sub-model and the second row a connection in *steam_house-boiler_2*. The same query could have been executed against OWL ontology without storing the ontology in the database. However, disaster management deals with a large number of simulation models making the use of a database preferable to store ontologies as OWL files.

This section presented a simple case study that has been intended to illustrate how Disaster-CDM can be used in order to provide a high-available, scalable and low-cost solution to manage heterogeneity and semantics in large/complex simulation models to benefit disaster management through the use of the cloud environment.

V. CONCLUSIONS

In recent years, we have witnessed a number of extreme weather events and natural disasters. At the same time, changes in software and hardware have created opportunities for new solutions in disaster management.

This paper has proposed Disaster-CDM, a KaaS framework for disaster data management. Disaster-CDM stores large amounts of data while maintaining high availability by using NoSQL and cloud solutions. Search of disaster data, interoperability, and integration are facilitated through knowledge acquisition and knowledge delivery. Knowledge acquisition applies language processing, information extraction, and retrieval techniques to add structure and metadata to largely unstructured disaster data. Knowledge delivery services integrate information from different databases and deliver knowledge to consumers.

Disaster-CDM is still at the design stage, and only a part of the simulation model data acquisition process is included in this work. The remaining part of the framework will need to be implemented to evaluate the entire framework fully. In addition, security and privacy concerns must be addressed.

TABLE I. QUERY OUTPUT

cell1Name	cell2Name	subModel1	subModel2
Combustion chamber	Super heater	Steam_house-boiler_3	Steam_house-boiler_3
Combustion chamber	Super heater	Steam_house-boiler_2	Steam_house-boiler_2
Air fan	Super heater	Steam_house-boiler_3	Steam_house-boiler_3
Air fan	Super heater	Steam_house-boiler_4	Steam_house-boiler_4

REFERENCES

[1] V. Hristidis, S. Chen, T. Li, S. Luis, and Y. Deng, "Survey of Data Management and Analysis in Disaster Situations," Journal of Systems and Software, vol. 83, no. 10, pp. 1701-1714, 2010.

[2] S. Sakr, A. Liu, D.M. Batista, and M. Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments," IEEE Comm. Surveys & Tutorials, vol. 13, no. 3, pp. 311-336, 2011.

[3] D. Kossmann and T. Kraska, "Data Management in the Cloud: Promises, State-of-the-Art, and Open Questions," Datenbank-Spektrum, vol. 10, no. 3, pp. 121-129, 2010.

[4] R. Hecht, S. Jablonski, "NoSQL Evaluation: A Use Case Oriented Survey," Conf. on Cloud and Service Computing, pp. 336-341, 2011.

[5] L. Palen, K.M. Anderson, G. Mark, J. Martin, D. Sicker, M. Palmer, and D. Grunwald, "A Vision for Technology-Mediated Support for Public Participation and Assistance in Mass Emergencies & Disasters," Conf. on Visions of Computer Science, pp. 1-12, 2010.

[6] A. Schram, K.M. Anderson, "MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability," 3rd Conf. on Systems, Programming, and Applications: Software for Humanity, pp. 191-202, 2012.

[7] T. Silva, V. Wuwongse, and H.N. Sharma, "Linked Data in Disaster Mitigation and Preparedness," Third Int. Conf. on Intelligent Networking and Collaborative Systems, pp. 746-751, 2011.

[8] K.M. Anderson and A. Schram, "Design and Implementation of a Data Analytics Infrastructure in Support of Crisis Informatics Research: NIER Track," Proc. 33rd International Conference on Software Engineering, pp. 844-847, 2011.

[9] S. Khoshnevis and F. Rabeifa, "Toward Knowledge Management as a Service in Cloud-Based Environments," Int. Journal of Mechatronics, Electrical and Computer Technology, vol. 2, no. 4, pp. 88-110, 2012.

[10] I. Lai, S. Tam, and M. Chan, "Knowledge Cloud System for Network Collaboration: A Case Study in Medical Service Industry in China," Expert Syst.Appl., vol. 39, no. 15, pp. 12205-12212, 2012.

[11] Y. Qirui, "Kaas-Based Intelligent Service Model in Agricultural Expert System," Proc. 2nd International Conference on Consumer Electronics, Communications, and Networks, pp. 2678-2680, 2012.

[12] C.P. Sumathi, G.Gayathri Devi, T. Santhanam, "A Survey on Various Approaches to Text Extraction in Images," Int. Journal of Computer Science and Engineering Survey, vol. 3, no. 4, pp. 27-42, 2012.

[13] K. Grolinger, M.A.M. Capretz, J.R. Marti, and K.D. Srivastava, "Ontology-based Representation of Simulation Models," 24th Int. Conf. on Software Engineering and Knowledge Engineering, 2012.

[14] K. Grolinger, M.A.M. Capretz, A. Shypanski, and G.S. Gill, "Federated Critical Infrastructure Simulators: Towards Ontologies for Support of Collaboration," IEEE CCECE - Workshop on Connecting Engineering Applications and Disaster Management, 2011.

[15] M. Wang, B. Ni, X. Hua, and T. Chua, "Assistive Tagging: A Survey of Multimedia Tagging with Human-Computer Joint Exploration," ACM Computing Surveys, vol. 44, no. 4, pp. 1-24, 2012.

[16] J. Han, M. Song, J. Song, "A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing," Proc. 10th IEEE/ACIS Int. Conf. on Computer and Information Science, pp. 351-355, 2011.

[17] H.A. Rahman, M. Armstrong, D. Mao, and J.R. Marti, "I2Sim: A Matrix-Partition Based Framework for Critical Infrastructure Interdependencies Simulation," Electric Power Conf., pp. 1-8, 2008.

[18] W3C OWL Working Group, "OWL 2 Web Ontology Language," http://www.w3.org/TR/owl2-overview/, 2009.

[19] "Neo4j," http://www.neo4j.org/, 2013.

[20] R. Abdullah, Z. D. Eri, A. M. Talib, "A model of knowledge management system for facilitating knowledge as a service (KaaS) in cloud computing environment," Int. Conf. on Research and Innovation in Information Systems (ICRIIS), 2011 , vol., no., pp.1-4.

[21] N. C. Q. Lino, C. A. Siebra, M. Amaro, A. Tate, " EmergencyGrid – Planning in Convergence Environments", 22nd Int. Conf. on Automated Planning and Scheduling, SPARK workshop, 2012