Environmental Modelling and Software 116 (2019) 1–6

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Building complex event processing capability for intelligent environmental monitoring

Alexander Y. Sun[a,*], Zhi Zhong[a], Hoonyoung Jeong[b], Qian Yang[a]

[a] Bureau of Economic Geology, Jackson School of Geosciences, The University of Texas at Austin, Austin, TX, USA
[b] Department of Energy Resources Engineering, College of Engineering, Seoul National University, Gwanak-ro, Gwanak-gu, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Intelligent monitoring
Complex event processing
Machine learning
Anomaly detection
Geological carbon sequestration

ABSTRACT

Rapid evolution of Internet-of-Things is driving the increased deployment of smart sensors in environmental applications, contributing to many big data characteristics of environmental monitoring. Most of the current environmental monitoring systems are not designed to handle real-time datastreams, and the best practices for datastream processing and predictive analytics are yet to be established. This work presents a complex event processing (CEP) engine for detecting anomalies in real time, and demonstrates it using a series of real monitoring data from the geological carbon sequestration domain. We show that the service-based CEP engine is instrumental for enabling environmental intelligent monitoring systems to ingest heterogeneous datastreams with scalable performance. Our CEP framework requires minimal coding from the user and can be easily extended to other similar environmental monitoring applications.

1. Introduction

Intelligent monitoring is an integrative system management technology that combines real-time sensing with project-specific data processing, event detection, predictive analytics, and collaborative tools for data interpretation and decision making. Although the concept of intelligent monitoring has been around since the popularization of PC in 1990s (Bache et al., 1990; Sixsmith, 2000; Athanasiadis and Mitkas, 2004), recent years saw a surge of interests and applications, largely because of the rapid evolution of Internet-of-Things (IoT), the more accessible cyberinfrastructure, and advances in artificial intelligence. A recent report predicted that there will be between 25 and 50 billion connected devices by year 2025 (Manyika et al., 2015). Each smart device, capable of sensing its surrounding environment and sharing information across a network, becomes a potential data generator. Together these devices will shape the so-called big data economy, creating 4 V (volume, variety, velocity, veracity) information assets that demand timely processing for enhanced insight and decision making (Gandomi and Haider, 2015). The capability to collect, process, and analyze big data in real-time is still lacking in many fields.

Key requirements of an environmental intelligent monitoring system (IMS) are data wrangling (e.g., data extraction, transformation, and loading), event detection, and visualization. Many legacy environmental IMS platforms, however, are built on relational databases, requiring data to be first stored and indexed before they can be processed, creating a significant latency. The ubiquitous presence of smart devices and sensor networks is calling for a fundamental shift in design paradigm, from the client-server-based IMS design to cloud-based, or even edge-based design where the bulk of computing is done at the edge (e.g., smart monitoring devices) (Shi et al., 2016; Wong and Kerkez, 2016; Granell et al., 2016). Regardless of the platform, a fundamental task is related to processing the information flow continuously as they arrive, with or without data persistence.

Complex event processing (CEP) refers to data processing techniques that operate according to a set of predefined rules dictating how information flows should be processed and what new event streams should be produced as outputs (Cugola and Margara, 2012). Events can be thought of as single occurrences of a quantity of interest (e.g., higher than normal pressure readings), while complex events are distilled events corresponding to situations or patterns that comprise a particular meaning for the system (e.g., consecutive high pressure readings) (Luckham, 2002).

A CEP engine is a software system consisting of a suite of data processing algorithms and knowledge representation sets working in a distributed manner. CEP engines for datastream processing typically have state management, fault tolerance, and high performance features. Environmental monitoring is inherently stateful, requiring the CEP engine to keep track of the state of the system, including event arrival,

* Corresponding author.
E-mail address: alex.sun@beg.utexas.edu (A.Y. Sun).

https://doi.org/10.1016/j.envsoft.2019.02.015
Received 21 July 2018; Received in revised form 7 December 2018; Accepted 12 February 2019
Available online 18 February 2019

ingestion, and processing times (Castro Fernandez et al., 2013). To handle datastreams, a CEP engine needs to process a large amount of continuous data with low latency (performance) and, in case of system outrage, the engine needs be able to quickly restore (fault tolerance).

Currently, quite a few open-source and commercial CEP products are available. Under the Apache Software Foundation, there are more than a half dozen projects with different levels of maturity, such as Apache Kafka, Flink, Storm, and Samza. A comprehensive review of CEP engines (pre-2012) was provided by Cugola and Margara (2012), and more recent surveys on big data oriented CEP engines can be found in (Liu et al., 2014; Flouris et al., 2017; de Assuncao et al., 2018). So far, few environmental monitoring systems have tapped into the power of CEP (Granell et al., 2016). In this study, we focus on Apache Kafka, which was initially created and open-sourced by the social network company LinkedIn in 2011. Originally designed as a messaging queue middleware (i.e., software acting as mediators between applications or systems), Kafka has evolved into a high-performance, distributed streaming platform that provides three main functionalities: (a) publish and subscribe to real-time applications or topics, (b) store event streams or data records in a fault-tolerant way, and (c) process data as they occur (Apache Kafka, 2018). Kafka is now one of the most mature CEP engines in the open-source world. Recently, Apache Kafka was integrated by Confluent.io (https://www.confluent.io/product/confluent-open-source) into a real-time streaming processing ecosystem consisting of a large collection of infrastructure services such as database connectors and configuration managers.

Adaptation of CEP is domain specific, especially with regard to the notion of intelligence, which means the CEP engine needs to have a reasonable knowledge representation of its world, understand what is happening in terms of events, and know what reactions and processes it should invoke. In this work, we present a CEP engine for anomaly detection in geological carbon sequestration (GCS) projects. Carbon capture and storage is a geoengineering measure for reducing anthropogenic greenhouse gas emission to the atmosphere (Haszeldine, 2009; Bickle, 2009). Potential GCS repositories may include depleted oil & gas reservoirs and deep saline aquifers, all having leakage risks. The safe and efficient operation of GCS repositories thus requires integrated monitoring to track the injected carbon dioxide (CO_2) plume as it moves in a storage formation. Current GCS projects are data intensive, as a result of proliferation of digital instrumentation and smart technologies. The success of GCS thus depends in a large part on the monitoring system's capability to access, assimilate, and analyze heterogeneous data in a timely manner, and to provide high-level intelligent information to the operators. So far few GCS studies have attempted to integrate computing components in an online environment to support intelligent monitoring (Sun et al., 2018). The major contributions of this work are in (a) adapting a high-performance CEP engine for GCS monitoring, and (b) creating a streamlined intelligent monitoring workflow that requires minimal coding effort from domain users. All modules of our system are loosely coupled so that the system is flexible and expandable, and its components are replaceable when connecting to new monitoring devices or applications. As part of the demonstration, we showcase the system features using both scalar and vector data collected during a GCS field campaign.

2. Data and methods

2.1. System architecture

Fig. 1 shows the overall system design for the IMS, which consists of three layers, namely, the data layer, processing layer, and knowledge discovery layer. In the data layer, the input data types may include time series measurements of point values and vectors (multidimensional data), and model outputs (multidimensional data). We use a No-SQL (non relational) database InfluxDB (https://github.com/influxdata/influxdb) to store the monitoring data. As opposed to the traditional SQL databases using predefined database schema, No-SQL databases use dynamic schema and are best suited for applications requiring high-performance, flexibility, and scalability. The processing layer hosts the CEP engine. We use the Kafka ecosystem distributed by Confluent. The discovery layer is a web portal supporting collaborative visual analytics.

Kafka is designed around four key concepts: broker, topic, producer, and consumer. A Kafka topic provides a way of organizing messages, which in turn serves as intermediate data containers for records to be transmitted between applications/systems. The topic data schema is defined by the user for different sensor types. Internally, each topic is organized in a number of partitions for faster information retrieval and data redundancy. A Kafka producer writes to a topic, while a Kafka consumer reads from a partition. A Kafka broker is a hardware node in a distributed system that handles the actual reading and writing, and load balancing.

The user is responsible for defining producer(s) and consumer(s). Kafka provides application programming interfaces (API) for developers to create customized producers and consumers. In addition, the Kafka connectors allow configuration of sources/sinks that connect Kafka topics to known applications or data systems via standard interfaces such as JDBC (relational SQL databases), Hadoop Distributed File System (HDFS), and Amazon S3. Custom connectors are used to link the CEP to the data layer and knowledge discovery layer, automating the information exchange between CEP and those layers. We also developed producer and consumer templates for our use case such that the workflow can be easily adapted to different types of sensor datastreams.

Visualization is instrumental for assisting knowledge discovery and decision support, especially in environmental sciences (Laniak et al., 2013). A number of open-source data portals have appeared in business intelligence in recent years. After preliminary evaluation of several products, we choose the Apache Superset (https://superset.incubator.apache.org) because of its rich collection of data visualization tools (including maps), easy-to-use interface for uploading and transforming data, and seamless integration with commonly used data stores. We emphasize that the role of the event database shown in Fig. 1 is only for storing processed results, while all datastream processing is done in the CEP.

The service-oriented architecture presented in Fig. 1 is general. For demonstration, we deploy both Kafka (Confluent v4.0) and Superset (v0.24) on the same Ubuntu Linux system (v16.04) running on a Cloud-based virtual machine instance, which is hosted on a cluster node with Intel Xeon Haswell processor and 128 GB RAM. The Superset is served using the nginx (https://www.nginx.com) web server. All programming is done in Python.

2.2. Data and event processing

For the purpose of this work, event processing is related to detecting anomalies in sensor data. Algorithms for anomaly detection have long been studied in statistics and computer science. Surveys of general and conventional event detection algorithms are provided in (Chandola et al., 2009; Aggarwal, 2015). Surveys of machine learning algorithms for anomaly detection are provided in (Zohrevand et al., 2017). Challenges specific to geosciences are (a) data tends to vary both spatially and temporally, representing disparate scales and storage formats (Chen et al., 2014), (b) no single anomaly detection algorithm fits all purposes, (c) the “nominal model” or baseline is elusive in many situations, (d) anomalies may be shadowed by noise, (e) labeled anomaly data is rare, creating imbalance in the training data, and finally (f) establishing the causal mechanism (i.e., event attribution) can be challenging for subsurface processes.

2.2.1. Data

Data used in this study was collected from a series of field

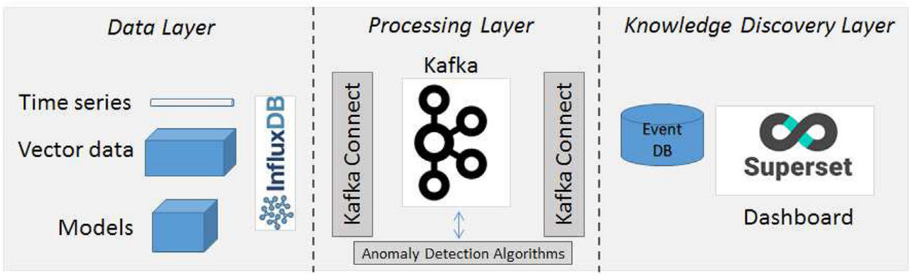

Fig. 1. System design of the IMS includes a data layer, a processing layer, and a knowledge discovery layer. Complex event processing engine is located in the processing layer. All layers are loosely coupled through Kafka connectors and web services. InfluxDB is used as a temporary datastream store, Confluent Kafka is used for event processing, and Apache Superset is used for data visualization.

experiments conducted in January 2015 at Cranfield, an active oil field located in Natchez, Mississippi, U.S. The original purpose of the experiments was to demonstrate a time-lapse, pressure-based leakage detection technique using modulated injection patterns. Three wells are located at the experimental site, a CO_2 injector (denoted as F1) and two monitoring wells (denoted as F2 and F3). The experiments consisted of two phases. In the first phase, the bottom-hole pressure and well casing temperature were monitored in the monitoring well (F2) to establish the base case. The raw pressure data was recorded every 2 s using a high-resolution downhole gauge (Ranger Gauge Systems, Sugar Land, Texas, USA). The raw distributed temperature sensing (DTS) data was collected using a fiber optic sensing device (Silixa Ltd, Houston, USA) in 10-min intervals. In the second phase, controlled CO_2 release tests were conducted in the adjacent well F3 to create leakage events while monitoring data was continuously acquired from F2. The bottom-hole distance between F1 and F2 is 60 m, between F1 and F3 it is 93 m, and between F2 and F3 it is 33.5 m. More details on the experimental setup and data collection methods are provided in Sun et al. (2016).

2.2.2. *Event processing methods*

The Cranfield data set provides a unique opportunity to demonstrate geospatial intelligent monitoring in real time. In particular, pressure data are analyzed using the IsolationForest (IFO) algorithm (Liu et al., 2008). Anomalies in time series may manifest as abrupt changes in signals or as shifts in the temporal trend. Traditional anomaly detection methods include both regression-based (e.g., Autoregressive Integrated Moving Average Model (ARIMA)) and classification-based methods (e.g., support vector machine (SVM)) (Aggarwal, 2015). Many of these traditional methods, however, are optimized to capture the normal data patterns, but not anomalies.

IFO is specially designed to detect anomalies. It is based on the premises that anomalies have attribute values that are very different from the rest of the data instances and that anomaly instances are relatively few. To isolate anomalies from a data set, IFO partitions data samples recursively using an ensemble of random trees. When a sample has anomalous attributes, the number of partitions required to isolate the data sample is smaller than that for a “normal” sample. In other words, anomalies are more susceptible to isolation under random partitioning. IFO calculates an anomaly score by averaging path lengths (equivalent to number of partitions) over all random trees. To help understanding, these main concepts of IFO are illustrated in the schematic plot in Fig. 2. The algorithm only requires two user parameters, namely, the number of trees to build and the subsample size. Subsampling is devised to alleviate the effect of masking (too many anomalies concealing their own presence) and swamping (normal instances located too close to anomalies), thus helping to build better trees more efficiently (Liu et al., 2008). Computational wise, IFO has a linear time complexity and a low memory requirement, and has the capacity to scale up to handle extremely large data size (Liu et al., 2008). We use the IFO function from the Python machine learning library, scikit-learn (Pedregosa et al., 2011).

DTS data were sampled every 12 cm along the well casing, resulting in a large number of data points per sampling time along the 3227 m total sampling depth. Visual inspection of the DTS data, shown in the

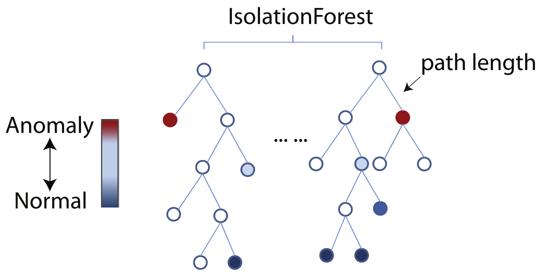

Fig. 2. The main idea behind the IsolationForest (IFO) algorithm is that anomalies are more susceptible to isolation under random partitioning.

Supporting Information (SI) Fig. S1, reveals that the data is highly correlated, both spatially and temporally. Thus, it makes sense to reduce the data dimension first. For this purpose, 50 sampling points in the 1000–2000 m interval are selected, with an average distance of about 20 m between consecutive points (see SI Section S1 for locations of selected points). Then a subspace anomaly detection method is applied on the 50 sampling points, for which the general idea is to determine a small set of latent variables in which the most important anomalies are revealed as quickly as possible (Aggarwal, 2015). IFO is mainly designed for processing single time series. Here we need to use a subspace anomaly detection algorithm that can operate on all selected time series, but in a reduced data space. The anomaly detector we adopted is based on the algorithm described in (Yin et al., 2012) and summarized in Algorithm S1 in SI. Briefly, the algorithm uses principal component analysis (PCA) to reduce the dimension of training data matrix, assuming that the nominal temperature profile can be effectively represented by using only a few principal components. The resulting principal components are used to calculate a test statistic, T^2, derived from the F-distribution. The threshold of T^2 defined for a certain significance level (e.g., 95%) is then applied online to detect anomalies on new data instances. The singular value decomposition (SVD) function from the NumPy library (http://www.numpy.org) is used to perform PCA, and the number of principal components retained is 3.

The trained pressure and DTS anomaly detectors are embedded in respective Kafka consumers to process monitoring data in real time. The resulting events are sent to the event database through Kafka producers. We comment that online anomaly detection in high-dimensional datastreams is still a challenging research topic, due to the fact that anomalies may often be buried in “small combinations of dimensions in a high dimensional data set” (Aggarwal, 2015). The most appropriate algorithm needs to be determined on a case-by-case basis, by incorporating domain insights.

3. Results

The base case pressure data are aggregated into 1-min intervals using InfluxDB web queries, which are then used to train and test the IFO model. Each row of the data matrix is a sliding window that includes information of pressure data and injection rate in a 90-min interval, corresponding to the duration of a full pulsing cycle used in the

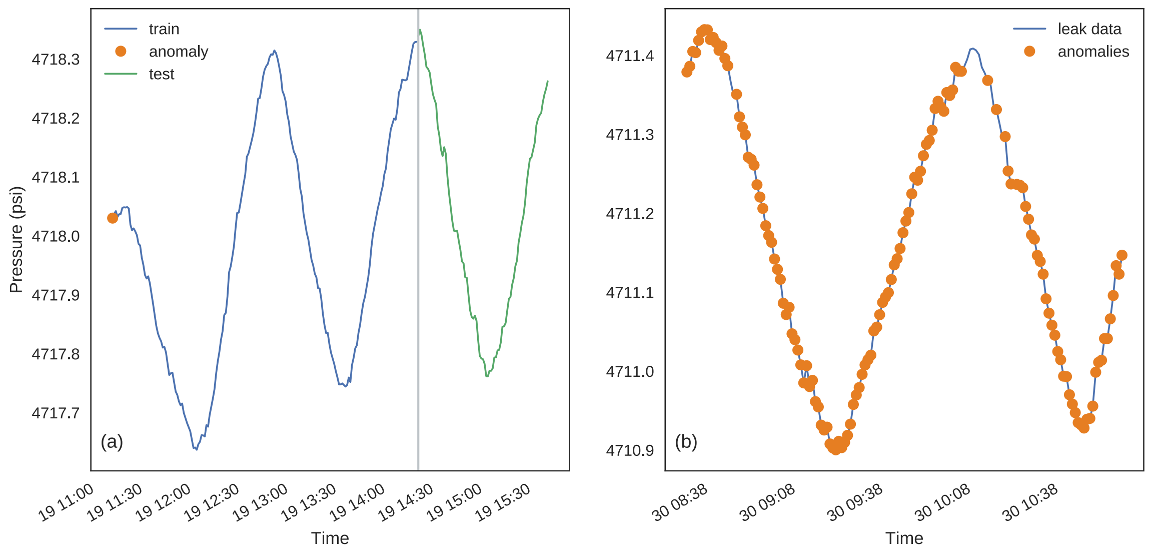

Fig. 3. Anomaly detection on pressure data using IsolationForest: (a) results on training and testing data using the base case data (Jan 19, 2015, 11:00–15:30), where the vertical gray line indicates the separation of training and testing periods; (b) results on controlled release data (Jan 30, 2015, 8:30–11:30). Anomalies are labeled with filled circles.

Cranfield experiments. The data matrix consists of a total of 360 sliding windows, (i.e., each sliding window is a shift of 1 min from the previous window), of which the first 70% are used for training and the rest for testing. The use of the full pulsing cycle duration as the width for sliding windows is critical for the machine learning algorithms to learn normal patterns that are related to injection rate changes only, so that pressure changes unrelated to injection rate changes (e.g., due to leakage) can be identified. The number of trees and subsample size are both set to 200 for the pressure data set.

Fig. 3 (a) shows the IFO training and testing results on the 90-min base case. The training period shows a single anomaly right at the beginning of the base case experiment, probably because of the initial perturbations. When the trained IFO model is applied to the controlled release data sequentially, the model correctly labels almost every data instance as anomaly (Fig. 3(b)). In comparison, Fig. S3 in SI shows the results of an SVM classifier, which has an overwhelmingly large false detection rate in this case. The main challenge in this case is that the base case and controlled release data have very similar sinusoidal temporal patterns, making it hard for SVM to distinguish normal data and anomalies.

The DTS results are given in Fig. 4. The entire DTS period has two major anomalies associated with controlled release events, as can be seen on the raw data plot in Fig. S1. During each release event, the warm CO_2 at reservoir temperature quickly rose along the wellbore and then started to absorb heat due to the Joule-Thompson expansion effect (Pruess, 2008). As a result, an abrupt disruption in the temperature profile can be observed. For training, an “eventless” period on Jan 24, 2015 is chosen, from which the T^2 threshold (see Algorithm S1 for definitions) is determined to be 8.4. After training the detector is applied in a sequential manner on each test sample and then compared to the threshold value. Fig. 4(b) shows that the T^2 statistic correctly identifies the temperature anomalies associated with the two controlled release events, as shown by the two spikes in T^2 values.

The two examples here illuminate the aforementioned challenges associated with anomaly detection in geosciences, namely, no single algorithm fits all purposes and a significant amount of prior knowledge and insight is required to develop customized event processors. Thus, the modular service-oriented design adopted in our case has a significant advantage.

The Apache Superset visualization platform offers highly customizable dashboards for decision makers. It can be set up to provide a holistic view of the project under monitoring. The Cranfield dashboard is shown in Fig. 5, which shows a map of site location, the raw pressure and DTS time series (from 5 different depths), and the event status reported by the CEP.

4. Conclusion

The need for real-time analysis will continue to push the development of low-latency, real-time complex event processing (CEP) engines in the IoT era (Gartner, 2017). Recent advances in big data analytics and distributed computing provide new abstractions to deal with complex data, and simplify programming of scalable and parallel systems. Intelligent environment monitoring, as a subdiscipline of Environmental Data Science (Gibert et al., 2018), needs to adapt to the ever-increasing speed of new data generation, and leverage the data in time to create new services and intelligent information to maximize the value of information.

In this work, we develop and demonstrate a CEP workflow for detecting anomalies in real GCS monitoring data. We show that (a) problem-specific online machine learning algorithms need to be carefully selected and trained to achieve robust performance in real time; (b) the microservice-oriented, distributed architecture is instrumental for scaling up computing systems to deal with syntactic and semantic heterogeneity, and (3) the use of highly interactive, easy-to-use web interfaces should be an integral component of intelligent monitoring system (IMS) development because they enable non-programmer domain users to stay in the loop, as well as to have easy access of the information and knowledge generated by the CEP.

Our case study is limited to a single site with high temporal frequency, structured data. CEP involving many monitoring sites and/or unstructured data is still a challenge for environmental IMS. While large volumes of data can be handled by horizontal scaling (i.e., adding more nodes), processing of large arrays may still be required (e.g., in the case of PCA) (Wu et al., 2018). Future IMS may require combining both distributed computing and edge computing to further improve system performance.

Software availability

The web system created in this work is hosted at http://129.114.110.45/login. Please contact the corresponding author for login authorization.

Programming language: Python.

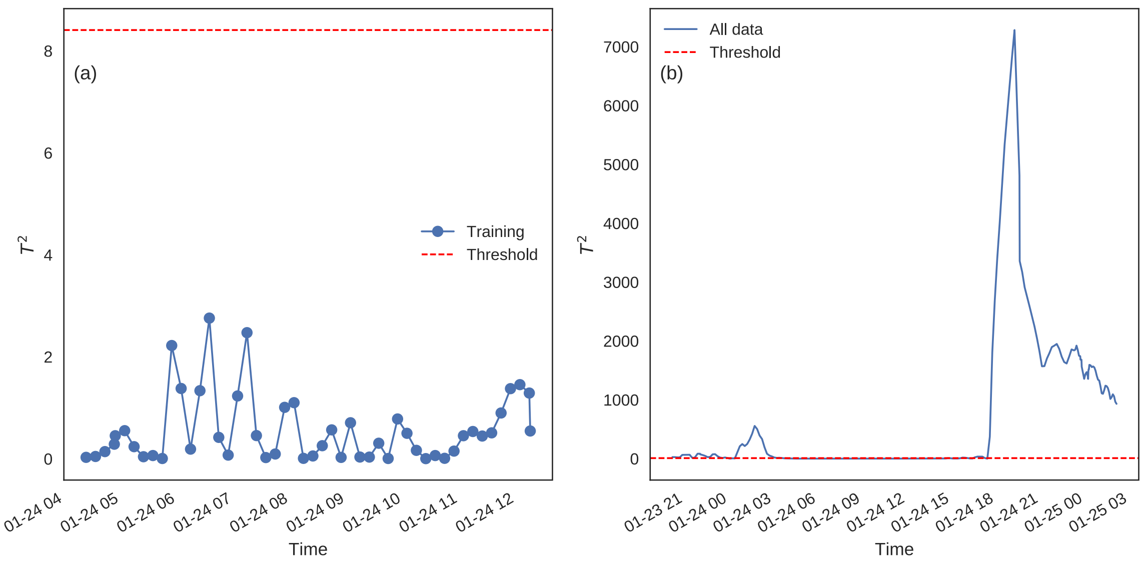

Fig. 4. DTS anomaly detection using T^2 statistic based on principal component analysis. Results of (a) training and (b) full DTS data set. The threshold is given by the horizontal bar (dashed line).

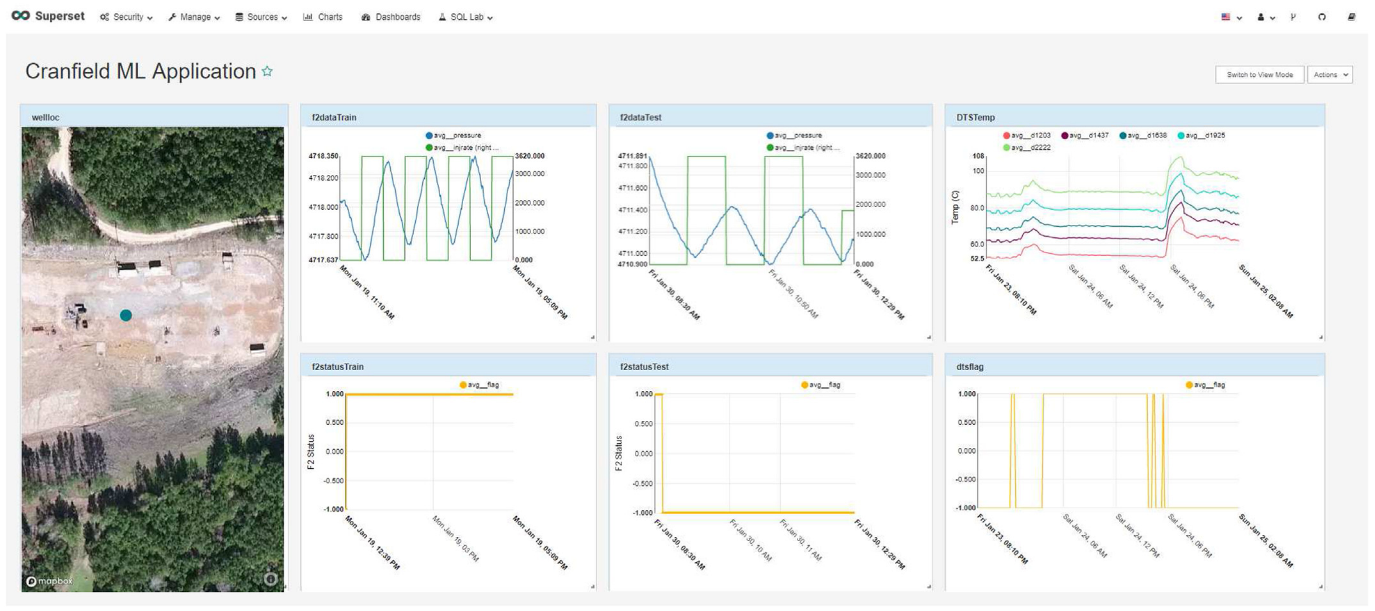

Fig. 5. Cranfield data monitoring portal. Left panel: the site map; mid 4 panels, time series plots of pressure monitoring well (F2) and the F2 status, with 1 indicating normal and -1 indicating anomaly; right panel, DTS temperature data and anomaly indicator. In real time, the dashboard automatically pulls the event database to update time series plots. The event database in turn is updated by a Kafka connector.

Acknowledgements

The work was supported by the U.S. Department of Energy, National Energy Technology Laboratory under grant number DE-FE0026515. We are grateful to the Texas Advanced Computer Center at UT Austin for providing cloud resources through their Chameleon Cloud project. We thank the associate editor and two anonymous reviewers for their constructive comments.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.envsoft.2019.02.015.

References

Aggarwal, C.C., 2015. Outlier Analysis. Springer.
Apache Kafka, 2018. User Documentation. https://kafka.apache.org/documentation.
Athanasiadis, I.N., Mitkas, P.A., 2004. An agent-based intelligent environmental monitoring system. Manag. Environ. Qual. Int. J. 15 (3), 238–249.
Bache, T.C., Bratt, S.R., Wang, J., Fung, R.M., Kobryn, C., Given, J.W., 1990. The intelligent monitoring system. Bull. Seismol. Soc. Am. 80 (6B), 1833–1851.
Bickle, M.J., 2009. Geological carbon storage. Nat. Geosci. 2 (12), 815.
Castro Fernandez, R., Migliavacca, M., Kalyvianaki, E., Pietzuch, P., 2013. Integrating scale out and fault tolerance in stream processing using operator state management. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, pp. 725–736.
Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: a survey. ACM Comput. Surv. 41 (3), 15.
Chen, N., Wang, K., Xiao, C., Gong, J., 2014. A heterogeneous sensor web node meta-model for the management of a flood monitoring system. Environ. Model. Softw 54, 222–237.
Cugola, G., Margara, A., 2012. Processing flows of information: from data stream to complex event processing. ACM Comput. Surv. 44 (3), 15.

de Assuncao, M.D., da Silva Veith, A., Buyya, R., 2018. Distributed data stream processing and edge computing: a survey on resource elasticity and future directions. J. Netw. Comput. Appl. 103, 1–17.

Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., Mock, M., 2017. Issues in complex event processing: status and prospects in the big data era. J. Syst. Software 127, 217–236.

Gandomi, A., Haider, M., 2015. Beyond the hype: big data concepts, methods, and analytics. Int. J. Inf. Manag. 35 (2), 137–144.

Gartner, 2017. https://www.gartner.com/doc/reprints?id=1-4gh0ftl&ct=171004&st=sb, accessed july 14 2018.

Gibert, K., Horsburgh, J.S., Athanasiadis, I.N., Holmes, G., 2018. Environmental data science. Environ. Model. Softw 106, 4–12.

Granell, C., Havlik, D., Schade, S., Sabeur, Z., Delaney, C., Pielorz, J., Usländer, T., Mazzetti, P., Schleidt, K., Kobernus, M., et al., 2016. Future internet technologies for environmental applications. Environ. Model. Softw 78, 1–15.

Haszeldine, R.S., 2009. Carbon capture and storage: how green can black be? Science 325 (5948), 1647–1652.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., et al., 2013. Integrated environmental modeling: a vision and roadmap for the future. Environ. Model. Softw 39, 3–23.

Liu, F.T., Ting, K.M., Zhou, Z.-H., 2008. Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. IEEE, pp. 413–422.

Liu, X., Iftikhar, N., Xie, X., 2014. Survey of real-time processing systems for big data. In: Proceedings of the 18th International Database Engineering & Applications Symposium. ACM, pp. 356–361.

Luckham, D., 2002. The Power of Events, vol. 204 Addison-Wesley, Reading.

Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J., Aharon, D., Jun. 2015. The Internet of Things: Mapping the Value Be-Yond Hype. McKinsey Global Institute.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pruess, K., 2008. Leakage of co2 from geologic storage: role of secondary accumulation at shallow depth. International Journal of Greenhouse Gas Control 2 (1), 37–46.

Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: vision and challenges. IEEE Internet of Things Journal 3 (5), 637–646.

Sixsmith, A., 2000. An evaluation of an intelligent home monitoring system. J. Telemed. Telecare 6 (2), 63–72.

Sun, A.Y., Jeong, H., González-Nicolás, A., Templeton, T.C., 2018. Metamodeling-based approach for risk assessment and cost estimation: application to geological carbon sequestration planning. Comput. Geosci. 113, 70–80.

Sun, A.Y., Lu, J., Freifeld, B.M., Hovorka, S.D., Islam, A., 2016. Using pulse testing for leakage detection in carbon storage reservoirs: a field demonstration. International Journal of Greenhouse Gas Control 46, 215–227.

Wong, B.P., Kerkez, B., 2016. Real-time environmental sensor data: an application to water quality using web services. Environ. Model. Softw 84, 505–517.

Wu, S.X., Wai, H.-T., Li, L., Scaglione, A., 2018. A review of distributed algorithms for principal component analysis. Proc. IEEE 106 (8), 1321–1340.

Yin, S., Ding, S.X., Haghani, A., Hao, H., Zhang, P., 2012. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee eastman process. J. Process Contr. 22 (9), 1567–1581.

Zohrevand, Z., Glässer, U., Tayebi, M.A., Shahir, H.Y., Shirmaleki, M., Shahir, A.Y., 2017. Deep learning based forecasting of critical infrastructure data. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, pp. 1129–1138.