CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2017; **00**:1–24
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

A Framework for Scalable Real-Time Anomaly Detection over Voluminous, Geospatial Data Streams

Walid Budgaga, Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara

Computer Science Department, Colorado State University, Fort Collins, CO, USA

SUMMARY

This study presents a framework to enable distributed detection, storage, and analysis of anomalies in voluminous data streams. Individual observations within these streams are multidimensional, with each dimension corresponding to a feature of interest. We consider time-series geospatial datasets generated by remote and in situ observational devices. Three aspects make this problem particularly challenging: (1) the cumulative volume and rates of data arrivals, (2) evolution of the datasets over time, and (3) spatiotemporal correlations associated with the data. Further, solutions must minimize user intervention and be amenable to distributed processing to ensure scalability.

Our approach achieves accurate, high-throughput classifications in real time, which we demonstrate with our reference anomaly detector implementations. We also provide interfaces that allow new implementations to be developed and parallelized automatically, ensuring applicability across problem domains. To help quantify the magnitude of anomalous observations, detector implementations provide a corresponding *degree of irregularity*. We have incorporated these algorithms into our distributed storage platform, Galileo, and profiled their suitability through empirical analysis that demonstrates high throughput (10,000 observations per-second, per-node) on a real-world Petabyte dataset.

Copyright © 2017 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Distributed Geospatial Anomaly Detection, Time Series Analytics, Online Anomaly Detection, Spatiotemporal Data Streams

1. INTRODUCTION

The focus of this study is twofold: (1) providing a distributed framework for detection and classification of anomalies in spatiotemporal data streams, and (2) supporting analytics activities that follow the discovery of an anomaly (including visualizations). In the context of this work, an *anomaly* may constitute an irregular event, inconsistent sensor readings, or other types of situations that result in data points outside of the expected norm. We leverage individual anomaly detector implementations to determine what data should be classified as anomalous for a given problem domain. Each *observation* comprises n-dimensional tuples with each dimension representing a feature of interest. Examples of such features include temperature, air pressure, humidity, etc. Features may also have linear or non-linear relationships with each other; for instance, there may be a relationship between temperature and precipitation at certain geographic locations. Ultimately, these relationships result in a classification by our framework as either *normal* or *anomalous* with a corresponding *degree of irregularity*.

The datasets we consider are composed of streams that continually report readings from observational devices. Some of these observational devices, such as radars and satellites, can

*Correspondence to: 1873 Campus Delivery, Fort Collins, CO 80523-1873, USA. Email: malensek@cs.colostate.edu

Copyright © 2017 John Wiley & Sons, Ltd.

remotely sense features of interest while other features may require in situ measurements by devices such as piezometers and barometers. The measurements are reported as observations in discrete packets that comprise a stream.

Anomaly detection is a precursor to the discovery of impending problems or features of interest. Timely detection of anomalies is critical in several settings. Often such detection needs to be made in real time to be able to detect potential emergencies. Our specific problem relates to voluminous data streams and anomaly detection that accounts for evolution of the feature space over time. Also, given the rate of data arrivals and the volumes involved, human intervention is rendered infeasible. This work is applicable in domains where observations have geospatial and chronological components, including atmospheric sciences, meteorology, environmental and ecological modeling, epidemiology, and traffic monitoring.

The range of values that each feature takes on may be rather large, and simple checks for breaching upper and lower bounds are generally not viable. Other dimensions such as location and time may determine whether a particular feature value is considered anomalous. For example, temperatures at night are often lower than those in the day for a particular location. Also, in the case of geospatial data, what is considered normal will vary by region and anomaly classifications must account for this as well.

Given the data volumes involved, observations must be stored over a collection of resources. However, disk I/O operations should also be reduced so as to not preclude real-time classification. In-memory data structures must be compact to avoid page-faults and thrashing. To cope with scaling issues, viable solutions must be able to take advantage of increases in the number of resources available to the system.

1.1. Research Challenges

There are several challenges involved with the proposed research:

1. Streams have no preset lifetimes, and readings arrive continually. This makes it infeasible to inspect all previous records when making a classification.
2. Observations are multidimensional. Individually, feature values (i.e. values along a dimension) may be normal, but when collectively accounting for all the dimensions, the tuple may be anomalous.
3. There are spatial and chronological dimensions associated with feature values. How features evolve is spatiotemporally correlated. What is considered normal for a particular geographical extent would be considered anomalous for another. Anomaly classifications must take these into account.
4. What is considered anomalous evolves over time. A good exemplar of this is temperature readings. Over the past several years, average temperatures at various geographic locations have trended higher overall but fluctuate from year to year.
5. The combination of legitimate feature values is very large. Building a single model of what constitutes anomalous data is infeasible and impractical.
6. Anomaly detection must be done in real time despite the constant feature space evolution and data volumes involved.

1.2. Research Questions

In this paper we explore the following research questions:

RQ-1 How can we account for spatiotemporal properties associated with features, and what are the implications for system design in such cases? (§2.3, §3.4)

RQ-2 How can we achieve timeliness when detecting anomalies in voluminous data streams while also ensuring accuracy and a reasonable balance of load? (§2.4, §3.2, §3.3)

RQ-3 What features are required in the anomaly detector interfaces to make them generalizable to a number of problem domains? (§3)

RQ-4 Given that human intervention is infeasible in this work, how can we allow anomaly detector implementations to adapt autonomously to changes in the data? (§4)

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

1.3. Approach Summary

Our approach provides a configurable framework for anomaly detection and analysis over continuous data streams. Since anomalies evolve over time, the framework addresses online adaptation of models in its anomaly detector interfaces. Our design allows for domain-specific behavior to handle changes in the data that must be treated as normal rather than anomalous, a feature that can be toggled at run time.

To deal with spatial properties in the dataset, we partition incoming streams based on geographical extents. We then initialize an instance of our anomaly detection model for the geographic regions, each of which continually adapts based on observations recorded for the region. Depending on the data volumes involved and the number of resources available within the system, both geographic boundaries and the number of model instances can be tuned appropriately. Anomaly detector instances and data routing/partitioning are managed by Galileo, our distributed storage framework [1, 2]. Galileo provides support for multidimensional, time-series geospatial datasets, and is organized as a distributed hash table (DHT). Each Galileo *storage node* manages a geographic subset of the incoming data streams and passes the information on to the anomaly detectors. This ensures that our framework is decentralized, scalable, and capable of achieving high throughput. We have also augmented the indexing structures in Galileo to provide efficient lookup and analysis capabilities for anomalous observations and any related data points.

Our framework provides a mechanism for incorporating different anomaly detection algorithms, and allows them to scale and be specific (based on temporal or spatial attributes). Scaling out is supported by allowing multiple model instances rather than an all-encompassing model. Specificity is achieved by restricting training and classification data for model instances to a particular geographical scope. In our reference anomaly detector implementations, we have incorporated support for multiple detection methods: ***density***, ***distance***, ***Bayesian***, and ***ensemble*** based approaches. We chose this wide range of anomaly detection techniques to illustrate the extensibility of our framework. However, we do not advocate for any particular approach over another.

In our density based implementation, we use Expectation Maximization (EM) to build Gaussian Mixture Models (GMMs) that model the densities of the training data by using different combinations of Gaussian distributions. EM is an iterative method that aims to maximize log-likelihood by modifying GMM parameters. Within each iteration, the log-likelihood is improved by fitting Gaussian distributions to the given data. The iterative process is stopped when no significant improvement can be achieved. For brevity, this method is referred to as "EM" throughout the text.

In our distance based approach, we rely on k-means to cluster observations in a multidimensional space. For each instance, we use canopy clustering to compute the initial number of clusters. We explore the use of two distance measures: the commonly-used Euclidean distance, and the Mahalanobis distance [3], which accounts for correlation between the dimensions (features). For brevity, this approach is referred to as "K-Means" in the text.

Our Bayesian approach employs Dirichlet Process Mixture Models (DPMMs) to generate clusters over observations without requiring parameterization [4]. DPMMs do not require specification of the number of underlying clusters and instead assume that the observations were derived from an infinite mixture of distributions. In this approach, clusters adapt dynamically to changes in the underlying data stream. As a result, a small number of data points being assigned to a particular cluster may indicate an anomaly. We refer to this approach as "DPMM" throughout the text.

In our ensemble approach, we use *Isolation Forests* (also known as iForests) [5] to create an ensemble of randomized decision trees. In this algorithm, trees with the shortest paths are most likely to represent anomalies. Compared to many other approaches, isolation forests are unique in that they separate anomalies from the rest of the dataset without needing to profile the norm. Throughout the text, we refer to this approach as "iForest" for brevity.

We provide support for continuous adaptation for each of our approaches. Determination of whether an observation is anomalous or not does not require specification of thresholds by the user. Observations are tagged as normal or anomalous based on their fitness to the underlying models and their locations within a d-dimensional space. Two factors determine whether an observation

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

is anomalous or not: *fitness scores* and *distance measures*. The fitness score represents how well an observation fits the current model, and the distance measure is calculated based on the distance between the observation and the cluster centroid to which the observation was assigned. These factors are also used to produce degree of irregularity scores that describe the magnitude of the anomalies.

To aid in analysis of events leading up to an anomalous observation, we provide several distributed query primitives. These queries can be used to determine what states led up to the anomaly, how detector instances in other geographic regions would classify the observation, and the geographic impact of various events. Distributed query results can be expressed as raw data points, a traversable graph, or graphical overlays on a map. Additionally, these queries can be leveraged by anomaly detector instances to submit observations to other, similar spatial regions for evaluation or to gain confidence in weak classifications by consulting neighboring detectors.

Each instance of the anomaly detector tunes itself autonomously based on the data distributions it has observed. The instances are executed in a thread pool, which allows us to calibrate the degree of concurrency in the system to better exploit available cores and execution pipelines. Our performance benchmarks show that each node in the system can evaluate more than 10,000 data stream packets per second.

1.4. Paper Contributions

This paper describes our approach for detecting and analyzing anomalies in voluminous multidimensional datasets. The work presented herein includes the following contributions:

1. Our approach scales with increases in data volumes and the number of machines available.
2. We provide a general interface for implementing and parallelizing anomaly detectors in a variety of problem domains, with support for *degree of irregularity* scores.
3. We can fine-tune the specificity of classifications by controlling the geographical scope associated with anomaly detection models.
4. Detector instances are able to come to a consensus on anomalous observations with distributed queries rather than needing explicit communication and coordination.
5. We demonstrate how a variety of clustering methods and anomaly detection techniques can be incorporated at scale.
6. Since model instances tune themselves autonomously based on the data available to them, we can cope with variability in the density and availability of readings from different geospatial locations.

Our empirical evaluations with diverse datasets and across different volumes of information confirm the suitability of our approach. Our benchmarks also demonstrate that we can perform these classifications in real time, processing 10,000 observations per second at each node.

1.5. Paper Organization

This paper is organized as follows. The next section provides background information on our distributed storage framework, followed by Section 3 that outlines the anomaly detection framework. Section 4 describes our reference anomaly detection implementations, and Section 5 discusses analysis and visualization features supported by the system. Section 6 provides a performance evaluation and experimental results. Finally, Section 7 reviews related work from the literature and Section 8 concludes the paper.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

2. DISTRIBUTED STORAGE FRAMEWORK

Our distributed storage system, Galileo, is responsible for managing incoming observations. Galileo offers spatiotemporal partitioning functionality, distributed indexing and query support, and ensures our approach can scale up as resources are added. In this study we added several anomaly detection features to Galileo, including support for visualization. However, it is worth noting that the components in our framework are loosely coupled and could be used with a variety of distributed storage systems, such as HBase (see Section 6.4).

2.1. Galileo

Galileo is a high-throughput distributed storage framework designed for managing multidimensional data. The system's network design is modeled as a hierarchical distributed hash table (DHT), which allows incremental assimilation of storage resources and the use of multi-tiered hash functions to enable development of novel partitioning schemes. By focusing on spatiotemporal datasets, Galileo provides functionality that is generally not provided by standard DHTs, such as expressive query support [2], time series analysis capabilities, and polygon- or proximity-based geospatial retrieval functions [6]. Galileo is decentralized and composed of a network of *storage nodes* that facilitate data management.

2.2. Storage Nodes

Each storage node in Galileo manages a single instance of our anomaly detection framework. Based on the classification output by the framework, the storage node can take appropriate action (which may vary across problem domains). The storage node does not need to wait for the result of each evaluated observation during the storage process, and will assume the observations are normal until informed otherwise.

The storage node treats the anomaly detection framework as a black box, but can control some of its behavior. Configurable parameters include enabling or disabling adaptive classifications, as well as how fast adaptations should be made. The storage node can also modify the size of the geospatial area that will be assigned to each detector instance, which is critical in situations where the geographic scope of the node changes due to fluctuations in the underlying resource pool.

2.3. Metadata Management and Information Retrieval

As data is streamed into the system, Galileo maintains a hierarchical *metadata graph* instance at each storage node. Metadata graphs are memory-resident data structures that index multidimensional data points and share some functionality with k-d trees [7] and tries [8]. Traversing through the graph hierarchy reduces the overall search space, which can be accomplished by interacting directly with the graph nodes or by issuing queries in an SQL-like syntax that supports range-based selections and join operators across distributed graphs. Leaf nodes represent on-disk data points, called *file blocks*, which are labeled by our anomaly detector framework as anomalous with a corresponding *degree of irregularity*. To ensure the metadata graph can fit in main memory, vertices in the graph may be responsible for ranges of values called *tick marks*. These ranges are augmented with incrementally-updating summary statistics that include the min, max, mean, and standard deviation of the data points stored under each vertex. Figure 1 contains a simplified representation of a metadata graph at one of the storage nodes in Galileo. Once a user has selected relevant portions of the overall graph, *subgraphs* can be used to launch locality-aware MapReduce computations on the cluster [9]. In the context of our anomaly detection framework, a distributed computation may include additional analysis on data points that have been labeled as anomalous.

2.4. Geospatial Data Partitioning

Data partitioning in Galileo is done based on the observed Geohash prefixes of incoming records [10]. The Geohash algorithm is a geocoding scheme that divides the earth into a hierarchy of spatial bounding boxes referenced by Base-32 strings. Figure 2 provides an example of how the Geohash

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

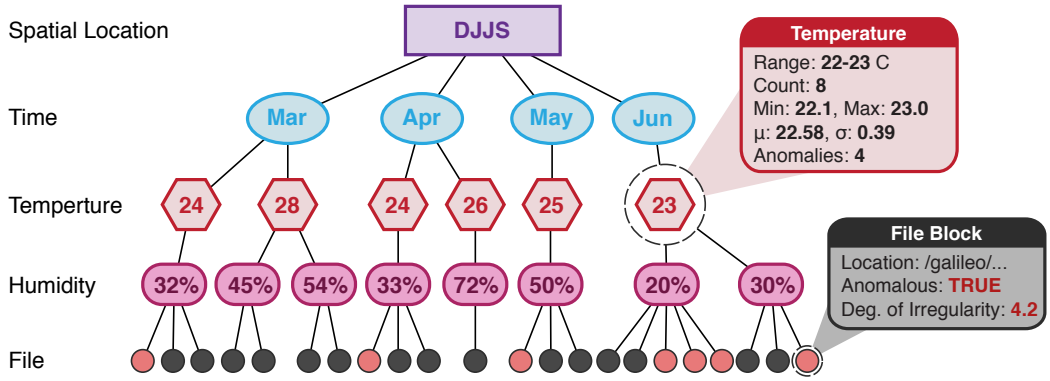

Figure 1. A simplified metadata graph hierarchy with our feature additions for anomaly detection. As more query parameters are specified, the graph is traversed to find matching *file blocks*. Note that each vertex in the graph manages a range of values, as well as statistics about the blocks under its purview. An *anomaly count* is provided at each vertex to summarize anomalous readings throughout the hierarchy; at the root node, this count represents the total number of anomalies observed by the storage node.

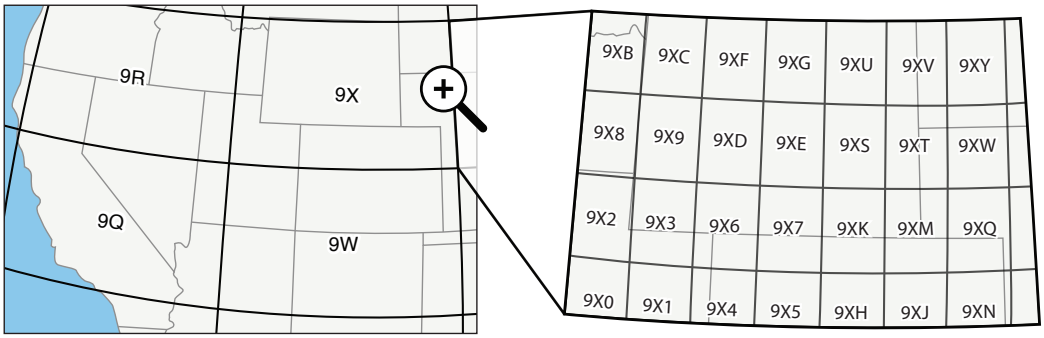

Figure 2. A visual overview of the Geohash geocoding scheme showing two hierarchical divisions of the western United States.

algorithm works. For example, the latitude and longitude of the coordinates N 28.8927, W 81.9796 would map to the Geohash string *djjsqeb2*. Longer strings result in higher precision coordinates, and two Geohash strings with the same prefixes will be located in a similar geographic region; a Geohash of *djjs* would describe a broader area that also encompasses the aforementioned coordinates. These prefixes ultimately determine where data records will be routed, processed, and stored.

The prefix length used for partitioning in Galileo is configured based on the size and scope of the particular deployment. The default length, four characters, assigns incoming records to 39.1 km x 19.5 km regions. This naturally divides a broad geographic region into manageable pieces that can be maintained by individual nodes in the system, and also allows future reconfiguration to scale up and down to meet changing problem requirements. This scheme also enables data partitioning to be carried out in a decentralized manner while facilitating parallel computations.

Classification accuracy is also impacted by how data is partitioned across nodes in the system. If a small number of partitions are used, classifications must be made across a broader range of geography and ambient conditions, whereas a fine-grained partition allows for similarly fine-grained classifications. Data is partitioned once again at each node using a longer (finer-grained) Geohash string to divide the workload up among anomaly detector instances, making analysis across a hierarchy of spatial regions possible.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

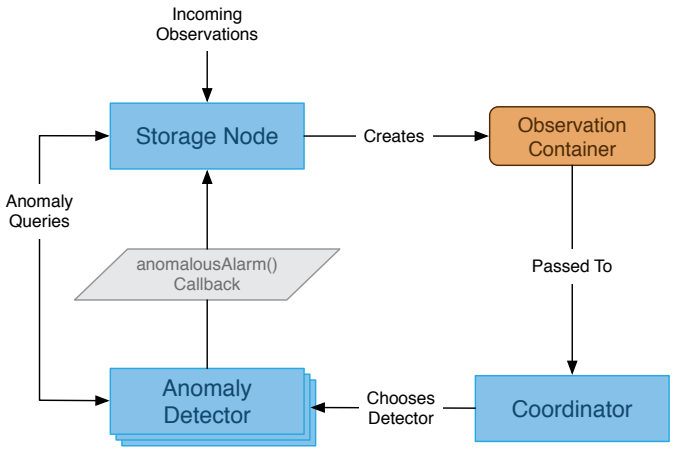

Figure 3. The anomaly detection processing cycle.

3. ANOMALY DETECTION FRAMEWORK

Our design simplifies integration of multiple anomaly detection algorithms. The framework assists these algorithms by controlling the amount and type of observations that are made visible to them. Rather than having an all-encompassing instance of the anomaly detection model, our approach allows multiple model instances to be dispersed over the collection of storage nodes; each model instance is responsible for a particular geospatial scope, allowing for specificity in the classification of anomalies. This also assists algorithms in accounting for spatial correlations between features since each model instance learns from the data in a particular geographical region. The use of multiple model instances ensures: (1) scalability: as the number of storage nodes increase, the number of model instances scales as well, with each responsible for a specific geographical scope, and (2) high throughput: there are no bottlenecks since observations do not need to be funneled through an all-encompassing model. Incidentally, properties (1) and (2) contribute to faster turnaround times for classifications. Finally, our approach allows multiple algorithms to operate on the same set of observations for a geographical scope. Such combinations allow us to exploit the properties of different techniques to minimize mis-classifications – specifically, observations classified as anomalous by multiple algorithms are likely to be truly anomalous, but a particular algorithm may capture subtle variations that were missed by the others.

3.1. Storage Node Integration

Each distributed storage node manages a single instance of the detection framework, which contains a single *coordinator* process and multiple *detector* instances. As observations are streamed into the system, the storage node forwards them to the appropriate anomaly detectors based on their geographical location. Figure 3 depicts how events are passed between each of the components; observations sent from the coordinator to the anomaly detectors are queued and classified in order. The detector tags observations that are classified as anomalous by triggering a callback, *anomalousAlarm*. This method can be overridden to perform post-processing steps, which generally includes flushing the identifiers of the anomalous observations to disk for further analysis.

To ensure high throughput and minimize I/O, data management tasks in our framework are performed asynchronously. Upon receipt of a new observation, the storage node updates its in-memory metadata graph and generates a unique file identifier for the data. Once this step is complete, the observation is encapsulated in an *observation container* and passed to the anomaly detection coordinator for processing. Note that no disk I/O has occurred up until this step; only after the anomaly detection process begins is the observation placed in a queue for on-disk storage. This approach helps interleave I/O and processing operations on separate threads. Further, when an

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

observation is classified as anomalous and assigned a degree of irregularity, the metadata graph is updated without requiring any changes to files stored on disk.

3.2. The Coordinator

The primary task of the *coordinator* is creating and managing anomaly detector instances. The coordinator receives observations from the node and then forwards them to the appropriate detector instances based on the spatial partitioning scheme in use. If a detector instance for the region does not exist, the coordinator creates a new instance. The coordinator uses an abstract class called *AnomalyDetector* to load and manage detectors that use different detection algorithms. This makes it possible to add, manage, and combine different anomaly detection techniques concurrently. Also, because of the loose coupling between the coordinator and detectors, any changes made to either component will not require corresponding changes in the other.

Our approach ensures available cores and execution pipelines are used efficiently by managing a thread pool to facilitate parallel detection activities. During initialization, the coordinator creates a thread pool of a configurable size and provides its reference to the anomaly detectors. Detector instances submit *classification tasks* that contain a queue of incoming observations to be processed concurrently, and the queues can be updated as more observations are assimilated.

3.3. Anomaly Detector

The primary concern of the *anomaly detector* is to collect training data from the coordinator, build models for finer-grained geospatial scopes, and then use the appropriate model to detect observations whose behaviors are outside the observed norm. Collecting the training data and training a model is done automatically for each detector regardless of the actual implementation of the anomaly detector instances. Each anomaly detector starts in the *data collection* phase where it collects observations in memory and transitions to the *training* phase when the amount of data collected reaches a configurable threshold. The coordinator can also override the threshold to begin training immediately, if the particular problem warrants such an action. In the training phase, a training task will be created and queued to the thread pool. While the training task is running in a separate thread, observations are buffered for classification until the training process is complete. Finally, in the *classification* stage, the models are used to classify incoming data.

Our framework allows anomalies to be detected in an online manner, enabling evaluations to occur on the fly without retraining the underlying models. Further, the detectors can adapt to changes that occur over time. For example, temperature values at a particular region that were unusual thirty years ago may be considered normal now. In such a case, detector implementations are given the opportunity to adapt to these changes and not tag such data as anomalous. However, this feature can be disabled to handle situations where adaptation is not beneficial; for instance, models dealing with heart rate measurements do not need to be adapted because valid measurements do not change over time. When an anomaly is detected, implementations can include a *degree of irregularity* to demonstrate the relative magnitude of the anomaly. Finally, the detector can serialize and deserialize its model to and from disk to allow migration to other systems and to cope with failures or planned outages.

3.4. Anomaly Queries

In some problem domains (especially spatial applications), it is useful for anomaly detectors to communicate and come to a consensus on their classifications. Rather than requiring anomaly detector implementations to coordinate amongst themselves manually, we provide a set of SQL-like query interfaces from Galileo that allow detectors to build confidence in their classifications without knowledge of the underlying distributed system. In essence, this enables detectors to be queried in a fashion similar to observations and feature values.

During classification operations, *anomaly queries* can be submitted to other detector instances that match a set of criteria. For example, a detector managing observations in Oregon, USA may submit a questionable observation to all other instances in the same state and use the resulting classifications

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

to make its final decision. In some cases these queries will be resolved on the same physical machine as the initiating anomaly detector, but submitting the requests and retrieving results proceeds in the same fashion regardless of the geographic scope of the query.

Since administrative boundaries often do not take geography into account, anomaly queries can be used to find regions with similar characteristics; for instance, a detector in a region with high altitudes and low humidity may constrain its anomaly queries to similar detector instances. To ease this type of search, we provide *feature similarity* queries that will inspect the underlying distributions of the observations at each storage node and return a set of candidate detector instances that can be used to confirm anomalous data points. With feature similarity queries, all the usual query operations are supported to further restrict the scope of the search.

4. REFERENCE ANOMALY DETECTOR IMPLEMENTATIONS

We designed our reference anomaly detectors with three primary goals in mind: (1) real-time classification, (2) minimizing human intervention, and (3) domain neutrality, allowing the detectors to be used for different problems without extensive adjustment. While achieving real-time classification is largely a function of the efficiency of the algorithms used, avoiding user-defined thresholds was a key factor in achieving goals (2) and (3). An overview of the design is shown in Figure 4; the classifier has access to the model's parameters and can update them at run time, making adaptation possible. This feature has been exploited in our implementation by modifying model parameters at a configurable time interval. We have incorporated support for two different algorithms — *density* and *distance* based multidimensional clustering — to detect anomalies in an N-dimensional feature space. We support a broad variety of functionality in our reference implementations to ensure the interfaces in our framework are generalizable and can be leveraged by other anomaly detector designs.

*4.1. **Anomaly Detection Algorithms***

We implemented four anomaly detection methods to demonstrate the flexibility of our framework: density, distance, Bayesian, and ensemble. Our density, distance, and Bayesian methods iterate over two main steps: (1) assigning observations to clusters and (2) updating cluster parameters based on the assignments. These steps are repeated until the models converge. The density based approach assumes the data points were generated by a finite number of mathematical distributions and focuses on iteratively improving the distributions' parameters to better fit the training data. On the other hand, the distance based algorithm clusters observations that are in close proximity on the multidimensional hyperplane and then refines the cluster centroids to minimize the overall distance between related observations. Our Bayesian approach, based on Dirichlet Process Mixture Models, supports an infinite number of mixture components. As a result, the number of clusters does not need to be predefined. Finally, our ensemble approach uses *isolation forests* to generate decision trees based on incoming observations, and uses path lengths to determine which data points are anomalous.

The density based anomaly detector employs Gaussian Mixture models (GMMs) built with Expectation Maximization (EM), while the distance based anomaly detector uses a clustering model

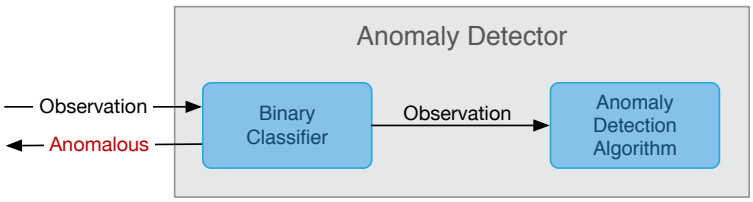

Figure 4. An overview of our reference anomaly detector designs, which are composed of a *Anomaly Detection Algorithm* and a *Binary Classifier*.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

built with K-Means. These techniques, as well as our ensemble approach, rely on implementations provided by the WEKA software package [11]. Our Bayesian approach was underpinned by functionality sourced from the Datumbox Framework [12].

4.1.1. Density Based Anomaly Detection: GMMs and EM

In general, *mixture models* are used to represent the densities of multidimensional data. Gaussian Mixture Models (GMMs) are able to approximate almost any continuous density distribution with high accuracy [13]. GMMs achieve this by: (1) applying a sufficient number of Gaussians, (2) fitting them to the data by adjusting parameters such as the means and covariances, and (3) choosing the coefficients for the linear combinations of different densities. GMMs are also used by many techniques that build models based on class densities. For instance, a mixture of Gaussian densities is used in linear and quadratic discriminant analysis to identify nonlinear decision boundaries in data [14].

We estimate GMM parameters using Expectation Maximization (EM). EM is a iterative refinement technique used for estimating the maximum likelihood of parameters in probabilistic models. For Gaussian Mixture models, EM is used to determine the means and covariance matrices for each Gaussian component.

4.1.2. Distance Based Anomaly Detection: K-Means

K-Means is a clustering technique that assigns N observations to K clusters. The algorithm starts with an initial set of clusters represented by their centroids and then iterates over two main steps: (1) assigning observations to clusters based on their proximity to cluster centroids and (2) updating each cluster's centroid by computing the mean of the assigned observations. The two steps are repeated until no changes occur in the clusters. We use canopy clustering [15] as a preprocessing step to determine the initial cluster centroids and reduce the overall number of distance calculations required for K-Means to converge. Our K-Means detector implementation also supports two functions to measure the distances of observations from the cluster centroids: *Euclidean* and *Mahalanobis*. Both distance measures use Formula 1 to compute the distance between two points (x and y). The distance is scaled in different dimensions based on the weight matrix ($\mathbf{W}$):

$$D(\mathbf{x}, \mathbf{y})^2 = (\mathbf{x} - \mathbf{y})\mathbf{W}^{-1}(\mathbf{x} - \mathbf{y})^T \tag{1}$$

The Euclidean distance uses the identity matrix as its weight matrix, which results in the features being equally weighted. Thus, Euclidean distances do not consider the correlations between features. On the other hand, the Mahalanobis distance [3] considers the correlation between features by weighting the equation with the covariance matrix Σ. In this case, features that are correlated will have a stronger influence on the distance measure.

4.1.3. Bayesian Anomaly Detection: Dirichlet Process Mixture Models (DPMMs)

Dirichlet Process Mixture Models (DPMMs) is a Bayesian unsupervised learning technique that does not require predefining the number of clusters, unlike many other approaches [4, 16, 17]. This property is particularly useful because it allows for dynamic adaptation in the number of clusters as the data points evolve. DPMMs construct a single mixture model in which the number of mixture components is infinite, and extensions can allow incoming data points to be assigned to multiple clusters. In our implementation, we used a Gaussian base distribution and a scaling parameter (α) of 0.01.

4.1.4. Ensemble Anomaly Detection: Isolation Forests

Isolation forests represent a particularly unique approach to anomaly detection. The algorithm generates decision trees by inspecting incoming observations and splitting them randomly based on the observed minimum and maximum values of each feature. This process *isolates* abnormal observations as only a few features must be deviate from the norm to produce an irregular path

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

through the trees. Path lengths are then used to determine an *anomaly score* (which is in turn used to calculate the *degree of irregularity* in our framework). Due to the stochastic nature of this approach, the process must be repeated several times to gain confidence in the resulting anomaly scores. However, isolation forests have linear time complexity and low memory requirements, allowing them to scale up to handle voluminous datasets [5].

4.2. *Binary Classifiers*

The clustering techniques we have described are used to create *binary classifiers*, which are responsible for making decisions about whether an observation is anomalous or not. The classification process begins by clustering incoming observations, deciding whether they are anomalous, and then updating model parameters for adaptation (if enabled). Our reference implementation computes thresholds that determine whether a given input sample is anomalous or not, with raw outputs used to assign *degrees of irregularity* to the data. The following subsections describe how the classifiers carry out anomaly detection and adaptation.

4.2.1. *Anomaly Detection Without Threshold Parameterization*

Given the volumes of data we consider in this work, human intervention in the anomaly classification process must be avoided. Additionally, manually choosing thresholds that accurately classify observations as *normal* or *anomalous* is often challenging [18, 19]. Further complicating matters, even if an optimal threshold could be found offline, it may not remain valid as the incoming data evolves. To address these issues, we autonomously select classification thresholds.

In our **density** and **Bayesian** clustering approaches, log-likelihood values are produced that describe the possibility that a given observation could be generated by the underlying distributions. Since the likelihood of GMMs is the average of the log-likelihood observed in 'normal' training data, it can be used as a baseline reference for determining a threshold for normal observations. Observations with likelihood values that are equal or greater than the GMMs' likelihood will not be tagged as anomalous. However, further investigation is required for observations whose likelihood is less than the GMMs' likelihood. We use the following formula to compute *fitness scores*, f_{score}, that will have negative values only for suspect observations:

$$f_{score} = \frac{obs_{llk} - model_{llk}}{|model_{llk}|} \tag{2}$$

Where:
obs_{llk} is the log-likelihood of an observation
$model_{llk}$ is log-likelihood of the model

We consider the distance of an observation from its assigned cluster centroid to adjust the values of f_{score} such that the observation's likelihood increases as its distance to the cluster centroid decreases, and vice versa. The fitness scores are adjusted by multiplying them by a distance factor X_{dist} which is derived from a normalized Euclidean distance and always has a positive value:

$$X_{dist} = \sqrt{\sum_{i=1}^{d} \frac{(x_i - \mu_i)^2}{\sigma_i}} \tag{3}$$

Where:
d is the dimensionality of the observation
x_i is the value of feature i of observation X
μ_i is the mean of the Gaussian distribution that models the feature density in space i
σ_i is the standard deviation of feature i

The value of X_{dist} will decrease as long as the distance from an observation to its cluster decreases, and vice versa. The final resulting value obtained from $f_{score} \times X_{dist}$ will be used to

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

classify an observation. We found empirically that comparing this value with -1.0 gives accurate classification results, even across different application domains. Therefore, an observation will be tagged as anomalous if the computed result is less than -1.0 and as normal if it is greater than -1.0.

In our **distance based** clustering approach we rely on the Mahalanobis distance, which gives us the scaled distance in terms of standard deviations. Based on Chebyshev's rule, 93.75% of the data of any distribution lies inside the range of 4 standard deviations of the mean. Thus, we compare the distance value with 4 to classify the observation.

Finally, our **ensemble based** approach with isolation forests does not require parameterization. However, iForests produce anomaly scores in the range $[0, 1]$ based on path lengths, where a score of 0 represents normal data with high confidence and a score of 1 represents definite anomalies. Path lengths are determined by the number of edges traversed from the isolation tree root node to the terminal edge node. While determining a threshold for what should be considered anomalous data is problem-specific, observations with an anomaly score much less than 0.5 are presumably normal. Additionally, if all observations return a score of 0.5, then there is a high likelihood that no anomalies exist in the data. Given these factors, a threshold of 0.6 is generally acceptable for most problem types [5]. We allow this default value to be changed by users, and alternatively can compute an appropriate score based on the dataset *contamination* (estimated number of probable outliers).

4.2.2. The Adaptive Algorithm

Since we are dealing with continuous data streams, our goal is to allow adaptive behavior without impacting performance. This is ensured for our distance, density, and Bayesian detectors by incrementally updating the clusters' parameters, μ and Σ, without the need to have all observations resident in memory. How often the cluster's parameters should be updated depends on a configurable *adaptation speed* that can be changed at run time.

Incremental updates of the current mean (μ_{t-1}) is performed by retrieving the previous sum and using it with the new observation to compute the new mean (μ_t). Equations 4 and 5 update the means and sizes of a cluster to which the observation x_i was assigned:

$$\mu_t = \frac{\mu_{t-1} * n_{t-1} + x_i}{n_t} \tag{4}$$

$$n_t = n_{t-1} + 1 \tag{5}$$

In both of our approaches, each element $\sigma^2_{i,j}$ in the covariance matrix is incrementally updated by following the same concept that was used for updating the mean. Equation 6 computes the new σ^2_t of the features x_i and x_j (We use Bessel's Correction to estimate variance):

$$\sigma^2_t = \frac{(n_t - 2)\sigma^2_{t-1} + (n_t - 1)(\mu_{i,t-1} - \mu_{i,t})(\mu_{j,t-1} - \mu_{j,t}) + (x_i - \mu_{i,t})(x_j - \mu_{j,t})}{n - 1} \tag{6}$$

By updating these values, we ensure that our models will adapt to evolutions in the dataset over time. This is particularly important in a variety of fields, such as atmospheric science, environmental modeling, and epidemiology.

Unlike our other anomaly detection methods, isolation forests do not require additional logic to handle adaptation as the observations evolve. Path lengths through the isolation trees will vary over time, responding to changes in the incoming data points. As a result, the isolation score produced by the algorithm for the same input data may change in response to transformations in the underlying data stream. Conversely, data streams that do not evolve over time will produce similar sets of isolation trees and path lengths.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

4.3. Evaluation of the Anomaly Detection Metric

To evaluate our reference anomaly detector, we used five classification datasets sourced from the UCI Machine Learning Repository [20]. The datasets were selected from different application domains and have varying amounts of observations and features. The datasets we used are shown in Table I.

Table I. UCI datasets used to evaluate the efficiency of our anomaly detection approaches.

Dataset	**Observations**	**Features**
Arrhythmia	452	279
Breast Cancer Wisconsin	569	32
Cardiotocography	2126	23
Seeds	210	7
Statlog (Shuttle)	58000	9

To establish ground truth for testing purposes, we considered data from the largest class as non-anomalous and used it for training, with the rest of the classes used for testing. The K-Means and EM based detectors were separately applied to each dataset. Binary classification results represented by sensitivity and specificity are listed in Table II. As illustrated by the results, both detectors perform well in most cases. The lowest-performing dataset, Arrhythmia, contained a large number of attributes but very few observations, which may explain its performance.

Table II. Binary classification evaluation for UCI datasets.

	EM		**K-Means**	
ID	**Sensitivity**	**Specificity**	**Sensitivity**	**Specificity**
D1	0.978	0.33	0.983	0.162
D2	0.740	0.972	0.917	0.642
D3	0.820	0.466	0.860	0.509
D4	0.919	0.951	0.961	0.793
D5	0.974	0.976	0.983	0.826

5. ANOMALOUS EVENTS: ANALYSIS AND VISUALIZATION

Once anomalies have been discovered and tagged with a degree of irregularity by a detector implementation, integrated analysis operations help deduce the cause of the event and what it means in the context of the particular problem domain. We support two types of analysis: *metadata inspection* to discover contextual information about anomalies, and *integrated visualization* to provide graphical representations of the data.

After being alerted of an anomalous observation, practitioners can issue Galileo queries to help discover why the event occurred. This metadata inspection process involves gathering data across distributed nodes to form traversable *result datasets* that contain metadata about the underlying observations. Metadata records include feature ranges, degrees of irregularity, and identifiers for the sensor(s) responsible for generating the observations in question. If the practitioner suspects that a sensor may be faulty, he/she can issue a query that selects all anomalous observations produced by a specific sensor identifier. The resulting datasets, which are returned as traversable graphs,

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

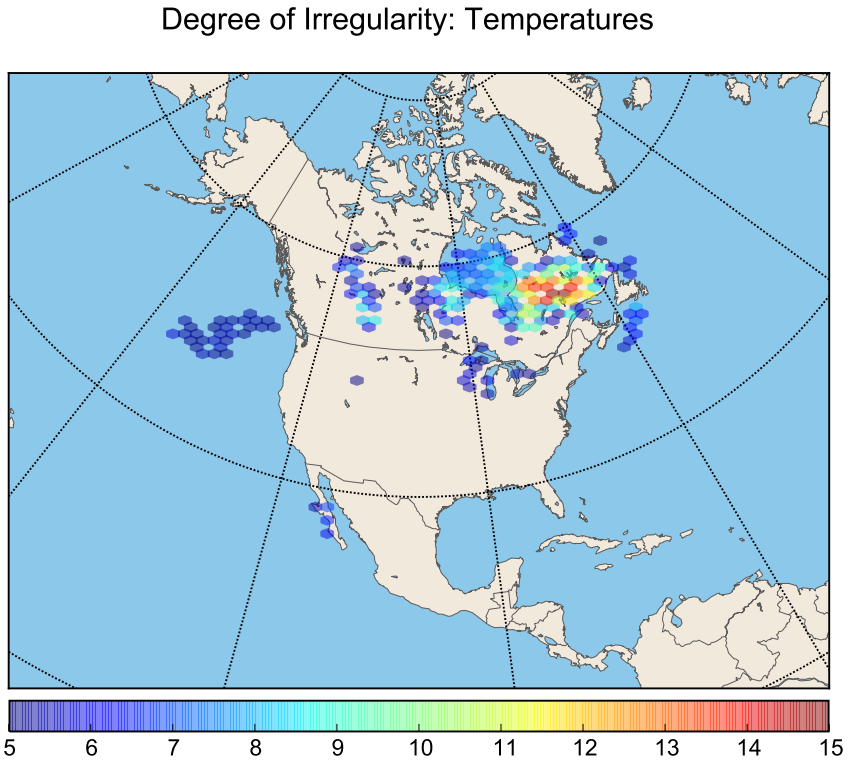

Figure 5. Degree of irregularity visualization generated by our framework. Detector instances were trained with data from 2008 – 2010, and then classified data from August, 2014. Anomalous observations are indicated in red.

provide *context* for the observations: if a certain combination of other environmental factors causes the malfunction, it will be evident in the dataset. Another scenario supported by our framework is temporal feature inspection: normal changes across a number of dimensions may influence another feature and cause an anomaly. In this case, a temporal range query can be issued to inspect the evolution of the feature space leading up to the anomalous event.

Our framework also supports several geospatial visualizations to provide graphical insights. These include contour maps, spatial binning, and scatter plots. When a visualization is requested, the storage nodes holding relevant data will produce graphical *tiles* that are stitched together to produce a final output graphic. Figure 5 demonstrates this capability; detector instances covering North America were trained with data from 2008 through 2010, and were then given temperature data from August 2014 to classify. As one might expect, recent temperatures have trended hotter, especially in colder northern regions. This resulted in several observations being flagged as anomalous (highlighted in red in the figure). Once a visualization has been created, it can be manipulated by the client (zoomed, cropped, etc.). We also support *brushing and linking* functionality, where layers or particular features can be turned on and off, ranges of interest can be modified, and query parameters updated [21].

6. EXPERIMENTAL RESULTS

We designed a series of experiments to evaluate the performance of each part of our distributed anomaly detection framework. The experiments answer the following questions:

1. Does the framework scale out as more resources are added, and is data partitioned efficiently among the resources?

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

2. Can our framework operate in real time while processing continuous data streams? While each anomaly detector implementation will have its own resource requirements, the framework itself should be lightweight.
3. Can anomalies be detected based on geospatial locations? Detector instances may be applicable to some regions but not others.
4. Given our adaptation interfaces, is the framework able to accurately adapt its model as the dataset evolves over time?
5. Can the framework detect anomalous observations whose features have normal values, but a combination of some features values are unusual?

6.1. Building the Models: Test Dataset

This study uses real-world climate data obtained from the National Oceanic and Atmospheric Administration (NOAA) North American Mesoscale Forecast System (NAM) [22]. The readings in this dataset are collected regularly from various weather and climate stations and stored in the self-describing NetCDF format [23]. Each file contains spatiotemporal information as well as several climate feature readings that include surface pressure, surface temperature, snow cover, snow depth, relative humidity, and wind speed. The particular data used in this study was collected over a eleven-year period from 2004 to 2014. Each year comprised roughly 1,000,000,000 observations on average (10 KB each), which were stored alongside compressed geographical map tiles (40 KB each). The entire dataset spanned over 1 PB of raw data and 20 billion files.

Observations from the first three years (2004 through 2006) were used as training data to build the models used in this section. Observations from the remaining years were used for verification and classification.

6.2. Experimental Setup

Our framework was run on a 78-node cluster with 48 HP DL160 servers (Xeon E5620 CPU, 12 GB of RAM) and 30 HP DL320 servers (Xeon E3-1220 V2 CPU, 8 GB of RAM). Each server was equipped with four hard disks. Ten clients running on machines outside the cluster were used to read the observations from the NOAA dataset and send them to Galileo, which operated under the OpenJDK Java Runtime, version 1.7.0_65.

Each storage node maintains an anomaly detection framework instance that receives incoming observations, and data is partitioned spatially using the Geohash-based scheme described earlier. Each storage node is responsible for a number of regions that are subdivided based on a 4-character prefix to assign a 39.1 $\times$ 19.5 km region to each anomaly detector instance. Figure 6 shows the distribution of anomaly detection instances across the cluster (note the relatively uniform distribution of load). In these benchmarks, the total number of anomaly detector instances was 60,922, and the training data for each model consisted of 49,241 observations on average.

6.3. Anomaly Detection Throughput

Our partitioning scheme and network layout help facilitate real-time anomaly detection at scale. To evaluate the scalability of our framework, we conducted a throughput test across all 78 machines in our test cluster with varying anomaly detection adaptation speeds. With adaptation disabled, the 78-node cluster could process about one million observations per second in parallel. Figure 7 illustrates the number of classified observations per second using both the EM and KM detectors at different adaptation speeds. Even when the model parameters are recomputed for every other new record, our framework is able to achieve an overall throughput of about 500,000 classifications per second. This efficiency could be leveraged in cloud based settings with low-power or low-cost resources, or used to process live data streams.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

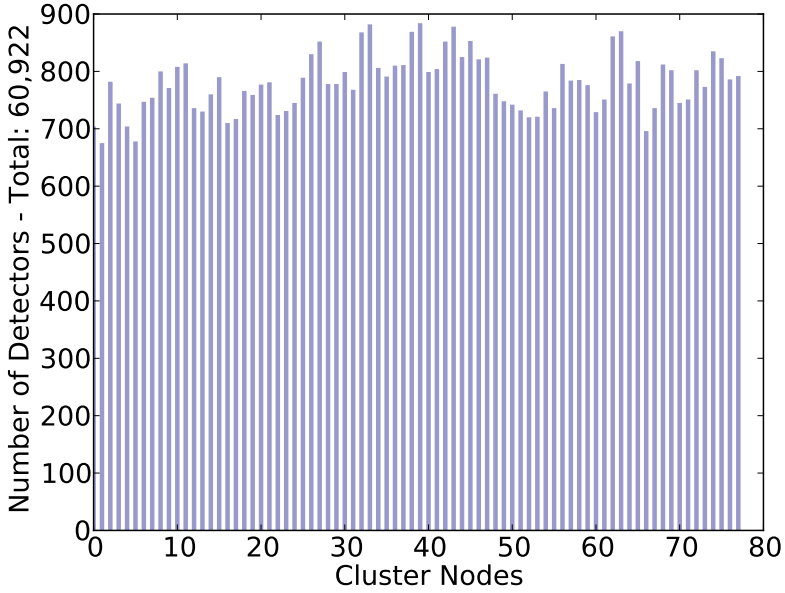

Figure 6. Anomaly detector instances at each storage node. Note the relatively uniform distribution of detectors due to our Geohash-based spatial partitioning scheme.

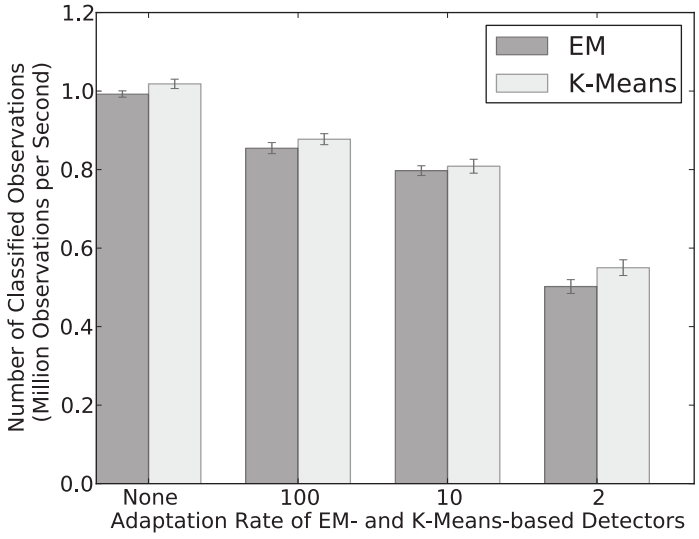

Figure 7. Observations classified per second for a variety of adaptation speeds; an adaptation rate of 100 means that the model parameters are updated each time 100 new observations have been received.

6.4. Storage Throughput

To evaluate our storage interfaces, we replaced Galileo with an HBase [24] cluster. HBase supports a wide column data model similar to BigTable [25], Cassandra [26], and Accumulo [27]. Wide-column stores are well-suited for observational data and allow for basic spatial query capabilities by using Geohash values as indexed *row keys*. HBase is also appropriate for smaller files or observational records such as those found in our atmospheric dataset, whereas systems such as HDFS are designed for general-purpose storage and larger files.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

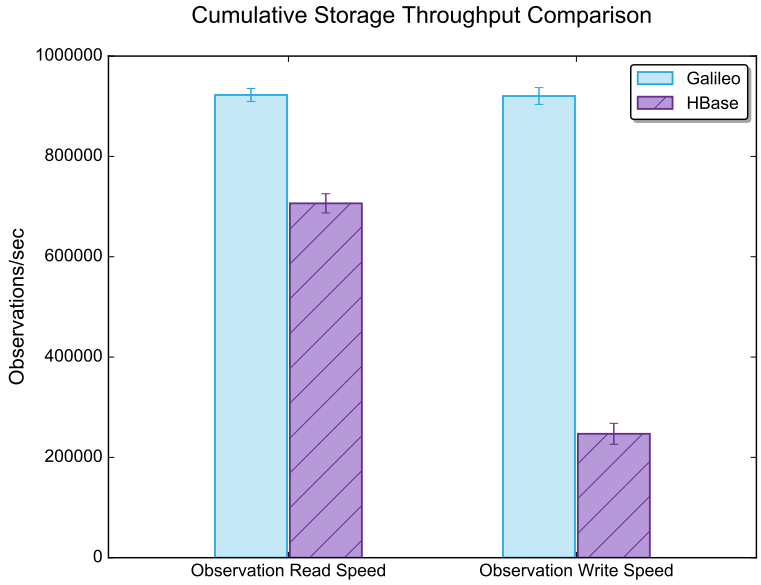

Figure 8. Storage throughput comparison between Galileo and HBase. Galileo is able to achieve higher write throughput due to its network layout and partitioning strategy.

In this benchmark, we retrieved and stored records from our test dataset using multiple concurrent client nodes. We increased the number of clients until the system reached a steady state for both read and write throughput and then compared the results produced by Galileo and HBase, shown in Figure 8. Note that for the read test, a randomized four-character Geohash lookup was performed, and in the write test adaptation was disabled to ensure the anomaly detection framework would not bottleneck the system. While both deployments exhibited similar read performance characteristics, Galileo provided considerably higher write performance due to its network layout and partitioning scheme; in Galileo, anomaly detector instances were collocated with appropriate storage nodes, resulting in reduced communication overhead.

6.5. *Impact of Geospatial Scope on Model Accuracy*

In this experiment, we demonstrate the impact of the geographic regions being managed by anomaly detector instances. We trained two models for each approach, one built with observations taken from Florida in the United States, and another built from observations belonging to Hudson Bay in Canada. Figure 9 shows the geographic differences between the two locations.

Random observations were sampled from both regions and used to test the models. If the models have specialized to their particular regions correctly, then being fed with observations from the other region should produce classifications that are definitively anomalous. On the other hand, observations that were taken from the model's own spatial region should be considered normal. Table III outlines the results of this experiment; observations from a different region are correctly tagged as anomalous while those from the model's own region are considered normal. This shows that each model instance captures fine-grained details within its own spatial region, and reinforces our decision to use multiple detection instances.

6.6. *Evaluating Model Adaptation*

In this experiment, we tested how the framework copes with changes occurring in observed behavior when the adaptation feature is turned on. We drew a random observation from the test data and created 300,000 copies of it. In each copy, the temperature was set to a value generated randomly between 330 and 360 Kelvin, which would constitute anomalous temperatures. Adaptation speed was set to 100, which means that adaptation will be performed once every 100 classification

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

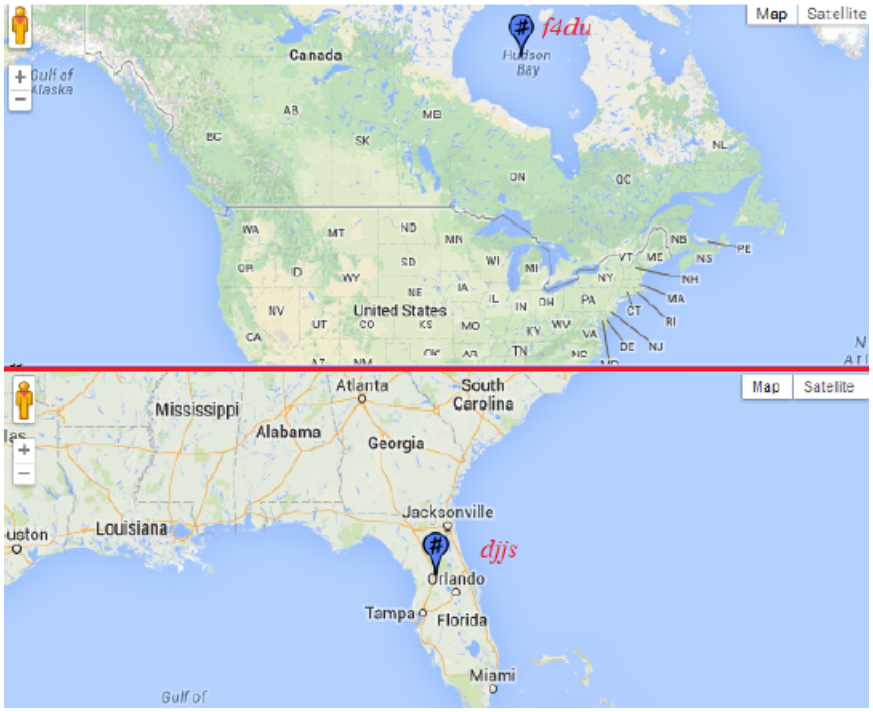

Figure 9. Two different geographic regions with correspondingly different climates: Hudson Bay, Canada, and Florida, USA.

Table III. Classification results obtained by applying EM and K-Means detectors to observations taken in January 2013 for both Hudson Bay (Geohash f4du) and Florida (djjs). To demonstrate that the models have specialized for their particular geographic region we fed in observations from the other region, which are correctly labeled as anomalous.

	EM		K-Means	
Sample Location	**Florida**	**Hudson Bay**	**Florida**	**Hudson Bay**
Hudson Bay	-1.1E22	-0.5	419.04	2.3
Hudson Bay	-1.1E22	-0.5	343.9	2.16
Hudson Bay	-1.1E22	-0.4	329.2	1.13
Florida	0.017	-538577	1.03	38.4
Florida	-0.089	-708725	1.27	46
Florida	0.007	-538253	0.34	61

operations. Because both EM and K-Means use the same adaptation algorithm we report results for only the EM-based approach, which was used to classify the 300,000 copies of the observation, followed by another 300,000 classifications with adaptation turned off.

The outcomes of two cases can be seen in Figure 10. In the figure, the x-axis represents the evaluation steps and the y-axis represents the outcomes. When the adaptation feature was disabled, the observation outcomes shown in blue were not changed over the 300,000 evaluation steps and were always considered anomalous based on our threshold of -1.0. However, when the adaptation feature was enabled, the same observations were evaluated as anomalous early on in the test but began being considered normal over time. Consequently, we can conclude that our

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

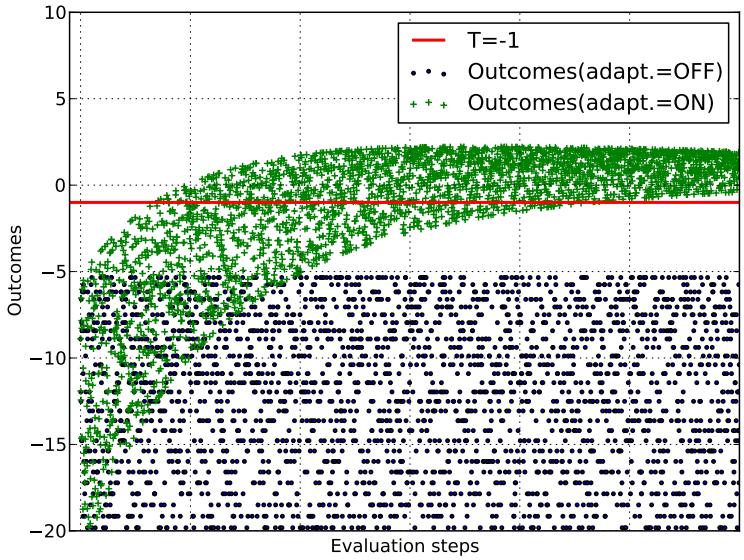

Figure 10. Outcomes obtained by running the EM-based approach twice, with and without the adaptation feature.

adaptation feature allows continuously-anomalous observations to eventually become normal over time, whereas the non-adaptive model would not be able to handle such changes.

6.7. Comparison of Anomaly Detection Methods

After the anomely detectors have been built on observations from the training data using each of our approaches, the detectors were used to classify observations from the remaining 9 years (2006 through 2014). Table IV lists the number of anomalies detected by each approach across eight different spatial locations, underscoring the influence of spatial regions on our climate dataset. Table V lists the number of common anomalies that were detected by both approaches (set intersection). This experiment illustrates that there are indeed corner cases in the dataset that can be detected by some approaches but not others. Isolation forests, for instance, have been shown to be robust to the effects of *swamping* and *masking* that cause the number of outliers to be over- or underestimated [5]. In general, it is important to be able to evaluate several approaches as each dataset contains unique properties that may be best suited for a particular approach.

Table IV. Number of anomalies detected by each of our classification methods.

Region	EM	K-Means	DPMM	iForest
9pkv	49,207	48,974	35,737	26,870
f2y8	329	308	1,153	276
dk6m	22,473	21,799	12,446	29,133
dk63	9,776	3,882	2,517	3,156
c3w6	1,014	982	1,897	1,071
c1f2	1,874	1,571	333	2,888
9wwh	430	429	2,160	929
9s0j	25,895	25,616	27,752	37,307

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

Table V. Intersection of anomalies detected by each of our classification methods.

Region	EM ∩ KM	KM ∩ DPMM	EM ∩ iForest	DPMM ∩ iForest
9pkv	48,974	34,580	9,675	10,246
f2y8	308	63	40	53
dk6m	21,799	11,079	4,640	4,659
dk63	3,882	1,472	321	436
c3w6	982	531	79	220
c1f2	1,571	271	289	198
9wwh	429	295	84	174
9s0j	25,616	20,644	6,196	9,735

7. RELATED WORK

Extensive research has been conducted in the realm of anomaly detection over data streams [28]. This includes identifying deviations based on historical trends [29], irregularities in the data itself based on entropy or relative uncertainty [30, 31], and other approaches such as Hierarchical Temporal Memory (HTM) [32] or RS-Trees/Forests [33, 34]. In general, streaming anomaly detectors require a training phase or ground truth data to define 'normal' observations. Over time, data points that are considered normal or anomalous may change.

While the main focus of this work is not anomaly detection itself, there are approaches that differ from our implementations that could also be used in conjunction with the distributed framework we have developed. A class of problems referred to as *conditional anomaly detection* requires dividing observation attributes into environmental (context) and indicator attributes [35, 36, 37, 38]. Solutions to this problem attempt to detect anomalies within specific contexts where a feature value could be normal in one context and anomalous in another. An anomalous observation in this case is one that has an unusual indicator value at a specific environmental value. Correlated attributes have to be specified by a user or detected by performing additional processing.

Approaches have also performed anomaly detection by comparing the log-likelihood value of an observation with a sorted list containing log-likelihood values of the training data [35]. An observation is tagged as anomalous if its log-likelihood is less than the threshold specified by the user. Unlike our anomaly detector implementation, this approach requires human intervention, including specification of the context and indicator variables. Some approaches tend to be non-general and require specification of a threshold. For example, Catterson et al. use an approach similar to the one described by Song et al. to monitor aging power transformers, where both environmental and indicator values are known in advance [37, 35].

Distributed approaches often incorporate anomaly detection into the devices or sensors that produce the data. This ensures scalability, but also enforces an upper bound on the computational complexity of the algorithm in use and constrains analysis from a high-level or broad-reaching perspective. Chhabra et al. investigates distributed anomaly detection in the context of network traffic flow analysis [39]. In this case, spatial anomalies may constitute broken or overwhelmed links and their respective locations in the network. Detection is performed in real time by participating routers, with anomalies confirmed by neighboring nodes. Anomalous classifications are accompanied by a significance score to provide a relative measure of confidence in the classification. While our approach also targets real-time classification of incoming data streams, its scope includes storage and analysis as well. In some cases, an anomaly may simply represent the first step towards a deeper investigation to uncover the origin or timeline leading up the event(s).

Distributed anomaly detection is particularly relevant in wireless sensor networks to determine faulty or misbehaving equipment. In general, these networks are composed of hundreds or thousands

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

of low-power, unsophisticated sensors that cannot be monitored or administered by traditional means. Analyzing data produced by sensors in close proximity or with similar functionality helps pinpoint events of interest, or potentially anomalous behavior in cases where a strong consensus among disparate sensors cannot be reached. Rajasegarar et al. proposes a distributed, spatial anomaly detection system for wireless sensor networks that uses a bottom-up approach to accumulate readings and classify them [40]. In this case, either the devices producing the data or intermediary nodes are used to combine and pre-process incoming records. To contrast, our framework allows untouched sensor readings (or other types of data) to be streamed in with the intention of both (1) detecting potential anomalies and their significance, (2) persisting the readings on stable storage for future analysis. An additional benefit of this scheme as opposed to pushing anomaly detection activities out to the devices producing them is scalability in situations where the computational requirements of the anomaly detection algorithm are high.

Zhao et al. proposes a parallel K-Means implementation that relies on the MapReduce framework [41, 9, 42]. However, as we have demonstrated, clustering-based anomaly detectors are only one of several valid approaches and performance may vary depending on the dataset. Further techniques that rely on MapReduce are proposed to detect anomalies in large scale datasets [43, 44]. Despite the powerful computation capabilities provided by MapReduce, its execution model is not always amenable for online anomaly detection; in many cases, effective detection of anomalies requires an iterative approach, with records kept in main memory while analysis is conducted.

As a general parallel computing abstraction for distributed data, Resilient Distributed Datasets (RDDs) along with MLBase could be leveraged for anomaly detection [45, 46]. RDDs are able to maintain datasets in memory across a cluster, allowing for iterative and multi-stage computations without needing to read or write from secondary storage. However, RDDs do not contain provisions for spatiotemporal data or integrated anomaly analysis. Hagedorn et al. [47] proposes a set of RDD transformation operators that allow spatiotemporal correlation analysis within the Spark framework [48]. After transforming raw textual input data into RDD-based *event records*, the framework supports calculating the spatiotemporal distance between events as well as clustering via the DBSCAN (density-based spatial clustering of applications with noise) [49] algorithm, which could be leveraged by our anomaly detector instances. By designing our framework for this specific problem domain, we can provide similar functionality as well as geospatially-aware data partitioning, computation primitives designed for anomaly detection, and analytics functionality for tracing and tracking the source of anomalies.

While not strictly designed for streaming data, Apache GeoMesa [50, 51] provides spatiotemporal storage and retrieval functionality that could be used as a storage backend for our anomaly detection components. Similar to Galileo, GeoMesa uses the Geohash algorithm to represent spatial locations. To provide indexing capabilities through an underlying wide-column store such as Cassandra [26] or Accumulo [27], spatiotemporal attributes are encoded as flat lexicographic strings. Unlike our framework, this approach favors load balancing characteristics over data locality. GeoMesa is extensible and can be used as a Kafka [52] data source or with GeoServer [53] for rendering and manipulating spatial data.

8. CONCLUSIONS

This paper presented our approach, encompassing algorithms and system design, for scalable detection of anomalies in multidimensional, geospatial data streams.

RQ-1: To deal with observations that are spatiotemporally correlated, each of our models are responsible for a particular geographical extent and include timestamps associated with incoming data points. Each model instance tunes itself independently based on the data it observes. Rather than employing a single model that attempts to preserve such correlations, using individual instances that span smaller geographical extents improves accuracy. This approach is well-suited for situations where there is variability in the data streams for particular geographical extents, allowing the regions to be adaptively refined with corresponding additions of model instances. Ultimately, our approach

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

allows fine-tuning of the specificity of the classifications by controlling the geographical scope associated with the classification models.

RQ-2: Associating model instances with particular geographical extents results in a scalable design; i.e., we can scale with increases in data volumes and the number of machines available. This approach is also amenable to dispersion, with models executing on multiple machines and performing concurrent, distributed classifications as observations arrive. Multiple model instances are also maintained at each node using a thread pool, which allows for concurrent classifications of data streams within a particular geospatial area. Our empirical results validate the scalability and throughput of our approach at both the individual nodes and in a distributed cluster, which is capable of reaching an overall throughput of over 1,000,000 classifications per second. The specificity of our models — controlled by the granularity of the Geohash — allows us to account for spatiotemporal correlations associated with the observations and achieve greater accuracy.

RQ-3: Our anomaly detector interfaces are agnostic to their underlying algorithms and implementations. The interfaces automate three key phases in detection of anomalies: collection, training, and classification. The training phase buffers and prepares data for training, while the training phase creates and trains models on the collected data. Finally, the classification phase uses the trained models to classify observations as either normal or anomalous. Our reference implementation incorporates support for distance (Euclidean and Mahalanobis), density, Bayesian, and ensemble-based clustering algorithms. The feasibility of this approach was verified with well-known datasets.

RQ-4: Given the associated data volumes, dimensionality, and the rates at which observations arrive, it is infeasible to employ human intervention in the anomaly classification process. Our approach does not require human intervention for the adjustment of anomaly detection thresholds and provides *degree of irregularity* scores to help users direct their efforts. Our classifiers continually and autonomously adapt themselves based on the data observed by models, i.e., both the adaptation and the classification are performed concurrently. Rather than requiring explicit coordination amongst detector instances, we allow a consensus to be reached through distributed anomaly queries. Our empirical benchmarks demonstrate both the efficiency (per-packet classifications) and accuracy of these adaptations.

ACKNOWLEDGMENT

This research was supported by funding from the US Department of Homeland Security (HSHQDC-13-C-B0018, D15PC00279), the US National Science Foundation's Advanced Cyberinfrastructure and Computer Systems Research Programs (ACI-1553685, CNS-1253908), and the Environmental Defense Fund (0164-000000-10410-100).

REFERENCES

1. Malensek M, Pallickara SL, Pallickara S. Exploiting geospatial and chronological characteristics in data streams to enable efficient storage and retrievals. *Future Generation Computer Systems* 2012; .
2. Malensek M, Pallickara SL, Pallickara S. Expressive query support for multidimensional data in distributed hash tables. *Utility and Cloud Computing (UCC), 2012 Fifth IEEE International Conference on*, 2012.
3. Mahalanobis PC. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)* 1936; **2**:49–55.
4. Escobar MD, West M. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association* 1995; **90**(430):577–588. URL http://www.jstor.org/stable/2291069.
5. Liu FT, Ting KM, Zhou ZH. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data* Mar 2012; **6**(1):3:1–3:39, doi:10.1145/2133360.2133363. URL http://doi.acm.org/10.1145/2133360.2133363.
6. Malensek M, Pallickara SL, Pallickara S. Polygon-based query evaluation over geospatial data using distributed hash tables. *Utility and Cloud Computing (UCC), 2013 Sixth IEEE International Conference on*, 2013.
7. Bentley JL. Multidimensional binary search trees used for associative searching. *Commun. ACM* Sep 1975; **18**(9):509–517.
8. Fredkin E. Trie memory. *Commun. ACM* Sep 1960; **3**(9):490–499.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

9. Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM* Jan 2008; **51**(1):107–113, doi:10.1145/1327452.1327492. URL http://doi.acm.org/10.1145/1327452.1327492.
10. Niemeyer G. Geohash 2008. URL http://en.wikipedia.org/wiki/Geohash.
11. Hall M, *et al.*. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.* Nov 2009; **11**(1):10–18.
12. Datumbox Machine Learning Framework. Datumbox 2016. URL http://www.datumbox.com.
13. Bishop CM. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 2006.
14. Hastie T, Tibshirani R, Friedman J, Hastie T, Friedman J, Tibshirani R. *The elements of statistical learning*, vol. 2. Springer, 2009.
15. McCallum A, Nigam K, Ungar LH. Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, ACM: New York, NY, USA, 2000; 169–178, doi:10.1145/347090.347123. URL http://doi.acm.org/10.1145/347090.347123.
16. Antoniak CE. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *Ann. Statist.* 11 1974; **2**(6):1152–1174, doi:10.1214/aos/1176342871. URL http://dx.doi.org/10.1214/aos/1176342871.
17. Görür D, Edward Rasmussen C. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology* 2010; **25**(4):653–664, doi:10.1007/s11390-010-9355-8. URL http://dx.doi.org/10.1007/s11390-010-9355-8.
18. Patcha A, Park JM. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks* 2007; **51**(12):3448–3470.
19. Raginsky M, Willett RM, Horn C, Silva J, Marcia RF. Sequential anomaly detection in the presence of noise and limited feedback. *Information Theory, IEEE Transactions on* 2012; **58**(8):5544–5562.
20. Bache K, Lichman M. UCI machine learning repository 2013. URL http://archive.ics.uci.edu/ml.
21. Koontz J, Malensek M, Pallickara SL. Geolens: Enabling interactive visual analytics over large-scale, multidimensional geospatial datasets. *Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing (BDC)*, 2014; 35–44, doi:10.1109/BDC.2014.12.
22. National Oceanic and Atmospheric Administration. the north american mesoscale forecast system. http://www.emc.ncep.noaa.gov/index.php?branch=nam 2013.
23. Rew R, Davis G. Netcdf: an interface for scientific data access. *Computer Graphics and Applications, IEEE* 1990; **10**(4):76–82.
24. The Apache Software Foundation. Apache HBase: A distributed database for large datasets ; URL http://hbase.apache.org.
25. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.* Jun 2008; **26**(2):4:1–4:26, doi:10.1145/1365815.1365816. URL http://doi.acm.org/10.1145/1365815.1365816.
26. Lakshman A, Malik P. Cassandra: A decentralized structured storage system. *SIGOPS Oper. Syst. Rev.* Apr 2010; **44**(2):35–40, doi:10.1145/1773912.1773922. URL http://doi.acm.org/10.1145/1773912.1773922.
27. The Apache Software Foundation. Apache Accumulo ; URL https://accumulo.apache.org.
28. Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. *ACM Comput. Surv.* Jul 2009; **41**(3):15:1–15:58, doi:10.1145/1541880.1541882. URL http://doi.acm.org/10.1145/1541880.1541882.
29. Aggarwal CC. On abnormality detection in spuriously populated data streams. *SDM*, SIAM, 2005; 80–91.
30. Lee W, Xiang D. Information-theoretic measures for anomaly detection. *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, IEEE Computer Society: Washington, DC, USA, 2001; 130–. URL http://dl.acm.org/citation.cfm?id=882495.884435.
31. Arning A, Agrawal R, Raghavan P. A linear method for deviation detection in large databases. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, AAAI Press, 1996; 164–169. URL http://dl.acm.org/citation.cfm?id=3001460.3001495.
32. Hawkins J, George D. Hierarchical temporal memory: Concepts, theory and terminology. *Technical Report*, Technical report, Numenta 2006.
33. Tan SC, Ting KM, Liu TF. Fast anomaly detection for streaming data. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, AAAI Press, 2011; 1511–1516, doi:10.5591/978-1-57735-516-8/IJCAI11-254. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-254.
34. Wu K, Zhang K, Fan W, Edwards A, Philip SY. Rs-forest: A rapid density estimator for streaming anomaly detection. *2014 IEEE International Conference on Data Mining*, IEEE, 2014; 600–609.
35. Song X, Wu M, Jermaine C, Ranka S. Conditional anomaly detection. *Knowledge and Data Engineering, IEEE Transactions on* 2007; **19**(5):631–645.
36. Wang X, Davidson I. Discovering contexts and contextual outliers using random walks in graphs. *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, IEEE, 2009; 1034–1039.
37. Catterson VM, McArthur SD, Moss G. Online conditional anomaly detection in multivariate data for transformer monitoring. *Power Delivery, IEEE Transactions on* 2010; **25**(4):2556–2564.
38. McArthur SD, Booth CD, McDonald J, McFadyen IT. An agent-based anomaly detection architecture for condition monitoring. *Power Systems, IEEE Transactions on* 2005; **20**(4):1675–1682.
39. Chhabra P, Scott C, Kolaczyk E, Crovella M. Distributed spatial anomaly detection. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, doi:10.1109/INFOCOM.2008.232.
40. Rajasegarar S, Leckie C, Palaniswami M, Bezdek J. Distributed anomaly detection in wireless sensor networks. *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*, 2006; 1–5, doi:10.1109/ICCS.2006.301508.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

41. Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce. *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, Springer-Verlag: Berlin, Heidelberg, 2009; 674–679.
42. Lämmel R. Googles mapreduce programming modelrevisited. *Science of computer programming* 2008; **70**(1):1–30.
43. Lee J, Cha S. Page-based anomaly detection in large scale web clusters using adaptive mapreduce (extended abstract). *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, RAID 08, Springer-Verlag.
44. Wang K, Wang Y, Yin B. A density-based anomaly detection method for mapreduce. *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, IEEE, 2012; 159–162.
45. Kraska T, Talwalkar A, Duchi JC, Griffith R, Franklin MJ, Jordan MI. Mlbase: A distributed machine-learning system. *CIDR*, 2013.
46. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, USENIX Association: Berkeley, CA, USA, 2012; 2–2. URL http://dl.acm.org/citation.cfm?id=2228298.2228301.
47. Hagedorn S, Sattler KU, Gertz M. *A Framework for Scalable Correlation of Spatio-temporal Event Data*. Springer International Publishing: Cham, 2015; 9–15, doi:10.1007/978-3-319-20424-6_2. URL http://dx.doi.org/10.1007/978-3-319-20424-6_2.
48. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, USENIX Association: Berkeley, CA, USA, 2010; 10–10. URL http://dl.acm.org/citation.cfm?id=1863103.1863113.
49. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, AAAI Press, 1996; 226–231.
50. Fox A, Eichelberger C, Hughes J, Lyon S. Spatio-temporal indexing in non-relational distributed databases. *2013 IEEE International Conference on Big Data*, Institute of Electrical and Electronics Engineers (IEEE), 2013, doi:10.1109/bigdata.2013.6691586. URL http://dx.doi.org/10.1109/BigData.2013.6691586.
51. Hughes JN, Annex A, Eichelberger CN, Fox A, Hulbert A, Ronquest M. GeoMesa: a distributed architecture for spatio-temporal fusion. *Geospatial Informatics, Fusion, and Motion Video Analytics V*, Pellechia MF, Palaniappan K, Doucette PJ, Dockstader SL, Seetharaman G (eds.), SPIE-Intl Soc Optical Eng, 2015, doi:10.1117/12.2177233. URL http://dx.doi.org/10.1117/12.2177233.
52. Kreps J, Narkhede N, Rao J, *et al.*. Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB*, 2011; 1–7.
53. Deoliveira J. Geoserver: uniting the geoweb and spatial data infrastructures. *Proceedings of the 10th International Conference for Spatial Data Infrastructure, St. Augustine, Trinidad*, 2008.

Copyright © 2017 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls