A Time Efficient Approach for Detecting Errors in Big Sensor Data on Cloud

Chi Yang, Chang Liu, Xuyun Zhang, Surya Nepal, and Jinjun Chen, *Senior Member, IEEE*

Abstract—Big sensor data is prevalent in both industry and scientific research applications where the data is generated with high volume and velocity it is difficult to process using on-hand database management tools or traditional data processing applications. Cloud computing provides a promising platform to support the addressing of this challenge as it provides a flexible stack of massive computing, storage, and software services in a scalable manner at low cost. Some techniques have been developed in recent years for processing sensor data on cloud, such as sensor-cloud. However, these techniques do not provide efficient support on fast detection and locating of errors in big sensor data sets. For fast data error detection in big sensor data sets, in this paper, we develop a novel data error detection approach which exploits the full computation potential of cloud platform and the network feature of WSN. Firstly, a set of sensor data error types are classified and defined. Based on that classification, the network feature of a clustered WSN is introduced and analyzed to support fast error detection and location. Specifically, in our proposed approach, the error detection is based on the scale-free network topology and most of detection operations can be conducted in limited temporal or spatial data blocks instead of a whole big data set. Hence the detection and location process can be dramatically accelerated. Furthermore, the detection and location tasks can be distributed to cloud platform to fully exploit the computation power and massive storage. Through the experiment on our cloud computing platform of U-Cloud, it is demonstrated that our proposed approach can significantly reduce the time for error detection and location in big data sets generated by large scale sensor network systems with acceptable error detecting accuracy.

Index Terms—Big data, cloud computing, data abnormality, error detection, time efficiency, sensor networks, complex network systems

1 Introduction

Recently, we enter a new era of data explosion which brings about new challenges for big data processing. In general, big data [1], [2] is a collection of data sets so large and complex that it becomes difficult to process with on-hand database management systems or traditional data processing applications. It represents the progress of the human cognitive processes, usually includes data sets with sizes beyond the ability of current technology, method and theory to capture, manage, and process the data within a tolerable elapsed time [1], [2], [12], [13], [14], [15], [16], [17], [30], [31], [32], [33]. Big data has typical characteristics of five 'V's, volume, variety, velocity, veracity and value. Big data sets come from many areas, including meteorology, connectomics, complex physics simulations, genomics, biological study, gene analysis and environmental research [1], [2]. According to literature [1], [2], since 1980s, generated data doubles its size in every 40 months all over the world. In the year of 2012, there were 2.5 quintillion (2.5×1018) bytes of data being generated every day. Hence, how to process big data has become a fundamental and critical challenge for modern society. Cloud computing provides a promising platform for big data processing with powerful computation capability, storage, scalability, resource reuse and low cost, and has attracted significant attention in alignment with big data.

One of important source for scientific big data is the data sets collected by wireless sensor networks (WSN). Wireless sensor networks have potential of significantly enhancing people's ability to monitor and interact with their physical environment. Big data set from sensors is often subject to corruption and losses due to wireless medium of communication and presence of hardware inaccuracies in the nodes. For a WSN application to deduce an appropriate result, it is necessary that the data received is clean, accurate, and lossless. However, effective detection and cleaning of sensor big data errors is a challenging issue demanding innovative solutions.

WSN with cloud can be categorized as a kind of complex network systems [21]. In these complex network systems [21], [22], [23], [24], such as WSN and social network, data abnormality and error become an annoying issue for the real network applications [25], [26], [27]. Therefore, the question of how to find data errors in complex network systems for improving and debugging the network has attracted the interests of researchers. Some work [28], [30] has been done for big data analysis and error detection in complex networks including intelligence sensors networks. There are also some works related to complex network systems data error detection and debugging with online data processing techniques [37], [38]. Since these techniques were not designed and developed to deal with big data on cloud, they were unable to cope with current dramatic increase of data size. For example, when big data sets are encountered, previous offline methods for error detection

- *C. Yang, C. Liu, X. Zhang and J. Chen are with the Faculty of Engineering and IT, University of Technology, Sydney, Australia, NSW2007. E-mail:{chiyangit, changliu.it, xyzhanggz, jinjun.chen}@gmail.com.*
- *S. Nepal is with the Centre for Information & Communication Technologies, Commonwealth Scientific & Industrial Research Organization, Marsfield, NSW 2122, Australia. E-mail: Surya.Nepal@csiro.au.*

Manuscript received 11 Oct. 2013; revised 24 Nov. 2013; accepted 6 Dec. 2013. Date of publication 15 Jan. 2014; date of current version 9 Jan. 2015. Recommended for acceptance by S. Naik.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.2295810

and debugging on a single computer may take a long time and lose real time feedback. Because those offline methods are normally based on learning or mining, they often introduce high time cost during the process of data set training and pattern matching.

WSN big data error detection commonly requires powerful real-time processing and storing of the massive sensor data as well as analysis in the context of using inherently complex error models to identify and locate events of abnormalities. In this paper, we aim to develop a novel error detection approach by exploiting the massive storage, scalability and computation power of cloud to detect errors in big data sets from sensor networks. Some work has been done about processing sensor data on cloud [38], [39]. However, fast detection of data errors in big data with cloud remains challenging. Especially, how to use the computation power of cloud to quickly find and locate errors of nodes in WSN needs to be explored. Cloud computing, a disruptive trend at present, poses a significant impact on current IT industry and research communities. Cloud computing infrastructure is becoming popular because it provides an open, flexible, scalable and reconfigurable platform. The proposed error detection approach in this paper will be based on the classification of error types. Specifically, nine types of numerical data abnormalities/errors are listed and introduced in our cloud error detection approach. The defined error model will trigger the error detection process. Compared to previous error detection of sensor network systems, our approach on cloud will be designed and developed by utilizing the massive data processing capability of cloud to enhance error detection speed and real time reaction. In addition, the architecture feature of complex networks will also be analyzed to combine with the cloud computing with a more efficient way. Based on current research literature review, we divide complex network systems into scale-free type and non scale-free type. Sensor network is a kind of scale-free complex network system which matches cloud scalability feature. Our proposed error detection approach on cloud is specifically trimmed for finding errors in big data sets of sensor networks. The main contribution of our proposed detection is to achieve significant time performance improvement in error detection without compromising error detection accuracy.

The reminder of this paper is organized as follows. In Section 2, we review related work and conduct problem analysis. In Section 3, the classification and definition are provided for differentiating error types in big data sets of complex network systems, such as WSN on cloud. In Section 4, based on the defined error types and models, a fast approach is developed to detect big sensor data with cloud computing. Section 5, the algorithms will be developed with related analysis. In Section 6, the experimental results will be presented and analyzed to show significant time performance improvement with accuracy. In Section 7, we conclude the research contributions of this paper with a brief outlook of future work.

2 Related Work and Problem Analysis

To address various challenges of big data, research works can be found intensively from the database view [30], [31], [32]. However, the problem can be also discussed from the perspective of parallel systems and cloud [35], [36]. In this section, related literature for big data processing on cloud, and data error detection for complex network systems will be reviewed and compared.

2.1 Big Data Processing on Cloud

With the fast development of modern information technology, we enter a new era of data. Hence, the technique to process big data has become a fundamental and critical challenge for modern society. Cloud computing can be regarded as an ingenious combination of a series of developed or develop-ing ideas and technologies, establishing a pay-as-you-go business model by offering IT services using economies of scale [5], [6], [7], [8], [9], [10], [11]. Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing provides an ideal platform for big data storage, dissemination and interpreting with its massive computation power [3], [4]. In many today's real world applications, such as social networks, complex network monitoring, the scientific analysis of protein interactions and wireless sensor networks self monitoring, it is unavoidable to encounter the problem of dealing with big data and big data streams on cloud.

At present, some work has been done for processing big data with cloud. Amazon EC2 infrastructure as a service is a typical cloud based distributed system for big data processing. Amazon S3 supports distributed storage. MapReduce [7], [10], [18], [19], [20] is adopted as a programming model for big data processing over cloud computing. Plenty of recent research has investigated the issues of processing incremental data on cloud. Kienzler et al. [8] designed a "stream-as-you-go" approach to access and process on incremental data for data-intensive cloud applications via a stream-based data management architecture. The extension of the traditional Hadoop framework [11] to develop a novel framework named Incoop by incorporating several techniques like task partition and memorization-aware schedule. Olston et al. [9] present a continuous workflow system called Nova on top of Pig/Hadoop through stateful incremental data processing. MapReduce has been widely revised from a batch processing framework into a more incremental one to analyze huge-volume of incremental data on cloud. It is a framework for processing parallelizable problems across big data sets using a large number of computers (nodes), collectively referred to as a cluster in which all computers (nodes) are on the same local network and use similar hardware; or a grid in which the nodes are shared across geographically and administratively distributed systems. It can sort a petabyte of data in only a few hours. The parallelism also provides some possibility of recovering from partial failure of servers or storage during the operation.

According to the above literature, most of current big data processing and analysis techniques on cloud focus on the workload distribution, scalability, data filtering speed,

and query accuracy. However, there is not enough work dedicated to the issue of error detection and correction for big data sets with cloud computing.

2.2 On-Cloud Processing for WSN

Recently, wireless sensor network systems have been used in different areas, such as environment monitoring, military, disaster warning and scientific data collection. In order to process the remote sensor data collected by WSN, sensor-cloud platform [28], [38], [39] has been developed including its definition, architecture, and applications. Due to the features of high variety, volume, and velocity, big data is difficult to process using on-hand database management tools or traditional sensor-cloud platform. Big data sets can come from complex network systems, such as social network and large scale sensor networks. In addition, under the theme of complex network systems, it may be difficult to develop time-efficient detecting or trouble-shooting methods for errors in big data sets, hence to debug the complex network systems in real time [21], [22], [23], [29].

Sensor-Cloud [27] is a unique sensor data storage, visualization and remote management platform that leverages powerful cloud computing technologies to provide excellent data scalability, fast visualization, and user programmable analysis. Initially, sensor-cloud was designed to support long-term deployments of MicroStrain wireless sensors. But nowadays, sensor-cloud has been developed to support any web-connected third party device, sensor, or sensor network through a simple OpenData API. Sensor-Cloud can be useful for a variety of applications, particularly where data from large sensor networks needs to be collected, viewed, and monitored remotely. For example, structural health monitoring and condition-based monitoring of high value assets are applications where commonly available data tools often come up short in terms of accessibility, data scalability, programmability, or performance. Sensor-Cloud represents a direction for processing and analyzing big sensor data using cloud platform.

The online WSN data quality and data cleaning issues are discussed in [37] by Elnahrawy and Nath. They deal with the problems of outliers, missing information, and noise. A novel online approach for modeling and online learning of temporal-spatial data correlations in sensor networks is developed. A Bayesian approach for reducing the effect of noise on sensor data online is also proposed [37]. The proposed approach is efficient in reducing the uncertainty associated with noisy sensors. However, the scalability and error detection accuracy are not dealt. It is an initial and important step for online error detection of WSN. But lots of work still needs to be done. Especially, under the cloud environment, the computational power and scalability should be fully exploit to support the real time fast error detection for sensor data sets.

To the best of our knowledge based on the above work, the error detection issues of big data from WSN are rarely discussed on current sensor-cloud technology or other online WSN data processing techniques.

2.3 Data Error Detection in Sensor Networks and Complex Networks

As an important scientific big data source, scientific sensor systems and wireless sensor network applications produce a variety of large data sets in real time through various monitored activities in different application domains, such as healthcare, military, environment, and manufacturing.

In many real world complex network systems, data error is unavoidable [22], [23], [24], [25], [26], [34]. With the dramatic increase of big data generated from complex network systems, such as social networks and large scale sensor networks, to find and locate the errors in big data sets becomes quite challenging with normal computing and network systems.

Wang et al. [22] provide a classification for errors on social networks based on error scenarios analysis. This classification includes 6 types of common errors with missing data or erroneous data. This work compares the robustness of four node-level network measures, clustering coefficient, network constraint, and centrality. It performs as a good base for developing error finding and detecting techniques for social networks. Social network is a typical instance of complex networks with graph data sets with it. Hence, the error models and types presented in [22] can be extended for the errors in complex network systems. Xiong proposed an approach [23] which can be used to detect the text data errors in data sets of social network.

Mukhopadhyay [24] proposed a model based error correction method for WSN. It is conducted over intelligent sensor network itself. This technique is based on the correction with data trend prediction. Because the work [24] is in-network fast error detection by intelligent sensors, its processing capability and time performance are extremely limited when encountering big data sets. Similar work can also be conducted with the consideration of data awareness and low cost according to the description of Mukhopadhyay. Slijepcevic analyses the location errors in sensor networks in [25]. The primary goal of this location error analysis is to demonstrate the practical use of the location errors for optimal resource consumption. Ramanathan presents a detailed study of sensor faults that occur in deployed sensor networks and a systematic approach to model these faults [26]. Sheth develops a decentralized fault diagnosis system for WSN in [28]. It enables efficient management of a WSN by diagnosing the true root cause of a degraded performance by combining multiple sensor observations. This diagnosis requires minimal data collection at the centralized base station. Khan in [29] presents a sensor network troubleshooting tool that helps the developer diagnose root causes of errors. The tool is geared towards finding interaction bugs. Anyway, in the paper [29], it is pointed out that scalability, user interface and detecting time still need to be improved.

It can be concluded that current data error detection techniques for complex network systems focus on in-network detecting with intelligent nodes or offline analysis at the root. They ignore the scalability, massive resource and powerful computation capability provided by cloud. The proposed approach in this paper aims to address this issue by utilizing the inherent features of cloud computing to realize fast error detection on cloud. In addition, the traditional error detection for WSN data sets has not paid enough

attention to making use of complex network features to improve the error detection efficiency on the cloud platform. Compared to the previous sensor data error detection and localization approach, complex network topology features will be explored with the computation power of cloud for error detection efficiency, scalability and low cost.

3 Error Types in WSN Big Data Sets

Many systems in nature can be described as large networks (nodes or vertices connected by links or edges): Friendship networks, Social networks, computer networks, Internet, metabolic networks, power grids, scientific citations, neural networks and large scale sensor networks. Network analysis has been troubled by the issue of measurement of error for a long time [21], [22], [23]. Before deploying an error detection approach on cloud, the error models for big data sets from wireless sensor network systems perspective should be presented first.

3.1 Error and Abnormality Classification

Under the theme of the big data sets from real world complex networks, there are mainly two types of data generated and exchanged within networks. (1) The numeric data sampled and exchanged between network nodes such as sensor network sampled data sets. (2) The text files and data logs generated by nodes such as social network data sets. In this paper, our research will focus on the error detection for numeric big data sets from complex networks.

In the previous work [22], the errors of complex networks can be classified as six main types for both numeric and text data as Appendix A.1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/ TPDS.2013.2295810. This error classification can effectively describe the common error types in complex network systems. However, when it comes to the errors in wireless sensor network data sets, the above classification loses the accuracy in separating node or edge data error caused by different wireless data communication failures. In addition, it is not enough in describing the error data phenomena in sensor data sets. To better capture the error features of sensor data sets, the above general error classification in [22] should be extended.

Considering the specific feature of numeric data errors, there are several abnormal data scenarios demonstrated in Fig. 1. The "flat line faults" indicates a time series of a node in a network system keeps unchanged for unacceptable long time duration. In real world applications, sampled data and transmitted data always have slight changes with the time flow. The "out of data bounds faults" indicates impossible data values are observed based on some domain knowledge. In real world applications, if a temperature value of water is reported as 300 °C, it can be treated as a data fault directly. The "data lost fault" means there are missing data values in a time series during the data generation or communication. The time series with "data lost fault" normally needs data cleaning. Finally, in Fig. 1, the "spike faults" indicates in a time series data items which are totally out of the prediction and normal changing trend. Because the above four types of errors can happen both at data generation and exchange stages, the error types can also be categorized into node side and edge side separately. Combining the data faults scenarios in Fig. 1 with the work in paper [21], [22], [23], [29], we present a classification of complex network systems data errors based on the time series analysis as Appendix A.2, available in the online supplemental material.

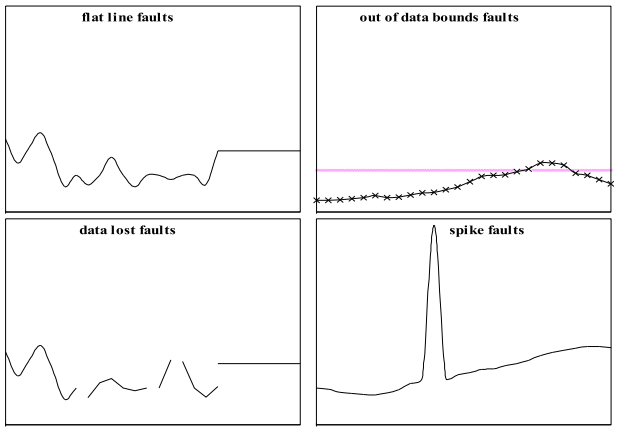

Fig. 1. Error scenarios from sensor network systems data.

3.2 Error Definition and Modeling

With the above classification, the definition of each error type is presented to guide our error detection algorithm. Suppose that a data record from a network node is denoted as *r(n, t, f(n, t), g(n, l))*, where n is the ID of the node in a network systems. t represents the window length of a time series. *f(n, t)* is the numerical values collected within window t from the node n. *g(n, l)* is a location function which records the cluster, the data source node and partition situation related to the node n. *g(n, l)* is used to calculate the distance between the data source node n and the node l which is the initial data source node. *g(n, l)* indicates that a current detected error data node is the initial data source node. Furthermore, *g(n, l)* is also used to parse the data routing between data communication nodes.

Definition 1 (node side flat line error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, $g(n_i, l)$) be a time series record from node n_i, where i is a time stamp. If any element $x \equiv \delta_i$, where δ_i is an effective constant during time window $t, x \in f(n_i, t_i)$, and $g(n_i, l) = 0, n_i$ is the data source node, there is a node side flat line error.*

Definition 2 (edge side flat line error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, $g(n_i, l)$) be a time series record from node n_i, where i is a time stamp. If any element $x \equiv \delta_i$, where δ_i is an effective constant during time window $t, x \in f(n_i, t_i)$, and $g(n_i, l)! = 0, n_i$ is the data source node, there is an edge side flat line error.*

Definition 3 (node side data lost error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, g(n, l)) be a time series record from node n_i, where i is a time stamp. If $f(n_i, t_i) = null \;\&\&\; t_i > \tau$, '$\tau$' is the time duration from outside application requirement, and if $g(n_i, l) = 0, n_i$ is the data source node, the error is a node side data lost error.*

Definition 4 (edge side data lost error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, g(n, l)) be a time series record from node n_i, where i is a time stamp. If $f(n_i, t_i) = null \;\&\&\; t_i > \tau$, '$\tau$' is the time duration from outside application requirement, and if*

$g(n_i, l)! = 0, n_i$ is the data source node, the error is an edge side data lost error.

Definition 5 (node side out of bounds error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, g(n, l)) be a time series record from node n_i, where i is a time stamp. If any element $x > \theta, x \in f(n_i, t_i)$, θ is a threshold defined from the application requirement, and if $g(n_i, l) = 0, n_i$ is the data source node, the error is a node side out of bound error.*

Definition 6 (edge side out of bounds error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, g(n, l)) be a time series record from node n_i, where i is a time stamp. If any element $x > \theta, x \in f(n_i, t_i)$, θ is a threshold defined from the application requirement and if $g(n_i, l)! = 0, n_i$ is the data source node, the error is an edge side out of bound error.*

Definition 7 (node side spike error). *Let $r_i(n_i, t_i, f(n_i, t_i)$, g (n, l)) be a time series record from node n_i, where i is a time stamp. If $|f(n_i, t_i) - f^p(n_i, t_i)|/t_i > \psi, \psi$ is the acceptable changing trend, $f^p(n_i, t_i)$ is the predicted time series with an adopted prediction model, and if $g(n_i, l) = 0, n_i$ is the data source node, the error is a node side spike error.*

Definition 8 (edge side spike error). *Let $r_i(n_i, t_i, f(n_i, t_i)$, g (n, l)) be a time series record from node n_i, where i is a time stamp. If $|f(n_i, t_i) - f^p(n_i, t_i)|/t_i > \psi, \psi$ is the acceptable changing trend, $f^p(n_i, t_i)$ is the predicted time series with an adopted prediction model, and if $g(n_i, l)! = 0, n_i$ is the data source node, the error is an edge side spike error.*

Definition 9 (Aggregation and Fusion error). *Let $r_i(n_i, t_i)$, $f(n_i, t_i)$, g(n, l)) be a time series record from node n_i, where i is a time stamp. If $\Sigma_i|f(n_i, t_i) - f^p(n_i, t_i)|/t_i > \psi* \;\&\&\; \forall|f(n_i, t_i) - f^p(n_i, t_i)|/t_i < \psi$, where ψ^* is a given total acceptable error bound, there is an aggregate and fusion error.*

4 Time-Efficient Error Detection for Big Sensor Data on Cloud

In this section, a cluster-head WSN will be introduced and processed as a kind of complex network system. These complex networks may have non-trivial statistical properties which will influence the data processing strategy on them.

4.1 Scale-Free Sensor Networks Systems

For a WSN with a hierarchical structure, it is a graph denoted as *G(V, E)*, the degree of a vertex V is denoted as $\deg(v)$. We define a function *s(G)* in formula (1).

$$s(G) = \sum_{(u,v)\in E} deg(u)\ deg(v). \tag{1}$$

If the high degree nodes are connected to other high degree nodes in G, we can get formula (2), where the maximum value of *s(H)*, and H are the graphs with degree distribution similar to G. *S(x)* denotes the distribution function corresponding to a probability mass function $\{P_k\}_{k=0}^{\infty}$.

$$S(G) = \frac{s(G)}{MAX}. \tag{2}$$

Because we assume that the sensor network has a hierarchical structure. If $S(G) \to 1$, the graph G is called "scale-free". The classification and prove for the complex networks are as follows.

Suppose there is a graph sequence $\{G_n\}, n[1, +\infty)$, we can calculate the vertices. n is the size of vertices in G_n.

The proportion of vertices with k degree in G_n is noted as $P_k^{(n)}$.

$$P_k^{(n)} = \frac{1}{n}\sum_{j=1}^{n} 1_{\left\{D_j^{(n)}=k\right\}}, \tag{3}$$

In formula (3), $D_i^{(n)}$is the degree of vertex $j \in \{1, \ldots, n\}$ in the graph G_n. The degree sequence of G_n is given by $\{P_k^n\}_{k=0}^{+\infty}$. The random graph process $\{Gn\}_{n=1}^{+\infty}$ is sparse for the $\{p_k\}_{k=0}^{+\infty}$. if formula (4) can be satisfied.

$$\lim_{n\to\infty} P_k^n = p_k. \tag{4}$$

Because the limit p_k in formula (4) is deterministic, the convergence in formula (4) can be taken as convergence in probability or in distribution. And $\{P_k^n\}_{k=0}^{+\infty}$ ends up as 1. In terms of a large value of n, a large number of vertices in G_n have a limited degree. Then, a random graph process with the above feature is called scale-free with an existing exponent τ which can be calculated by formula (5)

$$\lim_{k\to\infty} \frac{log\ p^k}{log 1/k} = \tau. \tag{5}$$

Hence, for a scale-free graph process, its degree converges to a limited probability described in formula (4).

Under some situation, there is too much restriction for the formula (5). For example, when the probability mass function $k- > p_k$ is not smooth, the formula (5) can be replaced with (6).

$$\lim_{k\to\infty} \frac{log\ [1 - S(k)]}{log\ 1/k} = \tau - 1, \tag{6}$$

where $S(X) = (\sum_{x>y} p_y)$ is the distribution according to the function $\{pk\}_0^{+\infty}$. When the formula (7) can be satisfied, we say that a graph process $\{G_n\}_{n=1}^{+\infty}$ has a highly clustered structure.

$$lim_{n\to\infty} C_{G\ltimes} = C_{G\ltimes} > 0 \tag{7}$$

If the formula (7) can be satisfied, the WSN graph G carries strong features of a scale-free complex network as a cluster-head WSN.

Based on the above analysis, the scale-free networks are inhomogeneous and only a few nodes have a large number of links. In real applications, the cluster-head WSN is similar to scale-free networks, which can be described with the scale-free complex networks and has the feature of scale-free networks. In Fig. 2, the instance of scale-free networks and exponential networks are compared. It can be concluded that the scale-free networks have a more clustered hierarchical nodes topology. Central nodes are highly connected by the out-layer nodes has only 1 or 2 links.

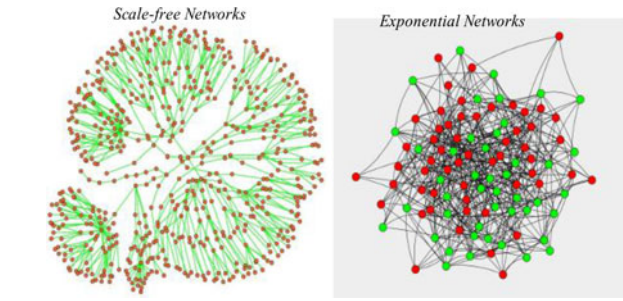

Fig. 2. Examples for scale-free networks and non scale-free networks.

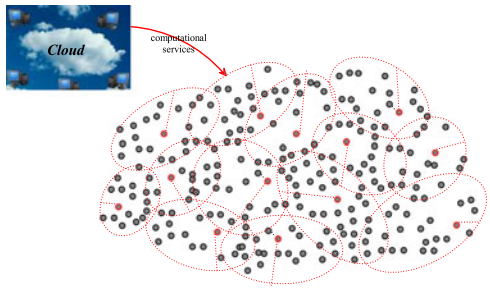

Fig. 3. Cluster based error detection strategy on cloud.

4.2 Model Based Error Detection on Cloud for Sensor Network Big Data

According to the above analysis, it is clear that complex network systems have a similar clustered network topology. During the filtering of big data sets, whenever an abnormal data is encountered, the detection algorithm needs to finish two tasks. They are depicted as two functions here. "$f_d(n/e,t)$" is a decision making function which determines whether the detected abnormal data is a true error. In other words, $f_d(n/e,t)$ has two outputs, "false negative" for detecting a true error and "false positive" for selecting a non-error data. "$f_l(n/e,t)$" is a function for tracking and returning the original error source. With the results from the above two functions, the error detection process can be successfully finalized.

As shown in Fig. 3, there is a complex network and cloud platform for running error detecting algorithms. Without any consideration of network features and data characteristics, the error detection algorithm needs to filter the whole big data set from the network. Whenever, an abnormality defined in Section 3 is encountered, the algorithm will call $f_d(n/e,t)$ and $f_l(n/e,t)$ to traverse the whole network big data set for the final decision making and error source location. However, based on the analysis of scale-free network systems, it has been proved that scale-free networks have a clustering and hierarchical topology. Only a few nodes in the whole network have large sets of links to other nodes. So, based on these nodes, the whole networks can be partitioned into a group of clusters (red circles). If there is certain abnormal data occurs for a certain node k, the high opportunity is that most of the related data for $f_d(n/e,t)$ and $f_l(n/e,t)$ will be located in the clusters where the node k locates. As a result, $f_d(n/e,t)$ and $f_l(n/e,t)$ only need to navigate the related clusters for error detection result. This is because of the fact that except for a few central nodes, most of nodes only have limited links within themselves in their clusters. Hence, the proposed clustering can significantly reduce the time cost error locating and final decision making by avoiding whole network data processing. In addition, with this detection technique, cloud resources only need be distributed according to each partitioned cluster in a scale-free complex network.

5 Algorithms

To deploy the proposed error detection model and identifying the location of the error, the algorithm can be divided into two parts, detection and location. In this section, we will introduce the big data error detection/location algorithm, and its combination strategy with cloud.

5.1 Error Detection

We propose a two-phase approach to conduct the computation required in the whole process of error detection and localization. At the phase of error detection, there are three inputs for the error detection algorithm. The first is the graph of network. The second is the total collected data set D and the third is the defined error patterns p. The output of the error detection algorithm is the error set D'. The details of the error detection algorithm can be found in Appendix B.1, available in the online supplemental material.

5.2 Error Localization

After the error pattern matching and error detection, it is important to locate the position and source of the detected error in the original WSN graph $G(V, E)$. The input of the Algorithm 2 is the original graph of a scale-free network $G(V, E)$, and an error data D from Algorithm 1. The output of the algorithm 2 is $G'(V', E')$ which is the subset of the G to indicate the error location and source. The details of the error detection algorithm can be found in Appendix B.2, available in the online supplemental material.

5.3 Complexity Analysis

Suppose that there is a sensor network system consisting of n nodes. For the error detection approach without considering the scale-free network feature, the error detection algorithm will carry out the error pattern matching and localization with whole network data by traversing the whole data set. Suppose that there is R nodes on the data routing, in the worst case, the detection algorithm without considering the scale-free network feature will be executed $R \times n$ time for error detection and localization, denoted as $O(R \times n), 1 \leq R \leq n$. Anyway, with the hierarchical network topology, the network can be partitioned in to m clusters. Based on our scale-free network definition and our algorithm, in each cluster, the nodes which are involved in error detection will be reduced to n/m on average. In addition, in each cluster, the data values are highly correlated. The data worst case of data traverse times for error detection and localization is determined by $O(R \times \sqrt{n/m})$, $1 \leq R \leq n/m, 1 \leq m \leq n$. Because our scale-free error detection approach limits most of computation within each cluster, the communication and data exchange between clusters can be ignored. Finally, the worst case algorithm complexity

of our scale-free error detection approach can outperform the traditional error detection algorithms.

5.4 Algorithm Calibration on Cloud

The big sensor data error detection and localization algorithms based on the scale-free topology feature of cluster-head networks are designed and analyzed in Sections 5.2 and 5.3. During the development of our scale-free error detection and location algorithm, how to make it more suitable for cloud implementation is already evolved in consideration.

5.4.1 Partition of Sensing Data Set

In order to effectively deploy our proposed algorithm on cloud, the data sets need to be partitioned before feeding to the algorithm on cloud. There are two points should be mentioned when carrying out partitioning. Firstly, the partition process could not bring new data errors into a data set; or change and influence the original errors in a data set. That is different to the previous partition algorithm which normally divides data set according certain application preference or clustering principles. Secondly, due to the scale-free network systems being a special topology, the partition has to form the data clusters according to the real world situation of scale-free network or Cluster-head based WSN. The partition process is as follows.

When the whole data set D is partitioned into $D_i, 1 \leq i \leq q$, we need to guarantee that the distribution of data set in a cluster D_i is similar to D. A sub data set D_i, here can be treated as a point in an m-dimension space, where m is the number of sensor nodes in a partitioned cluster. According to the partition principle, to avoid the new error or error type change, during the process of partition, light weighted error type matching has to be carried out for warning the new abnormalities during the partition. Specifically, the defined variables and functions including $r(n, t, f(n, t), g(n, l)), \psi, \tau, \theta$ in Section 3 will be used again for abnormality warning.

5.4.2 Deployment Strategies for MapReduce

MapReduce is a framework for processing parallelizable problems across huge data sets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogenous hardware). Computational processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing data on or near the storage assets to reduce data transmission. "Map" function. The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node. "Reduce" function. The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve. MapReduce allows for distributed processing of the map and reduction operations.

However, traditional MapReduce is very strict, which limits its application in complex systems, such as WSN. The following is a standard MapReduce example, it counts words. However, our algorithms in error detection and localization are not so ideal and it is hard to directly use one MapReduce to solve perfectly.

The Standard MapReduce Example:

```
function map(String name, String document):
// name: document name
// document: document contents
for each word w in document:
emit (w, 1)
function reduce(String word, Iterator partialCounts):
// word: a word
// partialCounts: a list of aggregated partial counts
sum = 0
for each pc in partialCounts:
sum += ParseInt(pc)
emit (word, sum)
```

Based on the knowledge for MapReduce and its wide applications, three technical changes are commonly adopted to transform the targeting problem for applying MapReduce on it.

1. *Original algorithm −> (embedded in)* ***Map ()/Reduce()***
2. *Partition the task flow of algorithm −> Identify which part of the task flow to generate a MapReduce job −> MapReduce generated result returns back to the task flow*
3. *Complete MapReduce design −> control flow parallelization/data parallelization.*

Based on the analysis of the above three strategies and the complicated flow of our error detection and location algorithms, in our implementation, we adopt different MapReduce strategies in terms of different control flow and data partition in the detection and localization algorithms.

6 Experiments

To verify the time efficiency and the effectiveness of our approach for detecting errors in big data with cloud, experiments are conducted on U-Could (cloud computing environment at the University of Technology Sydney) [12], [13], [14], [15], [16], [18]. There are three purposes for this experiment. 1) Demonstrate that the significant time-saving is achieved in terms of detecting errors from complex network big data sets. 2) Demonstrate the effectiveness of our proposed error detection approach in terms of different error types. 3) Demonstrate that the false positive ratio of our proposed error detection algorithm is limited within a small value.

6.1 Experiment Environment and Process

The U-Cloud system is set up as shown in Appendix C.1, available in the online supplemental material. Four types of data values collected by a real WSN (scale-free complex network system) are used as the testing data set. The total testing data set size is around 2,000,000 KB,

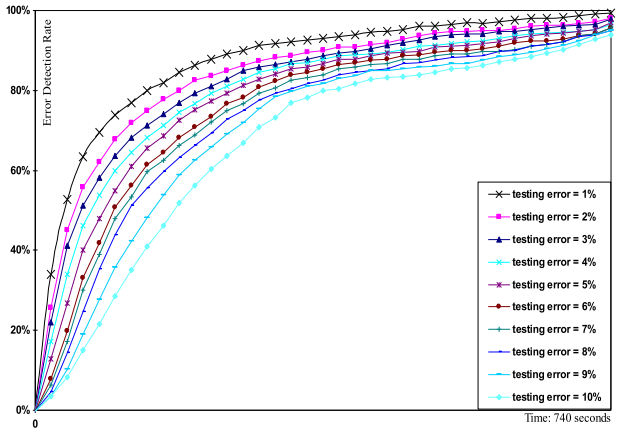

Fig. 4. Time cost for detecting errors from the testing data set.

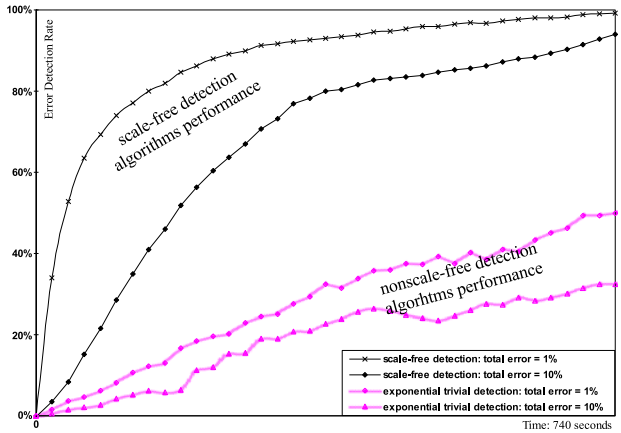

Fig. 5. Comparison of two error detection strategies.

including temperature, sound, light and vibration. Even only considering one node, four types of testing data are gathered with different frequency. In other words, the data sampling from each real world node is heterogeneous. Before the experiment, we conduct the normalization for the testing data set. The normalization process is described in Appendix C.2, available in the online supplemental material.

6.2 Experiment Results

In order to test the false positive ratio of our error detection approach and time cost for error findings, we impose five types of data errors following the definition in Section 3 into the normalized testing data sets with a uniform random distribution. These five types of data errors are generated equally. Hence, the percentage of each type of errors is 20 percent from the total imposed errors for testing.

The first imposed error type is the flat line error. The second imposed error type is out of bound error. The third imposed error type is the spike error. The forth imposed error type is the data lost error. Finally, the aggregate & fusion error type is imposed. By imposing the above listed five types of data error types, the experiment is designed to measure the error selection efficiency and accuracy during the on-cloud processing of data set.

In Fig. 4, the testing results show the time performance of our proposed scale-free error detection algorithm on U-Cloud after 740 seconds. Specifically, 10 different error rates are imposed into the experimental data set and tested independently. The testing error rate changes from 1 to 10 percent in 10 repetitive experiments. After about 100 seconds, the proposed algorithm can detect more than 60 percent errors whatever the testing error rate is within the domain between 1 and 10 percent . During the time duration between 0 and 100 second, all error detection rates increase dramatically with a steep trend. After the time point of 300 second, the error detection rates increase slowly with a flat trend. At the time of 740 second, the proposed error detection algorithm on cloud can find and locate more than 95 percent imposed errors from the testing data sets. When testing error rate is 1 percent , the best performance gains are achieved, as about 99.5 percent total errors detection. With the increase of the testing error rate, the error detection rate decreases.

It also can be found in Fig. 4 that during the first 300 seconds of data processing and error detection on cloud, almost more than 80 percent of total errors are detected whatever the testing error is. This fast detection is due to our scale-free error detection approach which only allocates the cloud computation resources for traversing and processing a small chunk of data instead of a whole data set analysis. With the testing time longer than 740 seconds, most of imposed errors are detected. This result also shows that the algorithm can provide near real time cloud error detection service for most of current scale-free network systems, such as wireless sensor networks.

A comparative experiment between our proposed scale-free big data error detection in WSN and non scale-free error detection algorithms is conducted. As shown in Fig. 5, when the testing data error rate changes from 1 to 10 percent , at any time slot, our proposed scale-free error detection algorithm achieves significant error detection performance gains compared to non scale-free error detection algorithms. Our proposed scale-free detection on cloud can fast detect most of error data (more than 80 percent) after 740 seconds time duration. However, the non scale-free error detection algorithm can only achieve as much as 44 percent error detection rate as the best case. So, it can be concluded from the experiment results in Fig. 5 that the scale-free detection algorithm on cloud for big data can significantly outperform non scale-free error detection algorithms in terms of error finding time cost.

Except for time cost, to measure an error or abnormality detecting algorithm, we also need to consider other statistic metrics for verifying the quality of an error detection algorithm. Suppose that we have n logical "T/F" hypotheses: h_1, $h_2 \ldots, h_i, \ldots h_n$. The number of true null hypotheses is denoted as n_0, an uncertain parameter. Then we can get the number of true alternative hypotheses $n - n_0$. If we further denote the null hypothesis being true as T, we can get T is the number of false positives. Hence we can calculated the false positive rate T/n_0 and the false positive ratio $E(T/n_0)$. In our experiment, T/n_0 and $E(T/n_0)$ depicts how many normal data are selected as errors during the error detecting process. A smaller false positive ratio in the experiment indicates a better accuracy for selecting error data items from the testing data set.

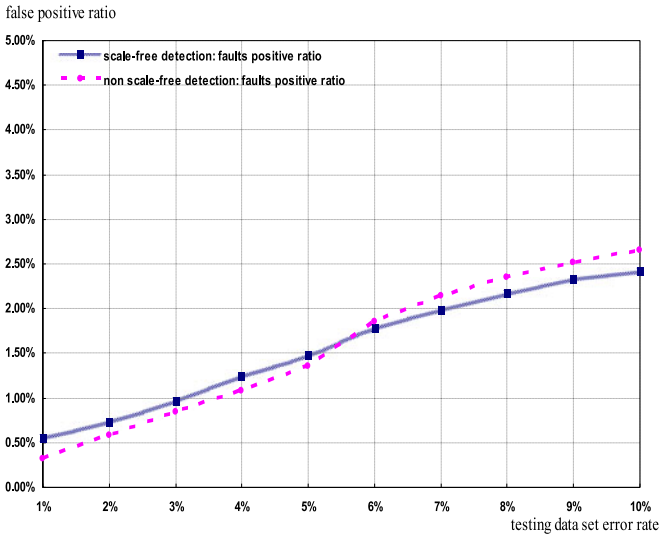

Fig. 6. Comparison of false positive ratio of two error detection strategies.

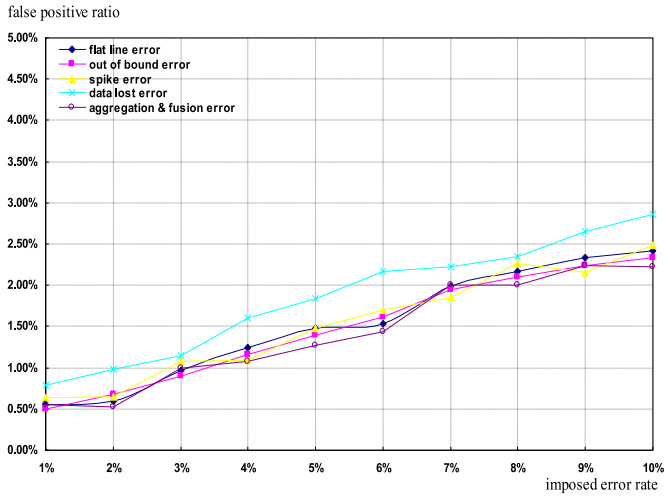

Fig. 7. Comparison of false positive ratios of the scale-free detection algorithm for five error types.

It is demonstrated in Fig. 6 that, with the testing data error rate changes from 1 to 10 percent , our scale-free detection algorithm can achieve similar false positive ration compared to the non scale-free algorithms. Initially, the non scale-free detection performs slightly better because the whole network data traversing and analyzing contribute to improve the decision making correctness for error detection. However, with the increase of error data size in the testing data set, the whole network data traversing and analysis will bring the influence of other error data from other parts of a network into the current error detection decision making. That influence will increase the false positive ratio, which is not expected. As shown in Fig. 6, when the data error rate is larger than 6 percent , our scale-free detection algorithm can outperform the non scale-free algorithm in terms of false positive ratio.

However, the false positive ratio in Fig. 6 is the overall testing result. The individual testing results for detecting each error type with our proposed error detection algorithm are compared in Fig. 7. It can be got from Fig. 7 that "flat line error", "out of bound error", "spike error" and "data lost error" curves of false positive ratio are similar to each other. In other words, our proposed error detection algorithm achieves similar error detection accuracy in detecting the above four types of errors. When it comes to "aggregate & fusion error", the false positive ratio runs slightly higher than the other four types of errors whatever the total imposed error rate is. In other words, the error detection accuracy of our proposed algorithm decreases when encountering "aggregate & fusion error" in the testing data set. The reason is that the "aggregate & fusion error" is caused by the accumulating error effect and multi-hop data communication. Lots of data drifting and data approximation may be involved in the error detection process, which influences the error detecting accuracy of the proposed algorithm for big data on cloud.

Based on the above experiment results and analysis, it can be concluded that our proposed error detection approach for big data processing on cloud can dramatically increase the error detecting speed without losing error selecting accuracy. Especially, when the error rate for a targeting big data set is limited and within a small value (1-10 percent), the algorithm can efficiently detect the error with high fidelity.

7 Conclusions and Future Work

In order to detect errors in big data sets from sensor network systems, a novel approach is developed with cloud computing. Firstly error classification for big data sets is presented. Secondly, the correlation between sensor network systems and the scale-free complex networks are introduced. According to each error type and the features from scale-free networks, we have proposed a time-efficient strategy for detecting and locating errors in big data sets on cloud. With the experiment results from our cloud computing environment U-Cloud, it is demonstrated that 1) the proposed scale-free error detecting approach can significantly reduce the time for fast error detection in numeric big data sets, and 2) the proposed approach achieves similar error selection ratio to non-scale-free error detection approaches. In future, in accordance with error detection for big data sets from sensor network systems on cloud, the issues such as error correction, big data cleaning and recovery will be further explored.

References

[1] S. Tsuchiya, Y. Sakamoto, Y. Tsuchimoto, and V. Lee, "Big Data Processing in Cloud Environments," *FUJITSU Science and Technology J.*, vol. 48, no. 2, pp. 159-168, 2012.

[2] "Big Data: Science in the Petabyte Era: Community Cleverness Required," *Nature*, vol. 455, no. 7209, p. 1, 2008.

[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Comm. the ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[4] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging it Platforms: Vision, Hype, and Reality for Delivering Computing As the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.

[5] L. Wang, J. Zhan, W. Shi, and Y. Liang, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale?" *IEEE Trans. Parallel and Distributed Systems*, vol. 23,, no. 2, pp. 296-303, Feb. 2012.

[6] S. Sakr, A. Liu, D. Batista, and M. Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments," *IEEE Comm. Surveys & Tutorials*, vol. 13, no. 3, pp. 311-336, Third Quarter 2011.
[7] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "A Platform for Scalable One-Pass Analytics Using MapReduce," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'11)*, pp. 985-996, 2011.
[8] R. Kienzler, R. Bruggmann, A. Ranganathan, and N. Tatbul, "Stream As You Go: The Case for Incremental Data Access and Processing in the Cloud," *Proc. IEEE ICDE Int'l Workshop Data Management in the Cloud (DMC'12)*, 2012.
[9] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V.B.N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, and X. Wang, "Nova: Continuous Pig/Hadoop Workflows," *Proc. the ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'11)*, pp. 1081-1090, 2011.
[10] K.H. Lee, Y.J. Lee, H. Choi, Y.D. Chung, and B. Moon, "Parallel Data Processing with MapReduce: A Survey," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11-20, 2012.
[11] "Hadoop," http://hadoop.apache.org, accessed on March 01, 2013.
[12] X. Zhang, C. Liu, S. Nepal, and J. Chen, "An Efficient Quasi-Identifier Index Based Approach for Privacy Preservation over Incremental Data Sets on Cloud," *J. Computer and System Sciences*, vol. 79, pp. 542-555, 2013.
[13] X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen, "A Privacy Leakage Upper-Bound Constraint Based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1192-1202, June 2013.
[14] X. Zhang, T. Yang, C. Liu, and J. Chen, "A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using Systems, in MapReduce on Cloud," *IEEE Trans. Parallel and Distributed*, vol. 25, no. 2, pp. 363-373, Feb. 2014.
[15] C. Liu, J. Chen, T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Kotagiri, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, Sept. 2014.
[16] W. Dou, X. Zhang, J. Liu, and J. Chen, "HireSome-II: Towards Privacy-Aware Cross-Cloud Service Composition for Big Data Applications," *IEEE Trans. Parallel and Distributed Systems*, 2013.
[17] C. Yang, X. Zhang, C. Zhong, C. Liu, J. Pei, K. Kotagiri, and J. Chen, "A spatiotemporal compression based approach for efficient big data processing on cloud," *J. Computer and System Sciences*, vol. 80, no. 8, pp. 1563–1583, 2014.
[18] J. Conhen, "Graph Twiddling in a MapReduce World," *IEEE Computing in Science & Eng.*, vol. 11,, no. 4, pp. 29-41, 2009.
[19] K. Shim, "MapReduce Algorithms for Big Data Analysis," *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 2016-2017, 2012.
[20] "Big Data Beyond MapReduce: Google's Big Data Papers," http://architects.dzone.com/articles/big-data-beyond-mapreduce, accessed Mar. 2013.
[21] R. Albert, H. Jeong, and A. L. Barabasi, "Error and Attack Tolerance of Complex Networks," *Nature*, vol. 406, pp. 378-382, July 2000.
[22] D.J. Wang, X. Shi, D.A. Mcfarland, and J. Leskovec, "Measurement Error in Network Data: A Re-Classification," *Social Networks*, vol. 34, no. 4, pp. 396-409, Oct. 2012.
[23] D. Xiong, M. Zhang, and H. Li, "Error Detection for Statistical Machine Translation Using Linguistic Features," *Proc. 48th Ann. Meeting of the Association for Computational Linguistics (ACL'10)*, pp. 604-611, 2010.
[24] S. Mukhopadhyay, D. Panigrahi, and S. Dey, "Model Based Error Correction for Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 8, no. 4, pp. 528-543, Sept. 2008.
[25] S. Slijepcevic, S. Megerian, and M. Potkonjak, "Charaterization of Lacation Error in Wireless Sensor Networks: Analysis and Application," *Proc. the Second Int'l Conf. Information Processing in Sensor Networks (IPSN '03)*, pp. 593-608, 2003.
[26] K. Ni, N. Ramanathan, M.N.H. Chehade, L. Balzano, S. Nair, S. Zahedi, G. Pottie, M. Hansen, M. Srivastava, and E. Kohler, "Sensor Network Data Fault Types," *ACM Trans. Sensor Networks*, vol. 5, no. 3, article 25, May 2009.
[27] A. Alamri, W.S. Ansari, M.M. Hassan, M.S. Hossain, A. Alelaiwi, and M.A. Hossain, "A Survey on Sensor-Cloud: Architecture, Applications, and Approaches," *Int'l J. Distributed Sensor Networks*, vol. 2013, pp. 1-18, 2013.
[28] A. Sheth, C. Hartung, and Richard Han, "A Decentralized Fault Diagnosis System for Wireless Sensor Netowrks," *Proc. IEEE Second Conf. Mobile Ad-hoc and Sensor Systems (MASS '05)*, Nov. 2005.
[29] M.M.H. Khan, H.H.K. Le, H. Ahmadi, T.F. Abdelzaher, and J. Han, "Dustminer: Troubleshooting Interactive Complexity Bugs in Sensor Networks," *Proc. ACM Sixth Conf. Embedded Network Sensor Systems (SenSys '08)*, pp. 99-112, 2008.
[30] N. Laptev, K. Zeng, and C. Zaniolo, "Very Fast Estimation for Result and Accuracy of Big Data Analytics: The EARL System," *Proc. IEEE 29th Int'l Conf. Data Eng. (ICDE)*, pp. 1296-1299, 2013.
[31] X.L. Dong and D. Srivastava, "Big data integration," *Proc. IEEE 29th Int'l Conf. Data Eng. (ICDE)*, pp. 1245-1248, 2013.
[32] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine learning on Big Data," *Proc. IEEE 29th Int'l Conf. Data Eng. (ICDE)*, pp. 1242-1244, 2013.
[33] A. Aboulnaga and S. Babu, "Workload Management for Big Data Analytics," *Proc. IEEE 29th Int'l Conf. Data Eng. (ICDE)*, p. 1249, 2013.
[34] S. Mukhopadhyay, D. Panigrahi, and S. Dey, "Data Aware, Low Cost Error Correction for Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '04)*, pp. 2494-2497, 2004.
[35] M.H. Lee and Y.H. Choi, "Fault Detection of Wireless Sensor Networks," *Computer Comm.*, vol. 31, no. 14, pp. 3469-3475, 2008.
[36] M.C. Vuranand and I.F. Akyildiz, "Error Control in Wireless Sensor Networks: A Cross Layer Analysis," *IEEE Trans. Networking*, vol. 17, no. 4, pp. 1186-1199, Aug. 2009.
[37] E. Elnahrawy and B. Nath, "Online Data Cleaning in Wireless Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (ACM Sensys '03)*, pp. 294-295, 2003.
[38] M. Yuriyama and T. Kushida, "Sensor Cloud Infrastructure," *Proc. 13th Int'l Conf. Network-Based Information Systems (NBiS)*, pp. 1-8, 2010.
[39] "Sensor Cloud," http://www.sensorcloud.com/, accessed on 30, Aug. 2013.

Chi Yang received the BS degree from Shandong University, Weihai, China, and the MS degree (by research) in computer science from the Swinburne University of Technology, Melbourne, Australia, in 2007.He is currently working toward the full-time PhD degree at the University of Technology, Sydney, Australia. His major research interests include distributed computing, XML data stream, scientific workflow, distributed system, green computing, big data processing and cloud computing.

Chang Liu received the BEng degree in computer science, in 2005, and the MSc degree in information security, in 2008, both from Shandong University, Jinan, China. He is currently working toward the PhD degree from the Faculty of Engineering and Information Technologies, University of Technology, Sydney, Australia. His research interests include cloud computing, scheduling and resource management, cryptography and data security.

Xuyun Zhang received the BS degree and the ME degree in computer science from Nanjing University, Nanjing, China, in 2008 and 2011, respectively. He is currently working toward the PhD degree at the Faculty of Engineering & IT, University of Technology, Sydney. His research interests include cloud computing, privacy and security, social computing, MapReduce and OpenStack.

Surya Nepal received the BE and ME degree from the National Institute of Technology, Surat, India, and the Asian Institute of Technology, Bangkok, Thailand, respectively and the PhD degree from RMIT University, Australia then has been working in CSIRO, Australia. He is a principal research scientist working on service and cloud computing. He has also worked on trust and security aspects of collaboration at CSIRO ICT Centre. At CSIRO, Surya undertook research in the area of multimedia databases, service oriented architectures, and security and trust in collaborative environment. He has several journal and conference papers in these areas.

Jinjun Chen received the bachelor's degree in applied mathematics in 1996, the master's degree in engineering in 1999 from Xidian University, China and the PhD degree in computer science and software engineering, in 2007, from Swinburne. He is an associate professor of Faculty of Engineering and IT, University of Technology, Sydney, Australia. His research interests include cloud computing, social computing, green computing, service computing, e-science, workflow management. He has published more than 100 papers in high quality journals and conferences, including *TOSEM*, *TSE*, and ICSE. He received IEEE Computer Society Outstanding Leadership Awards (2008-2009, 2010-2011), Vice Chancellor Research Award, and many other awards. He is a senior member of the IEEE.