Characterizing Incidents in Cloud-based IoT Data Analytics

Hong-Linh Truong
Faculty of Informatics, TU Wien, Austria
hong-linh.truong@tuwien.ac.at

Manfred Halper*
Alumnus, TU Wien, Austria
manfred.halper@gmx.at

***Abstract*—Systems for big Internet of Things (IoT) data analytics are extremely complex. Different software components at different software stacks from different infrastructures and providers are involved in handling different types of data. Various types of incidents may occur during execution of such systems due to problems in software stacks, the data itself, and processing algorithms. Here incidents reflect unexpected context-specific situations that might happen within data themselves, machine learning algorithms, data analytics pipelines, and underlying big data services and computing platforms. It is important to address any incident that prevents the pipeline running correctly or producing the expected quality of analytics. In this paper, we show the need to characterize incidents for IoT data analytics in the cloud with real-world examples. We characterize incidents based on various aspects in IoT data analytics, including analytics phases, status of data, software services, and stakeholders. We introduce a meta-model for capturing knowledge about incidents.**

I. Introduction

Nowadays a big IoT data analytics has many parts, e.g., (i) collecting data from IoT devices, (ii) pushing the data into data messaging systems, e.g., using IoT data hubs from Amazon, Google and Azure or using Apache Kafka, MQTT, and other pub/sub middleware, (iii) ingesting data from messaging systems to data storage or to streaming data processing engines, and (iv) performing data analysis, e.g., using machine learning (ML) algorithms atop Apache Spark or streaming analysis with Apache Flink. These parts are realized through various pipelines of software services, platforms and infrastructural resources. Thus different software components at different software stacks from different infrastructures and providers are involved in handling different types of data. Due to the complexity of systems and data and the speed of data processing, various types of incidents may occur during execution of such big data analytics systems, where *incidents* indicate unexpected situations that might happen within data themselves, ML algorithms, data pipelines, and underlying big data services and computing platforms[1]. Even running a data analytics pipeline using Apache Nifi[2], Hadoop[3] and Apache Spark with a small number of virtual machines (VMs) can be expensive, especially for small business and research companies/teams. Thus, it is important to address any incident that prevents the pipeline running correctly or producing the expected output.

Characterizing, quantifying and detecting incidents during the execution of cloud-based big IoT data analytics in an end-to-end view across software layers and infrastructures are of paramount importance. However, considering the current support for incident quantification and detection, most works deal with system/service incidents [1], [2], [3], e.g., the failure of VMs running ML algorithms, or with the error of data, e.g., missing values in a time series of IoT data, in a separate manner. Furthermore, many tools for performance monitoring and analysis of cloud services exist, but there is a lack of tools for understanding IoT data analytics incidents. Mostly, the data analyst, the developer of different software stacks and data analytics phases, and the operator of the analytics and infrastructures manually and separately deal with different types of incidents. Another important aspect is that studied incidents are mostly not data-centric (due to errors of IoT data or the processing of IoT data), e.g., they focus on security and system incidents [4]. While incidents might be detected in different specific points, e.g., within a cloud service and within a data processing task, without a coherent and correlated end-to-end cross-system view, we cannot understand and optimize incidents for the data analytics as a whole. An incident might be due to errors propagated through multiple middleware, libraries, and services across different layers and infrastructures within a data analytics pipeline.

In this paper we focus on the first aspect of incident quantification and detection for big IoT data analytics by characterizing incidents. We analyze incidents to determine their key relevant information, focusing on big IoT data analytics. We make the following contributions:

- analysis of incidents in IoT big data systems
- a classification of possible incidents in big IoT data analytics and a graph-based model for capturing knowledge about incidents
- analysis of data-centric incidents and monitoring techniques

Our contributions will help fostering the developing new incident monitoring and analytics tools for IoT big data analytics,

*This work was performed when the author was a master student at TU Wien

[1]The IT Infrastructure Library (ITIL) defines an incident *"as an unplanned interruption to an IT service or reduction in the quality of an IT service or a failure of a Configuration Item that has not yet impacted an IT service"*.

[2]http://nifi.apache.org

[3]https://hadoop.apache.org/

which are currently under researched.

The rest of this paper is structured as follows: Section II presents the motivation of our work. Section III characterizes incidents in big IoT data analytics and provides an incident information meta-model. Section IV presents related work. We conclude the paper and outline our future work in Section V.

II. Motivation

A. Motivating Examples

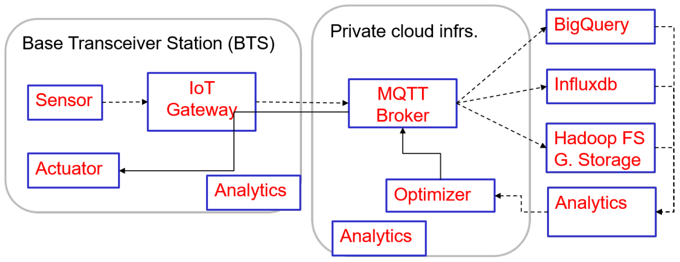

Fig. 1. A simplified big IoT data analytics system for BTS monitoring

Let us consider the case in which we use IoT and big data analytics to analyze telco infrastructures, such as equipment and operations of Base Transceiver Stations (BTS). Shown in Figure 1, a big data analytics for that case mostly relies on: (i) IoT or data collectors for collecting data, (ii) connectivity for pushing the data to the fog/edge and cloud services, (iii) and the fog/edge and cloud systems (public or private)[4]. The data analyst and developer identify and select suitable libraries, services and systems for building data pipelines. However, at runtime, an outstanding problem across software layers and systems is how to deal with data-centric incidents across complex data sources, and systems within the context of specific analytics. Let us see the following cases:
Case 1: We monitor BTS in a large-scale telco network of a leading mobile provider. We need to gather location data together with many IoT data to support clustering of behavior of equipment. Due to a minor error in the location data, several stations are wrongly clustered. This error is detected only by human analyst. This error may be easily detected by examining location information but not when we need to correlate IoT data with other dynamic data, such as quality of voice and text traffics from NodeB[5]. This might create errors in clustering tasks (due to the error of the data not the clustering) and might also create system errors, e.g., the failure of the clustering algorithm due to errors in data, or the failure of VMs due to the lack of memory.
Case 2: We performed the analytics of logs from various parts of the telco network, like NodeB, Radio Network Controller, and Base Station Controller. These parts are from specific systems of different vendors, including Huawei, Ericsson and Nokia, so logs are different. Each system has voluminous log data, which are extracted and then merged together. Due to some mistakes in tasks for extracting logs and merging data, tasks for data analytics have produced some wrong results. However, this incident is not detected until data analysts carefully examine the results and debug the Apache Spark tasks. This cannot be detected by existing functions in ML libraries. Even the developer can write some customized functions to check the problems, she/he does not have a systematic way to collect and relay such problems for different parts of the data analytics pipelines.

We observe such situations in our real-world applications for big data analytics by utilizing ML algorithms for IoT data, combined with Apache Nifi, Apache Hadoop, Apache Spark MLib, and Elasticsearch[6]. However, these problems are generic. State-of-the-art tools do not provide features for us to quantify and detect such incidents across the whole big data analytics. Current cloud monitoring tools [5] lack support for incident detection for big IoT data analytics.

B. Research Statements

The first research issue is that we need to *identify and classify potential incidents* in big IoT data analytics, e.g., related to IoT data sources, status of data (e.g., data in transit or data in processing), big data system services, or to stakeholders. To have a classification and means to manage incidents help us to determine relevant measurements and logs for detecting and evaluating incidents. The ultimate issue is that the data analyst, developer and provider must have a coherent view on types of incidents and relevant data for understanding the incidents. Current cloud and big data monitoring tools are very much at the system level (e.g., performance and failure of services and infrastructures), while we have to manually carry various error-prone, time-consuming tasks to monitor and analyze data incidents within analytics pipelines. Therefore, the second research issue is to *determine important, focused data-centric incidents* in IoT data analysis systems, fostering the development of new development of new performance measurement and monitoring tools for understanding issues related to data and its impact on performance and failure of services and infrastructures.

III. Characterization of Incidents in Cloud-based Big IoT data analytics

A. Cloud-based Big IoT data analysis pipelines

Conceptually, typical (big) data analysis pipelines have different phases: collecting data, ingesting data, data analysis and data visualization [6]. In this paper, we focus on big IoT data pipelines in which data is collected from IoT networks [7] but analyzed in the cloud. Therefore, a typical way of collecting data is that suitable protocols and message brokers, such as MQTT or AMQP, and Kafka, will be used to push data from IoT devices to the IoT data hubs in the cloud, as shown in the motivating example in Section II. From the IoT data hubs/message brokers data will be extracted to storage for

[4]The BTS scenario is extracted from the complex real-world BTS implemented from our collaborative industry. A simplified version for research purpose is currently provided at http://github.io/rdsea/IoTCloudSamples.

[5]https://en.wikipedia.org/wiki/NodeB

[6]https://www.elastic.co

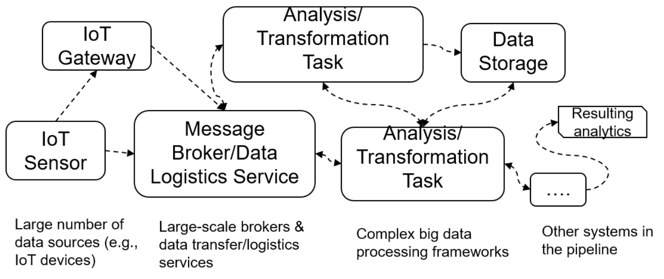
IoT Gateway
Analysis/ Transformation Task
Data Storage
Resulting analytics
IoT Sensor
Message Broker/Data Logistics Service
Analysis/ Transformation Task
....
Large number of data sources (e.g., IoT devices)
Large-scale brokers & data transfer/logistics services
Complex big data processing frameworks
Other systems in the pipeline

Fig. 2. A simplified view of main components in big IoT data analytics

batch processing later or be pushed into streaming processing components. We can use various frameworks and architectures to support this, such as, from industries including Google[7], Amazon IoT[8], Azure IoT[9], Predix[10], and from academics like [8]. Figure 2 abstracts a simplified view of main components in big IoT data analytics. With this model, incidents can occur as long as IoT data are being pushed to message brokers.

We see that incidents might be happened at the infrastructure and platform levels, e.g., within the underlying computing resources and middleware used for the pipelines, as well as at the application level, e.g., within instances of ML algorithms or data load/extract/transform tasks employed in analytics pipelines. Based on the NIST Big data architecture[11], existing big data view [6], and the concept of IoT Cloud, we focus on the four following phases, *Acquisition*, *Preparation*, *Analysis*, and *Delivery*. These phases are not sequential; instead they can be interwoven. We explain possible incidents in these phases through examples in the following subsections.

1) Acquisition phase: incidents might happen when we collect data from IoT devices. This can happen within sensors sending data, connectivity between sensors and messing brokers/IoT data hubs, or within the brokers and data hubs. Let us consider an example of IoT sensors sending data to an MQTT broker. In one design, the data acquisition tasks are realized through sensors, the MQTT-based message broker, and the MQTT-based connectivity protocol, whereas these tasks spread in IoT networks (sensors) and edge/fog/cloud systems (e.g., the broker). Incidents might happen when we have many sensors which suddenly increase the sending rate (e.g., by changing `setUpdateRate()` of sensors at runtime), leading to a sudden voluminous data sent to the broker to slowdown or even kill the broker.

2) Preparation phase: incidents might happen when we load/extract data (e.g, from data storage and message brokers) and transform the data to the form ready for analysis tasks. Let us consider the following example in which we need to load and merge various types of data – quality of service in NodeB, quality of traffic, and alarms[12] – before we perform analysis, such as clustering NodeBs/BTSs:

```
#...
#... input data

inputQoSData="hdfs://spark-m/data/rawdata/KPI-QoS-Data/KPI-
      QoS-Nodeb-DN*.csv"
inputTrafficData="hdfs://spark-m/data/rawdata/KPI-Traffic-
      Data/KPI-luu*.csv"
dfqos =spark.read.csv(inputQoSData,header=True,inferSchema=
      True)
dftraffic =spark.read.csv(inputTrafficData,header=True,
      inferSchema=True)
inputAlarmData="hdfs:///spark-m/data/rawdata/Alarm-nodeB-DN
      /Alarm_nodeB_DN*.csv"
dfalarm =spark.read.csv(inputAlarmData,header=True,
      inferSchema=True)

# ... preparation of data by merging
# filter data
dfqos2 = dfqos.select(['NODEBNAME','VOICE CSSR (%)'])
dftraffic2=dftraffic.select(['NODEBNAME','SHO ATT']).show()
dfalarm2 =dfalarm.withColumnRenamed("RNW Object Name", "
      NODEBNAME")
dfalarm3=dfalarm2.select(['NODEBNAME','Severity'])
#join
join1 = dftraffic2.join(dfqos2,"NODEBNAME")
join2 = join1.join(dfalarm2,"NODEBNAME")
cleanjoin =join2.distinct()
#...
# analytics
# ....
```

A simple incident might occur when wrong input data files are specified, leading to mismatch of input data (e.g., times among various data sources are not related). Incidents might occur in `select` and `join` due to missing data in the above-mentioned data. This might lead to the failure of the analysis tasks or the wrong result produced by the analysis.

3) Analysis phase: incidents can happen during analysis tasks, which examines data based on various statistical, machine learning or user-defined algorithms. For example, given the previous example in the *Preparation phase*, after the data preparation, we can run statistic analyses to find top problematic NodeBs or carry out clustering algorithms to group NodeBs based on their quality of services. In such analysis tasks, incidents might occur when the underlying system, e.g., Apache Spark, has not enough resources, e.g., run out of memory or compute nodes, to run various analysis tasks. Note that in many cases it is difficult to separate between Analysis and Preparation tasks. For example, in the above-mentioned example in the Preparation phase, `select()` and `join()` data frames might be interwoven with statistics functions.

In other cases, we also see separate sub pipelines for Preparation and Analysis phases using different techniques and systems. For example, let us consider the following sample of alarms and location of stations:

```
//sample data of alarms
//station_id,alarm_id,alarm_number,start_time_window,
      end_time_window
1161114077,309,16,2017-04-16 00:00:00,2017-04-16 01:00:00

//sample data of location
//station id,code,name,name_eng,address,description,
      latitude,longitude,status, ...
```

[7]https://cloud.google.com/solutions/architecture/real-time-stream-processing-iot

[8]https://aws.amazon.com/iot-platform/

[9]https://docs.microsoft.com/en-us/azure/iot-suite/iot-suite-what-is-azure-iot

[10]https://www.predix.io/

[11]https://bigdatawg.nist.gov/

[12]Examples are based on real data obtained from NodeB in BTS, but we simplified the code to illustrate our examples.

```
1161114077,DLMD15,DL Mdrak5,DL Mdrak5,"Thon Ho,...    ak Lak
    .",IMSV2–REMOVE,12.73235,108.7538,1,....
```

Preparation tasks are used to enrich results from a sub data analysis pipeline and then the enriched results can be used by another sub analysis pipeline, shown in Figure 3. In this example, various tools/services are used, such as Apache Spark, Elasticsearch, and Google Storage. Therefore, incidents might be propagated from one pipeline to another pipeline. For example, a wrong data within the location would lead to wrong clustering results of NodeBs/BTS based on alarms and location.

4) Delivery phase: incidents occur when we perform transfers of data and results, e.g., using message brokers and visualization services. Consider the following example in which we ingest the data into BigQuery. A problem in the network could prevent us to store data, leading to missing data for analytics.

```
//initialize bigQuery
const bigQuery = BigQuery({
        projectId: config.projectId,
        keyFilename: path.join(__dirname, 'keyfile.json'),
    });

//implementing function to ingest data to big query
function insert(topic, data){
    if(topics[topic]){
        return tables[topics[topic]].insert(data).catch((
            err) => {
        logger.error(`failure inserting into bigQuery
            object: ${JSON.stringify(data)}`);
        logger.error(err);
        });
    }
}
```

B. Characterizing incidents

Characterizing incidents identify possible data sources, monitoring techniques and analytics for incidents. This will help to build sufficient knowledge for developing monitoring and analytics of incidents. In complex big data analytics, we tend to discuss abstract phases, such as in [6], as each phase groups a set of relevant tasks and often relies on the same system to do these tasks. However, in addition to associate incidents according to phases in big data analytics, we need also to identify incidents based on (i) software layers (e.g., systems and infrastructures, services, algorithm libraries, applications), (ii) types of systems (IoT, edge or cloud), (iii) types of data, (iv) connectivity (e.g., error due to data transfers), and (v) dependencies among the above-mentioned types.

To characterize incidents, we focus on the following important aspects:

- Context dimension: we utilize existing context dimensions: Where, When, What and How [9]. `Where` indicates the location of the incident. `What` indicates affected areas. `When` indicates the time when an incident might happen. `How` indicates cause and contributing factors.
- Phases in big data analytics: We include basic phases of big IoT data analytics mentioned in Section III-A.
- Data, software and infrastructures in big IoT data analytics: we cover both platforms and application services. For example, we have analysis services/frameworks (e.g., Apache Spark), messaging services (e.g., RabbitMQ, Kafka or MQTT), storage resources (e.g., Amazon S3 or Hadoop FS), computing resources (e.g., Virtual machines and docker containers), and network connectivity (e.g., MQTT, AMQP and HTTP) between them.

1) Where – Location of incidents: The locations are related to software stacks as well as where in existing phases of IoT data analytics. Figure 4 outlines a simplified view of big data analytics components (with different software layers and systems) and where we should collect incident related data. Data collected for incidents will be based on, e.g., the type of data, within middleware like data brokers and data storage services, within analysis tasks, instances of ML libraries, and underlying machines/containers. These types of data must be aggregated for specific analytics, as different analytics might share various services and resources. The data will be analyzed together with the dependency graph of the big data analytics; the graph covers both application- and system-specific components in the data analytics. Another perspective in locating incidents is to link the incidents to data analysis pipelines. We note that a pipeline might consist of sub pipelines. Furthermore, in many cases, we might not be able to determine the exact location of incidents within a large pipeline but we could locate it within a certain sub pipeline. This way will also help us to identify how far potential incidents could be propagated through the analytics as well as to attribute incidents to appropriate service providers, which are utilized for certain pipelines.

2) What – types of incidents: Concrete incidents are context-specific as whether some errors and faults reduce the quality of analytics depend on system and user-specific constraints. However, evaluating incidents is based on general measurements, events and monitoring data that should be identified. In general, typical incidents in contemporary cloud and IoT, such as performance and system failure [1], can be part of incidents in IoT data analytics. One important aspect of incidents in IoT data analytics is related to data and the capability to handle big data by IoT and cloud services. From that view, we can classify incidents into: Data Incident, Network Incident, Storage Incident, Processing Incident, and Combined Incident. Furthermore, various sub-classes of incidents can be developed. For example, *Data Quality Incident*, used to indicate incidents due to data quality, can be a sub type of Data Incident. Typical incidents due to performance issues, *Performance Incident*, can be associated with Storage Incident and Processing Incident.

3) When – Time and frequencies of incidents: One aspect of incident classification and knowledge representation is time and frequencies. This will be captured using events (time) and quantitative numbers. The values of *When* can only be obtained through the monitoring and analysis. Based on analytics, we can also represent and update patterns about times and frequencies. In general, time and frequency characteristics could be linked to properties of big data (e.g., volume, velocity, veracity and variety). Another example is that incidents might occur at the early stage in the pipeline. For example during

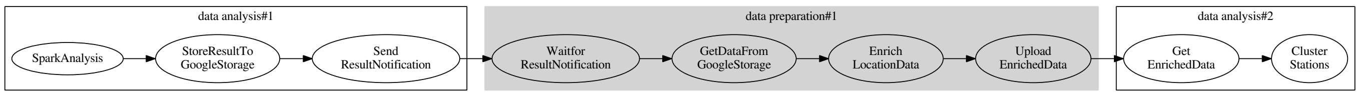

Fig. 3. Data Preparation and Analysis pipelines

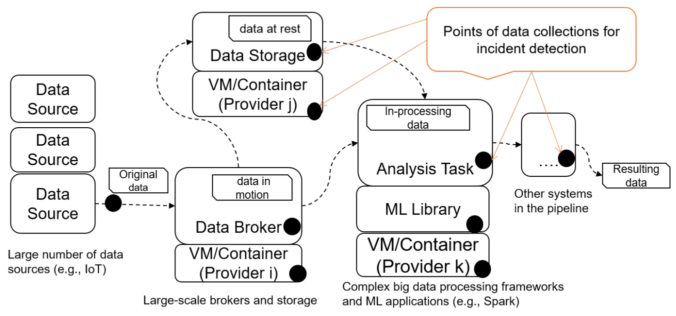
Data Source
Data Source
Data Source
Large number of data sources (e.g., IoT)
Original data
data at rest
Data Storage
VM/Container (Provider j)
data in motion
Data Broker
VM/Container (Provider i)
Large-scale brokers and storage
Points of data collections for incident detection
In-processing data
Analysis Task
ML Library
VM/Container (Provider k)
Complex big data processing frameworks and ML applications (e.g., Spark)
....
Other systems in the pipeline
Resulting data

Fig. 4. Illustrating places where incidents might occur

the Acquisition phase we miss data. However, due to various constraints in big IoT data analysis, the incident might be detected only at the end of the pipeline. Figure 5 presents one example of this case with alarms of electricity in BTS. Alarms occurred very often in BTSs then they did not appear later on. It might not be true that we do not have alarms any more but it is likely that we miss data during the Acquisition phase.

4) Phases and Stakeholders in Incident Analytics: There are mainly possible stakeholders related to incidents in big IoT data pipelines. In characterizing incidents, we identify which of following stakeholders are relevant to incidents:

- Data Provider: creates, aggregates and transforms data.
- Data Engineer: creates data models and data systems for IoT data.
- Service Provider: provides and operates tools, platforms, services and know-how for IoT and cloud services.
- Data End-user: utilizes the result of big data analytics pipelines.
- IoT Data Analyst: runs data analytics pipelines atop existing IoT and cloud services and infrastructures.
- Software developer: develop algorithms, technologies, methodologies, business models, products and services.
- Regulator: regulates governance and compliance rules for data.

An important note is that these stakeholders might not belong to the same organization. They are also responsible only for a part of the data analytics pipeline. Therefore, their view and work on monitoring and analysis of incidents are different.

C. Knowledge about Incidents

1) Meta-model of Incidents: From the above-mentioned characteristics, we classify incidents using a meta-model. Figure 6 provide an overview of information about incidents. `Incident` is used to capture information about incidents, which can be classified into sub types like `DataIncident`, `ProcessingIncident`, `NetworkIncident`, `StorageIncident`, and `CombinedIncident`. `Incidents` can be linked to `AnalyticsPhase`, `Stakeholder`, `SoftwareComponent`, different states of data `DataInMotion`, `DataAtRest`, `DataInAnalysis`. Incidents have IncidentMonitoringData being collected for evaluating the severity of incidents.

Based on the meta-model, we provide an implementation of the knowledge base graph databases. We use Neo4J[13] to capture incident classes and their relationships.

2) Runtime versus prior knowledge about incidents: From the classification, we can obtain many types of prior knowledge about the incidents, while other information will be only obtained at runtime and through incident analytics. This will be done using different monitoring and analysis techniques for incidents. To store concrete incidents for particular deployed IoT data analytics systems, we use a document-based system using MongoDB, which stores concrete (high-level) information sources related to incidents, such as links to log files, etc. Note that many other sources of information for incidents might be stored in different systems, such as Hadoop file systems and Google Storage, due to the diversity of raw data relevant to incidents[14].

IV. Related Work

Well-established data cleansing processes [10] can help reducing errors in data, reducing incidents. But they are not means for incident detection. We need mechanisms to detect and report data errors in correlation with other types of errors. Chua discusses about dangerous data and how to detect them [11]. But detecting data problems is just a component of incident analytics in big data. Abedjan and others discuss about error detection in data [12]. But it is just related to data sources and data while incident detection is a complex matter dealing with many inter-dependent data and system errors. Certain key challenges in big data management, including machine learning, are mentioned in the work of Polyzotis [13], although they focus more on data aspects, like validation and cleansing. The incident quantification and detection in this proposal are exactly for solving such challenges. There are several methods for cleaning and correcting errors within data [14]. They are limited to errors within data but, in general, they do not focus on detection of incidents (due to errors) through big data analytics. Tools and techniques for monitoring cloud systems and applications are many, from the research [5] and industries, such as Amazon Cloudwatch,

[13]https://neo4j.com/

[14]How to monitor, analyze and store concrete incidents for particular IoT data analytics systems is out of the scope of this paper.

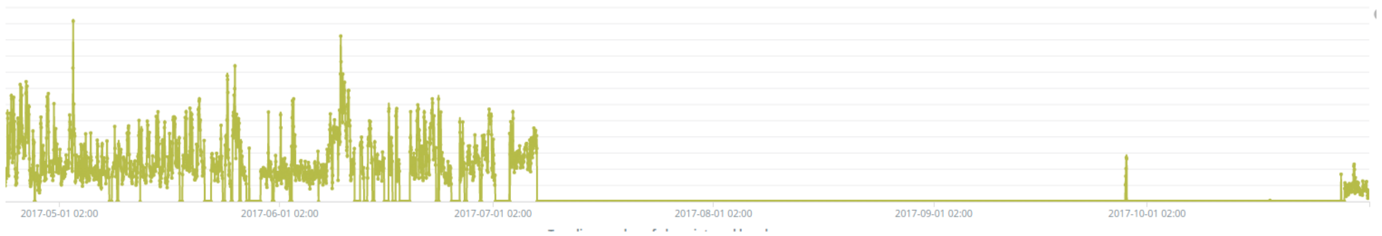

Fig. 5. Abnormal analytics results indicating some potential incidents in IoT data Acquisition, Preparation or Analysis.

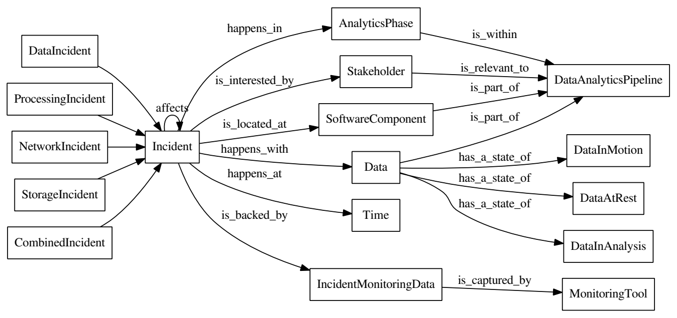

Fig. 6. Main types of information in classification of incidents

Stackdriver, and Prometheus. They enable many generic system monitoring features, thus we can use them for capturing logs and transferring monitoring for incidents in big data. However, they are not designed for incidents in big data. We can leverage these tools for capturing certain types of data for incident analytics across layers and across systems for big data in our cases. Similarly, we can utilize existing techniques for processing logs [15] to extract information for incident analytics, but these works are not for incidents in big data analytics ML techniques for detecting errors has been widely used. Our work is not focused on ML algorithms. However, based on incident classification and relevant monitoring data, our incident analytics will leverage ML techniques.

V. Conclusions and Future Work

Motivated by an increasing need of understanding potential incidents in IoT data analytics in order to guide the development of incident monitoring and analytics, in this paper we present a detailed characterization of incidents in IoT data analytics. We have presented a classification, proposed knowledge based and performed detailed analytic analysis of potential incidents. Our work is just at an early stage as the knowledge about incidents should be used for monitoring and analysis of incidents. Our current focuses are not only to improve the classification and characteristics of incidents but also to develop monitoring and analysis techniques for capturing and evaluating incidents. We will update our further results at https://github.com/rdsea/bigdataincidentanalytics.

Acknowledgment: Various examples shown in the paper are based on data and code extracted from our industry collaboration. Students also contribute to the writing code examples. We thank Huong Ngo, Khiem Ta and Lingfan Gao for their support.

References

[1] S. Sarkar, R. Mahindru, R. A. Hosn, N. Vogl, and H. V. Ramasamy, "Automated incident management for a platform-as-a-service cloud," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, ser. Hot-ICE'11. Berkeley, CA, USA: USENIX Association, 2011.

[2] B. Grobauer and T. Schreck, "Towards incident handling in the cloud: Challenges and approaches," in *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '10. New York, NY, USA: ACM, 2010, pp. 77–86.

[3] T.-F. Forti and V. I. Munteanu, "Topics in cloud incident management," *Future Gener. Comput. Syst.*, vol. 72, no. C, pp. 163–164, Jul. 2017.

[4] N. H. Ab Rahman and K.-K. R. Choo, "A survey of information security incident handling in the cloud," *Comput. Secur.*, vol. 49, no. C, pp. 45–69, Mar. 2015.

[5] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Survey cloud monitoring: A survey," *Comput. Netw.*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013.

[6] J. Klein, R. Buglak, D. Blockow, T. Wuttke, and B. Cooper, "A reference architecture for big data systems in the national security domain," in *Proceedings of the 2Nd International Workshop on BIG Data Software Engineering*, ser. BIGDSE '16. New York, NY, USA: ACM, 2016, pp. 51–57.

[7] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1 – 31, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414003168

[8] Z. Khan, A. Anjum, and S. L. Kiani, "Cloud based big data analytics for smart future cities," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, ser. UCC '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 381–386. [Online]. Available: http://dx.doi.org/10.1109/UCC.2013.77

[9] D. R. Morse, S. Armstrong, and A. K. Dey, "The what, who, where, when, why and how of context-awareness," in *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '00. New York, NY, USA: ACM, 2000, pp. 371–371.

[10] J. Van den Broeck, S. Argeseanu Cunningham, R. Eeckels, and K. Herbst, "Data cleaning: Detecting, diagnosing, and editing data abnormalities," *PLOS Medicine*, vol. 2, no. 10, 09 2005. [Online]. Available: https://doi.org/10.1371/journal.pmed.0020267

[11] C. E. H. Chua and V. C. Storey, "Dealing with dangerous data: Part-whole validation for low incident, high risk data," *J. Database Manage.*, vol. 27, no. 1, pp. 29–57, Jan. 2016.

[12] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang, "Detecting data errors: Where are we and what needs to be done?" *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 993–1004, Aug. 2016.

[13] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 1723–1726. [Online]. Available: http://doi.acm.org/10.1145/3035918.3054782

[14] H. Wang, M. Li, Y. Bu, J. Li, H. Gao, and J. Zhang, "Cleanix: A parallel big data cleaning system," *SIGMOD Rec.*, vol. 44, no. 4, pp. 35–40, May 2016.

[15] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 117–132.