Lujic, Ivan; Truong, Hong Linh

Architecturing elastic edge storage services for data-driven decision making

Published in:
Software Architecture - 13th European Conference, ECSA 2019, Proceedings

DOI:
10.1007/978-3-030-29983-5_7

Published: 01/01/2019

Document Version
Peer reviewed version

Please cite the original version:
Lujic, I., & Truong, H. L. (2019). Architecturing elastic edge storage services for data-driven decision making. In T. Bures, L. Duchien, & P. Inverardi (Eds.), *Software Architecture - 13th European Conference, ECSA 2019, Proceedings* (pp. 97-105). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 11681 LNCS). https://doi.org/10.1007/978-3-030-29983-5_7

Architecturing Elastic Edge Storage Services for Data-Driven Decision Making

Ivan Lujic[1 (✉)] [0000−0002−8564−6040] and Hong-Linh Truong[2] [0000−0003−1465−9722]

[1] Institute of Information Systems Engineering, TU Wien, Austria
`ivan.lujic@tuwien.ac.at`
[2] Department of Computer Science, Aalto University, Finland
`linh.truong@aalto.fi`

Abstract. In the IoT era, a massive number of smart sensors produce a variety of data at unprecedented scale. Edge storage has limited capacities posing a crucial challenge for maintaining only the most relevant IoT data for edge analytics. Currently, this problem is addressed mostly considering traditional cloud-based database perspectives, including storage optimization and resource elasticity, while separately investigating data analytics approaches and system operations. For better support of future edge analytics, in this work, we propose a novel, holistic approach for architecturing elastic edge storage services, featuring three aspects, namely, (i) data/system characterization (e.g., metrics, key properties), (ii) system operations (e.g., filtering, sampling), and (iii) data processing utilities (e.g., recovery, prediction). In this regard, we present seven engineering principles for the architecture design of edge data services.

Keywords: Edge data service · Architectural design · Edge computing · Adaptation · Service computing · IoT · Engineering.

1 Introduction

The introduction of edge computing can help dealing with time sensitive requirements for accurate decisions based on Internet of Things (IoT) data [12]. Unlike scalable cloud data repositories, edge systems have limited storage capacity, whereas certain amount of IoT sensor data have to be stored and processed in proximity of the data sources [13]. Consequently, any edge data service must store only the most relevant data for edge analytics (streaming or batch), whereas non-relevant data have to be either discarded or moved to cloud data centers. But the *relevancy* is determined by analytics contexts: these new edge infrastructure conditions and new application analytics requirements, regarding explosive growth of IoT data, force us to explore novel architectural design and further implementations critical for elastic edge data services. By investigating edge data services, we consider strategies, methods, mechanisms and operations for handling and storing constantly generated data at the network edge. We observe that even *within a single edge analytics system*:

(**O1**) IoT data are categorized into different model types representing multi-model data, in particular near real-time streaming data and log-based data, thus, requiring different storage types and governance policies. They also include different *significance* levels regarding to storage and edge analytics, especially for critical applications, such as healthcare [6] (e.g., keeping the most important data close to the data source) and smart manufacturing [11] (e.g., keeping significance levels among data streams coming from industrial equipment for maintenance purposes). Hence, *all applications and sensors do not have equal importance*;

(**O2**) Different IoT sensors include *various errors* such as missing data, outliers, noises and anomalies, affecting the designs of edge analysis pipelines and corresponding differently to decision making processes. In this context, incomplete and noisy data can be critical, e.g., for traffic-dependent near real-time route guidance [9], but can be tolerated by intelligent weather forecasts [8];

(**O3**) Data from different IoT sensors appear with *different data generation speed, consequently producing different data volumes for the same time interval.* Simultaneously, different types of monitored sensors require different data volumes to make meaningful analytics. In systems like smart cities, it is crucial, for example, to have big amount of frequent traffic measurements for managing traffic flow in real-time. On the other hand, due to lower volatility, a weather station can require much less data volumes from its sensors for accurate predictions.

Currently, all these highlighted issues are solved outside edge storage services. Solutions for these issues are not included in existing designs of edge data services because, as one might argue, such issues are *analytic context-specific*. However, we argue that they are generic enough that can be customized and must be incorporated into the design of (new) edge data storage systems. These observations indicate crucial changes for enhancing traditional approaches, which have assumptions on consistent low latency, high availability and centralized storage solutions, that cannot be generalized to the edge storage services and unreliable IoT distributed systems. Our first step in solving the above-mentioned issues is to focus on architectural requirements and designs. This paper will contribute: (1) a detailed analysis of edge storage services with application-specific edge analytics support and different utilities and analytics requirements; (2) a specification of necessary principles for engineering highly customized software-defined elastic storage services for dynamic data workload characterizations at the edge.

2 Motivation

In the IoT sensor environment, such as an exemplified university smart building shown in Figure 1, we can observe data workloads from different IoT applications and decide whether to (1) push data to the cloud data storage, (2) keep relevant data for local edge analytics or (3) discard data if they are not useful for future analytics. In the first case, traditionally, all data are transferred to resource-rich cloud data centers where storage and compute intensive workloads can be handled, resulting in necessary control commands for IoT actuators. However, increasing data streams and latency requirements arising from IoT applications makes distant cloud data transfer often impractical. Recent solutions for making crucial fast decisions in IoT systems have increasingly used edge nodes.

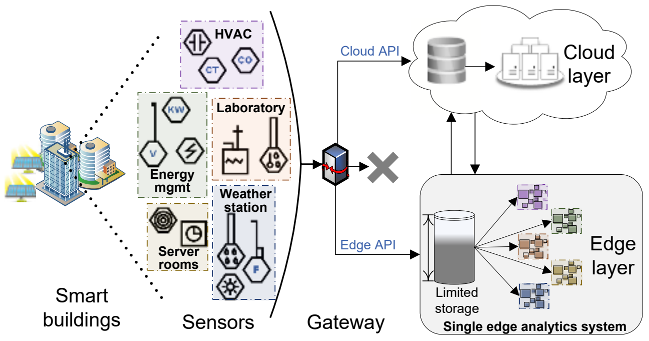

Fig. 1. Traditional single analytics system for university smart buildings use case

In an IoT system, such as a university smart building equipped with many sensors measuring internal subsystems, it is obvious that data from HVAC (Heating, Ventilation, and Air Conditioning) sensors do not have the same importance as data from smart meters and solar panels essential for energy management (O1); incomplete data from weather stations can occur due to external conditions while missing data coming from server room sensors can be caused by some internal failures (O2); an energy management subsystem has higher data generation frequency than a laboratory subsystem (O3). Accordingly, each of these subsystems requires different approach to sensor data analysis, although the same edge storage system is used to integrate data for edge analytics. In addition, limited storage capacities at the network edge prevents us from keeping all generated data. In the third case, due to the limited underlying network infrastructure, some data can be filtered or reduced to save bandwidth usage and storage space, but impacting later degradation of Quality of Service (QoS).

Edge analytics have to meet certain quality of analytics, including amounts of data available, timely decisions and certain levels of data accuracy. Therefore, we must identify which data should be kept at the edge nodes, how long should data be stored, and which processing utilities can assist these problems, providing ability to access the right data at the right time to make data-driven decisions.

3 Engineering Principles for Edge Data Services

Regarding three important aspects of edge storages, namely; edge data/system characterization, application context and edging operations, we present seven principles as guidelines for engineering of elastic edge storage services.

P1: Define and Provide Needed Metrics. To enable efficient customization and adaptation among elements of edge storage systems, it requires a clear definition and flexible monitoring of end-to-end metrics regarding data workloads, application context and system activities.

How: Figure 2 shows end-to-end monitoring metrics that can assist in elastic edge storage management. There are metrics present in four stages of data life

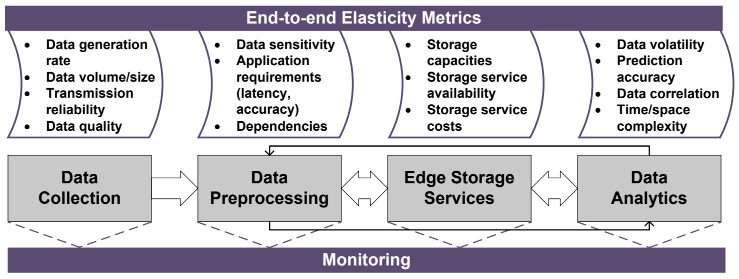

Fig. 2. End-to-end monitoring metrics of elastic edge services through four data stages

cycle, namely data collection, data preprocessing, storage service analysis and data analytics. However, the storage system must also allow definition of new metrics at runtime, depending on application-specific requirements.
Tooling: There are many tools for monitoring cloud systems, e.g., Prometheus[3], and Fluentd[4], but few able to monitor edge data metrics. These tools should be equipped with additional features including pluggable components for edge systems, such as fluentbit[5], providing AI support and tracing instrumentation, as a promising solution for providing end-to-end metrics for elastic storage services.
P2: Support Application-Specific Requirements. Based on sensor-specific metrics and relevancy, we can combine different solutions to deliver appropriate data to local analytics, while meeting application conditions, e.g., clean, complete or normalize sensor data before storage and analysis. Further, customization for secure and verifiable storage is required for applications with sensitive data.
How: Shown in Figure 3, depending on application information, different sensor data have corresponding data flow routed through the edge architecture to appropriate edge analytics, namely, descriptive, predictive or prescriptive. Interconnected storage nodes, with features including *data recovery* and edge *storage management* mechanisms, ensure access to the relevant data at the right time for different purposes. An algorithm repository contains a set of predefined processing utilities, which usage and order are application-specific and dynamically set at runtime in the elasticity management component. In addition, blockchain integrator component can capture certain types of application-specific data and pass them to the edge blockchain network for verification and auditing.
Tooling: A repository of available and pluggable microservices can speed up the DevOps of storage services by supplying needed utilities. Different microservices can be used to enable elastic activities, such as data cleaning, normalization and data integration [2]. To keep relevant and complete data in space-limited storage, nodes might incorporate an adaptive algorithm for efficient edge storage management and an automatic mechanism for recovery of incomplete datasets.
P3: Enable Adaptive Data Handling. From a software management standpoint, it is necessary to cope with heterogeneous data workloads including dynamic data streams, batch transfers, QoS critical requirements. Storage service

[3] https://prometheus.io/ [4] https://www.fluentd.org/ [5] https://fluentbit.io/

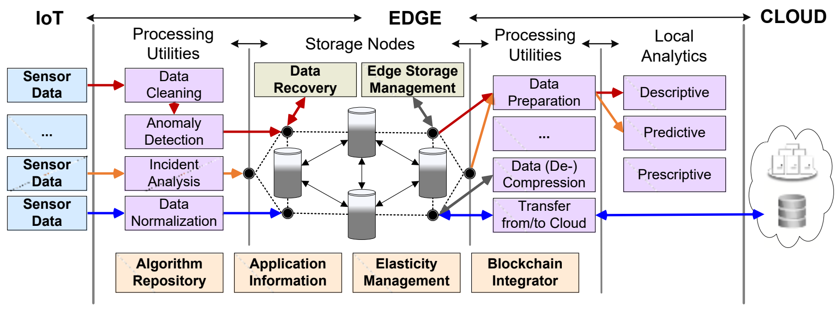

Fig. 3. Application-specific data flows through a holistic edge architecture

should ensure that stored data are always available, relevant and complete, i.e., keeping data integrity by utilizing different system and data operations.
How: In this context, critical software technology running on the edge can play an important role in storage resources abstraction, supporting communications, configuring suitable data handling features and on-demand data transfers. Techniques for auto-switch data handling algorithms/components should be explored.
Tooling: Fogger[6] could be used to support dynamic allocation and contextual location awareness of storage resources in distributed environment, and featuring blockchain technology. Microservices-based design concepts, such as Edgex[7] open source platform, might enable decentralized and independent data handling as well as reliable data integration supported by on-demand data services.
P4: Highly Customized System Bundling. Edge storage features should be highly customized and application-aware. Considering data workloads and deployment conditions, traditional inflexibility in software modules bundling can produce over- or under-bundled features for supporting edge application analytics. Thus, flexible storage configurations need to meet deployment situations.
How: Based on application-specific information and internal constraints (capacities, resources), the build and deployment process should bundle only components to match these constraints for the right infrastructures. This forces us to develop an optimizer for bundling and deploying different software modules. As shown in Figure 3, different utilities should be available for customized bundling.
Tooling: Existing deploying tools like Docker Compose[8], Ansible[9], and Terraform[10], allow us to bundle and deploy stack of services but they do not enable needed optimization. This requires us to leverage existing work and develop novel algorithms based on edge node characteristics. Developed algorithms should select application-specific and customized services to build dependent components.
P5: Runtime Software-Defined Customization. Different inputs, such as application information and data workload characteristics, have to be combined to support runtime customization of elastic operations and data processing utilities. A way of combining these inputs must enable dynamic, software-defined

[6] https://fogger.io/ [7] https://www.mainflux.com/ [8] https://docs.docker.com
[9] https://www.ansible.com/ [10] https://www.terraform.io/

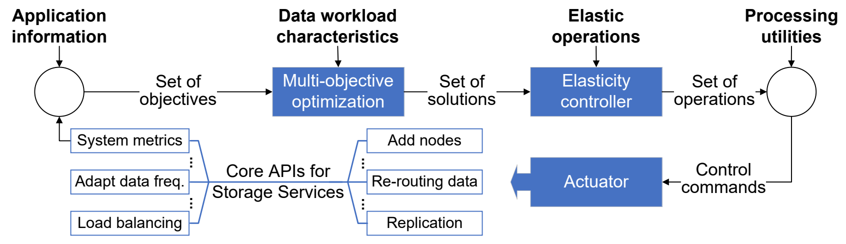

Fig. 4. Elasticity management for customized data flows and edge storage services

components for the overall system management. A multi-objective optimization mechanism should enable dynamic prioritization of IoT data and condition evaluation from SLOs at runtime, and thus would impact provided storage service.
How: Figure 4 illustrates potential control flow for elastic storage services. It incorporates a loop for managing internal storage system initially taking valid application information and current storage system metrics. To evaluate a set of defined objectives, dynamic workload characteristics are combined with static knowledge (elastic operations and processing utilities). To decide situational trade-offs for data quality and storage capacities, and utilizing edging system operations, we need to derive an optimization strategy for customized storage with core software-defined APIs for data management and service operations.
Tooling: We need to provide approaches of dynamic configuration, runtime code change (like model@runtime [4]) and services mesh, to combine different inputs from distributed storage nodes. The Kinetic Edge[11] could enable efficient load balancing between distributed storage locations. Multi-objective optimization of customized objectives, e.g., data quality and storage capacities, can be well addressed by using optimized data placement strategies in multi-cloud storage [14].
P6: Support IoT-Edge Continuum. This principle looks at impacting constant data flows between IoT systems and edge storage services, while supporting underlying protocols. According to edge storage performances, it requires triggering different actions with changing data generation frequency on-demand.
How: Both IoT and edge nodes require developing an edge-IoT connector to control data flows that can often be unpredictable. This connector should be able to (1) discard incoming poor quality data; (2) apply various sampling commands for collecting only relevant data; (3) trigger actions for turning off/on sensors in producing data; highly impacting overall performance of edge storage services.
Tooling: Novel mechanisms from data viewpoint can be considered allowing IoT sensors to securely receive and perform actuation requests from edge nodes and programmability viewpoint supporting actuation capabilities for remote IoT device programmability. New design patterns for data pipelines should be implemented to control unpredictable data flows and prevent low quality data.
P7: Support Edge-Cloud Continuum. This principle looks at inter-operation and data transmission between edge and cloud storage systems (Figure 3). De-

[11] https://www.vapor.io/kinetic-edge/

spite the advantages of edge nodes, it is obvious that for many applications, cloud repositories still have to keep large datasets for complex data mining and big data analytics. Thus, we need to support efficient and secure data transfer of large datasets. With an increasing number of data-intensive applications and bandwidth constraints, it will be crucial to reduce data traffic between the edge and the cloud. Further, once large datasets are available in the cloud, analytics models can be trained and then deployed at the edge for better decision making.
How: For efficient edge-cloud cooperation we must build an edge connector to the cloud, supporting: (1) operation viewpoint featuring timely techniques for data approximation, (de)compression and encryption/decryption; (2) network viewpoint featuring mechanism to avoid excessive data traffic through limited network infrastructure; (3) analytics viewpoint featuring coordination mechanism for consistent analytics models employing elasticity and deployment strategies.
Tooling: The approaches to push and pull data on-demand can be investigated for edge-cloud data transfer. Impact of symbolic data representation [10] can be considered as a good starting point to avoid excessive data traffic. There is need for a model to support secure data migration among multi-location data stores.

4 Related Work

System viewpoint. Various system operations have been used to build efficient edge storage, e.g., authors in [1] discussed a data life cycle while investigating the optimization of storage mechanisms and data management system design for the IoT. The concept of data-centric communication [12] proposed different management strategies to handle stored data from system viewpoint.
Application viewpoint. According to [3], it is possible to assign dynamic routes for IoT data based on application context information, considering four objectives, namely; lifetime, delay, reliability and data delivery, but only network viewpoint is examined. Authors in [7] proposed Storage as a Service model where unused storage space can be shared as a cloud-based service for different applications.
Design viewpoint. Some of the high-level requirements for dealing with a new design of the edge storage service in our paper is inline with IoT common design principles [15], but such IoT common principles do not dig into edge storage services and analytics scenarios. High-level self-adaptation for edge computing has been discussed in [5], but it does not focus on edge storage services for application contexts. In our approach, we bridge aforementioned gaps leading to customized software-defined elastic edge storage services.

5 Conclusions and Future Work

IoT data-intensive applications pose big challenges to satisfy their strict requirements for timely and accurate data-driven decision making, while relying on resource constrained edge systems. It is crucial to dynamically define a highly customized optimization strategy to handle incoming data from different perspectives as well as maintaining only the most relevant data for edge analytics.

To scale future edge analytics processes, we present engineering principles and demonstrate how they can potentially be implemented. In this context, proposed approaches can help researchers to improve revealed dependencies in edge data services. Although new insights are encouraging, many challenges are still open, considering other application contexts and the implementation of principles.

Acknowledgments: The work in this paper has been partially funded through Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015 and Ivan Lujic's netidee scholarship by the Internet Foundation Austria.

References

1. Ali, N.A., Abu-Elkheir, M.: Data management for the internet of things: Green directions. In: 2012 IEEE Globecom Workshops. pp. 386–390. IEEE (2012)
2. Ali, S., Jarwar, M.A., Chong, I.: Design methodology of microservices to support predictive analytics for iot applications. Sensors **18**(12), 4226 (2018)
3. Araújo, H.d.S., Rodrigues, J.J., Rabelo, R.d.A., Sousa, N.d.C., Sobral, J.V., et al.: A proposal for iot dynamic routes selection based on contextual information. Sensors **18**(2), 353 (2018)
4. Blair, G., Bencomo, N., France, R.B.: Models@ run.time. Computer **42**(10), 22–27 (Oct 2009)
5. D'Angelo, M.: Decentralized self-adaptive computing at the edge. In: International Conference on Software Engineering for Adaptive and Self-Managing Systems. pp. 144–148. ACM (2018)
6. Dimitrov, D.V.: Medical internet of things and big data in healthcare. Healthcare informatics research **22**(3), 156–163 (2016)
7. He, W., Yan, G., Da Xu, L.: Developing vehicular data cloud services in the iot environment. IEEE Transactions on Industrial Informatics **10**(2), 1587–1595 (2014)
8. Lai, L.L., Braun, H., Zhang, Q.P., Wu, Q., Ma, Y.N., Sun, W.C., Yang, L.: Intelligent weather forecast. In: International Conference on Machine Learning and Cybernetics. vol. 7, pp. 4216–4221 (2004)
9. Lederman, R., Wynter, L.: Real-time traffic estimation using data expansion. Transportation Research Part B: Methodological **45**(7), 1062–1079 (2011)
10. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and knowledge discovery **15**(2), 107–144 (2007)
11. ODonovan, P., Leahy, K., Bruton, K., OSullivan, D.T.: An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. Journal of Big Data **2**(1), 25 (2015)
12. Psaras, I., Ascigil, O., Rene, S., Pavlou, G., Afanasyev, A., Zhang, L.: Mobile data repositories at the edge. In: Workshop on Hot Topics in Edge Computing (2018)
13. Satyanarayanan, M., Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Hu, W., Amos, B.: Edge analytics in the internet of things. IEEE Pervasive Computing **14**(2), 24–31 (2015)
14. Su, M., Zhang, L., Wu, Y., Chen, K., Li, K.: Systematic data placement optimization in multi-cloud storage for complex requirements. IEEE Transactions on Computers **65**(6), 1964–1977 (2016)
15. Vogel, B., Gkouskos, D.: An open architecture approach: Towards common design principles for an iot architecture. In: Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings. pp. 85–88. ACM (2017)