Controlling Data Gravity and Data Friction: From Metrics to Multidimensional Elasticity Strategies

Boris Sedlak, Victor Casamayor Pujol, Praveen Kumar Donta, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology (*TU Wien*), Vienna 1040, Austria.
Email: {b.sedlak, v.casamayor, pdonta, dustdar}@dsg.tuwien.ac.at

***Abstract*—The growing amount of data generated at the edge of the network, e.g., by Internet of Things (IoT) devices, made it indispensable to relocate computational power close to the data source. Meanwhile, data tends to accumulate in chunks and is frequently subject to resource-intensive transformations, such as privacy enforcement. These phenomena, which are summed up as “data gravity” and “data friction”, have an impact on data processing and the overall system. However, whereas cloud centers are able to dynamically adapt services, e.g., by provisioning additional resources, edge devices provide fewer options to react to changing workloads. To retain the option to process data locally, we present the idea of controlling data gravity and friction with Service Level Objectives (SLOs). We introduce Markov SLO Configurations (MSCs) as a novel approach to organizing performance metrics and elasticity strategies. MSCs, in conjunction with our presented architecture, enable the evaluation of SLOs, the context-based selection of elasticity strategy (i.e., corrective measures), and the execution of strategies directly on edge devices. Thus, we lay the foundation for a new generation of SLOs that can operate across multiple elasticity dimensions, e.g., by scaling quality of service (QoS).**

***Index Terms*—Data Gravity, Data Friction, Service Level Objectives, Elasticity, Computing Continuum, Context Awareness**

I. Introduction

A Service Level Objective (SLO) is a commitment to maintaining a system in a desired state over a certain period of time [1]. It determines a system’s status by evaluating one or more Service Level Indicators (SLIs), usually performance metrics, and compares the result against a benchmark. If they diverge, the SLO is violated, which is corrected by a chain of countermeasures (i.e., elasticity strategies [2]). SLOs thus provide a system with “elasticity” – a degree of self-determination to adapt to changes in workload [3]. Elasticity strategies can span multiple dimensions, for example, by adjusting the amount of resources provisioned or by scaling the quality of service (QoS) [4]. However, to date, most SLOs are tied to a single elasticity strategy (e.g., scale resources in AWS EC2[1]) and are thus limited to one elasticity dimension [5]. This drastically limits the versatility of SLOs to react to more complex behavior within distributed systems, for example, compensating data gravity and data friction.

Data friction is a resistance that impedes data transfer between systems. It may be caused, for example, by incompatible data formats or privacy enforcement, resulting in processing delays, increased costs, and higher energy consumption [6]. Data gravity refers to the tendency of data to accumulate and attract further data and applications; thereby, it becomes increasingly difficult and costly to move the data. To cope with this, there is a tendency to relocate processing facilities to the edge of the network, i.e., where data is created [7], [8]. Processing data close to its source is motivated by numerous benefits, such as low latency and high bandwidth [9]. Friction-generating tasks (e.g., privacy enforcement) are equally assumed by edge or fog devices because, thus, unprotected data is less exposed to unauthorized access.

However, edge devices provide few options to scale provisioned resources [10]. To retain the option of processing data locally, edge devices must limit data gravity and data friction; otherwise, they find themselves unable to deal with growing data chunks and resource-intensive transformations. This inability to scale resources makes it attractive to explore other elasticity dimensions, e.g., by scaling the quality of generated data [11]. In this context, we present the idea of limiting data gravity and data friction by employing SLOs that extend into multiple elasticity dimensions. Our work conceptually builds upon the state of the art for constructing SLOs, in this case, the Polaris framework [5], and addresses challenges that arise when attempting to control data gravity and friction with SLOs. Within the presented paper we did not evaluate these concepts but laid the foundation for an upcoming implementation in future work. Our main contributions towards SLOs for data gravity and friction include:

1) A *mechanism to generate SLO configurations*, which uses Markov blankets (MB) for evaluating a system according to SLOs. This novel approach constructs around a central entity (e.g., data gravity) a graph of relevant metrics, elasticity strategies, and contextual information.
2) The *Markov SLO Configuration (MSC)* as a method to organize the SLO lifecycle from the collection of metrics to the enforcement of elasticity strategies. Information contained by the MSC (i.e., which metrics to collect and which elasticity strategies to apply) can be administered within a distributed system to create hierarchical SLOs that extend into the computing continuum [12].
3) The *context-based planning of elasticity strategies*, which regards the edge environment to select an elasticity strategy. Thus, it becomes possible to compare elasticity

[1]https://aws.amazon.com/ec2/

strategies that operate in different elasticity dimensions and pick one, e.g., depending on the corrective impact.

Although the goal of this work was to limit data gravity and friction, the results are transferable for SLOs that pose similar requirements, e.g., planning multidimensional elasticity strategies or orchestrating strategies directly on edge devices.

II. Background

We consider data gravity and data friction to be fairly new concepts for most readers; therefore, we use this section to provide background information about these phenomena and explain in more detail how they impact data processing. To illustrate the need for SLOs based on data gravity and friction, we further present (1) an exemplary use case that is referenced within the remainder of the paper and (2) an overview of motivating research challenges for creating SLOs that treat data gravity and friction.

A. Data Gravity and Friction

Data gravity describes the tendency of data to attract additional data. It is based on the idea that the more data and applications are stored at a particular location, the more attractive it becomes for other data to be stored there [13]. This phenomenon usually draws data toward the cloud but could occur anywhere along the computing continuum, e.g., the network edge. It promotes the creation of central data storage, which provides numerous benefits [14]: fewer inconsistencies, data unification, and improved security due to fewer attack vectors. Services and applications are equally drawn to larger amounts of data, which is motivated by two essential benefits: low latency and high bandwidth [13]. For measuring data gravity, the authors in [15] provide a formula based upon four network metrics: (1) data mass, i.e., the size of data; (2) data activity, i.e., the number of movements and interactions with the data; (3) bandwidth; and (4) latency. The authors raised these metrics for thousands of enterprises to compare data gravity between different countries and regions.

Data friction is a resistance that impedes data transfer, for example, when exchanging data between institutions [6]. It is frequently generated by preprocessing tasks, such as data enrichment [16] and privacy enforcement [11]. Data friction can be introduced by either socioeconomic or regulatory factors. Socioeconomic factors can be different understandings of data or metadata in scientific environments [17], or, more culturally, the simple desire to keep personal information confidential. Regulatory factors, on the other hand, are introduced to provide legal guidance, for example, the EU's General Data Protection Regulation (GDPR) [18]. When transferring information, data friction demands additional resources, such as time, computational power, or personal effort. Consider, for example, privacy enforcement for a data stream: Transforming data according to privacy policies provides benefits to stakeholders [19], it is thus evident that the transformation cannot be omitted. Nevertheless, data friction can be optimized by employing more efficient techniques or dividing it between

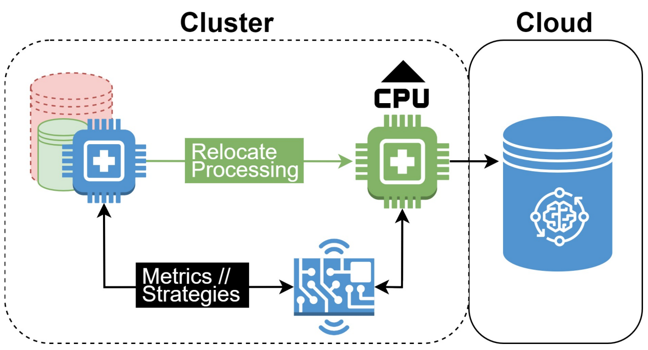


Fig. 1: Decrease data gravity and friction by applying SLOs

individual nodes. On the lines of data gravity, such a phenomenon must be measurable using a set of metrics, which will be explored further in Section IV.

B. Illustrative Scenario

To underline the benefits that emerge from SLOs on data gravity and friction, we embed the following research in a motivating example. Although the scenario is tailored to smart health, the concepts introduced can be applied to any field that uses distributed edge architectures, such as industrial automation or smart cities. For now, imagine a scientific institution conducting medical experiments; therefore, they require medical data from patients who are located in hospitals or home care. Depending on the experiment, different data is required, for instance, internal values (e.g., pulse, blood pressure, etc.) or skin mutations. The first is provided as a numeric stream, and the second as an image stream. Data is provided by IoT devices that are equipped with sensors, which stream it to the institution. Within the institution, data is accumulated until the experiment is evaluated and closed.

Patients agreed to participate if personal information was removed from the data; therefore, medical data must be transformed according to privacy policies before being stored centrally. To prevent unauthorized access to personal data, this transformation must occur directly on the IoT device. Technically, the smart health device could also provide additional data that is not part of the experiment, though this data is not authorized to leave the device. However, patients can decide to accumulate such data locally on the device, creating a personal data lake [14], and contribute this data to another experiment. This may be motivated by a monetary incentive.

However, IoT devices can be very restricted in terms of storage and computational power. Devices that fail to transform the data within a given time frame are facing "high data friction". The personal data lake on the IoT device further complicates the situation because increasing amounts of data become more difficult to manage. This can be summarized as "high data gravity." To deal with these issues, we introduce SLOs on data gravity and data friction. We organize IoT devices into clusters by grouping devices that perform similar tasks and have low latencies to each other. As far as possible, we add fog nodes (e.g., gateways and routers) to each cluster. Within each cluster, we elect the most powerful device as the cluster leader [20], which will be responsible for evaluating SLOs and orchestrating elasticity strategies.

The scenario is depicted in Fig. 1: Medical IoT devices are grouped into a cluster, which contains a fog device that assumes the role of the cluster leader. To evaluate the SLOs, IoT devices provide metrics to the cluster leader, which, once combined, reflect the high data friction and gravity. Due to this result, the cluster leader suggests the following elasticity strategies: (1) Transforming data is now performed nearby on a more powerful device, reducing data friction perceived by low-resource IoT devices; (2) scaling down the QoS (i.e., the data quality) reduces the data size and thus also data gravity within the personal data lake. The SLOs thus provided measures to detect and control high gravity and friction through multidimensional elasticity strategies. This improves the self-healing abilities of IoT devices [21].

C. Research Challenges

Our approach aims at addressing main research challenges, which include:

RC-1: *Performance metrics for gravity & friction:* Since data gravity was presented in [13], it has been used to describe data-centric phenomena [7], [14], but only the author in [15] would propose a formula for measuring gravity. Data friction, presented in [6], would be used for a wider range of socioeconomic phenomena [22]–[24], but never be measured within a system. We envision a set of metrics that reflect their behavior within a cluster of edge devices. Thus, exploring the factors that influence data gravity and friction, and creating the possibility to monitor a novel set of network properties.

RC-2: *SLO composition for data gravity & friction:* Next-level SLOs [5] comprise multiple low-level metrics to provide advanced guidelines on how a system should behave. As such, next-level SLOs could also be used to capture data gravity and friction, which are considered to be complex network behavior [7], [18]. However, composing SLOs on friction and gravity requires mechanisms to collect and accumulate metrics from edge devices. Based on this data, the SLO could be composed, evaluated, and paired with elasticity strategies.

RC-3: *Multidimensional elasticity strategies on the edge:* Cloud computing solutions (e.g., AWS EC2) provide customers with an environment that can scale according to changing demands, mitigating the risk of under- or overprovisioning resources. Although this concept is well established in the cloud [10], [25]–[27], there are few options to orchestrate elasticity strategies at the edge [11]. However, by evaluating SLOs on a device or cluster level, we would create the possibility of reacting to more fine-grained changes in the network behavior. Elasticity strategies could scale the system in various dimensions, for example, by adjusting the quality of a sensor [11] or by offloading processing [28].

III. Related Work

Although there exists work on data gravity and data friction that discusses their institutional and technical impact (see Background II-A), as of our knowledge, none of them is related to their implementation as SLOs. Identifying metrics that reflect these forces can be compared to the work in [25], [29], which discusses performance metrics for cloud computing environments. However, they focused solely on the computational load of the system to scale resources, thus, only operating in one elasticity dimension. This is similar to [30], where the authors discuss metrics for the elasticity of cloud databases. Complementarily, Fürst et al. [11] introduced a programming model that supports dynamic adaptations (comparable to elasticity strategies) within edge environments. Depending on the resource consumption of an IoT device, it was possible to dynamically adjust the QoS; however, it lacked the options to consider other elasticity dimensions.

The Polaris SLO Cloud project[2] provides a runtime environment that enables the combination of custom SLOs and elasticity strategies. Their work in [2], [5], [10], [31] provides fundamental concepts that are reused and extended over the course of this paper: Within [5] they introduced next-level SLOs, that is, SLOs that compose multiple low-level metrics. Data gravity and data friction demand such measures because, as explained in Section II-A, it requires a combination of metrics to capture such complex network behavior. The authors in [26] also support this thought; they state that “elastic behavior should be determined by a combination of factors”, which is similar to next-level SLOs.

A central component for evaluating next-level SLOs is the *SLO Controller* presented in [2] and [31], which provides the following features: i) Create and update mechanisms for SLOs, configurations can thus change over time; ii) SLOs and elasticity strategies are loosely coupled, that is, SLOs and elasticity strategies can be replaced and reused in multiple SLO mappings; iii) SLOs are evaluated periodically according to a configurable interval; iv) metrics required for the SLO evaluation are queried through a service that relies on native DB controllers; and v) elasticity strategies are translated to orchestrator-native representations, which are submitted to the orchestrator through an integrated controller. For the specification and configuration of SLOs, the authors have provided a language called *SLO Script* [2]. It is an extensible framework built on TypeScript, which provides type safety, that is, ensuring compatibility between SLOs and elasticity strategies at the time of configuration.

Existing work provided a framework for creating complex SLOs [5], measures for dynamically adapting services based on the system state [11], and identified the rising importance of treating data gravity and data friction [7], [17] at its source, i.e., the network edge. However, we can conclude from the presented related work that there exist no solutions that evaluate and resolve data friction or gravity within a distributed system, which we aim to address with the proposed SLOs.

[2]https://polaris-slo-cloud.github.io

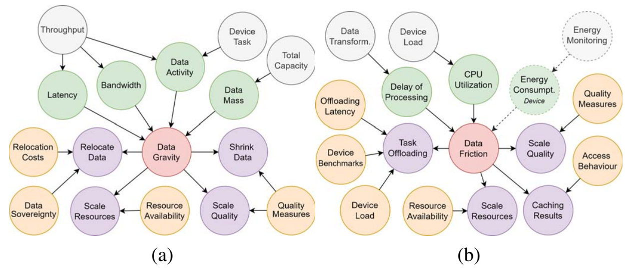


Fig. 2: Constructing a Markov Blanket for (a) Data Gravity (b) Data Friction

IV. Modelling Next-level SLOs and their Elasticity Strategies

Although mostly used in statistics and machine learning so far, we introduce Markov Blankets (MBs) [32] as a structured approach for exploring composed network metrics, such as data gravity and friction. Within an MB, we include the metrics required to determine the state of an SLO, elasticity strategies to correct a faulty system state, and contextual information for selecting between these strategies. For constructing an MB we assume a Directed Acyclic Graph (DAG), in which a central node x is connected to a set of incoming and outgoing nodes, i.e., its parents and children. Each parent node $p_1 ... p_n$ represents a random variable that influences the state of x at a certain time n; if for an arbitrary node y there exists no edge $y \rightarrow x$ in the graph, it means that y does not have an influence on the state of x at the time n.

To form an MB of data gravity, we (1) establish data gravity as central node x, and (2) arrange the metrics of Section II-A as parent nodes $p1 ... p_4$. Each of these parent nodes has an edge $p \rightarrow x$ that expresses its direct influence on the data gravity. Child nodes of x are random variables that are influenced by the state of x; thus, we (3) introduce elasticity strategies to treat data gravity. To construct a Markov blanket for a node x, we must further include parents of its children; we thus (4) identify additional factors that influence the probability of performing elasticity strategies. The resulting DAG is shown in Fig. 2a, which is created by performing steps (1-4): Data gravity as our composed metric (red); its parent variables that directly influence the gravity (green); further factors related to random variables (gray); however, these are not necessarily part of the MB since their influence can be derived indirectly through the colored nodes; child variables that contain elasticity strategies (purple); factors that provide contextual information for these strategies (yellow).

By definition, the MB must contain all colored nodes from Fig. 2a; thus including network metrics, elasticity strategies, and factors that influence these strategies. We define such a selection as Markov SLO Configuration (MSC), a set of variables that provides sufficient information for evaluating an SLO and planning which elasticity strategies to apply based on the context. Equally, Fig. 2b contains the DAG for constructing an SLO for data friction, i.e., all information required to build an MSC. The graph follows the same color code as Fig. 2a; any set that includes at least all colored nodes meets the requirements of an MB.

While an MSC determines how an SLO is evaluated and executed at a time n, this configuration can change over time, including: (1) adding or removing metrics; (2) adding or removing factors influencing the elasticity strategies; or (3) adding or removing entire strategies. Every change is tracked and indexed with the time n_x of the change, which determines how the system behaves until a new change is reported. The MSC can thus adapt over time.

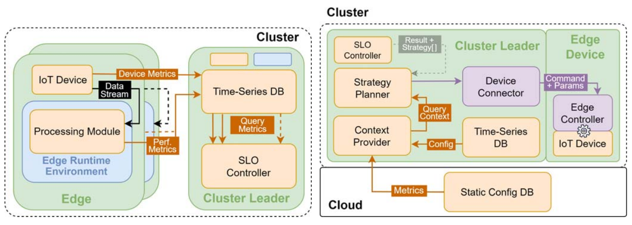
V. From Metrics to Elasticity Strategies

Based on the information contained in an MSC, it becomes possible to evaluate an SLO and plan which elasticity strategies to apply. However, from a technical perspective, there remain a variety of challenges to implementing these SLOs, including the following:

1) SLOs were traditionally based on metrics that are generated in the cloud, for example, resource consumption [25]. SLOs on data gravity or friction are based on metrics generated on the edge. To that extent, it requires the possibility of collecting metrics from edge devices, accumulating them close to the data source, and accessing them where the SLO is evaluated.
2) Continuously collecting metrics produces a considerable amount of data. Transferring this information to the cloud for evaluation increases the overall network traffic. To that extent, it lacks an architecture that collects metrics and evaluates SLOs directly on the network edge. Evaluating SLOs close to where metrics are created would also foster timely reactions to SLO violations.
3) Traditionally, only a single predefined elasticity strategy is applied to return the system to its desired state [5]. Contrarily, with the multitude of elasticity strategies contained in the MSC, it becomes possible to select the most suitable one. However, there exists no mechanism to compare elasticity strategies and select one of them.
4) Elasticity strategies proved useful for cloud-based processing (e.g. provisioning virtual resources); however, elasticity strategies that scale the QoS must be enforced directly at the data source, i.e., at the network edge. To that extent, it requires an architecture that evaluates SLOs along the computing continuum, from where elasticity strategies can be orchestrated to edge devices.

The given challenges can be summarized by the MAPE+K (Monitor, Analyze, Plan, Execute, Knowledge) cycle [21], which the authors in [31] used to capture all phases of an elastic cloud application. The cycle consists of (1) *monitoring* the system and collecting metrics, (2) *analyzing* whether the SLO is fulfilled based on this information, (3) in case the SLO was violated, *planning* which elasticity strategy to apply, and (4) *executing* the strategy to restore the system state. To pass information between the stages, but also to persist *knowledge*-based information, a state is shared between the stages.

In the remainder of the chapter, we move through the MAPE+K cycle and address the given challenges in their respective phases. We assume that the network is structured



(a) (b)

Fig. 3: (a) Collecting and evaluating metrics from edge devices (b) Context-based planning of elasticity strategies

as presented in Section II-B: edge devices and fog nodes are combined into clusters, each containing a powerful cluster leader. We use the Polaris runtime from Section III as a technical reference for specifying SLOs with *SLO Script* and evaluating and enforcing SLOs with their *SLO Controller*.

A. Metrics in the Computing Continuum

Collecting metrics requires a data store deployed close to the data source. To that extent, each cluster leader hosts (1) Prometheus[3], a time-series database (DB) used to store metrics, and (2) an instance of the *SLO Controller*, which will be required to evaluate the SLO. Unlike some edge devices, the cluster leader provides sufficient resources to run the *SLO Controller* and Prometheus, thus covering heterogeneity within the edge environment. The structure of the cluster is shown in Fig. 3a: Device metrics and operational metrics are generated on the edge devices and continuously ingested into the time-series DB on the cluster leader.

Following our approach, we collect metrics close to where they are created and provide them to the *SLO Controller* whenever the SLO is evaluated. We would thus solve challenge (1), by capturing metrics that represent data gravity and friction and accumulating them close to the edge device.

B. SLO Specification and Analysis

The second step in the MAPE+K control loop consists of analyzing the system state, i.e., evaluating the SLOs based on the provided metrics. As depicted in Fig. 3a, this is the responsibility of the *SLO Controller*: It first queries low-level metrics from a configurable data source, composes the SLOs of data friction and gravity, and then compares the result against a benchmark. The central entity in the *SLO Controller* would be the `DataGravitySLO` or `DataFrictionSLO` class, a direct representation of the SLO logic. It includes for each low-level metric a reference to a data source (i.e., Prometheus) and how it can be extracted.

Within the `DataFrictionSLO`, metrics are combined as shown in (1) and (2): $processingDelay$ and $cpuLoad$ are multiplied to accumulate their values. The $cpuLoad$ maintains a neutral factor until it rises above a certain degree (t_x); only then does it have a decisive impact on the composed metric. While $processingDelay$ and $cpuLoad$ are a representation of the metric parameters, the target CPU load (t_x) can be configured freely.

$$processingDelay_{ms} \times f\left(cpuLoad_{\%}\right) \tag{1}$$

$$f(x) = \begin{cases} \left(x/t_x\right)^2, & \text{if } x \geq t_x \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

We use *SLO Script* to configure the SLO and link it to an elasticity strategy. Listing 1 shows how `DataFrictionSLO` is mapped to `SensorQualityScale`, the corresponding elasticity strategy. Whether the SLO is met or not, is specified through `frictionThreshold`; the desired CPU utilization over `targetDeviceLoad`. Although the configuration in Listing 1 statically maps a single elasticity strategy to the outcome of the SLO evaluation, we will break up this connection in the planning phase.

```
export default new DataFrictionSloMapping({
  metadata: ...
  spec: new DataFrictionSloMappingSpec({
    targetRef: ...
    elasticityStrategy:
      new SensorQualityScale(),
    sloConfig: {
      frictionThreshold: 50,
      targetCPULoad: 70} }) });
```

Listing 1: Configuring an SLO on data friction with *SLO Script*

Following the architectural consideration presented, we evaluate the SLO directly on the edge, close to where the metrics were created and stored; thus, solving challenge (2).

C. Context-aware Planning of Elasticity Strategies

Suppose that an SLO on data gravity or friction was violated, the third stage of the MAPE+K cycle consists of planning corrective measures. To select between multiple elasticity strategies, we introduce a component that receives the result of the SLO evaluation, queries contextual information, and identifies the most beneficial elasticity strategy. Depending on the scenario, “beneficial” could mean e.g., lowest energy consumption or highest corrective impact on the system.

The architectural extension is illustrated in Fig. 3b: Instead of mapping only one strategy to an SLO, an array of strategies can be supplied. The *Strategy Planner* resolves contextual information (e.g., costs for relocating data) through the *Context Provider*. Static configurations (e.g., quality measures) are queried from a separate DB, which can be hosted in the cloud. Since this information rarely changes, it can be cached in the *Context Provider* and updated by a trigger function.

Based on contextual information, the *Strategy Planner* compares the impact of elasticity strategies and selects one. Thus, we answer challenge (3). As a side note, the *Strategy Planner* could even indicate the top n strategies to accumulate their effects on the system. However, for now, we assume that either `SensorQualityScale` or `TaskOffloadKind` was planned to decrease data gravity or friction.

[3]https://prometheus.io/

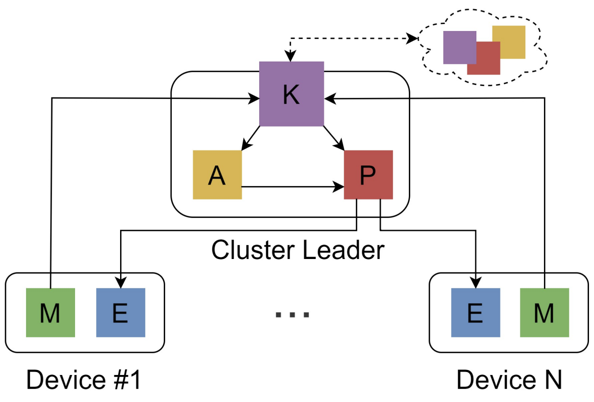


Fig. 4: Distribute MAPE+K steps over computing continuum[4]

D. Distributed Execution of Elasticity Strategies

After planning an elasticity strategy, it must be orchestrated for edge devices; therefore, two components were added to the architecture depicted in Fig. 3b: the *Device Connector* and the *Edge Controller*. The *Device Connector* is hosted on the cluster leader and is responsible for communicating elasticity strategies to edge devices, where they are received by the *Edge Controller*. In the planning phase, the *Strategy Planner* picked as elasticity strategy either `SensorQualityScale` or `TaskOffloadKind`, which are described below:

Quality Scaling adjusts the quality of the data that is produced on an edge device. Processing a stream of lower quality has been shown to decrease the computational load of devices [11], [33]; thus, reducing the data friction perceived by these devices. The managed IoT device must therefore support dynamic changes in the generated stream. Decreasing the quality of data reduces its size (i.e., mass); in cases where data is accumulated locally on the edge device, this decreases data gravity at the same time.

Task Offloading moves processing to a different device in the cluster, such as powerful fog nodes [34]. Offloading processing is motivated primarily by minimizing the streaming delay, but it can consider other factors, such as lower energy consumption [28], [35]. Offloading load to more powerful or less used devices relieves individual devices of excess load, while at the same time decreasing latency [36]. This reduces overall data friction within the cluster because devices are less likely to be pushed beyond their operational limits.

The two strategies presented can correct high gravity or high friction within a distributed system. For orchestration, they rely on the *SLO Controller*, which communicates the instructions to edge devices. Therefore, we declare challenge (4) as solved.

E. Knowledge Transfer and Markov SLO Configurations

Components involved in SLO enforcement frequently need to share information with each other; for example, the result of the SLO evaluation must be communicated from the *SLO Controller* to the *Strategy Planner*. This type of information is transient and can be passed between components without persisting. However, if we consider the content of an MSC (i.e., metric composition and elasticity strategies), this information must be federated between cluster leaders and manageable somewhere along the computing continuum [38]. Whenever the MSC changes, e.g., by adding new types of metrics or elasticity strategies, cluster leaders must be able to retrieve the latest version of the SLO configuration.

Fig. 4 visualizes how edge devices and the cluster leader exchange information depending on the MAPE+K stages: Edge devices monitor the status of the system (M) by ingesting metrics to intermediary storage. These metrics, along with other knowledge-based information (K), are accessible through an interface in the cluster leader. Whenever the system status is analyzed (A), metrics are queried to evaluate the SLO. Supposed the SLO was violated, contextual information is regarded to select an elasticity strategy (P). Eventually, elasticity strategies are orchestrated to edge devices for execution (E) to move the system back into its desired state.

The MSC can be administered by another entity, which hierarchically stands above the cluster leader, e.g., in the cloud. Thus, it becomes possible to erect multiple layers of SLOs that each react to changes in their respective environments.

VI. Conclusion & Future Work

In this paper, we presented the autonomous control of data gravity and data friction through SLOs. This was motivated by increasing data gravity, which requires relocating computational power to the network edge, and ubiquitous data friction, which demands additional resources when transferring data.

To construct SLOs based on data gravity and friction, we introduced Markov blankets as a novel approach to identify metrics, elasticity strategies, and contextual factors. Thus, our approach provides all information needed to evaluate the SLOs. Selecting a preferred elasticity strategy (i.e., one that operates in a certain elasticity dimension), considers the context of the edge environment. Evaluation of SLOs, planning of an elasticity strategy, and orchestration of a strategy are assumed by a single powerful node. This responsibility can rotate within the distributed system, cluster leaders can maintain a state and recover SLO configurations in case of failure. Thus, we foster the creation of hierarchically organized SLOs that each react to changes in their respective environments.

For future work, we plan to provide a prototype that combines the presented components and features the entire MAPE+K lifecycle of an SLO. Certain aspects, such as the accumulation of multiple elasticity strategies, will further require a sophisticated orchestration model.

Acknowledgment

Research received funding from the EU's Horizon Europe Research and Innovation Program under Grant Agreement No. 101070186. EU website for Teadal: https://www.teadal.eu/

[4]The graphic is an extension of the master-worker pattern presented in [37], our main modification is the concept of maintaining knowledge-based information (K) separately and providing it to the remaining steps.

References

[1] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57–81, Mar. 2003.

[2] T. Pusztai, A. Morichetta, V. C. Pujol, S. Dustdar, S. Nastic, X. Ding, D. Vij, and Y. Xiong, "SLO Script: A Novel Language for Implementing Complex Cloud-Native Elasticity-Driven SLOs," in *2021 IEEE International Conference on Web Services (ICWS)*. Chicago, IL, USA: IEEE, Sep. 2021, pp. 21–31.

[3] N. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," *International Conference on Autonomic Computing*, pp. 23–27, Jan. 2013.

[4] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of Elastic Processes," *Internet Computing, IEEE*, vol. 15, pp. 66–71, Nov. 2011.

[5] S. Nastic, A. Morichetta, T. Pusztai, S. Dustdar, X. Ding, D. Vij, and Y. Xiong, "SLOC: Service Level Objectives for Next Generation Cloud Computing," *IEEE Internet Computing*, vol. 24, no. 3, pp. 39–50, May 2020.

[6] P. N. Edwards, *A vast machine: computer models, climate data, and the politics of global warming*. Cambridge, Mass: MIT Press, 2010, oCLC: ocn430736496.

[7] M. Campbell, "Smart Edge: The Effects of Shifting the Center of Data Gravity Out of the Cloud," *Computer*, vol. 52, no. 12, pp. 99–102, Dec. 2019.

[8] S. Dustdar, V. C. Pujol, and P. K. Donta, "On Distributed Computing Continuum Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4092–4105, Apr. 2023.

[9] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[10] S. Nastic, T. Pusztai, A. Morichetta, V. C. Pujol, S. Dustdar, D. Vii, and Y. Xiong, "Polaris Scheduler: Edge Sensitive and SLO Aware Workload Scheduling in Cloud-Edge-IoT Clusters," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. Chicago, IL, USA: IEEE, Sep. 2021, pp. 206–216.

[11] J. Fürst, M. Fadel Argerich, B. Cheng, and A. Papageorgiou, "Elastic Services for Edge Computing," in *2018 14th International Conference on Network and Service Management (CNSM)*, Nov. 2018, pp. 358–362, iSSN: 2165-963X.

[12] V. Casamayor-Pujol, P. K. Donta, A. Morichetta, I. Murturi, and S. Dustdar, *Distributed Computing Continuum Systems – Opportunities and Research Challenges*, Mar. 2023.

[13] D. MacCrory, "Data Gravity – in the Clouds – Data Gravitas," Dec. 2010. [Online]. Available: https://datagravitas.com/2010/12/07/data-gravity-in-the-clouds/

[14] H. Alrehamy and C. Walker, "Personal Data Lake With Data Gravity Pull," Aug. 2015.

[15] Digital Realty, "Data Gravity Index DGx," Tech. Rep. V1.5, 2022. [Online]. Available: https://www.digitalrealty.com/platform-digital/data-gravity-index

[16] F. Xhafa, B. Kilic, and P. Krause, "Evaluation of IoT stream processing at edge computing layer for semantic data enrichment," *Future Generation Computer Systems*, vol. 105, pp. 730–736, Apr. 2020.

[17] P. Edwards, M. Mayernik, A. Batcheller, G. Bowker, and C. Borgman, "Science Friction: Data, Metadata, and Collaboration," *Social studies of science*, vol. 41, pp. 667–90, Oct. 2011.

[18] J. Bates, "The politics of data friction," *Journal of Documentation*, vol. 74, Aug. 2017.

[19] S. Gritzalis, E. R. Weippl, S. K. Katsikas, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, Eds., *Trust, Privacy and Security in Digital Business: 16th International Conference, TrustBus 2019, Linz, Austria, August 26–29, 2019, Proceedings*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, vol. 11711.

[20] I. Murturi, "Resource Management and Elasticity Control in Edge Networks," Ph.D. dissertation, 2022, p. 38-44.

[21] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[22] V. Aula, "Institutions, infrastructures, and data friction – Reforming secondary use of health data in Finland," *Big Data & Society*, vol. 6, p. 205395171987598, Jul. 2019.

[23] K. Eschenfelder and K. Shankar, "Of Seamlessness and Frictions: Transborder Data Flows of European and US Social Science Data," Mar. 2020, pp. 695–702.

[24] G. Panagiotidou, J. Poblome, J. Aerts, and A. Vande Moere, "Designing a Data Visualisation for Interdisciplinary Scientists. How to Transparently Convey Data Frictions?" *Computer Supported Cooperative Work (CSCW)*, pp. 1–35, Jul. 2022.

[25] M. Beltrán, *Defining an Elasticity Metric for Cloud Computing Environments*, Nov. 2016, vol. 2, journal Abbreviation: EAI Endorsed Transactions on Cloud Systems Publication Title: EAI Endorsed Transactions on Cloud Systems.

[26] M. M. Bersani, D. Bianculli, S. Dustdar, A. Gambi, C. Ghezzi, and S. Krstić, "Towards the formalization of properties of cloud-based elastic systems," in *Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*, ser. PESOS 2014. New York, NY, USA: Association for Computing Machinery, May 2014, pp. 38–47.

[27] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications," in *Cluster Computing and the Grid*, May 2013.

[28] H. Guo, J. Liu, and J. Lv, "Toward Intelligent Task Offloading at the Edge," *IEEE Network*, vol. 34, no. 2, pp. 128–134, Mar. 2020.

[29] S. Lehrig, H. Eikerling, and S. Becker, "Scalability, Elasticity, and Efficiency in Cloud Computing: a Systematic Literature Review of Definitions and Metrics," in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, ser. QoSA '15. New York, NY, USA: Association for Computing Machinery, May 2015, pp. 83–92.

[30] R. F. Almeida, F. R. C. Sousa, S. Lifschitz, and J. C. Machado, "On defining metrics for elasticity of cloud databases," 2013.

[31] T. Pusztai, A. Morichetta, V. C. Pujol, S. Dustdar, S. Nastic, X. Ding, D. Vij, and Y. Xiong, "A Novel Middleware for Efficiently Implementing Complex Cloud-Native SLOs," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, Sep. 2021, pp. 410–420, iSSN: 2159-6190.

[32] J. Pearl, *Probabilistic reasoning in intelligent systems : networks of plausible inference*. San Mateo, Calif. : Morgan Kaufmann Publishers, 1988.

[33] B. Sedlak, I. Murturi, and S. Dustdar, "Specification and Operation of Privacy Models for Data Streams on the Edge," in *IEEE 6th International Conference on Fog and Edge Computing (ICFEC)*, May 2022, pp. 78–82.

[34] P. Plebani, D. Garcia-Perez, M. Anderson, D. Bermbach, C. Cappiello, R. I. Kat, A. Marinakis, V. Moulos, F. Pallas, S. Tai, and M. Vitali, "Data and Computation Movement in Fog Environments: The DITAS Approach," in *Fog Computing: Concepts, Frameworks and Technologies*, Z. Mahmood, Ed. Cham: Springer International Publishing, 2018, pp. 249–266.

[35] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[36] S. Rabinia, H. Mehryar, M. Brocanelli, and D. Grosu, "Data Sharing-Aware Task Allocation in Edge Computing Systems," in *2021 IEEE International Conference on Edge Computing (EDGE)*, Sep. 2021, pp. 60–67, iSSN: 2767-9918.

[37] V. Cardellini, T. Galinac Grbac, M. Nardelli, N. Tanković, and H.-L. Truong, "QoS-Based Elasticity for Service Chains in Distributed Edge Cloud Environments," in *Autonomous Control for a Reliable Internet of Services: Methods, Models, Approaches, Techniques, Algorithms, and Tools*, ser. Lecture Notes in Computer Science, I. Ganchev, R. D. van der Mei, and H. van den Berg, Eds. Cham: Springer International Publishing, 2018, pp. 182–211.

[38] P. K. Donta, B. Sedlak, V. Casamayor Pujol, and S. Dustdar, "Governance and sustainability of distributed continuum systems: a big data approach," *Journal of Big Data*, vol. 10, no. 1, p. 53, Apr. 2023.