Performance-cost trade-offs in service orchestration for edge computing

Daniel Balouek

Performance-cost trade-offs in service orchestration for edge computing

Daniel Balouek
IMT Atlantique, Nantes Université, École Centrale Nantes
CNRS, Inria, LS2N, UMR 6004, F-44000 Nantes, France

ABSTRACT

Low latencies connections and decentralized servers are currently showcasing a new potential for distributed computing. By moving away from traditional centralized cloud models and toward edge computing, which allows for more autonomy and decision-making at the network's edge, almost any physical thing can be turned into an Internet of Things (IoT) device that can elaborate on data it senses from its environment. In this context, service management and adaptation routines in a highly dynamic and geographically distributed federation depends on a large number of factors ranging from performance to cost and the fluctuation of the data quality.

This paper presents mechanisms for monitoring resources at the edge in real-time, orchestrating service provisioning, performing data-driven decisions on behalf of applications, adapting service locations, and coordinating sensing tasks. The demonstration focuses on autoscaling of containers, service placement and distributed sensing, while considering utility metrics to help achieve a fluid workload in Kubernetes clusters.

ACM Reference Format:
Daniel Balouek, *IMT Atlantique, Nantes Université, École Centrale Nantes, CNRS, Inria, LS2N, UMR 6004, F-44000 Nantes, France* . 2024. Performance-cost trade-offs in service orchestration for edge computing. In *36th International Conference on Scientific and Statistical Database Management (SSDBM 2024), July 10–12, 2024, Rennes, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3676288.3676307

1 INTRODUCTION

Low latencies connections and decentralized servers are currently showcasing a new potential for distributed computing. This shift from traditional centralized Cloud models enables distributed analytics spanning resources at the edges, in the core and in-between, also referred to as the Computing Continuum [3, 4]. The Computing Continuum aggregates the architectural and algorithmic challenges of its individual layers while presenting new challenges related to their control and adaptation.

The Computing Continuum is an ideal platform for supporting urgent computing, i.e., computing under strict time and quality constraints to support decision making with the desired confidence within a defined time interval [5, 11]. However, it represents extreme heterogeneity in the capabilities and capacities of systems, furthered coupled with extreme uncertainty arising from the availability and quality of data, resources, and services [13]. Thus, the end-to-end performance of applications deployed across the continuum relies both on the understanding of application models and components, as well as the resources available during execution.

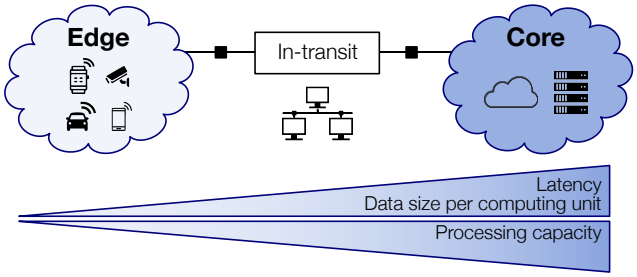

Figure 1: An abstract view of the Computing Continuum. Traversing the data path from the network edges, via in-transit resources, to the core, resources typically increase in scale and processing capacity, but also imply increasing latency and data movement costs.

Such applications benefit from a computing environment built on top of mixed, heterogeneous and even mobile resources, that promises to map user expectations and constraints in terms of response time, solution quality, data resolution, cost, energy, etc. Particularly, pervasive applications will need a flexible and dynamic provisioning of computing services , that is, a provisioning system capable of orchestrating (activating, deactivating, updating, integrating, etc.) processing, storage and networking resources.

This work focuses on adaptation routines for data-driven applications using Kubernetes, the *de facto* standard, for service orchestration. The demonstration consists in a (1) deployment of an edge-cloud cluster on Kubernetes, (2) a service placement defined by the location of data products and performance of data processing, and (3) an autoscaling mechanism for adapting the number of services instances with regards to over-provisionning.

2 MOTIVATION

2.1 Urgent Computing

Urgent science describes time-critical, data-driven scientific workflows that can leverage distributed data sources in a timely way to facilitate important decision making. Examples of urgent science span various domains ranging from applications that aim to improve quality of life, monitor civil infrastructures, respond to natural disasters and extreme events, and accelerate science. Urgent science workflows present requirements and constraints due

SSDBM 2024, July 10–12, 2024, Rennes, France

ACM ISBN 979-8-4007-1020-9/24/07
https://doi.org/10.1145/3676288.3676307

to the nature and distribution of the data, the complexity of the models involved, the stringent error thresholds, and the strict time constraints [2].

The Earthquake Early Warning (EEW) application [7] is representative of that class of problems, with two important data processing steps: a by-sensor time-series classification and a by-region combination of sensor predictions, as illustrated in Figure 2.

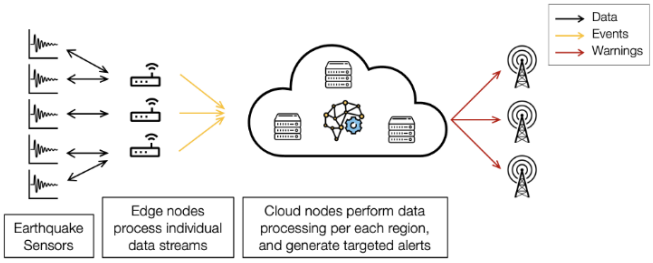

Figure 2: An illustration of the Distributed Multi-Sensor Earthquake Early Warning use-case [7]. Seismic sensors located in the Edge send measurements to gateways in the network which pre-process the data. Those preprocessed data are sent to cloud servers which complete the data processing and eventually broadcast earthquake alerts.

Such Cyber-Physical System (CPS) combines physical and computational components to monitor and control physical processes. CPSs integrates physical processes, control algorithms and communication networks to deliver alerts within an information-sharing network. In other words, our vision for Urgent Computing aims at taking advantage of a broad computing landscape that mixes Cloud Computing, Edge Computing and IoT resources.

It is essential in this setting to effectively integrate edge devices and cloud servers in order to quickly and accurately address any scenario and minimize the volume of data sent to the cloud. This necessitates the implementation of a reliable adaptation mechanism that consistently monitors, understands, and adjusts in response to the changing conditions of the computing environment.

3 METHODOLOGY

3.1 Data quality adaptation strategy

The system comprises a set of distributed and finite resources with varying computational and network capabilities. To analyze sensor data from numerous sources on these resources, one must select a system that balances latency and accuracy. This section describes an approach for adapting data quality in applications that rely on sensor data. This approach determines the allocation of data characteristics across the current data sources. It assesses whether the system is capable of handling the original data qualities of all data sources, or if a drop in quality is necessary. This approach ensures the quality of the produced data while ensuring that the system accuracy is above a predetermined threshold a_{min}. The service responsible for implementing this strategy is deployed on the Edge.

Let $B = \{b_1, ..., b_m\}$ be the set of possible quality distributions among data sources. Each distribution b_i is formulated as $\{\beta \cdot q_l \mid 1 \leq \beta \leq k \text{ and } q_l \in Q\}$, where β represents the number of data sources having the data quality q_l and k is the total number of data sources in the system. q_l might corresponds to their original or reduced data qualities. In the object detection use case $Q = \{512p, 416p\}$. For n=3, a possible data quality distribution can be $b_1 = \{(2) \cdot Q_{512p}, (1) \cdot Q_{416p}\}$.

The suggested technique chooses the distribution from the set B that has the shortest analysis latency and highest accuracy, while still meeting the minimum criterion a_{min}. However, in certain specific scenarios, the chosen distribution of data quality may not be the most efficient. The system achieves a balance between latency and accuracy, considering the scenario where there is an alternative distribution that offers a latency improvement of less than 10ms but a decrease in accuracy of 20% or more. The strategy is presented in Algorithm 1.

Algorithm 1: Select the quality distribution with a latency-accuracy trade-off.

```
   Result: quality distribution with the optimal trade-off.
1  begin
2      initialization;
3      for b in B do
4          L ← getEstimatedLatency(b)
5          A ← getEstimatedAccuracy(b)
6          if A ≥ a_min then
7              add({b, L, A}, list);
8      if isEmpty(list) == TRUE then
9          return ∅
10     best ← getMinLatency(list);
11     for x in list do
12         if △(x[L], best[L]) < 10 and
13          △(x[A], best[A]) ≥ 20% then
14             best ← x;
15     return(best);
```

In steps 2-6, it calculates the estimated latency and accuracy of each possible data quality distribution. Then, in step 5, it filters those with unacceptable accuracy. If no configuration provides acceptable accuracy, the data source can't join the system at that time period (steps 7 & 8). Otherwise, the preferred data distribution among the acceptable configurations is the one with the fastest analysis (step 9). In steps 10-12, it checks if there is another distribution that matches the use case presented above. If so, it will be selected as the preferred quality distribution in the system.

3.2 Resource adjustment strategy

Algorithm 2 is employed to minimize the resource demands of demanding jobs in situations where it is not feasible to provide the complete amount of required resources. It optimizes the utilization of system resources while guaranteeing a minimal threshold that is equivalent to 50% of their estimated requirements.

Algorithm 2 accepts as input the category of the work to be allocated and the list of resources allocated for high-demand tasks.

In steps 2 and 3, it gets the remaining free resources on the Cloudlet-Cloud tiers and selects the one with the maximum remaining capacity. If the remaining capacity selected is greater than or equal to the minimum threshold, the resource is reserved (steps 3-5). However, if not, the algorithm attempts to reach the minimum threshold by adjusting the computing capacity of the other reservations on the same selected resource (steps 6-13). In step 7, it gets the remaining capacity needed to reach the minimum threshold. The reservations on the selected resource that can handle a resource adjustment are those that remain above the minimum threshold even if their computing capacity is reduced (step 8). In steps 9 and 10, it reduces the remaining capacity needed evenly from the reservation list. After adjustment, the list of reserved resources is updated (steps 11-13).

Algorithm 2: Resource adjustment for tasks.

Data: listLI, listHI, category
Result: Adjust required resources of intensive tasks

```
Void Adjust_res(listLI, listHI, category)
2   begin
3     listFree ← getFreeResources();
4     res ← getMAX(listFree, category);
5     if res ≥ 50% × requiredResources then
6       entry ← reserve(res, category);
7       updateReservedList(category, entry);
8     else
9       remain ← getRemain(res);
10      listReservations←checkReservations(remain,
          listLI, listHI);
11      part ← (remain / size(listReservations));
12      newList ← Reduce(part, listReservations);
13      updateReservedList(newList);
14      entry ← reserve(res, remain);
15      updateReservedList(entry);
```

4 SERVICE ORCHESTRATION

This demonstration builds upon preliminary research on edge computing and urgent science to providing attendees with the required tooling for orchestrating services and adapting performance to given constraints [14]. From an application development perspective, lowering the entry barriers for scientists from related domains when mapping application expectations and constraints is essential. From a system perspective, improving resource orchestration (activating, deactivating, updating, integrating, etc.) at the edge of the network allows for better resource usage.

4.1 Demonstration Setup

We use Kubernetes for managing a containerized environment describing an Edge-Cloud cluster over the Grid5000 platform [1]. The Cloud is composed of a single core node with 8 cores, and the edge is composed of 1-core nodes, which the number can be adjusted during the experiments. EnosLib [6] enables the scripting of network emulation features that allows the specification of Edge-to-Cloud communication constraints (delay, loss, and bandwidth).

4.2 Scenario

The overall idea of resource orchestration at the edge is to leverage performance indicators such as performance or cost to manage the service deployment and placement. Additionally, the application data itself. through intelligent algorithm for training and inferencing, also influence the resource orchestration.

First, we will allow users to visualize the baseline performance of our Edge-Cloud cluster using a synthetic dataset. Second, we will showcase an adaptation routine for scaling the number of edge nodes according to a target and dynamic performance indicator. Third, we will leverage a simple machine learning process to identify out-of-scope values across edge-nodes.

5 RELATED WORK

Analyzing concurrently multiple data streams with limited resources forces resource-quality trade-offs [8, 16, 17]. As the incoming data are processed concurrently, resources available to each data stream are often unknown. Online/offline configurations adaptation is currently a promising solution to address the issue of limited resources [9, 10, 12, 15, 17]. In [15], Wang adopt an offline configuration adaptation and bandwidth allocation strategies to address the issue of limited resources between IoT devices and edge nodes. Similar to the approach presented in this work, the adaptation is triggered periodically. Systems in [9, 10] adopt an online configuration adaptation algorithms for video analytics in Edge computing. The configurations targeted are frame rate and resolution. However, these systems only focus on the performance of the analysis stage and not on a complete workflow. Additionally, they only target Edge-based video analytics applications. In [12], they present an Edge Network Orchestrator for Mobile Augmented Reality (MAR) systems. It boosts the performance of an Edge-based MAR system by optimizing the edge server assignment and video frame resolution selection for MAR users.

6 CONCLUSION

Computing is shifting from the traditionally centralized cloud to a distributed set of heterogenous resources located between the edge, the cloud and in-between. As computing has moved to this *Computing Continuum*, the tradeoffs between performance, availability and cost have become increasingly complicated. This paper presents mechanisms for adapting the orchestration of containers while considering utility metrics to help achieve a flexible resource management in Kubernetes clusters.

As future work, we aim at augmenting common operations of data-driven analytics (e.g. collection, filtering, processing, delivery) with similar adaptation and tunable parameters. This will provide software abstractions that will be able to better orchestrate operations by considering application context and real-time infrastructure metrics.

ACKNOWLEDGMENTS

This work is partially funded through the French project OTPAAS (BPiFrance).

REFERENCES

[1] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. 2013. Adding Virtualization Capabilities to the Grid'5000 Testbed. In *Cloud Computing and Services Science*, Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan (Eds.). Communications in Computer and Information Science, Vol. 367. Springer International Publishing, 3–20. https://doi.org/10.1007/978-3-319-04519-1_1

[2] Daniel Balouek-Thomert and al. 2020. Harnessing the Computing Continuum for Urgent Science. *SIGMETRICS Perform. Eval. Rev.* 48, 2 (Nov. 2020), 41–46. https://doi.org/10.1145/3439602.3439618

[3] Daniel Balouek-Thomert, Eduard Gibert Renart, Ali Reza Zamani, Anthony Simonet, and Manish Parashar. 2019. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. *International Journal of High Performance Computing Applications* 33 (11 2019), 1159–1174. Issue 6. https://doi.org/10.1177/1094342019877383/FORMAT/EPUB

[4] P Beckman and al. 2020. Harnessing the computing continuum for programming our world. *Fog Computing: Theory and Practice* (2020), 215–230.

[5] Alexander V Boukhanovsky, Valeria V Krzhizhanovskaya, and Marian Bubak. 2018. Urgent computing for decision support in critical situations. , 111–113 pages.

[6] Ronan-Alexandre Cherrueau, Marie Delavergne, Alexandre van Kempen, Adrien Lebre, Dimitri Pertin, Javier Rojas Balderrama, Anthony Simonet, and Matthieu Simonin. 2022. EnosLib: A Library for Experiment-Driven Research in Distributed Computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 6 (June 2022), 1464–1477. https://doi.org/10.1109/TPDS.2021.3111159

[7] K. Fauvel and al. 2020. A Distributed Multi-Sensor Machine Learning Approach to Earthquake Early Warning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[8] Angela H. Jiang et al. 2018. Mainstream: Dynamic Stem-Sharing for Multi-Tenant Video Processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 29–42.

[9] J. Jiang and al. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 253–266.

[10] Woo-Joong Kim and Chan-Hyun Youn. 2020. Lightweight Online Profiling-Based Configuration Adaptation for Video Analytics System in Edge Computing. *IEEE Access* 8 (2020), 116881–116899. https://doi.org/10.1109/ACCESS.2020.3004571

[11] Siew Hoon Leong and Dieter Kranzlmüller. 2015. Towards a General Definition of Urgent Computing. *Procedia Computer Science* 51 (2015), 2337 – 2346. https://doi.org/10.1016/j.procs.2015.05.402 International Conference On Computational Science, ICCS 2015.

[12] Q. Liu et al. 2018. An Edge Network Orchestrator for Mobile Augmented Reality. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 756–764. https://doi.org/10.1109/INFOCOM.2018.8486241

[13] Manish Parashar. 2024. Everywhere & Nowhere: Envisioning a Computing Continuum for Science. *arXiv preprint arXiv:2406.04480* (2024).

[14] Eduard Renart, Daniel Balouek-Thomert, and Manish Parashar. 2017. Pulsar: Enabling dynamic data-driven IoT applications. In *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE Computer Society, 357–359.

[15] C. Wang et al. 2020. Joint Configuration Adaptation and Bandwidth Allocation for Edge-based Real-time Video Analytics. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 257–266. https://doi.org/10.1109/INFOCOM41043.2020.9155524

[16] Ali Reza Zamani, Moustafa AbdelBaky, Daniel Balouek-Thomert, Juan J Villalobos, Ivan Rodero, and Manish Parashar. 2020. Submarine: A subscription-based data streaming framework for integrating large facilities and advanced cyberinfrastructure. *Concurrency and Computation: Practice and Experience* 32, 16 (2020), e5256.

[17] Haoyu Zhang et al. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 377–392.