Autonomics at the Edge: Resource Orchestration for Edge Native Applications

Ioan Petri, Omer F. Rana, Luiz F. Bittencourt, Daniel Balouek-Thomert, and Manish Parashar,

With increasing availability of edge computing resources there is a need to develop edge orchestration and resource management techniques to support application resilience and performance. Similar to the use of containers and microservices for cloud environments, it is important to understand the key attributes that characterise *edge native* applications. As edge devices increase in their autonomy and intelligence, orchestration techniques are needed to respond to changes in device properties, availability, security credentials, migration and network connectivity protocols. Implementing autonomics techniques for edge computing can increase resilience of the interaction between devices and applications reducing execution time and cost. The use of autonomics at the network edge can address the complexity requirement of industrial workflows to overcome execution latency, data privacy, and reliability constraints.

Index Terms—**Autonomics, edge computing, orchestration, resilience, cost models, resource management.**

I. INTRODUCTION

AS processing and communication capacities of edge devices increase, the need for a new computing environment to benefit from this additional capacity, closer to a user, has emerged. The complexity of applications hosted on these devices has continued to increase, including complex data processing of audio and video, to event-processing capabilities (e.g. triggers for service offloading in mobile devices). Understanding how *edge native* applications can be designed and deployed becomes a key research challenge, aligned with similar recent interest in cloud native applications development. Application developers must consider how offloading can improve quality of service levels, depending on connectivity between a user and the computing infrastructure – cloud or edge. In this context, pre-configured prior-to-use application configuration (with static behavior) is now unsuitable. Resource intensive applications (e.g. computational simulation) which are generally hosted on cloud systems can be steered through edge systems, leading to more effective use of cloud/data center resources. Application orchestration at the edge involves resource management, allocation and scheduling of application components/services to fulfil user requests. A single request may involve multiple services to acquire, transfer, and process data from different sources across multiple types of computational infrastructures.

Defining how services should be distributed across a hierarchical infrastructure from the edge to the cloud, including network elements, edge devices (e.g. Raspberry Pi) to cloud-hosted accelerators, remains a challenge. As service granularity can vary, from data analytics and sampling services to user authentication, an orchestrator to manage such service chains must take into account user requirements in terms of security, performance, and costs [1], [2]. We describe how self-management and autonomic computing can facilitate the use and coordination of edge resources – specifically: (i) we present how edge resources can be aggregated through an edge ensemble (a collection of resources that are able to operate as an adaptive group) and (ii) an industrial edge native model based on requirements from a food processing industry application – but which can be generalised to other process management/factory operations context.

This work shares a number of similarities with the use of autonomic computing for resource management in data centre-based cloud systems, however the key differences include: (i) connectivity & capability of edge resources can have significant variations; (ii) multiple resource managers can co-exist, as edge resources can exist across different administrative domains; (iii) network properties (latency, jitter, packet loss) between resources can vary significantly over time. This paper is organised as follows: Sections II provides a comparison with related approaches, followed by an overview of how autonomic techniques can be used to support edge orchestration in Section III. The application scenario we use to illustrate the proposed techniques, and experimental evaluation, is presented in Section IV. A discussion focusing on key lessons learned and conclusions from the work are presented in sections V and VI respectively.

II. RELATED WORK

Edge computing introduces computing capacity closer to IoT devices – to support increasing demand from such devices as their numbers continue to grow. In IoT, a (potentially large) set of heterogeneous devices are networked to collaborate towards a particular data monitoring/processing objective [3]. *Fog cells* is a concept adopted to define a grouping of IoT devices based on vicinity, where a device can coordinate with other IoT devices to, for example, improve performance, support data analyses, and increase security. As opposed to traditional cloud-based implementations, a Fog cell allows application services to run close to data sources and sinks, reducing communication delays and improving resource efficiency at edge devices. Pre-processing and data aggregation

I. Petri is with School of Engineering, Cardiff University, Wales, UK (email:petrii@cardiff.ac.uk).

O. F. Rana is with School of Computer Science and Informatics, Cardiff University, Wales, UK (email:ranaof@cardiff.ac.uk).

L. F. Bittencourt is with the Institute of Computing, University of Campinas, Brasil (email:bit@ic.unicamp.br).

D. Balouek-Thomert and Manish Parashar are with School of Computer Science, Rutgers University, New Jersey, USA (email:parashar@rutgers.edu).

from sensor node streams [4] as well as data processing for smart systems [5] are examples of applications that can take advantage of Fog infrastructures. Mobile devices can also act as a dynamic geodistributed computing infrastructure [6]. The Cloud of Things [7] proposes the introduction of *in-network* processing between IoT and Cloud systems. Consequently, operations over data (e.g. running queries in complex event processing – CEP) can occur in the path between data capture and cloud-based processing [8].

Edge ensembles have also been investigated to aggregate computation from different combinations (groupings) of edge resources. Such an ensemble has demonstrated benefit by delivering workflow execution efficiency with data protection and security [9]. The use of an orchestrator also enables execution of a service function chain, where functions are physically hosted on different devices. Such capability can be configured based on user and application constrains as a mean to respond dynamically to new data flows [10].

III. Autonomic Edge Orchestration

The autonomic edge comprises resources $E = \{e_1, e_2, e_3, .., e_m\}$, where each e_i can include processing units, storage, and *type*. Resource nodes can be grouped based on their type, using (similarity in) properties in relation to proximity, latency and/or cost. A set of processes $P = \{p_1, p_2, p_3, .., p_m\}$ are enacted over edge resources E with a view to increase resource efficiency and distribute workload across edge nodes. A process p_j is an executable function/operation that can be deployed over one or more e_i. The notation used in this work is based on [8]. The autonomic approach described in Figure 1 has the following components:

1) Monitoring and feedback: a feedback control loop that collects data from the three layers: cloud, fog nodes, and edge devices, and leads to orchestration across these layers in a unified way. Understanding dependencies between cloud, fog, and edge resources is essential to ensure that resources are used efficiently. As edge resources are represented as an ensemble e_i, monitoring of node properties is restricted to members of e_i.
2) Strategies: monitoring leads to event triggers that generate control actions – as illustrated in Figure 1. These actions can trigger one of three behaviours: (i) updates to the ensemble – i.e. addition/removal of resources; (ii) use of approximation techniques that enable "steering" actions to be carried out on the supported fog and cloud resources; (iii) adaptation techniques that lead to resource configuration updates, but which require use of learning techniques based on usage data of edge resources. These three behaviours, on their own or in combination, provide the key autonomic capability within the system.
3) Orchestration: involves carrying out the behaviour modification strategies in (ii). The orchestration involves the use of a number of controllers that interact with local schedulers in cloud, fog, and edge resources.

The ability to orchestrate resources at the edge represents a key advantage in heterogeneous edge environments where devices have different properties and different operating goals. As IoT devices can be geographically distributed, the orchestrator needs to take account of local requirements but also to coordinate the entire ensemble of edge resources. Therefore, an orchestrator can run based on two different types of goals:

- Local goals – refer to particular edge device objectives that need to be considered in a local (constrained) edge environment. Local goals (e.g. improve utilisation or reduce energy usage) are usually application specific, but need to be balanced with overall demands specified as part of one or more global goals.
- Global goals – identify more advanced strategies that an edge orchestrator needs to ensure in the wider management of edge resources. As an orchestrator may coordinate an ensemble, the global goals are intended to maximize the performance of the entire edge infrastructure while complying with the local goals identified over an ensemble.

In a smart factory scenario involving energy simulation, local goals can identify the period over which a simulation is to be carried out in relation to a number of simulation instances that can impact the quality of results. When an artificial neural network (for instance) is used, training and testing should take place in combination with a simulation process. For neural network learning, the local goal can be related to the error rate based on the quality and consistency of the historical data. From a global perspective, the edge orchestrator needs to achieve a trade-off between local goals within an edge ensemble, while providing an improvement in the global goals at the scale of the entire edge infrastructure. In a smart factory scenario, for example, where different buildings conduct specific industrial activities, the global goal can involve minimization of data transfer between various edge devices, which may be distributed throughout the factory. From Figure 2 we identify the following layers based on a fish factory example scenario.

- *The energy layer* – identifying energy production units such as photovoltaic panels and the power grid (with associated meters).
- *The appliance layer* – identifying a set of appliances necessary to clean, store, and process the fish in the factory. In our scenario we consider the following appliances: (i) cold room, (ii) ice-flake and (iii) box washing machine, (iv) lighting systems, (v) battery storage system.
- *The edge layer*– identifying Raspberry Pi nodes hosting optimization modules that periodically actuate and schedule physical appliances. These nodes are co-located with the appliance layer.
- *The cloud layer* – formed of cloud computing nodes that execute energy optimisation/simulation tasks.

Autonomics can be applied in the following way – the application and associated edge infrastructure is explained in section IV:

(i) Resource management mechanisms for training/executing neural network tasks on cloud resource when edge resources are congested – e.g. based on cost of execution, time-to-execute, and orchestration time.

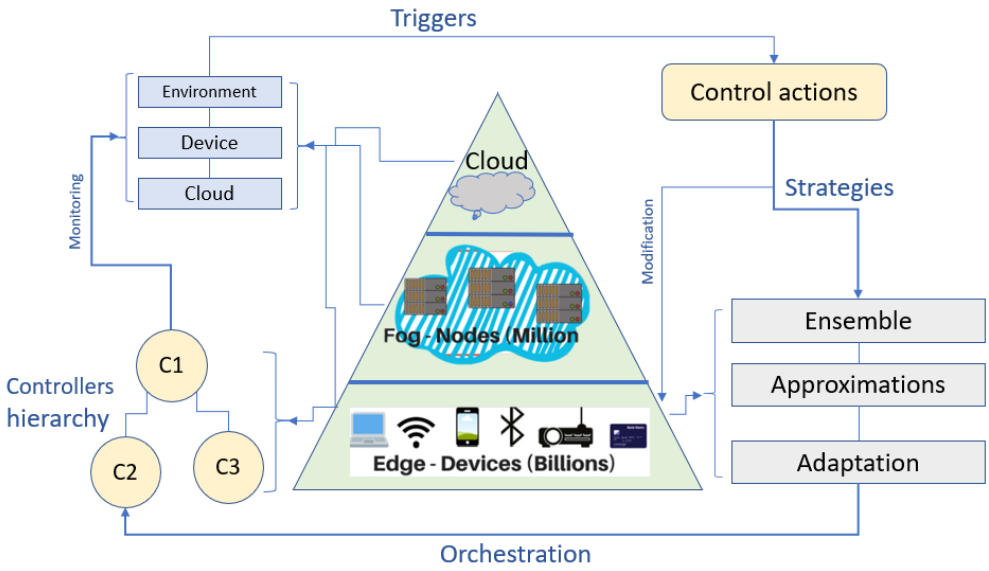

Fig. 1. Integrated edge layer

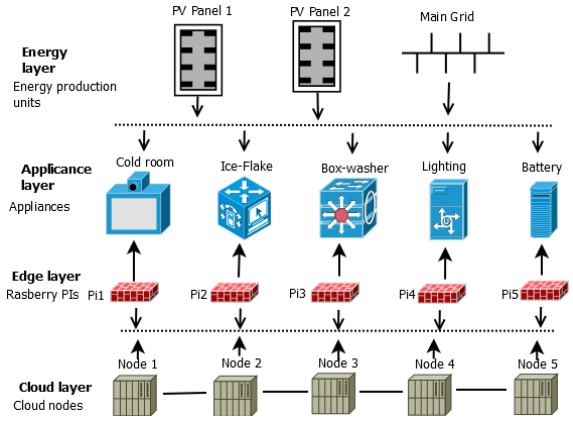

Fig. 2. The industrial edge layers

(ii) Performance improvement by migrating from edge to cloud resources when performance degrades. Such strategies include offloading tasks based on time-to-execute, execution cost, and deadlines.

(iii) Ensemble formation by combining edge resources to collectively execute tasks (e.g. training or execution of a neural network or federated learning tasks) and sharding data between different nodes.

(iv) Mobility support for more effective use of edge devices closer to a user as their device migrates. This involves migrating data and processing along the path of movement, following or anticipating these paths, so the latency and number of hops between users and their data is minimised. Data and computational task migration decision-making should consider application and network changes, as well as edge resource workloads [11].

(v) On-demand edge federation to support resource grouping based on an aggregation of edge and cloud capabilities. Cloud systems can be extended towards the network edge through a federation of providers and systems with different specifications and locations. Such a cloud continuum provides flexibility and adaptation to respond to task queue congestion, cost efficiency, and reliability.

(vi) Resilience that accounts for availability (and uptime) of edge resources (i.e. some edge resources fail) and network performance (communication with an edge device). This can occur if: (a) edge resources fail – requiring other edge resources in the ensemble to overcome the effect of failure, i.e. replicate execution on multiple resources and (b) network fails, i.e. where the network connecting the edge resources and the cloud platform is available: (i) intermittently – i.e. the connection bandwidth varies over time significantly; (ii) fails with no transfer capacity being available. Such metrics can involve cost, time-to-execute, failure rate, re-sync rate, etc.

(vii) Placement of services on edge or cloud resources impact overall user latency and response time. The key idea is to consider: (i) full edge-ward placement of services; (ii) full cloud-based placement (iii) movement of services between the edge and cloud–driven by performance and cost considerations.

IV. Smart Factory Scenario

We first consider the building energy optimisation scenario which makes use of EnergyPlus simulation [12]. Our proposed edge orchestrator has the following objectives:

1) measure performance for task deployment at the edge of the network and mechanisms to use when such deployment takes place in an industrial application scenario (i.e. tasks properties, time constraints, etc)
2) investigate execution of EnergyPlus simulation in a *native* edge environment involving multiple edge resources and shared executions; and
3) demonstrate the use of the autonomic edge to coordinate tasks that require execution closer to data source but also addressing a specific quality-of-results. Task allocation can be dynamically adapted based on tasks properties, using the edge orchestrator.

A. Simulation scenarios

The Milford Haven port is the largest port in South Wales where a fish processing industry generates increased energy demand with long operational times and intensive energy operations. The port has an annual energy consumption of about 1600 MWh, and produces carbon emission of about 790 tonnes. The key objective in the port is to reduce energy consumption and to eliminate carbon emissions. The Packaway (main building in the port) contains multiple energy consuming appliances that need be optimised in terms of energy consumption: a flake-ice machine, a cold-room, a box-washing machine, the lighting systems, and a battery systems. Each of these devices has augmented edge computing capability to process operational data – as explained below. These edge resources are able to receive data directly from these applicances and carry out an initial analysis.

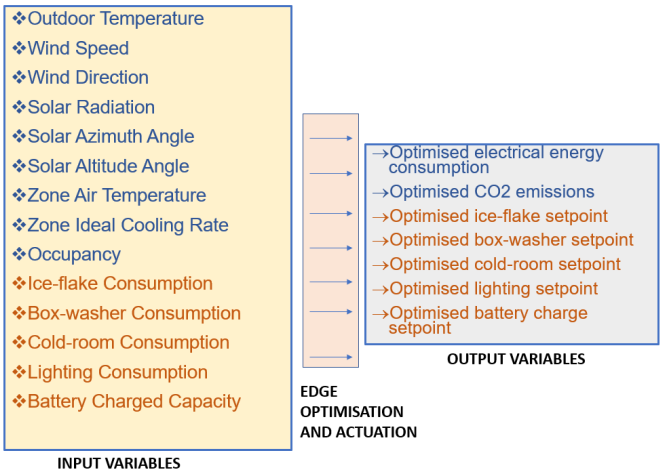

Fig. 3. Edge optimisation scenario for industrial appliances

The box washing machine– has a 50KW capacity and is active for short time intervals within a day. An efficient operation schedule and set-point needs to be identified in order to optimize the energy consumption of this appliance.

Lighting system– consumes 400W and is used when the actual fish processing operations are conducted during day and night schedules. An optimum schedule and set-point needs is needed to reduce energy consumption.

Flake ice– produces small pieces of ice for use in the cold storage to preserve the fish. The ice-flake consumes 32KW and requires an optimum operation schedule and an optimum set-point to support fish processing demand.

Cold storage– operates over a 24 hours period to meet fish processing demand and is considered to be the most power-consuming device in the building. The temperature is maintained between -10 and -55 degrees Celsius. Based on cold storage cycle, an optimised schedule and temperature set-point can significantly improve the energy efficiency and support fish operations in the building.

Battery system– comprises (i) main battery with a capacity of 6000Ah and (ii) a secondary battery of 4000Ah used to store excess of energy in production periods. The optimisation objective is to determine the correct amount of energy that needs to be stored on the two batteries.

The main objective is to determine the optimised set-points and actuate these on the five appliances using simulation (executed every 15 minutes). An orchestration between all appliances is required to determine operation schedules in the Packaway building. The actual optimisation process is executed on the Rasberry Pis linked to the appliances as in Figure 2 – and which constitute the edge devices used in this system. The range of set-points are: (i) On/Off for box-washer, ice-flake and lighting, (ii) optimum temperature set-point for the cold-room, and (iii) optimum quantity of energy to store in the battery storage system. Figure 3 presents the optimisation scenario with several variables forming the optimisation – with input variables related to energy generation (e.g. solar azimuth, wind direction and speed, outdoor temperature, etc) and energy consumption of appliances.

B. Autonomics testbed and scenarios

We consider an industrial edge environment with five Rasberry Pi (RPi) that run energy optimisation tasks. We run different optimisation scenarios that demonstrate the use of autonomics in the industrial edge context with five appliances that require optimised set-points. We use CometCloud [13] to form a federation between different edge and cloud environments. The experiment testbed makes use of two different types of resources: (a) computing resources identifying (i) five HPC nodes forming a cloud system able to accommodate energy optimisation tasks, and (ii) five edge resources (RPi) forming an industrial edge layer hosting optimisation tasks closer to data source; and (b) industrial resources identifying (i) an appliance layer identifying a set of industrial devices, and (ii) an energy production layer with different energy production units. Our industrial edge layer aggregates computing resources across different appliances, where each optimisation task can be executed in the local edge resource and across the entire ensemble of edge resources.

C. Experiments

We conduct our experiment on a real industrial testbed with multiple appliances deployed at the factory. We seek to evaluate and implement optimised appliance set-points with a view to reduce energy consumption in the pilot project through the use of edge-autonomics. The experiments are applied to the Packaway building with the set of appliances that require different types of optimisation tasks as presented in Table I.

TABLE I
TASK PROFILES

Job Type	Data Size	No. of Tasks
Type1	50MB	16
Type2	100MB	24
Type3	150MB	32

We apply a cost perspective by calculating a total cost with computation in relation to storage, transfer, and execution of tasks.

$$Cost = exec._{time} * cost_{excution} + net._{tranfer} \\ *cost_{transfer} + storage_{time} * cost_{storage} \quad (1)$$

where $cost_{execution}$ refers to CPU processing time required, $cost_{transfer}$ measures the amount of data being transferred and $cost_{storage}$ identifies the cost of storage. We calculate the costs based on Amazon EC2 pricing schemes in dollars ($). To support the experimental scenarios, we use the following configurations:

- **Traditional Cloud Only (T-C)**: The cloud configuration involves 5 HPC nodes that have capability to execute computationally demanding tasks. Such configuration identifies regular scheduling strategies where tasks are allocated for execution randomly based on the availability of nodes. Tasks execution criteria is related to time and budget constraints and access to edge resources is disabled.
- **Edge Resources (ER)**: The edge configuration assumes that cloud and autonomic mechanisms are disabled. The edge nodes can be grouped (in our example we use an ensemble formed of 5 Rasberry Pis). The edge system is located in proximity of data source(s).
- **Edge Autonomics (EA)**: The role of autonomics at the edge is to coordinate edge resources based on incoming tasks properties and user requirements. We consider different tasks types with associated deadlines that are managed by an orchestrator with a view to optimise costs and efficiency in relation to execution. The autonomics also involve communication between different nodes based on a set of incoming tasks requirements.

a) Data Transfer Time: The aim of this experiment is to show the edge autonomics capability to improve the overall data transfer time. This experiment emphasizes that proximity to data source can play a key role for tasks execution. As the edge layer is closer to data capture devices, transfer is often not required whereas for clouds multiple network hops need to be transited before the actual tasks execution can take place. Figure 4 shows the transfer time required to compute different types of tasks (see Table I) and demonstrates that an edge configuration delivers the lowest data transfer time, with edge-autonomics requiring additional data transfer for the coordination of the edge resources. A Cloud-based deployment shows a linear increase in data transfer time based on the number of hops in the network path, and the complexity of the tasks to execute. This experiment shows that autonomics at the edge can ensure a trade-off related to data transfer showing that tasks with an increased size of data do not necessarily generate an increase in the transfer time.

b) Completion time: The objective of this experiment is to demonstrate the impact of edge autonomics in terms of completion time and task complexity. In edge environments, time constraints are tied up to the quality of results, therefore providing comparing analysis for clouds, edge and edge-autonomics systems from a time perspective can lead to more informed decisions for users. The results reported in Figure 5 show that edge and edge-autonomics have increased time completion based on different types of tasks. When tasks identify a higher number of sub-tasks such as “type 2” and “type 3”, edge autonomics seem to deliver a lower completion time comparing to a regular edge configuration. Clouds have a linear increase for different task types and their performance depends on the complexity of the workflow. This experiment demonstrates that edge-autonomics can bring an improvement in the task execution process, as task complexity (identified in task data size and number of sub-tasks) seems to have a small impact on the completion time, showing that orchestration at the edge can keep the overall completion time to reasonable levels. Therefore, autonomics at the edge can support a more efficient orchestration of resources which leads to improved completion time, whereas cloud and edge systems use a more generic task allocation which incurs an increase of the time to complete.

c) Cost with task execution: To demonstrate the benefits of implementing autonomics at the edge, we seek to understand the impact of the task execution in the actual computing cost. We consider different cost components such as storage, transfer, and execution to calculate the impact of deploying tasks at the network edge using three different scenarios such as clouds, edge and edge-autonomics. Figure 6 demonstrates how cost evolves with different tasks types and associated size of data. The experiment shows that using lighter tasks (i.e. small number of sub-tasks) is more advantageous in terms of cost than when using more complex tasks (i.e. higher number of sub-tasks). In a native edge environment storage and transfer costs are low therefore the impact in terms of costs is also limited. When using an edge-autonomics configuration, multiple coordination strategies are required with managing decentralised edge resources which incur a higher cost. However, for large edge ecosystems with an increased number of nodes such edge-autonomics is beneficial in terms of resource orchestration and tasks scheduling. Clouds identify the higher cost as pricing of resources in cloud systems is more expensive.

V. Improving Adoption: Autonomics in Edge Applications

In edge native applications, the user and the edge system are interacting based on various objectives, where such objectives have particular cost implications which must be taken into account. In industrial edge applications, the resilience of the edge infrastructure depends on the WiFi access points and accessibility of services, but is also subject to costs that need to be covered usually by (i) cloud providers; (ii) local businesses; or (iii) from public funding [14]. Such cost models are based on a demand/workload profile of an application. From an autonomics perspective, there are several generic principles which characterise edge infrastructures, that once satisfied, can enable a wider adoption. Such principles include:

1) Usage – the autonomics of edge infrastructures need to deliver a set of common usage policies and utilise generic management techniques to support required computing operations and running time.

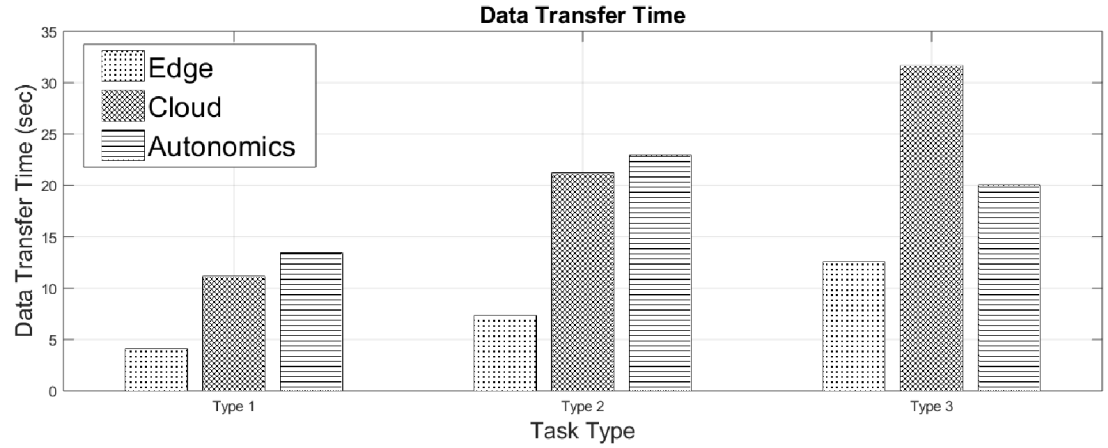

Fig. 4. Data Transfer Time.

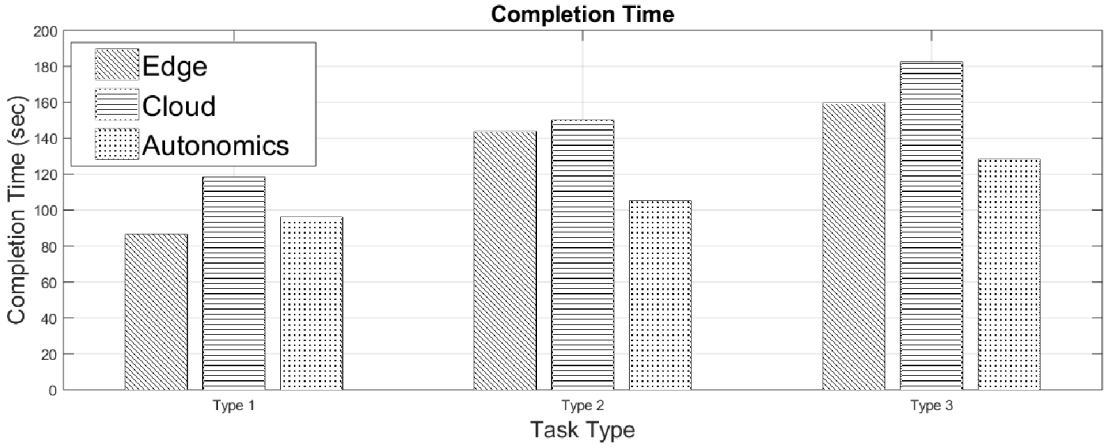

Fig. 5. Completion Time.

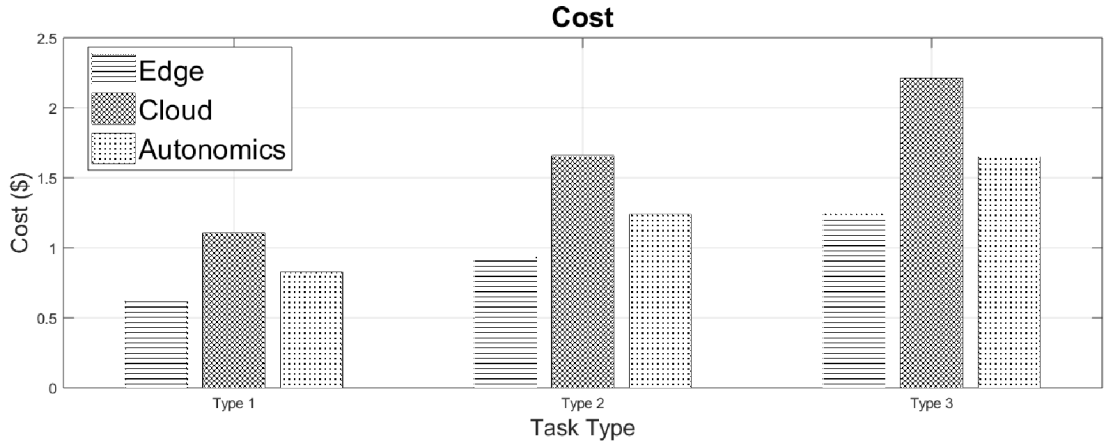

Fig. 6. Cost with task execution

2) Availability – referring to the ability of edge infrastructures to deliver a certain Quality-of-Service and accessibility for potential users. Techniques for maximising availability identify service replication or service migration in relation to a set of application requirements. Availability also embeds some adaptation techniques associated with dependability such as readiness for usage, reliability, and maintainability.
3) Cost – the use of autonomics should be support cost efficient process that can be attractive for businesses while also having flexibility in terms of pricing models based on application requirements.

4) Performance – edge infrastructures should have the ability to accommodate different workloads and associated scaling methods in relation to the type of application. Performance also refers to system responsiveness identifying the time required for the system to respond to processing events.
5) Security – autonomics should enable certain flexibility of the edge infrastructures while delivering a high level of security, especially for applications involving data confidentiality and sensitivity. Such security clauses need to be specified in terms of confidentiality, integrity, and availability.

The autonomics and self-adaptability of edge applications can be considered with respect to functional and non-functional properties commonly used to assess the reliability of a computing infrastructures. The functional properties that an edge infrastructure needs to expose are [15]: (i) self-configuration – applies to automated device initialization at run time. It refers to the ability of an edge infrastructure to automatically reconfigure based on high-level rules and triggers; (ii) self-optimization – represents the system's capacity to continually improve the value of non-functional resources (i.e., quality attributes such as efficiency, or resource use such as power consumption) in relation to a set of incoming application requirements or business objectives; (iii) self-restoration – refers to the system's capacity to locate and restore malfunctions on its own at run-time; (iv) self-protection – identifies the system's capacity to react and eliminate malicious attacks or intrusions by implementing different trust and protection strategies.

VI. Conclusion

Edge computing models and services are adopted for industrial applications as a mean to address time constraints and proximity to data source requirements. Although the commercial edge computing market is still maturing, it is important to consider the types of techniques that are required to orchestrate and ensure a level of autonomics in edge environments. Whereas previously resources available at the edge of a network have been utilised to capture and collect data, the increase in performances and pervasive nature of the edge has enabled hosting of analysis and tasks execution at the edge for different applications. The possibility to create a hybrid edge-cloud environment represents a significant advantage for applications with sensitive constraints such as quality of results or time-to-complete often identified in an industrial process. As industries have entered in a transition from old centralised operations to decentralisation of energy management, the orchestration of edge resources at the network edge for energy optimisation can provide a competitive advantage on the market and a higher order of cost efficiency. In relation to non-functional properties, adding autonomic capability to edge infrastructures also supports: (i) stability, in terms of system behaviour and resilience, (ii) improved accuracy and convergence for application outcomes (e.g. error rate in a neural network), (iii) a mechanism to address heterogeneity of edge resources, e.g. replacement of devices and components.

Acknowledgement This work was supported by the EU INTERREG piSCES Project: "Smart Cluster Energy Grid Systems for Fish Processing Industries", grant number: 504460.

References

[1] G. Douglas, B. Drawert, C. Krintz and R. Wolski, "CloudTracker: Using Execution Provenance to Optimize the Cost of Cloud Use", Proc. GECON conference, Sept. 16-18, 2014. Cardiff, UK. Springer.
[2] Jorn Altmann, Mohammad Mahdi Kashef, Cost model based service placement in federated hybrid clouds, Future Generation Computer Systems, Vol. 41, 2014, pp 79-90.
[3] L. Atzori, A. Iera, G. Morabito, "The internet of things: a survey". Computer Networks 54:2787–2805, 2010.
[4] A. V. Dastjerdi, H.Gupta, R. N. Calheiros, S. K. Ghosh SK and R. Buyya, "Fog computing: principles, architectures, and applications". In: Internet of things: principles and paradigms, chap. 4, MorganKaufmann, 2016.
[5] S. Rohjans, C. Dnekas and M. Uslar, "Requirements for SmartGrid ICT-architectures". In: 3rd IEEE PES international conference and exhibition on innovative smart grid technologies. Berlin, Germany, pp 1–8, 2012.
[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in Big Data and Internet of Things: A Roadmap for Smart Environments. Springer, 2014, pp. 169–186.
[7] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in Intl. Conference on Future Internet of Things and Cloud (FiCloud). IEEE, 2014, pp. 464–470.
[8] A. R. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, and M. Parashar. A computational model to support in-network data analysis in federated ecosystems. Future Generation Computer Systems, 80 (2018): 342-354.
[9] I. Petri, A. R. Zamani, D. Balouek-Thomert, O. Rana, Y. Rezgui, and M. Parashar. "Ensemble-based network edge processing." In IEEE/ACM 11th Int. Conf. on Utility and Cloud Computing (UCC), pp. 133-142, 2018.
[10] I. Petri, O. Rana, A. Zamani, Y. Rezgui, "Edge-Cloud Orchestration: Strategies for Service Placement and Enactment". In IEEE Int. Conf. on Cloud Engineering (IC2E), 24 June 2019 (pp. 67-75).
[11] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar. Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, 4(2):26–35, March 2017.
[12] V. Garg, K. Chandrasen, S. Tetali, J. Mathur, "Energyplus Simulation Speedup Using Data Parallelization Concept." ASME Energy Sustainability Conf., New York: American Society of Mechanical Engineers. pp. 1041-1048, 2010.
[13] H. Kim and M. Parashar, "CometCloud: An Autonomic Cloud Engine," Cloud Computing: Principles and Paradigms, Wiley, 2011, pp 275-297.
[14] L. F. Bittencourt., M. Lopes, I. Petri, O. F. Rana, "Towards Virtual Machine Migration in Fog Computing", 10th Int. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing, November 4-6, 2015, Krakov, Poland.
[15] N. Villegas, G. Tamura, and H. A. Müller. "Architecting software systems for runtime self-adaptation: Concepts, models, and challenges." Managing Trade-offs in Adaptable Software Architectures. Morgan Kaufmann, 2017. 17-43.