On Distributed Computing Continuum Systems

Schahram Dustdar, *Fellow, IEEE,* Victor Casamayor Pujol, Praveen Kumar Donta *Member, IEEE,*

(Invited Paper)

Abstract—This article presents our vision on the need of developing new managing technologies to harness distributed "computing continuum" systems. These systems are concurrently executed in multiple computing tiers: Cloud, Fog, Edge and IoT. This simple idea develops manifold challenges due to the inherent complexity inherited from the underlying infrastructures of these systems. This makes inappropriate the use of current methodologies for managing Internet distributed systems, which are based on the early systems that were based on client/server architectures and were completely specified by the application software.
We present a new methodology to manage distributed "computing continuum" systems. This is based on a mathematical artifact called Markov Blanket, which sets these systems in a Markovian space, more suitable to cope with their complex characteristics.
Furthermore, we develop the concept of equilibrium for these systems, providing a more flexible management framework compared with the one based on thresholds, currently in use for Internet-based distributed systems. Finally, we also link the equilibrium with the development of adaptive mechanisms. However, we are aware that developing the entire methodology requires a big effort and the use of learning techniques, therefore, we finish this article with an overview of the techniques required to develop this methodology.

Index Terms—Distributed Systems, Computing continuum, Edge Computing, Markov Blanket

1 Introduction

The last decade has been dominated by applications based on Cloud infrastructures, however, during the recent years we are witnessing the appearance of new computing tiers, such as the Edge or the Fog. They provide valuable features to applications, such as very low latency or privacy enhancements. Ultimately, new applications are emerging that take advantage of all the computing tiers available, hence, they are systems that are simultaneously executed on the Edge, Fog, and Cloud computing tiers. These systems are known as "computing continuum" systems.
The applications provided by the distributed "computing continuum" systems enable developments that belong to the infrastructure landscape for some years now, such as smart cities with greener and sustainable spaces; non-polluted avenues with autonomous vehicles; sustainable and efficient manufacturing with accurate traceability of products, or health systems able to personalize anyone's treatment independently of their current location.
Most research efforts have been devoted to solve specific challenges of these systems such as data caching, resource allocation or scheduling among others. However, general methodologies for design and management of these systems remain one of the fundamental challenges. This has been overlooked because the community has been using the traditional methods for design and management developed for the first Internet-based systems, those comprising a server and a client, which are completely specified through the software application itself. In general, these systems are being developed from a top-down perspective. Hence, each system's architecture is defined ad-hoc to solve a precise problem. This methodology is still valid for the Cloud paradigm, but it falls short for the "computing continuum". For instance, the concept of developing an architecture for a system needs to be re-thought for "computing continuum" systems. One can find a manifold of different architectures within the same system. Also, in some situations the desired architecture will not be possible to implement, and from a management perspective, the mechanisms developed need to be able dealing with situations, where the architecture is another dynamic feature of the system.
We claim that new methodologies are required as "computing continuum" systems have a fundamental characteristics that make previous methods inadequate: the application and the underlying software infrastructure are seamlessly blended. Therefore, system characteristics are driven by their infrastructure, which are similar to the ones of a complex systems, as cited in [1] *a system is complex if its behavior crucially depends on the details of the system*.
We foresee a methodology that takes into account the shared characteristics of all such systems and, changes the role of the underlying infrastructure's software. Hence, with this change of mindset and by leveraging each system's infrastructure data, we will be able to manage these systems obtaining robust, adaptive, and cooperative systems.
This article's objectives are twofold. First, to present a general methodology inhered from cloud computing showing its limitations in the "computing continuum" context. Second, to unravel that a completely new approach is required to design and manage these systems, sketching our vision on how we foresee this new methodology.
The rest of this article is organized as follows, in section 2 we show related work for managing cloud systems, then in section 3 we explain the reasons why we need a new approach for managing the "computing-continuum" systems. Section 4 details our vision for the management approach and underlines the required learning techniques that will be required to develop it. Finally in section 6 we present our conclusions.

Authors are with Distributed Systems Group, TU Wien, 1040, Austria

2 Related work

In recent years research effort aimed at developing Edge and Fog tiers. Through a review on literature one can find three different approaches.

First, there is research targeting specific topics of these tiers, such as data caching [2], [3], service deployment [4], computation offloading [5], [6], resource allocation [7], scheduling [8], [9], traffic grooming [10] or service decomposition [11], among others.

Second, there is research that addresses these specific topics but they are contextualized in a specific domain. For example, in [12] a QoE and energy-aware methodology for offloading is contextualized in a smart city, or in [13] a scheduling solution is presented in a smart manufacturing context. Similarly, in [14] the scheduling is taken to the healthcare application.

Third, there is research that focuses on general architectures for specific contexts, such as in [15], where they describe a specific fog architecture for banking applications or in [16] where architectures for a healthcare system are presented.

But, to the best of our knowledge, there is a shortcoming for transversal and general methodologies for managing "computing continuum" systems. Specifically, transversal methodologies apply to any application domain and general methodologies are able to solve any specific issue.

In Cloud computing, which has been on research for a couple of decades now, there is a methodology that we claim is required for the "computing continuum" systems. In this regard, the work on elasticity [17]–[19] develops a methodology for managing Cloud systems that matches the need for the "computing continuum" systems in terms of generality and transversality. This starts by abstracting any application with three variables: *Resources*, *Quality* and *Cost*. Then, the system keeps track at the values of these variables and if any of them goes beyond a certain threshold, elasticity strategies are applied. Simply put, these change the application configuration in the cloud infrastructure, so that its current needs are matched with the expected performance, and therefore, the *Resources*, *Quality* and *Cost* variables return at their expected range of values. It is worth mentioning that in [20] some difficulties are already identified when dealing with multi-cloud systems.

Now the evolution of Cloud computing is going towards the server-less computing [21], this basically implies that the thresholds, commonly known as Service Level Objectives (SLOs) that controlled the application state, require a higher level of abstraction. This is more convenient for cloud users, as they do not longer require to determine low-level specifications of the application, such as the range of CPU usage, but higher level ones such as the ratio between the cost of the infrastructure and the efficiency of the application, called cost-efficiency as developed in this work [22]–[24], which proposes a Cloud management system completely controlled through high-level SLOs.

A similar approach is required to manage "computing continuum" systems, however, the characteristics of the underlying infrastructure of such systems make this Cloud computing approach not adequate.

Before continuing it is worth detailing two concepts that will be very recurrent along the article. A "computing-continuum" application only refers to an application developed on top of the "computing-continuum" fabric of resources, which can range from a video analysis or an entire vehicle fleet management for a smart city. A "computing-continuum" system refers to all the resources required to enable an application of the "computing-continuum", this includes also the application itself but just as another component of the system.

3 System management in the Cartesian space

In this section we will discuss how the Cloud computing management system could be applied to "computing continuum" systems. But once there, by reasoning about the characteristics of these emerging systems, we will be able to show why this methodology is not sufficient.

3.1 Cartesian blanket

Similarly as done with the Cloud, the system space of a "computing continuum" application can be represented with the three variables: *Resources*, *Quality* and *Cost*. Hence, it can be represented in a three dimensional Cartesian space. Cloud based systems can be usually abstracted as virtually unlimited and homogeneous source of resources, which simplifies the system's Cartesian representation using *Resources*, *Quality* and *Cost*. However, given the heterogeneity of "computing continuum" systems, their encoding on this space is not straight forward.

Nevertheless, we can assume that given a domain this Cartesian frame could be shared. Simply put, we foresee that healthcare applications can have the same quality axis, as it can be interpreted similarly for all cases. But this might not be the same case for an application devoted to control product distribution. In any case, it is conceivable to develop transformations between Cartesian frames to relate applications from different contexts. Ideally, formalizing relations between systems from different domains could allow the further development of transversal methodologies seeking, for instance, cooperation between systems.

Then, Cloud systems use SLOs to check if the system is performing as expected. In this regard, only high-level SLOs can be represented in the Cartesian space, as it is a high-level abstraction of the system and low-level SLOs can not be represented.

In general, high-level SLOs will be expressed as lower and higher boundaries for each of the axis. For instance, two limits on the *Cost* axis represent the range in *Cost* that the "computing continuum" system can assume. One could argue that *Cost* could only have an upper limit. However, it is important to take into account that by simply using infrastructure there is an associated *Cost*, and actually, being below a threshold could imply that there is some required infrastructure component that is not being properly used, hence, the system performance could be endangered.

Visually, this develops an hexahedron on the Cartesian space that represents the available configurations for the system state. Hence, within that space the system is operating with its specified characteristics. Unfortunately, this is not entirely true.

As previously mentioned, "computing continuum" systems do not have an "unlimited" pool of homogeneous infrastructure resources, as it is usually assumed for the Cloud. Therefore, their operative system's space is linked to the actual system's infrastructure. In other words, the underlying infrastructure of the system has to be represented in the Cartesian space to understand the possible configurations for the system's space.
Infrastructure has fixed characteristics of *Resources*, *Quality* and *Cost*. Hence, they are represented as points in that space. At this point one could argue that the same infrastructure can have different characteristics, and this is true for the Cloud, where you can have virtual machines with different specifications depending on the need, making the space fully continuous. But, as the infrastructure is approaching the Edge, the resources are limited, and therefore, we assume that using an infrastructure is using all its capabilities. Then, the system's infrastructure can be added as points inside the Cartesian space. Certainly, infrastructure that is out of the hexaedron will not be adequate for the "computing continuum" system.
Therefore, the possible configurations for the system state space can be visually interpreted as a stretched blanket linked to the infrastructure points and limited by its specification hexahedron, as can be seen in Figure 1.
Finally, an elastic strategy, which can be understood as a reconfiguration the system's underlying infrastructure, is put in place if the system state is outside of its SLO defined hexahedron, to make the system return into its space.

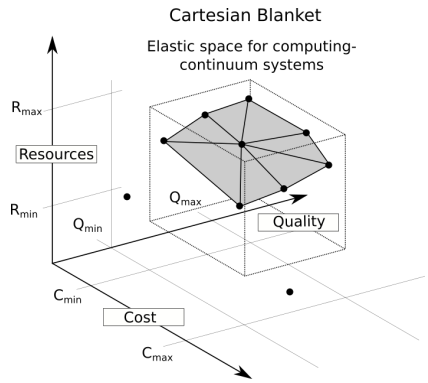

Fig. 1. The figure shows the three axis of the Cartesian space: *Resources*, *Quality* and *Cost*. For each of them there is a couple of thresholds that limit the system state from a requirement perspective. On top, the hexaedron generated from the thresholds is sketched, inside the Cartesian blanket is drawn by linking the points that represent the system's underlying infrastructure.

3.2 "Computing continuum" characteristics

In subsection 3.1, it has been already observed that "computing continuum" systems have different characteristics than Cloud systems. In this regard, characteristics of "Computing continuum" systems are inherited from its underlying infrastructure, similarly as it happens with the Cloud, but the latter can abstract its infrastructure as virtually unlimited, homogeneous and centralized, which simplifies its analysis and allows a management system as the one explained in section 2.

On the contrary, "computing continuum" systems' infrastructure is large, diverse and distributed. These systems can have applications that target an entire city, which means that its infrastructure is spread through the city with components ranging from sensors or micro-controllers up to large computing units or robots.
Another characteristic of distributed "computing continuum" systems is that its performance depends on many interactions between its components. Hence, the application pipeline is large and contains many ramifications. This, together with the size of the system makes it fragile, as it becomes difficult to know which part of the pipeline might break first and how this issue will propagate. Furthermore, this impedes knowing with certainty how this issue will affect the system's performance.
Finally, the system has to be considered as an open system. This means, that the system is influenced by environmental events. To put it simple, network congestion or a high increase of users request can unpredictably occur, as they can be consequence of a crowd gathering due to a non-related social event. Hence, this creates a high level of uncertainty with respect to which components can stop servicing as expected and break the application's pipeline.
The combination of the described characteristics makes "computing continuum" systems behave similarly as complex systems, therefore, the required methodology to manage them has to take this into account, which completely changes the paradigm with respect previous Internet systems.

3.3 Discussion

The previous analysis of the system state representation, its management methodology and the characteristics of these systems have opened a set of issues that have motivated the need for a new methodology for representing and managing "computing continuum" systems.

3.3.1 Reactive management system

The first issue is that the managing system is reactive. Hence, the corrective action is performed once the system state is outside the configured thresholds. This approach is not valid for a system that behaves similarly as any complex systems. Their propensity to have cascade errors makes the reactive action useless as derived errors arise.
It could be argued that the previous can be solved by taking the action before, so by narrowing down the thresholds. But this has other implications, as it forces to constraint the state space of the system, which implies much less design flexibility in terms of *Resources*, *Quality* and *Cost*. To sum up, thresholds are not a suitable solution for "computing continuum" systems.

3.3.2 System's stability linked to infrastructure

One can also realize that once the state space of the system is out of the Cartesian blanket stitched to the infrastructure points, the system is no longer under control, as at least a component of its state is not properly linked to its current infrastructure state. Hence, the stability of the system does not depend on the location of the system state with respect to the SLOs frames, but with respect to its underlying

infrastructure. This different behavior with respect to Cloud systems stems from the fact that the infrastructure components are not as flexible as in a Cloud based system. In this regard, their relation with the underlying infrastructure needs to be further developed.

3.3.3 Unknown system derivatives

Following the previous discussion, it can be argued that the managing action has to be performed as soon as the system state slightly deviates from its Cartesian blanket space. However, this reveals another shortcoming of this representation. The Cartesian representation does not encode the system derivatives, in other words, it is a representation that does not provide information on the evolution of the system. The characteristics of the system do not allow to define derivatives of the state in order to understand whether a small deviation is negligible or if it is actually going to further deviate from its state. And given that these systems are not free from environmental noise it is relevant to understand the nature of the deviation. Furthermore, in such a case, the performed action needs to be very carefully sized as it could aggravate the situation. To sum up, "computing continuum" systems require to include the sense of evolution in their management mechanisms.

3.3.4 Lack of causality relations

This also relates with the last shortcoming identified on this representation, which is the lack of the notion of causality. The actions that can be taken in the cloud computing paradigm are basically vertical or horizontal scaling or a combination of both. However, the infrastructure of the "computing continuum" systems requires more detailed actions, as they depend on each infrastructure component. Therefore, understanding the cause of a deviation on the system's state is required in order to act to overcome it, which implies that a causality model is required.

4 System management in the Markovian space

This section first develops our vision for the new approach to manage distributed "computing continuum" systems and then it provides research lines with respect to learning techniques that can be used to fully develop the methodology.

4.1 Vision

4.1.1 System state

Understanding the state of the system through high-level variables provides an abstraction that is useful in order to generalize to any type of system. Furthermore, it facilitates the communication between infrastructure providers and developers of applications. Therefore, our vision keeps the same idea of using *Resources*, *Quality* and *Cost* as done in the Cloud computing systems, in this regard, $Sys = (R, Q, C)$. Nevertheless, the heterogeneity of these systems presents a challenge, as already identified in [20]. As previously mentioned, systems that share a domain or that run similar applications are easier to relate by abstracting their common characteristics. But this is not the case when dealing with systems from different domains. For instance, developing

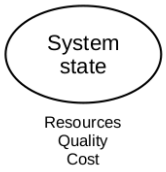

Fig. 2. The system state represented as a node, encoding its high-level variables: *Resources*, *Quality* and *Cost*.

the same abstraction to represent quality in a system that allows autonomous driving and another used for managing city waste is a challenge. Hence, our methodology will define the system state, its *Resources*, *Quality* and *Cost* depending on the application domain. Nevertheless, further research will focus on inter-domain generalization mechanisms.

The high-level definition of the system state limits its observability, therefore, a set of metrics, which are observable, have to be defined based on the application requirements to provide the link between computing-continuum resources and the system state. Simply put, it is not feasible to observe the low-level resources of the computing-continuum fabric to directly obtain the system state. In Figure 3 it can be seen how the low-level resources are aggregated and filtered into a set of metrics. Similarly, this set of metrics is then related to the system state. Again, depending on the application domain the system will require a higher dependency on network latency aspects or in an efficient use of the computational resources. It is worth noticing two details from Figure 3, first, the arrows keep knowledge of causality relations, which are important to understand the reasons of the system's behavior. Second, both the set of metrics and the relation of these metrics with the system state are dependent on the application requirements, whereas the relation between the "computing-continuum resources" and the set of metrics are not.

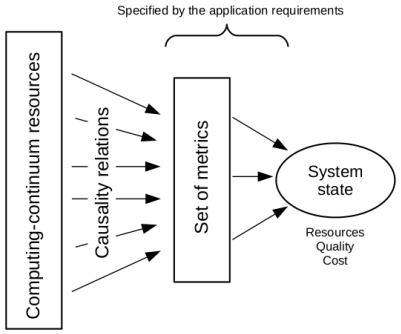

Fig. 3. The figure shows how the system state is influenced by a set of metrics, both the set of metrics and level of influencing mostly depends on the specified application requirements. Furthermore, the figure shows how causal relations for the metrics values can be connected with the "computing-continuum" resources, or in other words, its underlying infrastructure.

The methodology that it is being discussed in this article also requires to act in order to provide adaptive mechanisms to the system. Simply put, the system has to be able to change the configuration of its underlying infrastructure in

order to overcome any possible issue encountered. Again, providing the system state with capabilities to directly act on the computing-continuum resources is not feasible due to the large conceptual difference between a high-level representation of the system and the actual low-level adaptation on the underlying infrastructure. Hence, there is the need of create a set of actions to move from the higher-level to the lower-level, as seen in Figure 4. This can be a set of different configurations that allows the system running within its expected requirement. However, these configurations might not all be possible to be described at design phase, therefore, this requires to have the capacity to learn new configurations given applications constraints and the current state of the computing-continuum resources. In any case, this will be further expanded in subsection 4.1.4.

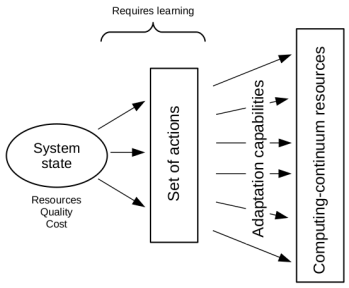

Fig. 4. The figure shows the relation of the system state with the set of actions. In this case, the set of actions require to be learnt as well as the precise relation of the state with them in order to develop a system with autonomous and adaptive capabilities. Finally it can be seen that the adaptation capabilities of the system influence the "computing-continuum" resources.

4.1.2 Markov Blanket

It is now possible to provide a wider view of the system representation, by merging together the previous ideas, as seen in Figure 5. This gives the opportunity to introduce a concept, the Markov Blanket (MB), that provides several key features to the managing methodology for emerging "computing continuum" systems. As defined in [25], the MB of a variable are those variables that provide enough information to infer its state. Formally, if the Markov Blanket of the variable V are variables Y and X, we can say that $P(V|X,Y,Z) = P(V|X,Y)$.

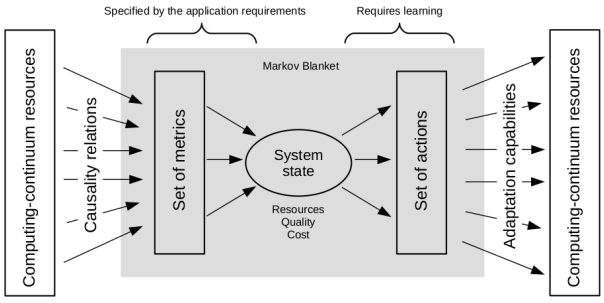

Fig. 5. This figure shows the complete perspective on the required representation of the system to apply the methodology depicted in this article. It is worth noticing the central gray space that sets the boundaries of the Markov Blanket that defines the system state.

The previous definition implies that without a direct observation of the variable V it is possible to infer its state given the observation of its MB variables. This a useful tool in order to infer the system state, given that as previously stated it is not possible to obtain direct observations. Additionally, the MB is used as a causality filter in our methodology, simply put, the granularity or detail of the observations to infer the system state can be chosen, given that the selected set of observations is meaningful to the system state. Hence, the set of metrics and actions can be selected in order to easily match the application requirements. Furthermore, it allows to create nested representations of the system, so besides providing a tool to choose granularity, it allows to focus into a precise application on the entire system to act on that part keeping the knowledge of the relations and using the same methodology for adaptation. Simply put, if the application is managing the mobility of autonomous cars on a city, a nested MB representation allows to focus on a smaller application, such as the visualization of the street cameras in order to solve any issue there, but keeping clear track of the relations with the other parts of the system thanks to the causality links encoded.

It is important to remark that the MB variables include both the metrics and the actions, as seen in the gray area of Figure 5, given that to understand the current state of the system the metrics that relate with the underlying infrastructure are required but also the current configuration of the system, provided by the state of the action variables. Usually, the MB is represented as a directed acyclic graph (DAG) which provides framework for using Bayesian inference, which initially can help to construct the graph to represent the system, and later to develop knowledge on causality reasons of an event and their propagation up or downstream. Formally, the Markov Blanket DAG consists of the parents, the children and the spouses of the central node.

Figure 6 shows a more detailed representation of a Markov Blanket for a "computing continuum" system. It can be seen how several metrics influence the system state, and how this influence action states, additionally, there can be direct links from metrics to actions and also from resources of the underlying infrastructure to actions. Hence, building the DAG representation is not trivial and requires a proper definition of the components and also on their links to keep track of causality effects between them. To that end, subsection 4.2 will take advantage of the DAG representation to detail learning techniques that can be used to build it.

4.1.3 Equilibrium

The causality filter provided by the Markov Blanket also provides a high-level perspective on the system scope in terms of the data, or information, that it can use and in the adaptation capabilities that it can have. This scope develops a separation between the system and the environment, which brings a new concept: the system's equilibrium. We can put in place some of the explained concepts to create a visual metaphor, picture that the system is a blanket on top of an underlying infrastructure where you can tie and stretch it. In this scenario having the blanket properly stretched and attached to the underlying infrastructure would mean having the system in equilibrium. Hence, once

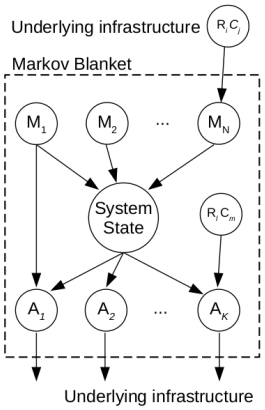

Fig. 6. The Markov Blanket for the system state is represented as directed acyclic graph (DAG). Each M represents en element from the set of metrics, similarly, each A represents an element from the set of actions. Also each R makes reference to a resource and C to a components of the underlying infrastructure of the system.

the underlying infrastructure change due to its dynamic and complex behavior, the blanket would become rippled or uneven, therefore an adaption capability would tie and stretch the blanket on another infrastructure resource. In this regard, we are using the concept of equilibrium in order to quantify the need for the system to adapt, simply put, if the system is not in equilibrium an adaptation is required, otherwise it can stay as it is. The management decisions over the system equilibrium aims at freeing the system from the thresholds dependencies, providing a much more flexible framework.

"Computing continuum" systems do not have this intrinsic equilibrium defined. However, this does not mean that it can not be described and leveraged. In this sense, the equilibrium of a "computing continuum" system can be defined as an operation mode defined by a specific infrastructure configuration. This implies that the equilibrium depends on the evolution of the metrics that define the system state and also on the specific configuration of the "computing-continuum" resources. Simply put, this depends on the Markov Blankets nodes. Therefore, if relations are created between the MB nodes during the graph construction, their derivatives will provide the meaningful information to talk about system evolution, and consequently, equilibrium, providing a framework that allows proactive behaviors over reactive ones. It is worth emphasizing that this research is framed to a probabilistic approach, in this regard the system derivatives will be probability distributions over a random variable variation.

The equilibrium and the derivatives of the system bring another concept to the surface, this is the temporal evolution of the system given that all the previous does not make sense if the system is static. Figure 7 shows, in its bottom, the temporal dimension on the system, the initial state of the configuration of the "computing-continuum" underlying infrastructure changes, which is sensed by the system metrics', which leads to a degradation of the equilibrium state of the application, therefore, a set of actions is activated in order to reconfigure the state of the underlying infrastructure in order to preserve the equilibrium. It is worth noticing that given the complexity of these systems, the equilibrium of the system state can be found at different configurations, which will lead to different equilibrium states or operation modes, here we assume that a different configuration of the underlying infrastructure will not assure that the operation mode of the system is maintained, only its requirements.

4.1.4 Adaptation

Triggered by an equilibrium disturbance, an adaptation process needs to be performed. This adaptation process (set of actions) aims at changing the system configuration with respect to the underlying infrastructure, to reach a new operation mode for the system and recover the equilibrium for the system state. However, this process faces two main challenges: first knowing the configuration capabilities of the system, so determining the complete set of actions, and second selecting the best strategy or set of actions to adapt.

The first challenge rises a crucial issue for these systems, given that the space of possible configurations for the system increases with its size and complexity, which is very large, and may be partially unknown. We foresee two approaches to deal with it, and both require a learning framework, given that is not feasible to encode all possible configurations. The first option would be to expose to the system low-level options for adaptation and given some constraints let the system learn how low-level adaptations are mapped into solving higher-level disturbances. This would allow the system to change the initial set of low-level adaptations to higher-level ones which could eventually lead to better decisions for the second challenge of selecting the best action. The second option is developing high-level adaptations from partial representations of the system using the nested capacity of the MB representation. This clearly reduces the space of possible configurations, which might allow to develop a significant list of possible high-level adaptations, however, this will never cover the entire space and might not be able to expose adequate solutions for the overall system. In any case, it is require to provide the system with learning capabilities to develop its configuration space or set of actions.

Furthermore, given the space of possible adaptations of the system, the system requires to chose the best possible option, which is the second challenge faced in adaptation. In general, finding the best action to recover the system's equilibrium given its complexity, the partially observable data and the stochastic nature of several underlying phenomena is already a challenge and it is usually solved with heuristics. Furthermore, given that systems are open and have many interconnected relations with the underlying infrastructure, finding a solution for this problem might also require a learning framework. Simply put, the selected actions can affect the system from different perspectives preventing the system state to reach its equilibrium, which can lead to the need of developing larger policies until the new equilibrium states are found. Figure 7 adds to the previous schema of the system the learning framework, where the chosen re-configuration for the system is taken as input, and following this iterative process learning can be achieved.

Interestingly, there is a principle, called Free Energy Principle (FEP) [26], from neuroscience, that explains the

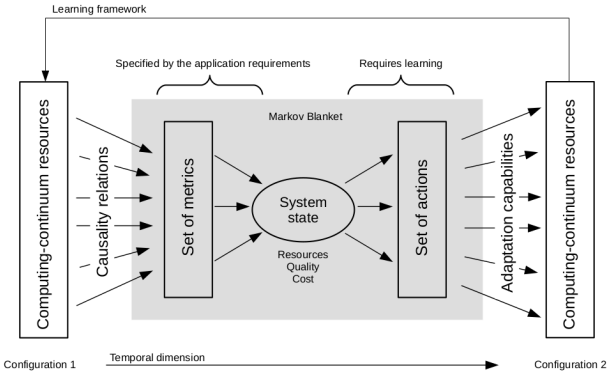

Fig. 7. This figure presents again a complete perspective on the system's representation. This figure emphasizes the temporal dimension of the system and how the configuration of the "computing-continuum" resources follows this temporal dimension. Additionally, it presents the structure of the learning framework for the adaptation capabilities of the system, which resembles to the typical framework of RL.

adaptive behaviour of the brain, and in general for any system that has adaptive capacities. However, this generalization is actually being disputed by the work of [27] which claims that the principle is not that general. In any case, there are several similarities with our approach that are worth mentioning.

The FEP already uses the Markov Blanket to develop a representation for the brain, furthermore it categorizes the Markov Blanket nodes in two: the sensing and the acting. In our representation, the sensor nodes are metrics, and the active nodes are these capabilities of the system to modify the underlying infrastructure. The FEP states that, as an adaptive system, the brain acts as if it was minimizing the free energy of the system, or similarly, maximizing the evidence lower bound (ELBO). In this regard, the free energy is the difference between an expected observation and the obtained observation. There are two main challenges that need to be addressed in order to determine if the FEP would also apply for the systems that we are considering in this work. First, it would be required to quantify the relations between the Markov Blanket nodes and the system state, which in the case of the metrics seems more attainable but when dealing with the actions this is still a very open issue. Second, it is required that our system is able to have a representation of the environment, hence it can have an expected observation given an action, this is commonly known as a generative model of the environment, which is also a relevant challenge that stills need to be tackled.

4.1.5 Illustrative example

The following presents an illustrative metaphor in order to better understand all the previous methodology. Imagine that the application of the "computing-continuum" is simply about transporting containers through the ocean, hence, the "computing-continuum" system is a boat that carries them. For the sake of the metaphor, we will assume that the containers can't be fixed to the boat, hence if the boat tilts too much and it looses its "equilibrium" then the containers can fall and be lost in the ocean.

In any case, there are two main requirements for the boat in order to properly handle the transportation. The first is reaching the destination harbor, and the second is not to loose any container. The first requirement is assumed that will be achieved given that the boat has sophisticated technologies to be guided, such as GPS or a compass, and the destination harbor is fixed within the navigation frame. However, the second requirement is more tricky, given that the stability of the boat depends on the weather and sea conditions. Going back for a moment to the computing continuum, the first requirement would be for an application to work without the need of any "computing-continuum" underlying infrastructure, such as first Internet applications. However, the second takes into account that the environment of the application, is dynamic and complex.

Therefore, to assess the stability of the containers the boat has several inclinometers, sailors that observe the surrounding state of the ocean waves, and a report on the weather condition of its current area for the next couple of hours. However, the causal relations for these observations can be more deeper, hence, one could think that to better understand or predict the tilt of the boat it is required to observe the ocean and atmosphere characteristics, such as the temperature of the undersea currents, or the position and trajectory of the closest squall. This could actually improve some prediction models, but it drastically increases the complexity on the system. A similar idea can be cast towards the possible configurations of the boat in order to overcome a wind gust or a large wave, if the sailors are experienced enough, just adapting the sail position can be enough to manage that, but if they are not, they could try more options, which in some situation can lead to worse scenarios. This idea mimics the benefits brought by the Markov Blanket in terms of filtering causality for the system state, and to have focus on the system state by observing the metrics and the system's configuration.

This metaphor also provides a clear view on the idea of the system equilibrium, if the boat tilts when waves come and go then it can lose containers. Therefore, the equilibrium is required to dynamically change the system configuration to maintain the application performing as expected.

In this subsection we have developed our vision of the required methodology to manage "computing-continuum" systems. In this regard we have identified several aspects of the methodology that will require to leverage learning techniques. These aspects can be separated in two classes, the design phase, where the methodology is adapted to the specific "computing-continuum" system, and the run-time phase, where the methodology is used to manage the system. From this perspective we can identify that the design phase will require learning algorithms to optimize the DAG representation of the system, in terms of the nodes that are represented (set of metrics and set of actions) and their relations. For the run-time phase we will need algorithms that can update the system representation based on the current situation of the system but more importantly algorithms that learn which adaptive mechanism to select and how to perform the adaptation.

4.2 Learning

This section details the learning on system representation during the design phase and during the run-time phase.

TABLE 1
summary of Design-phase learning

Algorithms	Proximity	Stability	Reactive	Handling Missing Values	Prediction Accuracy	Complexity
MF	✔	✗	✗	✗	–	Low
RWL	✔	✗	✔	✔	High	Reasonable
SRL	✔	✗	✔	✗	Low	High
GNN	✔	✗	✔	✗	High	High
RL	✗	✔	✔	✔	Medium	High
SDNE	✔	✗	✗	✗	Low	High
GDL	✔	✔	✔	✔	High	High

In the design phase, we discuss various graph learning algorithms used to refine the system and MB learning. In the run-time phase, the FEP performs the predictions and action selections through active inference. These two learning approaches are operated iteratively, as shown in Fig. 7.

4.2.1 Design phase learning

The DAG is built up with conditional dependencies between the states from a given set of available data (even though missing values in the data) [28]. Explaining these data using DAG is considered as an NP-hard problem [29]. Still, several software tools (Java-based) and libraries (for MATLAB, Weka, R, C++, Python, etc.) are available to build initial DAG [30], [31]. Once the DAG is constructed, the learning can take place to keep track of causality relations (strength and weakness of a relation), identify redundant relations and states, optimal MB discovery, etc. There are several graph learning algorithms in the literature to extract the knowledge from the graph data. These algorithms are used to estimate the proximity, structural information, failure state predictions, etc. Some of these graph learning algorithms are also useful to extract the knowledge from the DAGs. The majority of these learning algorithms use the DAG as an adjacency matrix to simplify the problem. The summary of graph learning algorithms for DAG is presented in Table. 1. Here, the proximity indicates the strength of the causality relation among the two states. The stability in the table refers the small change in the input of data does not affect the results or outcome drastically. A reactive means the algorithm responds and acts to the unprotected occurrence of events during the learning process.

A *matrix factorization* (MF) or decomposition mechanism is used to estimate the strength of the relations (proximity) on a DAG. In general, the MF reduces a matrix into constituent parts to understand the complex relations easier and faster [32]. It also helps to identify the contested edges (the removal of a link improves the prediction accuracy among the states) in the DAG [33]. Nonnegative MF further helps to identify the irrelevant states or relations in the DAG. It is further improved by combining with manifold learning strategy to exploit the nonlinear relations on the DAG [34]. Besides these benefits, there are a few limitations, i.e., sensitivity to outliers and noises. So, the MF is unstable until it removes the noises and outliers. The computing continuum systems' are complex because of several characteristics discussed in subsection 3.2. However, MF does not result from accurate results for these systems.

Random walk learning (RWL) can identify the similar property states in a short hop distance over the DAG so that we can quickly identify and remove the redundant states. Unlike MF, the random walk can handle the missing information and noise data. But, the learning accuracy of a DAG is more if data is available. The dynamic characteristics of the computing continuum, such as the states or relations, may evolve over time and require more effort to learn because new relations appear while old relations may not be valid for a long time. The random walk can perform the learning on such dynamic systems through continuous learning [35]. Besides these advantages, there are certain limitations of random walk learning. It can predict the proximity only if the relationship existed between any two states in the system. It can create uncertainty in the relations for the small systems because of random strategies and less data availability. RWL is computationally high. The RWL can benefit from computing continuum system representations in predicting the strength of causality relations between the states. It also adopts the dynamic changes in the computing continuum representation and learns accordingly.

Statistical Relational Learning (SRL) is used for link predictions (properties of relations) on a complex and uncertain DAG. There are several categories of SRL techniques such as Probabilistic Relational Models (PRM), Markov Logic Networks (MLN), Bayesian Logic Programming (BLP), and Relational Dependency Network (RDN) [36]. These techniques efficiently predict the strength of relations among the states. PRM, BLP, and MLNs suffer from extracting the inference for the small systems or the large systems with unlabeled information. The SRL fails to forecast accurate predictions in the case of the unlabeled instances in the system. SLR causes too expensive computations if missing labels in the system. The SLR may not fit the computing continuum unless the label data is provided without missing information. Due to the scalability and dynamic nature of the computing continuum, it is strenuous to provide such accurate information, so SRL may not fit the computing continuum.

Graph neural networks (GNN) is a supervised learning strategy and uses a neural network framework to process the DAG. The GNN can be operated learning on three general types such as a system, a state, or a relation [37]. The system-level and state-level learning strategies can result in classification (categorize the states of representations according to their characteristics), predict the possible relations among the states, and group the similar property nodes in the system for simplifying the learning strategies. The relation-level learning strategies can predict the possible relations among the states and also classify the relations depends on their characteristics. Combining the GNN and SRL techniques can help us to estimate the relational density between the states on representations [38]. There are certain limitations of GNN such as (i) challenging to represent the system compatible with neural networks, (ii) being computationally expensive, (iii) Customization of the GNN model (deciding the number of GNN layers, aggregation function, style of message passing, etc.). On computing continuum representation, the GNN is useful for classifying the states, predict strength of the relations, and grouping the similar property nodes or relations.

Reinforcement learning (RL) can be used to learn and extract knowledge from a DAG by considering accuracy,

diversity, and efficiency in the reward function [39]. RL and GNN are combined together to extract the behavior of a DAG [40]. This combination is also used to identify the cascades of failures states in the DAG [41]. These techniques are suitable for small systems because they are computational hungry mechanisms. These methods are also fit for dynamic environments such as computing continuum because of adaptability.

Structural deep network embedding (SDNE) exploit the first and second-order proximity together into the learning process to achieve the structure-preserving, and sparsity [42]. The first-order proximity can extract the pairwise similarity between any two states (local network). In contrast, the second-order proximity identifies the missing legitimate links throughout the system. Therefore, the second-order proximity can identify the redundant states in DAG. The characteristics of the local and global systems learn using these two proximities. It can remember the initial structure of the DAG and easily reconstruct the original one in case of any failures during the learning. Along with this, the SDNE results at most performance in terms of classification and link predictions. On the other hand, it is essential to identify the balance point among the first and second-order proximities to achieve optimal results. However, it increases the complexity due to unnecessary computations. The SDNE is helpful to predict the strength of causality relations on computing continuum representations. Since SDNE is high computational, it is possible to minimize by grouping similar states or relations using GNN.

Geometric deep learning (GDL) efficiently classifies the similar states on a system and parse the heterogeneous systems efficiently. In contrast, the traditional GNN, deep learning, Convolutions Neural networks (CNN) algorithms are inaccurate. The primary reason is that the GNN, deep learning, CNN are worked based on convolutions, and they can work efficiently on Euclidean data [43]. In contrast, computing continuum data or representations are non-euclidean. It means there are different states with a variety of neighbors or their connectivity. So, it isn't easy to apply the convolution because of non-euclidean data. However, the GDL does not resolve the curse of dimensionality issue. GDL does not require additional abstractions to perform the learning on computing continuum representations. It is dynamic, accurate, and low computational for learning complex computing continuum systems.

Bayesian network structure learning (BNSL) extracts the correlations among the random variables from the data. The BNSL is performed in three learning approaches, including constrained-based, score-based, and hybrid (which combines both score-based and constrained-based techniques). The constrained-based approach learns structure dependencies from the data using conditional independent tests. The score-based approach minimize/maximize the scores as objective functions [44]. These three approaches are primarily useful to perform learning on a set of relations between the states and quantify the strength of causality relations [45].

Sun et al. [46] uses a Particle swarm optimization algorithm for BNSL to minimize the computational complexity through the derivation of the optimal objective function. A constraint-based parallel BNSL approach is introduced in [47] to perform the parallel conditional independence test to minimize the time-consuming steps during the learning process. He et al. [48] derived efficient BNSL for large DAGs using multi-granularity information. The DAG is partitioned using the hierarchical agglomerative clustering algorithm and each sub-partition learn independently and construct a refined DAG. Qi et al. [49] propose a BNSL algorithm based on sharing mutual information between the two states. In this approach, the mutual information decides either the edge removal (when a piece of mutual information is weaker) or condition set generation (when a degree of mutual information is weaker). Vogogias et al. [50] introduce a score-based BNSL tool (named BayesPiles) to visualize the consensus representation using a heuristic search algorithm. Tsamardinos et al. [51] proposed a constraint-based Max-Min Hill-Climbing algorithm for BNSL. This algorithm combines search-and-score techniques with local learning to improve the efficiency of the representations. A recursive anatomy identification (RAI) algorithm is introduced for BNSL in [52] to learn the edge directions, conditional Independence, and sub-systems recursively and efficiently. The RAI algorithm provides the best classification accuracy, correct representations with low computational complexity.

During the design-phase learning, the algorithms discussed above identify and remove the redundant states and relations from the system representations. It also reconstructs the system representation according to the strength of the causality relations. It is worth detailing here that the MB concept can also be used to optimize a DAG. In this regard, we need to separate the conceptual perspective of having the system's representation using the state variables as a central node for the MB, from using the MB concept in order to optimize the final representation of the system. The latter allows to center a MB at each node of the DAG in order to verify the dependency of the current relations, which is a very popular technique for feature selection in machine learning.

Markov Blanket Learning (MBL) follows a supervised learning strategy to extract the DAG that characterizes the target state. Unlike other supervised learning algorithms (For example, Neural networks, Regression, Support vector machine, Decision tree, etc.), the MBL returns a generative model (return more robust models even though some missing values and redundant features). MBL can perform either constraint-based or score-based approaches, which are similar to BNSL. The constraint-based approach is further divided into topology-based (tackle the data efficiency) and non-topology-based (greedily test the states' independence). The time complexity of topology-based learning is reasonable, over the non-topology and score based learning approaches. The primary goal of these learning strategies is to reduce the loss in the accuracy in MB discovery with a reasonable complexity [53]. Few improvements made on these learning algorithms to overcome this challenge. A recursive MBL algorithm is introduced in [54] for learning the BN and removing the states in which the statistical dependencies do not affect the other states of the system. The primary goal of the MBL is feature selection or discovering the best set of MBs) [55]–[57].

The MBL produces a robust system by reducing the number of states, and non-linear relations [58]. By using the independence relation between the states, MBL is discov-

ered for each node and then connected the MB consistently to form an updated global system [55]. Learning the MB with low computations and high efficiency is crucial. In this context, several MBL algorithms are introduced in the literature. Ling et al. [59] proposed an efficient approach to discover the MB using local structure learning. Here, the MB are identified based on the currently selected parents and children (PC). This approach results in more true positive states (for the selected MBs) in a large DAG with minimal computations. A local structure learning algorithm is introduced for efficient MB discovery in [60] to distinguish the parents from children based on the edge directions from the DAG. A minimum message length-based MBL algorithm is introduced in [61] for large-scale DAG with perfect and imperfect data. The perfect DAG means the detailed information about the states and their relations and is maintained using a conditional probability table. In this approach, the MBL is smooth for perfect DAG. The imperfect DAG uses a Naive Bayes to assume the independence between the MB and the remaining states. A topology-based MB discovery algorithm is introduced by Gao et al. [62] using a simultaneous MB algorithm, in which the false PC and coexistence property of spouses states are identified and removed simultaneously to minimize the computational time. A selection via group alpha-investing (SGAI) technique is introduced for efficient MB selection from a set of multiple MBs using representation sets [63]. The SGAI algorithm avoids unnecessary computations for parameters regularization. So, the overall complexity of the MB discovery will minimize through this approach.

There are certain benefits of using MBL on computing continuum. MBL-based feature selection help us to identify the best resource (state) for service placements [64] with rapid actions through quick decisions. The MBL can also be beneficial to decide a set of metrics that influence a resource's workload among all the metrics. Identifying fewer metrics to determine the influence on a resource will decrease the computation burdens.

4.2.2 Run-time phase learning

As we discussed in subsection 4.1.4, the states of a system representation are differentiated by MB as sensory and active states. In a MB, the internal and external states are conditionally independent of each other. These two states are influenced via sensory and active states [65]. The internal and active states directly affect the structural integrity of a MB through active inferences. The active inferences are helpful to minimize the uncertainties of the sensory states in the MB and derive minimal free energy for the internal states. The FEP mainly works based on active inference and it carries out learning and perception on the system [66]. The active inference understand the behaviour of the system depending on the generative model and it can able to predict the sensory states. The active inference is helpful to obliterate the prediction errors by updating an internal system that generates predictions through a perceptual interface and perceptual learning.

The FEP majorly performs predictions of future states and through active inference is able to improve them. Furthermore, FEP through active inference is able to learn about the best action to select. Initially, predictive coding consistently updates the system representation based on the predicted sensory state information. The FEP minimizes the prediction errors by comparing the actual sensory state information with the predicted data. The predictions generated by the FEP are more accurate than the traditional machine learning algorithms. FEP is very powerful in minimizing the prediction errors through errors backpropagation [67]. Actions are the source of the system control. These actions are selected and controlled by the FEP through variational message passing based on the sensory state predictions. The theory of active inference especially inspires the proper selection of an action. The active inference is also useful to generate a deep generative model for adoptive actions through exploration. The critical limitation of the active inference is the scalability, and it can be addressed using the deep neural networks or deep RL [68].

The traditional machine learning strategies and FEP are combined in the literature to achieve the better performance of the systems. For example, an artificial neural network is incorporated with FEP to enhance the learning rate in terms of better action selection, and control [69]. In [70], the FEP is used as Active inference for the RL to speed up the learning process to reach the goal state quickly. FEP is used for reinforced Imitation learning to minimize the explorations through learning and perception. This discussion concludes that combining the traditional ML with FEP can extend the learning paradigm. This strategy reduces the cost and produces high performance by reaching the goal state quickly.

5 USE CASE INTERPRETATION

In this section we present a use case in order to exemplify how we envision our methodology.

5.1 Application description

We set the application in the traffic control area. It consists of checking if car drivers are using their phone with their hands while driving. To do so, several cameras are installed at the roads and the videos recorded are able to detect driver activity through AI inference[1].

The system is composed by a set of sensors distributed through the roadside, the basic set would require cameras to record driver activity, radars to get car speed, ground sensors to count the number of cars in the road and a light system to allow recordings without daylight. Additionally, the sensors require to connect to small computational units to pre-process data, and in the case of the cameras, AI inference boards are expected to be within their proximity. Additionally, the application will have a larger server to process the data as well as storage capacity. Finally, the application is also ready to offload some of the inference or processing needs to a public cloud. Figure 8 shows a simplified schema of the application architecture.

5.2 SLOs as application requirements

Service Level Objectives (SLOs) are a tool for Cloud providers to map requirements into measurable metrics of

1. www.newscientist.com/article/2300329-australias-ai-cameras-catch-over-270000-drivers-using-their-phones/amp/

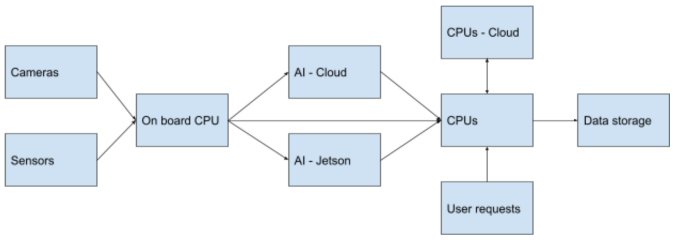

Fig. 8. Schema of the application's architecture

the system. In our approach, we go one step further and we focus on high-level SLOs [22] in order to encode service oriented requirements and to being able to relate these SLOs with the three highest-level variables of the system: *Resources*, *Quality* and *Cost*. However, this advantage comes with the cost to map these into measurable parts of the system, which is not always straightforward. For this use case we have select the following SLOs:

- **SLO 1: Percentage of drivers recognized.**

This SLO is strongly linked with the *Quality* of the system. It would be computed by taking the total number of drivers recognized by the system and dividing it between the total number of cars.
To get the total number of cars we would need the output of sensors, such as the ground sensors, and to obtain the number of drivers recognized we would require the output of the AI-based inference system. However, in order to have a complete picture of what is the status of the application regarding this SLO, we will also consider the number of cameras in place, to account for obstructions between cars, the video quality which involves the image resolution but also if its daylight or if it rains, finally the inference model used will also be taken into account. All this information does not properly provide the SLO value, but is expected to be able to explain the current situation of the SLO, it can be understood as a Markov Blanket around the SLO.

- **SLO 2: Percentage of drivers recognized faster than twice the speed limit.**

This SLO is very related with the previous but is easily related with the business strategy of the application, given that drivers at the phone, which are at twice the speed limit, are more severely fined. The only difference that we account for this SLO, with respect to the previous, is the need to incorporate data from radars on the overall equation.

- **SLO 3: System up-time.**

This SLO is related with the *Resources* that the application is using. It would be assessed by considering the total up time of the application with respect to the total time elapsed. However, we want to account for the causes and the relations between the system components that allow this level of availability. Hence, with this SLO we will also take into account other ways to measure the system's availability, such as having a periodical ping to the computing units or checking the continuous stream of data from the sensor. More specifically, we imagine end-to-end tests that could also perform an overall check of the entire system, so that we can check that it is up and well-functioning.

- **SLO 4: Expected remaining energy above certain value.**

IoT and Edge devices can have energy constraints if they are not plugged to a main energy source. Therefore this SLO will ensure that this devices do not run out of energy. Hence, this SLO relates both *Resources* and *Cost* of the system. To measure it, we would need to track the operation time of energy-constrained devices and their charging times. This information already provides everything we would consider to have a broader picture of the system.

- **SLO 5: Percentage of inferences at Public cloud.**

The use of a Public cloud can be required at some specific moments due to periods of over-request on the application from both the users side and the sensors side. However, this type of offloading highly increases the cost of the application due to the fees applied by the Public cloud providers. Therefore this SLO is used to have control over unexpected *Cost*s. It would be computed by keeping track of the offloadings at the public cloud.

- **SLO 6: Privacy level.**

A big challenge is to provide qualitative measure for privacy standards of an application, however, it is clear to us that this type of requirement is a must for "computing-continuum" applications. For this use case, we are considering a metric called *privacy concern* that will categorized the privacy of the system with respect of the system topology and the use of public cloud, in which extra privacy measurements might need to be taken into account. This metric will be balanced with the privacy measures in place, which will provide the final privacy level of the system. Therefore, by keeping track of the offloaded instances as well as the current privacy concern of the system and the implemented privacy measures, it would be possible to provide the proposed SLO.

5.3 Developing the DAG

From the previous description of the required SLOs, we can draft a DAG that express the relation between low-level metrics, which can be understood as a description of the "computing continuum" resources, the set of high-level SLOs and the system state variables. This DAG expresses the first part of our new representation, see Figure 3. Additionally, we expect to find a mapping between this representation and a causal representation of the system. The second part of the system's representation, see Figure 4, relates the high-level system variables and, possibly, some SLOs with the system's adaptive means, however, we will not yet unfold this part in this article.
A first approximation to the DAG can be seen in Figure 9. In order to build this first approximation, we have gone through the SLO needs and we have identified the measurable resources that could be taken into account. Nevertheless, it is clear that this might not be the final graph, as we might be not taking into account relevant resources or some of the depicted on the graph are not really influencing the

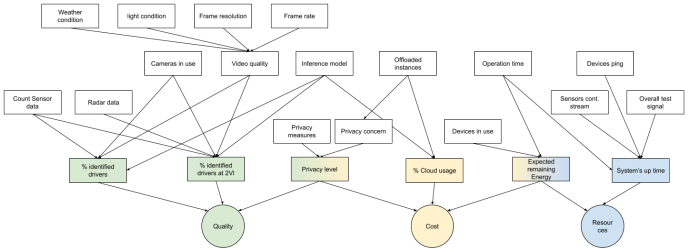

Fig. 9. DAG of the system: from the measurable "computing continuum" resources to the high-level system variables.

SLO. Further, it can even change due to new constraints or requirements. Hence, from this point it is required to use the different techniques depicted in Section 4.2 in order to adjust the representation to the final configuration of the system.

It is important to highlight that this DAG is also a germinal causal graph for the application, this causal graph is expected to encode the capability for the system to trace back the causes of its current status, leading to the capacity of performing precise actions to solve any encountered issue. However, the causal graph will also require further refinement in order to ensure that causality relations are well defined. We expect both DAG (the system's Markov Blanket and the causal graph) to deviate and differentiate as they precisely encode their expected features. However, building both from the system's SLOs ensures finding common points in order to move from the abstraction of causality analysis to the actual measurements and components of the system.

6 Conclusion

This article highlights the need of developing new methodologies to manage "computing continuum" systems, this includes developing a new representation of theses systems and a framework to develop tools and mechanisms for them. The elasticity paradigm developed for managing cloud systems falls short when dealing with "computing continuum" systems. In short, it has been seen that the Cartesian space it is not able to properly grasp the characteristics of the underlying infrastructure on which these systems are based, and it does not provide active mechanisms to solve the perturbation that they can suffer. Nevertheless, the system's space composed by *Resources*, *Quality* and *Cost* is kept to provide a high-level representation.

The methodology develops a new representation for these systems that is based on the Markov Blanket concept. This provides manifold advantages to deal with complex systems: i) provides a causality filter to control the scope of the system representation; ii) allows to develop nested representations to focus on specific issues; iii) develops a formal separation between the system and the environment providing space for the development of cooperative interfaces between different systems; vi) encodes causality relations within the system components to provide knowledge on the system's evolution.

The managing framework is based on the concept of equilibrium, leaving behind the use of thresholds. In this regard, the equilibrium aims at alerting the system in advance so that the adaptive mechanisms performed are more efficient. Additionally, it also encodes the system derivatives, which allows to adjust the adaptive mechanism to the precise needs of the system.

To sum up, this new approach focuses on the complexity inherent to these systems due to their underlying infrastructure. However, this does not solve the problem of their complexity, just set the tools to manage them. Therefore, this article also presents a survey on learning methods required to completely develop the methodology, ranging from methods to build and maintain an optimized representation of the system, until mechanisms to develop the self-adaptive capacities of the "computing continuum" systems.

References

[1] G. Parisi, "Complex systems: a physicist's viewpoint," *Physica A: Statistical Mechanics and its Applications*, vol. 263, no. 1-4, pp. 557–564, feb 1999.

[2] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing Cache Data Integrity in the Edge Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2021.

[3] X. Gao, X. Huang, Y. Tang, Z. Shao, and Y. Yang, "History-Aware Online Cache Placement in Fog-Assisted IoT Systems: An Integration of Learning and Control," *IEEE Internet Things J.*, p. 1, 2021.

[4] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and Revised Content-Centric Networking-Based Service Deployment and Discovery Platform in Mobile Edge Computing for IoT Devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4162–4175, 2019.

[5] D. Kimovski, R. Matha, J. Hammer, N. Mehran, H. Hellwagner, and R. Prodan, "Cloud, Fog or Edge: Where to Compute?" *IEEE Internet Computing*, p. 1, 2021.

[6] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An Energy-Efficient Dynamic Task Offloading Algorithm for Blockchain-Enabled IoT-Edge-Cloud Orchestrated Computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, 2021.

[7] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Collaborative Edge Computing and Caching With Deep Reinforcement Learning Decision Agents," *IEEE Access*, vol. 8, pp. 120 604–120 612, 2020.

[8] H. Li, K. Ota, and M. Dong, "Deep Reinforcement Scheduling for Mobile Crowdsensing in Fog Computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, apr 2019.

[9] S. Nastic, T. Pusztai, A. Morichetta, V. Casamayor Pujol, S. Dustdar, D. Vij, 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)*, 2021.

[10] R. Zhu, S. Li, P. Wang, M. Xu, and S. Yu, "Energy-efficient Deep Reinforced Traffic Grooming in Elastic Optical Networks for Cloud-Fog Computing," *IEEE Internet Things J.*, p. 1, 2021.

[11] B. Alturki, S. Reiff-Marganiec, C. Perera, and S. De, "Exploring the Effectiveness of Service Decomposition in Fog Computing Architecture for the Internet of Things," *IEEE Transactions on Sustainable Computing*, p. 1, 2019.

[12] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-Efficient Fair Cooperation Fog Computing in Mobile Edge Networks for Smart City," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7543–7554, 2019.

[13] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[14] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, "Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures," *Computer Networks*, vol. 179, p. 107348, oct 2020.

[15] E. Hernández-Nieves, G. Hernández, A.-B. Gil-González, S. Rodríguez-González, and J. M. Corchado, "Fog computing architecture for personalized recommendation of banking products," *Expert Systems with Applications*, vol. 140, p. 112900, feb 2020.

[16] D. C. Klonoff, "Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things," *Journal of diabetes science and technology*, vol. 11, no. 4, pp. 647–652, 2017.

[17] S. Dustdar, Y. Guo, B. Satzger, and H. L. Truong, "Principles of elastic processes," *IEEE Internet Computing*, vol. 15, no. 5, pp. 66–71, sep 2011.

[18] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "Multi-level Elasticity Control of Cloud Services," in *Service-Oriented Computing*, vol. 8274 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 429–436.

[19] P. Hoenisch, D. Schuller, S. Schulte, C. Hochreiner, and S. Dustdar, "Optimization of Complex Elastic Processes," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 700–713, sep 2016.

[20] H. L. Truong, S. Dustdar, and F. Leymann, "Towards the Realization of Multi-dimensional Elasticity for Distributed Cloud Systems," *Procedia Computer Science*, vol. 97, pp. 14–23, jan 2016.

[21] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become," *Communications of the ACM*, vol. 64, no. 5, pp. 76–84, may 2021.

[22] 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.

[23] T. Pusztai, S. Nastic, A. Morichetta, V. Casamayor Pujol, S. Dustdar, 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)*, 2021.

[24] ——, "SLO Script: A Novel Language for Implementing Complex Cloud-Native Elasticity-Driven SLOs," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021.

[25] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[26] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of Physiology Paris*, vol. 100, pp. 70–87, 2006.

[27] V. Raja, D. Valluri, E. Baggs, A. Chemero, and M. L. Anderson, "The Markov blanket trick: On the scope of the free energy principle and active inference," *Physics of Life Reviews*, sep 2021.

[28] X.-W. Chen, G. Anantha, and X. Lin, "Improving bayesian network structure learning with mutual information-based node ordering in the k2 algorithm," *EEE Trans. Knowl. Data. Eng.*, vol. 20, no. 5, pp. 628–640, 2008.

[29] M. Bartlett and J. Cussens, "Integer linear programming for the bayesian network structure learning problem," *Artificial Intelligence*, vol. 244, pp. 258–271, 2017.

[30] M. Scanagatta, A. Salmerón, and F. Stella, "A survey on bayesian network structure learning from data," *Progress in Artificial Intelligence*, vol. 8, no. 4, pp. 425–439, 2019.

[31] Z. Ling, K. Yu, Y. Zhang, L. Liu, and J. Li, "Causal learner: A toolbox for causal structure and markov blanket learning," *arXiv preprint arXiv:2103.06544*, 2021.

[32] J. Li, C. Hua, Y. Tang, and X. Guan, "A fast training algorithm for extreme learning machine based on matrix decomposition," *neurocomputing*, vol. 173, pp. 1951–1958, 2016.

[33] J. Strahl, J. Peltonen, H. Mamitsuka, and S. Kaski, "Scalable probabilistic matrix factorization with graph-based priors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5851–5858.

[34] C. Peng, Z. Kang, Y. Hu, J. Cheng, and Q. Cheng, "Nonnegative matrix factorization with integrated graph and feature learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 3, pp. 1–29, 2017.

[35] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Dynamic network embeddings: From random walks to temporal random walks," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1085–1092.

[36] A. Popescul and L. H. Ungar, "Statistical relational learning for link prediction," in *IJCAI workshop on learning statistical models from relational data*, vol. 2003. Citeseer, 2003.

[37] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[38] D. S. Dhami, S. Yan, and S. Natarajan, "Bridging graph neural networks and statistical relational learning: Relational one-class gcn," *arXiv preprint arXiv:2102.07007*, 2021.

[39] S. Madjiheurem and L. Toni, "Representation learning on graphs: A reinforcement learning application," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 3391–3399.

[40] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. C. Pereira, and M. Pavone, "Graph neural network reinforcement learning for autonomous mobility-on-demand systems," *arXiv preprint arXiv:2104.11434*, 2021.

[41] E. Meirom, H. Maron, S. Mannor, and G. Chechik, "Controlling graph dynamics with reinforcement learning and graph neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7565–7577.

[42] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conf. on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[43] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[44] M. Scutari, C. E. Graafland, and J. M. Gutiérrez, "Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms," *Int J Approx Reason.*, vol. 115, pp. 235–253, 2019.

[45] S. Lee and S. B. Kim, "Parallel simulated annealing with a greedy algorithm for bayesian network structure learning," *EEE Trans. Knowl. Data. Eng.*, vol. 32, no. 6, pp. 1157–1166, 2019.

[46] B. Sun, Y. Zhou, J. Wang, and W. Zhang, "A new pc-pso algorithm for bayesian network structure learning with structure priors," *Expert Systems with Applications*, p. 115237, 2021.

[47] A. L. Madsen, F. Jensen, A. Salmerón, H. Langseth, and T. D. Nielsen, "A parallel algorithm for bayesian network structure learning from large data sets," *Knowledge-Based Systems*, vol. 117, pp. 46–55, 2017.

[48] C. He, H. Yu, S. Gu, and W. Zhang, "A multi-granularity information-based method for learning high-dimensional bayesian network structures," *Cogn. Comput.*, pp. 1–13, 2021.

[49] X. Qi, X. Fan, Y. Gao, and Y. Liu, "Learning bayesian network structures using weakest mutual-information-first strategy," *Int J Approx Reason.*, vol. 114, pp. 84–98, 2019.

[50] A. Vogogias, J. Kennedy, D. Archambault, B. Bach, V. A. Smith, and H. Currant, "Bayespiles: Visualisation support for bayesian network structure learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 1, pp. 1–23, 2018.

[51] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[52] R. Yehezkel and B. Lerner, "Bayesian network structure learning by recursive autonomy identification." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[53] T. Gao and Q. Ji, "Efficient score-based markov blanket discovery," *Int J Approx Reason.*, vol. 80, pp. 277–293, 2017.

[54] E. Mokhtarian, S. Akbari, A. Ghassami, and N. Kiyavash, "A recursive markov blanket-based approach to causal structure learning," *arXiv preprint arXiv:2010.04992*, 2020.

[55] S. Fu and M. C. Desmarais, "Fast markov blanket discovery algorithm via local learning within single pass," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2008, pp. 96–107.

[56] Z. Ling, K. Yu, H. Wang, L. Liu, W. Ding, and X. Wu, "Bamb: A balanced markov blanket discovery approach to feature selection," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 5, pp. 1–25, 2019.

[57] X. Yang, Y. Wang, Y. Ou, and Y. Tong, "Three-fast-inter incremental association markov blanket learning algorithm," *Pattern Recognition Letters*, vol. 122, pp. 73–78, 2019.

[58] J.-P. Pellet and A. Elisseeff, "Using markov blankets for causal structure learning." *Journal of Machine Learning Research*, vol. 9, no. 7, 2008.

[59] Z. Ling, K. Yu, H. Wang, L. Li, and X. Wu, "Using feature selection for local causal structure learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 4, pp. 530–540, 2021.

[60] S. Yang, H. Wang, K. Yu, F. Cao, and X. Wu, "Towards efficient local causal structure learning," *IEEE Trans. on Big Data*, pp. 1–1, 2021.

[61] Y. Li, K. B. Korb, and L. Allison, "Markov blanket discovery using minimum message length," *arXiv preprint arXiv:2107.08140*, 2021.

[62] T. Gao and Q. Ji, "Efficient markov blanket discovery and its application," *IEEE Trans. Cybern.*, vol. 47, pp. 1169–1179, 2017.

[63] K. Yu, X. Wu, W. Ding, Y. Mu, and H. Wang, "Markov blanket feature selection using representative sets," *IEEE Trans Neural Netw Learn Syst .*, vol. 28, no. 11, pp. 2775–2788, 2017.
[64] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Adaptive management of volatile edge systems at runtime with satisfiability," *ACM Trans. Internet Technol.*, vol. 22, no. 1, Sep. 2021.
[65] M. Kirchhoff, T. Parr, E. Palacios, K. Friston, and J. Kiverstein, "The markov blankets of life: autonomy, active inference and the free energy principle," *Journal of The royal society interface*, vol. 15, no. 138, p. 20170792, 2018.
[66] K. J. Friston, L. Da Costa, and T. Parr, "Some interesting observations on the free energy principle," *Entropy*, vol. 23, p. 1076, 2021.
[67] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.
[68] B. Millidge, "Applications of the free energy principle to machine learning and neuroscience," *CoRR*, vol. abs/2107.00140, 2021.
[69] M. W. Cho, "Simulations in a spiking neural network model based on the free energy principle," *Journal of the Korean Physical Society*, vol. 75, no. 3, pp. 261–270, 2019.
[70] N. Sajid, P. J. Ball, T. Parr, and K. J. Friston, "Active inference: demystified and compared," *Neural Computation*, vol. 33, no. 3, pp. 674–712, 2021.

o

Schahram Dustdar (F'16) is Full Professor of computer science heading the Research Division of Distributed Systems at the TU Wien, Austria. He is founding Co-Editor-in-Chief of the new ACM Transactions on Internet of Things (ACM TIoT) as well as Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. Dustdar is Recipient of the ACM Distinguished Scientist Award (2009), the ACM Distinguished Speaker ward (2021), the IBM Faculty Award (2012), an Elected Member of the Academia Europaea: The Academy of Europe, where he is Chairman of the Informatics Section, as well as an IEEE Fellow. In 2021 Dustdar was elected Fellow and President for the Asia-Pacific Artificial Intelligence Association (AAIA).

Víctor Casamayor Pujol is a project assistant in the Distributed Systems Group at TU Wien. In 2020 he obtained his PhD in ICT by Universitat Pompeu Fabra in Barcelona, Spain. He has a master in Intelligent Interactive Systems (MIIS) by UPF in Barcelona, Spain and a specialized master in space systems engineering (TAS-Astro) by ISAE in Toulouse, France. He has also worked in space propulsion at the CNES in Paris, France. His current research interest relates robotic applications and the methodologies for computing continuum systems.

Praveen Kumar Donta (M'17) is a University Assistant in the Distributed Systems Group, TU Wien, Austria. He received his PhD from the Department of Computer Science and Engineering in IIT (ISM), Dhanbad, India in June 2021. He was a visiting PhD student for 6 months during PhD at Mobile & Cloud Lab, University of Tartu, Estonia. He received his M.Tech and B.Tech from JNTU Anantapur, India in 2014, and 2012, respectively. He is serving as Editor for Physical Communication and Computer Communications Journals. His current research is in Machine learning for WSNs, IoT, and Fog/Edge Computing.