Designing Reconfigurable Intelligent Systems with Markov Blankets

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

Distributed Systems Group, TU Wien, 1040 Vienna, Austria
{b.sedlak,v.casamayor,pdonta,dustdar}@dsg.tuwien.ac.at

Abstract. Compute Continuum (CC) systems comprise a vast number of devices distributed over computational tiers. Evaluating business requirements, i.e., Service Level Objectives (SLOs), requires collecting data from all those devices; if SLOs are violated, devices must be reconfigured to ensure correct operation. If done centrally, this dramatically increases the number of devices and variables that must be considered, while creating an enormous communication overhead. To address this, we (1) introduce a causality filter based on Markov blankets (MB) that limits the number of variables that each device must track, (2) evaluate SLOs decentralized on a device basis, and (3) infer optimal device configuration for fulfilling SLOs. We evaluated our methodology by analyzing video stream transformations and providing device configurations that ensure the Quality of Service (QoS). The devices thus perceived their environment and acted accordingly – a form of decentralized intelligence.

Keywords: Intelligent Systems · Computing Continuum · Markov Blankets · Sensory State · Service Level Objectives · Exact Inference

1 Introduction

Computing Continuum (CC) systems as envisioned in [2,5] are large-scale distributed systems composed of a wide variety of devices. Applications running in the CC pose ambitious requirements, e.g., near real-time latency while dealing with huge volumes of data. Additionally, requirements may change over time; to provide the best possible service, the CC system must adapt. However, given the highly distributed nature of the CC, it is a challenging task to dynamically reconfigure all contained devices, while ensuring high-level system objectives.

In this regard, we envision CC systems employing decentralized intelligence, which allows system parts to make decisions independently, in favor of the application running on top. Smaller units in the CC (e.g., edge devices) would thus obtain the ability to evaluate their own state to ensure requirements are fulfilled. One promising option to model this, is the behavioral concept introduced

Funded by the European Union (TEADAL, 101070186).

by Friston et al. [6,8]. Essentially, it comprises sensory information and actions within a Markov blanket (MB) [10], through which a *thing* interacts with its environment. The MB shields the *thing* from all the variables it is conditionally independent of. Therefore, to determine the state of the *thing*, only the variables in the MB must be considered. Transferring this concept to the CC, you could model each device's behavior through MBs [11] and evaluate device requirements by considering a limited amount of variables. Existing work [7,9,11], however, assumes prior knowledge of how metrics are related to the system state; this approach is not scalable if requirements change during operation or metric correlations are unknown at design time. Thus, existing approaches would fail to ensure the intricate requirements of CC systems.

Each tier in the CC poses its own requirements, which must be fulfilled to create a composable and unified service. To model requirements, Cloud Computing introduced Service Level Objectives (SLOs) as a means to achieve business agreements between infrastructure provider and application developer. However, we propose to expand SLOs to requirements that directly influence the system behavior and the application performance. Inspired by the work of Friston et al., and continuing the research agenda set in [3,5], we aim to leverage the behavioral concept of MBs to represent SLOs throughout the CC. The causality filter of the MB reduces the scope of variables that each device must analyze; thus, decreasing the computational effort of analysis. This empowers resource-constrained devices along the Edge to evaluate SLOs themselves.

In this paper, we propose to evaluate application requirements through MB-based SLOs. The method constrains each SLO to a set of metrics and infers configurations that fulfill them. Further, the output is explainable due to the graphical model used. Hence, the contributions of this article are the following:

- A statistical reasoning model for analyzing conditional dependencies between metrics in distributed systems. Whenever requirements change, the model may thus itself answer which metrics are related to their fulfillment.
- The graphical representation of the device state as MB, which allows interpreting the device behavior. The state can be broken down into several SLOs; in case any of them is violated, it can be explained why.
- A mechanism to infer optimal configurations from MBs given mutable system requirements. It was evaluated under two scenarios in which our approach provided the only configuration that did not violate any SLO.

2 Methodology

From a high-level perspective, we plan to analyze the device state, map selected variables to the SLO fulfillment, and provide adaptive device configurations. Our three-step methodology to achieve this is visualized in Fig. 1: Edge devices produce metrics about ongoing processing; then Bayesian Network Learning (#1) is used to identify correlations between metrics and reflect the impact of environmental changes (e.g., increased incoming requests). Next, we introduce system

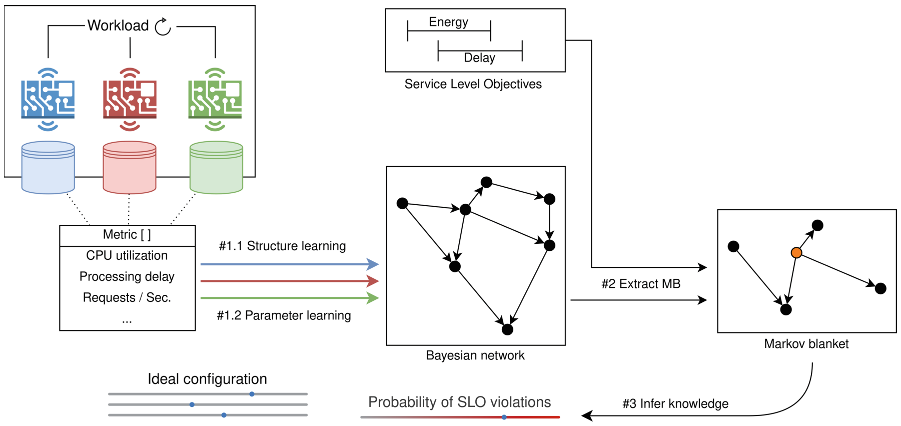


Fig. 1. Methodology for training Bayesian networks and extracting knowledge

requirements (i.e., SLOs) and extract a minimum subset of metrics for SLO fulfillment (#2). Ultimately, we use these MBs to estimate the probability of SLO violations and (#3) infer the configuration with the highest compliance level.

While the proposed methodology describes a sequence of actions, the tools themselves (e.g., algorithms for structure learning) can be optimized depending on the data. This three-step methodology will be our main mechanism for predicting the probabilities of SLO violations given a device configuration. If an SLO is violated due to an environmental change, e.g., a high request count and thus exceeded application delay, we compare possible configurations and provide the one with the highest probability of fulfilling the SLO. This matches our envisioned level of intelligence, i.e., "understanding a situation and reacting according to needs", and neatly fits the principles of elastic computing [4].

3 Case Study

The following case study will be used to evaluate our methodology. In particular, we present two video streaming scenarios that require privacy-preserving transformations. We analyze device metrics to build a Bayesian Network (BN), specify SLOs that characterize the QoS, extract the MB around each SLO, and finally, infer system configurations that have the lowest chance of violating SLOs.

3.1 Setup

Training a BN requires data; therefore, we use the framework introduced in [12], which allows edge devices to detect privacy-violating patterns (e.g., screen, face, or voice) in a stream and transform it continuously to resolve possible privacy violations. As a workload, it fits our methodology because it (1) provides an ample set of metrics reflecting the QoS of ongoing processing, (2) can be

Table 1. (Parameterizable) Metrics captured during workload execution

Name	Unit	Description	Param
delay	ms	processing time per frame	No
CPU	%	utilization of the CPU	No
memory	%	utilization of the system memory	No
pixel	num	number of pixel contained in a frame	Yes
fps	num	number of frames received per second	Yes
bitrate	num	number of pixels transferred per second	No
distance	px	relative distance of object between frames	No
transformed	T/F	if the model detected a pattern (i.e., face)	No
GPU	T/F	if the device employs a GPU	No
config	nominal	mode in which the device operates	Yes
consumption	W	energy pulled by the device	No

parameterized, and (3) can be executed on edge devices. Using the framework, we specify a privacy model that detects faces within a video stream and blurs the respective region, a scenario useful for office monitoring or AR setups [1, 12]. During execution, 11 metrics are captured, which we introduce in Table 1. Each row contains a short description, the measurement unit, and if it can be parameterized. For example, *pixel* and *fps* are video stream properties; however, the producer can adapt them to create a variable *bitrate*. *Config* determines the device operation mode; devices such as Nvidia Jetson Xavier NX[1] can thus limit their energy consumption and the number of active CPU cores. It is worth mentioning the metric *distance*, which tracks the relative position of a detected face between frames, indicating how fluent/sluggish an object is tracked.

To explore correlations between metrics, we simulate an adaptive bitrate; precisely, the producer periodically switches between different *fps* (12, 16, 20, 26, 30) and *pixel* (120p, 180p, 240p, 360p, 480p, 720p), while the edge device moves through *config* modes. Current parameter assignments are part of the metrics set, which is persisted with every processed frame. Metrics are directly observable by the device; except for *consumption*, which is captured through an external power plug[2] over a telemetry period of 10 s. Metrics are accumulated in a CSV file, which will contain 756,000 rows, captured within 2.5 h.

We identified five SLOs that describe the system state in terms of QoS and Quality of Experience (QoE); however, each applicable scenario can have its own subset of relevant SLOs. We assign a name to each SLO and highlight the metrics from Table 1 (e.g., *bitrate*) that are used to evaluate the state of the SLO. Some SLOs are constructed by combining metrics (i.e., **within_time**), others are com-

[1] https://docs.nvidia.com/jetson/archives/r34.1/DeveloperGuide/text/SO/JetsonXavierNxSeries.html, accessed June 13th 2023.

[2] https://www.delock.com/produkt/11827/merkmale.html, accessed June 13th 2023.

pared against a customizable threshold (e.g., **pixel_distance**), while other SLOs directly mimic the value (True/False) of the metric (i.e., **transform_success**).

network_usage Edge devices have limited network interfaces, and in some cases, limited network bandwidth. Since video streams are transferred over the network, *bitrate* is important to control network congestion.

energy_cons Edge devices are restricted in terms of resources and thus must economize or limit their energy *consumption* while ensuring compliance with the remaining system requirements (i.e., other SLOs).

within_time Video processing introduces a considerable streaming *delay*, which can lead to dropping frames and consequently poorer QoE. Hence, the stream's *fps* can be adjusted to limit/avoid dropping frames.

pixel_distance Measures the quality of the object tracking capacity; we expect the tracked object not to jump, but to have a smooth trajectory. Hence, we define a range for the acceptable *distance*.

transf_success Private or confidential information must not be disclosed; therefore, it must maximize the number of transformed faces in the stream.

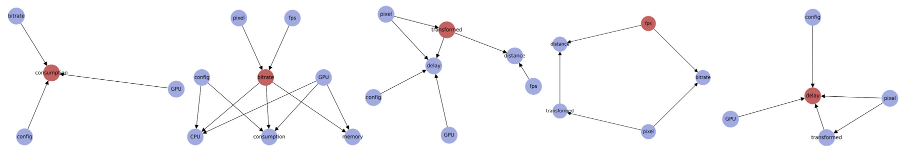
The workload was executed on the Jetson Xavier NX, which supports GPU-accelerated video processing over NVIDIA CUDA. To explore correlations between *GPU* and other metrics, we execute the entire workload twice on the Xavier NX – once with and once without CUDA acceleration enabled.

3.2 Model Construction

For constructing the BN, we leverage *pgmpy*[3], a Python-based framework that supports an ample set of algorithms for structure and parameter learning, e.g., Hill-Climb Search (HCS) and Maximum Likelihood Estimation (MLE). We train the BN with HCS and MLE on all captured metrics, which takes roughly 30 s on the Xavier NX. After training the BN, we extract the MB for each SLO; the resulting MBs are visualized in Figs. 2: The first three graphics show simple SLOs, i.e., such that require exactly one metric for evaluation. For example, **energy_cons** must evaluate *consumption* to determine the state of the SLO. Metrics that have an edge pointing to *consumption* (i.e., *bitrate*, *config*, and *GPU*) causally influence the variable and thus the SLO fulfillment. On the other hand, **within_time**, is composed of two metrics and thus features two MBs. Complex SLOs [9] (i.e., such that consist of n metrics) would produce n MBs; therefore, increasingly complex SLOs will require a sophisticated mechanism to merge and compress MBs.

We argue that the MBs extracted for each SLO are plausible because contained edges can be rationally explained. Further, all MB SLOs contain at least one parameterizable metric within their sensory state, i.e., among the variables that influence the SLO outcome. From a requirements perspective, this is essential because it allows a device to adapt dynamically to fulfill given SLOs.

[3] https://pgmpy.org/, accessed June 14th, 2023.



(a) energy_cons (b) network_use (c) transf_succ (d) *fps* (e) *delay*

Fig. 2. Markov blankets of the SLOs extracted from the Bayesian network

3.3 Device Configuration Inference

To infer device configurations that comply with the SLOs, we extract information from the BN with Variable Elimination (VE). Instead of querying the entire BN, we execute the queries on the minimum subset of relevant variables, i.e., the MB of each SLO. Since the MBs of the SLOs contained all three parameterizable metrics (i.e., *fps*, *pixel*, and *config*), a device must include these parameters in an inferred configuration; otherwise, there is no full control over the SLOs. However, suppose we would only trace a subset of the SLOs (e.g., **network_usage** & **transf_success**), a configuration must only include the respective parameters contained in the MBs, e.g., *fps* & *pixel*, but not *config*.

VE computes the probability of SLO violations for exactly one parameter assignment; we repeatedly apply this approach for all assignments. To be precise, the parameter space for ($pixel$: fps : $config$) consists of (5 : 6 : 3) possible assignments. Iterating over $5 * 6 * 3 = 90$ combinations and 5 SLO-MBs produces $5 * 90 = 450$ inference queries, which require roughly 700ms on the Jetson Xavier NX. The result is a list of configurations that fulfill the given SLOs, e.g., one could be (240p : 20fps : 4C_20W). To deal with changing requirements and heterogeneous characteristics of CC devices, it is possible to provide additional constraints to the VE (e.g., GPU=False), or customize SLOs to rank a metric rather than limiting it (e.g. minimize *consumption*).

3.4 Evaluation

To evaluate the quality of inferred configurations, we compare the number of SLO violations between devices that apply inferred or arbitrary configurations. We envision two scenarios that are based on the workload for face blurring. The scenarios are described below, while the corresponding SLO thresholds are presented in Table 2. We intend to minimize **energy_cons** for both scenarios regardless of whether the energy supply would be constrained:

Scenario A: To create a virtual map (like Google Street View[4]), a camera-equipped car captures street videos. The car has an edge device installed to transform the stream; the result is directly rendered to a local map and only accessed remotely in case of inspection, so **network_usage** is of less importance. We assume the rendering process to run in the background; therefore, the GPU

[4] https://www.google.com/streetview/, accessed June 18th 2023.

is not available for processing. To create a detailed map, **pixel_distance** must be low, and **within_time** fulfilled in most cases. However, the stream can be re-rendered to blur undetected faces, thus **transf_success** is less critical.
Scenario B: Within a smart factory, employees equipped with head-mounted cameras conduct an audit. To protect privacy, the video stream is transformed on an edge device before streaming to remote consumers. Video content is intended for live inspection only; therefore, **pixel_distance** and **within_time** are less important, while high **transf_success** prevents privacy breaches. However, since audits involve various providers and consumers, low **network_usage** is desired.

Table 2. SLO thresholds that reflect the scenarios' requirements

Scenario	transf_success	distance	network_usage	within_time	energy_cons	GPU
A	$\geq$ 90%	$\leq$ 35	$\leq$ 8.2 $Mio.\ px/s$	$\geq$ 95%	$min(x)$	No
B	$\geq$ 98%	$\leq$ 60	$\leq$ 1.6 $Mio.\ px/s$	$\geq$ 75%	$min(x)$	Yes

Table 3. List of configurations generated by exact inference or picked naively

Scenario	Source	Resolution	FPS	Mode	GPU
A	inferred	240p	20	4C_15W	No
	naive	360p	30	6C_20W	
	random #1	120p	16	6C_20W	
	random #2	720p	12	2C_10W	
B	inferred	240p	16	2C_10W	Yes
	naive	180p	26	4C_15W	
	random #1	360p	20	2C_15W	
	random #2	480p	30	6C_20W	

We supply the SLO thresholds to the inference mechanism; the resulting configurations are presented in Table 3: The first line contains the inferred configuration, and the second line the naive assumption; the third and fourth lines are randomly generated. To evaluate the number of SLO violations, we measured each configuration's performance over 10 min; results are presented in Table 4. Over the measurement course, the inferred configurations fulfilled the SLOs for both scenarios. The naive assumption, on the other hand, violated one SLO within each scenario (red cell), i.e., in Scenario A it failed to fulfill **within_time**, while in Scenario B **transf_success** was violated. The randomly generated configurations committed two SLO violations in Scenario A and one in Scenario B each. The results show that our inferred configurations fulfilled all given SLOs while also consuming the least energy.

Table 4. Fulfillment of SLOs depending on scenario and configuration

Scenario	Source	**transf_success**	**distance**	**network_usage**	**within_time**	**energy_cons**
A	inferred	98%	15 (97%)	2.0 Mio.	100%	6.0W
	naive	100%	10 (100%)	6.9 Mio.	92%	8.0W
	random #1	4%	127 (2%)	0.4 Mio.	100%	7.0W
	random #2	100%	28 (89%)	11 Mio.	100%	6.0W
B	inferred	98%	18(98%)	1.6 Mio.	100%	6.0W
	naive	92%	11(99 %)	1.5 Mio.	100%	6.5W
	random #1	99%	15 (100%)	4.6 Mio.	100%	6.0W
	random #2	100%	10 (100%)	12.3 Mio.	97%	7.5W

4 Conclusion and Future Work

This paper proposed a statistical reasoning model for explaining causal relations between metrics and the system state, which is reflected by a set of SLOs and their MBs. Essentially, this provides individual edge devices with decentralized intelligence, which helps to cope with the scale and complexity of CC systems. Our methodology was able to provide configurations that would not commit SLO violations; however, the scale of CC systems makes it necessary to assess the impacts of increasingly large Bayesian networks in terms of performance and precision. Furthermore, to cover heterogeneity among CC devices, we aim to infer configurations for arbitrary devices.

References

1. Baniya, P., et al.: Towards policy-aware edge computing architectures. In: 2020 IEEE International Conference on Big Data (Big Data), December 2020
2. Beckman, P., et al.: Harnessing the computing continuum for programming our world. In: Fog Computing, pp. 215–230. John Wiley & Sons, Ltd., April 2020
3. Casamayor Pujol, V., Raith, P., Dustdar, S.: Towards a new paradigm for managing computing continuum applications. In: IEEE 3rd International Conference on Cognitive Machine Intelligence, CogMI 2021, pp. 180–188 (2021)
4. Dustdar, S., Guo, Y., Satzger, B., Truong, H.L.: Principles of elastic processes. Internet Comput. IEEE **15**, 66–71 (2011)
5. Dustdar, S., Pujol, V.C., Donta, P.K.: On distributed computing continuum systems. IEEE Trans. Knowl. Data Eng. **35**(4), 4092–4105 (2023). https://doi.org/10.1109/TKDE.2022.3142856
6. Friston, K.: Life as we know it. J. R. Soc. Inter. **10**(86), 20130475 (2013). https://doi.org/10.1098/rsif.2013.0475
7. Fürst, J., Fadel Argerich, M., Cheng, B., Papageorgiou, A.: Elastic services for edge computing. In: 2018 14th International Conference on Network and Service Management (CNSM), pp. 358–362, November 2018
8. Kirchhoff, M., Parr, T., Palacios, E., Friston, K., Kiverstein, J.: The Markov blankets of life: autonomy, active inference and the free energy principle. J. R. Soc. Inter. **15**(138), 20170792 (2018)

9. Nastic, S., et al.: SLOC: service level objectives for next generation cloud computing. IEEE Internet Comput. **24**(3) (2020). https://doi.org/10.1109/MIC.2020.2987739
10. Pearl, J.: Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. Morgan Kaufmann, San Mateo, California (1988)
11. Sedlak, B., Casamayor Pujol, V., Donta, P.K., Dustdar, S.: Controlling data gravity and data friction: from metrics to multidimensional elasticity strategies. In: IEEE SSE 2023, Chicago, IL, USA, July 2023
12. Sedlak, B., Murturi, I., Donta, P.K., Dustdar, S.: A privacy enforcing framework for transforming data streams on the edge. IEEE Trans. Emerg. Top. Comput. (2023). https://doi.org/10.1109/TETC.2023.3315131