Distributed Intelligence in the Computing Continuum – Part 1

Prof. Dr. Schahram Dustdar
TU Wien

SummerSoC 2026

Thanks to my collaborators: Guodong Wang, Victor Casamayor-Pujol, Alireza Furutanpey, Boris Sedlak, Praveen Donta

The Computing Continuum

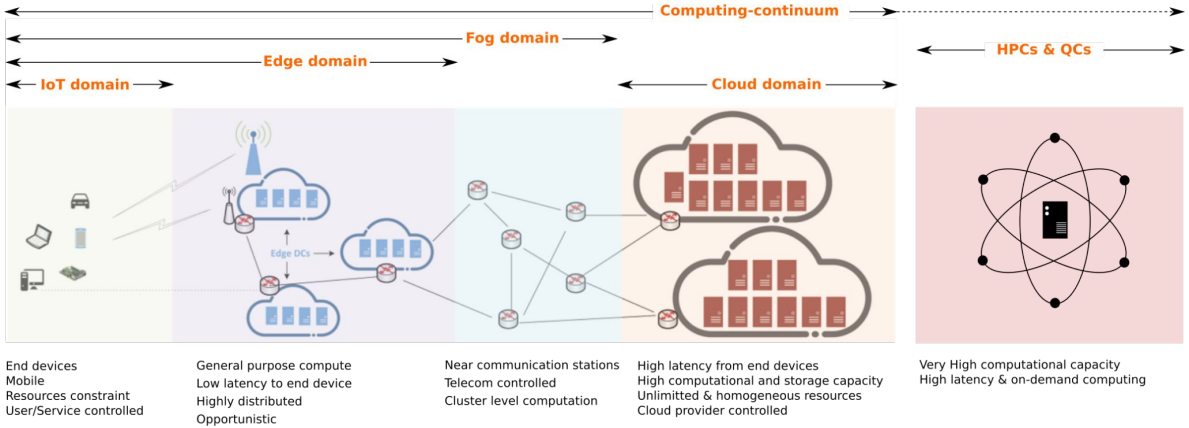
Computing fabric composed of all current computational tiers.

A seamless integration of
the computing infrastructure.

Leverages the best of each tier.

Expected applications:

- ➔ eHealth
- ➔ Autonomous vehicles
- ➔ Smart cities
- ➔ Resources management



Today we have a centralized and limited visibility over the system performance, quality of service (QoS), and Quality of data.

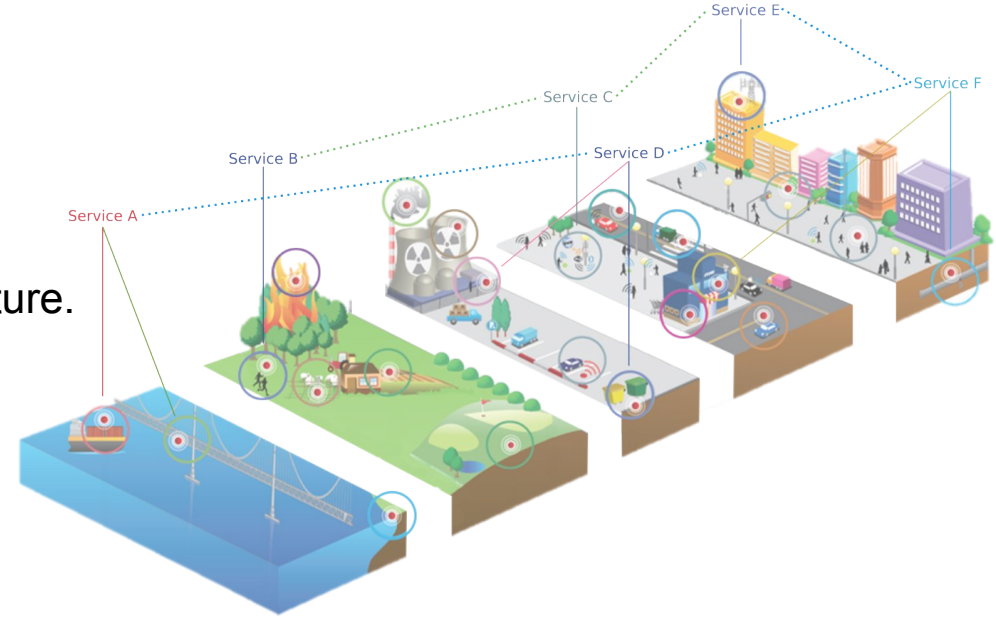
The Computing Continuum

Multi-proprietary: Shared infrastructure ownership

System *issues* propagate

Each stakeholder has:

- ➔ Own global interest
- ➔ Local requirements of its infrastructure.



We need tools to understand the relationship between each SLO (requirement) and how propagation unfolds.

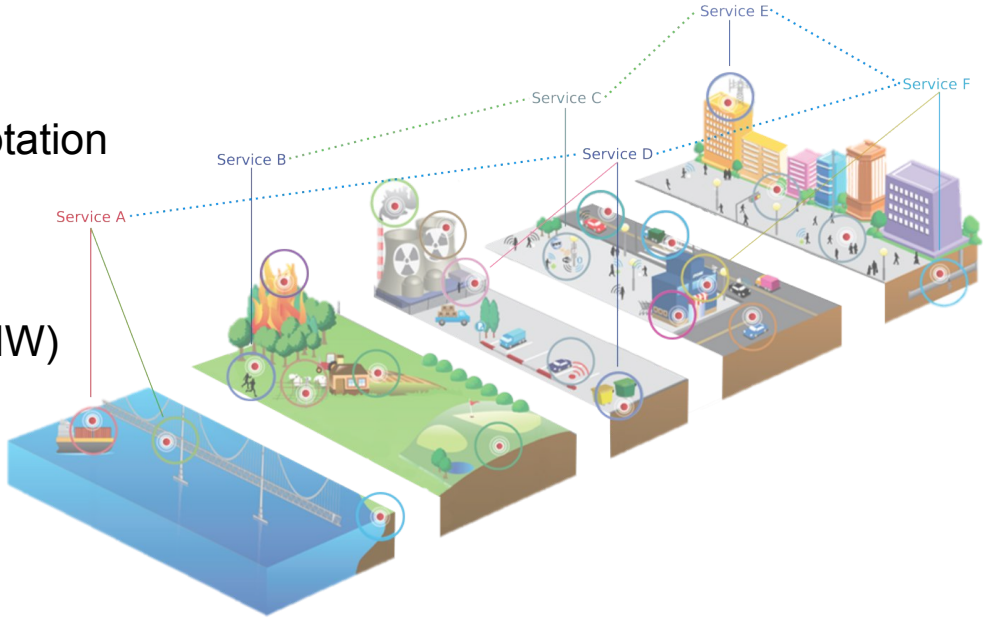
The Computing Continuum

Geographically distributed

Challenges deployment and service adaptation

Centralized governance falls short
(intensified by stricter requirements)

Tailored runtime adaptations (Service + HW)



We need decentralized governance (intelligence), which considers local characteristics of the service and the host.

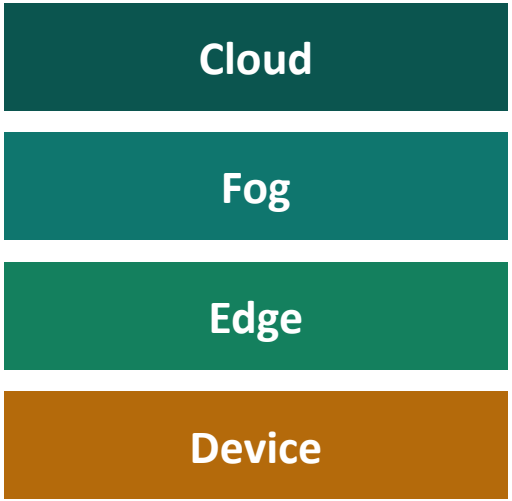
The Computing Continuum

A unified substrate that integrates device → edge → fog → cloud and treats them as one resource pool rather than separate silos.

- Workloads shift across tiers by load, locality, energy budget, or privacy constraints.

- Service-Level Objectives (SLOs) must hold across all tiers despite nonstationarity and bounded adversarial perturbations.

- The management problem: who decides what runs where, on what signal, and how often?



one continuum, one SLO

DeepSLO — the contract abstraction

A hierarchically decomposable Service-Level Objective that binds latency, throughput, accuracy, and resource usage together.

- "Deep" = the SLO factorises over the BN vocabulary V — each node carries a local slice of the contract.
- The agent never sees the whole contract; it sees its Markov-blanket slice plus a preference over outcomes.

FORMALLY

Joint preference distribution

$$C(o) = p(o_1, o_2, \ldots, o_N)$$

o = (o_1, …, o_N): outcome vector observed across tiers
C(o) encodes "what on-contract looks like"

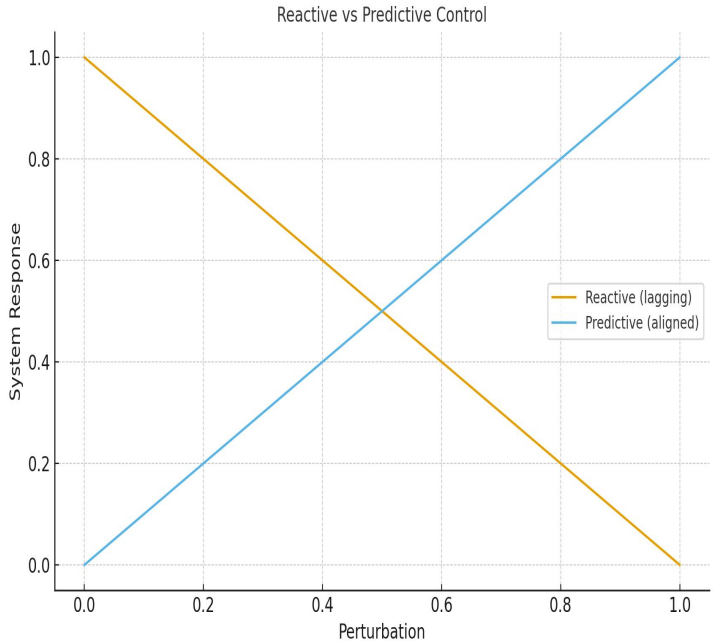
Why reactive / threshold control fails

Threshold rules presume stationarity and centralised observability — both assumptions break at continuum scale.

- Nonstationary workloads: the distribution we calibrated thresholds on is not the distribution we observe.
- Drifting topology: nodes join / leave / migrate; Markov blankets move.
- Bounded adversarial perturbations: packet loss bursts, GPU co-tenancy spikes, accuracy drift.
- Reactive controllers lack a predictive signal — they only notice violations after the fact, after a violation has already accrued.
- **What we need: predictive, distributed, adaptive control with a first-class gauge of "model reality mismatch".**

Governance (Intelligence) in Continuum Computing

- Distributed apps span sensors, edge, fog, and cloud.
- Reactive, centralized management often fails.
- Non-stationarity breaks SLO compliance.
- Need predictive regulation over reactive.

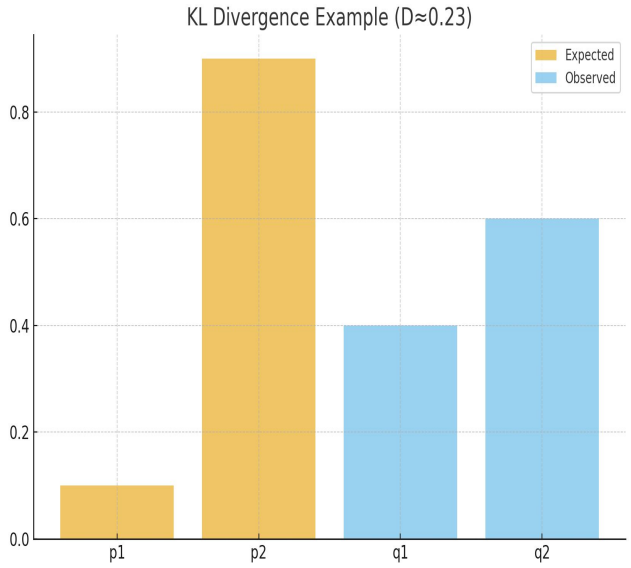


Biological Lens: Predictive Regulation

- Organisms regulate by anticipating changes.
- Homeostasis = reactive; Allostasis = predictive.
- Free Energy Principle: minimize prediction error.
- Analogy: components should anticipate loads.

Physics Lens: Fluctuation–Dissipation Theory (FDT)

- FDT links fluctuations to responses.
- Departures signal nonequilibrium dynamics.
- In computing: compare expected vs observed responses.
- Alignment = predictive equilibrium.

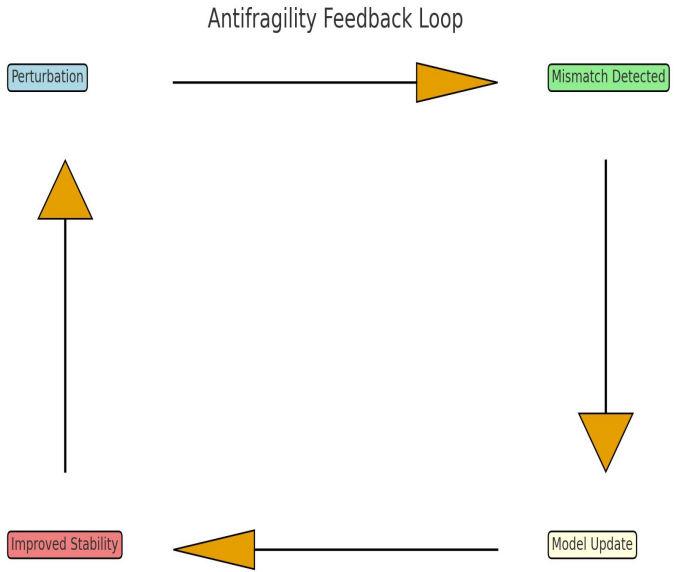


Predictive Equilibrium Defined

- Alignment between predicted and observed outcomes.
- Active property: sustained by adaptation.
- Combines dynamic balance, reconfiguration, predictive consistency.
- Basis for antifragility.

Antifragility in Distributed Systems

- Systems improve because of stress, nc despite it.
- Perturbations expose mismatches → learning opportunity.
- BNs + KL provide learning gradient.
- Equilibrium supplies safety rails.



How the four notions nest

After sufficient exposure:

What the controller has to remember:

Robustness
just the bound

Resilience
the set-point

Antifragility
a structural improvement

Taleb's four notions of how systems respond to stress

Fragility

Worse than linear in stress: doubling the shock more than doubles the damage.

Response curve is concave down with a cliff.

Robustness

Insensitive up to a threshold: the system holds $R \geq R_{min}$ for all stress $u \leq u^*$.

Remembers only the bound.

Resilience

Returns to its set-point within bounded time after a shock.

Remembers the set-point.

Antifragility

Strictly better after repeated exposure — the system gains from disorder.

Remembers a structural improvement.

Architecture: Predictive Equilibrium in Action

- Local BNs at edge nodes predict responses.
- Fog nodes run perturbation campaigns and compute divergence.
- Cloud meta-controller aggregates signals.
- Balances exploration vs exploitation.

Mathematical Foundations: Bayes' Theorem

Bayes' rule updates beliefs given new observations.

- Prior p(s): belief before seeing data.
- Likelihood p(o|s): how observations arise from state
- Evidence p(o): normalizing constant.
- Posterior p(s|o): updated belief after observation.

$$p(s|o) = \frac{p(o|s)\,p(s)}{p(o)}$$

Posterior = Likelihood x Prior / Evidence

Bayes' Theorem Example: Rain and Wet Grass

- Prior: P(Rain) = 0.3 (30% chance of rain).
- Likelihood: P(Wet|Rain) = 0.9 (grass usually wet if it rains).
- Alternative: P(Wet|¬Rain) = 0.1 (sprinkler can also make grass wet).
- Observation: Grass is wet → update belief about rain.
- Posterior: P(Rain|Wet) ≈ 0.64 (now more likely it rained).

$$P(Rain|Wet) = \frac{P(Wet|Rain)\,P(Rain)}{P(Wet)}$$

Posterior = Likelihood × Prior / Evidence

Active Inference

Describes how systems maintain their states and make predictions about their environment to minimize free energy.

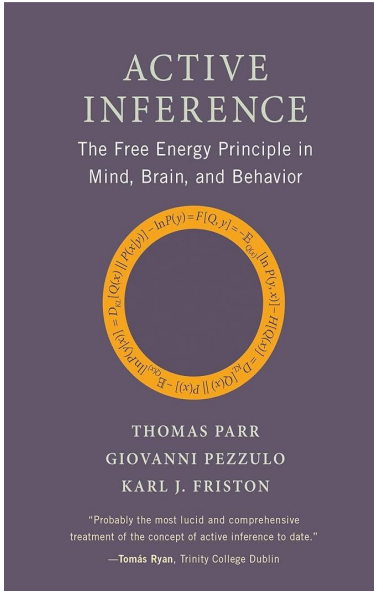
Active Inference Framework

1. **Objective**: Minimize free energy (a measure of uncertainty) to maintain homeostasis and predict environmental changes.

2. **Core Concepts**:
   - Free Energy Principle: Systems minimize the difference between predicted and actual sensory inputs.
   - Bayesian Inference: Use of probabilistic models to update beliefs about the state of the world based on new data.
   - Generative Models: Systems use models to generate predictions about sensory inputs and outcomes.

3. **Mechanisms**:
   - Perception: Involves updating beliefs about the state of the environment based on sensory inputs.
   - Action: Involves selecting actions that minimize expected free energy by reducing prediction errors.
   - Learning: Adjusting the parameters of generative models to improve future predictions and actions.

Active Inference is a framework that integrates perception, action, and learning through Bayesian inference and generative models to minimize prediction errors and free energy. It has significant applications in fields like robotics, machine learning, and cognitive computing, where systems need to predict, adapt, and learn from their environment efficiently.



[1] Friston et al.,Designing Ecosystems of Intelligence from First Principles, https://doi.org/10.48550/arXiv.2212.01354
[2] Friston, Life as we know it, https://doi.org/10.1098/rsif.2013.0475
[3] Palacios et al., On Markov blankets and hierarchical self-organisation, https://doi.org/10.1016/j.jtbi.2019.110089
[4] Kirchhoff et al., The Markov blankets of life: autonomy, active inference and the FEP, https://doi.org/10.1098/rsif.2017.0792
[5] Parr et al., Active Inference: The Free Energy Principle in Mind, Brain, and Behavior, https://doi.org/10.7551/mitpress/12441.0

Theoretical Foundation: The Free Energy Principle & Active Inference

The Free Energy Principle (FEP) states that any self-organizing system that resists disorder must minimize variational free energy — the divergence between its internal model and sensory evidence. Active Inference (AIF) operationalizes FEP: an agent maintains a generative model and selects actions to minimize Expected Free Energy (EFE).

Variational Free Energy (VFE)

$$F = \mathbb{E}q(s)\,[\,\ln q(s) - \ln p(o, s)\,]$$

Smith, Friston & Whyte (2022), J. Math. Psych.

Complexity–Accuracy Decomposition

$$F\pi = D_{KL}[\,q(s|\pi) \,\|\, p(s|\pi)\,] \;-\; \mathbb{E}q(s|\pi)[\,\ln p(o|s)\,]$$

complexity *accuracy* *— Friston (2010)*

Expected Free Energy (EFE) — Epistemic + Pragmatic Decomposition

$$G\pi \;=\; -\,\mathbb{E}q(o,s|\pi)[\,\ln q(s|o,\pi) - \ln q(s|\pi)\,] \;-\; \mathbb{E}q(o|\pi)[\,\ln p(o|C)\,]$$

epistemic value (information gain) *pragmatic value (goal achievement)* *— Parr & Friston (2019)*

Epistemic Value: Drives the agent to actively seek informative observations, reducing model uncertainty (novelty).
Instrumental Value: Drives the agent to minimize deviation from prior preferences — achieving goals (SLO compliance).

Friston (2010) Nat. Rev. Neurosci.; Parr & Friston (2019) Biol. Cybern.; Parr, Pezzulo & Friston (2022) Active Inference, MIT Press

Variational Free Energy (VFE)

- Measures how well an approximate posterior Q(s) matches the true posterior.
- Minimizing F ≈ maximizing model evidence (probability of data under model).
- Balances Accuracy (fit to data) and Complexity (change in beliefs).
- Tractable alternative to exact Bayesian inference.

$$F = D_{KL}[Q(s)|P(s|o)] - \ln P(o)$$

Expected Free Energy (EFE)

Extends free energy to future outcomes.

- Evaluates policies π: sequences of actions.
- Captures both pragmatic value (rewards) and epistemic value (information gain).
- Provides solution to explore–exploit dilemma

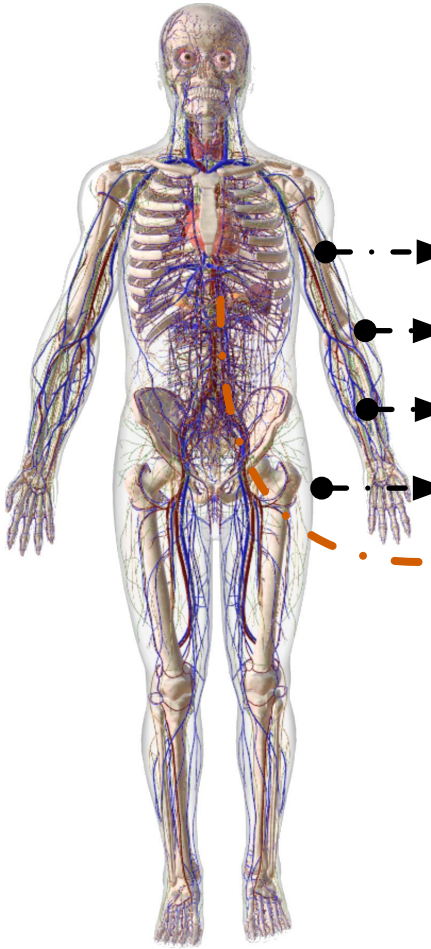
$$G(\pi) = \mathbb{E}_Q[\ln Q(s|\pi) - \ln P(o, s|\pi)]$$

Kullback–Leibler (KL) Divergence

Measures dissimilarity between two probability distributions.

- D=0 when Q (posterior) and P (prior) match perfectly; larger values mean greater divergence.
- In Active Inference: diagnostic of model–world mismatch.
- Used as both early warning and learning signal.

$$D_{KL}[Q(x)|P(x)] = \sum_{X} Q(x) \ln \frac{Q(x)}{P(x)}$$



Human Ecosystem

The human body is comprised of a series of complex systems, including:

- Skeletal System
- Nervous System
- Cardiovascular System
- Lymphatic System
- Endocrine System

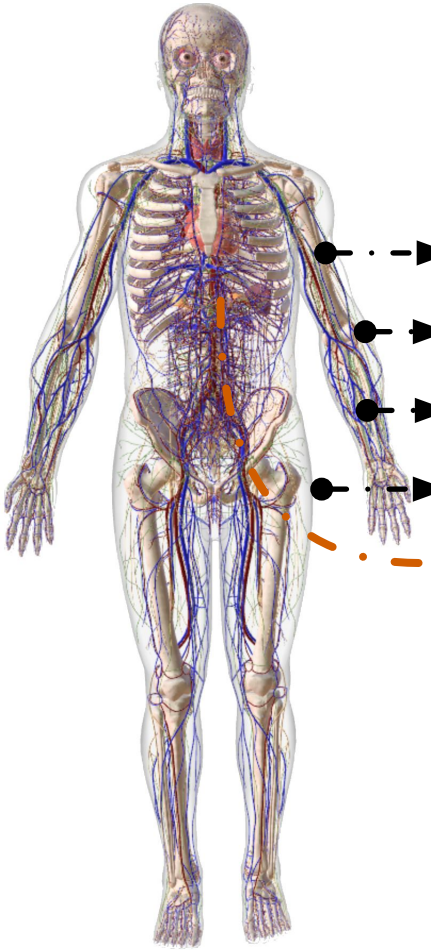
→ Infrastructure Systems

→ Regulation Systems

- Brain
- Spinal Cord
- Cranial Nerves
- Spinal Nerves
- Oxygen
- White Blood Cells
- Hormones
- Nutrients

Helping the body meet the demands (**40k neurons**)

Control Internal Environment, Memory and Learning (**86 billion neurons**)



Human Ecosystem

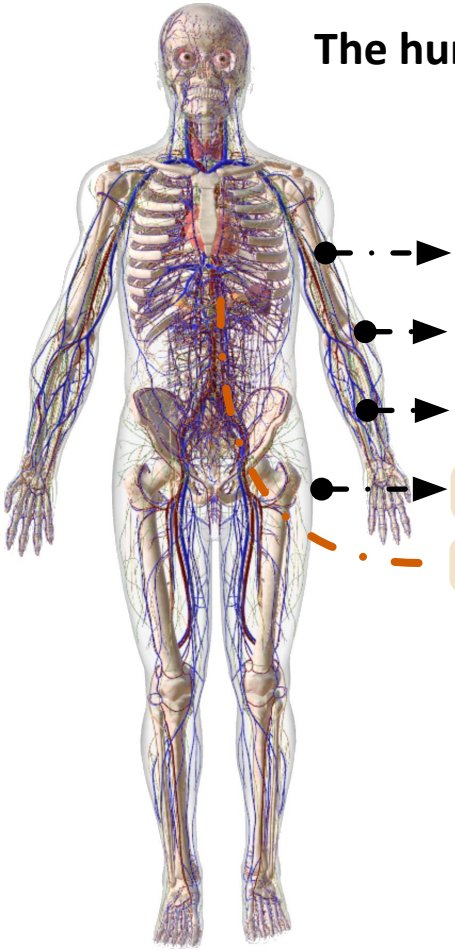
The human body is comprised of a series of complex systems, including:

- Skeletal System
- Nervous System
- Cardiovascular System
- Lymphatic System
- Endocrine System

→ Infrastructure Systems

→ Regulation Systems

The human body is comprised of a series of complex systems, including:



Human Ecosystem

- Skeletal System
- Nervous System
- Cardiovascular System
- Lymphatic System
- Endocrine System

→ Infrastructure Systems

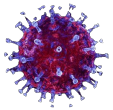
DeepSLOs

Collaborative Learning

Representation Learning

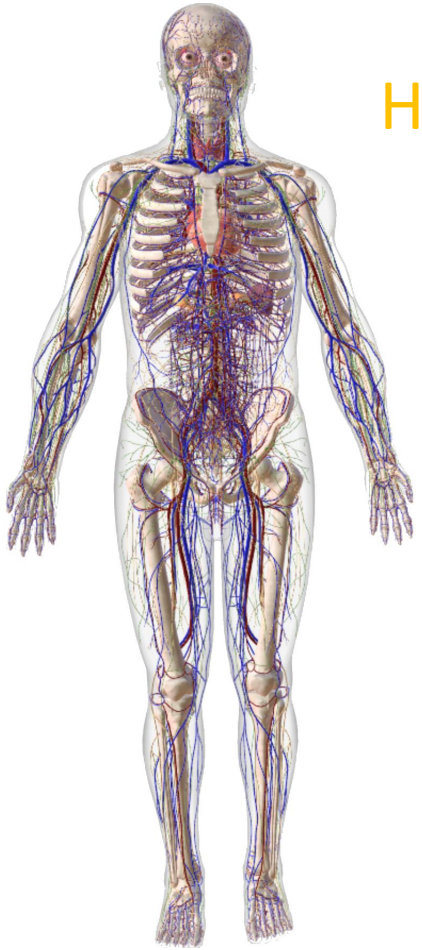
→ Regulation Systems

Zero Trust



- Part of the immune system
- Protects your body against foreign invaders

Homeostasis and Resilience in DCCS



Nervous system

Human Ecosystem

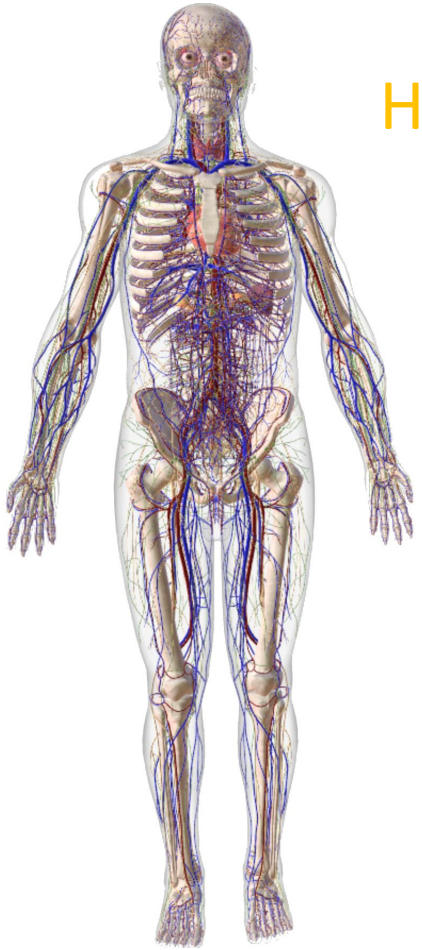
Human body self-regulates:

- Temperature
- Blood pressure
- …

Human body self-heals

Humans also learn how to maintain her/his needs satisfied.

Homeostasis and Resilience in DCCS



Human Ecosystem

Nervous system

Overall state - Top-bottom sensing.

From feeling *good-bad* to actual problem.

We also need this feature for DCCS due to their scale and interconnections.

Elasticity (Resilience)

(Physics) The property of returning to an initial form or state following deformation

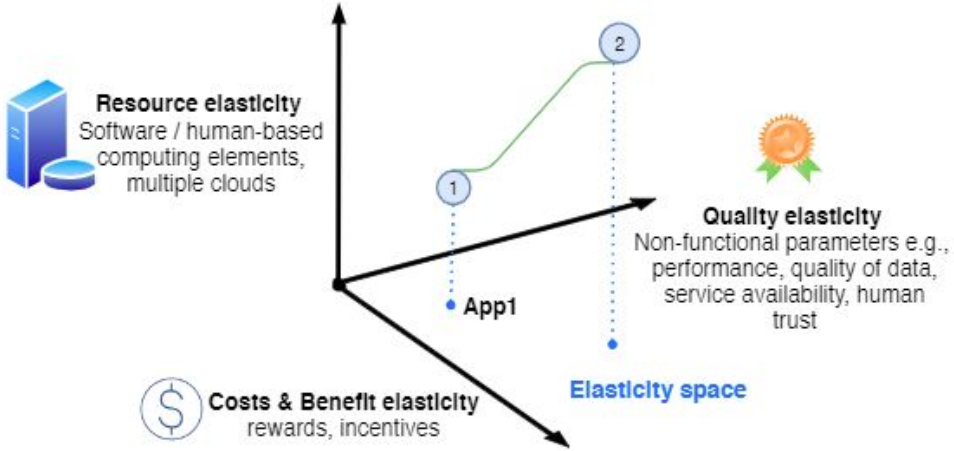
stretch when a force stresses them

e.g., ***acquire*** *new resources,* ***reduce*** *quality*

shrink when the stress is removed

e.g., ***release*** *resources,* ***increase*** *quality*

Elasticity > Scalability

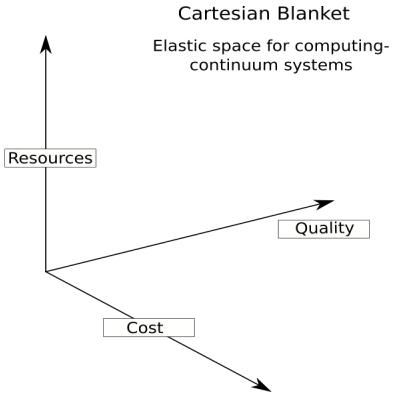


Dustdar S., Guo Y., Satzger B., Truong H. (2012) Principles of Elastic Processes, IEEE Internet Computing, Volume: 16, Issue: 6, Nov.-Dec. 2012

High-level state

Resources, Quality, Cost

- Highest-level description of system state from Cloud computing/elasticity work [1].
- DCCS have many different stakeholders with different interests, RQC can frame a common language.



Operational equilibrium

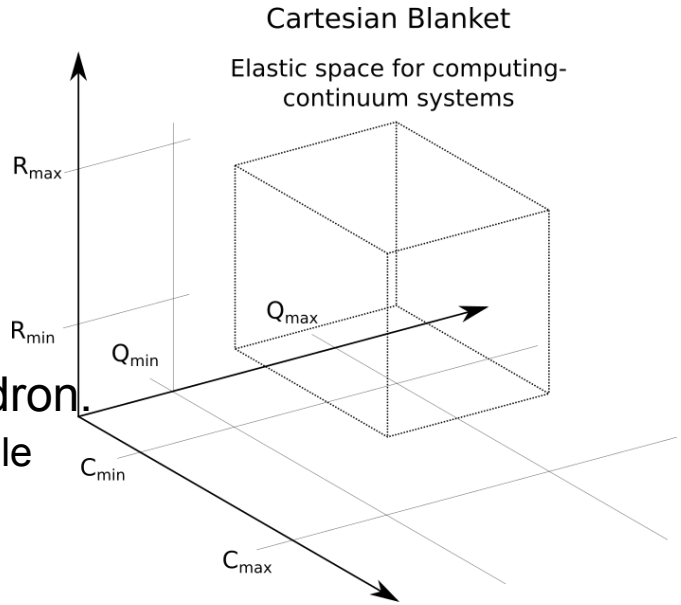
- Defined as an operational mode of the application, from the highest level state.
- Any system can have several operational equilibria, leading to different configurations of the underlying infrastructure

[1] 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, doi: 10.1109/MIC.2011.121.

The Cartesian Blanket

Adapting elasticity in the continuum

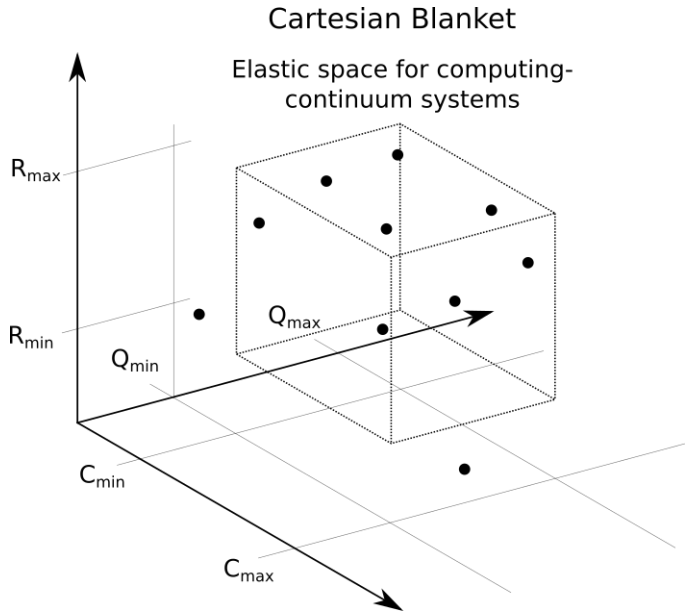
- System control based SLOs (Service Level Objectives)
- SLOs are represented as thresholds on the Cartesian space
- The system space is delimited within an hexahedron.
  - There is minimum and maximum value for each variable



The Cartesian Blanket

Adapting elasticity in the continuum

- The space is constraint to the actual infrastructure characteristics; not homogenous.
- The infrastructure is represented as points, not unlimited.
- The only valid infrastructure is the one **inside** the hexahedron.



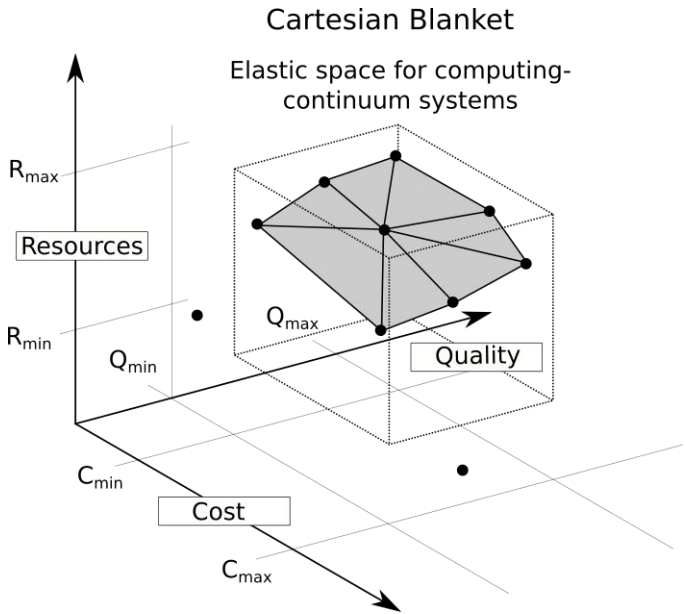
The Cartesian Blanket

Adapting elasticity in the continuum

- The system space possible configurations can be visualized as a stretched blanket over the infrastructure points.
  - Assuming linear interpolation on the space between the infrastructure components.

- Now we have the system represented, but

How can this representation help on the design and management of the distributed computing continuum systems?



The Problem: Why Current AI Is Not Truly Autonomous

1. Reactive, Not Proactive

Current systems rely on pre-trained pattern matching without an internal generative model of the world. They predict, but do not understand causality.

2. No Self-Diagnosis Capability

When performance degrades due to environmental drift, there is no causal mechanism to distinguish sensor noise (irreducible variability) from model failure (reducible novelty).

3. Fragmented Optimization Objectives

Perception (cross-entropy), control (RL reward), and deployment (heuristics) are governed by separate loss functions — no unified "purpose" drives the system.

Paradigm Contrast

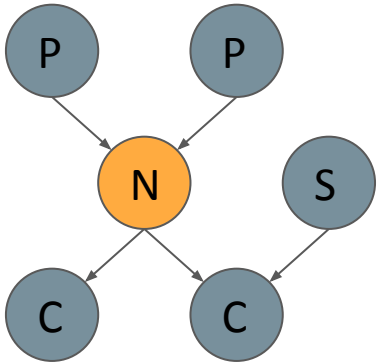
Dimension	Data-Driven (Passive)	Model-Driven / AIF (Active)
Data Efficiency	Requires massive labeled data	Small-sample, online learning
Causal Reasoning	Correlational only	Explicit causal structure
Planning Horizon	Myopic (instant reward)	Long-horizon (minimize surprise)
Edge Footprint	Heavy (GPU-dependent)	≤ 300 MB memory

Friston (2010) Nature Reviews Neuroscience; LeCun (2022) "A Path Towards Autonomous Machine Intelligence"

Markov Blanket

The Markov Blanket of a random variable is the subset of nodes that provide enough information to statistically infer its value. Concept from Judea Pearl [1].

In a Bayesian Network, the Markov Blanket of a node (N) is composed of the parents (P), the children (C) and the co-parents of the children (S).

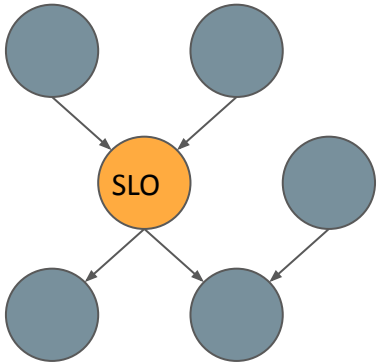


A tool for *causal* filtering.

[1] Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc.

Causal Inference

- Discover & leverage causal relationships.
- 3 Rungs on the ladder of causation. [2]
  - Observational
  - Interventional
  - Counterfactual
- Explainability capacity



[2] J. Pearl and D. Mackenzie (2018), The Book of Why: The New Science of Cause and Effect. USA: Basic Books, Inc.,

Markov Blankets: Defining the Agent–Environment Boundary

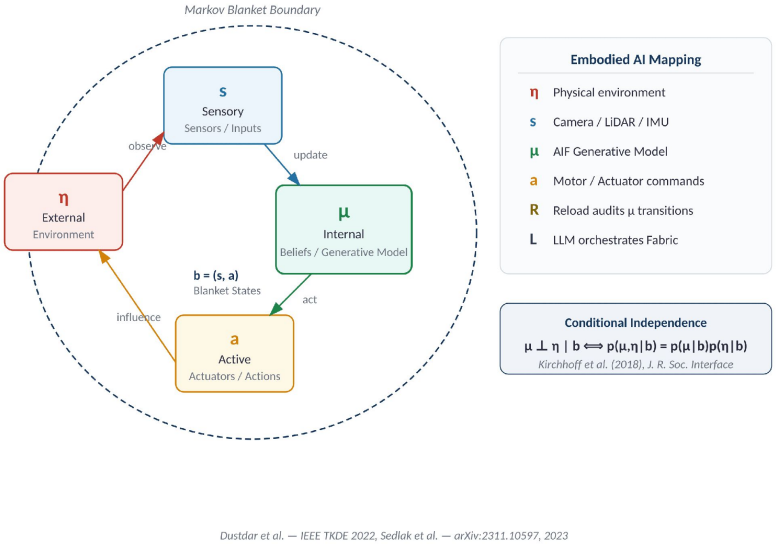
The Markov Blanket formalism partitions system states into four categories, defining the causal information boundary between an agent and its environment. This formalism has been extended by Dustdar et al. (2022) to distributed computing continuum systems spanning IoT → Edge → Fog → Cloud.

State Partition	Definition	Embodied AI Mapping
External (η)	Hidden environmental variables	Physical environment, unseen perturbations
Sensory (s)	Probabilistic observations	Camera feeds, LIDAR, IMU sensor streams
Active (a)	Agent output actions	Actuator commands, gimbal rotation, OTA
Internal (μ)	Agent beliefs (generative model)	Onboard world model, edge compute state

Conditional Independence

$$\mu \perp \eta \mid b \iff p(\mu, \eta \mid b) = p(\mu \mid b) \cdot p(\eta \mid b)$$

where b = (s, a) are blanket states — Kirchhoff et al. (2018) J. R. Soc. Interface



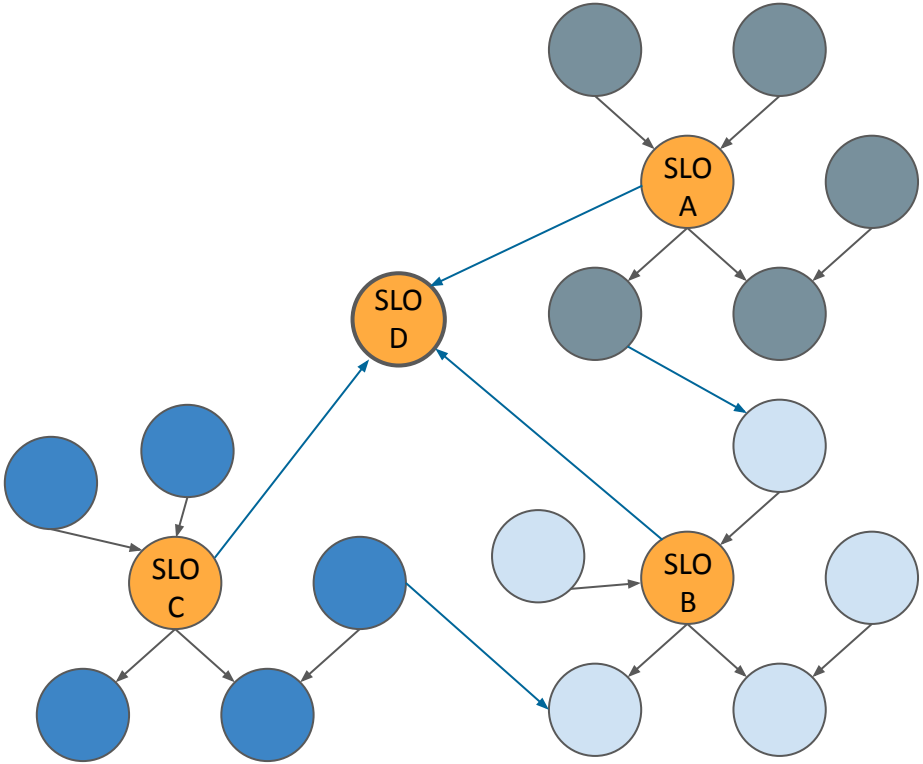
Dustdar et al. — IEEE TKDE 2022, Sedlak et al. — arXiv:2311.10597, 2023

Key Insight: The transition logic from sensory states to internal beliefs generates an auditable ***Rationale Trace*** — providing causal explainability for every decision. Dustdar's group has applied nested Markov blankets as **causality filters** for decentralized SLO evaluation on edge devices.

Dustdar et al. (2022) IEEE TKDE; Sedlak, Casamayor-Pujol, Donta & Dustdar (2023) arXiv:2311.10597; Sedlak et al. (2024) IEEE EDGE
Kirchhoff et al. (2018) J. R. Soc. Interface

DeepSLOs

- A construct we envision relating SLOs
- Provides a complete view of DCC system
- Allows aggregation towards higher abstractions

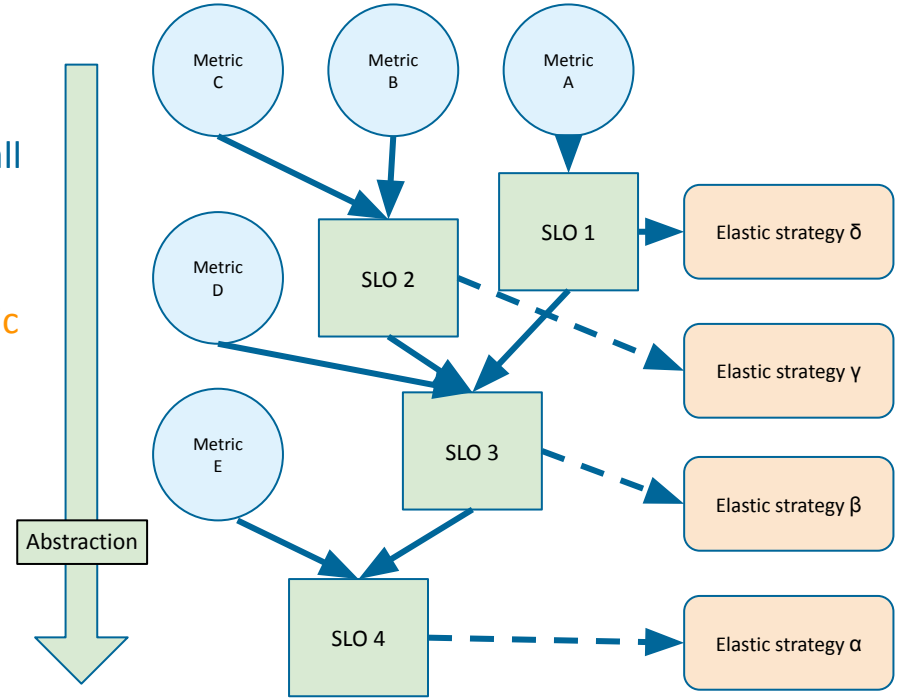


DeepSLOs

DeepSLOs as a hierarchically structured set of SLOs that relate causally and purposefully, holistically integrating all system needs.

1. A single DeepSLO can be in charge of an autonomic component of the system, providing ad-hoc objectives and elastic strategies at different abstraction levels, and mapping into the infrastructure.

2. Horizontal relations are within the same level of abstraction, vertical relations incorporate purpose and lead to different abstraction levels.



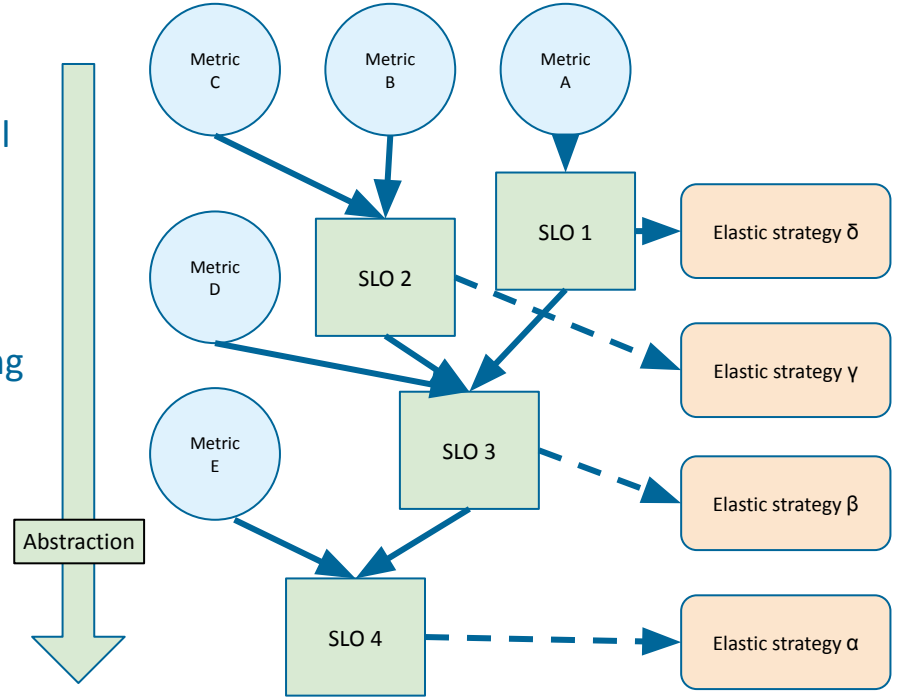
DeepSLOs

DeepSLOs as a hierarchically structured set of SLOs that relate causally and purposefully, holistically integrating all system needs.

3. A complete DCCS can be mapped with several DeepSLO that connect at their highest level, allowing each DeepSLO to properly propagate towards the infrastructure the shared objectives.

4. They provide a framework to solve the *multiple elasticity strategy problem*.

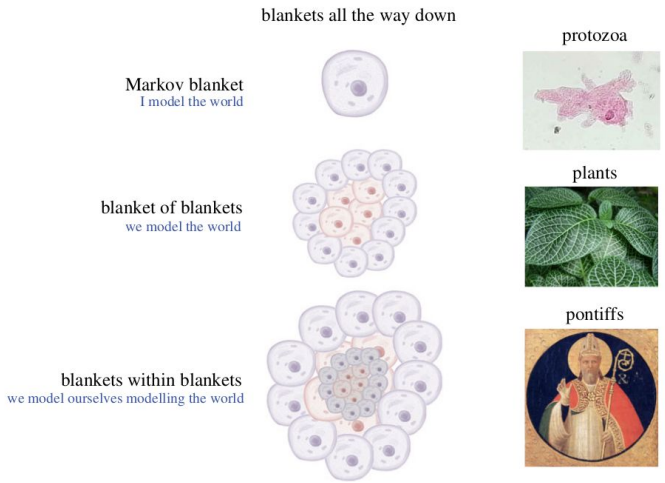
5. Integrate transversal features such as privacy, security, energy-efficiency, reliability…



Research Scope

Intersection between distributed service assurance and Active Inference:

- **Structural causal models**
  - **Causality** to tame large scale networks
  - Revealing and managing dependencies
- **Self-evidenced cellular structures**
  - Evaluate continuously how to fulfill SLOs
  - Based on empirical values (i.e., metrics)
- **Homeostasis – Equilibrium**



[3]

[3] Kirchhoff et al., The Markov blankets of life: autonomy, active inference and the FEP, https://doi.org/10.1098/rsif.2017.0792

Preliminary Work

- Local Requirements assurance by employing BN and MB [6] →

  “Static Bayesian Network Learning”

- Design Study for AIF agents in distributed systems [7]

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).

F. Monti et al. (Eds.): ICSOC 2023, LNCS 14419, pp. 42–50, 2023.
https://doi.org/10.1007/978-3-031-48421-6_4

Active Inference on the Edge: A Design Study*

Boris Sedlak, Victor Casamayor Pujol, Praveen Kumar Donta, and Schahram Dustdar
Distributed Systems Group, Vienna University of Technology (*TU Wien*), Vienna 1040, Austria.
Email: {b.sedlak, v.casamayor, pdonta, dustdar}@dsg.tuwien.ac.at

***Abstract*—Machine Learning (ML) is a common tool to interpret and predict the behavior of distributed computing systems, e.g., to optimize the task distribution between devices. As more and more data is created by Internet of Things (IoT) devices, data processing and ML training are carried out by edge devices in close proximity. To ensure Quality of Service (QoS) throughout these operations, systems are supervised and dynamically adapted with the help of ML. However, as long as ML models are not retrained, they fail to capture gradual shifts in the variable distribution, leading to an inaccurate view of the system state. Moreover, as the prediction accuracy decreases, the reporting device should actively resolve uncertainties to improve the model's precision. Such a level of self-determination could be provided by Active Inference (AIF) – a concept from neuroscience that describes how the brain constantly predicts and evaluates sensory information to decrease long-term surprise. We encompassed these concepts in a single action-perception cycle, which we implemented for distributed agents in a smart manufacturing use case. As a result, we showed how our AIF agent was able to quickly and traceably solve an optimization problem while fulfilling QoS requirements.**

***Index Terms*—Active Inference, Machine Learning, Edge Intelligence, Service Level Objectives, Markov Blanket**

I. Introduction

Recent years have reported a constant transition of logic —from the central cloud towards the edge of the network [3], thus, closer to the Internet of Things (IoT) devices that actually generate data. This transition includes the training of Machine Learning (ML) models (i.e., to save bandwidth and improve privacy), as well as data processing (i.e., to decrease latency) [2]. As soon as training has finished, ML models are a common measure to interpret and predict the behavior of distributed systems, e.g., to estimate the impact of redeployment [3] or forecast potential system failures [4], which must be circumvented to ensure the Quality of Service (QoS).

ML models are applied throughout the Computing Continuum (CC), i.e., from the cloud, over the fog, to the network edge – close to where models were trained. However, in many cases, ML models are not retrained, although new observations would be available [3], [4]; this inevitably leads to an inaccurate view of the system state, which, in turn, decreases the quality of any inference mechanism that uses the ML model. Imagine an elastic computing system, like envisioned in [3], [6], which observes the system through a set of metrics, evaluates whether QoS requirements – also called Service Level Objectives (SLOs) – were fulfilled, and dynamically reconfigures the system to ensure SLOs are met. If the variable distribution changes and the ML model is not adjusted, this makes it impossible to interpret the metrics correctly, and any consequential reconfiguration will fail to fulfill its purpose.

Ensuring the precision of an ML model requires a continuous feedback mechanism; such behavior could, for example, be achieved by optimizing a value function, as in reinforcement learning [7], [8]. However, we believe that this requires a more holistic approach, which starts with making the SLOs first-class citizens during the ML training process. Further, we want to highlight the responsibility of any service that uses the ML model to actively resolve or report ambiguities. Such a level of self-determination could be provided by Active Inference (ACI), a concept from neuroscience that describes how the brain constantly predicts and evaluates sensory information to decrease long-term surprise. ACI combines various concepts that have already been rudimentarily implemented in distributed systems, e.g., causal inference to identify dependencies between system parts [3], or dynamic adaptations of the system to ensure QoS – called homeostasis. This shows the potential of ACI.

In this paper, we advance one step further by combining the ACI concepts in a comprehensive design study of an ACI agent that optimizes the throughput of a smart factory. Internally, the agent follows an action-perception cycle: First, it estimates which parameter assignments would violate given SLOs, then it compares this expectation with new observations, and finally, it adjusts its beliefs (i.e., the ML model) accordingly. The agent focuses on exploring values that promise a high throughput while avoiding such that are likely to violate the SLOs. Furthermore, it favors solutions that are likely to improve the model precision, which, in turn, provides the agent with a clear understanding of the causal relations between model variables. Hence, the contributions of this article are the following:

- A novel ML paradigm based on ACI that continuously evaluates the quality of predictions. Thus, agents improve the model precision to ensure QoS for ongoing operations.
- The composite representation of an agent's behavior throughout the action-perception cycles. The distinct factors can be fine-tuned to determine the agent's preferences.
- A complete design study for a smart manufacturing agent that paves the way for other researchers to implement ACI in related automative use cases.

The remainder of the paper is structured as follows: Section II provides background information on ACI principles in distributed systems; in Section III, we present existing work that included ACI; within Section IV we outline the design process of an ACI agent, which we implemented and evaluated in Section V. Finally, Section VI concludes the paper.

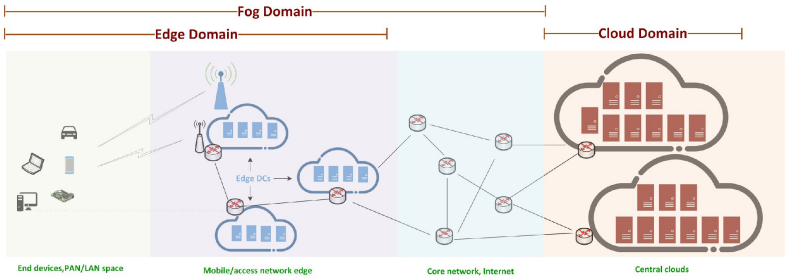
* Funded by the European Union (TEADAL, 101070186). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Distributed Intelligence in the Computing Continuum with Active Inference, Casamayor V., Sedlak, B., Salvatori, T., Friston, K., Dustdar, S., under review

[6] Designing Reconfigurable Intelligent Systems with Markov Blankets, ICSOC 2023, https://doi.org/10.1007/978-3-031-48421-6_4
[7] Active Inference on the Edge: A Design Study, pending at IEEE PerconAI 2024, https://doi.org/10.48550/arXiv.2311.10607

Equilibrium in the CC through Active Inference

- Core problem stems from **CC architecture**
- Impossible to centrally evaluate requirements
- Heterogeneity and context-dependence



- Requires components to operate **decentralized**
- Devices unaware of how to fulfill their SLOs
- Active Inference can provide this knowledge

Future Generation Computer Systems 160 (2024) 92–108

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Equilibrium in the Computing Continuum through Active Inference

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

Distributed Systems Group, TU Wien, 1040 Vienna, Austria

ARTICLE INFO

Keywords:
Active Inference
Computing Continuum
Scalability
Edge intelligence
Transfer learning
Equilibrium

ABSTRACT

Computing Continuum (CC) systems are challenged to ensure the intricate requirements of each computational tier. Given the system's scale, the Service Level Objectives (SLOs), which are expressed as these requirements, must be disaggregated into smaller parts that can be decentralized. We present our framework for collaborative edge intelligence, enabling individual edge devices to (1) develop a causal understanding of how to enforce their SLOs and (2) transfer knowledge to speed up the onboarding of heterogeneous devices. Through collaboration, they (3) increase the scope of SLO fulfillment. We implemented the framework and evaluated a use case in which a CC system is responsible for ensuring Quality of Service (QoS) and Quality of Experience (QoE) during video streaming. Our results showed that edge devices required only ten training rounds to ensure four SLOs; furthermore, the underlying causal structures were also rationally explainable. The addition of new types of devices can be done a posteriori; the framework allowed them to reuse existing models, even though the device type had been unknown. Finally, rebalancing the load within a device cluster allowed individual edge devices to recover their SLO compliance after a network failure from 22% to 89%.

1. Introduction

Computing Continuum (CC) systems, as envisioned in [1–3], are large-scale distributed systems composed of multiple computational tiers. Each tier serves a unique purpose, e.g., providing latency-sensitive services (i.e., Edge), or an abundance of virtual, scalable resources (i.e., Cloud). However, the requirements that each tier must fulfill are equally diverse, as they span a wide variety of edge devices and fog nodes. Assume that requirements would be ensured in the cloud, e.g., by analyzing metrics and reconfiguring individual devices, massive amounts of data would have to be transferred. Also, if edge devices fail to provide their service to a satisfying degree, the latency for detecting and resolving this would be high.

Given the scale of the CC, requirements must be decentralized; this means that the logic to evaluate requirements must be transferred to the component that they concern. Cloud-level requirements, i.e., Service and SLO fulfillment [5]. This promotes the usage of Active Inference (AIF) [6], an emerging concept from neuroscience that describes how the brain continuously predicts and evaluates sensory information to model real-world processes. By extending individual CC components with AIF, they could develop a causal understanding of how to adjust their environment to ensure preferences (i.e., SLOs).

Ensuring SLOs autonomously (i.e., evaluating the environment to infer adaptations) makes components intelligent [7]; any system composed entirely of such intelligent, self-contained components becomes more resilient and reliable. No central logic must be employed to ensure SLOs; thus, higher-level components can rely on the SLO fulfillment of underlying components. Ascending from intelligent edge devices, the next level would be intelligent fog nodes; those we see in the ideal position to orchestrate the service of edge devices. Thereby, edge devices in proximity are bundled into a device cluster, administered

[8] Equilibrium in the Computing Continuum through Active Inference, FGCS 160 (2024), 92-108

AIF as Meta-Intelligence: The Three-Tier Architecture

Active Inference does not compete with LLMs or specialized vision models — it governs them. The system operates as a three-tier hierarchy unified under free energy minimization.

Tier 1 — Meta-Intelligence (AIF)

Maintains global generative model. Monitors free energy across all subsystems. Decides WHEN to learn (epistemic drive) and WHAT actions to take (instrumental drive). Detects model degradation via prediction error dynamics.

Tier 2 — Cognitive Orchestration (LLM)

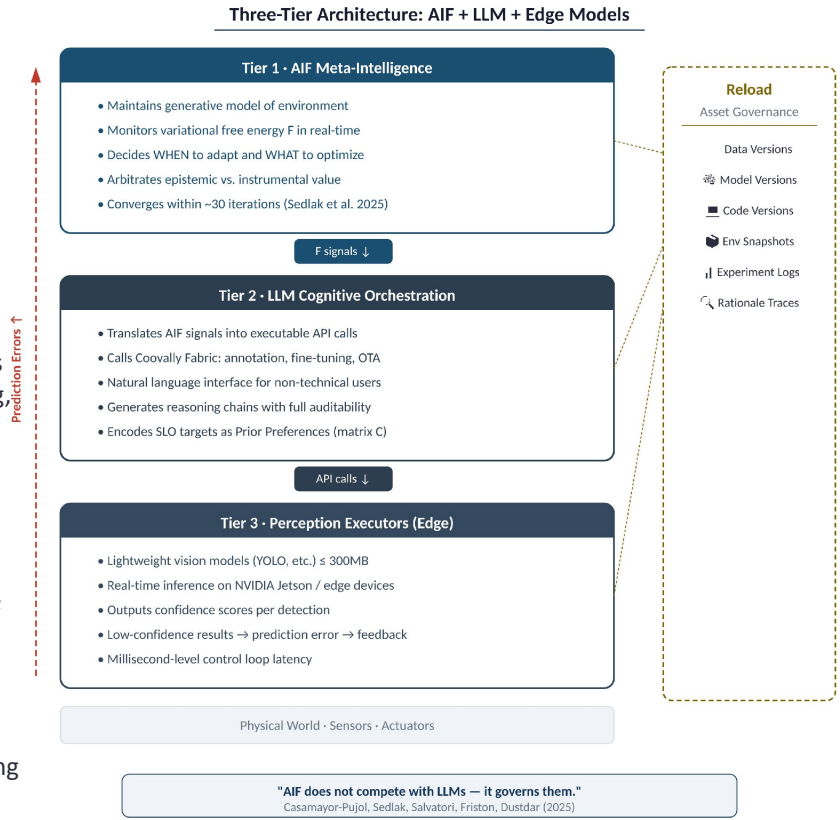
Reasoning and orchestration layer. Guided by AIF signals, translates high-level goals into executable workflows — calling Coovally Fabric APIs for annotation, fine-tuning, or deployment. Operates within AIF's prior preference constraints.

Tier 3 — Perception Executors (Small Models)

Specialized lightweight vision models (detection, segmentation, OCR, tracking) deployed on edge. Handle real-time perception. Confidence scores feed back to AIF as sensory states.

Asset Governance Layer

Captures all artifacts (data, code, model, environment versions) across every learning iteration. Enables audit, rollback, and reproducibility.



The Autonomous Learning Loop

The closed-loop self-learning cycle is driven entirely by AIF's free energy minimization — not by human intervention. This implements what Dustdar et al. term system "homeostasis" across the computing continuum.

Step 1 **Anomaly Detection via Free Energy**

Model confidence drops → prediction error in AIF sensory states → free energy spikes above steady state threshold.

Step 2 **Epistemic Action — Data Acquisition**

Epistemic drive flags low-confidence samples → routes to Coovally Fabric for automated pre-annotation via LLM + model ensemble.

Step 3 **Belief Revision — Model Update**

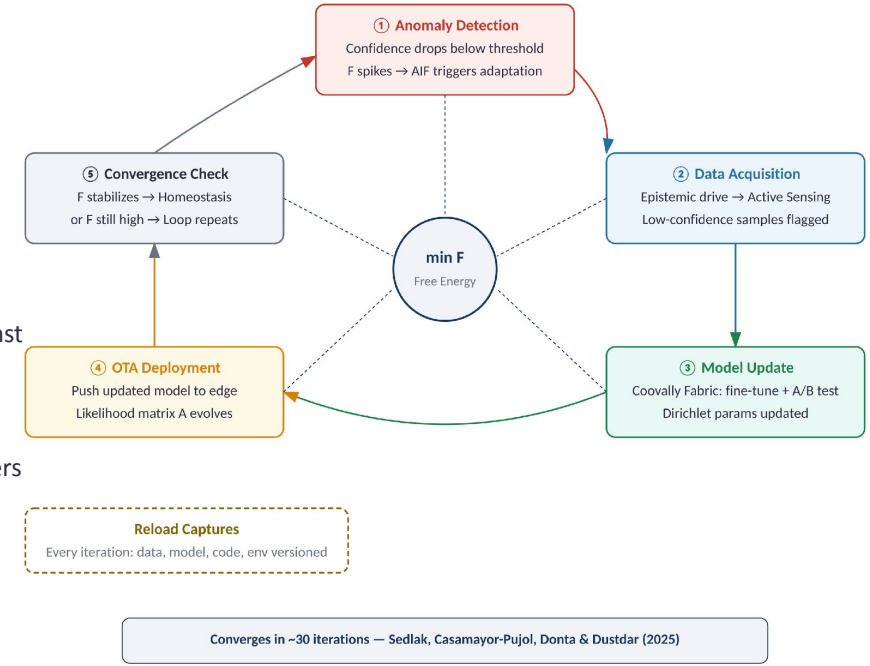
Coovally Fabric: fine-tuning on new data → A/B version testing against production model → model registry update.

Step 4 **Active State Update — OTA Deployment**

Validated model deployed via OTA. Dirichlet concentration parameters of likelihood matrix A evolve. Free energy decreases.

Step 5 **Convergence Monitoring**

AIF monitors free energy stabilization at new steady state. If not converged → loop repeats. Reload captures artifact lineage.



Dirichlet Continuous Learning

$$a_i^{post} = a_i^{prior} + \Sigma_\tau o_{i,\tau} \cdot s_\tau$$

Da Costa et al. (2020) J. Math. Psych.; Sedlak et al. (2025) Evolving Systems — AIF converges within 30 iterations

Discussion: Why Active Inference, Not Reinforcement Learning?

Dimension	Active Inference	Reinforcement Learning
Optimization Objective	Single quantity: free energy. Naturally balances exploration and exploitation.	Reward maximization. Requires separate exploration bonuses, curriculum design.
Causal Transparency	Generative model provides rationale traces. Every decision causally auditable.	Policy is opaque. Post-hoc explainability methods approximate at best.
Data Efficiency	Small-sample via Dirichlet-parameterized continuous belief updates.	Typically requires millions of environment interactions.
Environmental Drift	Prediction errors provide immediate, graded signals. Graceful degradation.	Often fails catastrophically before retraining is triggered.
Edge Feasibility	≤ 300 MB footprint. Single-step gradient planning.	Heavy compute for policy optimization. Often cloud-dependent.
Perception–Action Unity	Perception and action derived from same generative model.	Perception and control are separate modules with different objectives.

Core argument: AIF provides a ***first-principles*** framework — derived from physics (FEP), not from engineering heuristics. It does not need reward engineering because minimizing surprise *is* the reward. This makes it fundamentally suited to serve as meta-intelligence for autonomous systems.

Da Costa, Lanillos et al. (2022) Entropy — "How Active Inference Could Revolutionise Robotics"; Pio-Lopez et al. (2016) J. R. Soc. Interface

Part 1 - Conclusion

Three Core Contributions

I. AIF as Meta-Intelligence

A principled, first-principles framework that unifies perception, learning, and action for embodied AI under a single variational objective — free energy minimization.

II. Three-Tier Autonomous Architecture

AIF (meta-intelligence) + LLM (cognitive orchestration) + Coovally Fabric (technical execution) + Coovally Reload (asset governance) forming a complete self-evolving embodied system.

III. From Autonomy to True Independence

The system acts autonomously while maintaining full auditability, reproducibility, and rollback capability. Not just self-learning — self-governing.

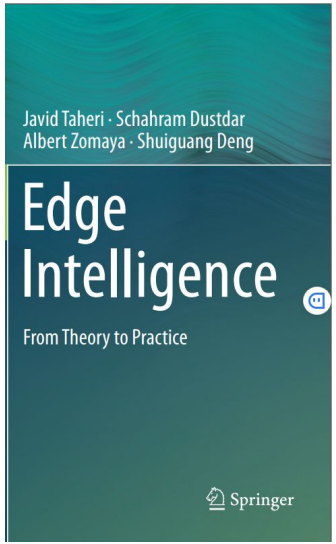
Future Directions

1. Multi-agent coordination — multiple embodied agents sharing generative models via nested Markov blankets
2. Hierarchical generative models — scaling to increasingly complex, multi-domain environments
3. Foundation model integration — cross-domain transfer learning governed by AIF meta-objectives

---

Active Inference transforms embodied AI from a collection of tools into a living system governed by the same principles that govern biological intelligence — the minimization of surprise.

Thanks for your attention



Prof. Schahram Dustdar

IEEE Fellow | EAI Fellow | I2CIC Fellow | AAIA Fellow

Member *Academia Europaea*

President of the AIIA (Asia-Pacific Artificial Intelligence Association)

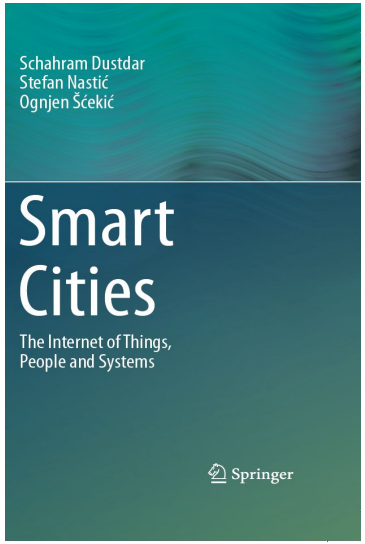
ACM Distinguished Scientist | ACM Distinguished Speaker

TCI Distinguished Service Award by the IEEE Technical Committee on the Internet (TCI)

IEEE TCSVC Outstanding Leadership Award in Services Computing

IEEE TCSC Award for Excellence in Scalable Computing

IBM Faculty award



Distributed Intelligence in the Computing Continuum – Part 2

Boris Sedlak, Víctor Casamayor Pujol, and Schahram Dustdar

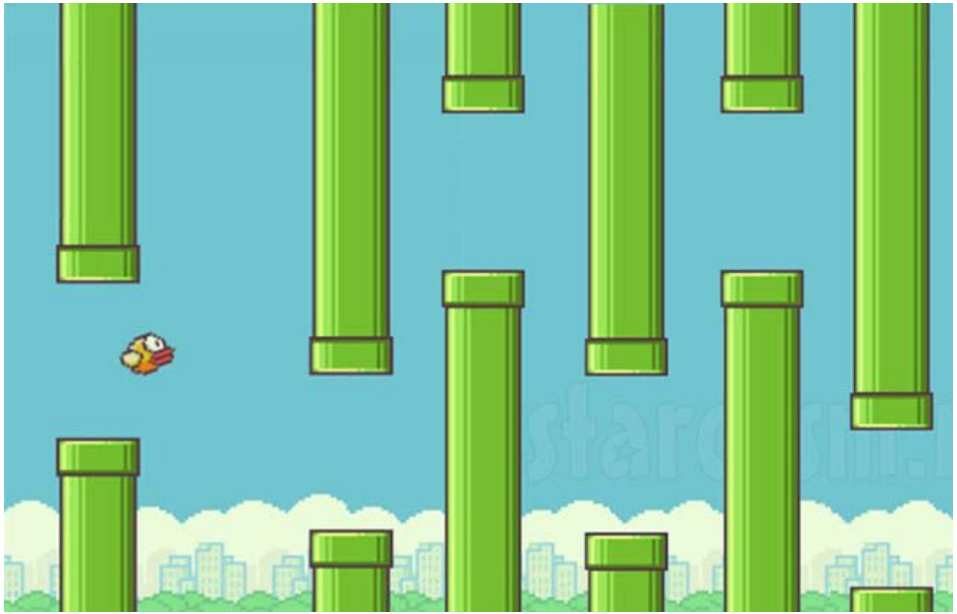
Session Outline

Theoretical parts (10 min)
- Motivational use case
- High-level architecture

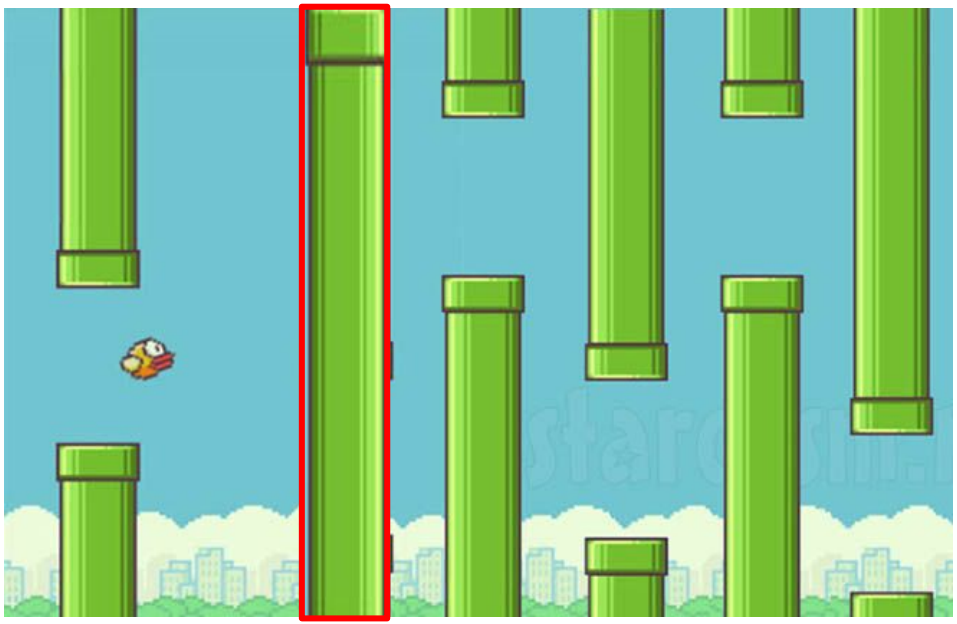
Coffee Break (15 min)

Practical parts (30 min)
- Four coherent notebooks
- Demo app & animation

Future extensions of this work (5 min)

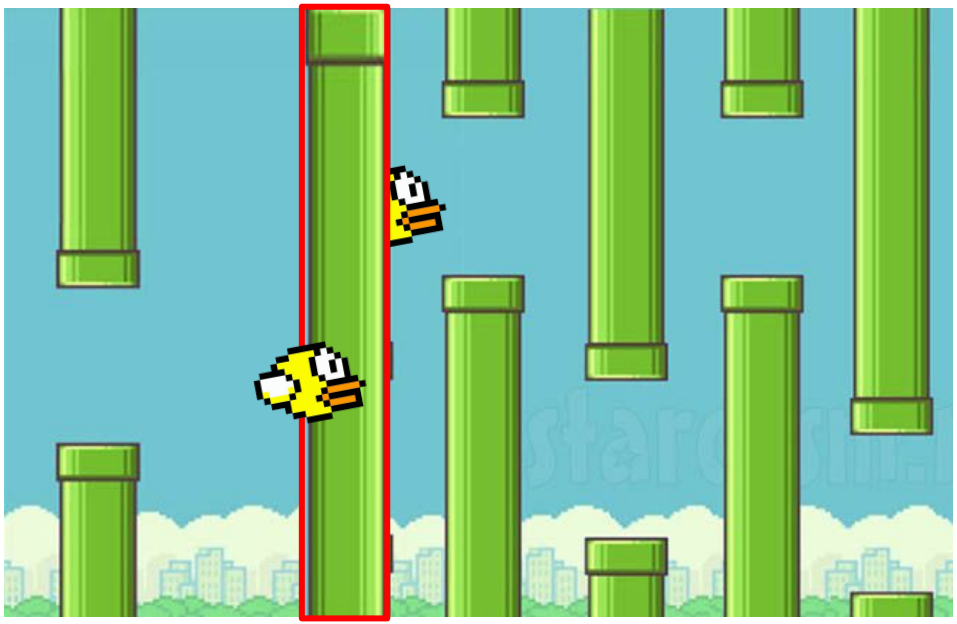


Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)



Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)

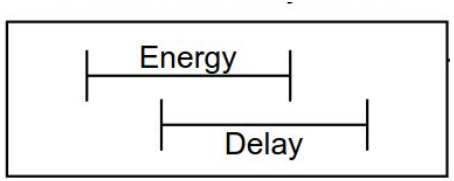
Can’t pass obstacle?



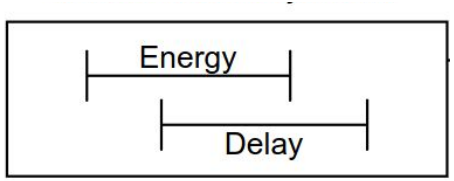
Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)

Can’t pass obstacle?

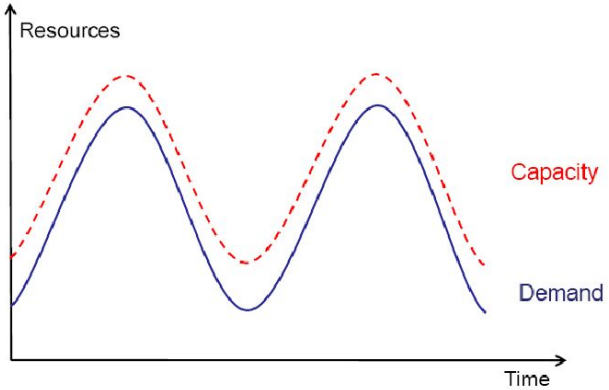
Use another dimension to improve flexibility of agent (i.e., action space)



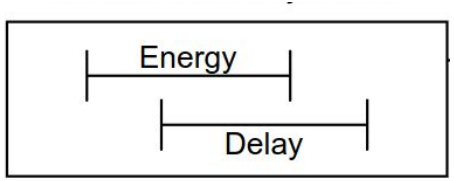
want to ensure **requirements** (= goals) of systems, e.g., latency or energy use



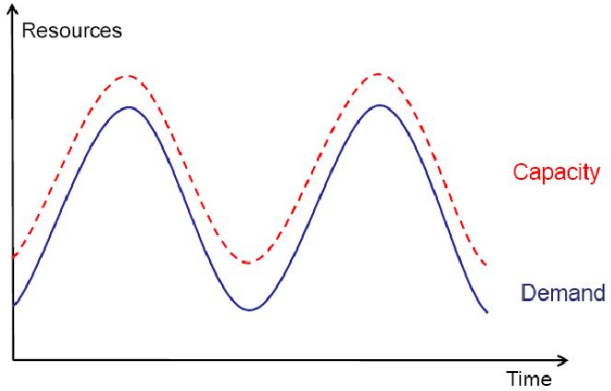
want to ensure **requirements** (= goals) of systems, e.g., latency or energy use



dynamic resource allocation according to current **demand**, e.g., submission time

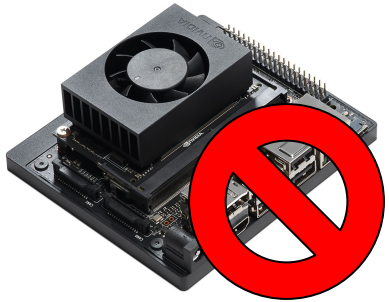


want to ensure **requirements** (= goals) of systems, e.g., latency or energy use



dynamic resource allocation according to current **demand**, e.g., submission time

works well with infinite resources



doesn't work under resource limits

Structure of Tutorial Contents

We will address two fundamental research questions:

- How to ensure service performance under **finite resources**?
- How to build **robust** decisions models that adapt **over time**?

We focus on two separate parts in our architecture:

- Infrastructure and interfaces for running, monitoring, and reconfigure processing services **during runtime**.
- Logic for building controlling agents that connect to these interfaces for optimizing **service performance**.

We created four tutorial notebooks that show these activities:

- How an agent can gradually learn how to optimize services.
- How multiple elasticity dimensions help under finite resources.

Structure of Tutorial Contents

We will address two fundamental research questions:

- How to ensure service performance under **finite resources**?
- How to build **robust** decisions models that adapt **over time**?

We focus on two separate parts in our architecture:

- Infrastructure and interfaces for running, monitoring, and reconfigure processing services **during runtime**.
- Logic for building controlling agents that connect to these interfaces for optimizing **service performance**.

We created four tutorial notebooks that show these activities:

- How an agent can gradually learn how to optimize services.
- How multiple elasticity dimensions help under finite resources.

Structure of Tutorial Contents

We will address two fundamental research questions:

- How to ensure service performance under **finite resources**?
- How to build **robust** decisions models that adapt **over time**?

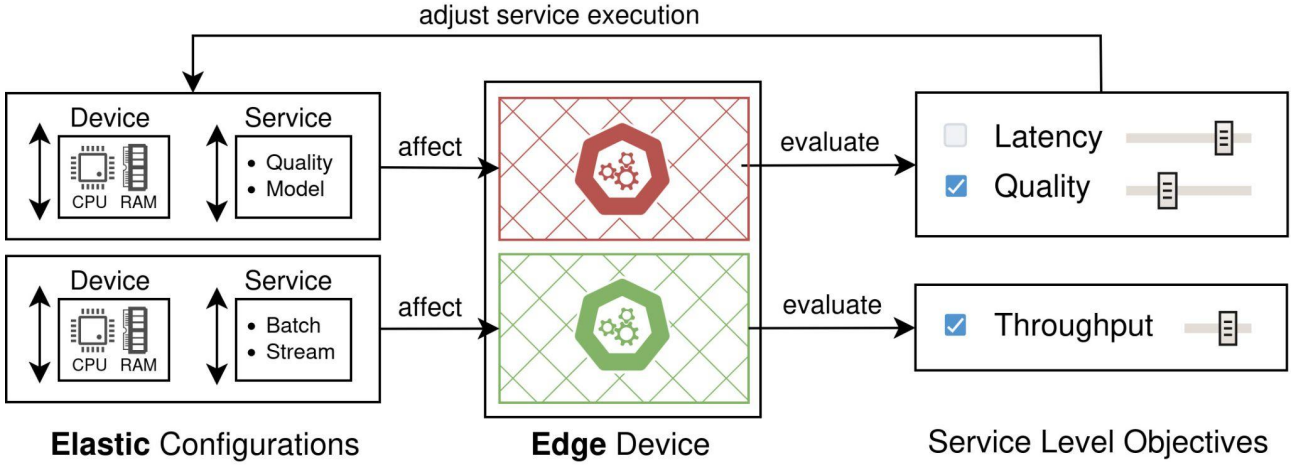
We focus on two separate parts in our architecture:

- Infrastructure and interfaces for running, monitoring, and reconfigure processing services **during runtime**.
- Logic for building controlling agents that connect to these interfaces for optimizing **service performance**.

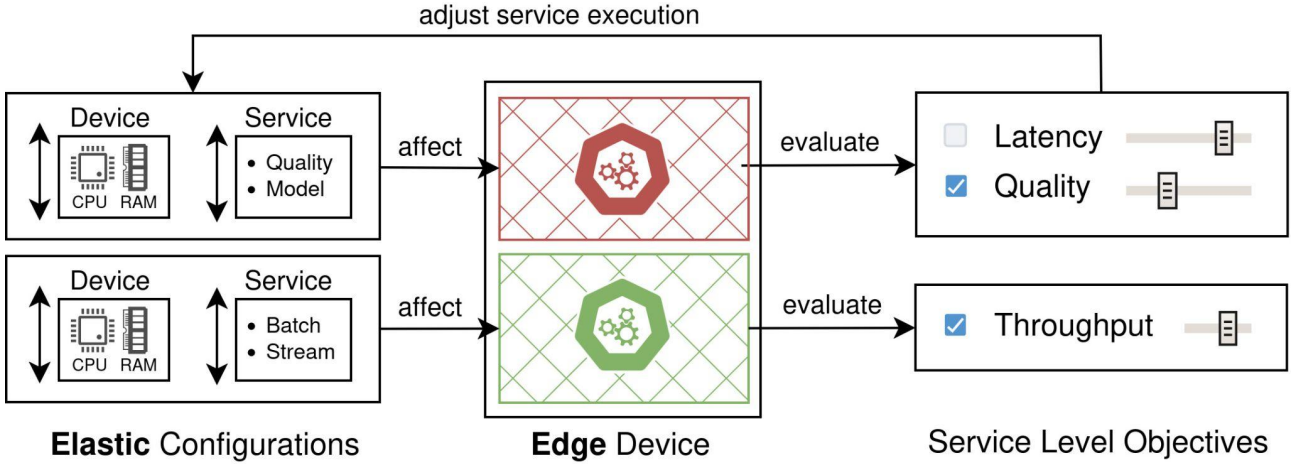
We created four tutorial notebooks that show these activities:

- How an agent can gradually learn how to optimize services.
- How multiple elasticity dimensions help under finite resources.

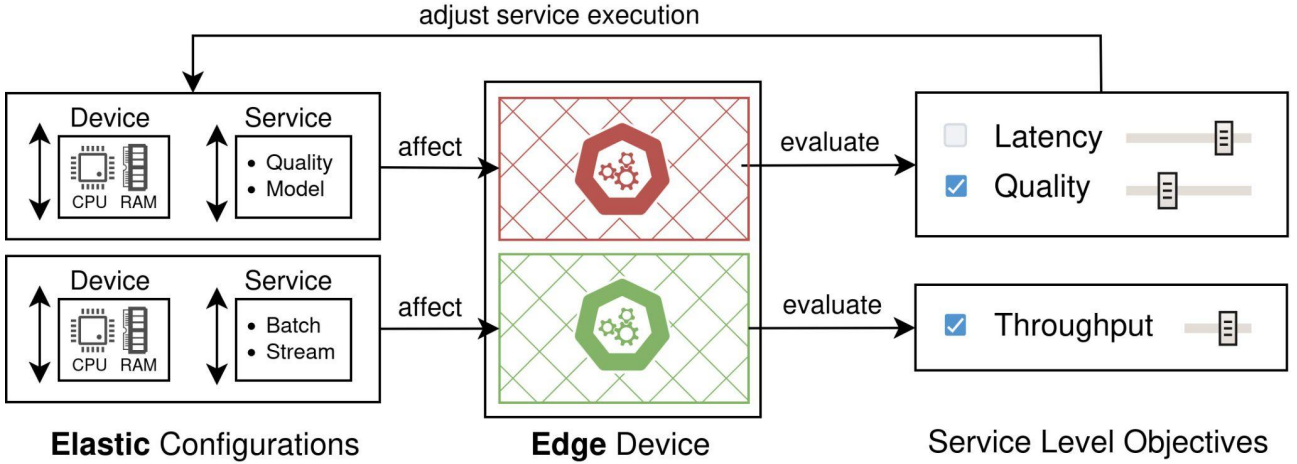
Infrastructure & Interfaces: High-Level View



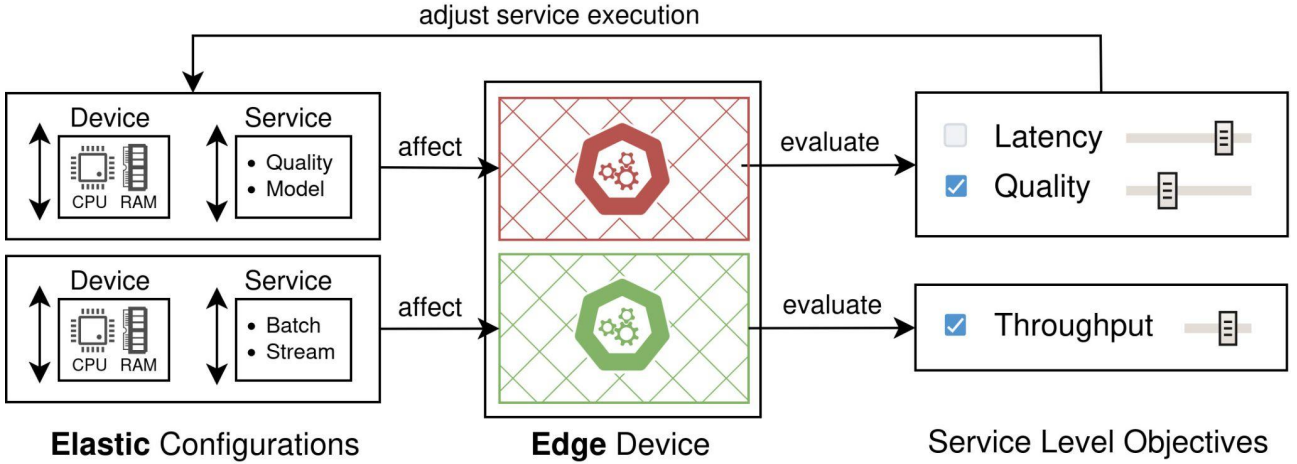
Infrastructure & Interfaces: High-Level View



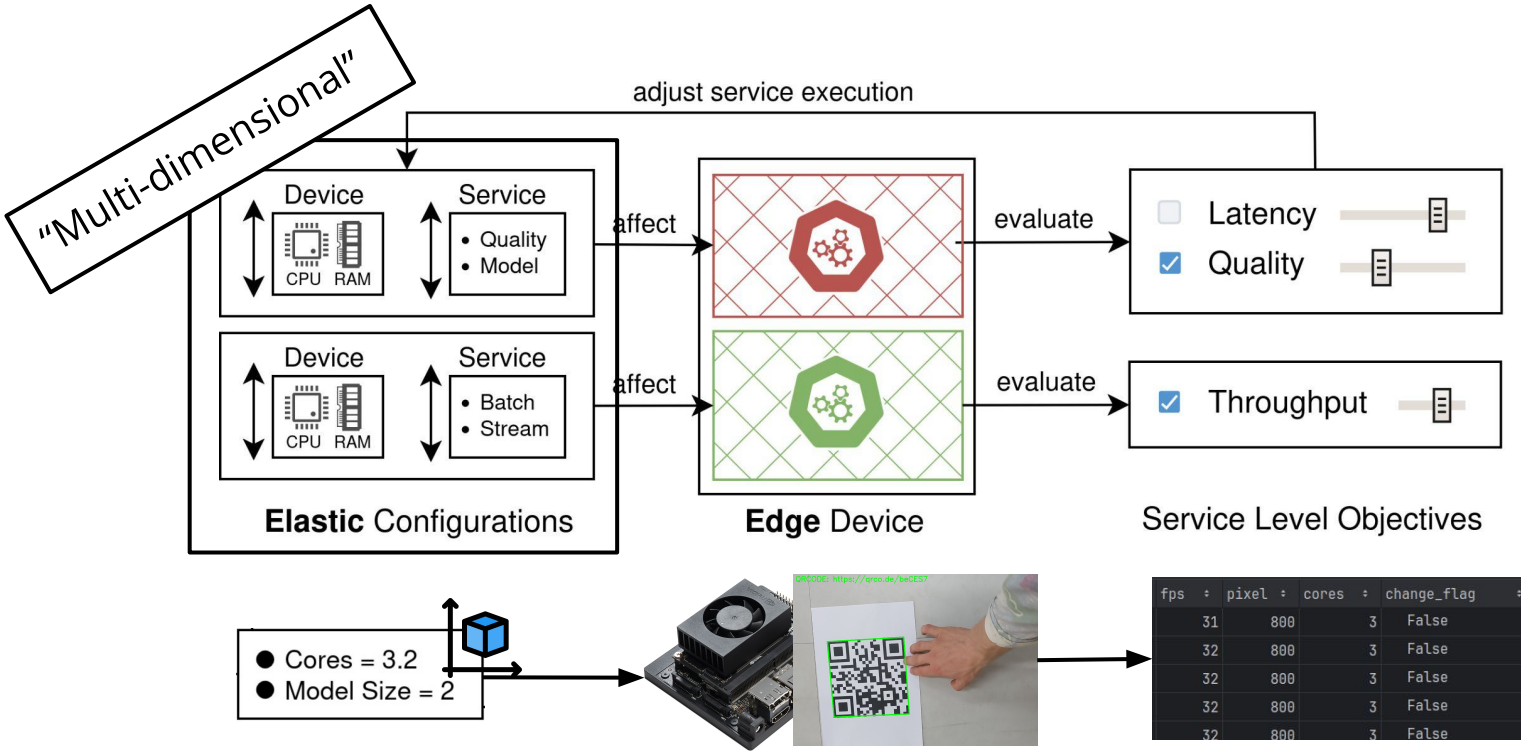
Infrastructure & Interfaces: High-Level View



Infrastructure & Interfaces: High-Level View



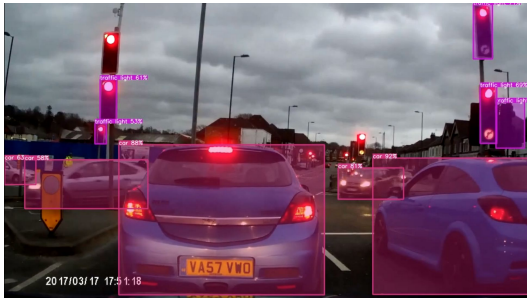
Infrastructure & Interfaces: High-Level View



Stream Processing Use Cases

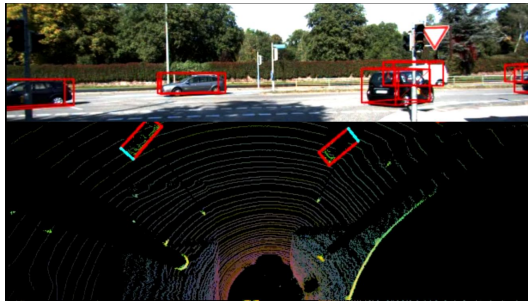
Commonly addressed use cases revolve around continuous **stream processing**; in case **time-critical** adaptations are required, this poses a higher need for sophisticated adaptation mechanisms.

Video Processing (Yolo V8)



Object detection in a video stream using Yolo [6]

Mobile Mapping (Lidar)



Creating a mobile map from binaries using Lidar [6]

QR Scanner (OpenCV)

QR code scanning in a video using OpenCV [6]

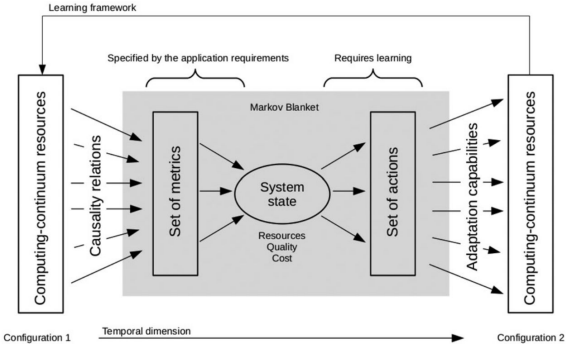
[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference,** in Springer Evolving Systems (2025)

II – Markov Blanket (MB)

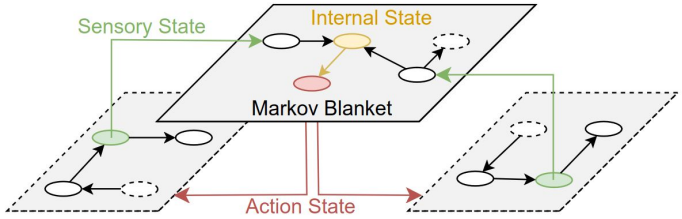
Interactions between **systems** (e.g., human in world) can be expressed through MBs; fulfill Markov property – so decisions can be taken based on system's current state

Creates **formal boundary** between a system and external states; within MB, discard all variables that show no impact on internal states and requirements (i.e., SLO fulfillment)

Provides clear interfaces for **sensory** and action **states**; policy (e.g., scaling) as a mapping between these states



Behavioral Markov blanket of a system [3]



Action-perception cycle between multiple entities [4]

[3] Dustdar et al., **On Distributed Computing Continuum Systems** (2023)
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services Edge 2024

II – Active Inference Architecture

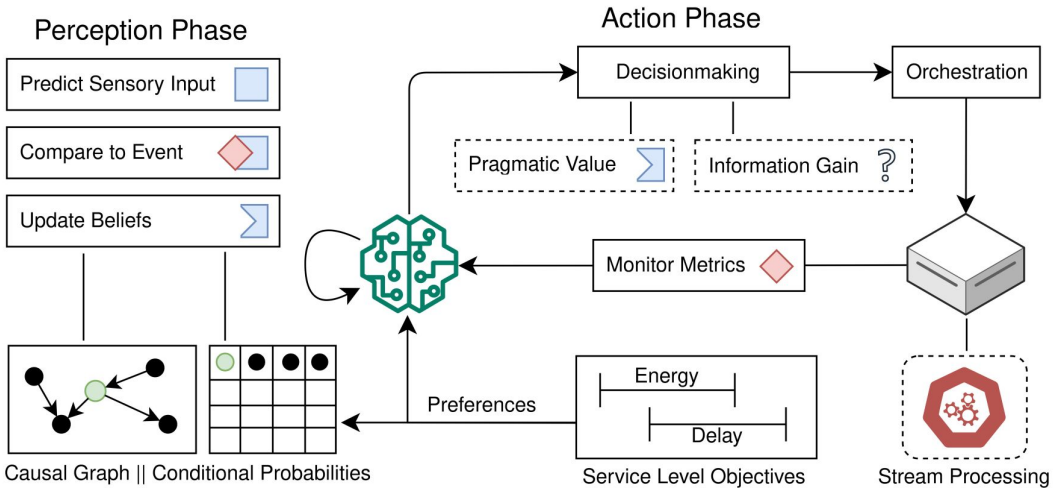
Approach

(1) **Specify** ideal runtime behavior through SLOs

(2) **AIF agents** monitor their environment & collect metrics

(3) **Perception phase** predicts expected SLO fulfillment and adjusts the generative model

(4) **Action phase** orchestrates the processing environment to optimize both SLOs and model



Action and perception cycles performed by the AIF agent to create an accurate model and shape the world [6]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

copied

II – Active Inference Architecture (cont.)

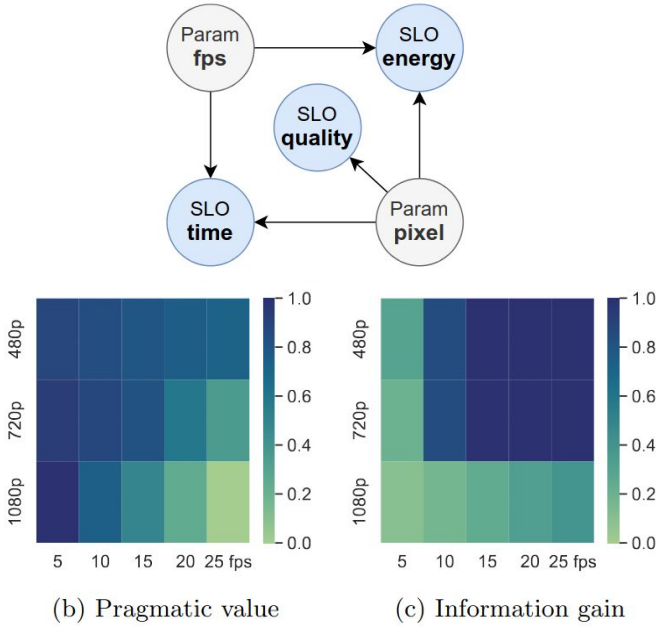
Interpretable behavior [6]

– Empirically verify variable relations in the BNs, e.g., increasing quality (pixel) leads to high energy usage; adjust **parameters** (i.e., pixel & fps) according to SLOs

– Quantified preferences of the agent: (1) expected SLO fulfillment or (2) potential model improvement; determine the behavior of the scaling agent

Continuous composition [4]

– Gradually create increasingly accurate models for individual processing services; continuously compose to estimate the **impact** they have on each other



(b) Pragmatic value (c) Information gain

[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

II – Active Inference Architecture (cont.)

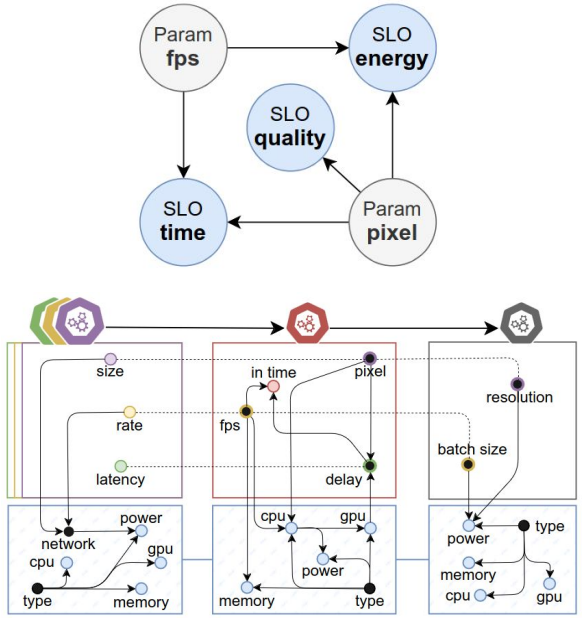
Interpretable behavior [6]

– Empirically verify variable relations in the BNs, e.g., increasing quality (pixel) leads to high energy usage; adjust **parameters** (i.e., pixel & fps) according to SLOs

– Quantified preferences of the agent: (1) expected SLO fulfillment or (2) potential model improvement; determine the behavior of the scaling agent

Continuous composition [4]

– Gradually create increasingly accurate models for individual processing services; continuously compose to estimate the **impact** they have on each other



[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

II – Active Inference Architecture (cont.)

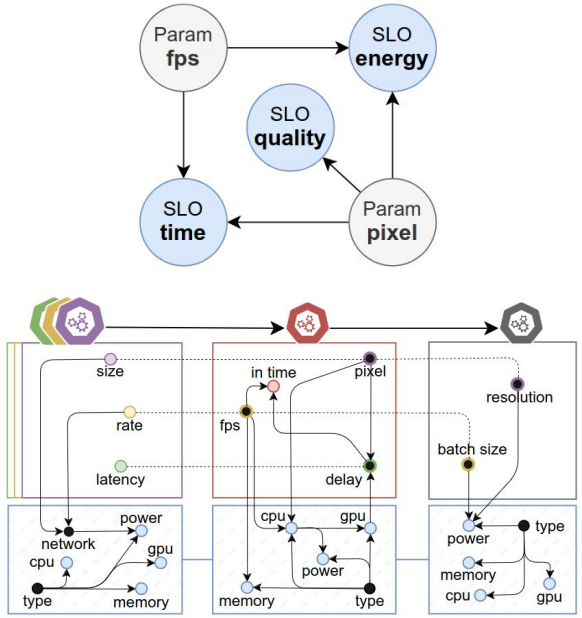
Interpretable behavior [6]

– Empirically verify variable relations in the BNs, e.g., increasing quality (pixel) leads to high energy usage; adjust **parameters** (i.e., pixel & fps) according to SLOs

– Quantified preferences of the agent: (1) expected SLO fulfillment or (2) potential model improvement; determine the behavior of the scaling agent

Continuous composition [4]

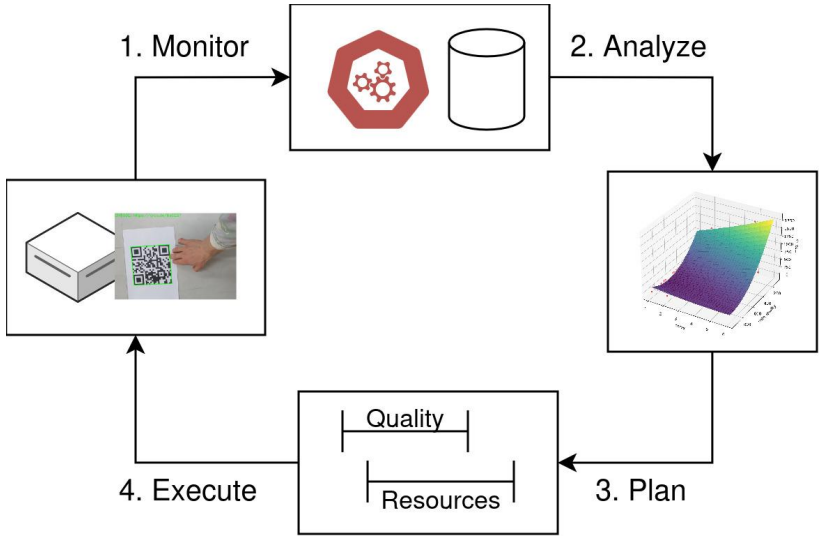
– Gradually create increasingly accurate models for individual processing services; continuously compose to estimate the **impact** they have on each other



[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

Demo: Notebooks

https://github.com/borissedlak/composable-autonomous-offerings/tree/main/notebooks/summersoc



The Github Repo contains four notebooks that describe a traditional adaptive cycle of four phases: Monitor, Analyze, Plan, and Execute

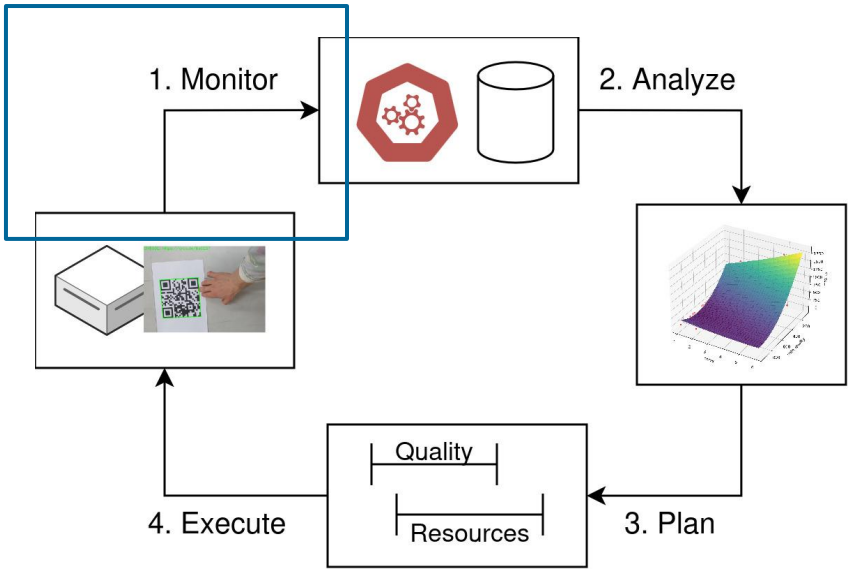
Demo: Notebooks

15 min coffee break

Demo: Notebooks

"trace what is happening inside the processing environment"

https://github.com/borissedlak/composable-autonomous-offerings/tree/main/notebooks/summersoc

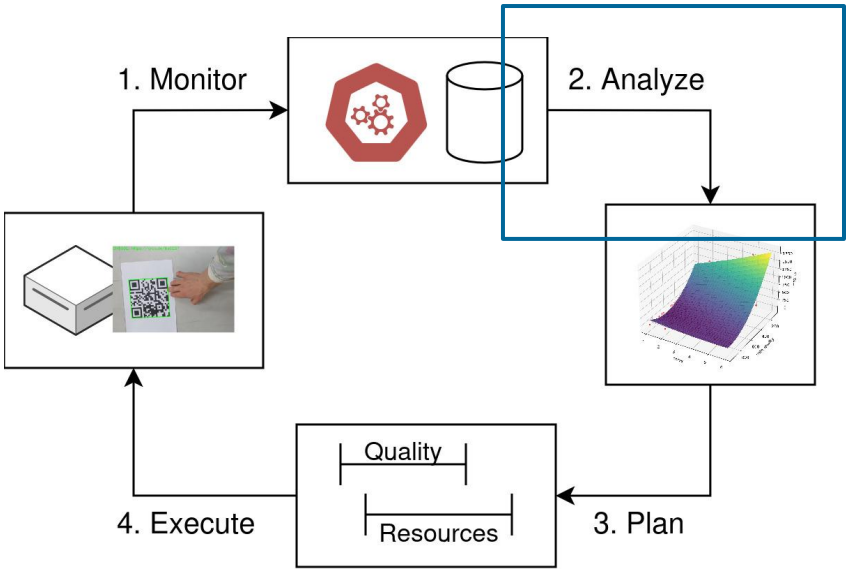


The Github Repo contains four notebooks that describe a traditional adaptive cycle of four phases: Monitor, Analyze, Plan, and Execute

Demo: Notebooks

"understand the current system state and how it was created"

https://github.com/borissedlak/composable-autonomous-offerings/tree/main/notebooks/summersoc

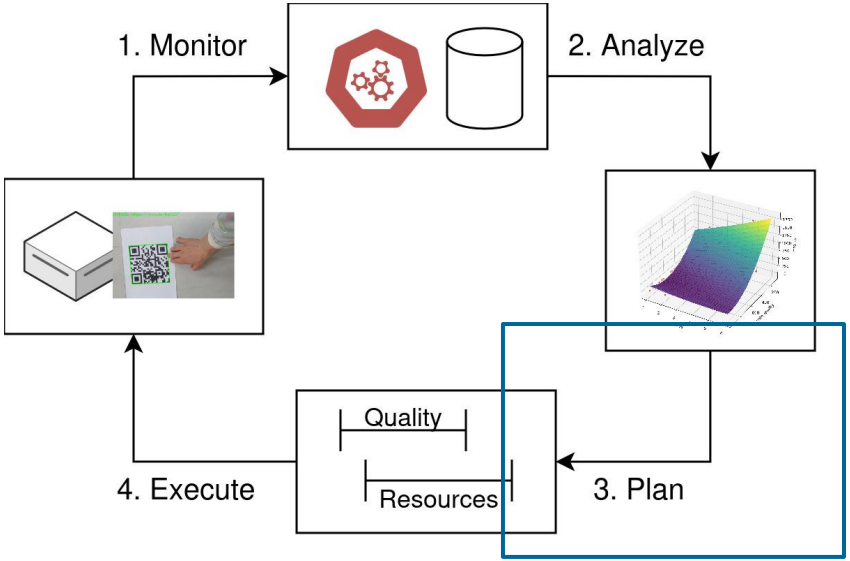


The Github Repo contains four notebooks that describe a traditional adaptive cycle of four phases: Monitor, Analyze, Plan, and Execute

Demo: Notebooks

"infer corrective actions that should restore the desired state"

https://github.com/borissedlak/composable-autonomous-offerings/tree/main/notebooks/summersoc

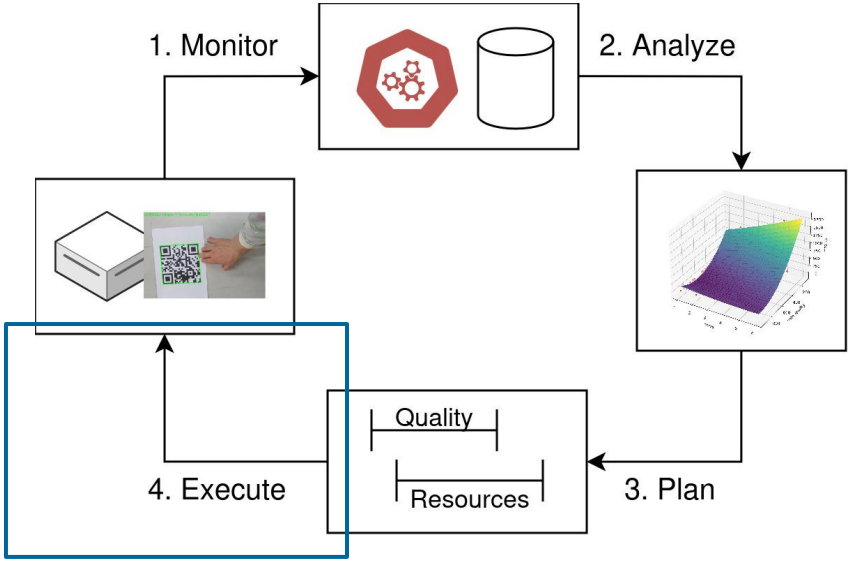


The Github Repo contains four notebooks that describe a traditional adaptive cycle of four phases: Monitor, Analyze, Plan, and Execute

Demo: Notebooks

"close the loop by applying these changes to the environment"

https://github.com/borissedlak/composable-autonomous-offerings/tree/main/notebooks/summersoc



The Github Repo contains four notebooks that describe a traditional adaptive cycle of four phases: Monitor, Analyze, Plan, and Execute

Demo: Notebooks (Summary)

Monitor (Prometheus)

We are able to capture and consume the current states across distributed processing devices and services.

Analyze (Regression / Gaussian process)

We use structural knowledge to interpret *why* a service is behaving as is is; {model size, resources, data quality } → performance.

Plan (Numerical solver)

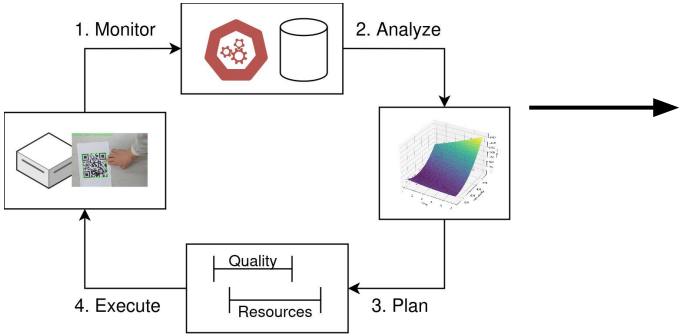
We infer a set of actions that will move the system to the desired state, or simply serve for exploring the environment.

Execute (REST API)

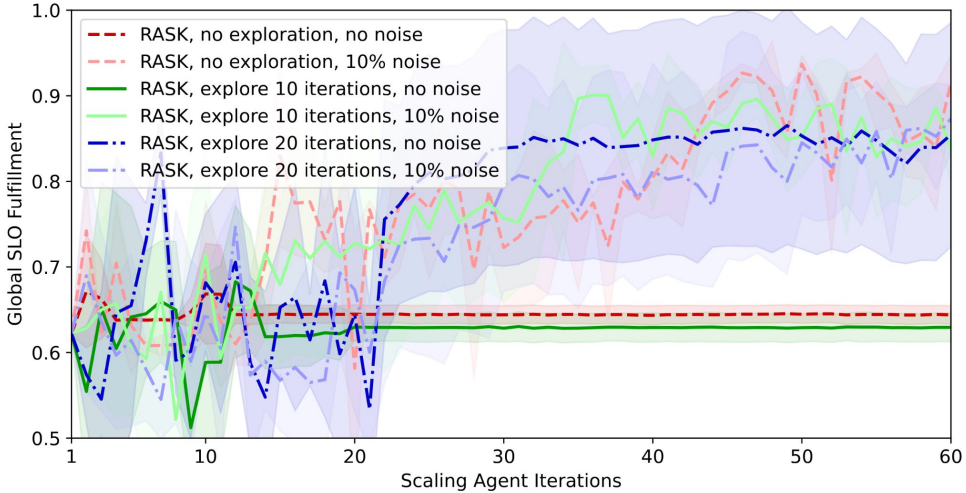
We call the service and device endpoints to run the actions.

Complementary Analysis: Role of Exploration

"How to arrive at an ample general knowledge of the environment?"



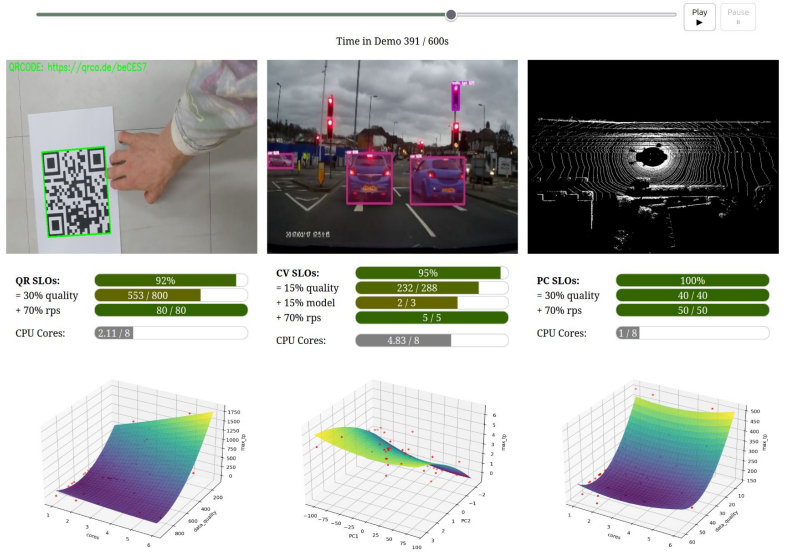
Explore: uniformly sampling new actions
Vary: Add Gaussian noise to the inferred actions



Sedlak et al., **Multi-Dimensional Autoscaling of Stream Processing Services on Edge Devices**, under revision at IEEE Transaction on Service Computing

Demo: Animation

https://borissedlak.github.io/percom-demo-2026/



The link contains a demo application that summarizes the tutorial contents: the development of the regression models and the inference

Sedlak et al., **Visual Insights into Agentic Optimization of Pervasive Stream Processing Services**, at IEEE Percom Demo (2026)

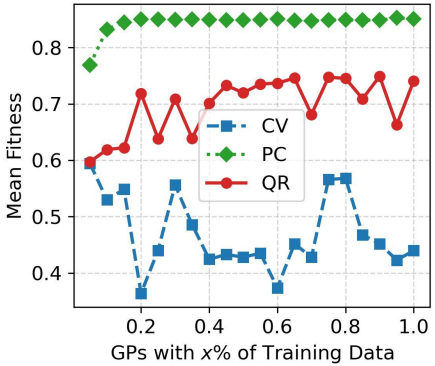
Demo: Animation (Summary)

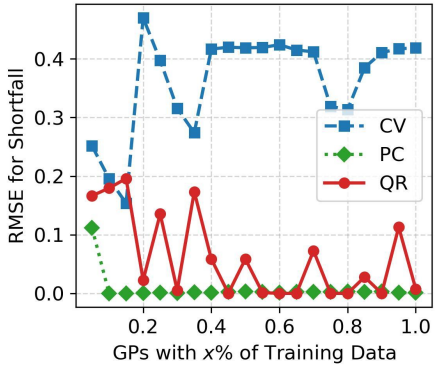
Scaling agent was able to optimize the performance of three processing services that are competing for **limited resources** on and Edge device.

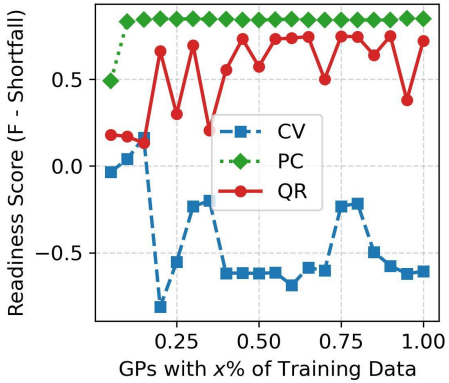
Methodology is extremely **sample-efficient**: 30 iterations in the environment allowed to have a highly **interpretable** model of the system dynamics.

Training the models can be resource-intense (Gaussian process), but inference is always simplified through the structural knowledge

Complementary Analysis: Prediction Accuracy



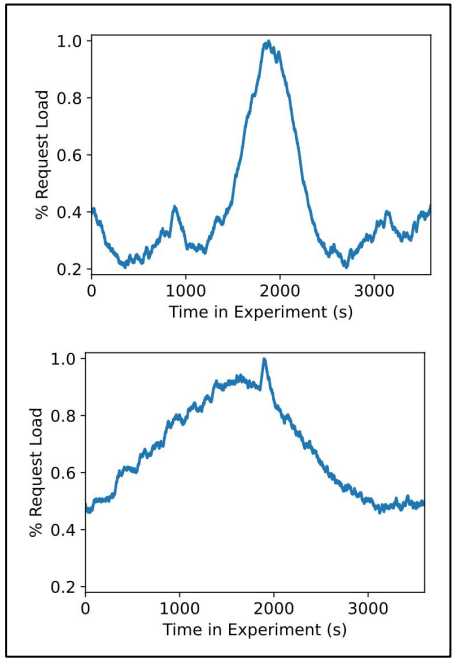




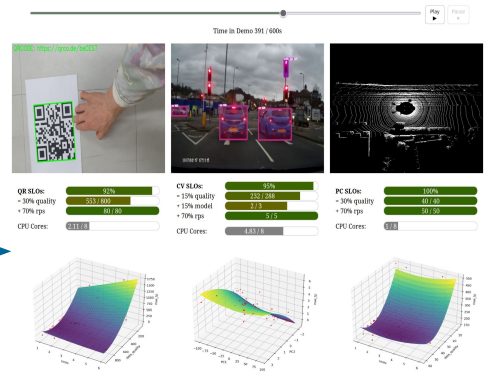
Sedlak et al., **Building Probabilistic Performance Guarantees for Composable Computing Services**, under submission

Complementary Analysis: Multi-dimensional Scaling

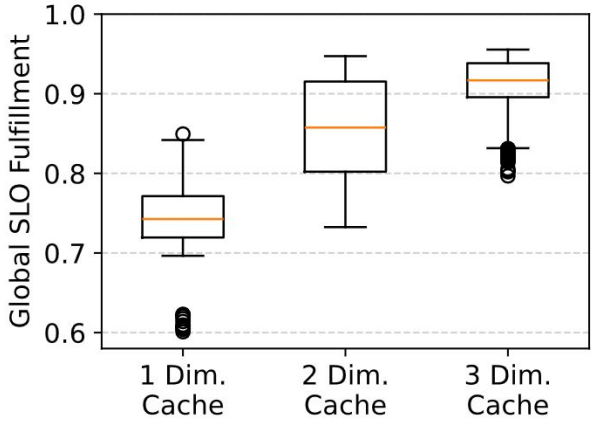
Workload pattern



Autoscaling platform



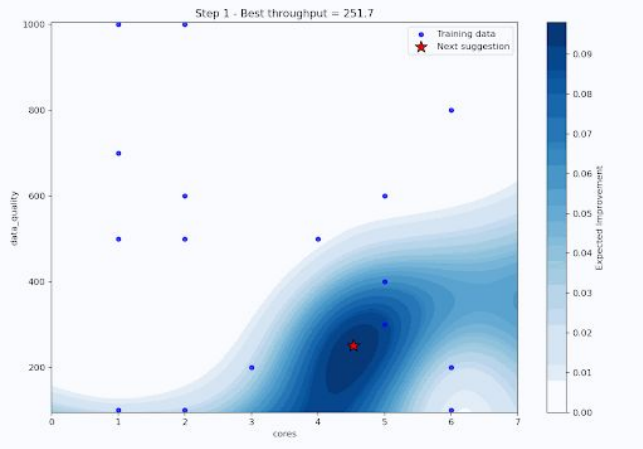
System Performance



Sedlak et al., **Multi-Dimensional Autoscaling of Stream Processing Services on Edge Devices**, under minor revision at IEEE Transaction on Service Computing

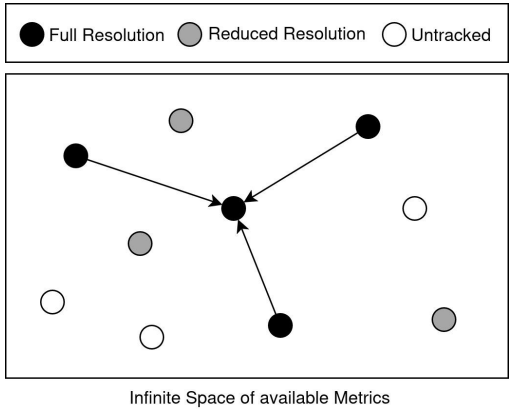
Future Work: Upcoming Extensions

Efficient Model Exploration



Allows to efficiently explore the solution space to kickstart service performance (e.g, composition).

Dynamic Instrumentation



Identify new metrics and actions during application runtime; not trivial, but essential for evolving applications.

Conclusion

Motivation: Large-scale computing systems pose infinity of optimization problems; must explore behavior during runtime due changing variable distributions.

Solution: Model processing environment through POMDP and train state transition models; compare four agents.

Benefit: Create stable autoscaling policies; embed AIF agents into common use cases and allow comparison with contemporary ML approaches (e.g., DQN).

Future work: sophisticated exploration schemes for continuous variables built on Gaussian processes.