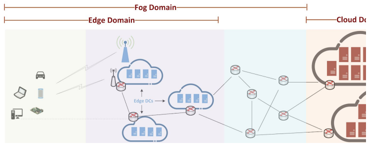
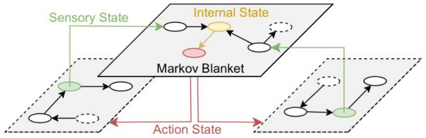
Operating Distributed Computing Continuum Systems through Active Inference

Boris Sedlak (TUW)
Advisor: Prof. Schahram Dustdar

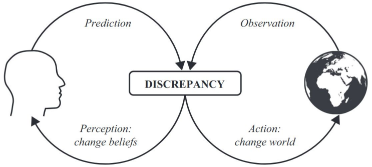
Agenda



Introduction



Dynamic service adaptation in the computing continuum



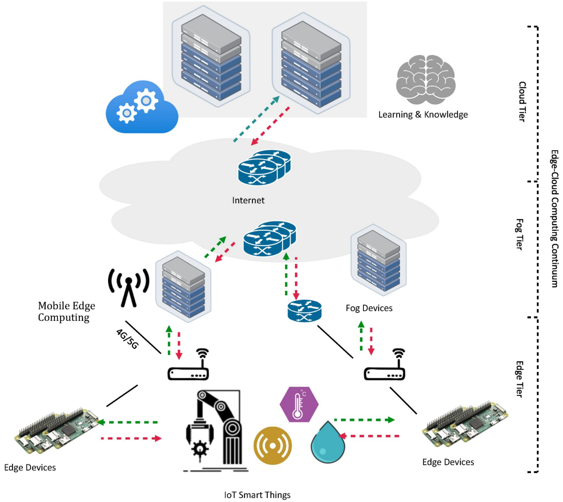
Learning through active inference

I – Computing Continuum

Computing Continuum (CC) as a composition of multiple processing tiers that stretch from IoT and edge computing, over Fog resources, to distant Cloud centers

Combines the benefits of all its tiers, i.e., low-latency and privacy-protecting computation from Edge, high availability and virtually unlimited processing resources from Cloud

Smart Cities are a common instance of distributed systems, where interconnected services (e.g., traffic surveillance or road surveillance) collaborate based on collected sensor data



Example of a behavioral model for data gravity [1]

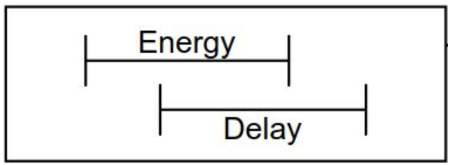
[1] Donta et al., **Exploring the Potential of Distributed Computing Continuum Systems** (2023)

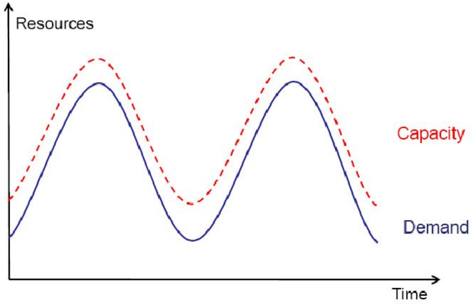
I – Service Level Objectives

Service Level Objectives (SLOs) specify requirements that must be ensured throughout operation (e.g., traffic routing latency < *t*). Usually consist of one or two thresholds

Elasticity Strategies [4] are countermeasures to scale a system (i.e., resources, quality, cost) according to current demand; whenever SLOs are violated, these can be used as an answer

Originates from cloud computing, which uses Service Level Agreements (SLAs) to guarantee a service to clients, e.g., provider has to pay a penalty if not available for > 99.9%





Elasticity allocates the right amount of resources [5]

[4] Dustdar et al., **Principles of Elastic Processes** (2011)
[5] Ricciardi et al., **Saving Energy in Data Center Infrastructures** (2011)

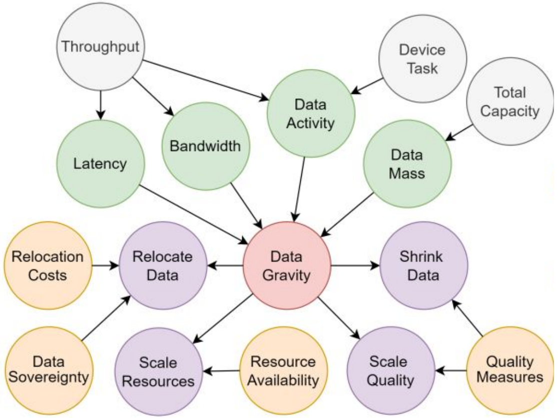
I – Complex SLOs

Elastic requirements assurance generally limited to **single metrics** and elasticity strategies, e.g., scaling container size if load is high; constrained by **limited resource** on the Edge – not really elastic

Requires a complex **behavioral** model that expresses how to counter environmental impacts; must consider hardware **heterogeneities** and device context when choosing strategies

Behavioral model

Internal state (●) contains objectives and how these relate to external sensory inputs (●); can interact with the world through action, i.e., elasticity strategies (●), which are influenced by contextual factors (●)



Example of a behavioral model for data gravity [6]

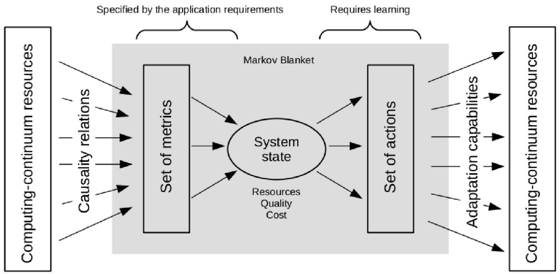
[6] Sedlak et al., **Controlling Data Gravity and Data Friction: From Metrics to Multidimensional Elasticity Strategies** (2023)

I – Computing Continuum (2)

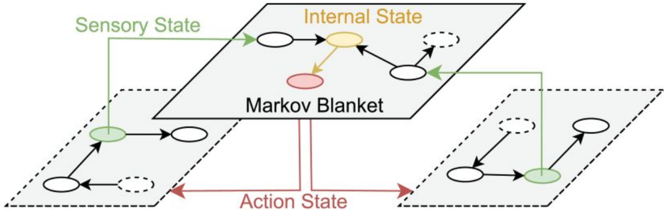
Problem
Large amount of sensors and input sources, how to find (and limit the system to) the ones that have the highest impact on the requirements fulfillment

Example
* What conditions have a causal influence on the frequency of accidents? Adjust speed limit

Our approach
Reductive probabilistic modelling through **Bayesian networks** and their **Markov blankets**



Behavioral Markov blanket for a system [2]



Action-perception cycle between multiple entities [3]

[2] Dustdar et al., **On Distributed Computing Continuum Systems** (2023);
[3] Sedlak et al., **Markov Blanket Composition of SLOs** (2024)

Skipped in main presentation

I – Complex SLOs (2)

Problem

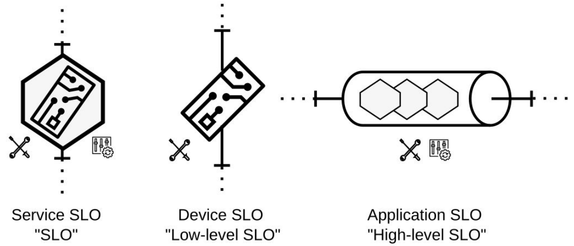
How can you constrain different levels of the system (i.e., low-level computation and high-level orchestrations) in one **cohesive** and **manageable** framework?

Example

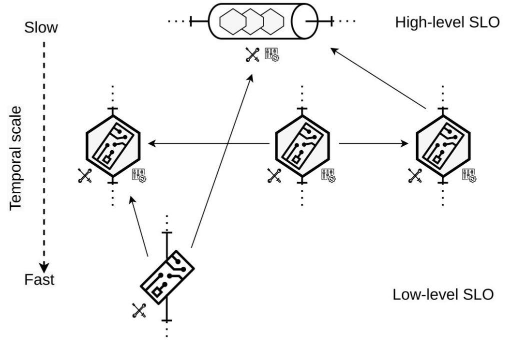
* Which components impact the processing latency of the application? Maintain and scale them more closely

Our approach

Create a network of **DeepSLOs** that constrain different aspects of the application; high-level goals can be ensured by chopping them up into lower-level goals



Different Instances of SLOs for different layers [7]



DeepSLOs are used for fine-grained control [7]

[7] Pujol et al., **DeepSLOs for the Computing Continuum** (2024)

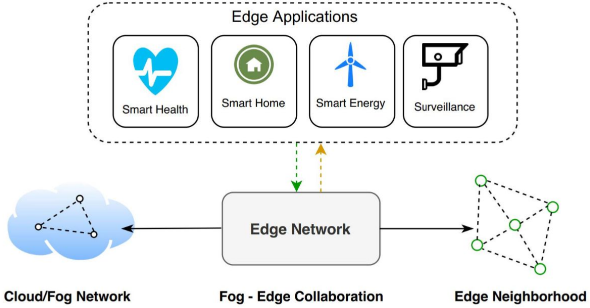
Skipped in main presentation

I – Edge Intelligence

Ensuring the desired levels of service requires reactive software components close to the data source, i.e., transfer logic from distant cloud servers to the **Edge**

Edge Intelligence as a game changer to learn how to ensure requirements and enforce mart adaptations with low latency, e.g., close to a video camera

Nevertheless, decentralizing the intelligence brings various problems in terms of orchestration and synchronization, but the benefits weight more!



Embedding intelligence directly at the Edge [8]

[8] Dustdar et al., **Towards Distributed Edge-based Systems** (2020)

II – Stream Processing Scenarios

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.

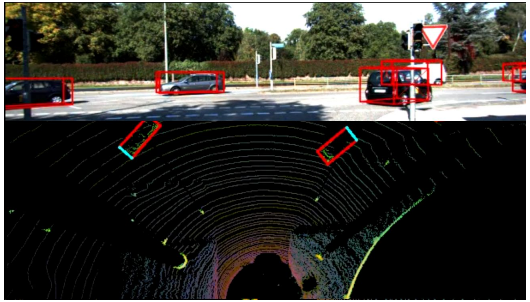
Video Processing (Yolo V8)

Object detection in a video stream using Yolo [9]

QR Scanner (OpenCV)

QR code scanning in a video using OpenCV [9]

Mobile Mapping (Lidar)



Creating a mobile map from binaries using Lidar [9]

[9] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2024)

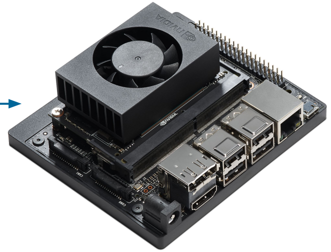
II – Dynamic Service Adaptation

Objective

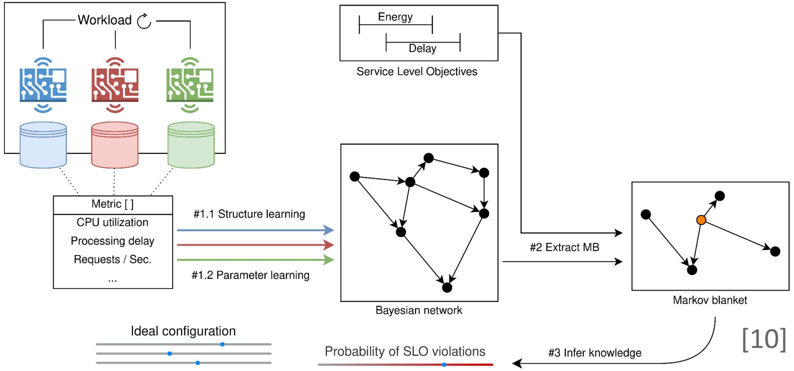
Extract an interpretable representation for one **processing tasks** on one individual **processing device**, aim to infer adaptations that allow ensuring SLOs; resulting model contains:

- ❑ Target objectives (i.e., SLOs)
- ❑ Reduction to influential factors
- ❑ Optimal **system configuration**

3-Step methodology for providing this model through (**1**) Bayesian Network Learning (BNL), (**2**) Markov Blanket (MB) extraction, and (**3**) Exact Inference.



Jetson Xavier [10]

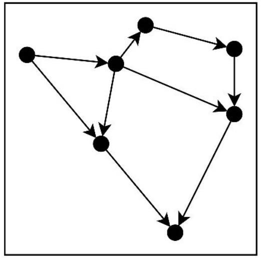


[10]

[10] Sedlak et al., **Designing Reconfigurable Intelligent Systems with Markov Blankets** (2023)

II – Dynamic Service Adaptation (2)

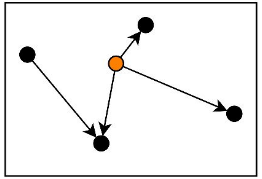
Bayesian Network Learning



Bayesian network

- **Structure Learning**
  Hill-Climb Search (HCS)
  Directed Acyclic Graph (DAG)
- **Parameter Learning**
  Max. Likelihood Estimation
  Conditional Prob. Table (CPT)

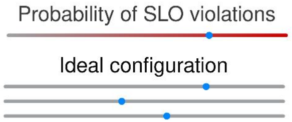
Markov Blanket Selection



Markov blanket

- **Causality filter**
  Various algorithms available
  Extract a subset of variables
- Identify variables that have an impact on **SLO fulfillment**

Knowledge Extraction



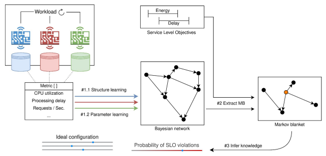
- P(SLO < x) for all variable combinations
- Find **Bayes-optimal** system configuration

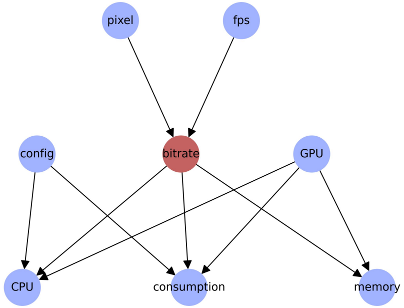
II – Dynamic Service Adaptation (3)

Allows extracting a network of causal variable relations that help **interpret** internal system metrics

Given the MB, it is possible to infer the “optimal” service configuration to fulfill **SLO**, e.g., adjust video res. & fps

Base mechanism that is embedded to constrain and supervise **microservices**





MB around streaming **bitrate** [10]

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

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

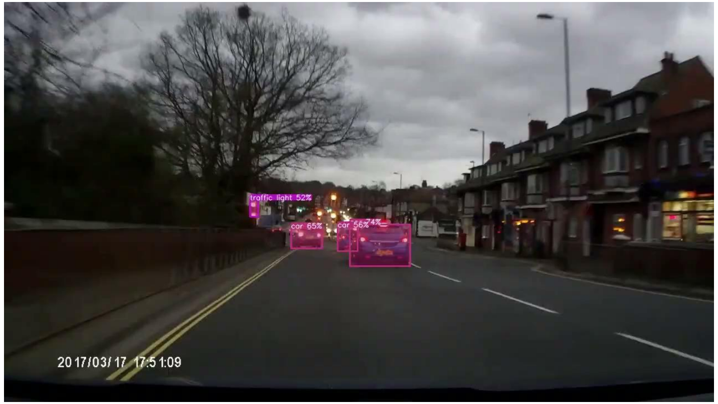
Optimal device configurations according to SLO thresholds [10]

II – Transitive SLOs in Microservice Pipelines

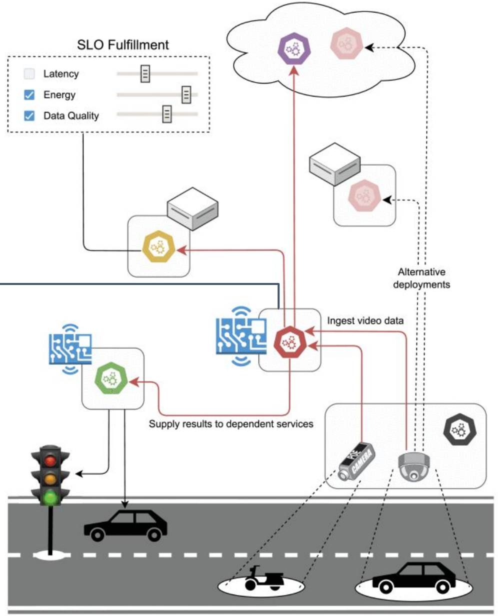
Problem

Microservice pipelines in smart city pose different **SLOs**, e.g., road surveillance → object detection → traffic routing

Various types of processing devices available; **transitive SLOs** and **device heterogeneity** constrain service deployment



Processing services in a microservice pipeline [2]



Assigning microservices to infrastructure in a smart city [2]

II – Transitive SLOs in Microservice Pipelines (2)

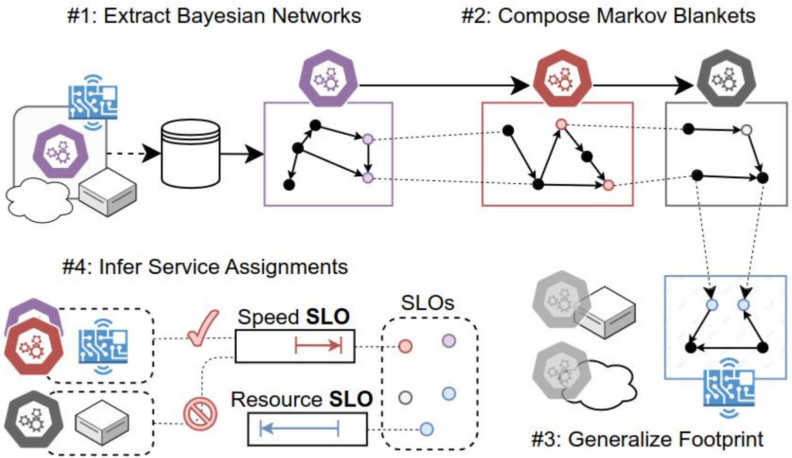
Approach

(**1**) **Extract BN**; find internal variable links in system

(**2**) **Compose MB**; identify links between multiple dependent microservices

(**3**) **Generalize "footprint"**; estimate SLO-F for unknown service-host combinations

(**4**) **Infer** the Bayes-optimal assignment of services to hosting devices in the CC



Finding optimal assignment for microservice pipeline [2]

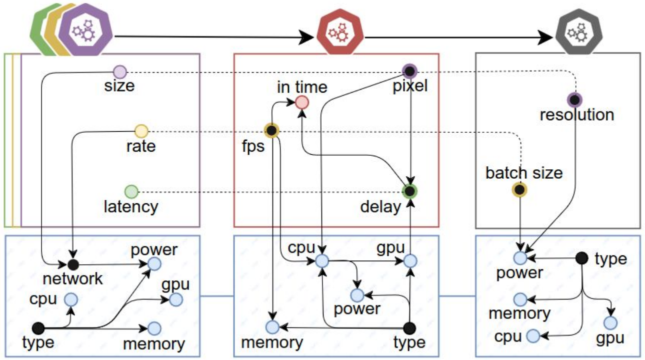
II – Transitive SLOs in Microservice Pipelines (3)

Outcomes

Identify, and in further consequence, assure SLOs posed by hierarchical services (e.g., **latency** or **quality**)

Assign individual services to target devices according to **expected** SLO fulfillment of the **pipeline** and the maximum capabilities of heterogeneous devices

Set of 5 devices in the CC and 5 devices with different processing demands; evaluate all permutations of how to assign services or make greedy assignment



Identified variable dependencies between microservices [2]

Full Device Name	ID	Price[6]	CPU	RAM	CUDA GPU	p [1,4]	g [0,3]	$nl_{N\rightarrow\_}$
Custom Server Build	*Server (S)*	2500 €	AMD Ryzen 7700 (8 core)	64 GB	RTX 3090	Very High (4)	High (3)	20 ms
ThinkPad X1 Gen 10	*Laptop (L)*	1700 €	Intel i7-1260P (16 core)	32 GB	———	High (3)	None (0)	10 ms
Nvidia Jetson Orin	*Orin (O)*	500 €	ARM Cortex A78 (6 core)	8 GB	Volta 383	Medium (2)	Medium (2)	5 ms
Nvidia Jetson Xavier	*Xavier (X)*	300 €	ARM Carmel v8.2 (6 core)	8 GB	Amp 1024	Medium (2)	Low (1)	3 ms
Nvidia Jetson Nano	*Nano (N)*	200 €	ARM Cortex A57 (4 core)	4 GB	———	Low (1)	None (0)	——

List of devices in the architecture and their **relative hardware capabilities** [2]

#	SLO Σ	W	CPU	GPU	Mem	C_3	Power Σ
1	1.70	*Orin*	50	30	119	*Orin*	8 W
2	1.52	*Orin*	24	30	73	*Xavier*	15 W
3	0.92	*Server*	3	31	12	*Laptop*	97 W
...	...	...	...	...	...	...	...
24	0.00	*Nano*	122	35	93	*Laptop*	26 W
25	0.00	*Laptop*	54	0	27	*Laptop*	21 W

Assigning two services {W,C} over the infrastructure [2]

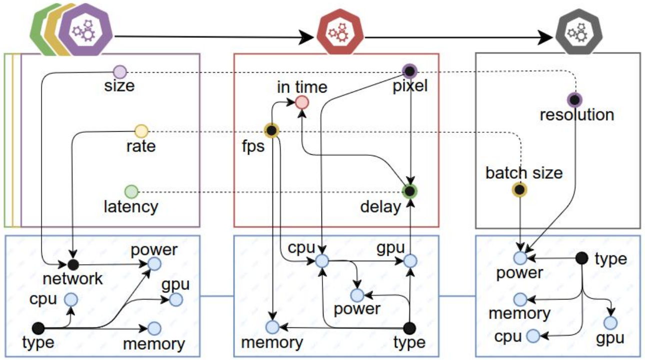
II – Transitive SLOs in Microservice Pipelines (3)

Outcomes

Identify, and in further consequence, assure SLOs posed by hierarchical services (e.g., **latency** or **quality**)

Assign individual services to target devices according to **expected** SLO fulfillment of the **pipeline** and the maximum capabilities of heterogeneous devices

Set of 5 devices in the CC and 5 devices with different processing demands; evaluate all permutations of how to assign services or make greedy assignment



Identified variable dependencies between microservices [2]

Full Device Name	ID	Price[6]	CPU	RAM	CUDA GPU	p [1,4]	g [0,3]	$nl_{N\to\_}$
Custom Server Build	*Server (S)*	2500 €	AMD Ryzen 7700 (8 core)	64 GB	RTX 3090	Very High (4)	High (3)	20 ms
ThinkPad X1 Gen 10	*Laptop (L)*	1700 €	Intel i7-1260P (16 core)	32 GB	——	High (3)	None (0)	10 ms
Nvidia Jetson Orin	*Orin (O)*	500 €	ARM Cortex A78 (6 core)	8 GB	Volta 383	Medium (2)	Medium (2)	5 ms
Nvidia Jetson Xavier	*Xavier (X)*	300 €	ARM Carmel v8.2 (6 core)	8 GB	Amp 1024	Medium (2)	Low (1)	3 ms
Nvidia Jetson Nano	*Nano (N)*	200 €	ARM Cortex A57 (4 core)	4 GB	——	Low (1)	None (0)	——

List of devices in the architecture and their **relative hardware capabilities** [2]

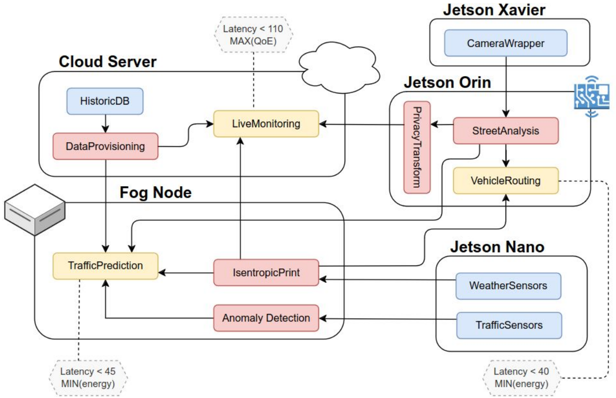
#	SLO Σ	W	CPU	GPU	Mem	C_3	Power Σ
1	1.70	*Orin*	50	30	119	*Orin*	8 W
2	1.52	*Orin*	24	30	73	*Xavier*	15 W
3	0.92	*Server*	3	31	12	*Laptop*	97 W
...	...	...	...	...	...	...	...
24	0.00	*Nano*	122	35	93	*Laptop*	26 W
25	0.00	*Laptop*	54	0	27	*Laptop*	21 W

Assigning two services {W,C} over the infrastructure [2]

II – Diffusing SLOs in Microservices

Problem

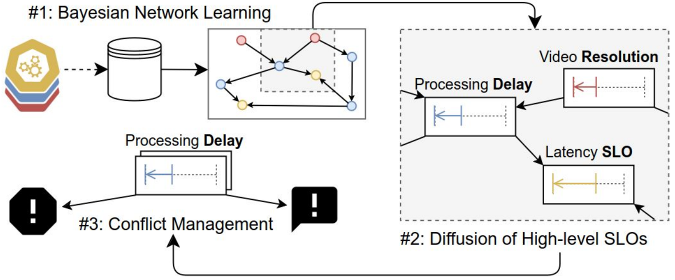
Microservice pipelines deployed over heterogeneous computing infrastructure; stakeholders focused on high-level SLO fulfillment (i.e., **KPIs**), how must lower level components operate to improve SLO fulfillment



Microservice pipelines with SLOs assigned to consumer services [11]

Approach

(**1**) Build a BN comprising all relevant services;
(**2**) traverse the tree from the leaves (i.e., high-level SLO) and infer assignments that benefit higher SLOs
(**3**) Resolve or report conflicts as far as possible



Diffusing SLOs into lower-level constraints [11]

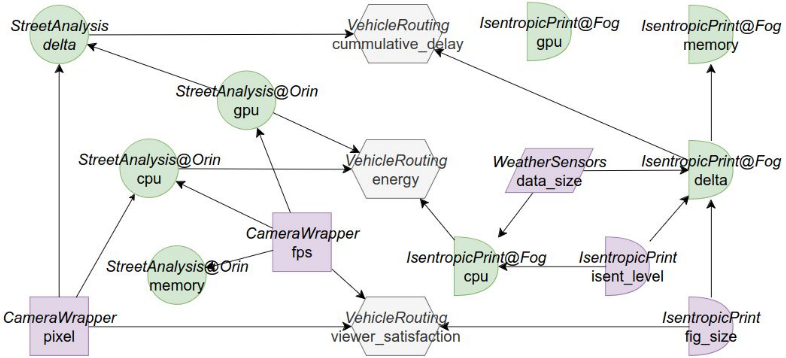
[11] Sedlak et al., **Diffusing High-level SLO in Microservice Pipelines** (2024)

II – Diffusing SLOs in Microservices (2)

Outcomes

Service pipeline **distributed** over multiple devices; diffusing high-level SLOs (e.g., delay, **energy**, and QoE) to lower-level constraints raises SLO fulfillment significantly

Individual services in charge for assuring their local SLOs – decentralizes requirements assurance; provokes **conflicts**



Variable dependencies between hierarchical microservices [11]

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay energy viewer_sat	$\leqslant$ 45 ms $\leqslant$ 19 W ——	High-level
StreetAnalysis *StreetAnalysis* *StreetAnalysis* *IsentropicPrint* *IsentropicPrint*	delta cpu (Orin) gpu (Orin) delta cpu (Fog)	$\leqslant$ 35 ms $\leqslant$ 21 % $\leqslant$ 40 % $\leqslant$ 37 ms $\leqslant$ 17 %	Low-level
CameraWrapper *CameraWrapper* *IsentropicPrint* *IsentropicPrint* *WeatherSensors*	pixel fps fig_size isent_level data_size	= 480 p = 15 f $\leqslant$ 50 p $\leqslant$ 200 k $\leqslant$ 30 pi	Parameter

High-level SLOs diffused to lower-level **constraints** [11]

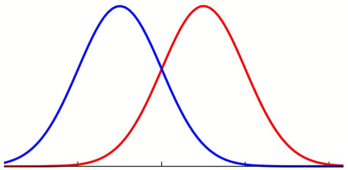
Microservice	High-level SLO	% Min	% Fulfill	% Max
VehicleRouting	cumm_delay $\leqslant$ 45 min(energy)	0.00 0.53	0.94 0.99	1.00 1.00

Parameter assignments reach close to optimal solution [11]

II – Shortcomings of Static Model Training

Problems

BNL approach requires large amounts of training data in upfront to capture all system states, however there will always be **unseen data**; variable **distributions** can change over time, distorting the ML model

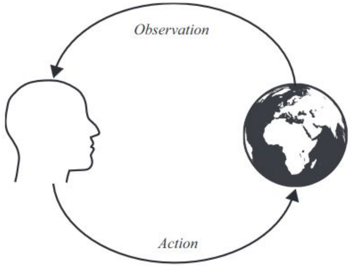


Ideal solution

Continuous learning to create accurate world models, which involves **curiosity** to develop causal understanding of system mechanics; allow understanding which elasticity strategies can fulfill SLOs (e.g., latency)

Active Inference

Partly comparable to reinforcement learning; involves perception to understand **why** certain observations happened, and **enacts** on the environment in order to make the preferred outcomes more probable.



Action-perception cycle [12]

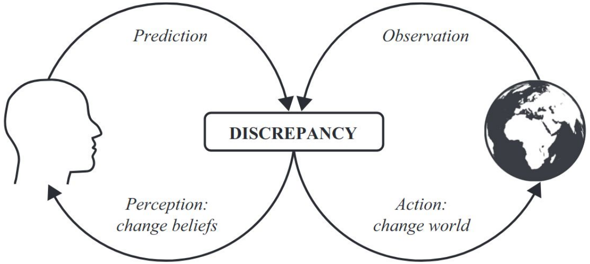
[12] Parr, Pezzulo, and Friston; Active Inference: The Free Energy Principle in Mind, Brain, and Behavior (2022)

III – Active Inference

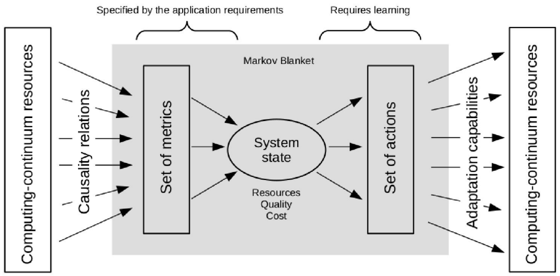
Concept from **neuroscience** developed by Friston et al. [12,13,14] that explains human cognition through minimization of free energy, i.e., resolving uncertainty

Explains world processes (i.e., computation) through **generative models** trained by agents; in case agents are surprised by external stimuli (i.e., sensory data), they **adjust** their perception or **enact** on their environment

Connects well to the concept of the **Markov blanket**, which allows to express how one system is impacted by the actions or states of another system



Resolving discrepancy through action and perception [12]



Behavioral Markov blanket for a system [2]

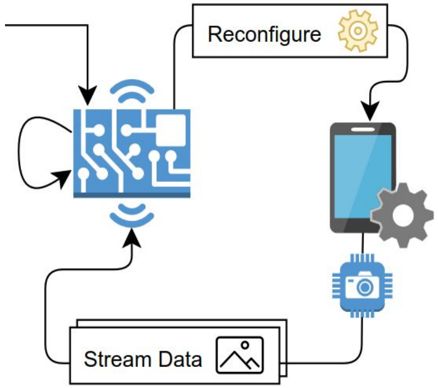
[13] Friston et al., **Designing ecosystems of intelligence from first principles** (2024)
[14] Kirchhoff et al., **The Markov blankets of life: autonomy, active inference and the free energy principle** (2018)

III – Active Inference

Mapping between neuroscience and distributed computing systems [15,16]; understanding processing requirements (i.e., SLOs) as a form of **homeostasis,** e.g., cell temperature

Create autonomous components that identify how to ensure requirements and resolve them independently, clear modelling between higher-level and low-level components

Rather experimental since it originates from a metaphor from biology but with a lot of potential due to its history



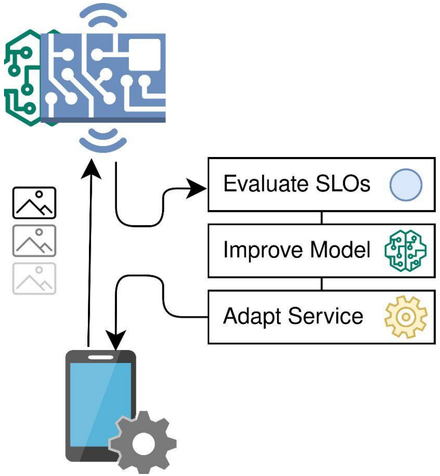
Ensure internal requirements [15]

[15] Sedlak et al., **Active Inference on the Edge: A Design Study** (2024)
[16] Sedlak et al., **Equilibrium in the Computing Continuum through Active Inference** (2024)

III – Continuous SLO Fulfillment

3 major contributions in interplay:

1. Continuous model accuracy and local SLO fulfillment

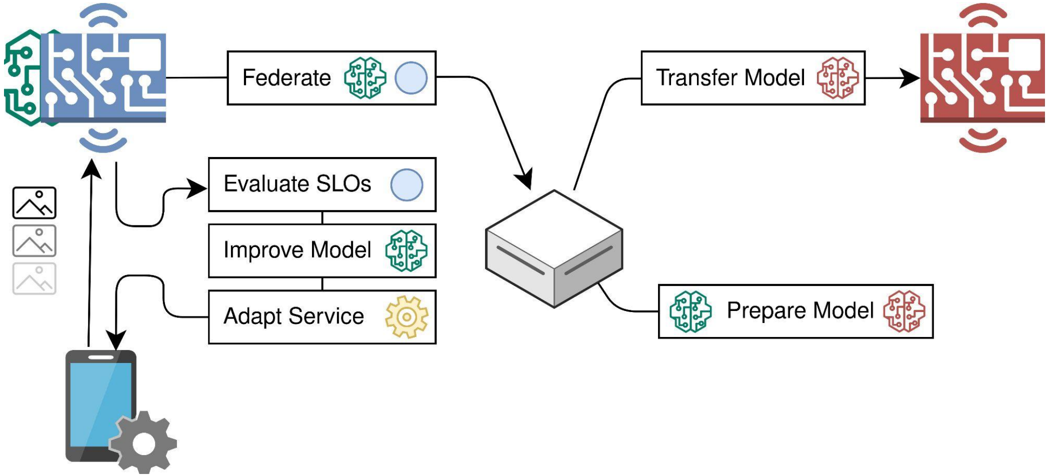


High-level AIF methodology for training and exchanging causal models between devices [16]

III – Continuous SLO Fulfillment

3 major contributions in interplay:

1. Continuous model accuracy and local SLO fulfillment
2. Federation and combination of models

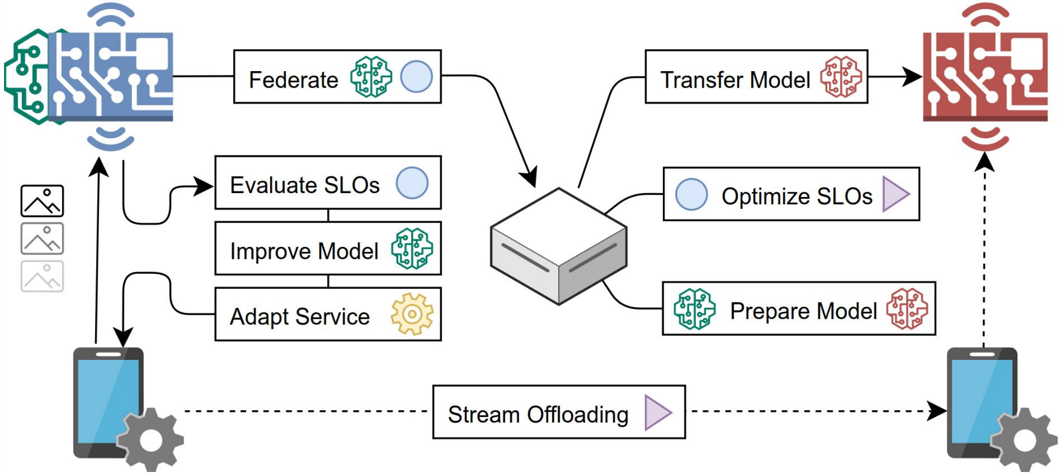


High-level AIF methodology for training and exchanging causal models between devices [16]

III – Continuous SLO Fulfillment

3 major contributions in interplay:

1. Continuous model accuracy and local SLO fulfillment
2. Federation and combination of models
3. Collaboration between cellular structures



High-level AIF methodology for training and exchanging causal models between devices [16]

III – Continuous BNL and Inference

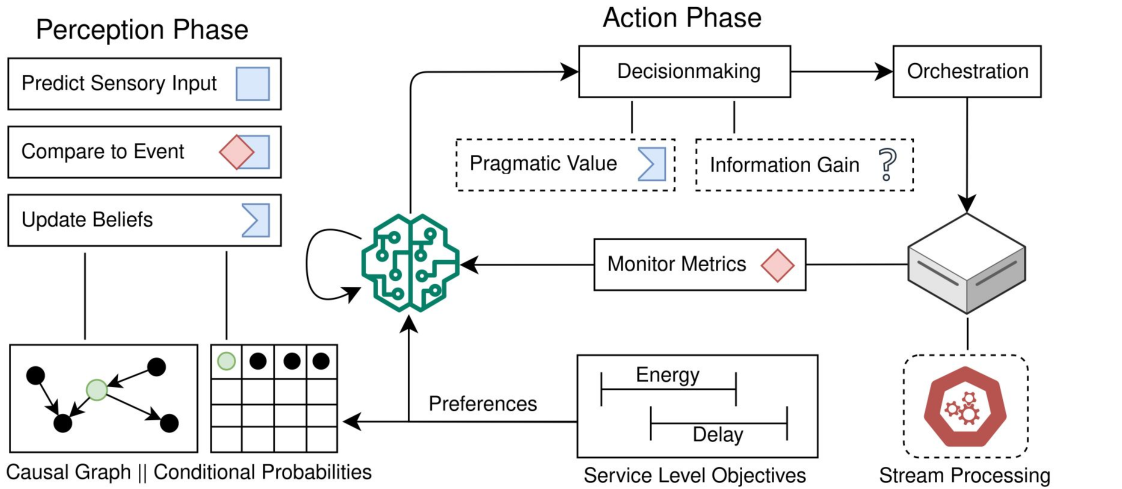
Approach

(1) **Specify** processing boundaries through multiple SLOs

(2) **AIF agents** perceive their environment and enact on it

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

(4) **Action** phase reconfigure local processing environment to minimize FE and fulfill SLOs



Action and perception cycles performed by the AIF agent to create an accurate model and shape the world [16]

III – AIF Agent Behaviour

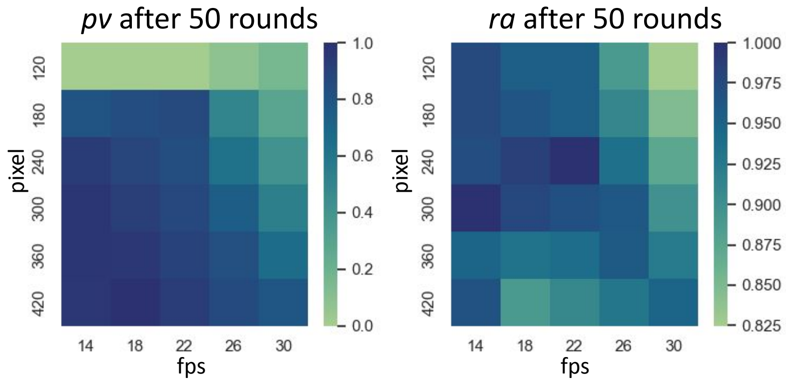
Determined by three factors:

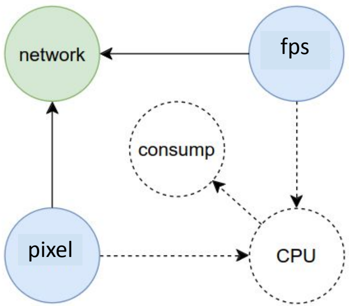
- Pragmatic value (*pv*)
  Summarizes **QoE** SLOs (e.g., resolution)
- Risk assigned (*ra*)
  Summarizes **QoS** SLOs (e.g., network limit)

pv & *ra* calculated as **separate factors** from MBs; configurations rated according to SLO fulfillment; **interpolation** between known configurations

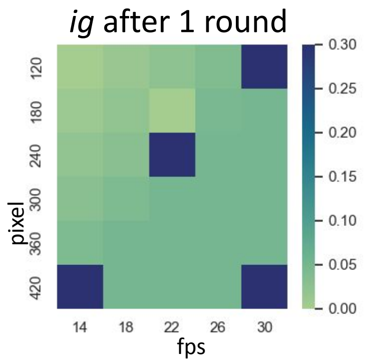
- Information gain (*ig*)
  Continued on the next slide

$$u_c = pv_c + ra_c + ig_c$$





III – AIF Agent Behaviour (cont.)

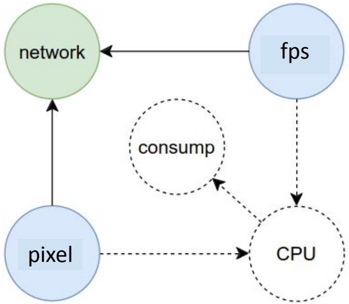


- Information gain (*ig*)
  - Favors configurations that promise **model improvement**
  - Summarizes surprise for observations included in the **MB**
  - Hyperparameter (*e*) allows exploring designated areas

AIF agent cycle:

$$ig(c) = e + \left(\frac{\tilde{\mathfrak{S}}_c}{\bar{\mathfrak{S}}}\right) \times 100$$

1. Calculate **surprise** for current batch of observations
2. Retrain structure (or parameters) depending on surprise
3. Calculate behavioral factor for **empirically evaluated** configs
4. **Interpolate** between known configurations in 2D (or 3D) space
5. Choose the highest-scoring (device) configuration



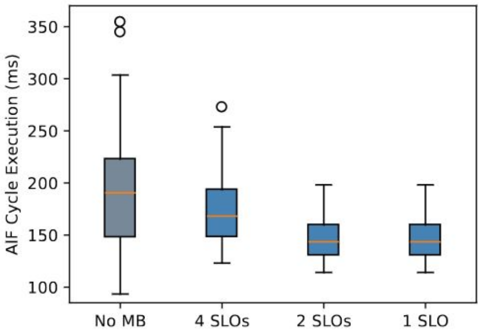
Agent gradually develops **understanding** how to ensure SLOs

III – Outcomes

Evaluation included a total number of 12 aspects

Question: Do MBs reduce the time for each inference cycle?

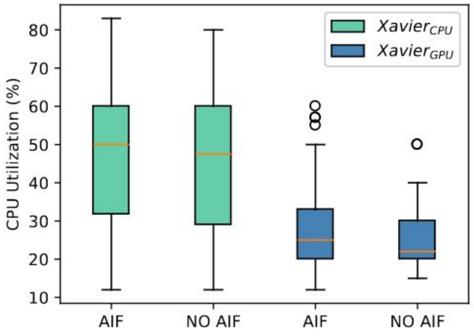
Answer: Filtering the BN to a lower subset of nodes, i.e., the MBs of x SLOs, reduced the time



Inference time for entire BN or subsection [16]

Question: How high is the AIF agent's operational overhead?

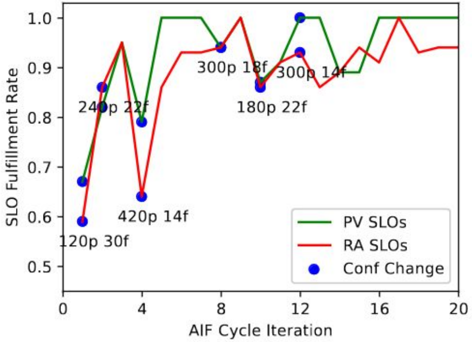
Answer: For the two evaluated devices the AIF overhead was reported around 2 %



Overhead of running the AIF agent [16]

Question: How long might a AIF agent require to ensure 4 SLOs?

Answer: Starting from no prior knowledge, the agent required 16 rounds and 5 reconfiguration



SLO fulfillment starting from scratch [16]

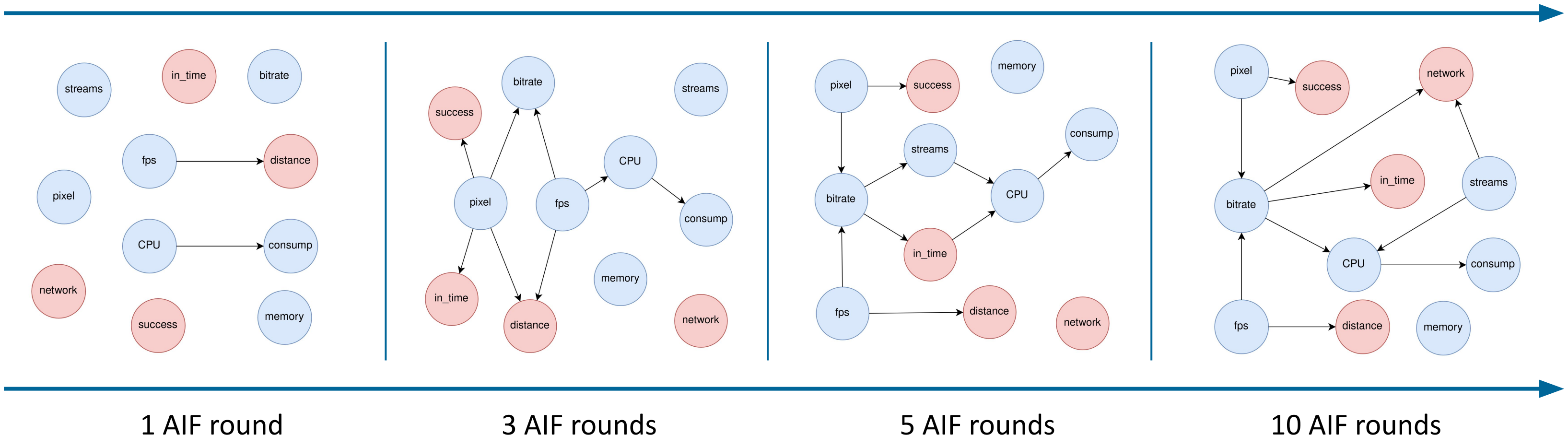
Skipped in main presentation

III – Outcomes

Evaluation included a total number of 12 aspects

Question: Are the produced causal graphs interpretable?

Answer: Gradually training an empirically verifiable graph that allow to extract MBs around the target SLO variables (●), thus identifying influential factors

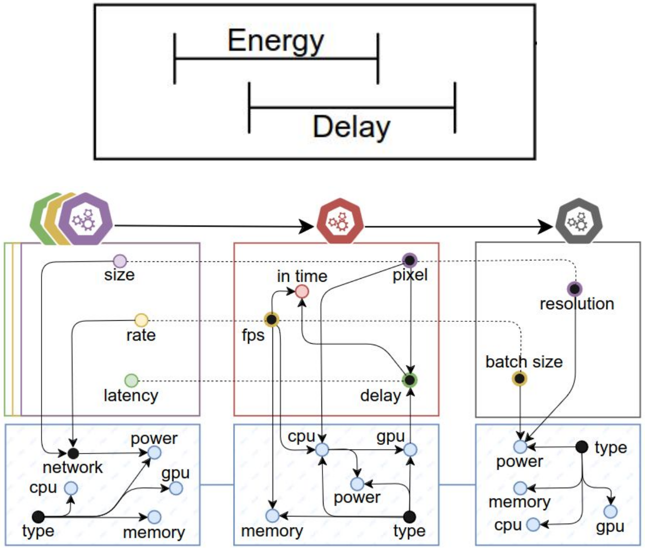


Summary

Processing SLOs must be continuously ensured; presented mechanisms designed to analyze and supervise various stream processing use cases

Edge intelligence as a measure to train generative models with low latency and perform inference, i.e., find the Bayes-optimal service configurations

Active Inference as a novel method to combine perception (i.e., interpretation) of processing and adjust (i.e., reconfigure) it dynamically accordingly



Thank you a lot for listening attentively!
Please give me your **opinions** and **ideas**

@

boris.sedlak@dsg.tuwien.ac.at

https://www.linkedin.com/in/boris-sedlak/