Markov Blanket Composition of SLOs

TU Wien: Boris Sedlak, Victor Casamayor Pujol, Praveen Kumar Donta, Schahram Dustdar

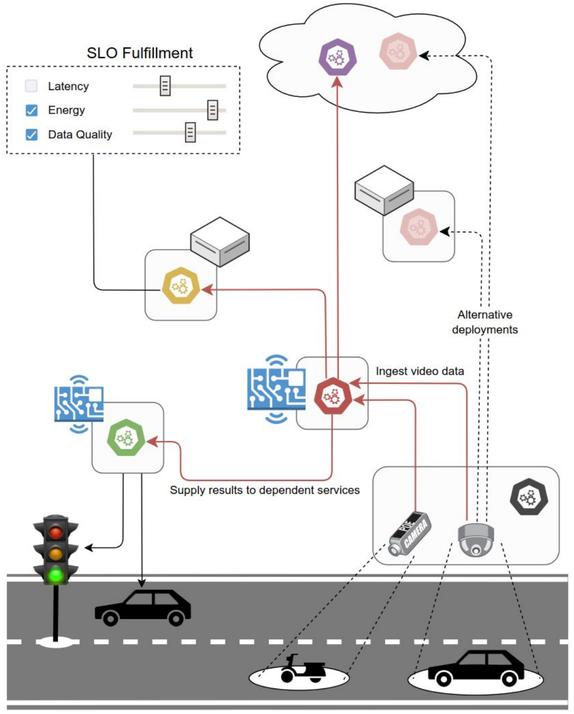
Problem Statement

Sensor systems are ubiquitous, e.g., **smart cities**, factories, etc

Composable microservice pipelines in smart city (e.g., road surveillance ⟶ object detection ⟶ traffic routing / monitoring)

Each service is dependent on the **outcome** of its predecessor; must fulfill QoS and QoE requirements, e.g., latency or quality. Requirements specified as Service Level Objectives (**SLOs**)

Hardware **heterogeneity** and transitive SLOs (imposed by dependent services) constrain where each service should be deployed, i.e., **Edge**, Fog, Cloud



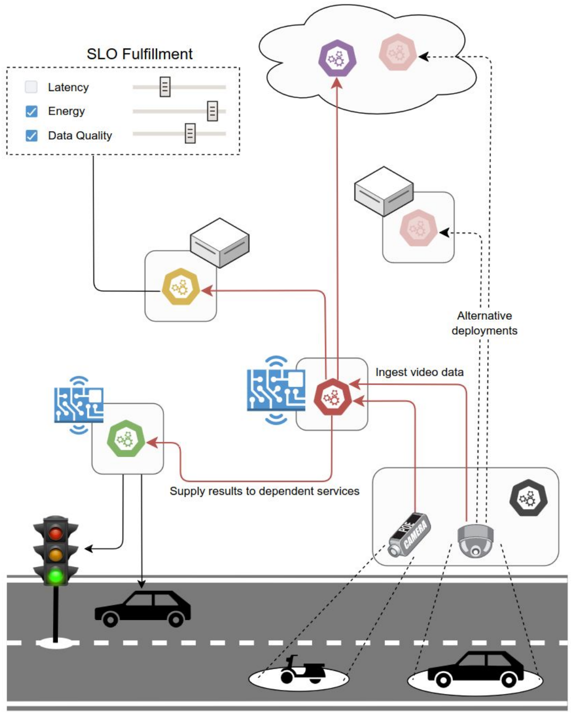
Problem Statement

Sensor systems are ubiquitous, e.g., **smart cities**, factories, etc

Composable microservice pipelines in smart city (e.g., road surveillance → object detection → traffic routing / monitoring)

Each service is dependent on the **outcome** of its predecessor; must fulfill QoS and QoE requirements, e.g., latency or quality. Requirements specified as Service Level Objectives (**SLOs**)

Hardware **heterogeneity** and transitive SLOs (imposed by dependent services) constrain where each service should be deployed, i.e., **Edge**, Fog, Cloud



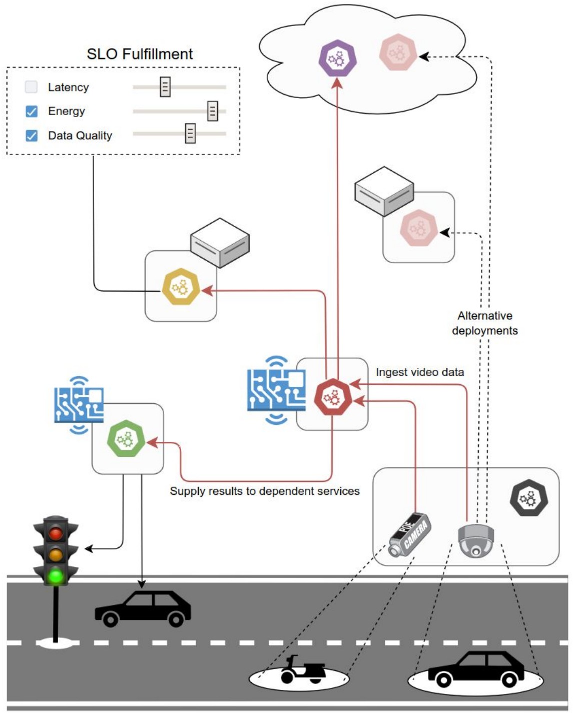
Problem Statement (2)

Given a **microservice pipeline** and a set of **heterogeneous hosts**, where to deploy each individual service so that its **own** SLOs and those of **dependent** services are fulfilled?

"Assignment between services and hosts"

Exhaustive comparison of all assignments is impractical due to complexity and lack of understanding; services and hosts change over time, so knowledge must be **transferable** to other setups

→ Intersections between services' dependencies modeled and evaluated through composition of **Markov Blankets** (MBs)



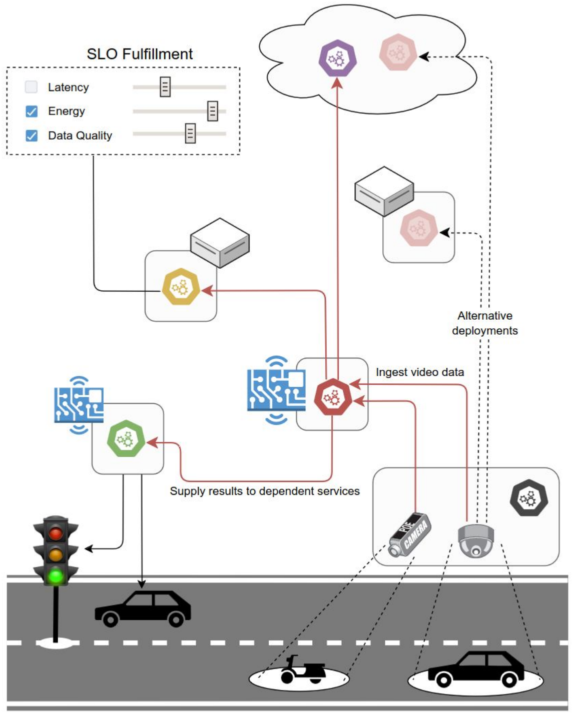
Problem Statement (2)

Given a **microservice pipeline** and a set of **heterogeneous hosts**, where to deploy each individual service so that its **own** SLOs and those of **dependent** services are fulfilled?

“Assignment between services and hosts”

Exhaustive comparison of all assignments is impractical due to complexity and lack of understanding; services and hosts change over time, so knowledge must be **transferable** to other setups

→ Intersections between services’ dependencies modeled and evaluated through composition of **Markov Blankets** (MBs)

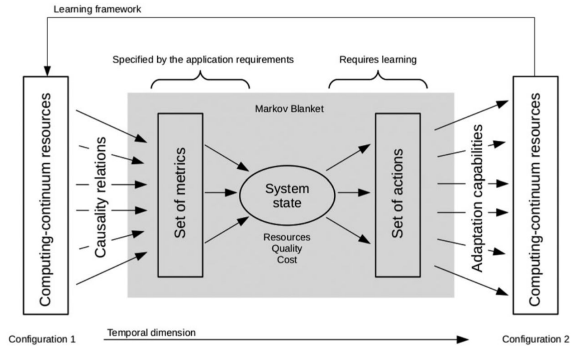


Behavioral Markov Blankets

$$P(x \mid \mathrm{MB}(x), Y) = P(x \mid \mathrm{MB}(x))$$

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

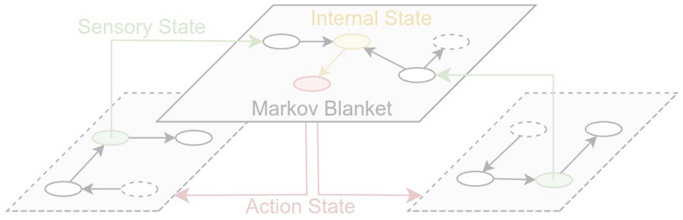
Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions



Behavioral Markov blanket for a system [1]

Composition of MBs to model hierarchical dependencies between different entities and their interfaces

→ How individual services impact SLOs of dependent service



Action-perception cycle between multiple entities

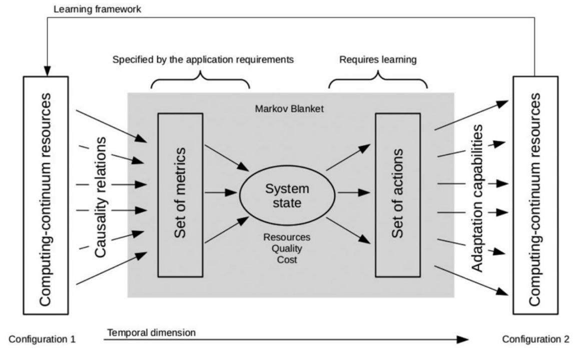
[1] Dustdar, Casamayor Pujol, and Dustdar; On Distributed Computing Continuum Systems (2023)

Behavioral Markov Blankets

$$P(x \mid \mathrm{MB}(x), Y) = P(x \mid \mathrm{MB}(x))$$

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

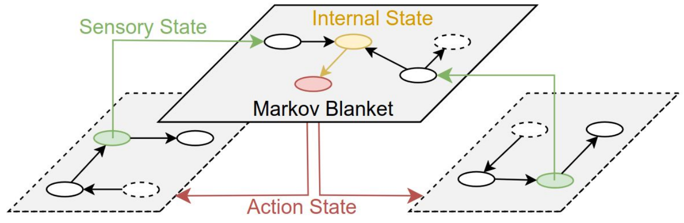
Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions



Behavioral Markov blanket for a system [1]

Composition of MBs to model hierarchical dependencies between different entities and their interfaces

→ How individual services impact SLOs of dependent service



Action-perception cycle between multiple entities

[1] Dustdar, Casamayor Pujol, and Dustdar; On Distributed Computing Continuum Systems (2023)

Methodology

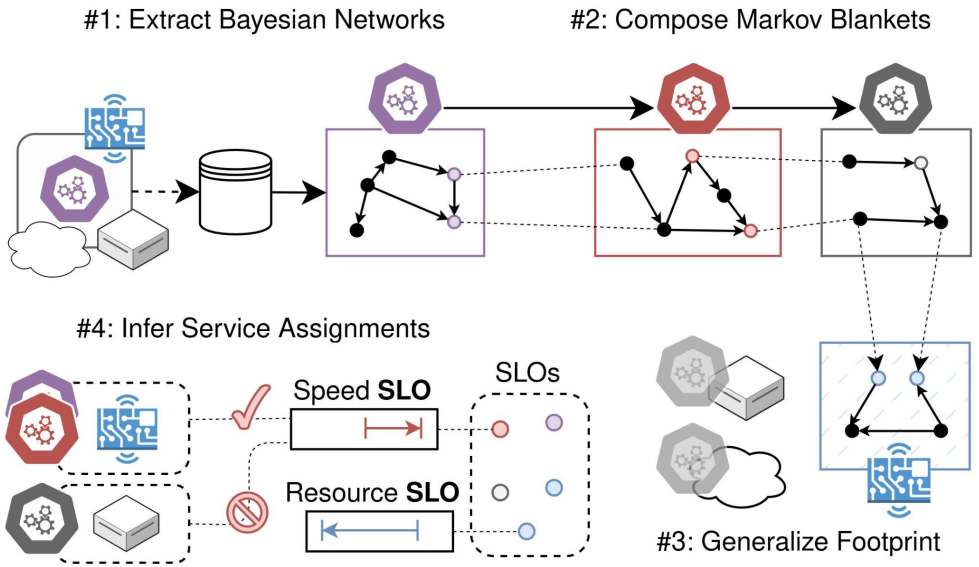
4-Step approach

#1 Extract BN as a probabilistic view into the service execution

#2 identify variable links between dependent services

#3 estimate SLO fulfillment under unknown service-host combinations

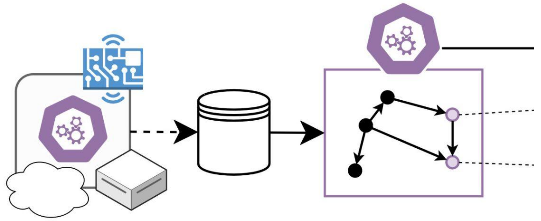
#4 infer the optimal assignment of services and hosting devices


#1: Extract Bayesian Networks
#2: Compose Markov Blankets
#4: Infer Service Assignments
SLOs
Speed SLO
Resource SLO
#3: Generalize Footprint

#1 Markov Blanket Extraction

Monitor service executions on target devices to gain insights into **causal dependencies** of service variables, e.g., *video resolution* → *processing latency*, and the impact on the hardware, e.g., *cpu*

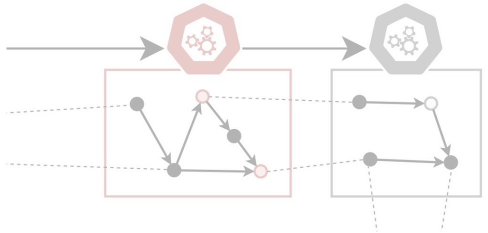
Train causal graph (i.e., BN) and extract the MB around the **SLO variables**, i.e., getting all nodes that impact SLO fulfillment



#2 Markov Blanket Composition (MBC)

Identify **interface variables** that impact the SLOs of **subsequent** services in the processing pipeline. Uses the Wasserstein distance (WSD) to find statistical dependencies between variables.

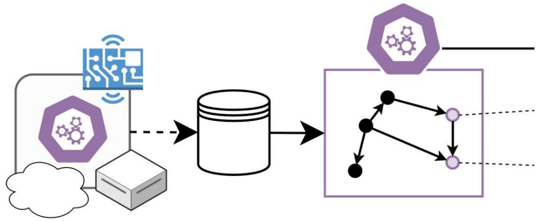
Can be distorted by linear **confounding variables** external to the MBs, e.g., networking latency, which is still detected by WSD



#1 Markov Blanket Extraction

Monitor service executions on target devices to gain insights into **causal dependencies** of service variables, e.g., *video resolution* → *processing latency*, and the impact on the hardware, e.g., *cpu*

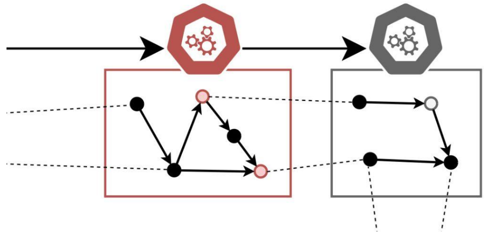
Train causal graph (i.e., BN) and extract the MB around the **SLO variables**, i.e., getting all nodes that impact SLO fulfillment



#2 Markov Blanket Composition (MBC)

Identify **interface variables** that impact the SLOs of **subsequent** services in the processing pipeline. Uses the Wasserstein distance (WSD) to find statistical dependencies between variables.

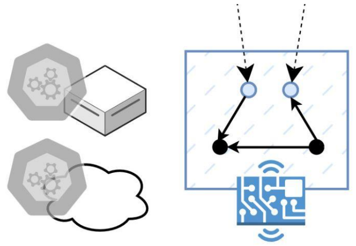
Can be distorted by linear **confounding variables** external to the MBs, e.g., networking latency, which is still detected by WSD



#3 Footprint Generalization

Extrapolate from the implications of running a service at a specific host (i.e., its “footprint”) to other services and devices

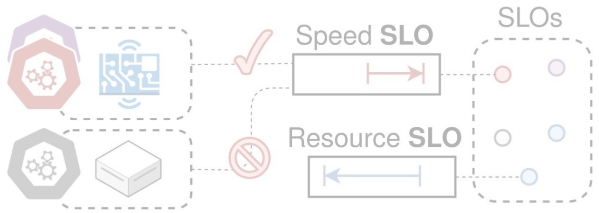
Estimate the expected SLO fulfillment and hardware utilization according to (i) the **relative difference** to known hardware, and (ii) the type of pipeline service, e.g., CV task or consumer



#4 Service Assignment

Assigns a set of services to a set of hosts, while respecting device capabilities and (transitive) SLOs specified per service

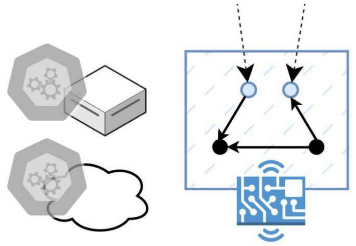
Can either assign services **greedily**, i.e., start from the consumer and place at the host that best fulfills its requirements, or **collaboratively**, which exhaustively compares the 2^n options



#3 Footprint Generalization

Extrapolate from the implications of running a service at a specific host (i.e., its “footprint”) to other services and devices

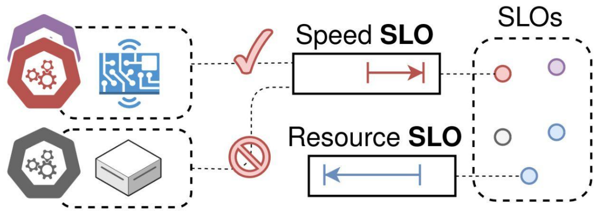
Estimate the expected SLO fulfillment and hardware utilization according to (i) the **relative difference** to known hardware, and (ii) the type of pipeline service, e.g., CV task or consumer



#4 Service Assignment

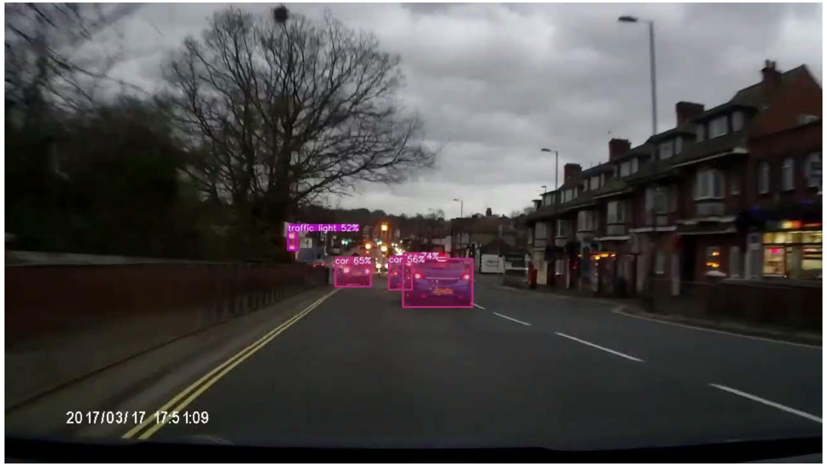
Assigns a set of services to a set of hosts, while respecting device capabilities and (transitive) SLOs specified per service

Can either assign services **greedily**, i.e., start from the consumer and place service the host that best fulfills its requirements, or **collaboratively**, which exhaustively compares the 2^n options

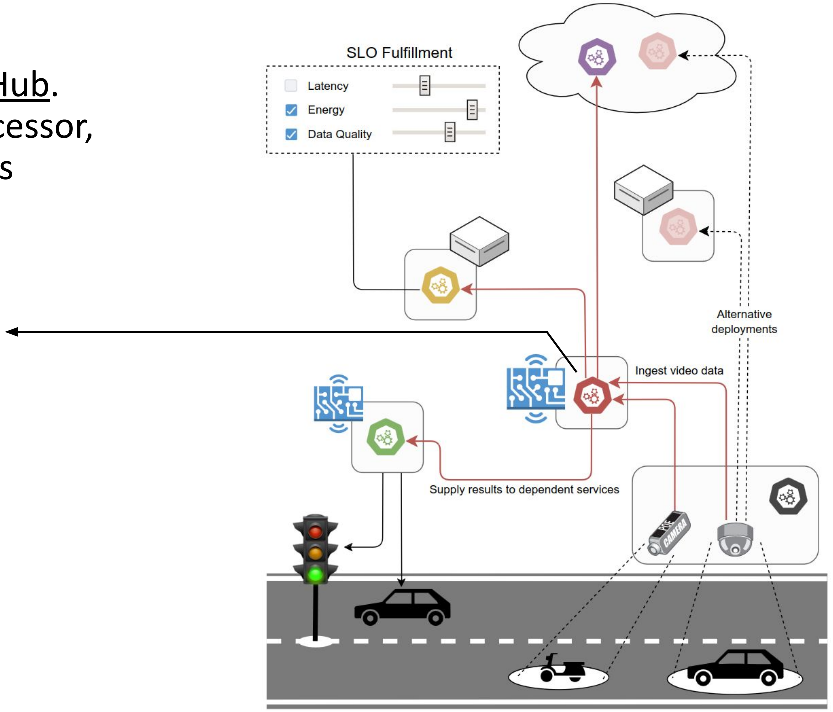


Implementation

Prototype implemented in Python and available on GitHub. Comprises services for (**a**) video provider, (**b**) video processor, and (**c**) three instances of consumers with different SLOs



CV Service with Yolov8 running on the produced videos



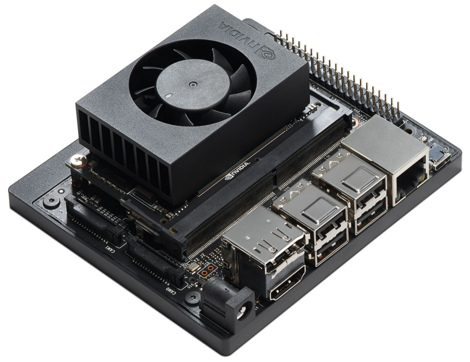
Assigning microservices to infrastructure in a smart city

Experimental Setup

Contains a set of 5 **heterogeneous** devices that will be available for assignment, each classified according to cpu (*p*) and gpu (*g*) capabilities

SLOs specified for each of the services, i.e., for producer (*P*), worker (*W*), and the three consumers (C1, C2, C3). Each with different **thresholds** according to the requirements, e.g., live monitoring has different SLOs than the adaptation of smart traffic lights.

For eight setups, execute the methodology and find best assignment


Nvidia Jetson **Xavier NX** board (image from here)

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

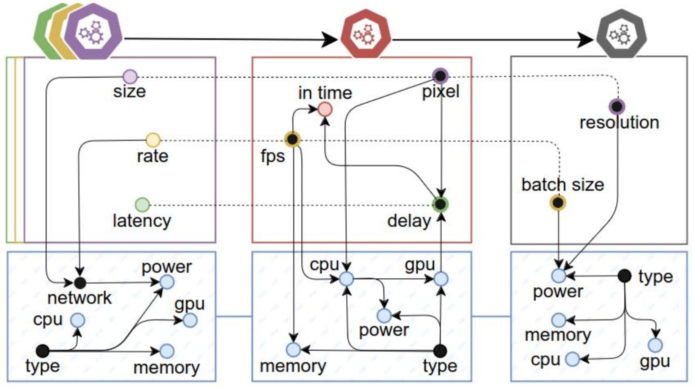
	P	W	C_1	C_2	C_3
latency	–	–	$\leqslant$ 1s	$\leqslant$ 70ms	$\leqslant$ 40ms
image size	–	–	$\geqslant$ 720p	–	–
data rate	–	–	–	$\geqslant$ 25f	–
delta	–	$\leqslant \frac{1}{fps}$	–	–	–
*energy**	$min()$	–	–	–	$min()$

SLOs posed **by** individual services

Results

Composition of MBs provides (1) **understanding** how different service parameters depend on each other, e.g., latency ⟶ delay. Thus, consumers constrain **red's** operation, and **grey's** in turn.

Further, the MB shows (2) the precise impact of the services on the computing hardware, means to what extent **red** utilizes the processing resources (e.g., cpu, gpu) of its host.



Identified variable dependencies between microservices

Considers for pair of dependent services (W & C3) how assigning them to different hosts impacts resulting SLO fulfillment of the latter, i.e., how red must to deploy to more powerful hosts

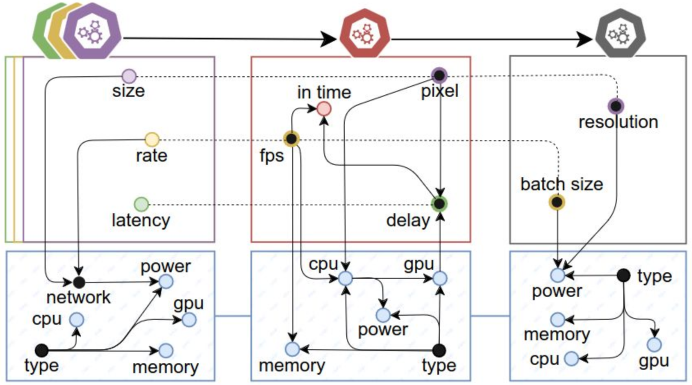
#	SLO Σ	W	CPU	GPU	Mem	C_3	Power Σ
1	1.70	*Orin*	50	30	119	*Orin*	8 W
2	1.52	*Orin*	24	30	73	*Xavier*	15 W
3	0.92	*Server*	3	31	12	*Laptop*	97 W
...	...	...	...	...	...	...	...
24	0.00	*Nano*	122	35	93	*Laptop*	26 W
25	0.00	*Laptop*	54	0	27	*Laptop*	21 W

Assigning two services {W,C} over the infrastructure

Results

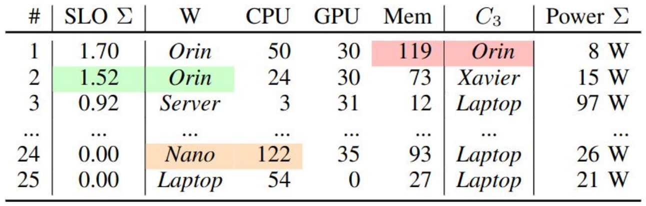
Composition of MBs provides (1) **understanding** how different service parameters depend on each other, e.g., latency → delay. Thus, consumers constrain **red's** operation, and **grey's** in turn.

Further, the MB shows (2) the precise impact of the services on the computing hardware, means to what extent **red** utilizes the processing resources (e.g., cpu, gpu) of its host.



Identified variable dependencies between microservices

Considers for pair of dependent services (*W* & *C3*) how assigning them to different hosts impacts resulting SLO fulfillment of the latter, i.e., how **red** must to deploy to more powerful hosts



#	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

Results (2)

Depending on the evaluation scenario, different services had to be assigned to different sets of hosts. Thus, **SLOs**, **processing parameters**, and the optimal **assignment** changed accordingly.

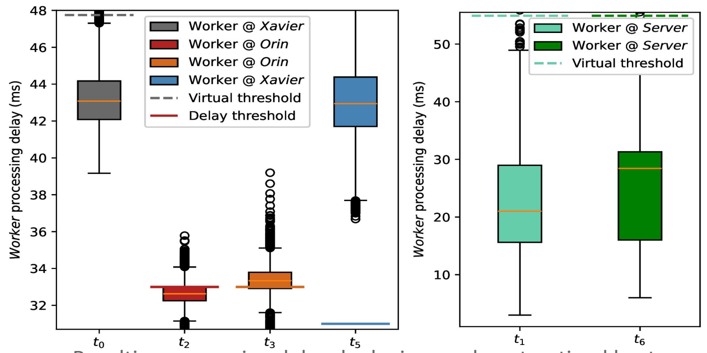
Fulfillment of worker's processing SLO highly dependent on chosen **host** and imposed latency **threshold**. For example, in scenario **t2**, deploying to Orin fulfilled the SLOs, whereas **t3** not

---

Comparing the empirical SLO fulfillment of all the possible assignments exhaustively showed that we indeed selected the optimal ones, though we extrapolated from other assignments

	Services					Hosts				
t_i	P	W	C_1	C_2	C_3	S	L	O	X	N
0	✓	✓	✓	–	–	✓	✓	✓	✓	✓
1	✓	✓	–	✓	–	✓	✓	✓	✓	✓
2	✓	✓	–	–	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

List of eight evaluation scenarios with services and hosts

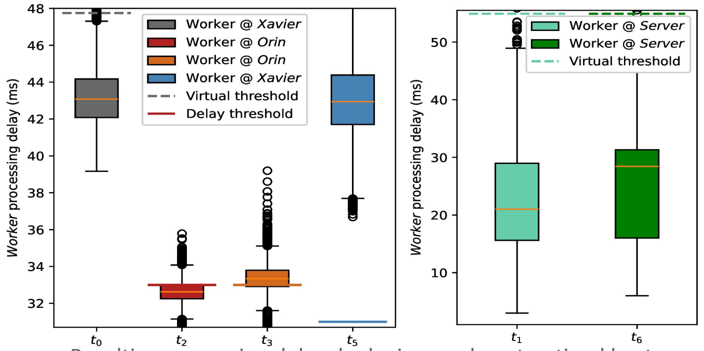


Resulting processing delay deploying worker at optimal hosts

Results (2)

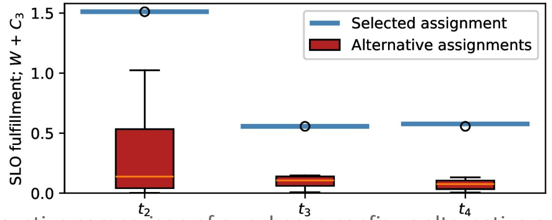
Depending on the evaluation scenario, different services had to be assigned to different sets of hosts. Thus, **SLOs**, **processing parameters**, and the optimal **assignment** changed accordingly.

Fulfillment of worker's processing SLO highly dependent on chosen **host** and imposed latency **threshold**. For example, in scenario **t2**, deploying to Orin fulfilled the SLOs, whereas **t3** not



Resulting processing delay deploying worker at optimal hosts

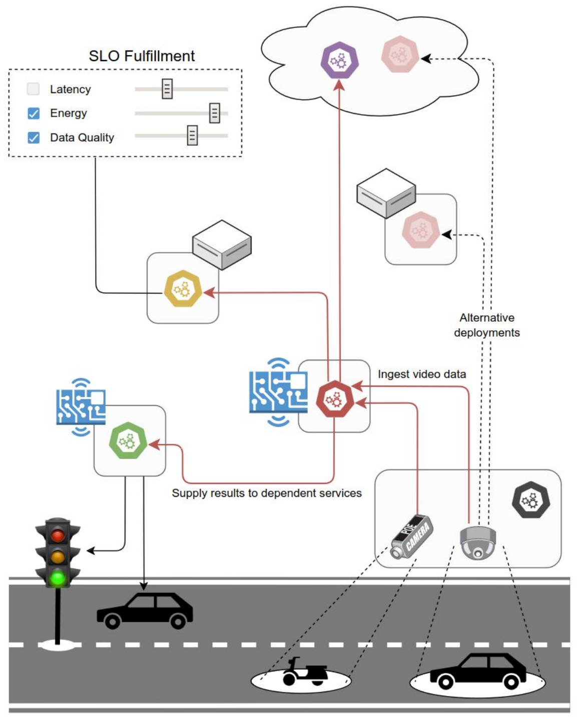
Empirical comparison of the SLO fulfillment of all possible assignments showed that we indeed selected the optimal assignment, though we extrapolated from other assignments



Exhaustive comparison of our chosen config vs alternative ones

Summary

- ❏ Individual services in a microservice pipelines pose SLOs to the overall performance and **dependent** services
- ❏ Constrain how services operate and where they can be deployed over different heterogeneous hosts
- ❏ Find statistical dependencies between services through **composition** of their MBs and inference of assignments
- ❏ Prototype showed how a smart-city pipeline was deployed over a **mutable** set of hosts while maximizing the resulting SLO fulfillment



Let’s **discuss**!

Please come forward with any **question** you have

@

boris.sedlak@dsg.tuwien.ac.at