ADA-PIPE

Data-aware pipeline scheduling and adaptation

Narges Mehran, Dragi Kimovski, Radu Prodan

University of Klagenfurt

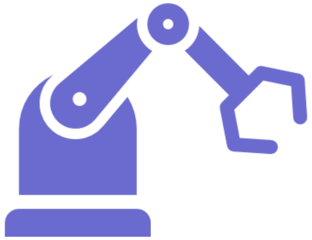
Lyon Plenary Meeting
20.06.2023

ADA-PIPE objectives

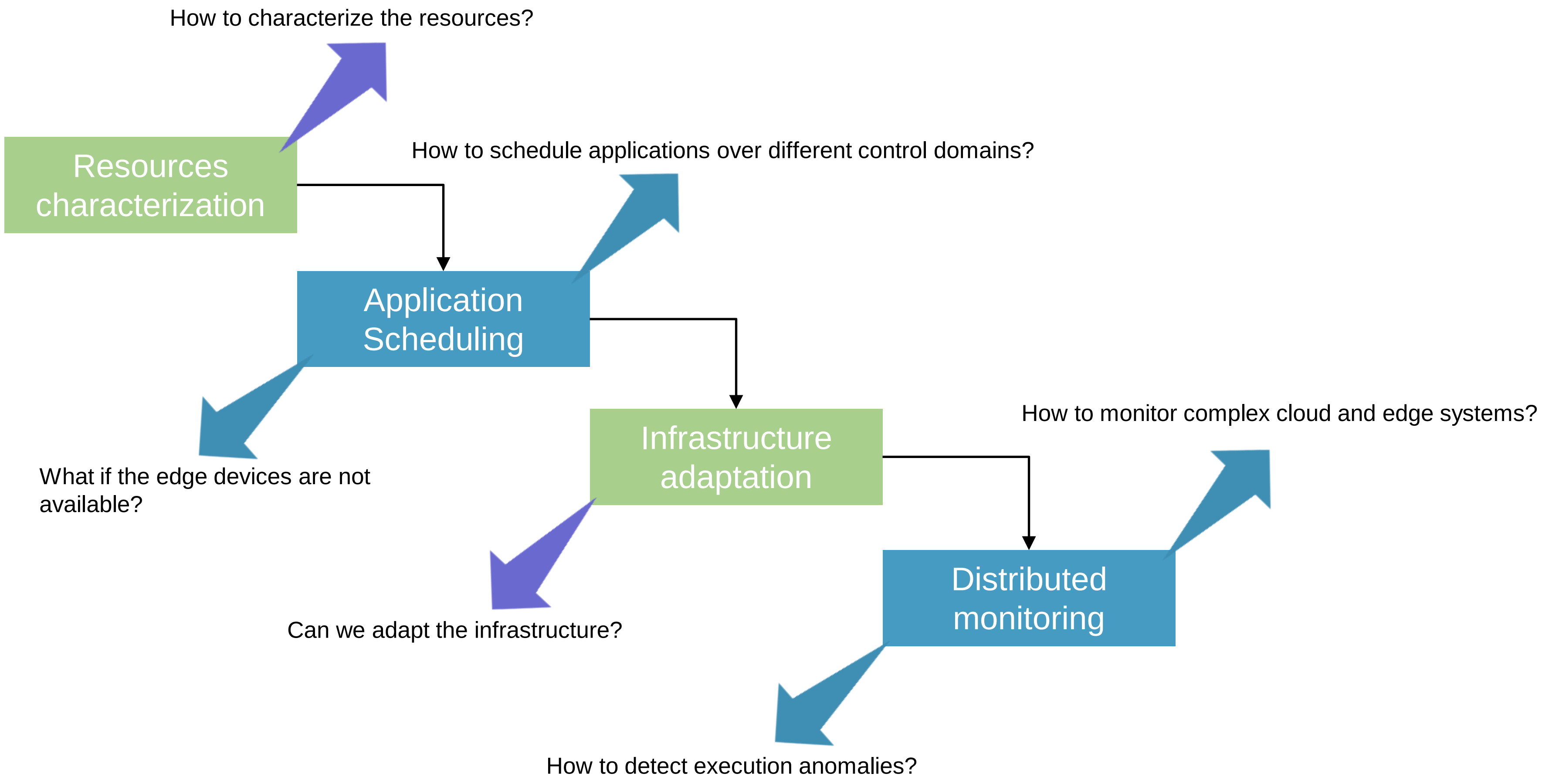
ADA-PIPE provides means for:

- Data-aware scheduling and adaptation

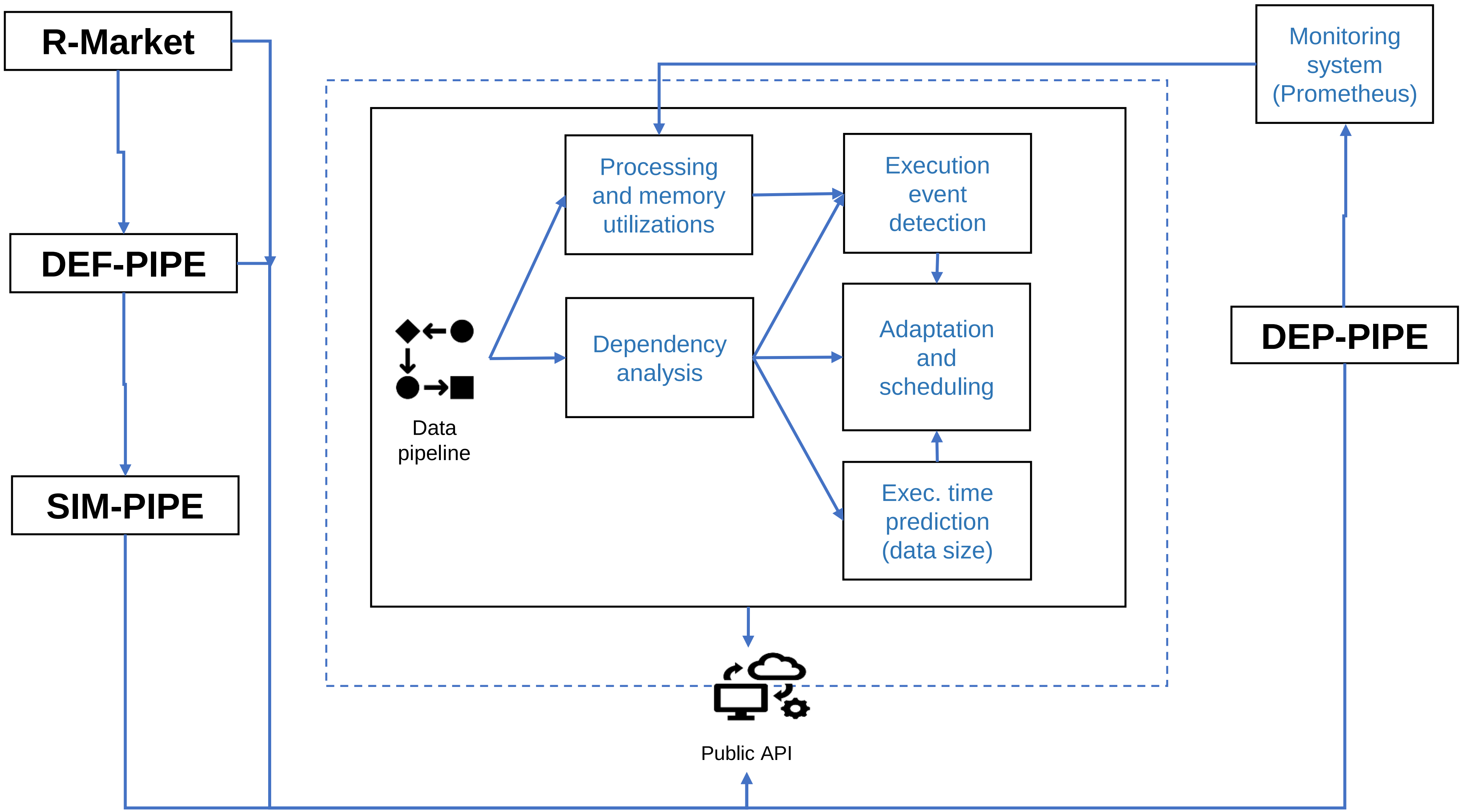
- Continuous automated infrastructure monitoring, analysis, and adaptation



Which functionalities should ADA-PIPE provide?



The ADA-PIPE software design



Collect metrics with Netdata

Netdata as a free and open-source monitoring agent, collects thousands of metrics directly from:

- OS’s of physical and virtual systems,
- IoT/edge devices, and
- containers.

Netdata metrics interact for health monitoring and performance troubleshooting, collected and visualized by

- ***proc.plugin,*** gathers metrics from the /proc and /sys folders in Linux systems.
- ***cgroups.plugin*** collects rich metrics about containers and virtual machines using the virtual files under /sys/fs/cgroup.
- ***ebpf.plugin*** extended Berkeley Packet Filter (eBPF) collector monitors Linux kernel-level metrics for file descriptors, virtual filesystem IO, and process management.

The raw data...

	t-13	t-12	t-11	t-10	t-9	t-8	t-7	t-6	t-5	t-4	t-3	t-2	t-1	t
metric_1	0.53	0.14	0.91	0.25	0.52	0.65	0.82	0.48	0.43	0.29	0.73	0.76	0.77	0.78
metric_2	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
metric_d	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Preprocessing the raw data?

	t-13							t-6	t-5	t-4	t-3	t-2	t-1	t
metric_1	0.53							0.48	0.43	0.29	0.73	0.76	0.77	0.78
metric_2	...							...	...	...	...	...	...	...
...	...	...						...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...								...	...		...	...	...	...
...									...					
...									...					
...									...					
...									...	...	...	...	...	...
...									...	...	...	...	...	...
...									...	...	...	...	...	...
...								...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
metric_d	...	...	...	...	...	...	...	...	...	...	...	...	...	...

1

At time **t** we want to answer "How ***strange*** looking is metric_1 right now?"

2

We could just look at the value of metric_1 at time t and try make a decision. For example compare it to the average or median value for metric_1.

But we are interested in also being able to find "strange **patterns**" as opposed to just cases where the individual value is oddly large or small.

3

So, instead, we look at the recent values for metric_1 to ask a slightly more expanded question - "How ***strange*** looking are the ***recent values*** of metric_1 right now"

4

These recent values become the basis of how we will try to produce some numbers to quantify the "**strangeness**" of metric_1 at time t.

Feature vector

- A “feature vector” is used for training the ML model along with the prediction:

We first take **differences** for metrics that have trends in their values.

0.73	0.76	0.77	0.78	→	0.44	0.03	0.01	0.01

We then **smooth** the values a bit so that things work a bit better with metrics that can tend to be a bit spikey.

0.44	0.03	0.01	0.01	→	-0.02	0.07	0.09	0.12

This is the final “**feature vector**”.
So Netdata anomaly detection works on a **differenced** and **smoothed** collection of recent measurements.

Anomaly model and parameters

In every time interval, the Netdata generates the most recent feature vector for a metric and produces an **anomaly score** for that metric.

1

ADA-PIPE **trains** a k-means clustering model for each metric based on all the feature vectors in a recent time window (the last 24 hours).

2

The clustering model generates a set of feature vectors based on the training data. By default, two "**cluster centers**" are defined.

3

The anomaly score is the "distance" between the feature vector and the trained cluster centers.

4

We convert this raw distance measure into an **anomaly score** by "normalizing" based on max distance, observed during training. For example, if the max distance during training was 120 and the min was 80 and the distance for most recent vector was 125 then the normalized anomaly score would be (125 - 80) / (120 - 80) ~= 113%. By default anything over 99% is considered anomalous. So, in english, anything as "strange" or stranger as the most strange 1% of observations during training would be considered anomalous.

Training data

-0.02	0.07	0.09	0.12
-0.09	-0.02	0.07	0.09
-0.02	-0.09	-0.02	0.07
0.06	-0.02	-0.09	-0.02

Cluster Centers

-0.39	0.19	-0.09	0.00
0.2	0.3	-0.1	0.2

Feature Vector @ time t

-0.02	0.07	0.09	0.12

Raw Distance @ time t

126.77

Anomaly Score @ time t

118%

Anomaly Bit @ time t

1

Anomaly detector

1

To try and make use of our matrix of 0/1's, we need some logic to determine when we have more than typical 1's in a recent rolling window on our device.

2

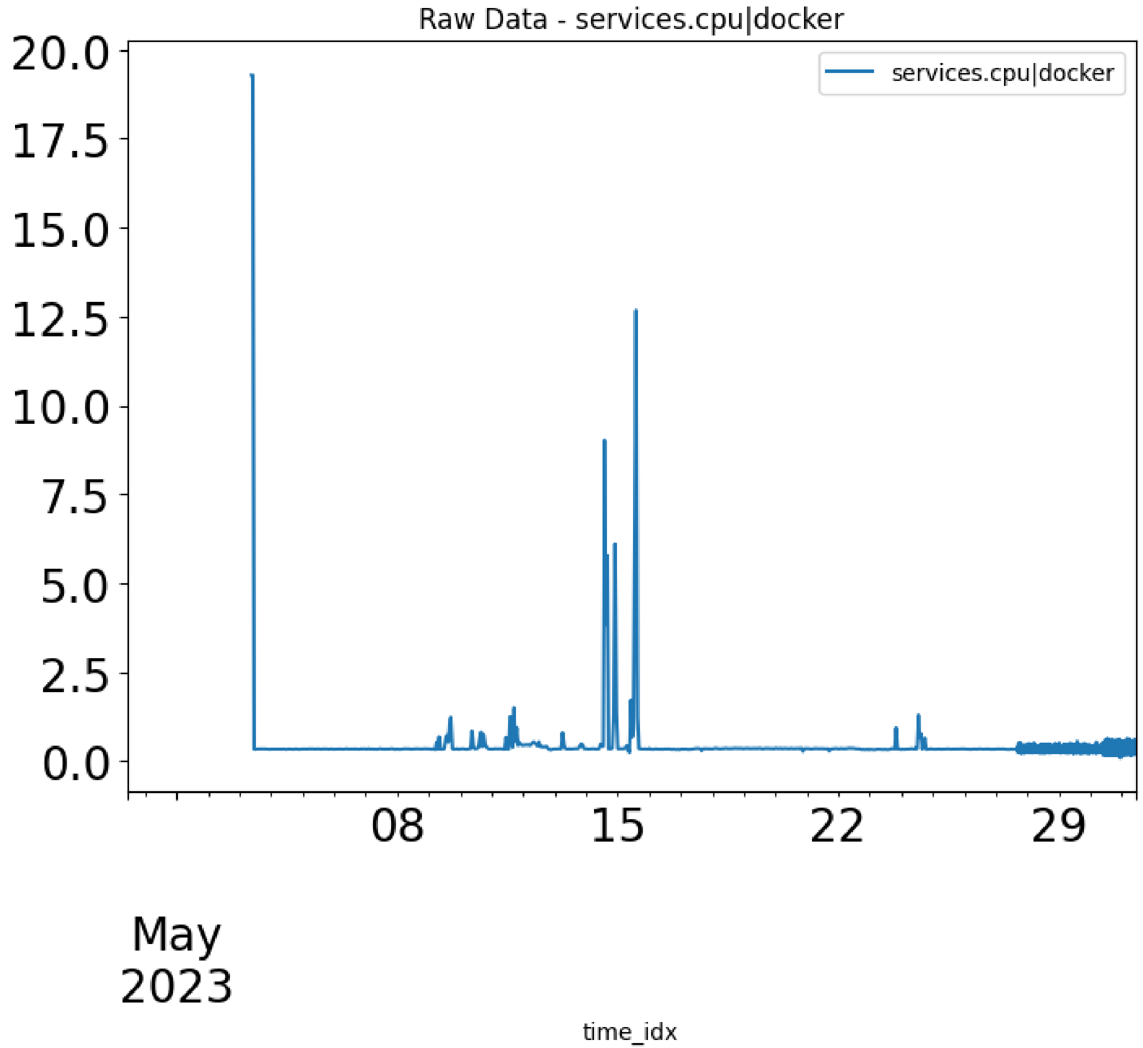
If the Anomaly Rate over all metrics at a specific time step is above a threshold we flag the device itself as anomalous. In the example, we use 10% anomaly rate threshold.

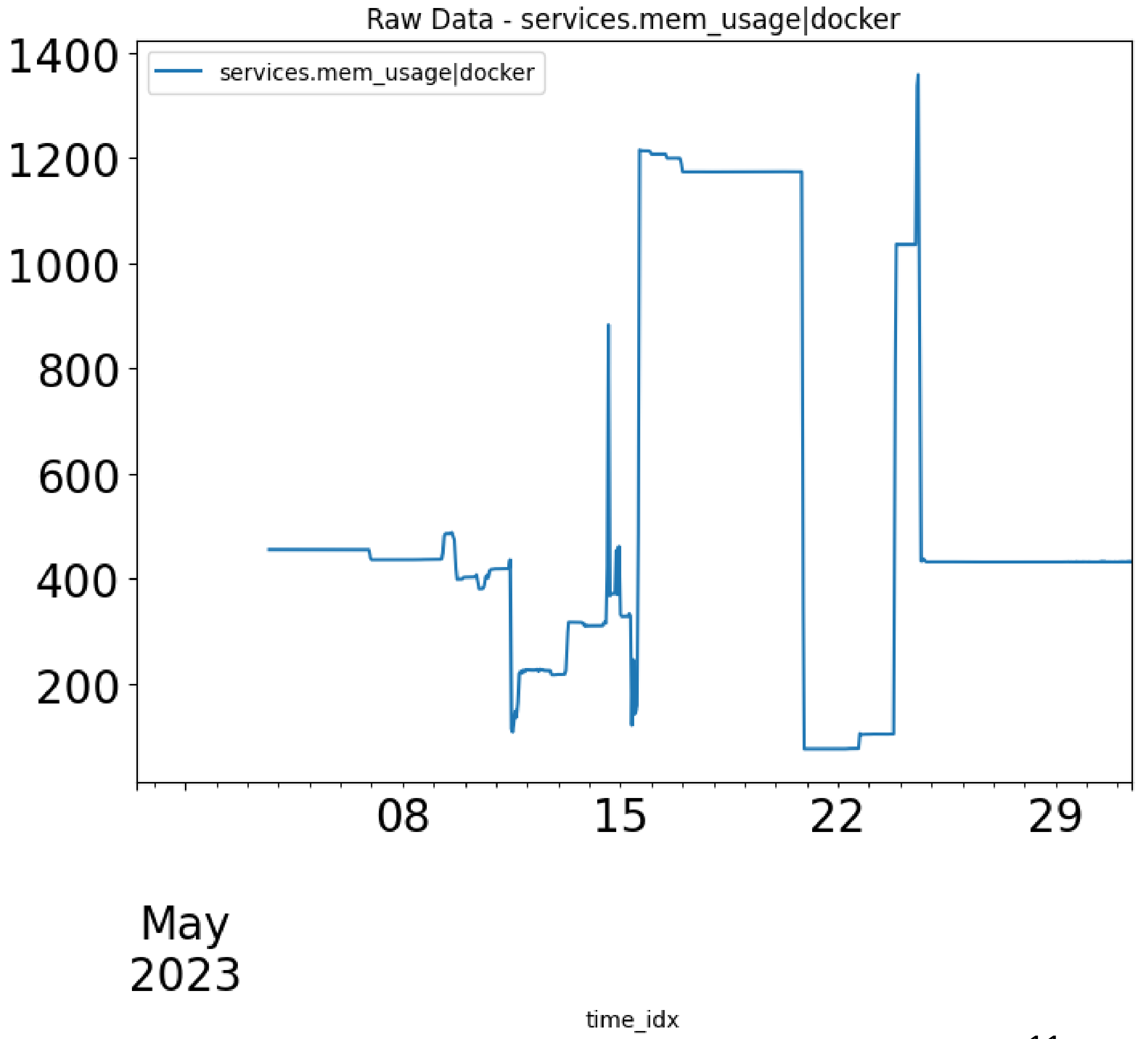
	t-9	t-8	t-7	t-6	t-5	t-4	t-3	t-2	t-1	t
...	0	1	0	0	0	0	0	0	0	0
...	0	1	0	0	1	0	0	0	0	0
...	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	0	0	0	0	0
...	0	0	0	0	0	0	1	1	0	0
...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
Anomaly Rate	0.0%	28.6%	0.0%	0.0%	14.3%	0.0%	14.3%	14.3%	0.0%	0.0%
Device Anomaly	0	1	0	0	1	0	1	1	0	0

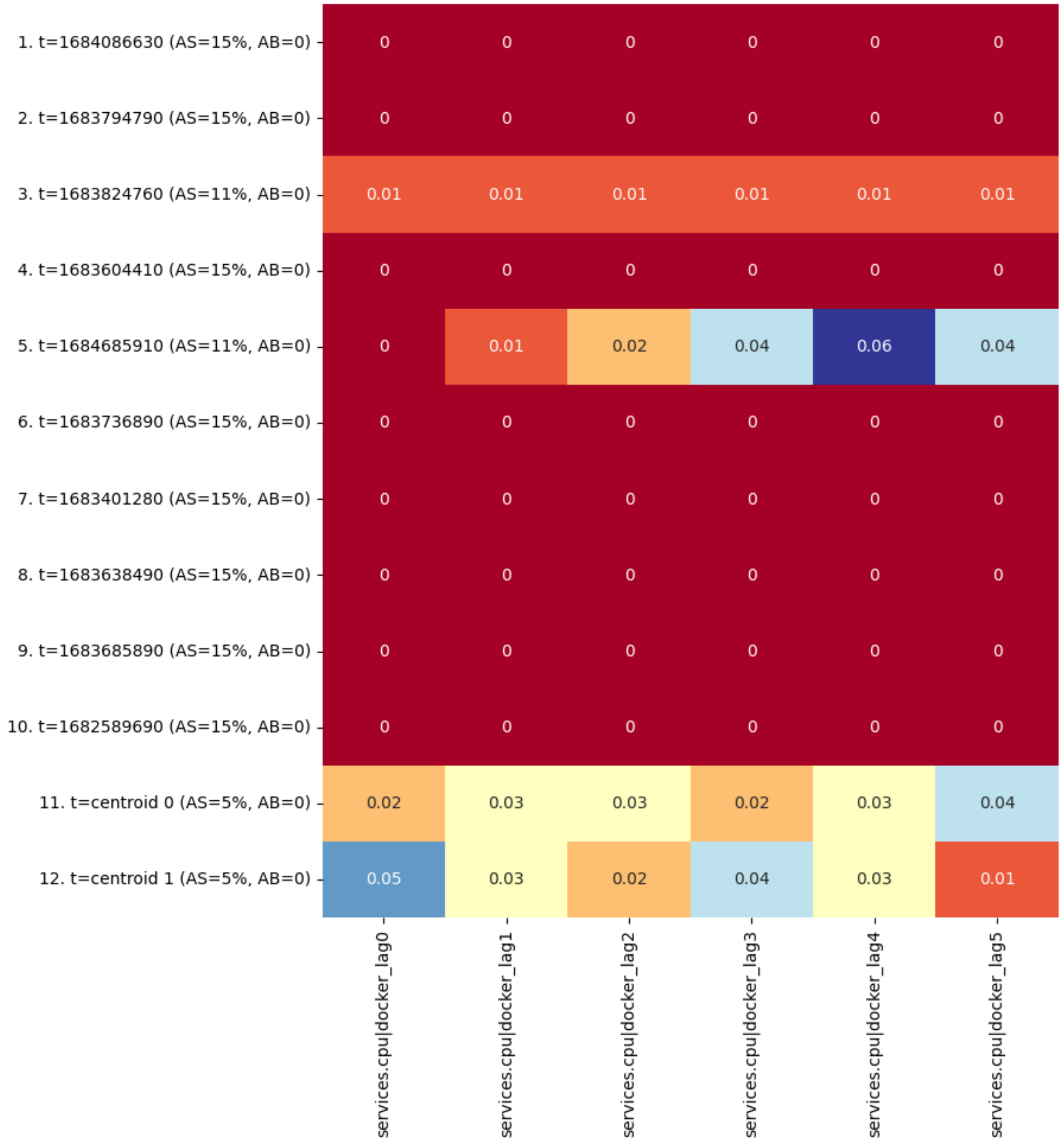
3

The anomaly detector will maintain a rolling window of "Device Anomaly" values. Once the device is detected as anomalous for a period within this rolling window, an anomaly event will be triggered while the device anomaly counter stays above the threshold.

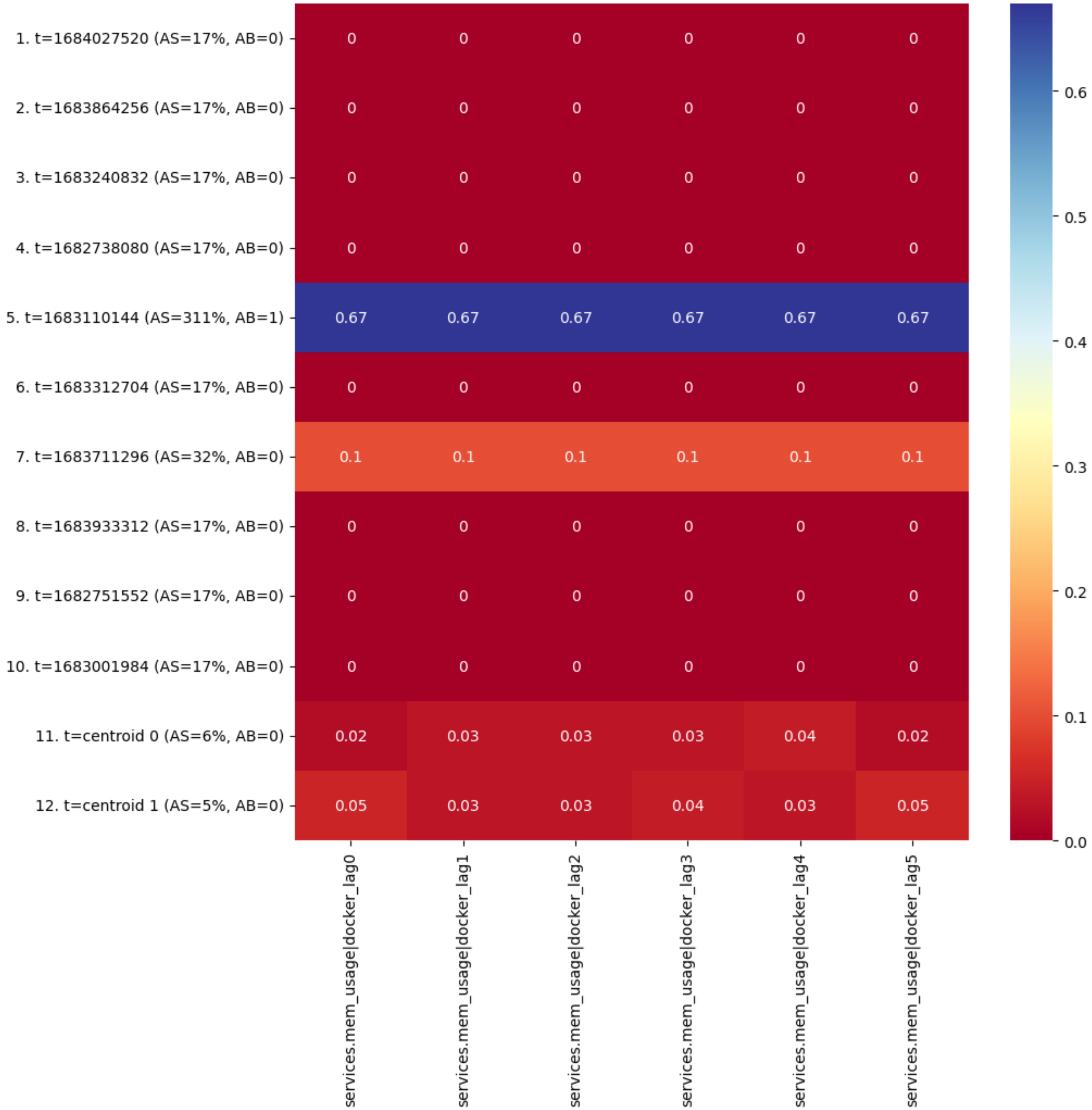
CPU and memory utilization of Docker service



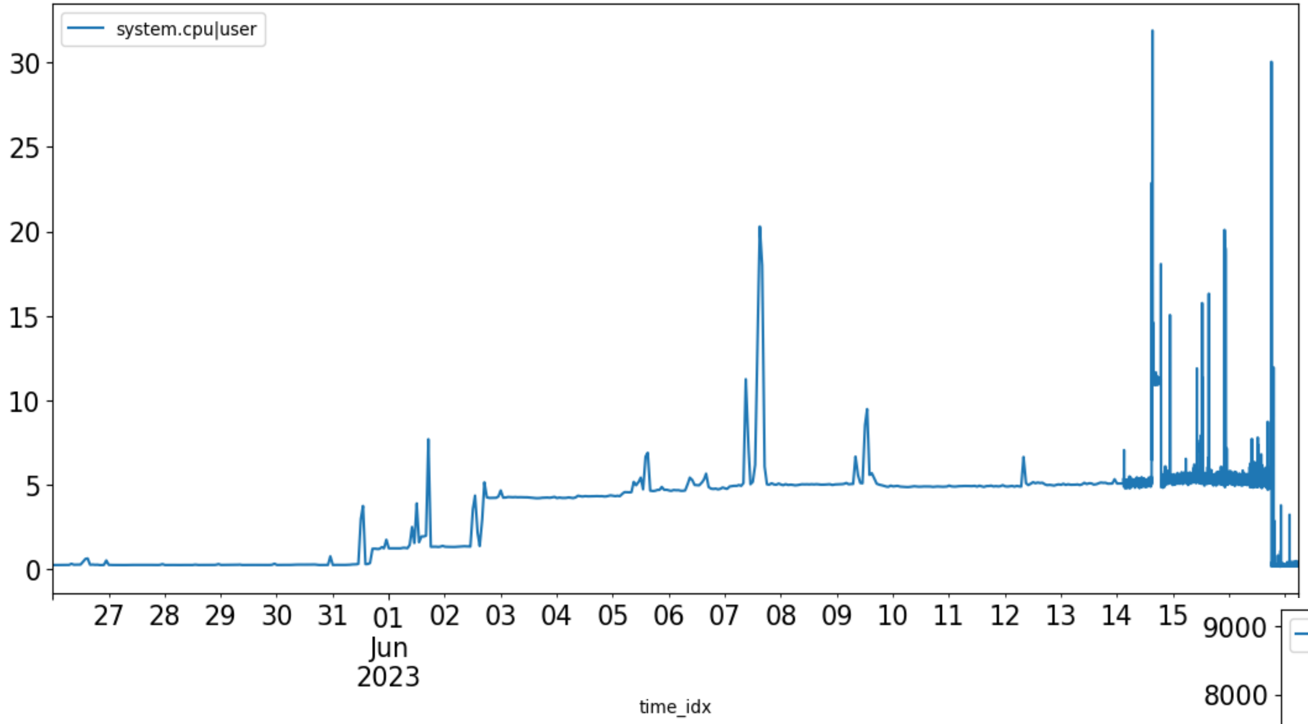


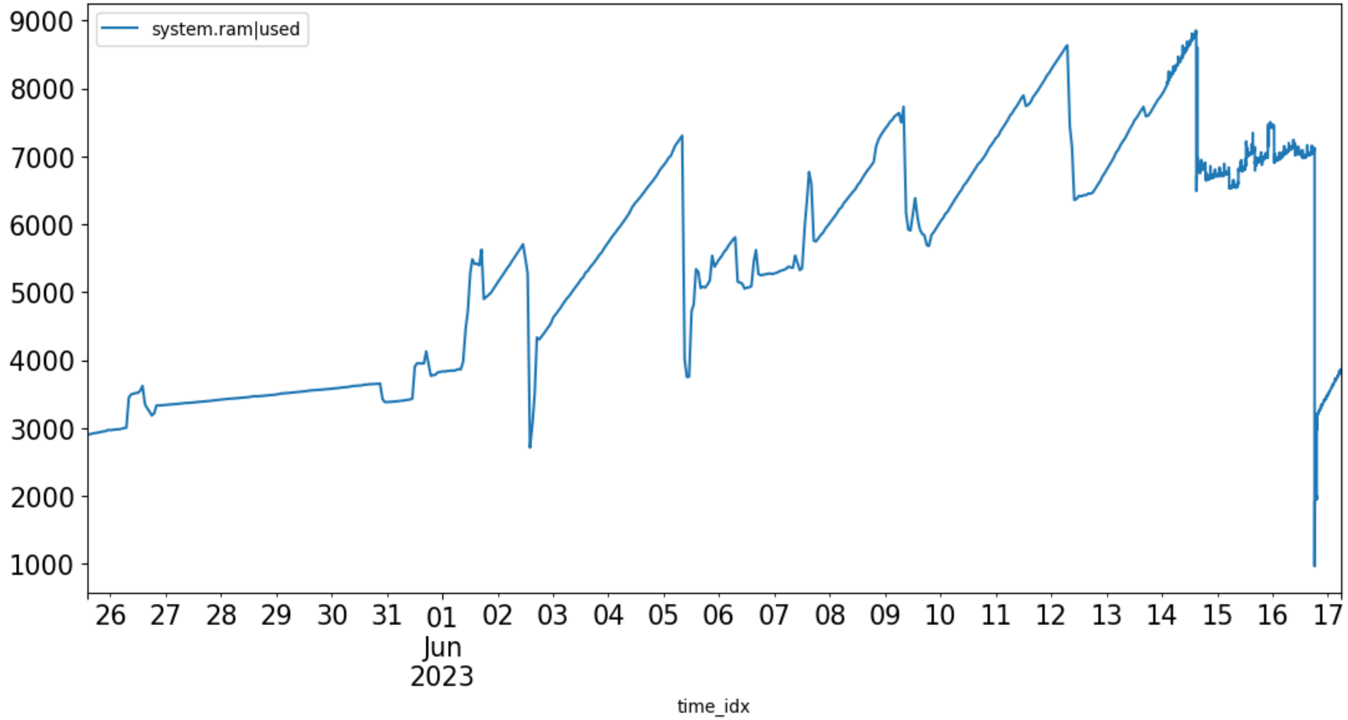


Docker service CPU utilization.

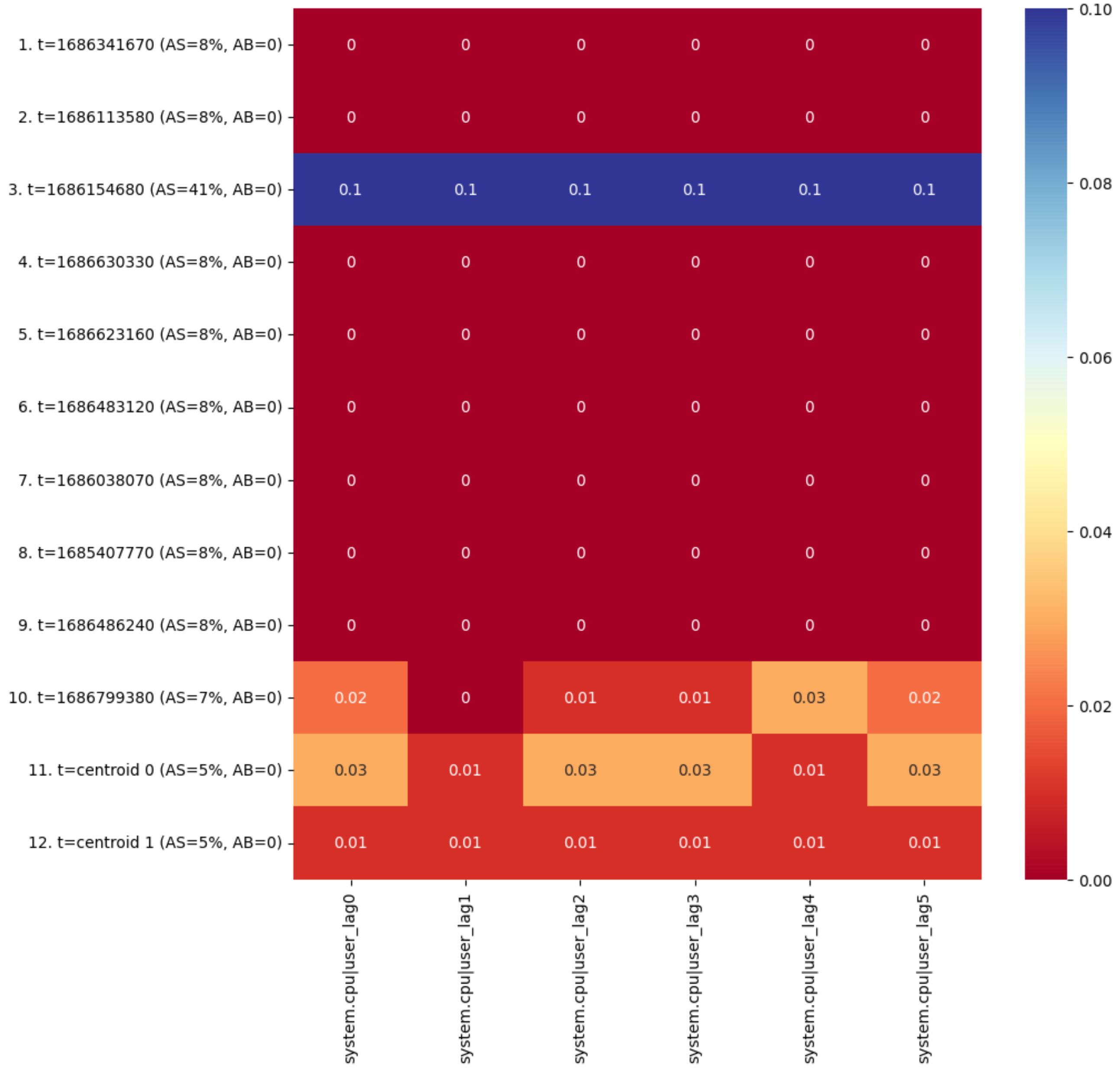


Docker service memory utilization.

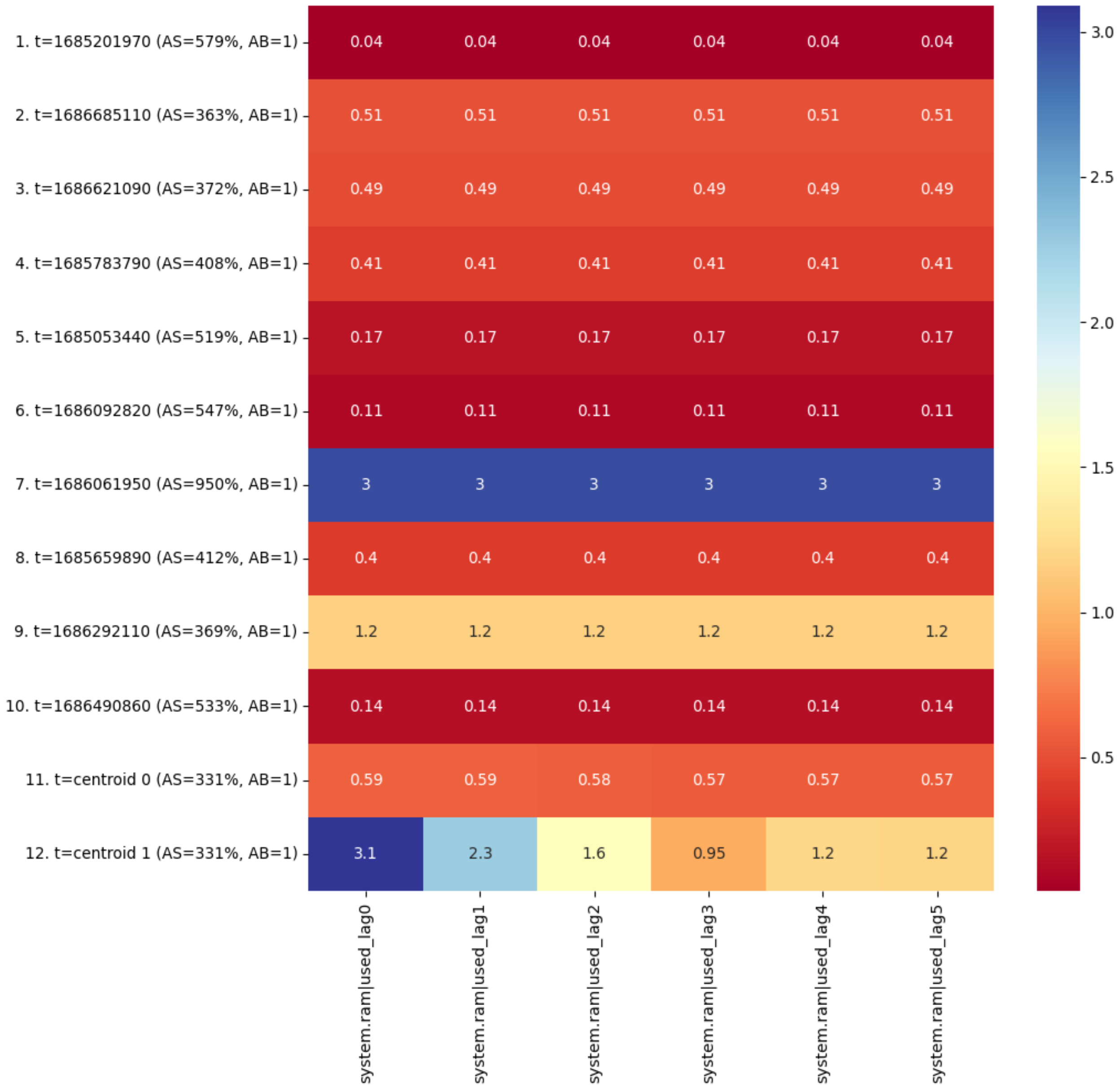




CPU and memory utilization of device



CPU utilization of device.



Memory utilization of device.

THANK YOU!

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101016835, the DataCloud Project.

https://datacloudproject.eu/