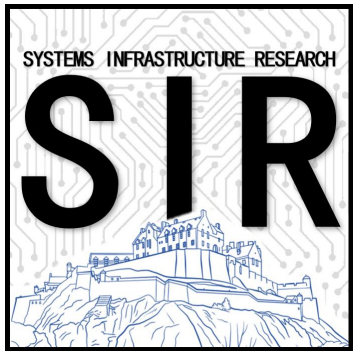


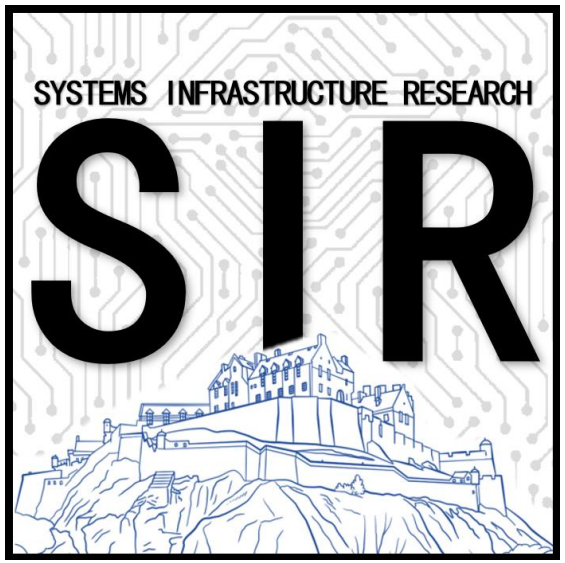
Huawei Cloud Time Series Forecasting Challenge

University of Edinburgh EPSRC CDT in Machine Learning Systems

Artjom Joosen

Introduction

- Systems Infrastructure Research (SIR) Lab
- Improving efficiency of data center scheduling using ML
- Split into 2 sub-groups: systems group and ML group



Topics

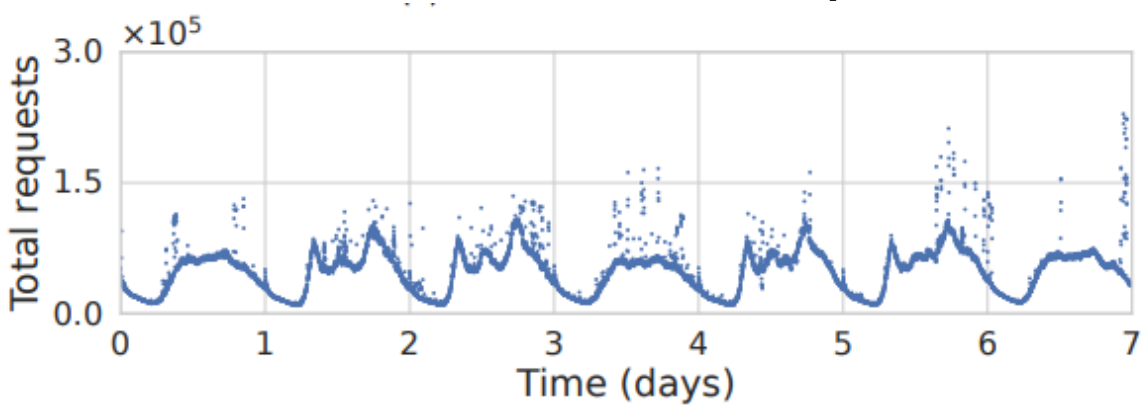
1. Why we need time series forecasting in cloud platforms
2. System architecture
3. The challenge
4. Existing approaches
5. More resources

Handouts

1. Task description document
2. GitHub page: https://github.com/sir-lab/forecasting-challenge/
   - Data download instructions
   - Example forecasting code
   - conda environment
   - Submission format instructions
   - Evaluation method explanation

Time series forecasting in cloud

- Save resources when they are not needed
- Prevent services from slowing down or even breaking during peaks
- Insufficient resources causes *cold starts* (more on that later)



(b) Huawei Public.

Figure 2: Sum of all requests per minute plotted for the first 7 days of Huawei Private and Huawei Public.

How Does It Function? Characterizing Long-term Trends in Production Serverless Workloads, Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, Adam Barker, ACM Symposium on Cloud Computing 2023

Function as a service (FaaS)

- In the past, we have worked a lot on serverless systems in the form of Function as a Service (FaaS)

Function as a service (FaaS)

```
def classify_image(img, args):
  # Image classification code here
  ...

  return result
```

Function as a service (FaaS)

```
def classify_image(img, args):
  # Image classification code here
  ...

  return result
```

request

response

Cloud platform

Function as a service (FaaS)

```python
def classify_image(img, args):
  # Image classification code here
  ...

  return result
```

```python
def preprocess_image(img, args):
  # Preprocessing code here
  ...

  return result
```

```python
def postprocess_result(img, args):
  # Postprocessing code here
  ...

  return result
```

Function as a service (FaaS)

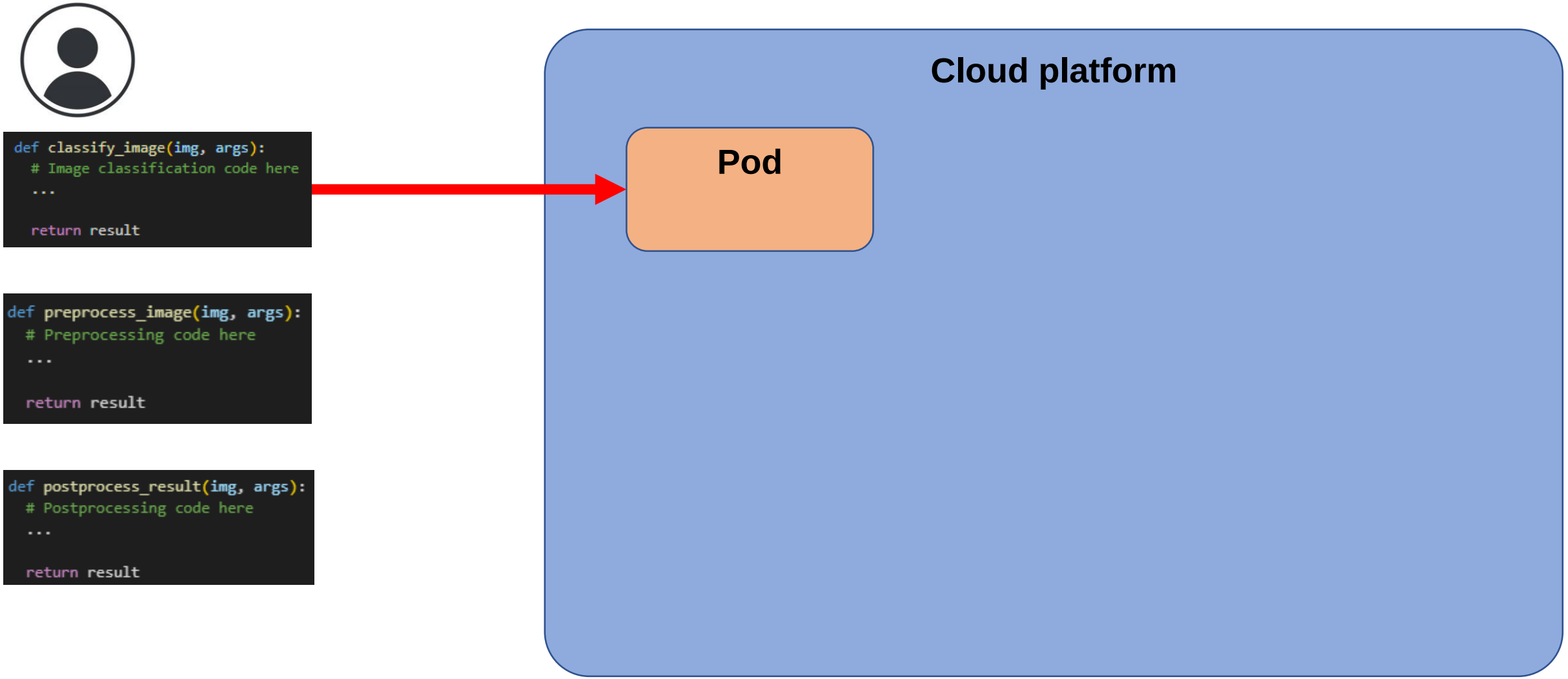
```
def classify_image(img, args):
  # Image classification code here
  ...

  return result
```

```
def preprocess_image(img, args):
  # Preprocessing code here
  ...

  return result
```

```
def postprocess_result(img, args):
  # Postprocessing code here
  ...

  return result
```

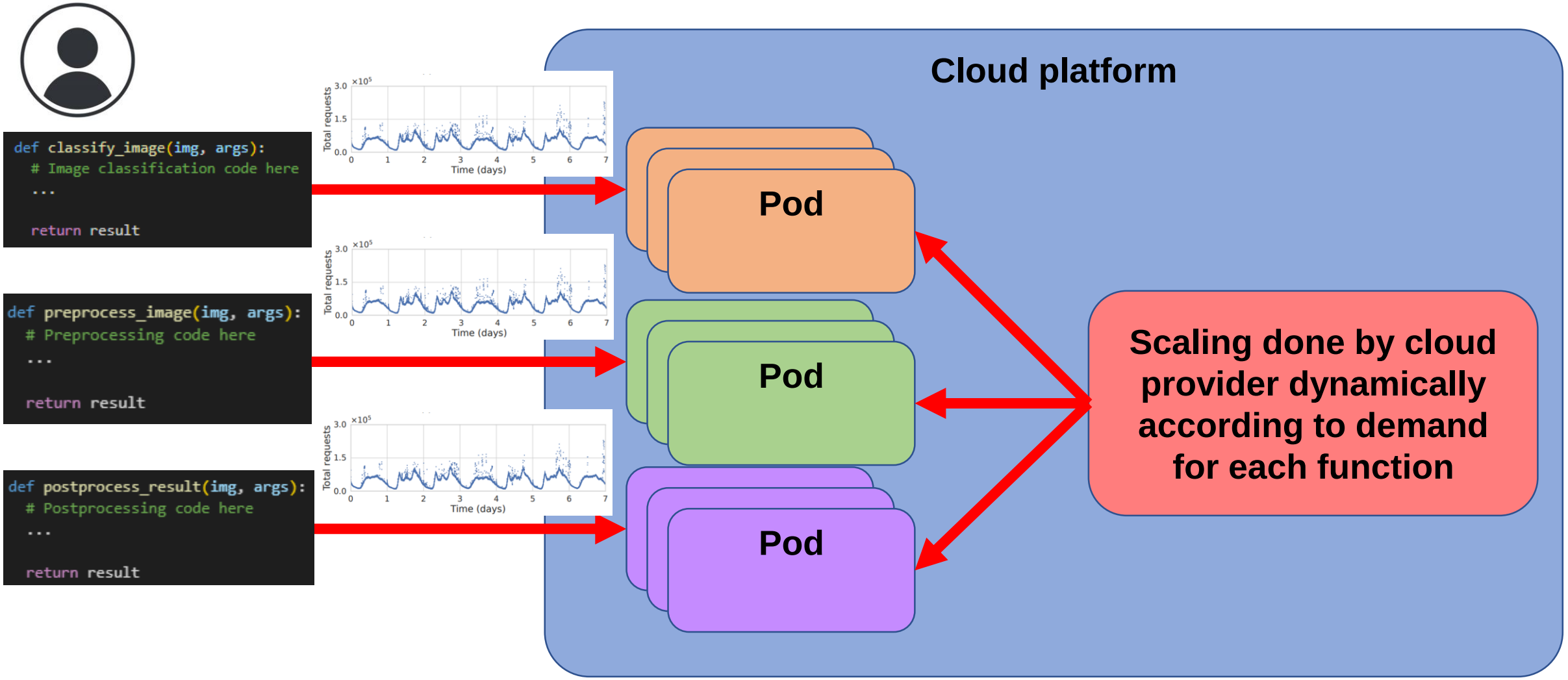
- Developer focuses on developing application
- Infrastructure and scaling managed by cloud provider

Function as a service (FaaS)



A pod as a unit of computing resources, e.g. with 0.3 CPU cores and 128MB memory, that runs on a server

Function as a service (FaaS)



Scaling of pods (and servers) managed by cloud provider

Function as a service (FaaS)

- Serverless setup with function invocation prediction model
- Without model, scaling is handled *reactively* instead of *predictively*
- Under-predicting causes pods to be started from scratch (*cold started*), which is very slow
- Over-predicting wastes resources

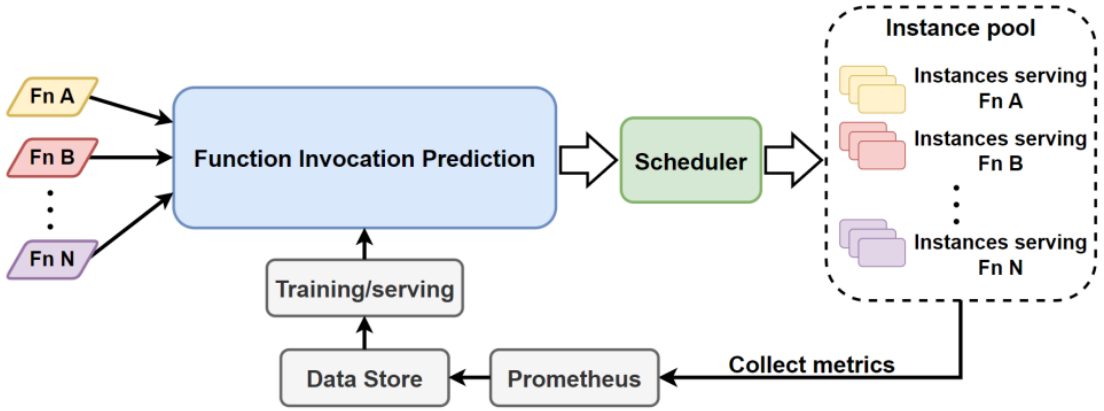
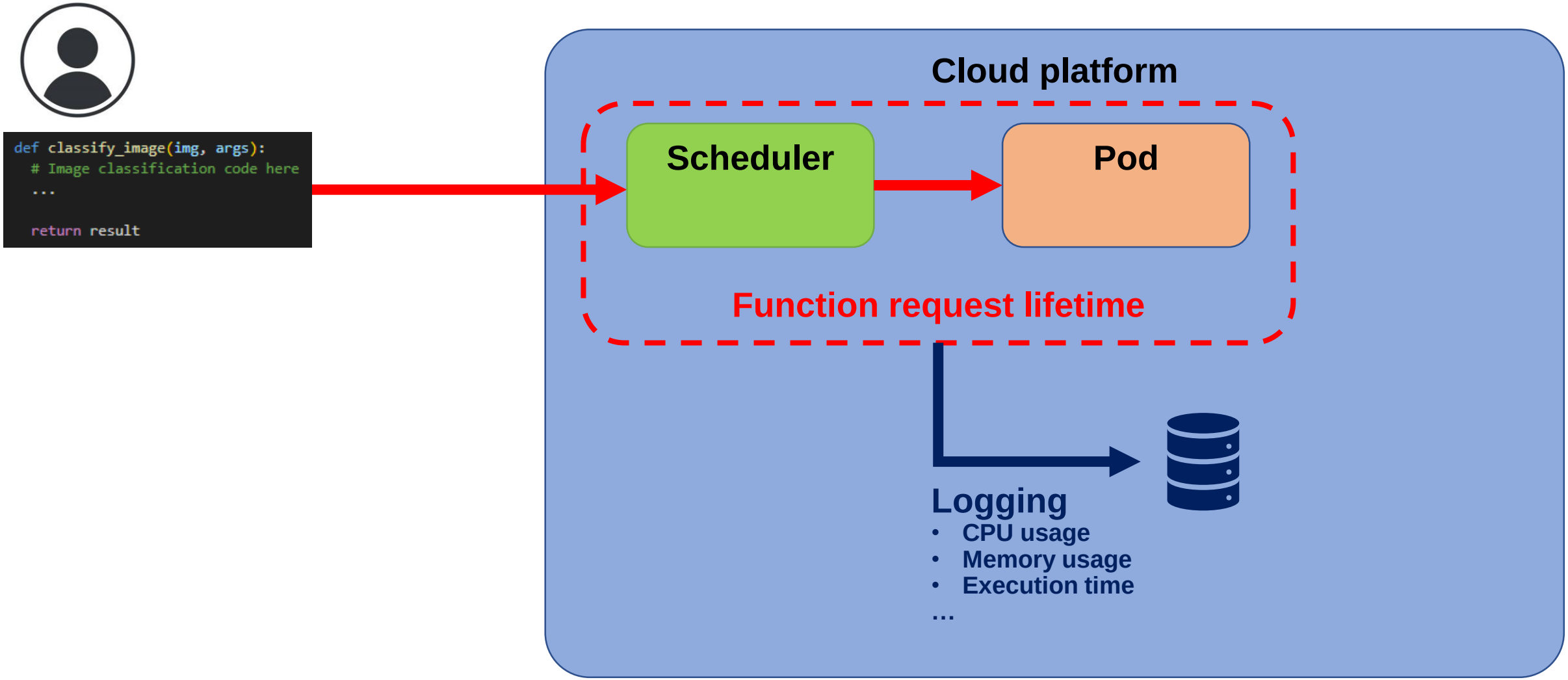


Figure 1: Architecture of our serverless platforms and where our function invocation prediction model will be used.

How Does It Function? Characterizing Long-term Trends in Production Serverless Workloads, Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, Adam Barker, ACM Symposium on Cloud Computing 2023

Function as a service (FaaS)



Many metrics from the function request's lifetime are logged.

Function as a service (FaaS)

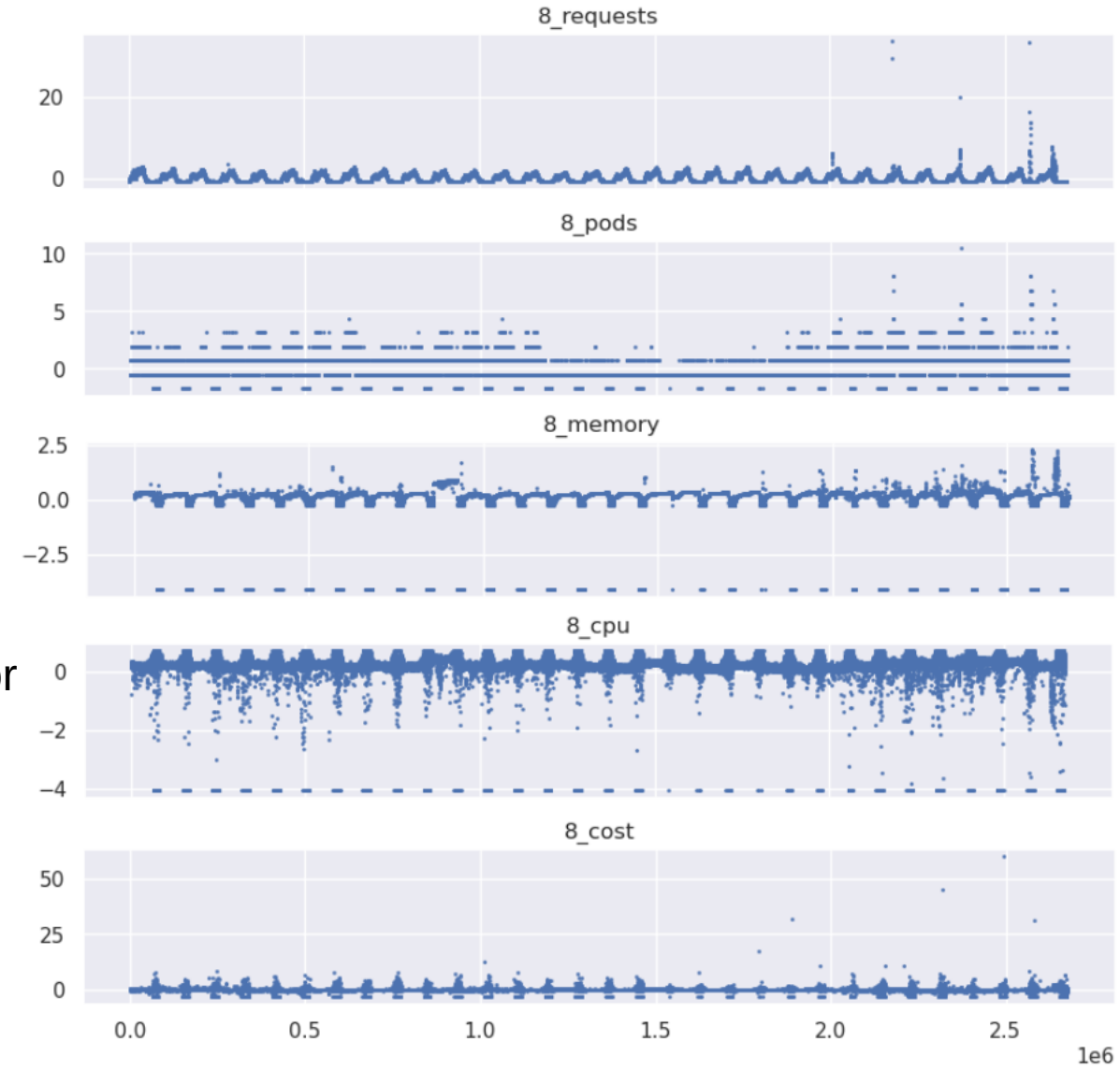
- Logs are stored in a table like this (publicly available on GitHub)

time_worker	time_frontend	requestID	clusterName	funcName	podID	userID	totalCost_worker	workerCost	runtimeCost	totalCost_frontend	frontendCost	busCost	readBodyCost	wri
2592000.0	2592000.0	fc03aaa407f6f5da6912162009d94481bd19ec0425961a757916ab30d1001452	2	1197	pool30-1800-2048-0006942849	197	0.005671	1.19E-4	0.005552	0.006833	1.05E-4			
2592000.0	2592000.0	6c396006bf2afb8c96bc6387547beb3f4a05faf1503ec0e63bc1286c3e09230a	4	1197	pool30-1800-2048-0005816609	197	0.005742	1.56E-4	0.005586	0.007614	8.5E-5			
2592000.0	2592000.002	cabfe5e54f7e6b19678003d1917a712fb75931fd0725cfb08a858b62109580c7	2	1412	pool22-300-128-0000120868	264	0.229177	1.16E-4	0.229061	0.232269	0.00173			
2592000.0	2592000.002	d5a778b667be799ca5d29ed2b014667bc7be2a13d0727cd518d0205067b82054	2	1197	pool30-1800-2048-0006850020	197	0.011188	1.36E-4	0.011052	0.013333	1.31E			
2592000.0	2592000.001	b4513de8a3e990e9b664995bbbf7011726d3a823639c18b77713dd92d479a0a1	1	405	pool24-600-512-0000119311	274	0.037886	1.23E-4	0.037763	0.039219	8.9E-5	8		
2592000.0	2592000.001	d74899e9d7795df490016e473729e6f512c72fbbf37b01883a78066910e8b3b8	4	682	pool22-300-128-0002506259	184	0.306004	1.37E-4	0.305867	0.307853	9.56E-4			
2592000.0	2592000.002	73a9eecc928caab6b2bd7a21ffd72b5539c80cfe0851f63a3d2d7e952b1f5fd5	3	2119	pool28-1400-1536-0005254840	197	0.099046	1.36E-4	0.098911	0.107026	0.004			
2592000.0	2592000.001	df412d55802d53de55d3d8c93e1163f4cbaa756cf61baea50ec33b81e646f367	1	1197	pool30-1800-2048-0004850133	197	0.009275	1.33E-4	0.009142	0.013027	3.75E			
2592000.0	2592000.001	da12ba9769211a1504f0f0ce77a4418f6a142962684f851cd40d1d34ef02372d	3	1197	pool30-1800-2048-0002373972	197	0.007075	1.32E-4	0.006943	0.008538	1.12E			
2592000.001	2592000.003	7cd29f8db77382c1e87c047b5b5a05d19b48ab875e12cac83b64564f2516a09e	2	1797	pool34-3400-4096-0001149049	197	2.723918	1.91E-4	2.723727	2.736723	0.0			
2592000.001	2592000.002	0dd31179483eec5b41fea92b3be59515482905d32001ab3f687aab8af2f651d3	4	1197	pool30-1800-2048-0003694085	197	0.011241	9.9E-5	0.011142	0.013133	2.52			
2592000.001	2592000.001	a34b84c18089c75985c7572b549eb3e51aba5d72a3f7083c311742dfa31985ae	3	1197	pool30-1800-2048-0004458314	197	0.012392	1.58E-4	0.012235	0.015394	2.2			
2592000.002	2592000.002	a287a5b2a77928149c7341deb8ccfac6fc10899cf4412ab2b4724e8bb9a5e5a5	1	1197	pool30-1800-2048-0004671743	197	0.008545	8.4E-5	0.008461	0.010361	1.07			
2592000.002	2592000.003	cafeadeb9705b7b9050052345427c9ad784c3ffeec2569605ed68330f0cb8ea9	1	1538	pool28-1400-1536-0000976295	197	0.673111	2.23E-4	0.672887	0.680231	0.0			
2592000.002	2592000.003	bd90cc8418ad2c1a30c10e78022dd29f50fa76d9205c5f670c5762d0a1562ca9	2	1197	pool30-1800-2048-0004030227	197	0.006746	1.22E-4	0.006624	0.008783	3.3			
2592000.002	2592000.003	44f03f85fce6d6d6bfc0aaae463d68d527d5d9653dd9da7ce3103a8dc518bf4a	1	1197	pool30-1800-2048-0001261446	197	0.010016	1.08E-4	0.009908	0.011437	3.5			
2592000.002	2592000.004	277509a2a75d29f0d5c5bd08571fb367a0cc5bd13cf687c92d642021c5ac40fe	2	1197	pool30-1800-2048-0000486632	197	0.010442	7.7E-5	0.010365	0.012605	1.57			
2592000.002	2592000.003	3ca8e8e8b6d3d4d16782b0139a6c594756f17f3f63d1a68c673070cf22f01b50	2	1197	pool30-1800-2048-0006673389	197	0.008367	1.64E-4	0.008203	0.009712	1.0			
2592000.003	2592000.003	f43074053f669601ba6582b7e605e4f4c17f45e634cacbcf675774e5607cc2e1	4	482	pool26-1000-1024-0006288063	602	5.21712	1.53E-4	5.216967	5.223274	0.001			

- We can then compute aggregate statistics per minute, e.g. number of requests per minute, avg. CPU utilization, etc.

Data you will be working on

- 48 functions x 5 metrics per function = 240 time series
  - Data is 31 days long at per-minute granularity
  - Data has already been normalized
- Schema
  1. **i_requests:** Number of invocations per minute for function i
  2. **i_pods:** Number of pods serving requests for function i in that minute
  3. **i_memory:** Mean memory usage across all pods running function i in that minute
  4. **i_cpu:** Mean CPU usage across all pods running function i in that minute
  5. **i_cost:** Mean execution time of all requests from function i in that minute



Your task

- Predict 240 time series at 3 different horizons
  - 1 day
  - 3 days
  - 7 days
- You will be evaluated by your *average rank* by MAE per horizon on a held-out test dataset.
- For each of the 240 time series, we will rank each group by MAE. This gives us 240 ranking values per group. We will then compute the average rank of group 1, 2, 3.
- Highest ranked group wins a prize
  - Not sure what it is yet, but it is worth fighting for
- There will be a prize per forecasting horizon (3 prizes to be won total)

Your task

- Detailed instructions of submission format on GitHub:
  - https://github.com/sir-lab/forecasting-challenge
- GitHub also includes example code

Deployment considerations

- A model would be deployed in a container with CPU only.
- In your final presentation you should include a section that describes:
  - How long did it take to train your model? Did you use a GPU?
  - How long does it take to do inference for one time series? For the whole dataset? How much memory does it take for inference on CPU?
- You will not be ranked on this, but you should include it.

Time series forecasting

- How do you forecast 240 time series?
  - One model per time series?
    - Ok, but you have to store 240 models. Maybe you can do it if they are lightweight.
  - One model for all of the time series?
    - Ok, but you have to make sure it still predicts individual time series well enough. Maybe you can use a larger neural network that can adequately capture the trends in many time series.
  - One model per metric? e.g. one model for all ‘requests’ time series, one model for all ‘pods’ time series, etc.
    - Maybe, but above considerations still apply.
  - Clustering time series, e.g. simple models for simple time series and complex models for complex time series
    - Ok, but then you have to store all those different models again. If you have a new time series, how will you predict it?

More data and prior work by our lab

- We have publicly released thousands of time series and hundreds of gigabytes of raw logs
  - https://github.com/sir-lab/data-release
- You may find this useful for training your models.
- Papers by our lab:
  1. *How Does It Function? Characterizing Long-term Trends in Production Serverless Workloads* (ACM SoCC 2023) - Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, Adam Barker
     - Explains how forecasting is done in Huawei. See link above for data release.
  2. *Serverless Cold Starts and Where to Find Them* (EuroSys 2025) - Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, Qiwen Deng, Adam Barker
     - Newer data release from Huawei studying cloud data from 5 regions. The data you will be using is a cleaned version of this data release. See link above for data release.
  3. *DAM: Towards A Foundation Model for Time Series Forecasting* (ICLR 2024) - Luke Darlow, Qiwen Deng, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Artjom Joosen, Adam Barker, Amos Storkey
     - Foundation model for time series forecasting. Can make decent ‘zero-shot’ predictions on unseen time series.
  4. *An Analysis of Linear Time Series Forecasting Models* (ICML 2024) - William Toner, Luke Darlow
     - Analysis of various linear time series forecasting models
  5. *FoldFormer: sequence folding and seasonal attention for fine-grained long-term FaaS forecasting* (EuroSys 2023 MLSys workshop) - Luke Nicholas Darlow, Artjom Joosen, Martin Asenov, Qiwen Deng, Jianfeng Wang, Adam Barker
     - Custom transformer model built to predict periodic data.

We are hiring interns!

Send an email to adam.barker@huawei.com if you are interested in an internship with us!

Potential machine learning topics include:

- Continual/online learning
- Improving performance of LLM serving and training in cloud context
- Time series forecasting
- ML-driven scheduling

Timing is flexible.

Any questions?

Send an email to artjom.joosen@huawei.com