Computing Paradigms for the Next-Generation Computing Landscapes

The 2nd AIoTwin Summer School

Ass. Prof. Dr. Stefan Nastic

17th September 2024

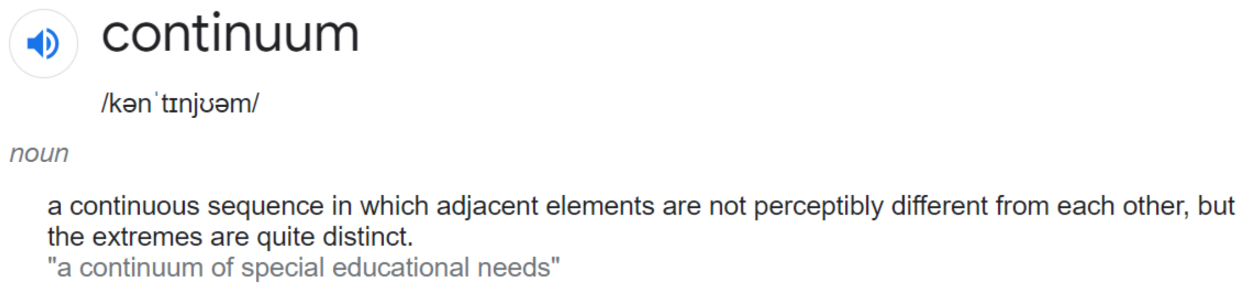
About me

- Assistant Professor at Distributed Systems Group at TU Wien
- Founder and CEO at IntelliEdge GmbH
- Leading various commercial and research projects over the past 10+ years
- My core research interests
  - Serverless Computing
  - Edge-Cloud Continuum
  - AI & Edge AI
- Get in touch via LinkedIn or find me at nastic.at

dsg | DISTRIBUTED SYSTEMS GROUP

IntelliEdge

Edge-Cloud Continuum in a Nutshell



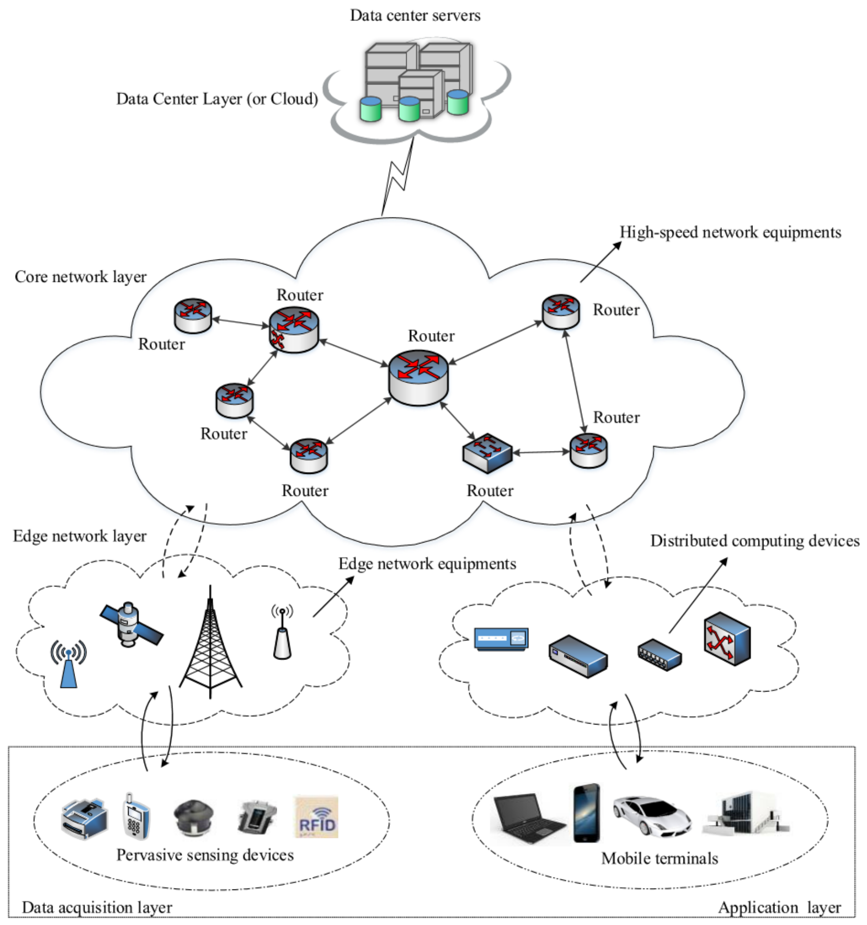
continuum

/kənˈtɪnjʊəm/

noun

a continuous sequence in which adjacent elements are not perceptibly different from each other, but the extremes are quite distinct.
"a continuum of special educational needs"

- A large pool of interconnected resources
- Usually hierarchical/layered and globally heterogeneous



The Promises of Serverless

No infrastructure management

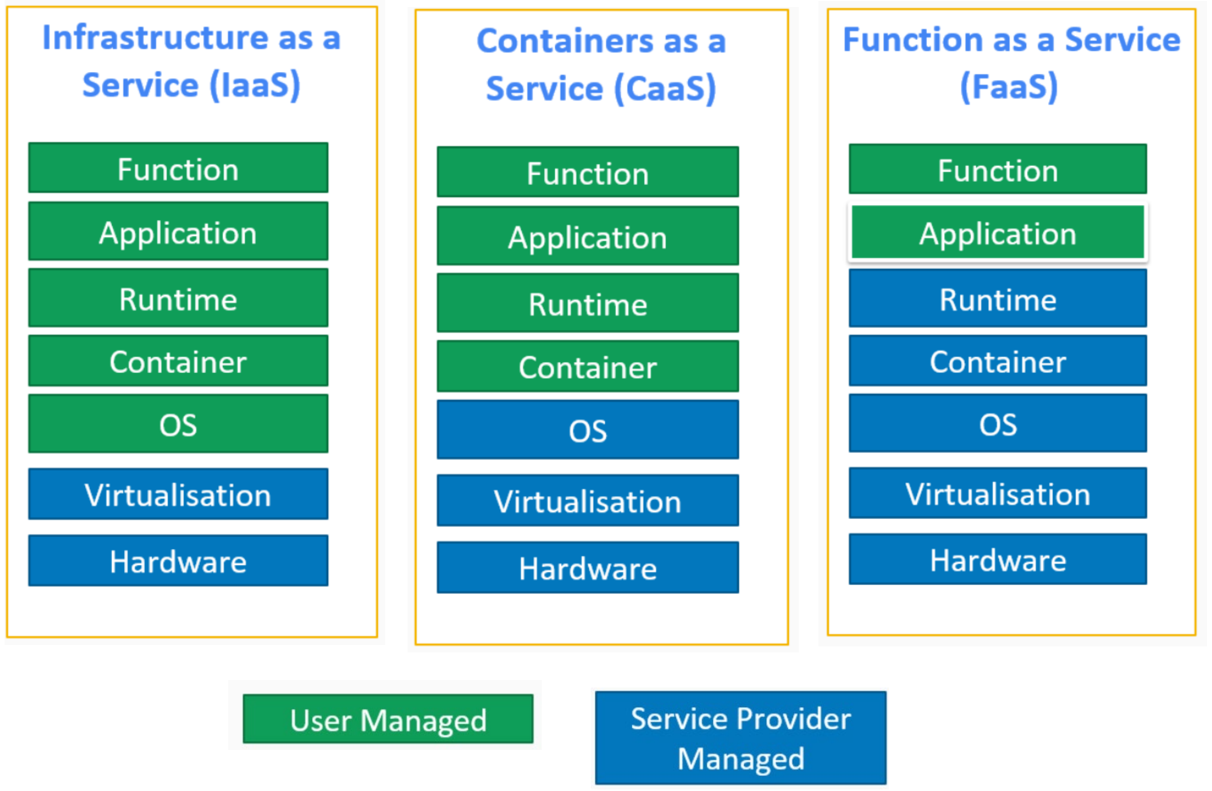
Flexible elastic scaling

High reliability & availability

No idle capacity

Serverless Computing in a Nutshell

- Serverless ≠ No Servers
- Natural evolution in Cloud Computing
- Novel Function as a Service (FaaS) Paradigm
- Novel execution model

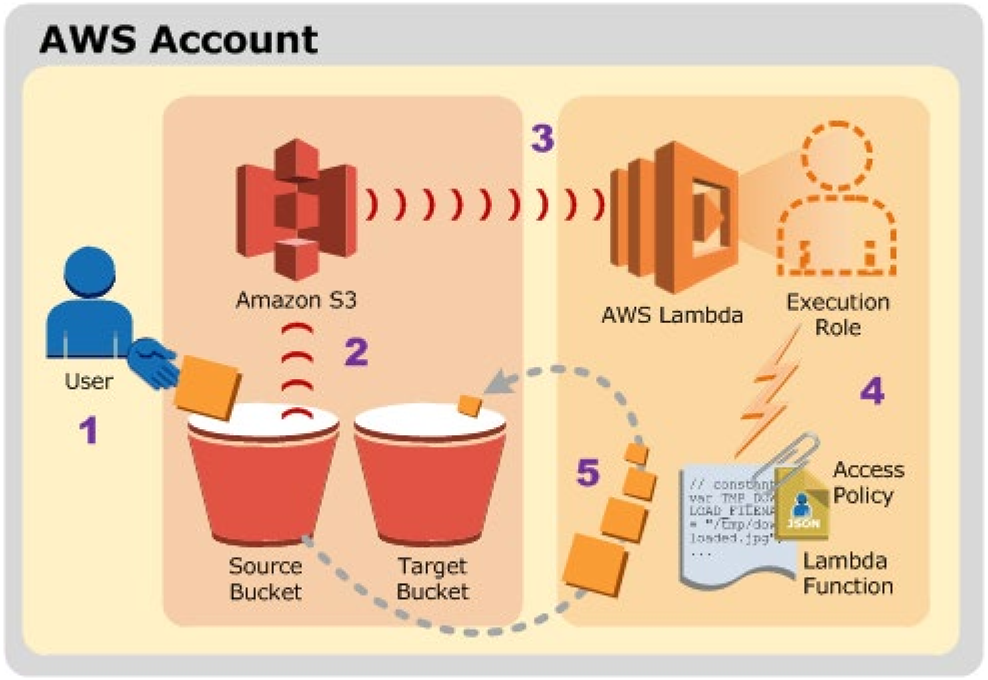


Function Isolation and Virtualization Models

- Containers (K-Native on Kubernetes)
- MicroVMs - Firecracker (AWS Lambda)
- V8 Isolates (Cloudflare, Deno Deploy)
- WASM runtime (Fastly)
- Unikernels (Mainly academia)

Function as a Service (FaaS)

- A type of serverless computing where developers can deploy and run code in response to events or requests
- Application is decomposed into "triggers" and "actions" (functions)
  - Developers write functions and upload them to a FaaS platform
  - Functions are triggered by events, such as a user request or a change in data
- Example: Thumbnail image processing



Popular Serverless Platforms

- Cloud-First
  - AWS Lambda
  - Azure Functions
  - Google Cloud Functions
- Edge-First
  - Cloudflare Workers
  - Fastly Compute@Edge
- Open source
  - OpenWhisk
  - OpenFass

Serverless Beyond the FaaS

Serverless "support BaaS services"

- API Gateways for synchronous (request-based) communication
- Object stores for persistent state management
- Database change streams for reacting to data changes (DynamoDB Streams)
- Function orchestration (Step Function)

T. Larcher, P. Gritsch, S. Nastic and S. Ristov, "BAASLESS: Backend-as-a-Service (BaaS)-Enabled Workflows in Federated Serverless Infrastructures," in IEEE Transactions on Cloud Computing

Serverless managed Cloud services

- Serverless relational database (Amazon Aurora)
- Serverless container management (AWS Fargate)
- Serverless warehouse (Google BigQuery)

Tools
Security
Framework
Hosted Platform
Installable Platform
dapr
CNCF Incubating
nitric
KEDA
Knative
CNCF Incubating
CNCF Incubating

Serverless Beyond the Cloud

- Serverless was born in the Cloud and remains largely limited to the Cloud
- True potential of Serverless is unlocked in the *Edge-Cloud continuum*
  - Localized computation (Edge locations vs datacenters)
  - Low network latency due to proximity to the user
  - Flexible geo-restrictions, compliance, etc.

Serverless Compute Fabric

- Edge-Cloud native BaaS services
- Distributed and decentralized messaging solutions (Emma)
- Lightweight and secure virtualization and isolation (WASM)
- Edge-native state management (Durable Objects)

Nastic, S., Dustdar, S., Philipp, R., Alireza, F., & Pusztai, T. (2022). A Serverless Computing Fabric for Edge & Cloud. In 4th IEEE International Conference on Cognitive Machine Intelligence (CogMi).

The Promises of Serverless

No infrastructure management

Flexible elastic scaling

High reliability & availability

No idle capacity

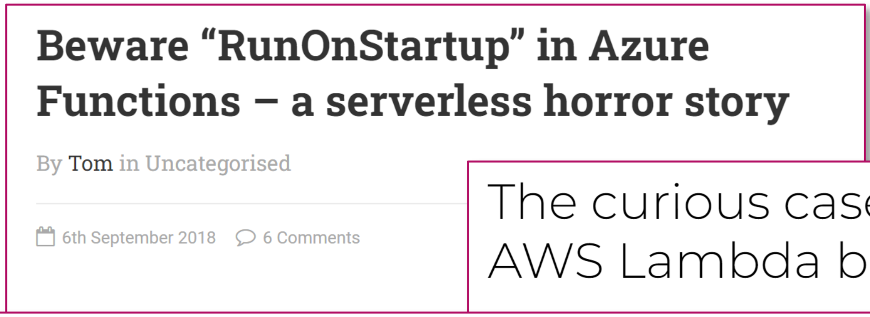
Many Opportunities …

- Fine-grained pay-per-use model with preserved SLOs
- Optimal resource usage with Scale-to-Zero
- Rapid elasticity & auto-scaling due to lightweight virtualization
- Native cloud offloading
- True utility-based resource consumption
- Disaggregation of compute and storage
- Stateless workflows and life-cycle
- Effortlessly parallelizing computing
- Extremely bursty workloads
- CUP-bound applications
- Developing scalable Edge-native solution
- True event-driven applications
- ….

Serverless Horror Stories

Examples from the Wild: A runaway function

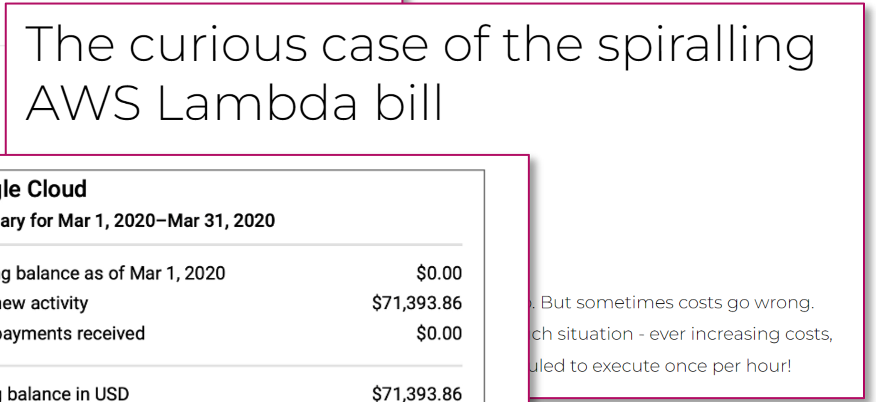
- What is a runaway function?
- There are no default safeguards in place to prevent runaway functions



Beware "RunOnStartup" in Azure Functions – a serverless horror story

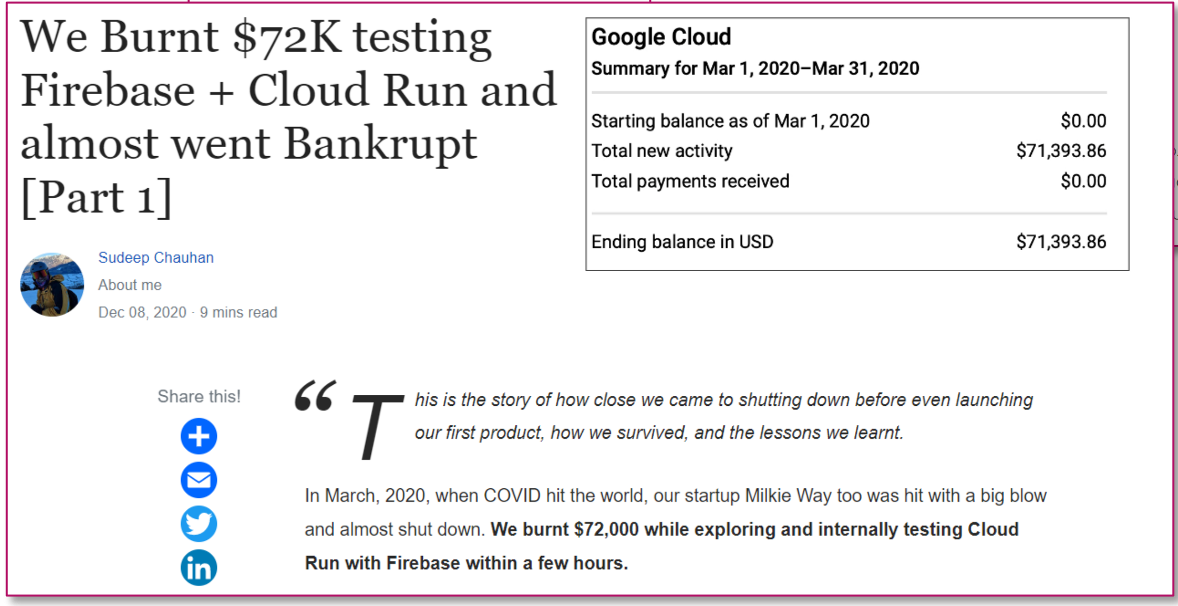
By Tom in Uncategorised

6th September 2018 6 Comments



The curious case of the spiralling AWS Lambda bill

. But sometimes costs go wrong.
ch situation - ever increasing costs,
uled to execute once per hour!



We Burnt $72K testing Firebase + Cloud Run and almost went Bankrupt [Part 1]

Sudeep Chauhan
About me
Dec 08, 2020 · 9 mins read

Google Cloud
Summary for Mar 1, 2020–Mar 31, 2020

Starting balance as of Mar 1, 2020	$0.00
Total new activity	$71,393.86
Total payments received	$0.00
Ending balance in USD	$71,393.86

Share this!

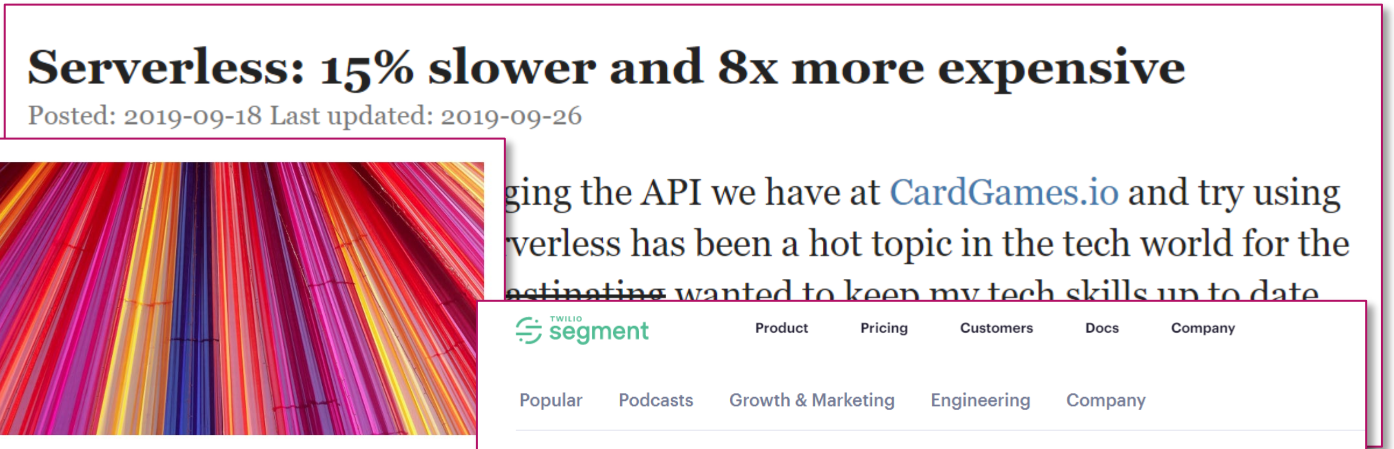
"This is the story of how close we came to shutting down before even launching our first product, how we survived, and the lessons we learnt.

In March, 2020, when COVID hit the world, our startup Milkie Way too was hit with a big blow and almost shut down. **We burnt $72,000 while exploring and internally testing Cloud Run with Firebase within a few hours.**

Serverless Horror Stories

Examples from the Wild: A service misconfiguration

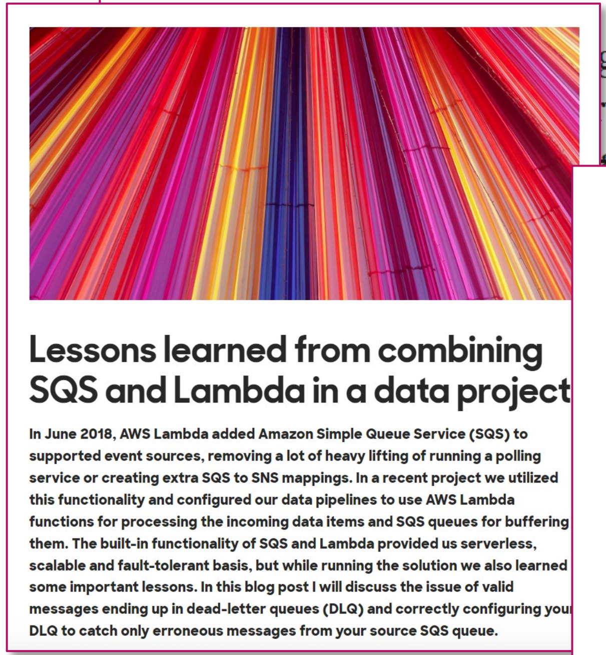
- Serverless functions don't live in isolation ("support services")
- Serverless says nothing about configuring & managing other edge-cloud services
- Arguably more complex ecosystem



Serverless: 15% slower and 8x more expensive

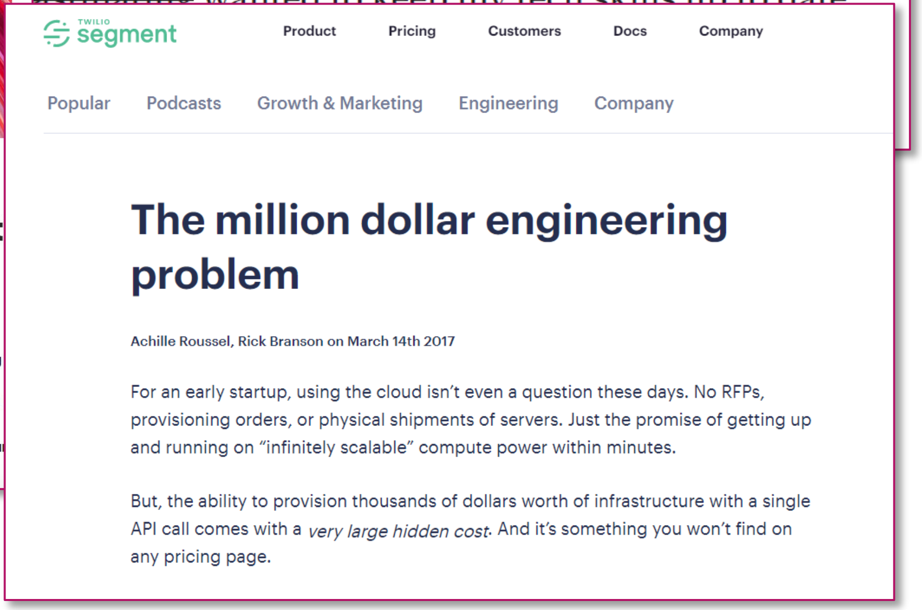
Posted: 2019-09-18 Last updated: 2019-09-26

ging the API we have at CardGames.io and try using
rverless has been a hot topic in the tech world for the
~~estinating~~ wanted to keep my tech skills up to date



Lessons learned from combining SQS and Lambda in a data project

In June 2018, AWS Lambda added Amazon Simple Queue Service (SQS) to supported event sources, removing a lot of heavy lifting of running a polling service or creating extra SQS to SNS mappings. In a recent project we utilized this functionality and configured our data pipelines to use AWS Lambda functions for processing the incoming data items and SQS queues for buffering them. The built-in functionality of SQS and Lambda provided us serverless, scalable and fault-tolerant basis, but while running the solution we also learned some important lessons. In this blog post I will discuss the issue of valid messages ending up in dead-letter queues (DLQ) and correctly configuring you DLQ to catch only erroneous messages from your source SQS queue.



The million dollar engineering problem

Achille Roussel, Rick Branson on March 14th 2017

For an early startup, using the cloud isn't even a question these days. No RFPs, provisioning orders, or physical shipments of servers. Just the promise of getting up and running on "infinitely scalable" compute power within minutes.

But, the ability to provision thousands of dollars worth of infrastructure with a single API call comes with a *very large hidden cost*. And it's something you won't find on any pricing page.

Main Challenges with Serverless in Edge-Cloud Continuum

Performance Challenges

- Startup/scheduling latency ("cold start")
- Lack of performance isolation ("noisy neighbors")
- Inconsistent performance due to hardware heterogeneity (also present in Cloud only solutions!)

Data Management Challenges

- Function execution latency due to lack of data locality
- State management for stateful computations
- Efficient and cost-effective caching solutions
- Particularly important for emerging Edge AI workloads

Nastic, S., Dustdar, S., Philipp, R., Alireza, F., & Pusztai, T. (2022). A Serverless Computing Fabric for Edge & Cloud. In 4th IEEE International Conference on Cognitive Machine Intelligence (CogMi).

Main Challenges with Serverless in Edge-Cloud Continuum

Reliability Engineering Challenges

- Dealing with function failures beyond simple retries
- Network partitioning can render functions useless (detached storage)
- SLO-aware provisioning of functions (beyond memory and CPU)
- Interoperability and portability of functions

Software Development Challenges

- A misconfiguration of services and resources
- Bad coding practices
- Wrong or incomplete error handling mechanism
- Testing of the functions, beyond the unit tests
- Resource quotas & limits

Nastic, S., Dustdar, S., Philipp, R., Alireza, F., & Pusztai, T. (2022). A Serverless Computing Fabric for Edge & Cloud. In 4th IEEE International Conference on Cognitive Machine Intelligence (CogMi).

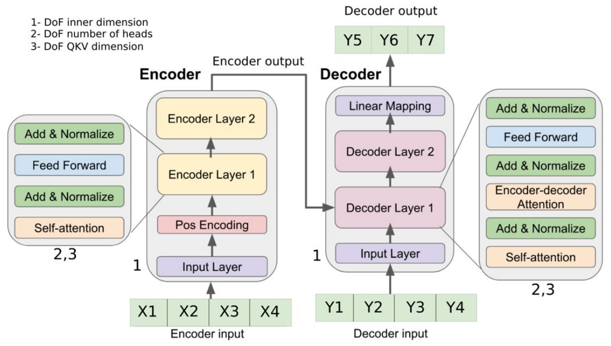
Project Polaris

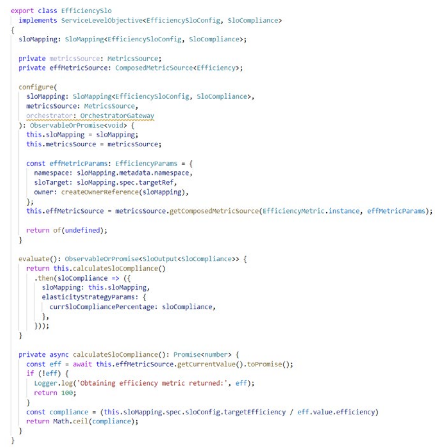
- Enabling SLO- and reliability- awareness in the Edge-Cloud
- Evolved into an ecosystem of tools and framework for the serverless Edge-Cloud continuum
- Polaris is hosted by Linux Foundation as a SIG of a broader Project Centaurus
- Very active GitHub: https://github.com/polaris-slo-cloud

Nastic, S., Morichetta, A., Pusztai, T., Dustdar, S., Ding, X., Vij, D. and Xiong, Y., 2020. SLOC: Service level objectives for next-generation cloud computing. IEEE Internet Computing, 24(3), pp.39-50.

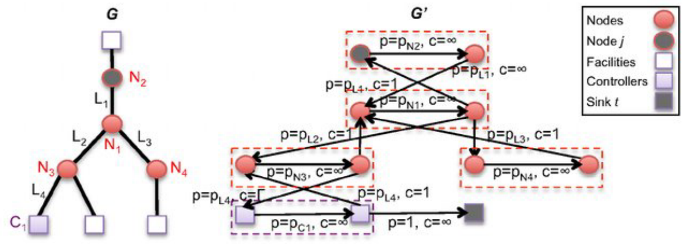
Polaris SLO Framework

Monitoring → **SLO Script** → **Elastic Strategies**





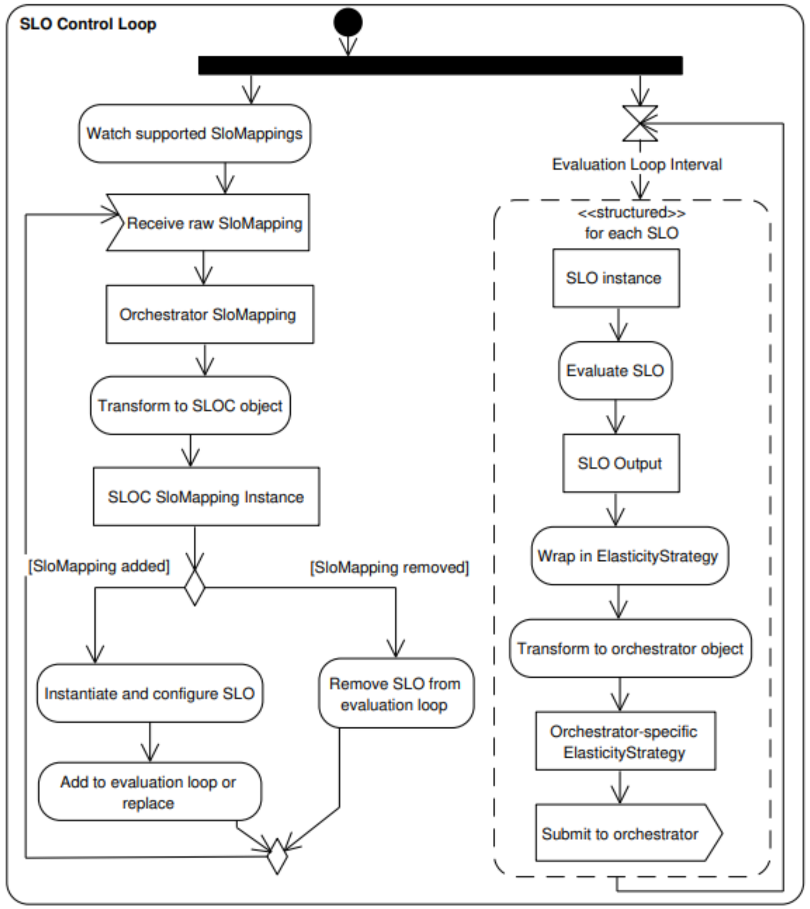
```
export class EfficiencySlo
  implements ServiceLevelObjective<EfficiencySloConfig, SloCompliance>
{
  sloMapping: SloMapping<EfficiencySloConfig, SloCompliance>;

  private metricsSource: MetricsSource;
  private effMetricSource: ComposedMetricSource<Efficiency>;

  configure(
    sloMapping: SloMapping<EfficiencySloConfig, SloCompliance>,
    metricsSource: MetricsSource,
    orchestrator: OrchestratorGateway
  ): ObservableOrPromise<void> {
    this.sloMapping = sloMapping;
    this.metricsSource = metricsSource;

    const effMetricParams: EfficiencyParams = {
      namespace: sloMapping.metadata.namespace,
      sloTarget: sloMapping.spec.targetRef,
      owner: createOwnerReference(sloMapping),
    };
    this.effMetricSource = metricsSource.getComposedMetricSource(EfficiencyMetric.instance, effMetricParams);

    return of(undefined);
  }

  evaluate(): ObservableOrPromise<SloOutput<SloCompliance>> {
    return this.calculateSloCompliance()
      .then(sloCompliance => ({
        sloMapping: this.sloMapping,
        elasticityStrategyParams: {
          currSloCompliancePercentage: sloCompliance,
        },
      }));
  }

  private async calculateSloCompliance(): Promise<number> {
    const eff = await this.effMetricSource.getCurrentValue().toPromise();
    if (!eff) {
      Logger.log('Obtaining efficiency metric returned:', eff);
      return 100;
    }
    const compliance = (this.sloMapping.spec.sloConfig.targetEfficiency / eff.value.efficiency)
    return Math.ceil(compliance);
  }
}
```



Pusztai, T., Morichetta, A., Pujol, V.C., Dustdar, S., Nastic, S., Ding, X., Vij, D. and Xiong, Y., 2021, September. SLO script: A novel language for implementing complex cloud-native elasticity-driven SLOs. In 2021 IEEE International Conference on Web Services (ICWS) (pp. 21-31).

SLO Controllers

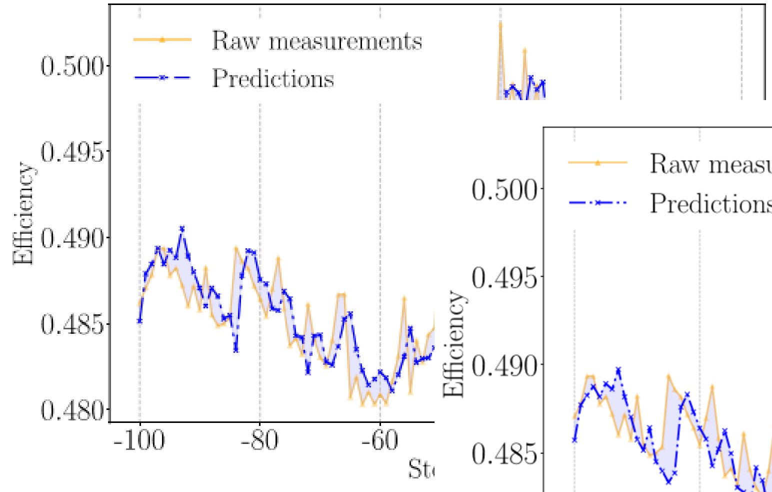
- Developed as Monitoring, SLO, and Scaling Controllers
- Reference implementation extends Kubernetes
- Orchestrator/platform independent control loop



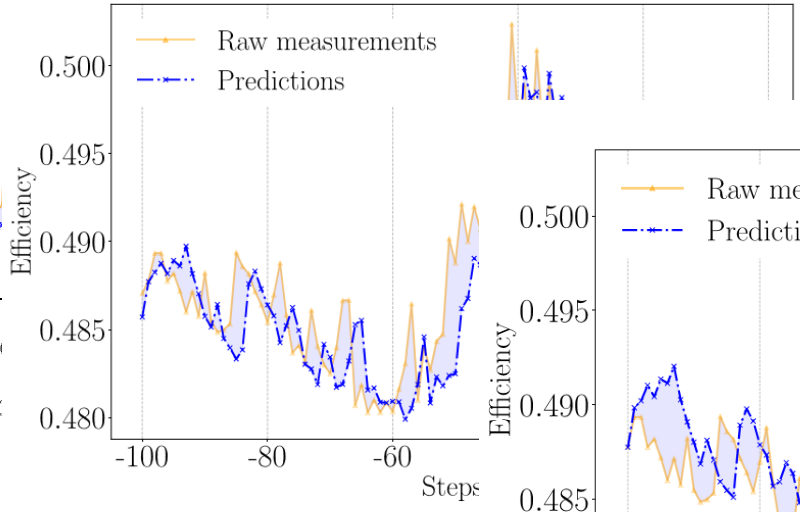
SLO Control Loop
Watch supported SloMappings
Receive raw SloMapping
Orchestrator SloMapping
Transform to SLOC object
SLOC SloMapping Instance
[SloMapping added]
[SloMapping removed]
Instantiate and configure SLO
Add to evaluation loop or replace
Remove SLO from evaluation loop
Evaluation Loop Interval
<<structured>> for each SLO
SLO instance
Evaluate SLO
SLO Output
Wrap in ElasticityStrategy
Transform to orchestrator object
Orchestrator-specific ElasticityStrategy
Submit to orchestrator

Pusztai, T., Morichetta, A., Pujol, V.C., Dustdar, S., Nastic, S., Ding, X., Vij, D. and Xiong, Y., 2021, September. A novel middleware for efficiently implementing complex cloud-native SLOs. In 2021 IEEE 14th International Conference on Cloud Computing (CLOUD) (pp. 410-420).

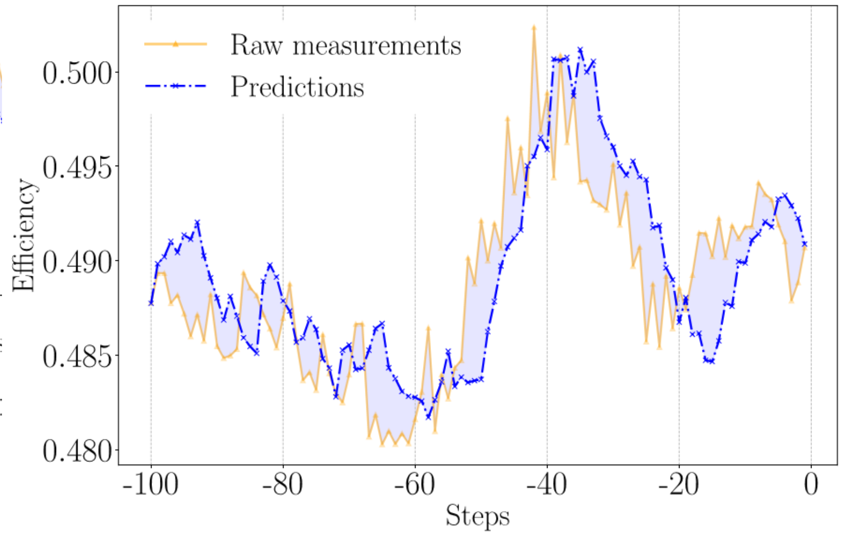
Predictive Monitoring Controllers Results



(a) Predictior



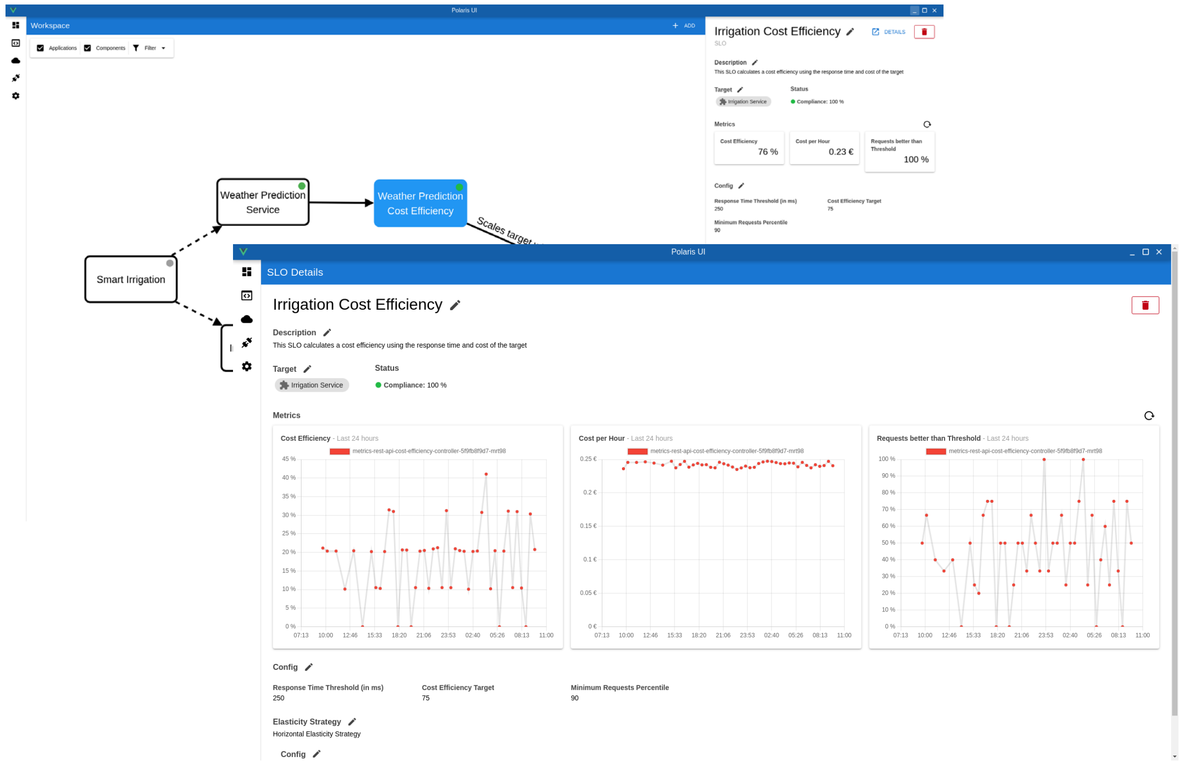
(b) Prediction



(c) Prediction for t+3

SLO User Tool Suite

SLO Framework UI


Polaris UI
Workspace
Applications
Components
Filter
ADD
Weather Prediction Service
Weather Prediction Cost Efficiency
Smart Irrigation
Scales target
SLO Details
Cost Efficiency - Last 24 hours
Cost per Hour - Last 24 hours
Requests better than Threshold - Last 24 hours

Irrigation Cost Efficiency

Description
This SLO calculates a cost efficiency using the response time and cost of the target

Target: Irrigation Service
Status: Compliance: 100 %

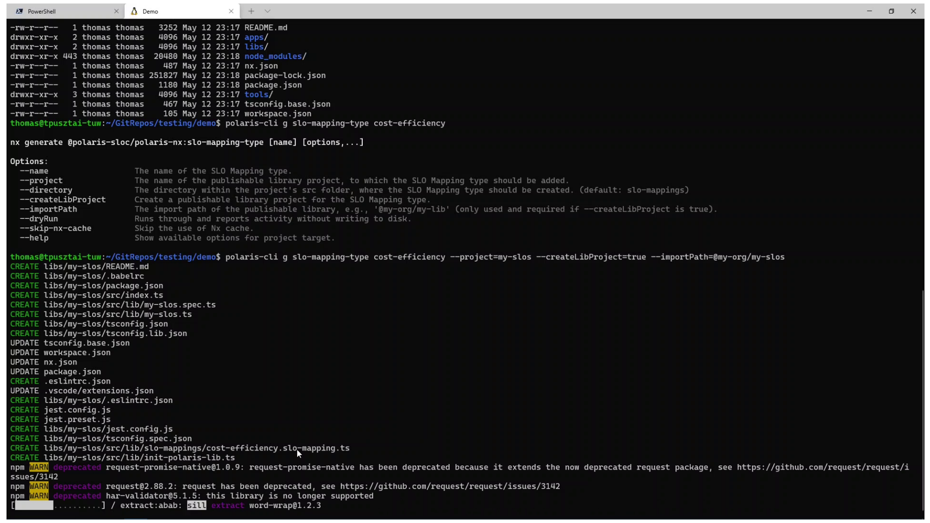
Metrics
Cost Efficiency 76 %
Cost per Hour 0.23 €
Requests better than Threshold 100 %

Config
Response Time Threshold (in ms): 250
Cost Efficiency Target: 75
Minimum Requests Percentile: 90

Elasticity Strategy
Horizontal Elasticity Strategy

Config

SLO Script CLI

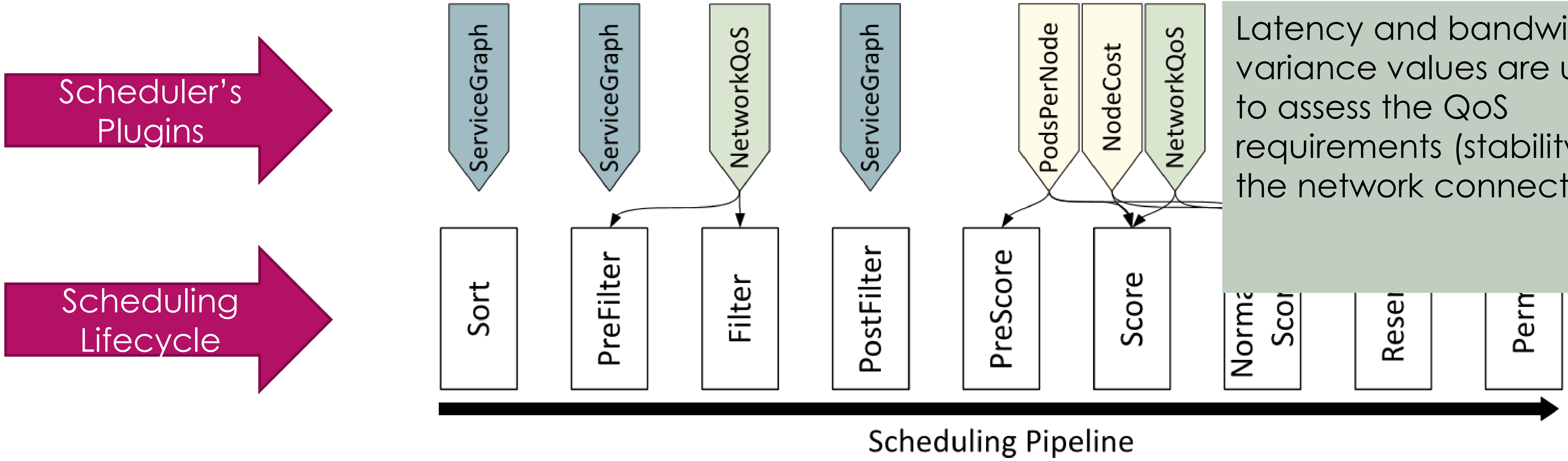


```
-rw-r--r--   1 thomas thomas   3252 May 12 23:17 README.md
drwxr-xr-x   2 thomas thomas   4096 May 12 23:17 apps/
drwxr-xr-x   2 thomas thomas   4096 May 12 23:17 libs/
drwxr-xr-x 443 thomas thomas  20480 May 12 23:18 node_modules/
-rw-r--r--   1 thomas thomas    487 May 12 23:17 nx.json
-rw-r--r--   1 thomas thomas 251827 May 12 23:18 package-lock.json
-rw-r--r--   1 thomas thomas   1180 May 12 23:18 package.json
drwxr-xr-x   3 thomas thomas   4096 May 12 23:17 tools/
-rw-r--r--   1 thomas thomas    467 May 12 23:17 tsconfig.base.json
-rw-r--r--   1 thomas thomas    105 May 12 23:17 workspace.json
thomas@tpusztai-tuw:~/GitRepos/testing/demo$ polaris-cli g slo-mapping-type cost-efficiency

nx generate @polaris-sloc/polaris-nx:slo-mapping-type [name] [options,...]

Options:
  --name                     The name of the SLO Mapping type.
  --project                  The name of the publishable library project, to which the SLO Mapping type should be added.
  --directory                The directory within the project's src folder, where the SLO Mapping type should be created. (default: slo-mappings)
  --createLibProject         Create a publishable library project for the SLO Mapping type.
  --importPath               The import path of the publishable library, e.g., '@my-org/my-lib' (only used and required if --createLibProject is true).
  --dryRun                   Runs through and reports activity without writing to disk.
  --skip-nx-cache            Skip the use of Nx cache.
  --help                     Show available options for project target.

thomas@tpusztai-tuw:~/GitRepos/testing/demo$ polaris-cli g slo-mapping-type cost-efficiency --project=my-slos --createLibProject=true --importPath=@my-org/my-slos
CREATE libs/my-slos/README.md
CREATE libs/my-slos/.babelrc
CREATE libs/my-slos/package.json
CREATE libs/my-slos/src/index.ts
CREATE libs/my-slos/src/lib/my-slos.spec.ts
CREATE libs/my-slos/src/lib/my-slos.ts
CREATE libs/my-slos/tsconfig.json
CREATE libs/my-slos/tsconfig.lib.json
UPDATE tsconfig.base.json
UPDATE workspace.json
UPDATE nx.json
UPDATE package.json
CREATE .eslintrc.json
UPDATE .vscode/extensions.json
CREATE libs/my-slos/.eslintrc.json
CREATE jest.config.js
CREATE jest.preset.js
CREATE libs/my-slos/jest.config.js
CREATE libs/my-slos/tsconfig.spec.json
CREATE libs/my-slos/src/lib/slo-mappings/cost-efficiency.slo-mapping.ts
CREATE libs/my-slos/src/lib/init-polaris-lib.ts
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated request package, see https://github.com/request/request/i
ssues/3142
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
[        .........] / extract:abab: sill extract word-wrap@1.2.3
```

Tooling demo video on YouTube: https://www.youtube.com/watch?v=JVZ4hB2AmGs

Polaris Scheduler for Edge-Cloud Continuum

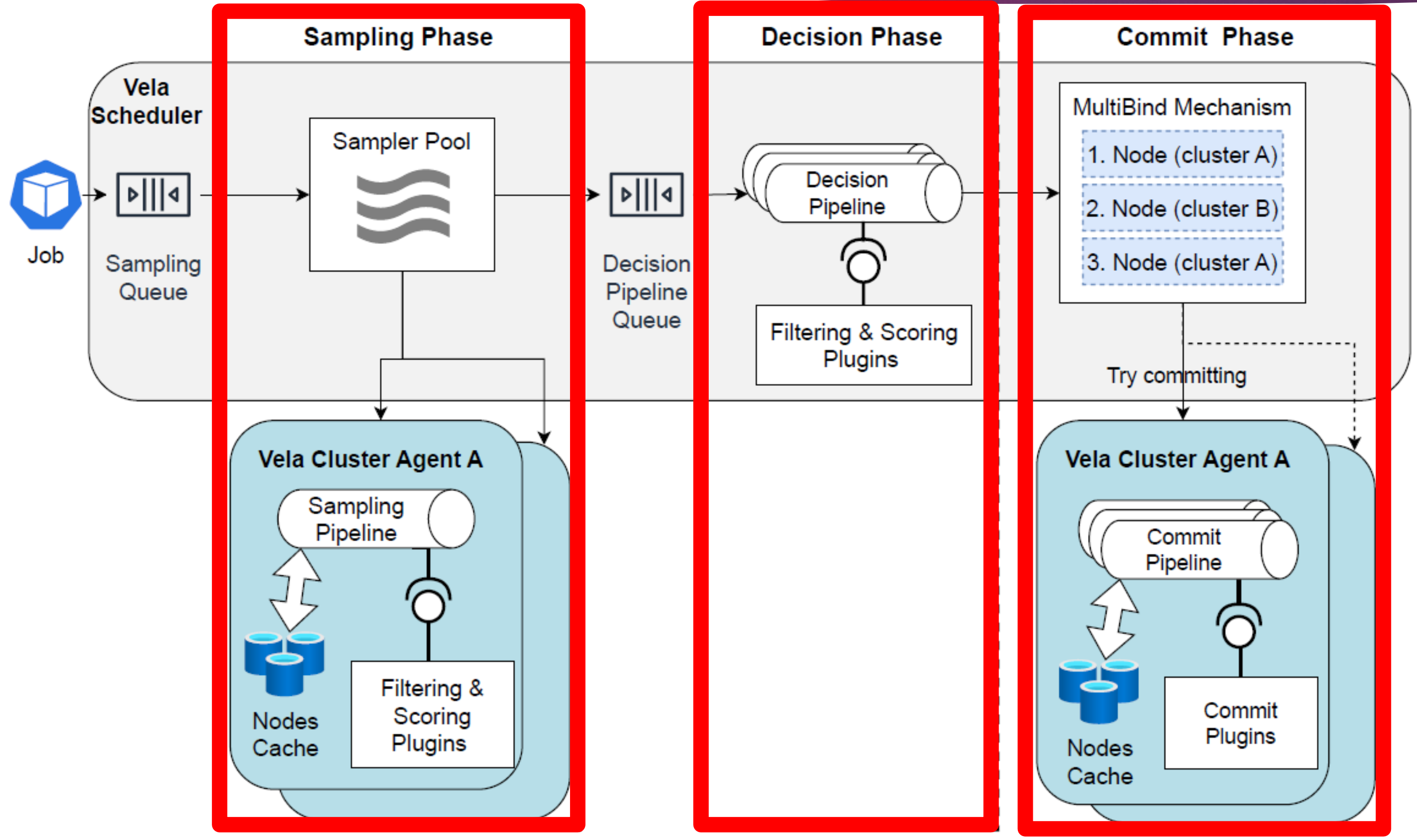
- Plugins-based, Edge-sensitive, and SLO-aware scheduling framework for Edge-Cloud Continuum



Latency and bandwidth variance values are used to assess the QoS requirements (stability of the network connections)

Nastic, S., Pusztai, T., Morichetta, A., Casamayor Pujol, V., Dustdar, S., Vii, D., & Xiong, Y. (2021). Polaris scheduler: Edge sensitive and slo aware workload scheduling in cloud-edge-iot clusters. In IEEE 14th International Conference on Cloud Computing (CLOUD) (pp. 206–216).

Dealing with the Scale of the Edge-Cloud Continuum



- The Edge-Cloud Continuum is very large in scale
- We need to work with a sample of the infrastructure nodes
- We introduced a 3-phase distributed scheduler

Pusztai, T., Nastic, S., Casamayor Pujol, V., Raith, P., Dustdar, S., Vij, D., & Xiong, Y. (2023). Vela: A 3-Phase Distributed Scheduler for the Edge-Cloud Continuum. In The 11th IEEE International Conference on Cloud Engineering (IC2E 2023)

Scheduler Global Scalability Results

Scalability with Respect to Infrastructure

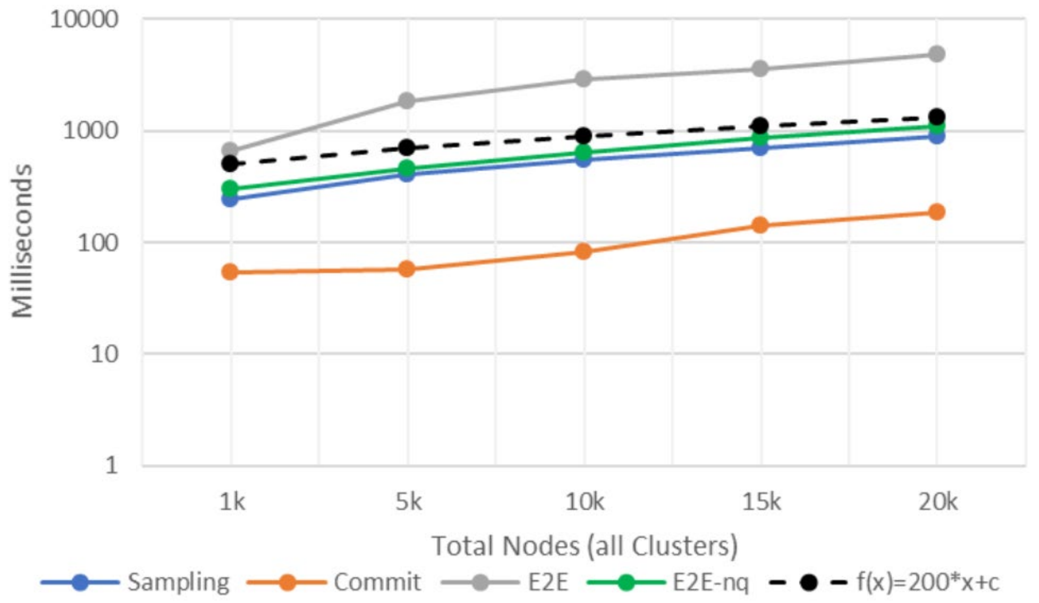


Fig. 2: Mean Scheduling Times (ms) at `percentageOfClustersToSample` = 50% and `nodesToSampleBp` = 4% for Total Nodes.

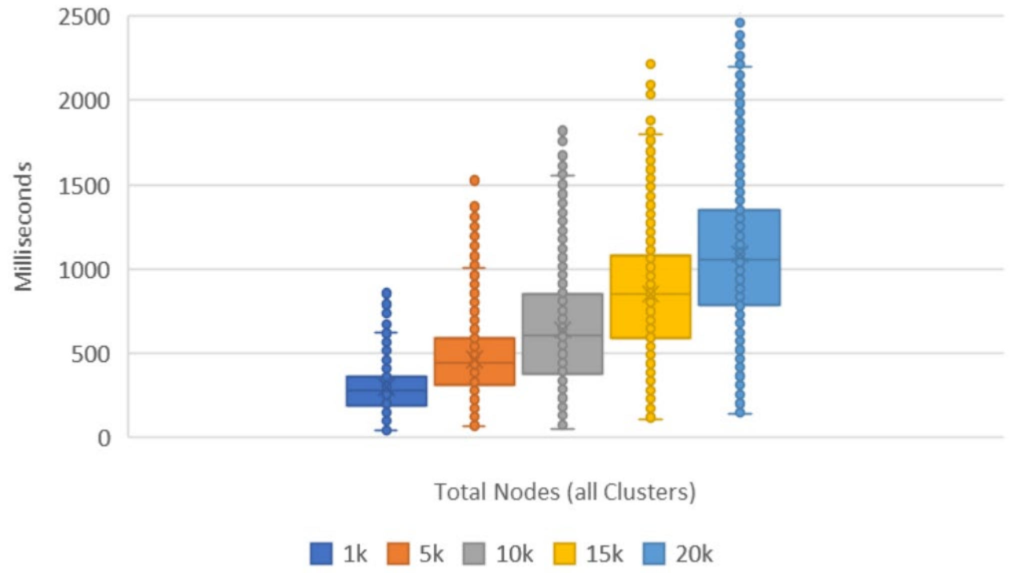


Fig. 5: End-to-End Times (ms), without Sampling Queue at `percentageOfClustersToSample` = 50% and `nodesToSampleBp` = 4% for Total Nodes.

Scheduler Global Scalability Results

Scalability with Respect to Workload Throughput

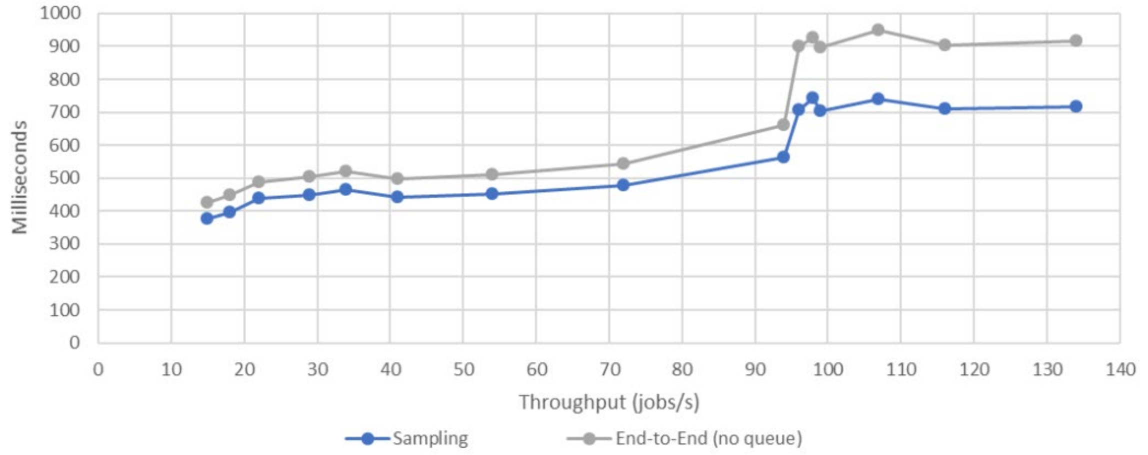


Fig. 8: Mean Sampling and E2E-no-queue Times (ms) with Respect to Throughput.

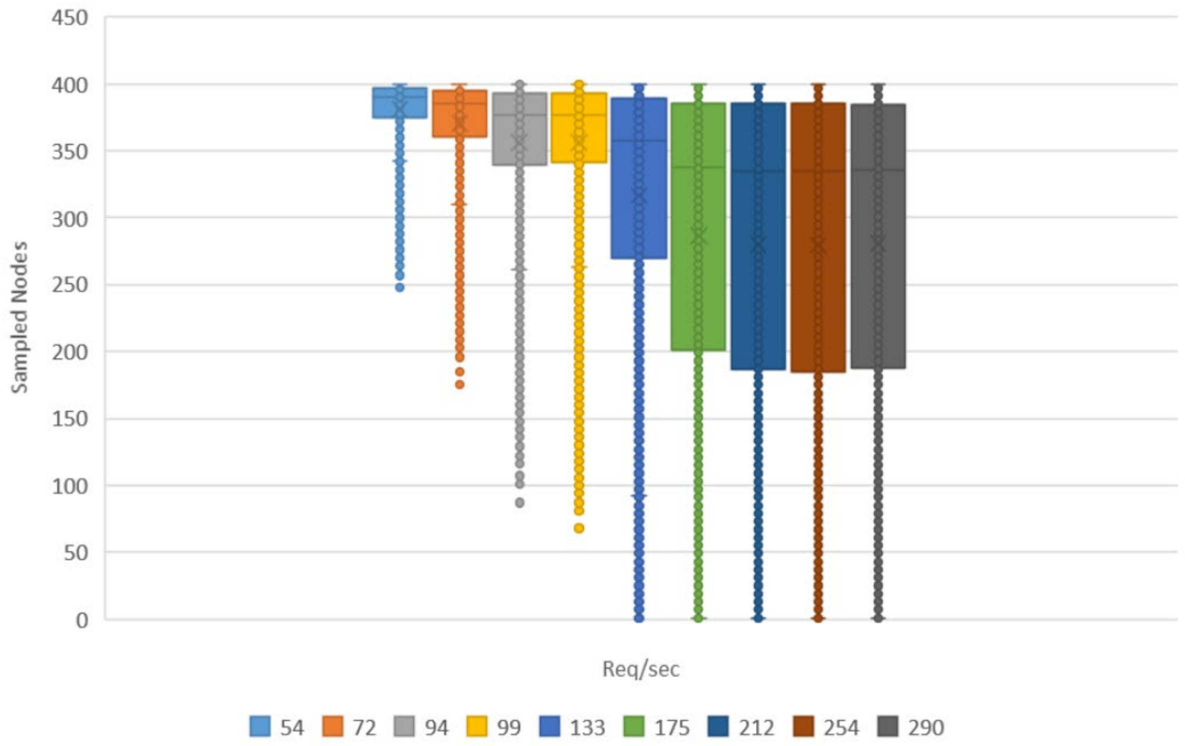
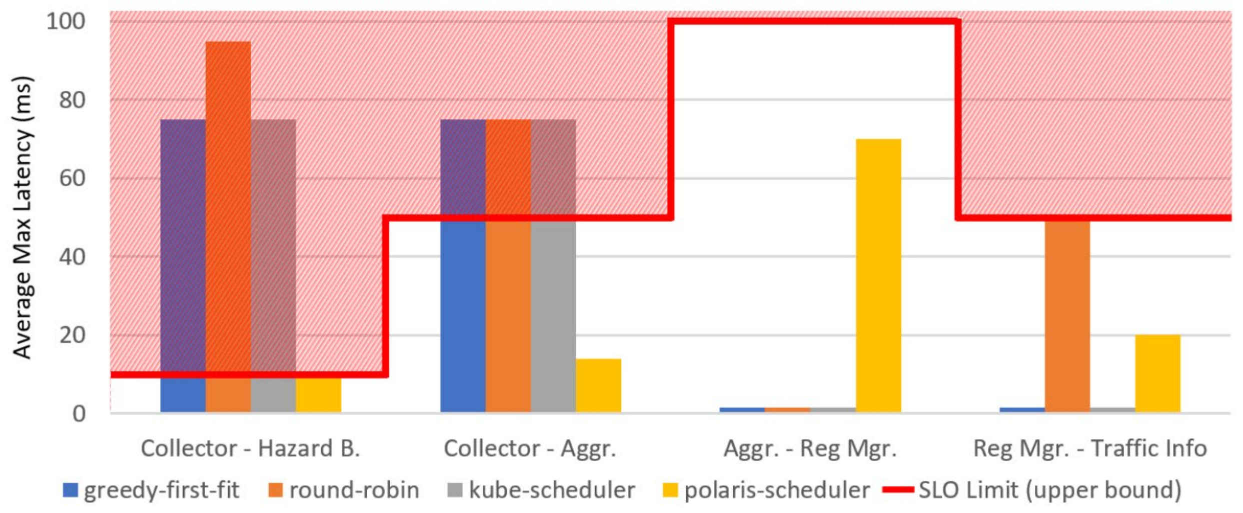
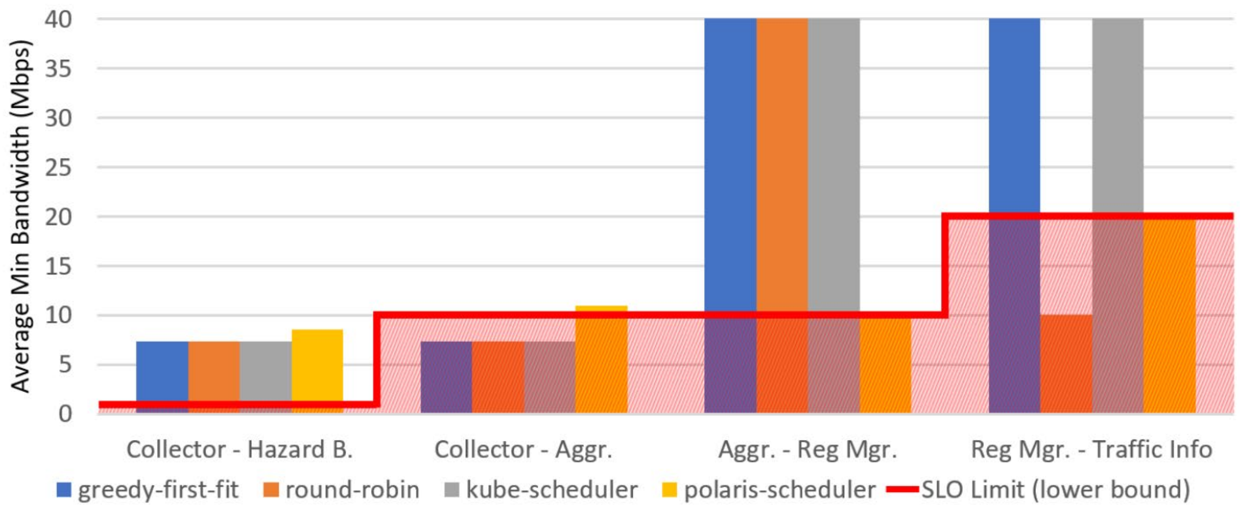


Fig. 7: Sampled Nodes.

Scheduler SLO Compliance Results



Avg. Max Latencies vs SLO Bounds

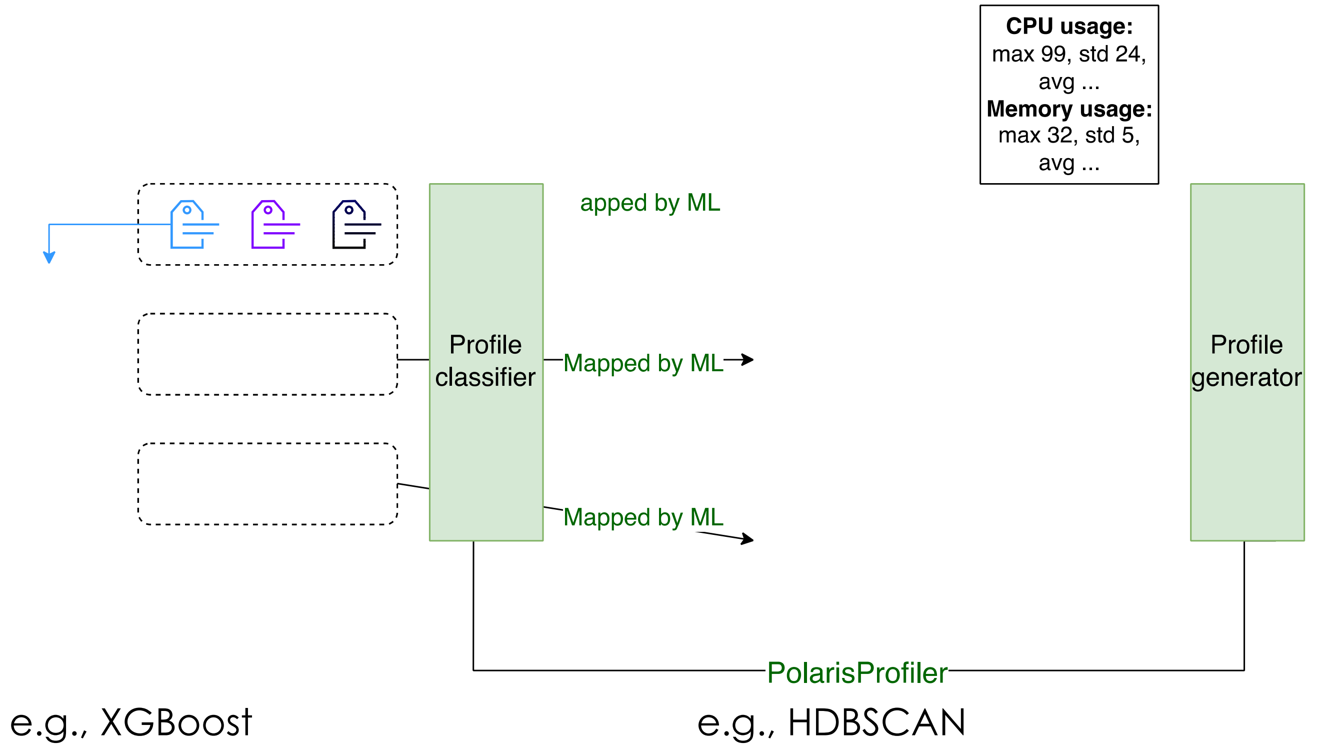


Avg. Min Bandwidth vs SLO Bounds

Pusztai, T., Nastic, S., Morichetta, A., Casamayor Pujol, V., Raith, P., Dustdar, S., … Zhang, Z. (2022). Polaris Scheduler: SLO- and Topology-aware Microservices Scheduling at the Edge.

Metadata-based Workload Profiling

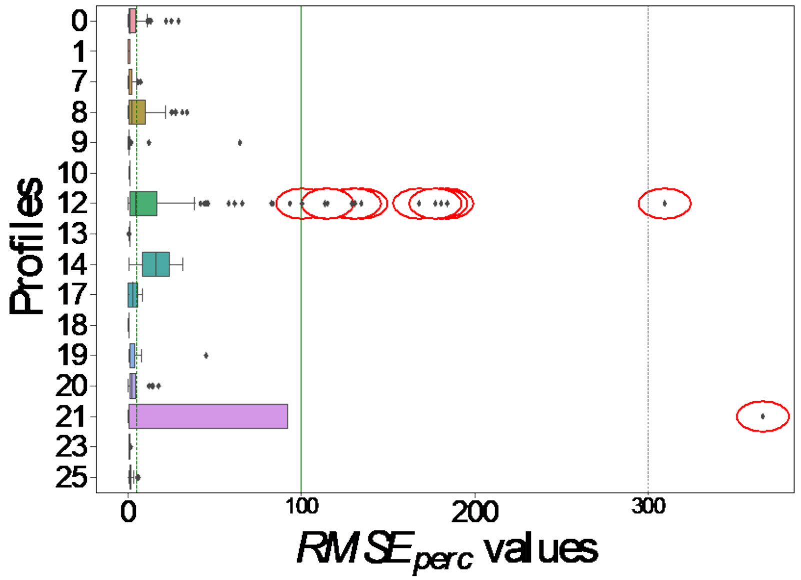
- Address the SLO and scheduling bootstrapping problem
- Main idea: use apriori available meta data to profile workloads
  - Profile generation based on dynamic data
  - Profile classification based on static metadata
- Main insight: We can learn a lot about the future workload by looking at its metadata

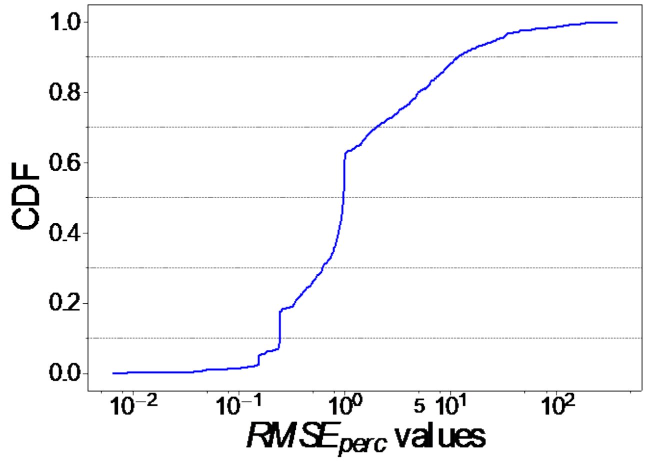


Metadata-based Profiler Results

Predicting job duration

- 1000 proportionally-sampled jobs not used for training
- Use avg duration in each profile as prediction
- RMSE_{perc} below 5% for more than 80% of the profiled jobs

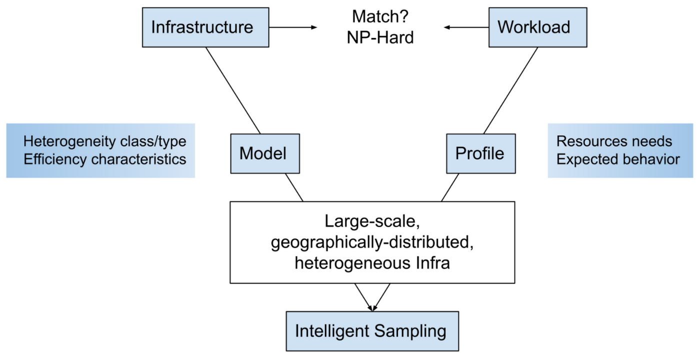




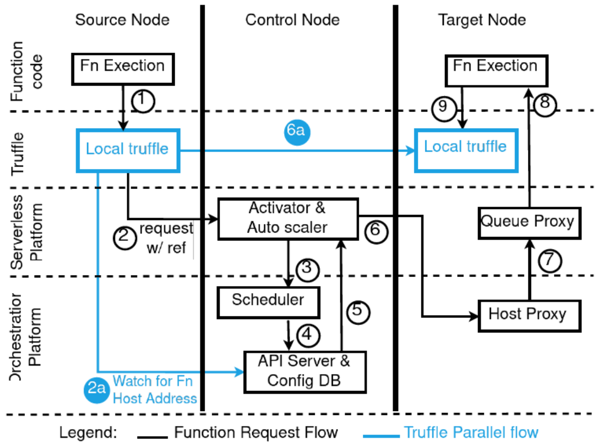
Morichetta, A., Casamayor Pujol, V., Nastic, S., Dustdar, S., Vij, D., Xiong, Y., & Zhang, Z. (2023). PolarisProfiler: A Novel Metadata-Based Profiling Approach for Optimizing Resource Management in the Edge-Cloud Continnum. In The 18th IEEE International Symposium on Service-Oriented System Engineering (SOSE 2023).

Work in progress

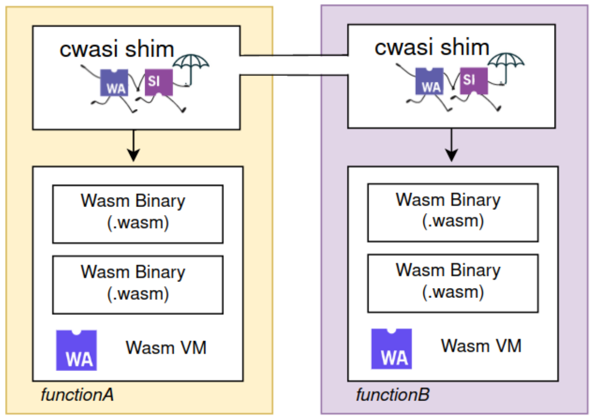
Intelligent sampling in a heterogeneous infrastructure



Using cold starts to improve data transfers between functions



Optimal function invocation for WASM-based FaaS



Conclusion

- Main promises of serverless computing
- Main concepts of serverless computing paradigm
  - FaaS – Function as a Service
  - BaaS – Backend as a Service
  - Function virtualization and isolation approaches
- Serverless = FaaS + BaaS
- Project Polaris – towards serverless computing fabric for the edge-cloud continuum
  - SLO managements
  - Scheduling
  - Profiling

Thank you for your attention!

- Find me at: **www.nastic.at**
- Drop me an email: **snastic@dsg.tuwien.ac.at**