Kubernetes distributions for the edge: serverless performance evaluation [1]

Santo Cariotti

University of Bologna
Jun 17, 2024

Content

Content

Content

Content

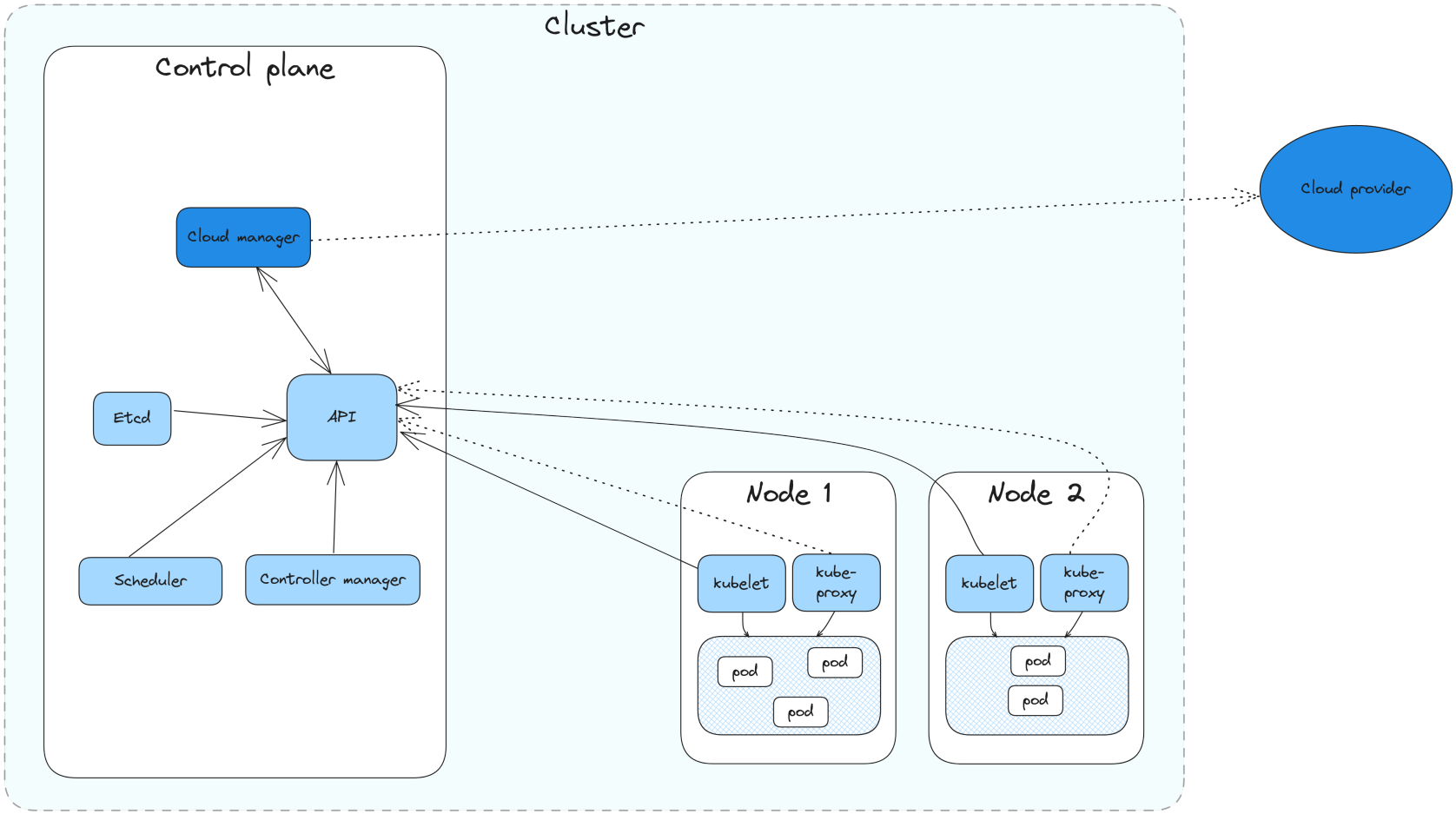
Content

1. What is Kubernetes?
2. What is "the edge"?
3. What is Serverless?
4. How can we combine them?
5. Which K8s distribution is more efficient for serverless development?

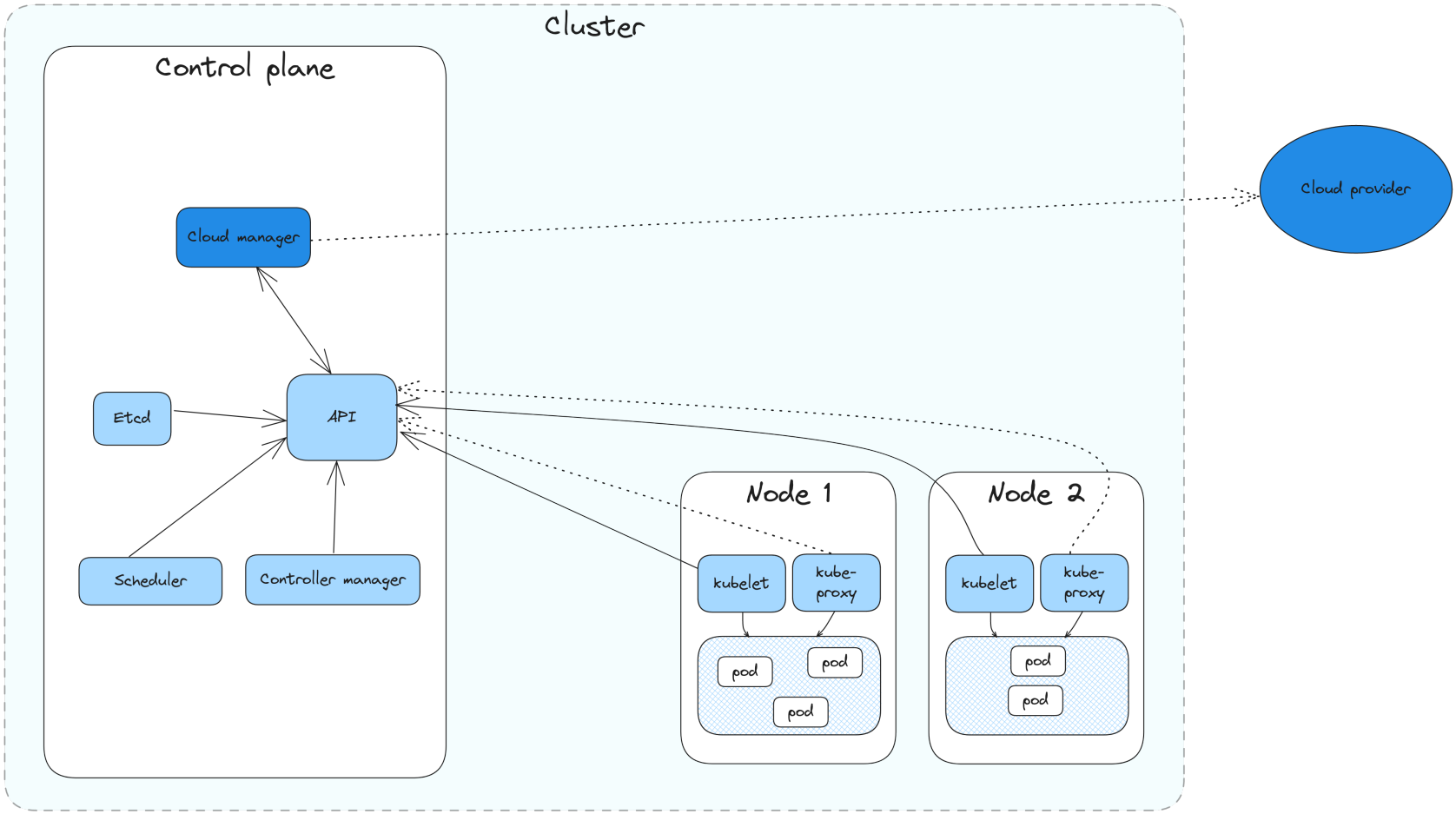
Kubernetes

The development was started by Google in 2014, but is now mantained by Cloud Native Computing Foundation. It is the most widely used container orchestrator.

Kubernetes: Architecture



Kubernetes: Architecture



- Something a bit less complex?

Kubernetes: Distributions

There are a lot of distributions of K8s such as:

- Kubespray
- K3s
- MicroK8s

Kubernetes: Distributions

There are a lot of distributions of K8s such as:

- Kubespray *It uses a set of Ansible playbooks*
- K3s
- MicroK8s

Kubernetes: Distributions

There are a lot of distributions of K8s such as:

- Kubespray *It uses a set of Ansible playbooks*
- K3s *Lightweight packaged as a single binary*
- MicroK8s

Kubernetes: Distributions

There are a lot of distributions of K8s such as:

- Kubespray *It uses a set of Ansible playbooks*
- K3s *Lightweight packaged as a single binary*
- MicroK8s *It works on any GNU/Linux distributions using Snap package manager*

Edge computing

We make deal with low-powered and resource constrained device so to have:

- Faster response times for latency sensitive apps.

Edge computing

We make deal with low-powered and resource constrained device so to have:

- Faster response times for latency sensitive apps.
- Increase user privacy by sending only filtered data upstream to the cloud.

Edge computing

We make deal with low-powered and resource constrained device so to have:

- Faster response times for latency sensitive apps.
- Increase user privacy by sending only filtered data upstream to the cloud.
- Reduced network congestion.

Serverless

- Serverless computing abstracts the underlying infrastructure, focusing solely on the logic that needs to be performed to solve a given task.

Serverless

- Serverless computing abstracts the underlying infrastructure, focusing solely on the logic that needs to be performed to solve a given task.
- A developer just write a function using their favourite programming language and put it online.

Serverless

- Serverless computing abstracts the underlying infrastructure, focusing solely on the logic that needs to be performed to solve a given task.
- A developer just write a function using their favourite programming language and put it online.
- A new concept of Function-as-a-Service.

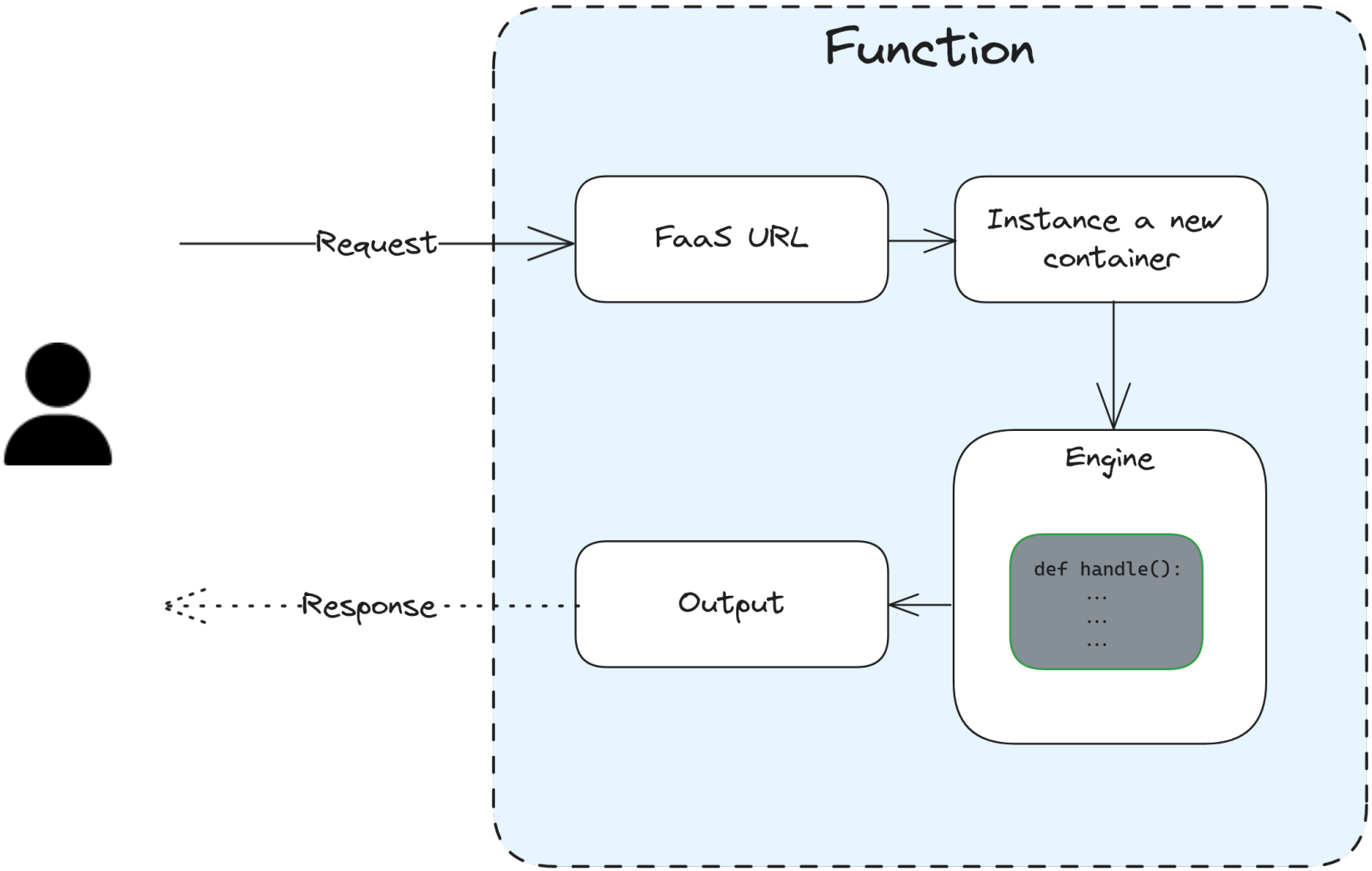
Serverless

- Serverless computing abstracts the underlying infrastructure, focusing solely on the logic that needs to be performed to solve a given task.
- A developer just write a function using their favourite programming language and put it online.
- A new concept of Function-as-a-Service.
- Edge computing is recommended by setting up compute infrastructure closer to the data source.

Serverless

- Serverless computing abstracts the underlying infrastructure, focusing solely on the logic that needs to be performed to solve a given task.
- A developer just write a function using their favourite programming language and put it online.
- A new concept of Function-as-a-Service.
- Edge computing is recommended by setting up compute infrastructure closer to the data source.
- It is a new frontier for IoT computing [2].

Serverless architecture



Serverless platforms

There is a new market for serverless platforms, both open and closed source.

- AWS Lambda
- OpenWhisk
- Kubeless
- Knative
- OpenFaaS

Serverless platforms

There is a new market for serverless platforms, both open and closed source.

- AWS Lambda
- OpenWhisk
- Kubeless
- Knative
- OpenFaaS *we chose this one!*

OpenFaaS

Its architecture is composed by:

- API Gateway
- Prometheus [3]
- Watchdog
- Docker Swarm or Kubernetes
- Docker

It supports two different function scaling modes:

- Native scaling based on internal customized metrics
- Kubernetes Horizontal Pod Autoscaler (HPA)

Set up the testing machine

- Ubuntu 20.04
- CPU Intel Xeon X5647
- 8 GB RAM
- 1 Gbps
- Kubernetes 1.20.7 and 1.23.0
- OpenFaaS 0.21.1
- 1 master node
- Calico as CNI
- Longhorn for persistent volumes
- Docker Engine as CRI

Set up the testing machine

- Ubuntu 20.04
- CPU Intel Xeon X5647
- 8 GB RAM
- 1 Gbps
- Kubernetes 1.20.7 and 1.23.0
- OpenFaaS 0.21.1
- 1 master node
- Calico as CNI
- Longhorn for persistent volumes
- Docker Engine as CRI *dockershim is no longer supported after 1.24*

Choose the testing functions

Let's choose 14 tests from the FunctionBench serverless benchmarking suite [4].

- 8 from CPU & Memory category.
- 4 from Disk I/O category.
- 2 from Network performance category.

How do we run?

5 tests for each Kubernetes distribution. Executed using the recommended OpenFaaS of-watchdog template for Python 3.7.

How do we run?

5 tests for each Kubernetes distribution. Executed using the recommended OpenFaaS of-watchdog template for Python 3.7. We'll see performances for:

- Cold start
- Serial execution
- Parallel execution using a single replica
- Parallel execution using native OpenFaaS auto scaling
- Parallel execution using Kubernetes HPA

Cold start performance

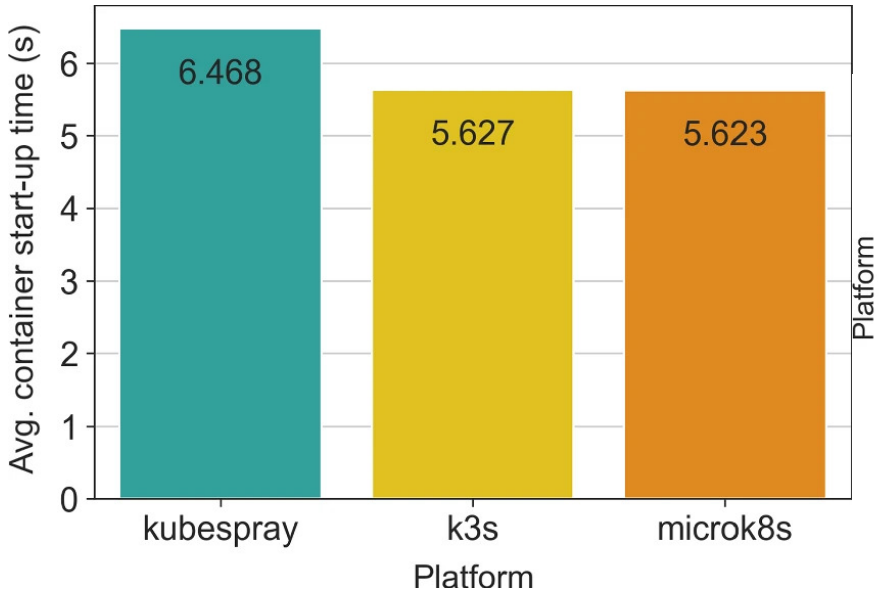
Each function is executed 100 times to test the cold start delay.

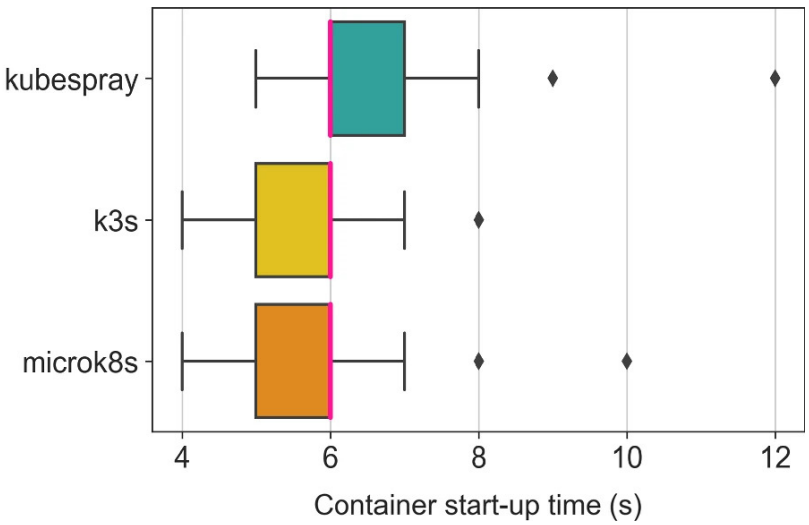
Cold start performance

Each function is executed 100 times to test the cold start delay. After every execution, the number of instances for the function is scaled to 0. A new container instance needs to be created before a response is returned.

Cold start performance — Results

Kubespray exhibits a 15% increase compared to both K3s and MicroK8s.





Cold start performance — Results

Is the performance difference between K3s and MicroK8s statistically significant?

Cold start performance — Results

Is the performance difference between K3s and MicroK8s statistically significant?
Using a Mann-Whitney U test with $\alpha = 0.05$ and

- H0: the two populations are equal
- H1: the two populations are not equal

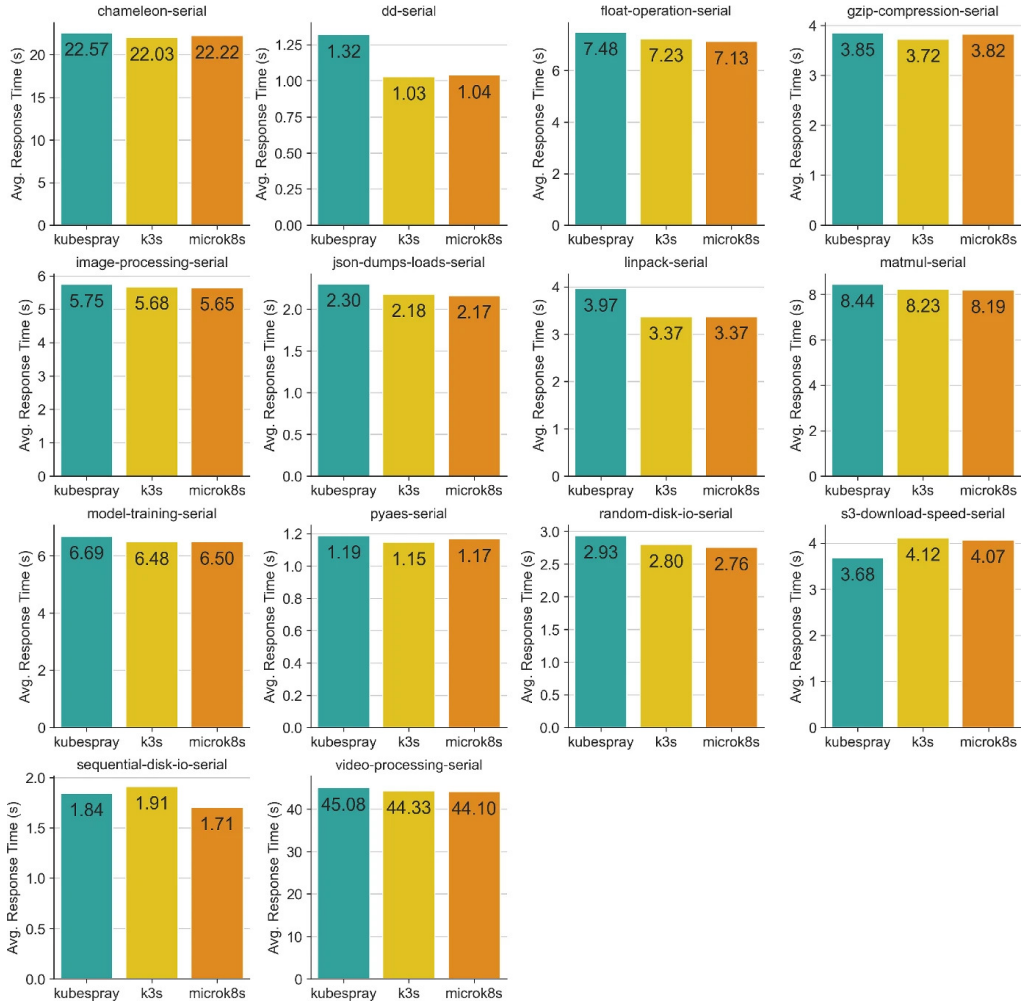
we have a p-value $= 0.202 > 0.05$, so we can't reject the null hypothesis.

Serial execution performance

Each function is continuously invoked for a period of 5 min using a single thread. Once a response is received, a new request is immediately sent. Auto-scaling is manually disabled.

Serial execution performance — Results

Once again, Kubespray results are slower than both K3s and MicroK8s.



Serial execution performance — Results

How can I compare these three results?

Serial execution performance — Results

How can I compare these three results?
Using a Kruskal-Wallis test with $\alpha = 0.05$ and

- H0: the population medians are equal
- H1: the population medians are not equal

where the null hypothesis failed to be rejected for the video-processing test.

Serial execution performance — Results

How can I compare these three results?
Using a Kruskal-Wallis test with $\alpha = 0.05$ and

- H0: the population medians are equal
- H1: the population medians are not equal

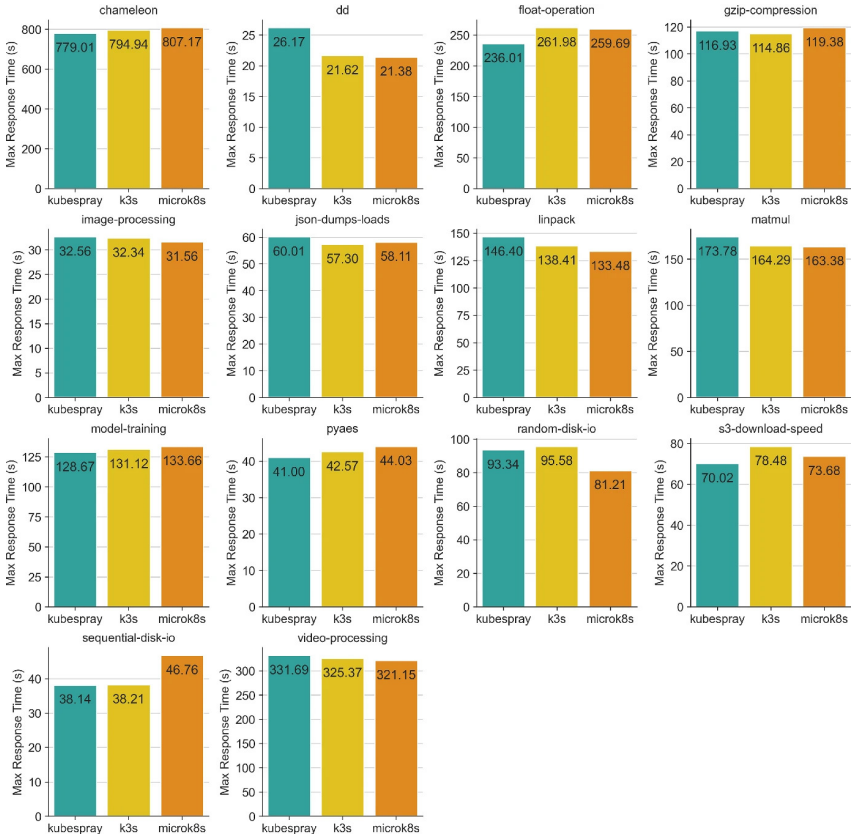
where the null hypothesis failed to be rejected for the video-processing test.
Keeping the same hypothesis we can perform the Mann-Whitney U test for both K3s and MicroK8s and, this time, the null hypothesis is rejected in 10 of the 14 tests. The null hypothesis can't be rejected for 3 CPU and 1 network benchmarks.

Parallel execution performance using a single replica

Each function is invoked for a fixed amount of time using varying concurrency to determine the performance of the auto-scaling behavior. *Reduced isolation.*

Parallel execution performance using a single replica — Results

This time Kubespray has better performance than both K3s and MicroK8s for 6 of the 14 tests.



Parallel execution using native OpenFaaS auto scaling

Each function is invoked for a fixed amount of time using varying concurrency to determine the performance of the auto-scaling behavior.

Parallel execution using native OpenFaaS auto scaling

Each function is invoked for a fixed amount of time using varying concurrency to determine the performance of the auto-scaling behavior.
It tests the number of successful invocations per second for the last 10 seconds: if the number is larger than 5, it scales up the number of function instances up to a preconfigured maximum.

Parallel execution using native OpenFaaS auto scaling — Results

Performances are tests by 1 req/s from 6 concurrent workers for more than 200s, successfully reaching the defined threshold for the maximum number of replicas.
The current number of deployed replicas are not taken but this leads to suboptimal scaling decision scaling to maximum number of configured replicas or not scaling at all under a consistent load.

Parallel execution using Kubernetes Horizonal Pod Autoscaler

Each function is invoked as the same way as the previous test using the Kubernetes native mechanism. For this test, HPA is configured with a profile which is fired whenever the float-operation function has used more than 350 CPU shares (0.35 of a core).

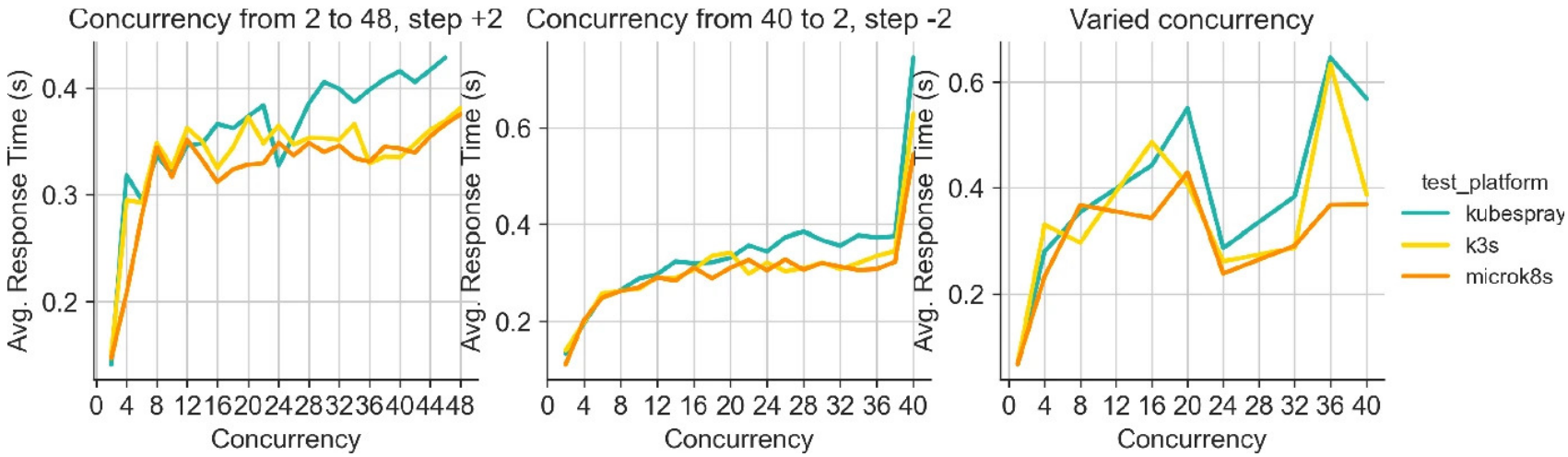
Parallel execution using Kubernetes Horizonal Pod Autoscaler

Each function is invoked as the same way as the previous test using the Kubernetes native mechanism. For this test, HPA is configured with a profile which is fired whenever the float-operation function has used more than 350 CPU shares (0.35 of a core). We find out results for three different execution strategy:

- Start from 1 replica, execute 2 concurrent req/s, increasing the concurrency rate by 2 every 5 min, until 48 req/s are achieved.
- Start from 1 replica, execute 40 concurrent req/s, decreasing the concurrency rate by 2 every 5 min, until 2 req/s are achieved.
- Start from 1 replica and vary the number of concurrent requests every 5 min using the strategy 8, 1, 20, 4, 40, 24, 1, 4, 16, 1, 36, 32.

Parallel execution using Kubernetes Horizonal Pod Autoscaler — Results

Kubespray exhibits higher response times across the three tests, while the results obtained from K3s and MicroK8s are similar.



Conclusion

- Kubespray takes longer to instantiate new function instances.
- K3s and MicroK8s, removing unnecessary components, have reduced the deployment time and complexity, improving performance for the majority of serverless edge workloads.

Conclusion

- Kubespray takes longer to instantiate new function instances.
- K3s and MicroK8s, removing unnecessary components, have reduced the deployment time and complexity, improving performance for the majority of serverless edge workloads. **We can say that they are comparable for out serverless performance evaluation.**

References

- [1] Kjorveziroski, V. and Filiposka, S., 2022. Kubernetes distributions for the edge: serverless performance evaluation. The Journal of Supercomputing, 78(11), pp.13728-13755.
- [2] Kjorveziroski, V., Bernad Canto, C., Juan Roig, P., Gilly, K., Mishev, A., Trajkovikj, V. and Filiposka, S., 2021. IoT serverless computing at the edge: Open issues and research direction. Transactions on Networks and Communications.
- [3] https://prometheus.io
- [4] Kim, J. and Lee, K., 2019, July. Functionbench: A suite of workloads for serverless cloud function service. In 2019 IEEE 12th International Conference on Cloud Computing (CLOUD) (pp. 502-504). IEEE.