Owl: Performance-Aware Scheduling for Resource-Efficient Function-as-a-Service Cloud

Huangshi Tian[1], Suyi Li[1], Ao Wang[2,3], Wei Wang[1], Tianlong Wu[3], Haoran Yang[3]

[1]*HKUST,* [2]*George Mason University,* [3]*Alibaba Cloud*

FaaS Gaining Popularity

> A new report from Datadog has found that serverless computing could be entering the **mainstream** with **over half** of all organizations using serverless ...
>
> —— TechCrunch[1], Jun. 2022

> AWS Lambda ..., and more than **a million** customers are using it today, according to AWS.
>
> —— Protocol[2], Aug. 2022

1. Datadog finds serverless computing is going mainstream, https://tcrn.ch/3D5GhHB
2. Amazon's Werner Vogels: Enterprises are more daring than you might think, https://bit.ly/3F8Xtij

FaaS Gaining Popularity

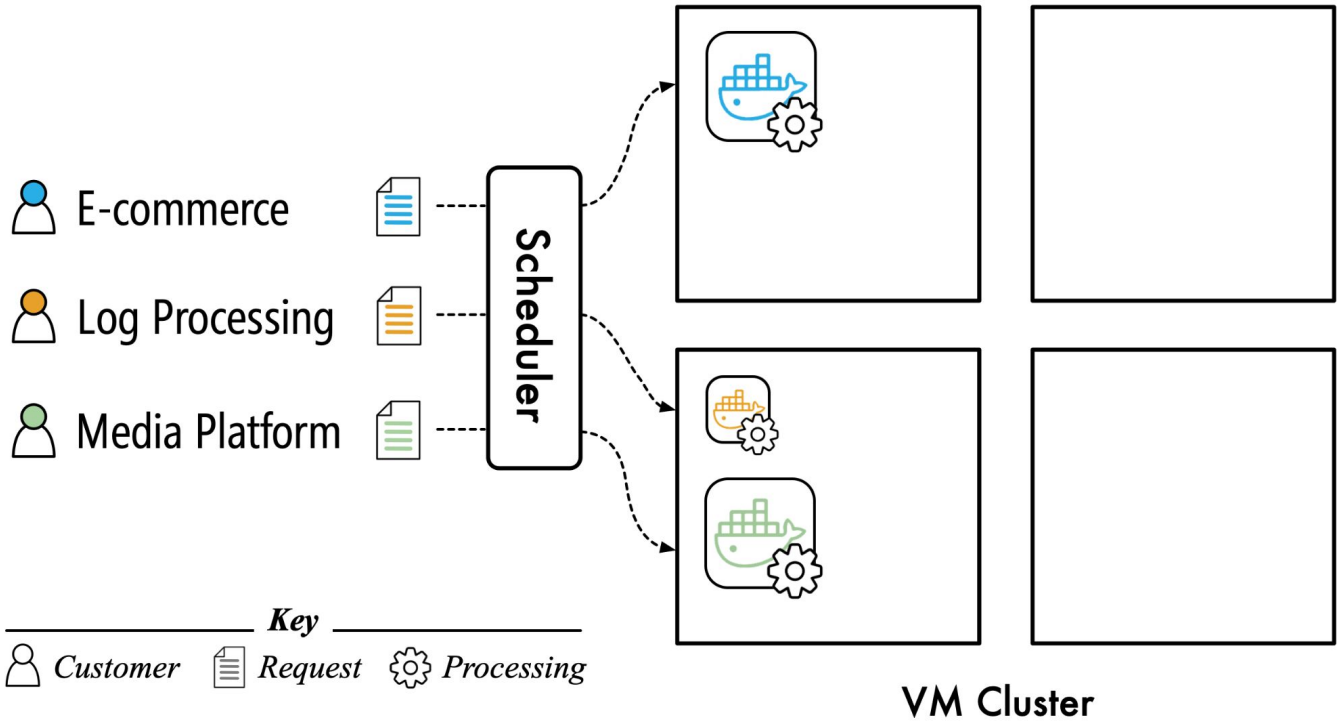
> A new report from Datadog has found that serverless computing could be entering the **mainstream** with **over half** of all organizations using

Problem: How to serve functions efficiently?

> AWS Lambda ..., and more than **a million** customers are using it today, according to AWS.
>
> —— Protocol[2], Aug. 2022

1. Datadog finds serverless computing is going mainstream, https://tcrn.ch/3D5GhHB
2. Amazon's Werner Vogels: Enterprises are more daring than you might think, https://bit.ly/3F8Xtij

Scheduling in Public FaaS



Scheduling in Public FaaS

FaaS Scheduling

=

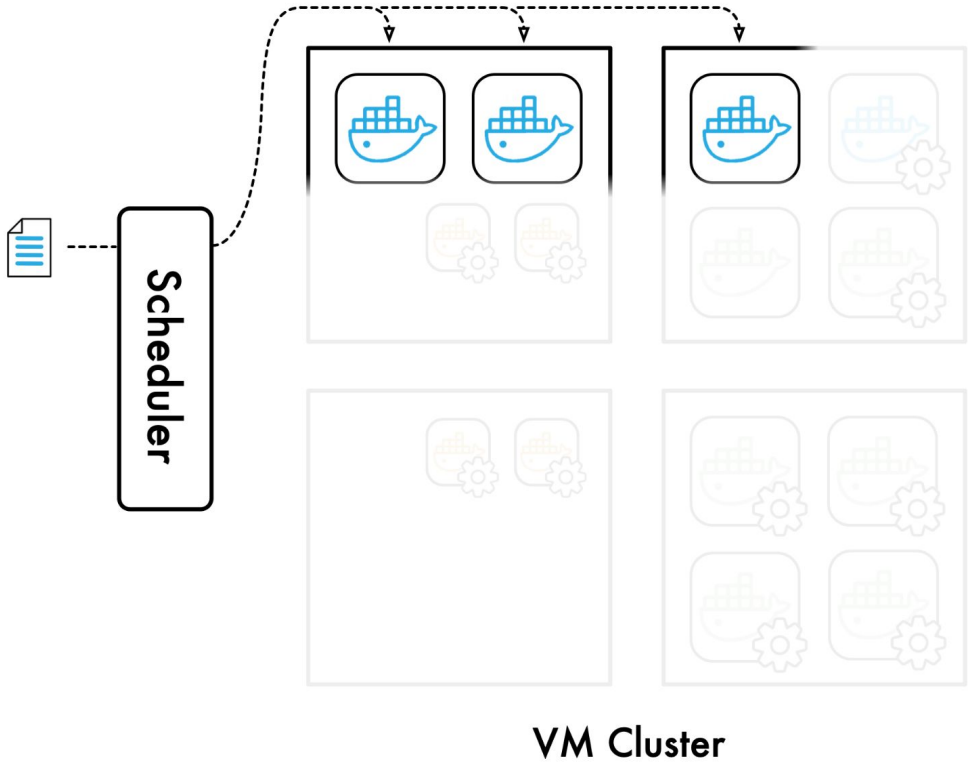
Request Routing

which sandbox to serve request?

+

Sandbox Placement

which VM to place sandbox?



Scheduling in Public FaaS

FaaS Scheduling

=

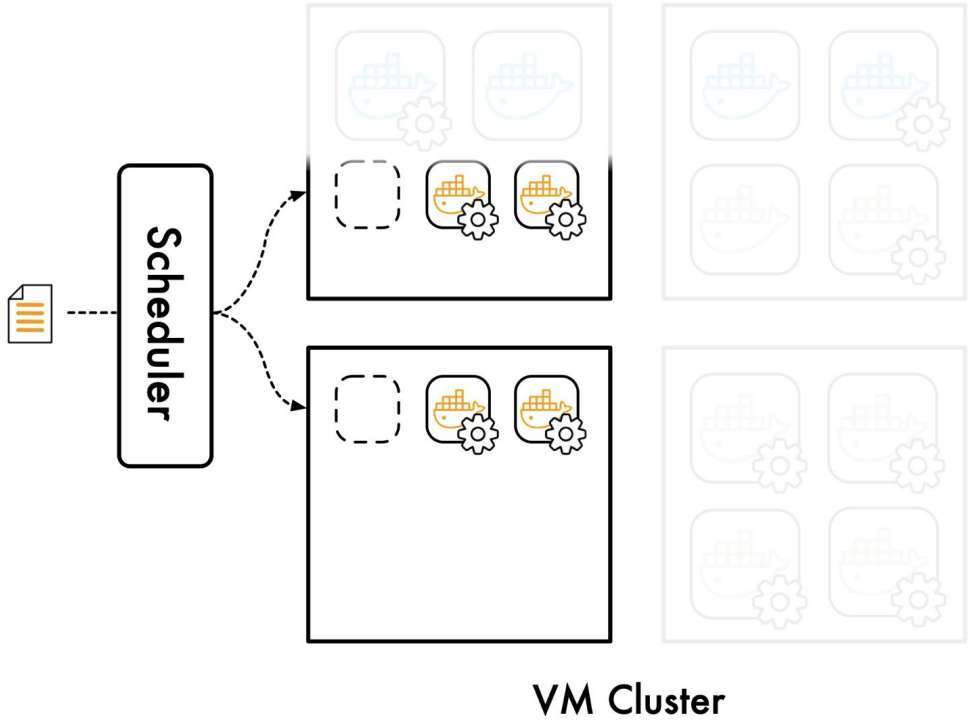
Request Routing

which sandbox to serve request?

+

Sandbox Placement

which VM to place sandbox?



Scheduling in Public FaaS

FaaS Scheduling

=

Request Routing

which sandbox to serve request?

+

Sandbox Placement

which VM to place sandbox?

Model*

1. Each **VM** has a memory capacity.
2. Each **sandbox** has a memory size.

Goal

Pack sandboxes onto VMs.

*** similar as bin-packing**

Status Quo

FaaS Scheduling

=

Request Routing
which sandbox to serve request?

+

Sandbox Placement
which VM to place sandbox?

Setting*

1. Each **VM** has a memory capacity.
2. Each **sandbox** has a memory size.

Goal

Pack sandboxes onto VMs.

* **similar as bin-packing**

State of the Practice

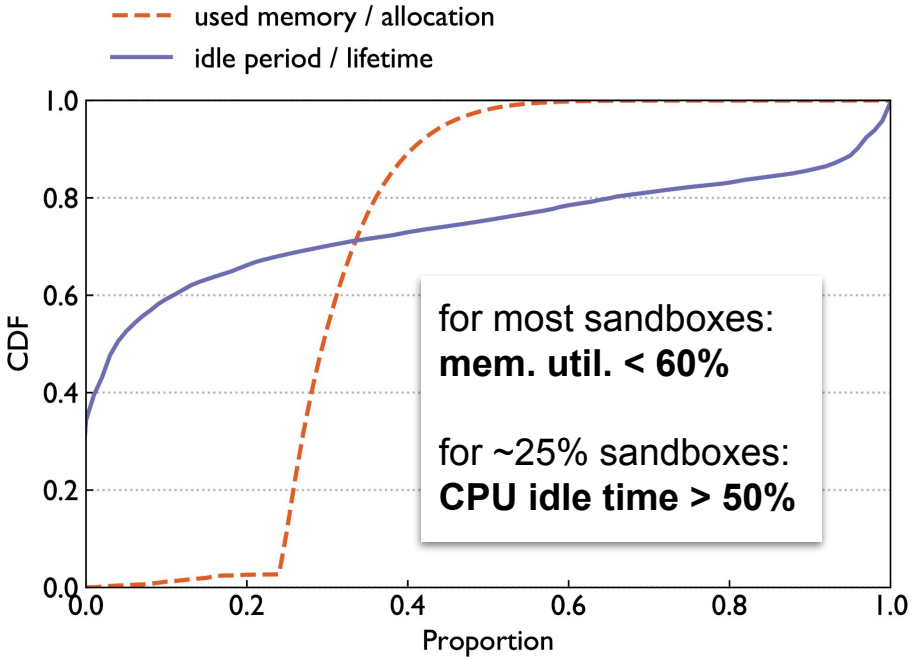
=

Most-Recently Used
for keeping more sandboxes idle

+

First Fit
for quick decision

Resource Inefficiency



State of the Practice

=

Most-Recently Used

for keeping more sandboxes idle

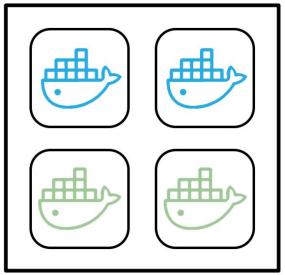
+

First Fit

for quick decision

* data collected from an one-day production trace

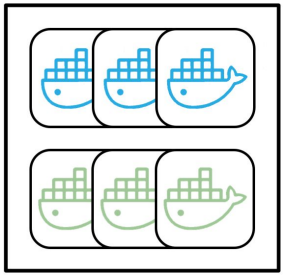
Naive Overcommitment



Normal VM

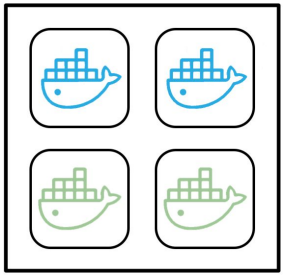
—Sandbox Overcommitment→

usage-based heuristic:
allocate what sandbox utilizes



Ovct.'ed VM

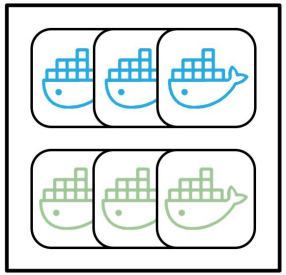
Naive Overcommitment Falls Short



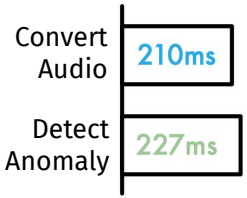
Normal VM

—Sandbox Overcommitment→

usage-based heuristic:
allocate what sandbox utilizes



Ovct.'ed VM



P95 Lat.[1]

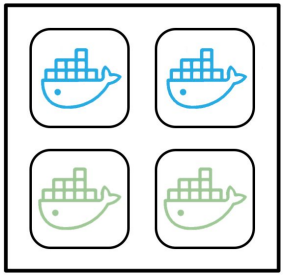
Performance Degrades

unacceptable in production

Ovct.'ed P95 Lat.[1]

1. The latency is measured from a benchmark workload. (See §7.1 in the paper).

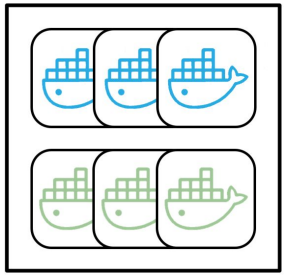
Naive Overcommitment Falls Short



Normal VM

—Sandbox Overcommitment→

usage-based heuristic:
allocate what sandbox utilizes



Ovct.'ed VM

Convert Audio 210ms

435ms (2.07x)

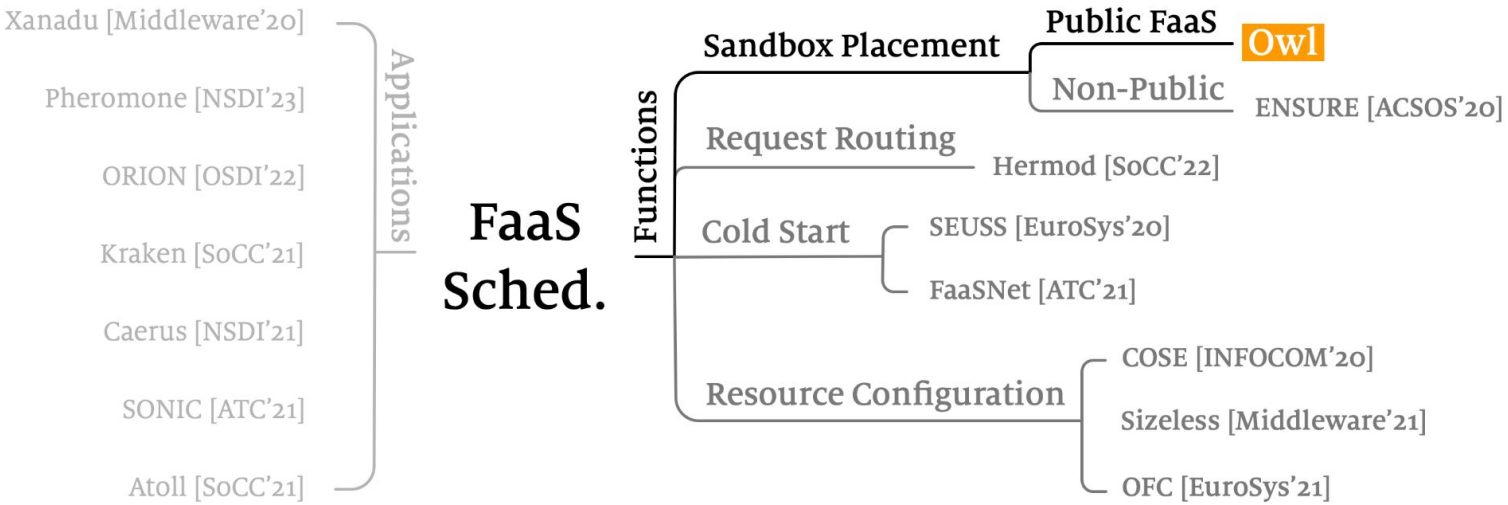
Objective:
Resource-Efficient and Performance-Aware FaaS Scheduling

P95 Lat.[1]

Ovct.'ed P95 Lat.[1]

1. The latency is measured from a benchmark workload. (See §7.1 in the paper).

Related Works



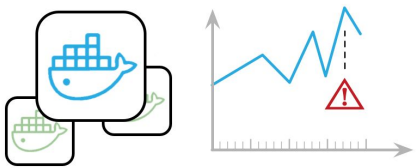
Objective:
Resource-Efficient and Performance-Aware FaaS Scheduling

Outline

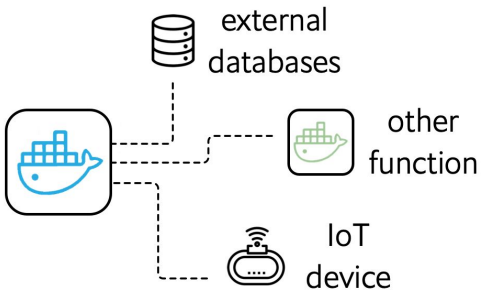
1. Background and Motivation
2. Profile-Guided Overcommitment
   - Collocation Profiling
   - Profile-Guided Placement
3. Performance-Monitored Overcommitment
   - Comparative Validation
4. Implementation and Evaluation

Improve Overcommitment via Profiling

No latency **degradation**.



No **offline** invocation (b/c of side-effect).



Restrictions for Profiling:

☐ **in-production** ☐ **performance-protected**

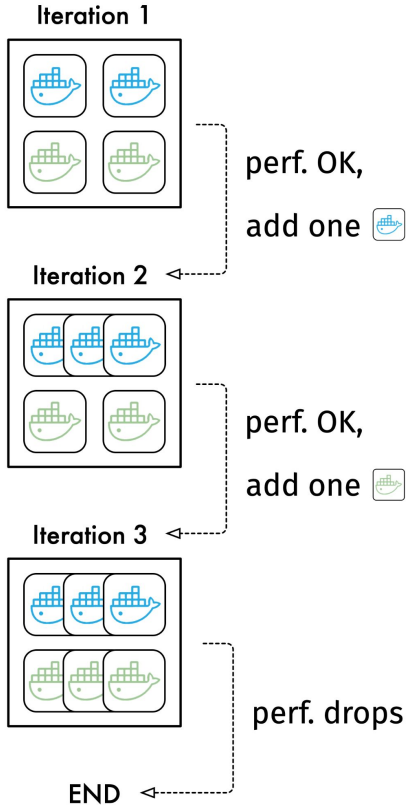
Collocation Profiling

Key Question

How many sandboxes can a VM host?

Procedure

1. **Saturate** sandboxes with requests.
2. Iteratively add more sandboxes …
3. … until perf. starts dropping.



☐ **in-production** ☐ **performance-protected**

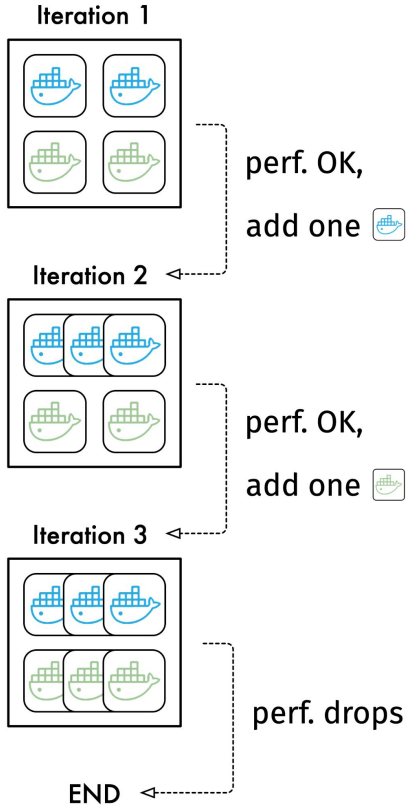
Collocation Profiling

Key Question

How many sandboxes can a VM host?

Procedure

1. **Saturate** sandboxes with requests.
2. Iteratively add more sandboxes ...
3. ... until perf. starts dropping.



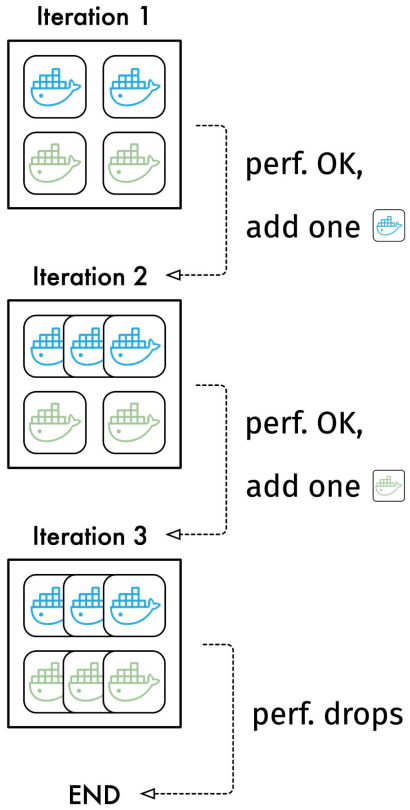
☑ **in-production**
cf. Step 1, 2

☑ **performance-protected**
cf. Step 3

☑ **resource-overcommitted**
cf. Step 2

Collocation Profiling

Collocation[2]	Util.[1]
×3 ×3	1.48
×3 ×5	1.42
×3 ×3	1.26
×5	1.22
×5	1.24
×7	1.20



1. Util. = Allocated Memory / Total VM Memory
   (> 1 means *overcommitment*)

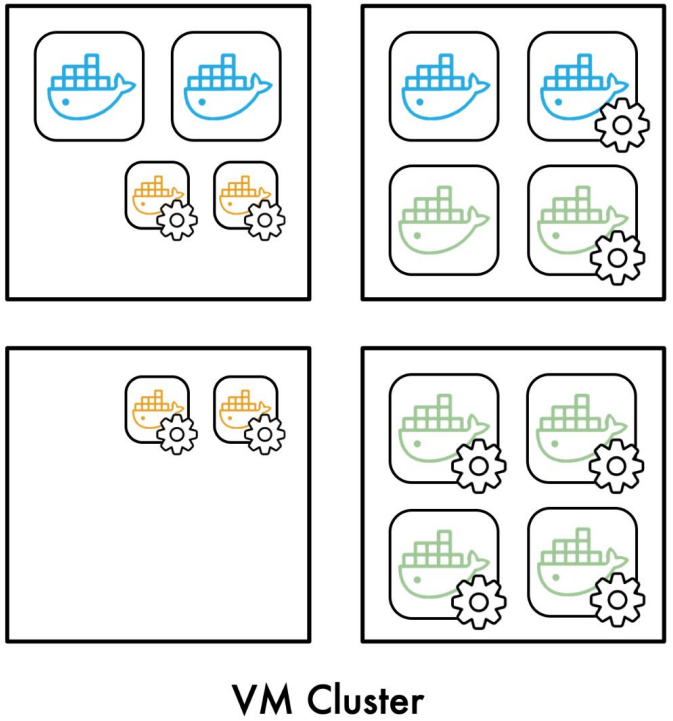
2. Limited to two functions b/c of complexity.

Profile-Guided Overcommitment

Collocation	Util.
×3 ×3	1.48
×3 ×5	1.42
×3 ×3	1.26
×5	1.22
×5	1.24
×7	1.20

Key Question

How to place sandboxes using profiles?



VM Cluster

Profile-Guided Overcommitment

Collocation	Util.
×3 ×3	1.48
×3 ×5	1.42
×3 ×3	1.26
×5	1.22
×5	1.24
×7	1.20

Greedy Algorithm

Collocating sandboxes with highest util.

VM Cluster

Profile-Guided Overcommitment

Collocation	Util.
×3 ×3	1.48
×3 ×5	1.42
×3 ×3	1.26
×5	1.22
×5	1.24
×7	1.20

Greedy Algorithm

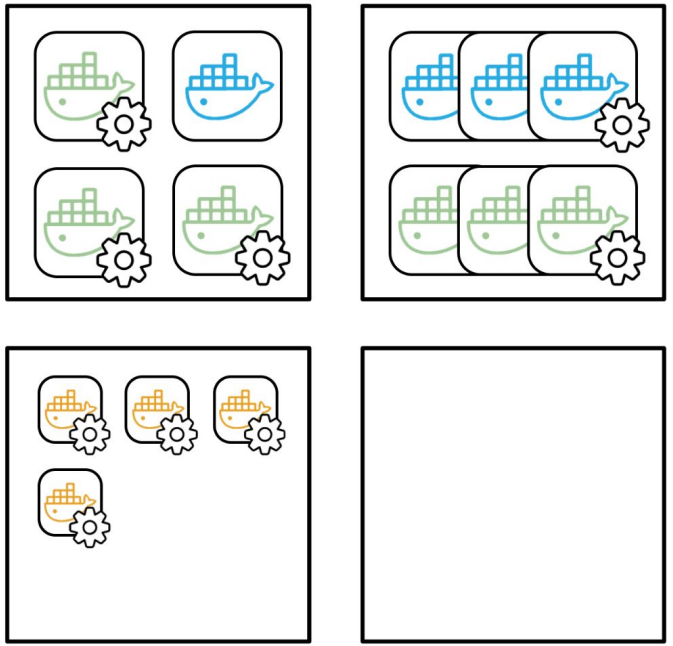
Collocating sandboxes with highest util.

VM Cluster

Profile-Guided Overcommitment

Offline

Collocating sandboxes with highest util.



VM Cluster

Profile-Guided Overcommitment

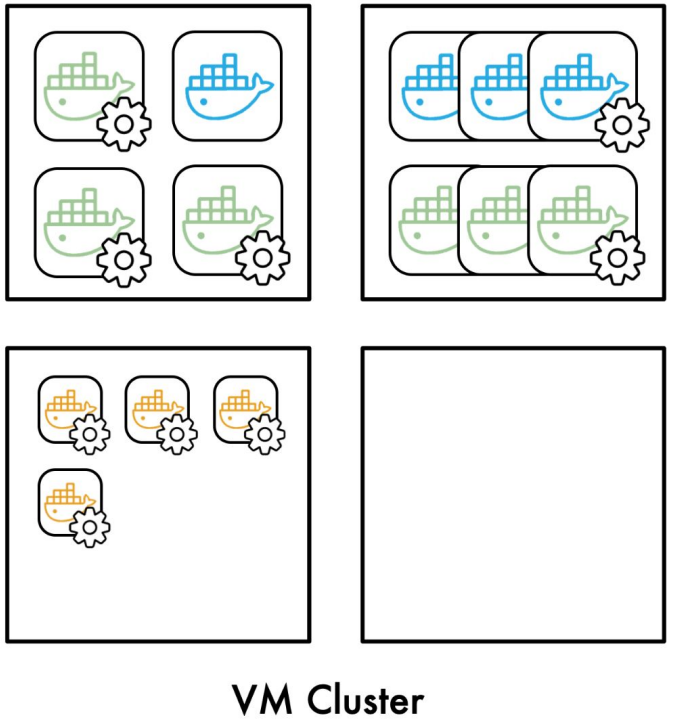
Offline

Collocating sandboxes with highest util.

Online

- **Periodically** update placement …
- … among VMs with **sandbox change**.

[more details and optimizations in paper]



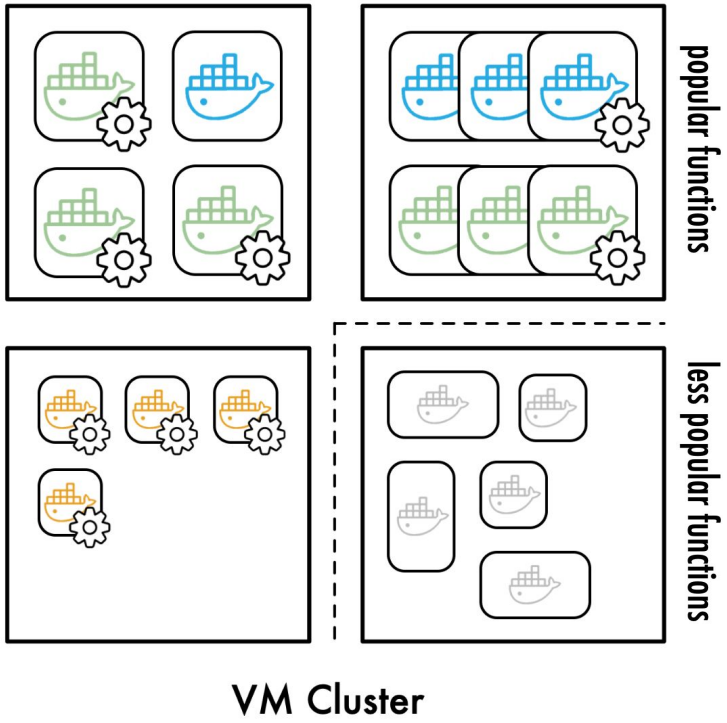
Profile-Guided Overcommitment

Problem

profiling requires **continuous** requests

⇩

only applies to **popular** functions



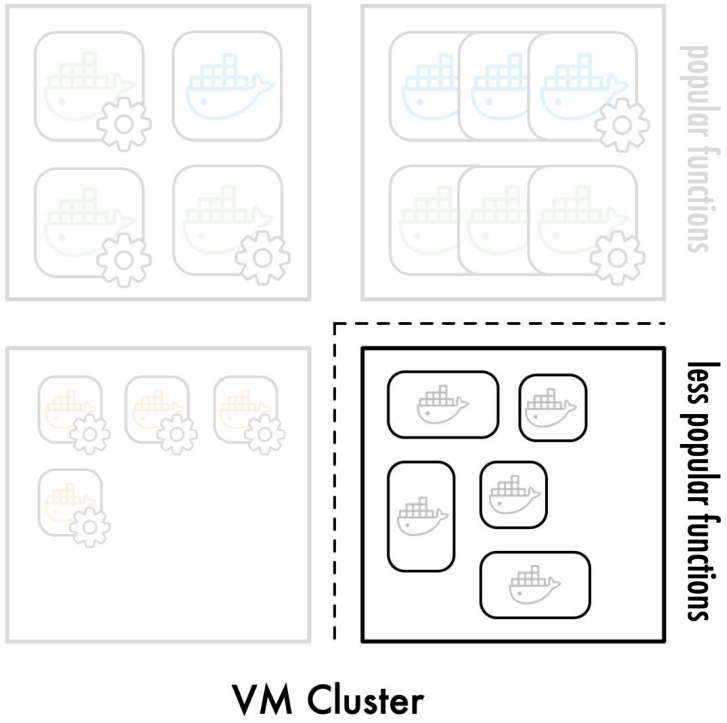
Performance-Monitored Overcommitment

Solution

1. Usage-based overcommitment.

2. Keep **monitoring** performance.

3. **Remedy** degradation (e.g., sandbox migration).



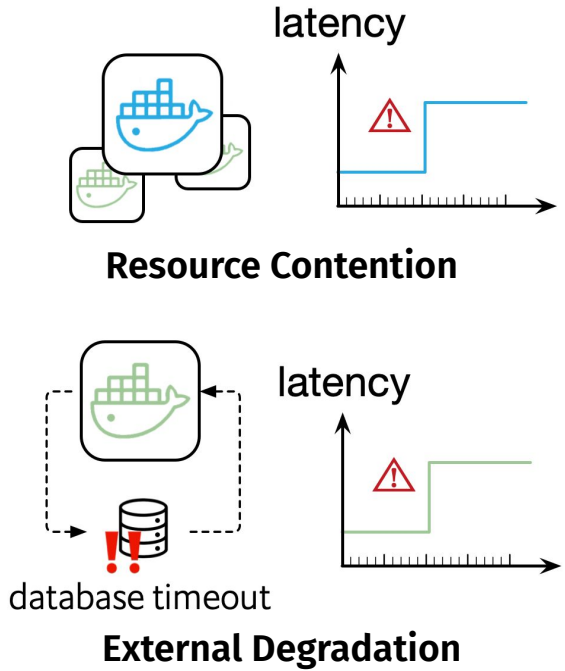
Problem: External Degradation

Solution

1. Usage-based overcommitment.

2. Keep **monitoring** performance.

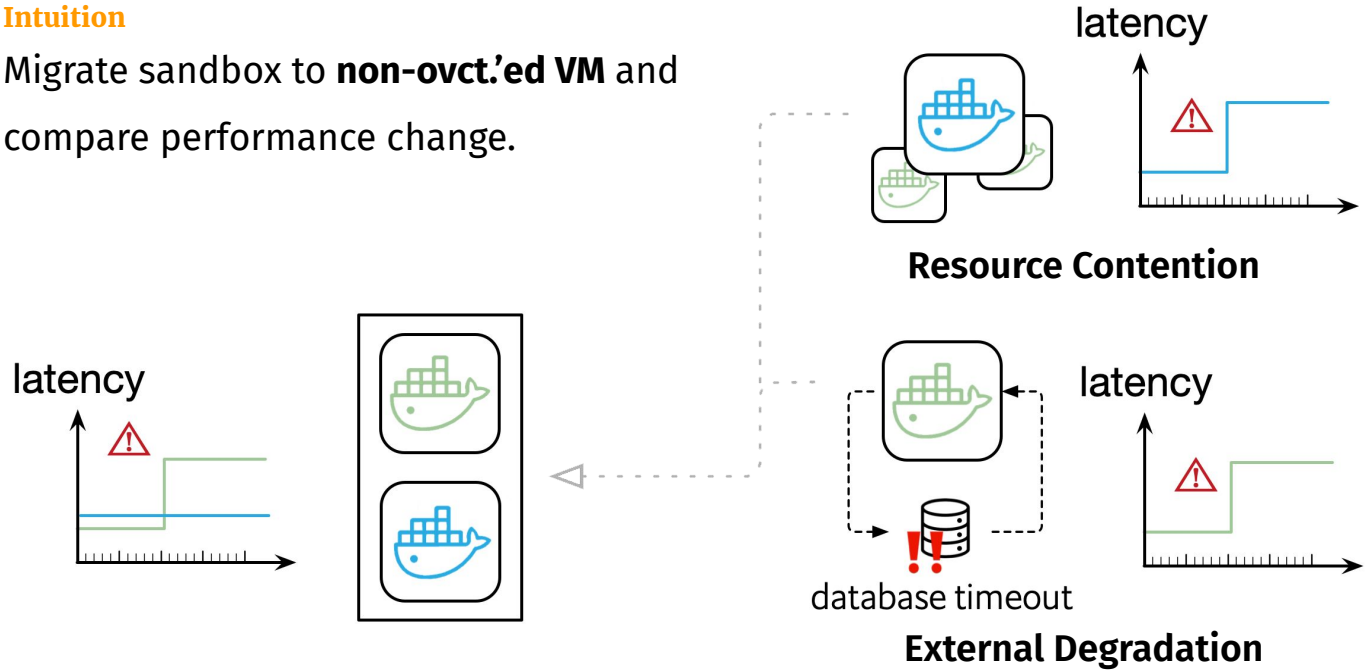
3. **Remedy** degradation (e.g., sandbox migration).



New Technique: Comparative Validation

Intuition

Migrate sandbox to **non-ovct.'ed VM** and compare performance change.



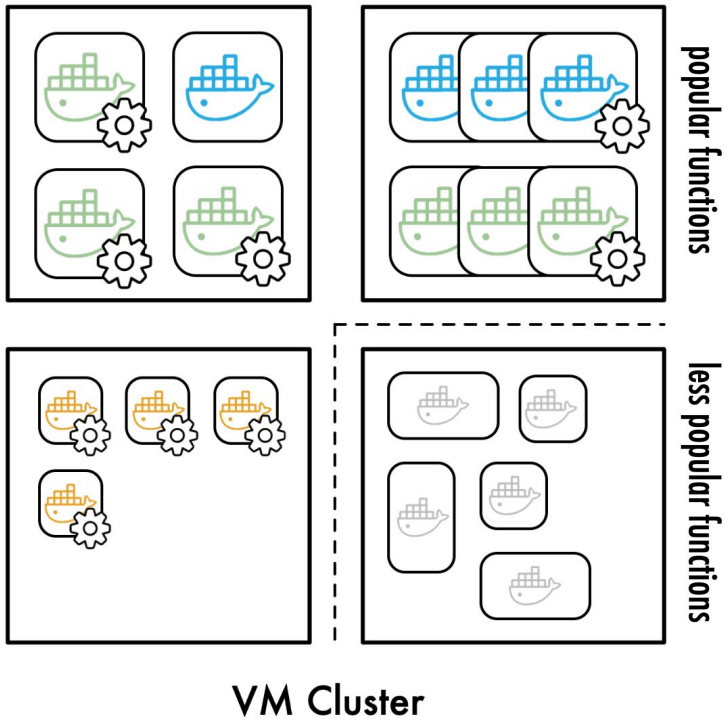
Owl: Putting it all Together

Popular Functions

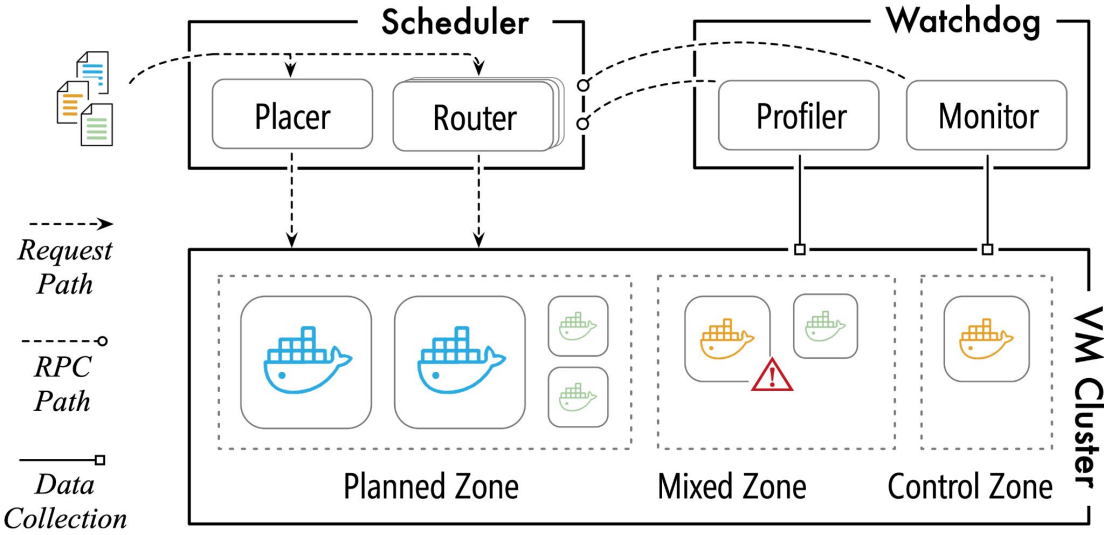
1. Profile Collocations
2. Collocate Sandboxes
3. Consolidate Idle Ones (see paper)

Less Popular Functions

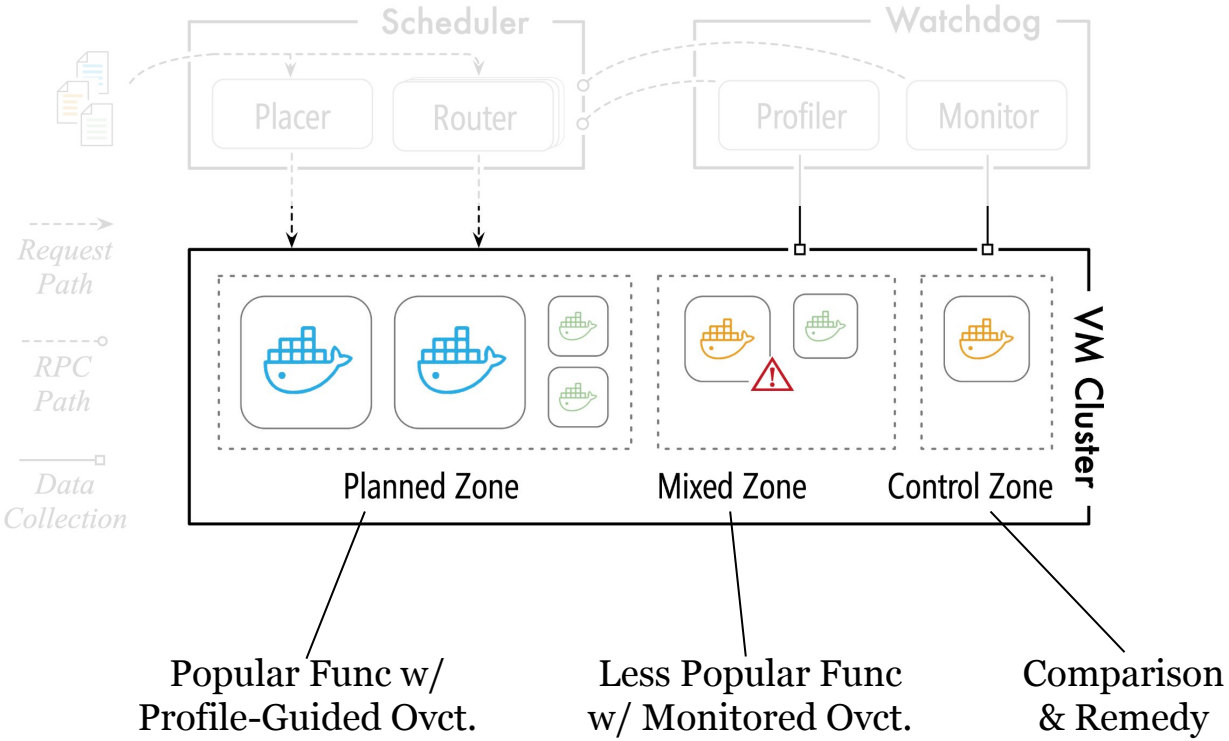
1. Monitor Performance
2. Remedy Degradation
3. Validate Cause



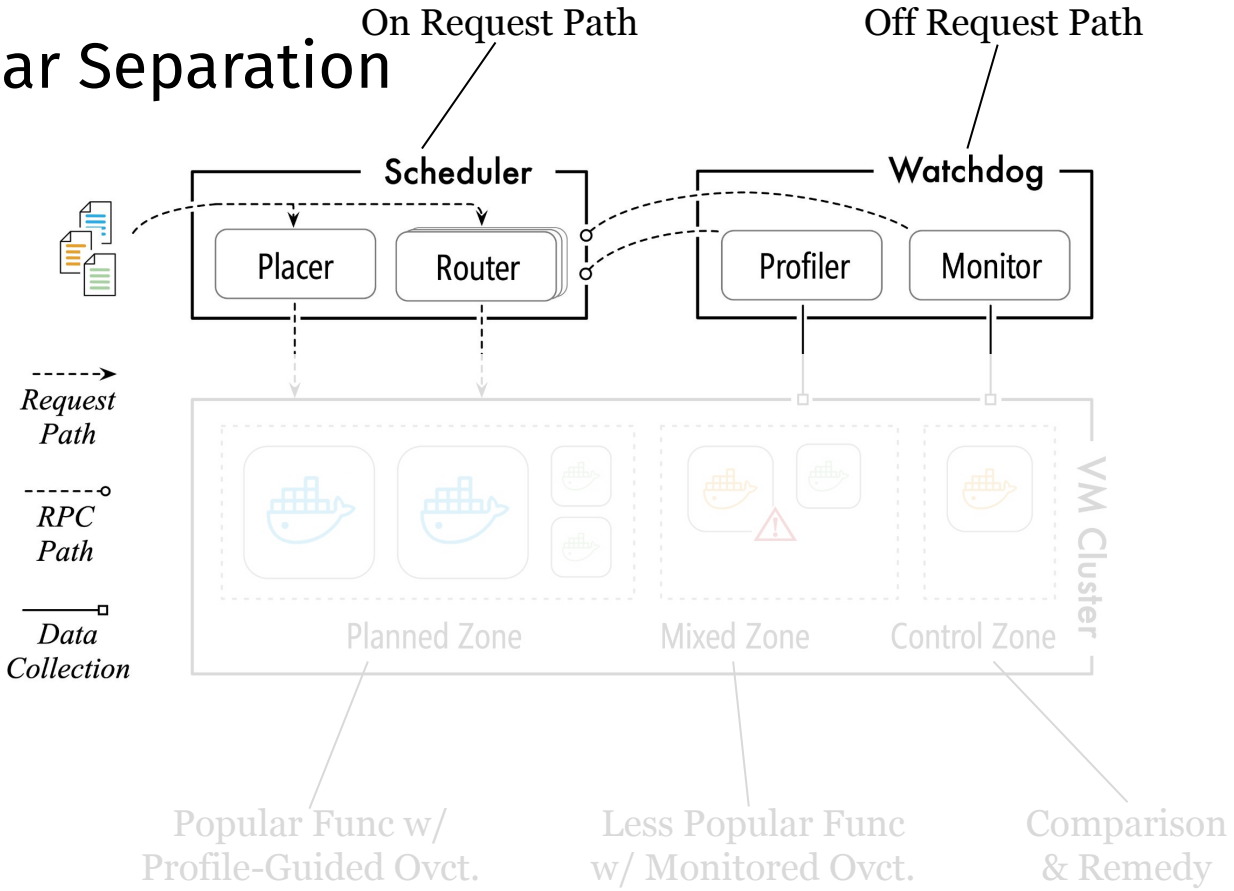
Owl Implementation



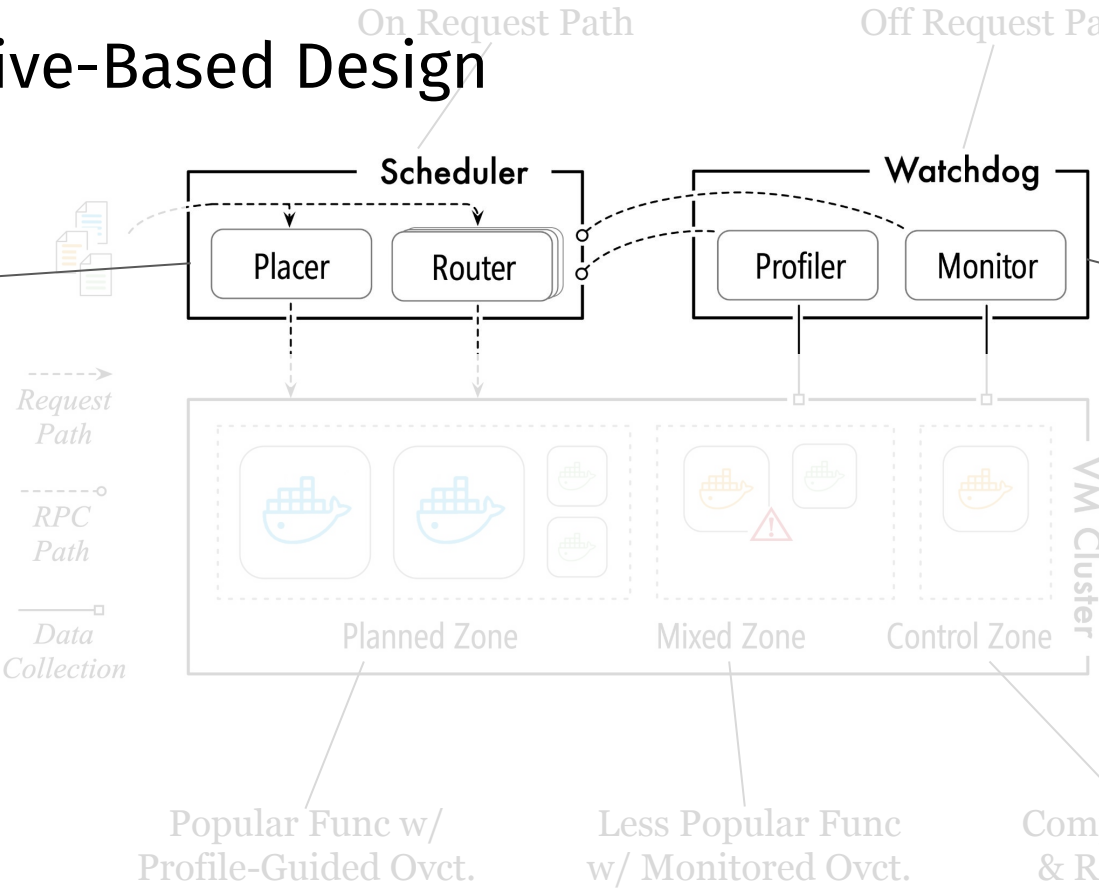
Cluster Zoning



Modular Separation



Primitive-Based Design



Evaluation: Benchmark Design

Goal

Simulate *production-like* workload.

Abbreviation	Function	Memory Size	Actual Usage	Lanaguage	Dependencies
QV	Query Vacancy	256 MiB	~70MiB	JavaScript	Key-Value Store
RS	Reserve Spot	256 MiB	~70MiB	JavaScript	Key-Value Store, Message Queue
AL	Anonymize Log	1024 MiB	~20MiB	Rust	Message Queue
FL	Filter Log				essage Queue
DO	Detect Object				odel Serving Framework
CI	Classify Image	2560 MiB	~500MiB	Python	Model Serving Framework
GMM	Get Media Meta	128 MiB	~20MiB	Python	Object Store
CA	Convert Audio	256 MiB	~100MiB	Python	Object Store
ID	Ingest Data	768 MiB	~10MiB	C++	SQL Database
DA	Detect Anomaly	768 MiB	~10MiB	C++	SQL Database

Open-sourced on GitHub

Evaluation: Settings

Workload

- 20-min production traces …
- … invoking benchmark functions

Metrics

1. VM Cost (# VMs used)
2. Function Tail Latency

Baselines

1. First-Fit
2. Naive Ovct.

Evaluation: Result Highlights

Workload

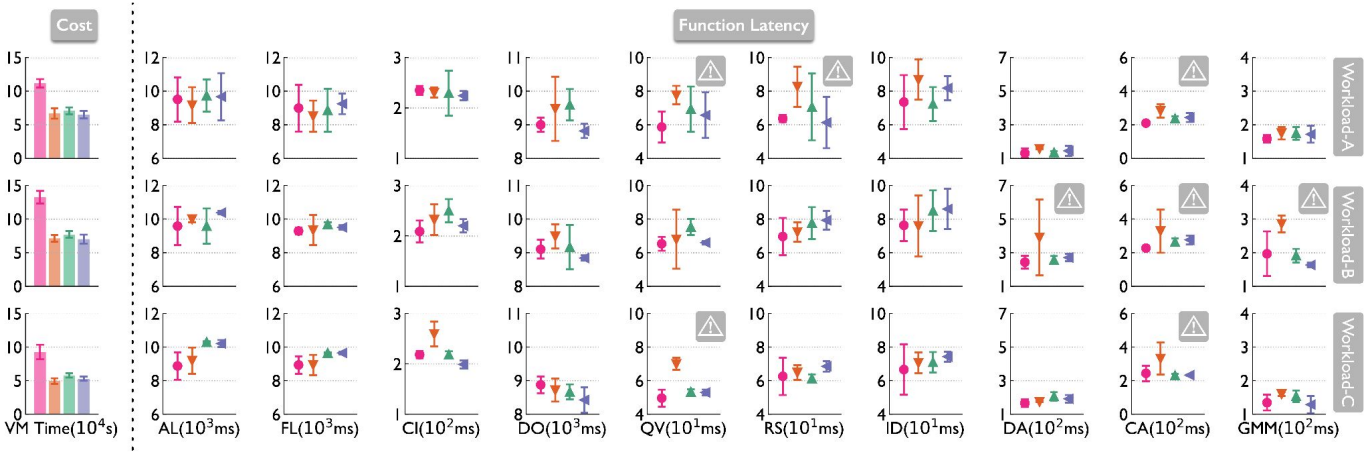
- 20-min production traces …
- … invoking benchmark functions

Metrics

1. VM Cost (# VMs used)
2. Function Tail Latency

Baselines

1. First-Fit
2. Naive Ovct.



Evaluation: Result Highlights

Workload

- 20-min production traces …
- … invoking benchmark functions

Metrics

1. VM Cost (# VMs used)
2. Function Tail Latency

Baselines

1. First-Fit
2. Naive Ovct.

	Cost (norm.)	Performance
First-Fit	1	Normal
Naive Ovct.	0.55	Degraded
Owl	**0.56**	**Normal**

Evaluation: Result Highlights

Workload

- 20-min production traces …
- … invoking benchmark functions

Metrics

1. VM Cost (# VMs used)
2. Function Tail Latency

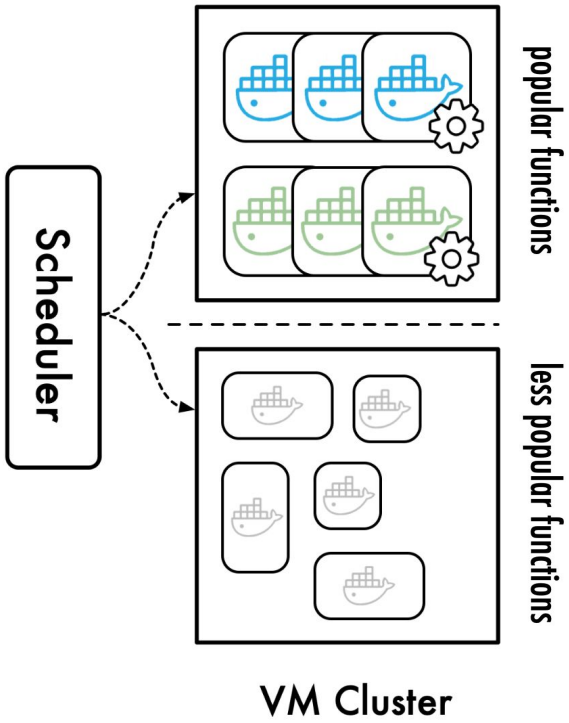
Baselines

1. First-Fit
2. Naive Ovct.

	Cost (norm.)	Performance
First-Fit	1	Normal
Naive Ovct.	0.55	Degraded
Owl	**0.56**	**Normal**

More results in the paper.
(1) scheduling latency
(2) validation effectiveness
(3) large-scale verification

Conclusion: Scheduling for Public FaaS



⚠️ *Restricted* Profiling Environment

- 💡Collocation Profiling
- 💡Profile-Guided Placement

⚠️ *External* Perf. Degradation

- 💡Comparative Validation

Owl: *Resource-Efficient* and *Performance-Aware*