nsdi '26

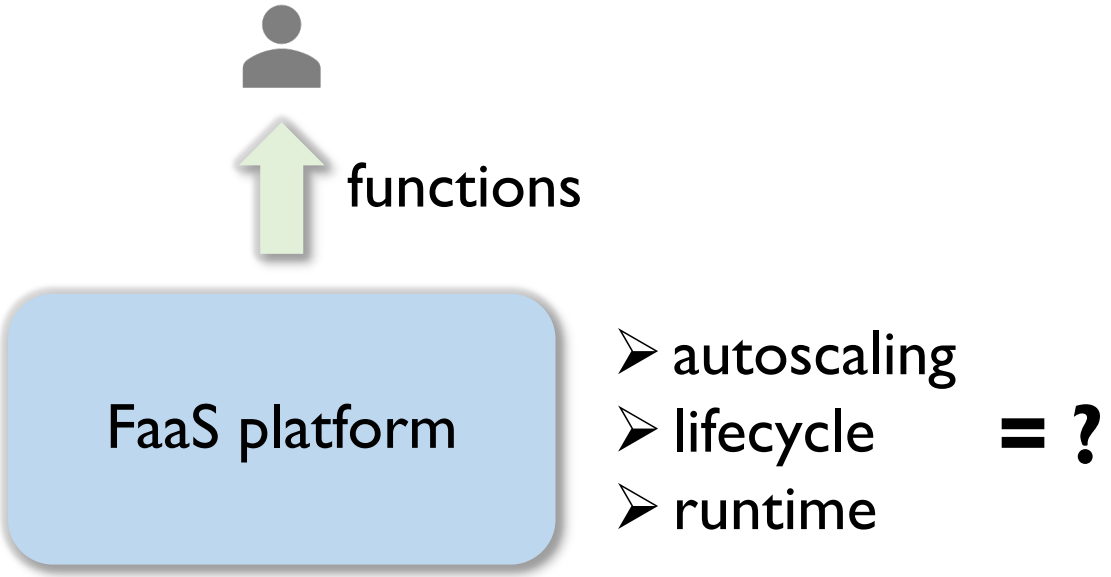
Kubedirect: Unleashing the Full Power of the Cluster Manager for Serverless Computing

Sheng Qi, Zhiquan Zhang, Xuanzhe Liu, Xin Jin

PEKING UNIVERSITY 北京大学

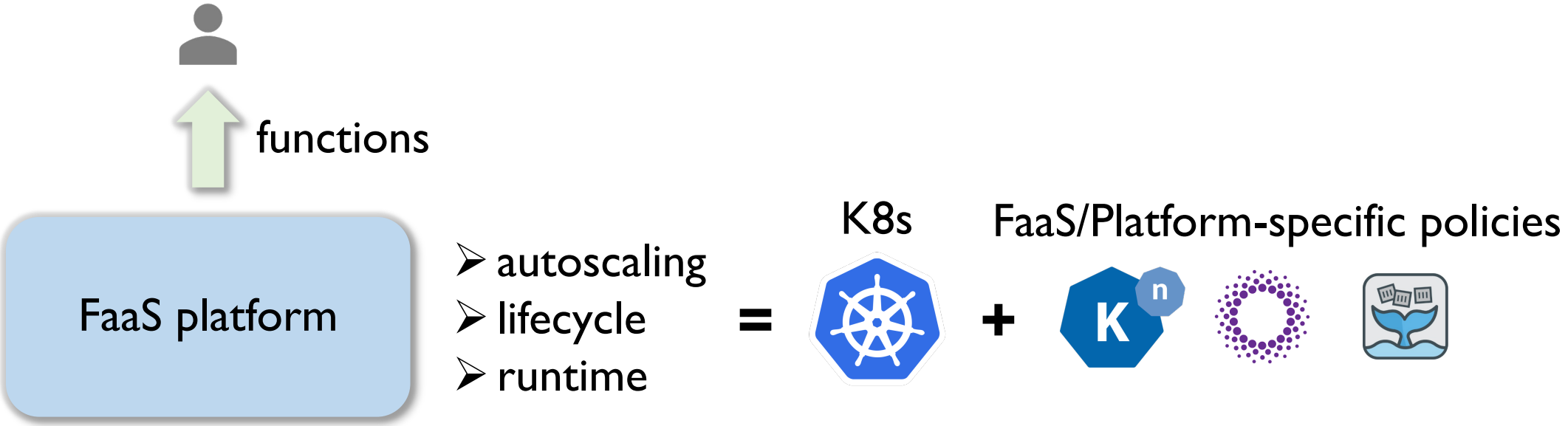
FaaS platforms on top of cluster managers

- FaaS platforms offer simple function-level computing services to users
- Take over low-level resource management responsibilities



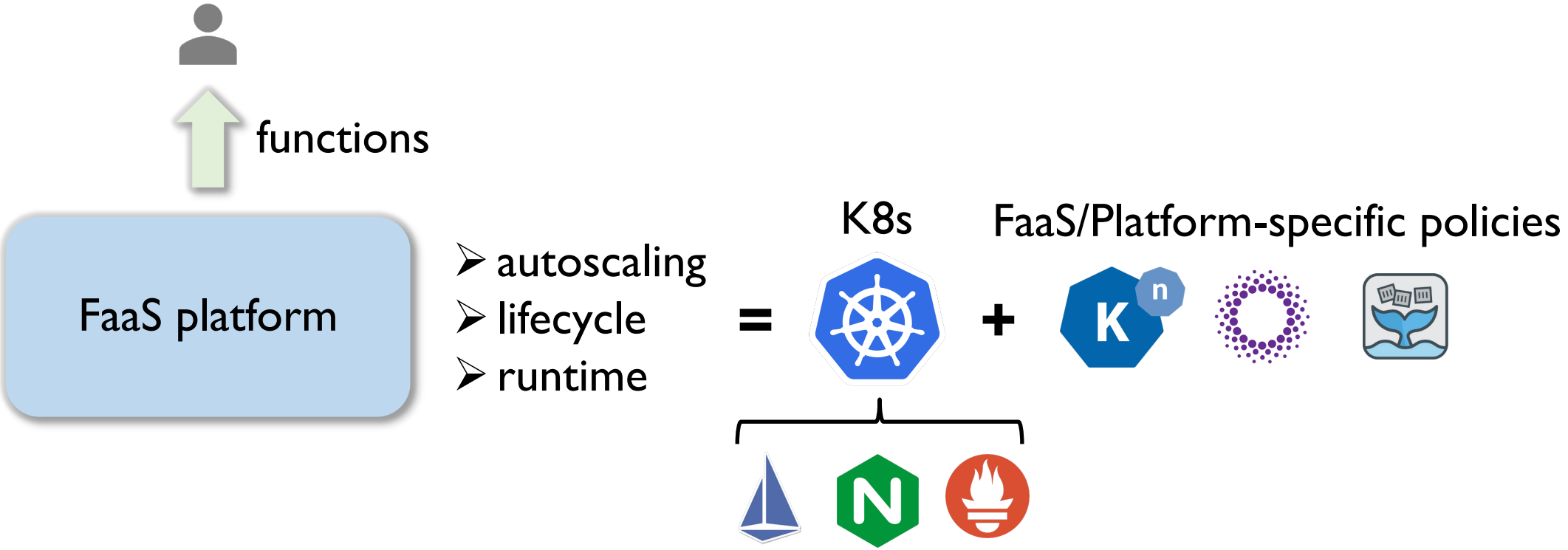
FaaS platforms on top of cluster managers

- Cluster managers like K8s are already good at it
- Extending K8s becomes a popular choice for building FaaS platforms



FaaS platforms on top of cluster managers

- Cluster managers like K8s are already good at it
- K8s also comes with a rich ecosystem of extensions to make FaaS easy



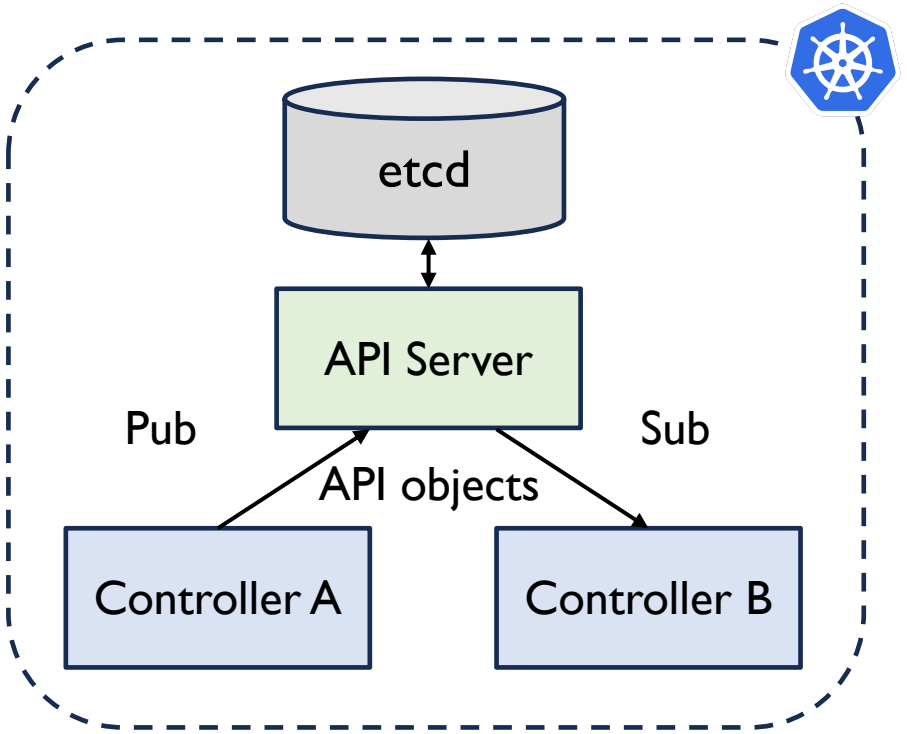
The gap between FaaS and K8s

- K8s' popularity comes from its state-centric design philosophy
- Pro: high extensibility and declarative interfaces

state = collection of API objects

```
# Pod API object
metadata:
  name: my-pod
spec:
  nodeName: worker1
  containers: ...
status:
  phase: Running
  podIP: 10.244.1.1
...
```

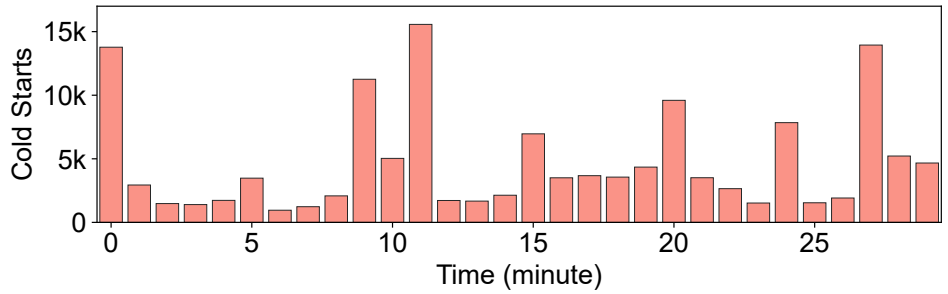
management = collection of controllers



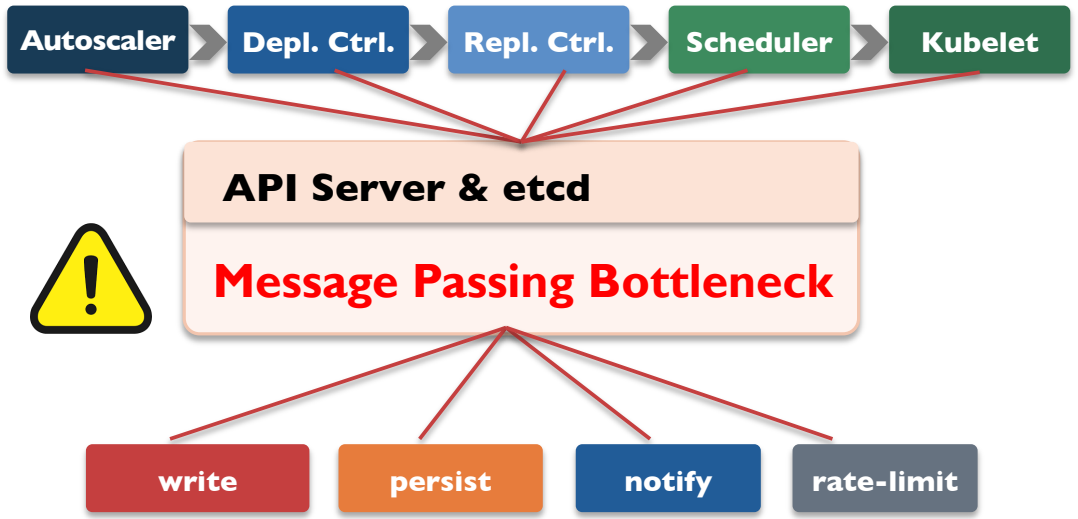
The gap between FaaS and K8s

- Con: extensive state exchange on the critical path
  - hierarchical abstractions + indirect write/notify path
- Unfortunately, FaaS is especially bursty…

10K+ instance creations per minute



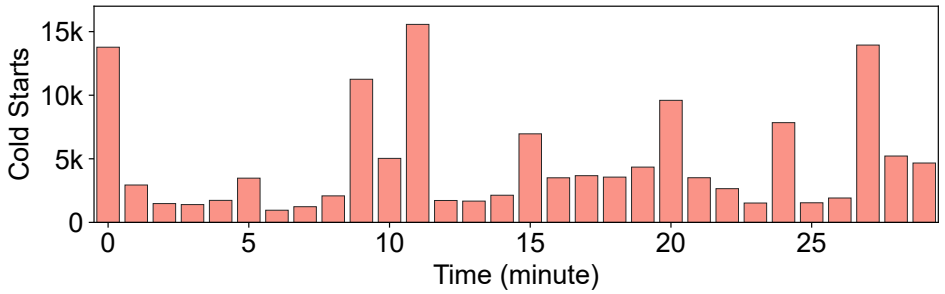
State exchange through K8s



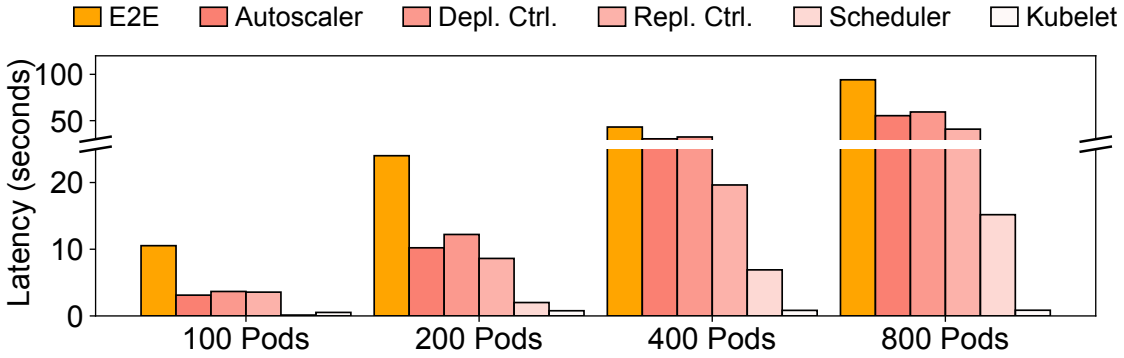
The gap between FaaS and K8s

- Con: extensive state exchange on the critical path
  - hierarchical abstractions + indirect write/notify path
- Unfortunately, FaaS is especially bursty…

10K+ instance creations per minute



tens of seconds for 100s of instances



The gap between FaaS and K8s

- And this happens on the critical path

Functions are short-lived

Average **~800 ms** in the Azure Functions Trace

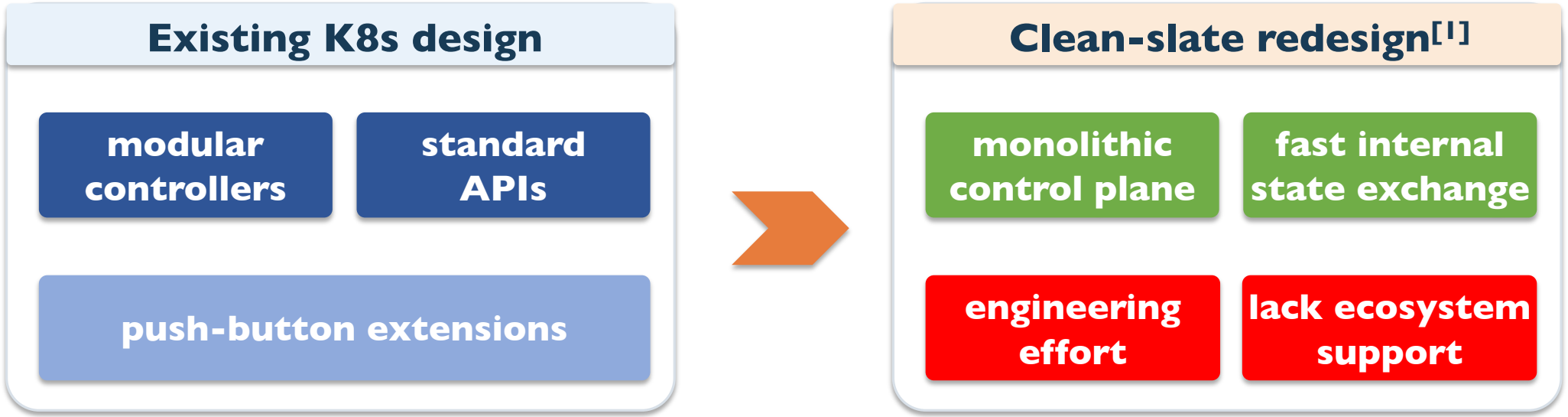
Data plane is fast

Sandbox startup in **orders of milliseconds**

We needs a much faster control plane for FaaS

Performance or compatibility?

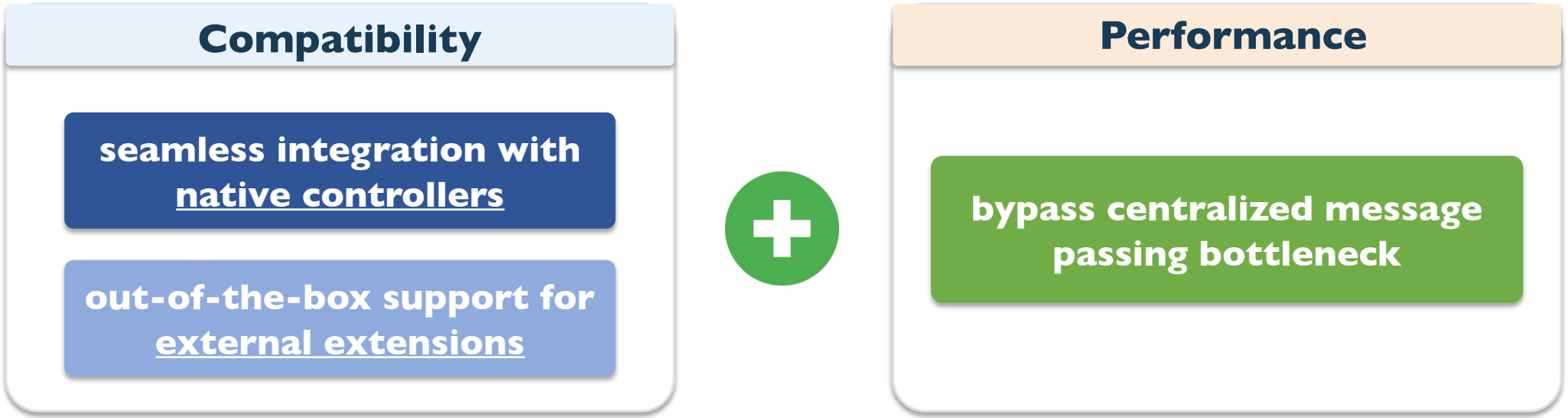
- Rebuilding K8s control plane trades compatibility for performance



[1] Cvetkovic et al. Dirigent: Lightweight Serverless Orchestration. In SOSP'24

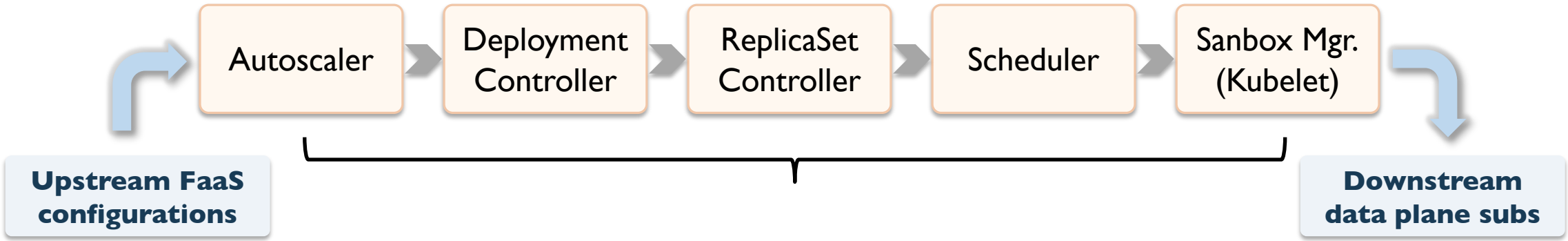
Performance or compatibility?

- We want to have both…which lead to
- Kubedirect: a K8s-like cluster manager, but scales out much faster



Opportunities

- There is a common "narrow waist" in K8s-based FaaS platforms



Why this is the right scope for optimization?

- Dominates critical path latency
- Cleanly separated from upstream/downstream

Opportunities

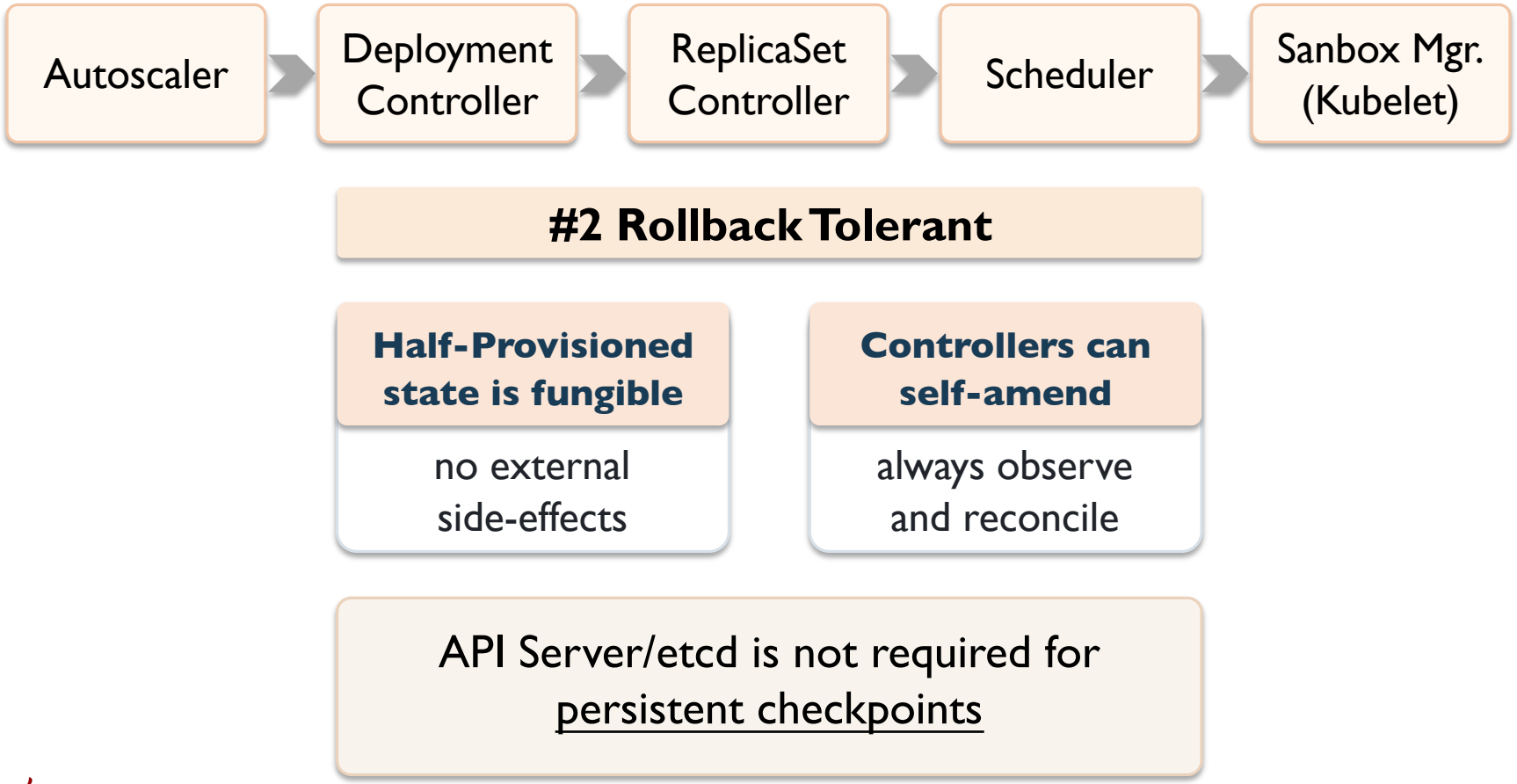
- The narrow waist has two nice properties for API Server bypassing

#1 Sequential Writes

API Server is not required for conflict resolution

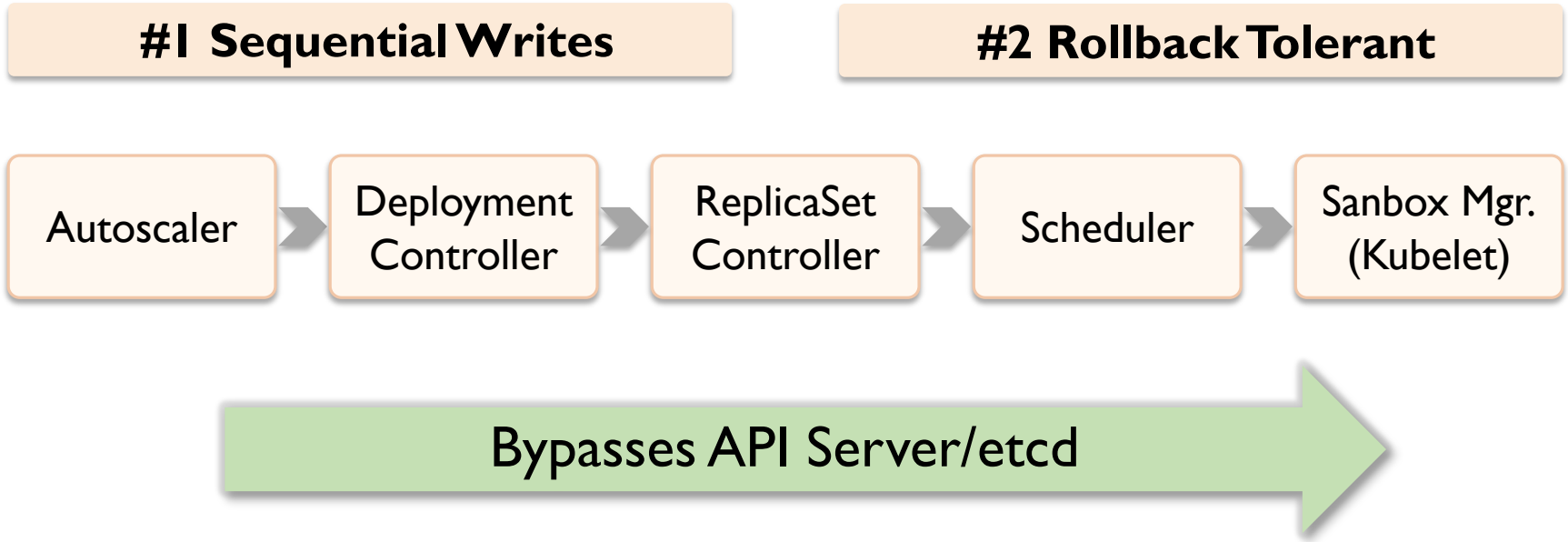
Opportunities

- The narrow waist has two nice properties for API Server bypassing



Opportunities

- Therefore, Kubedirect adopts P2P direct message passing



Challenges

1. State Consistency

How do controllers agree on cluster state without a single source of truth?

2. Lifecycle Semantics

Provisioning can rollback

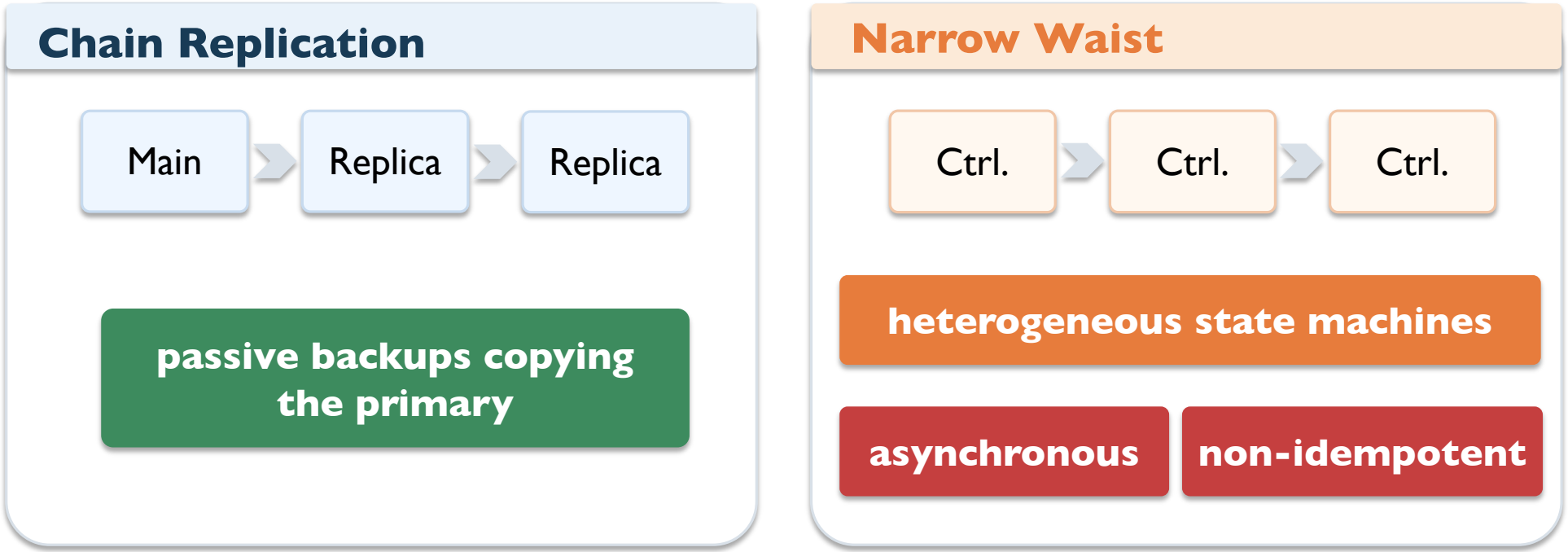
Termination cannot

3. Integration Constraints

We want minimal controller changes, not a rewrite

Problem Analysis

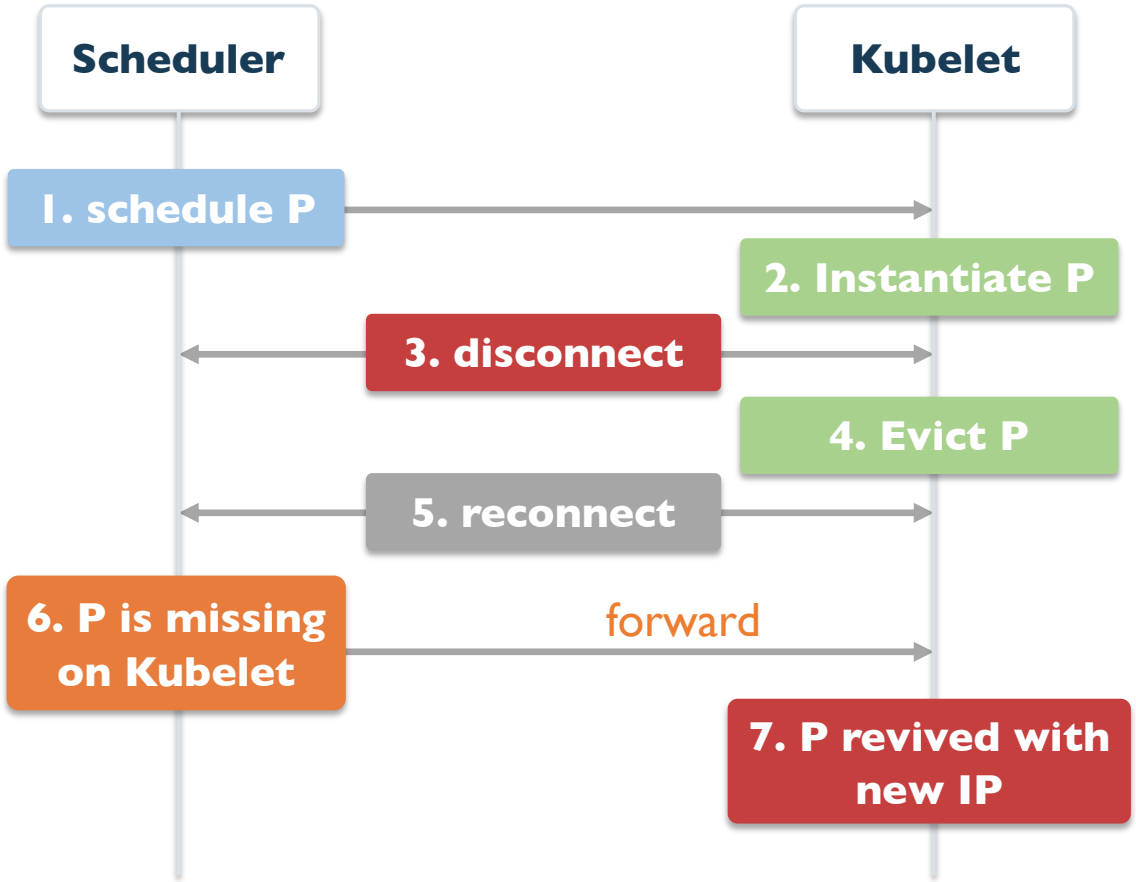
- Comparison with Chain Replication



The chain shape is similar, but the semantics are much harder

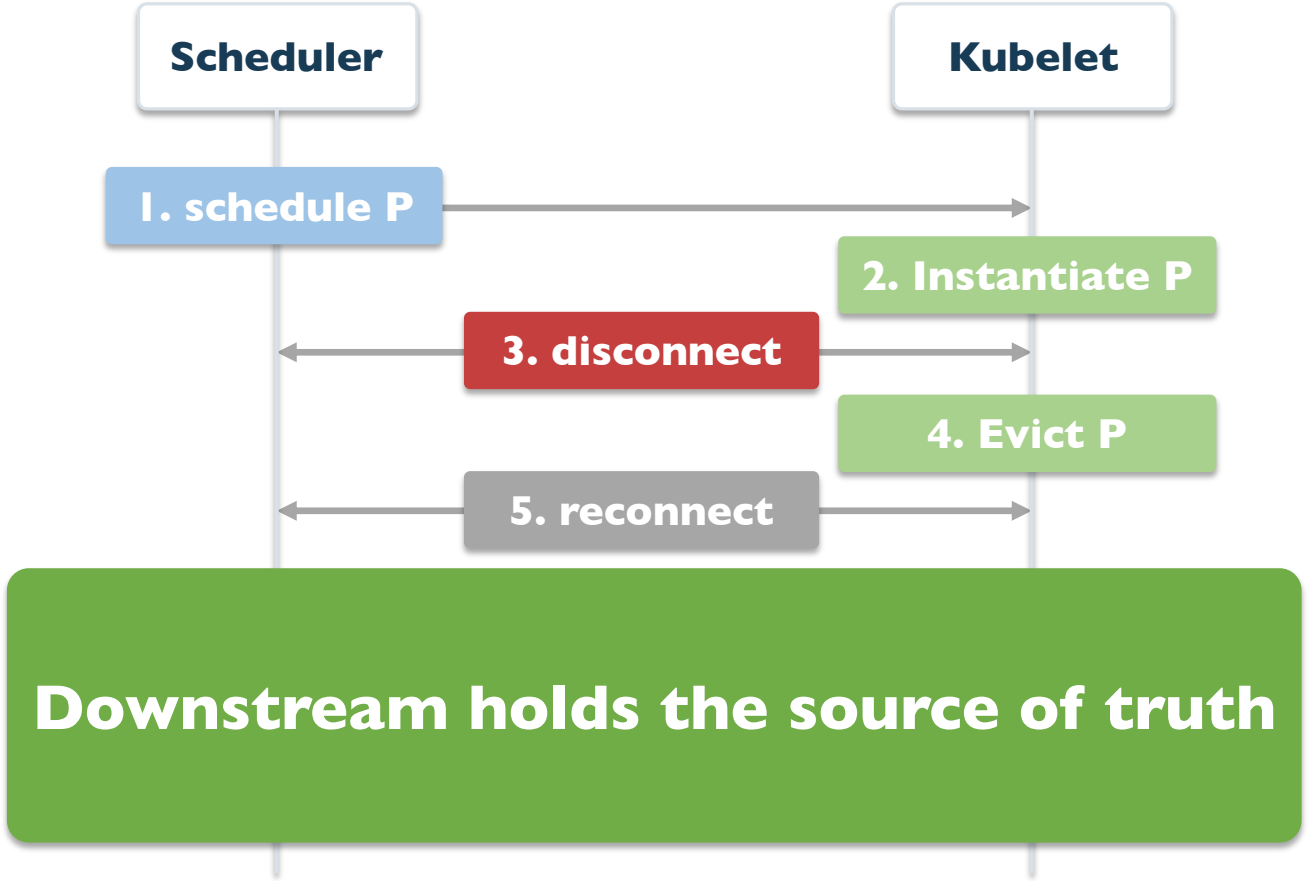
Problem Analysis

- Why fast-forwarding can go wrong



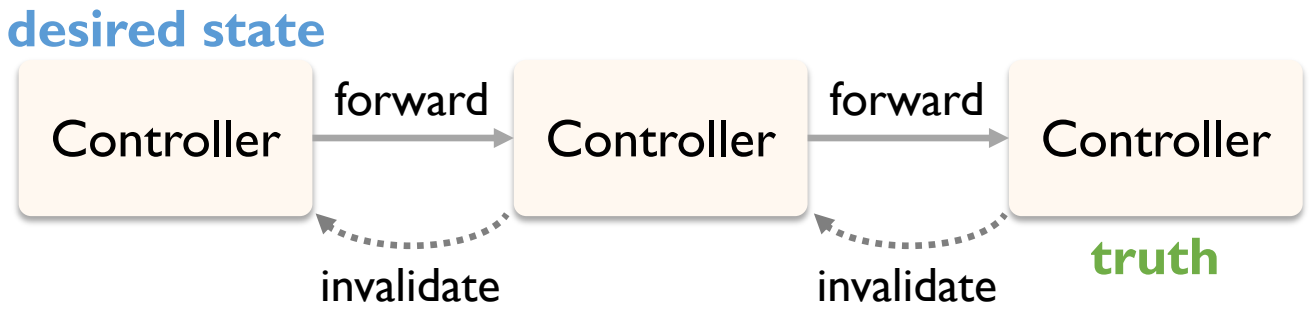
Problem Analysis

- Why fast-forwarding can go wrong



Hierarchical write-back cache

- The right state model is a hierarchical write-back cache
  - opportunistically forward desired state
  - feedback downstream truth as cache invalidation



Pairwise agreements replace centralized coordination

Hierarchical write-back cache

- Two types of invalidation signals

Hard Invalidation

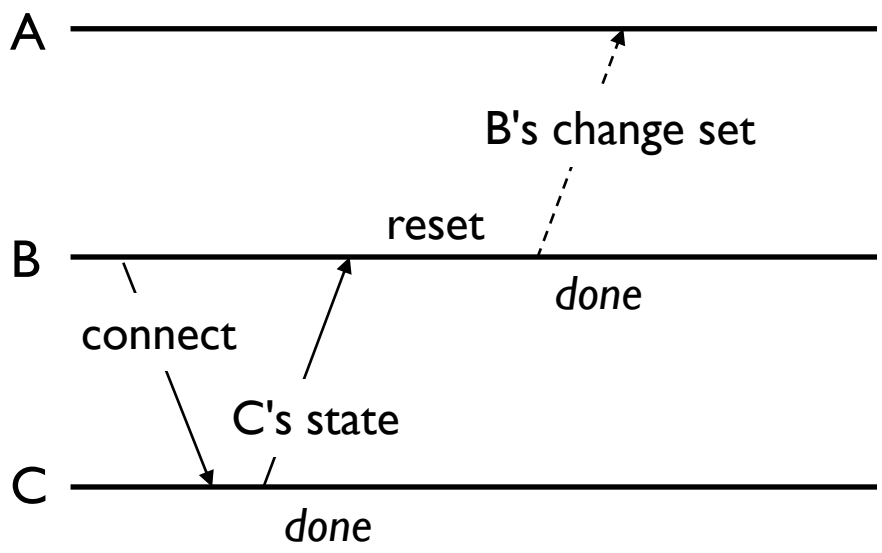
- On disconnect/crash
- Forcefully rollback state
- Synchronous handshake

Soft Invalidation

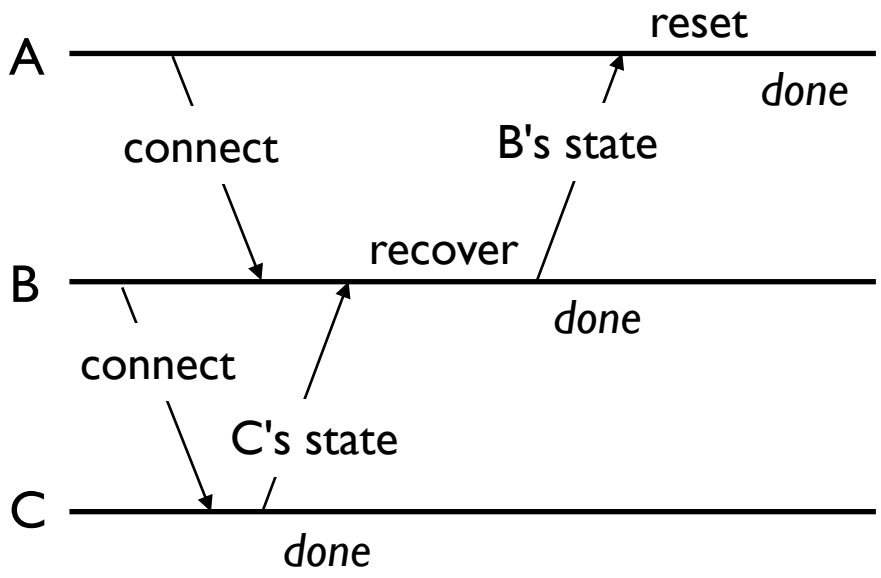
- On live connection
- Incremental changes
- Asynchronous feedback

Examples

(a) B-C disconnection



(b) B crash failure



Lifecycle semantics on termination

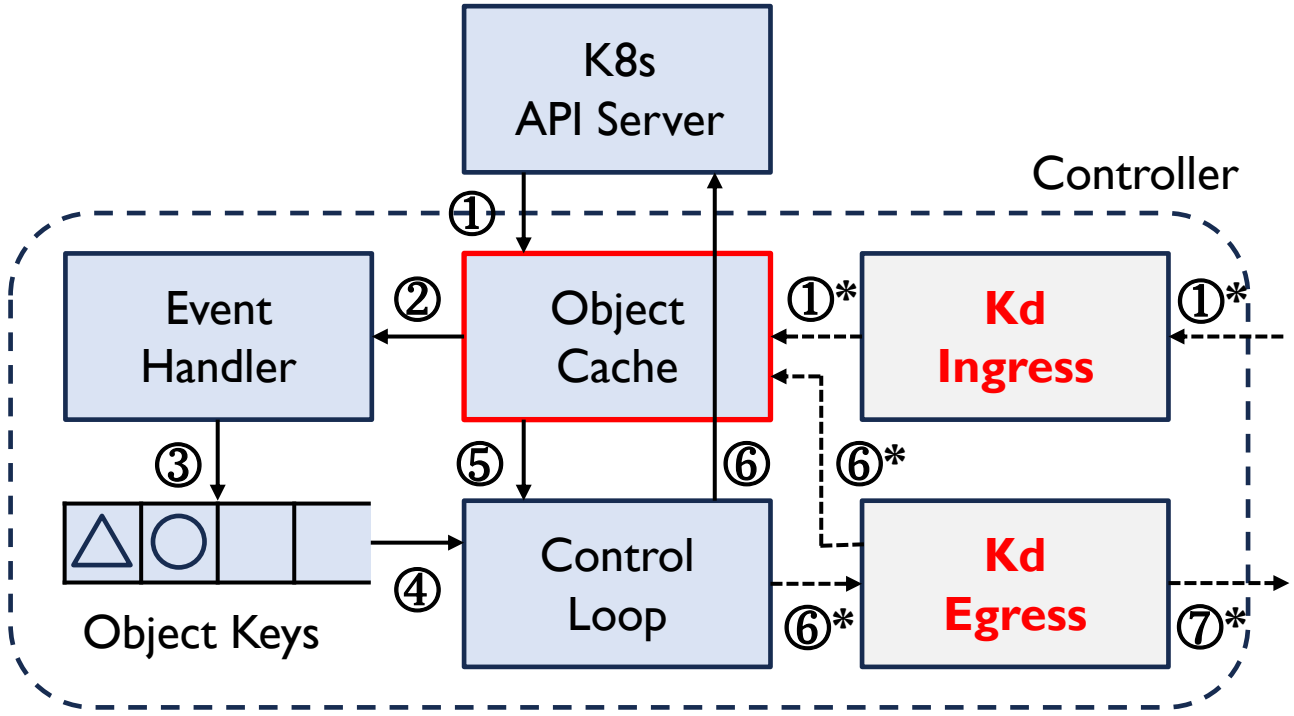
- Provisioning can rollback; termination cannot
- Solution: replicate *Tombstones*

Tombstones

- Replicate termination intent downstream
- Inherently Idempotent → CR-style replication
- Covers downscaling and preemption

Integration with controllers

- Extra ingress/egress modules for propagation/invalidation
- Normalized by common object cache; transparent to control loop



~150 LoC

per controller

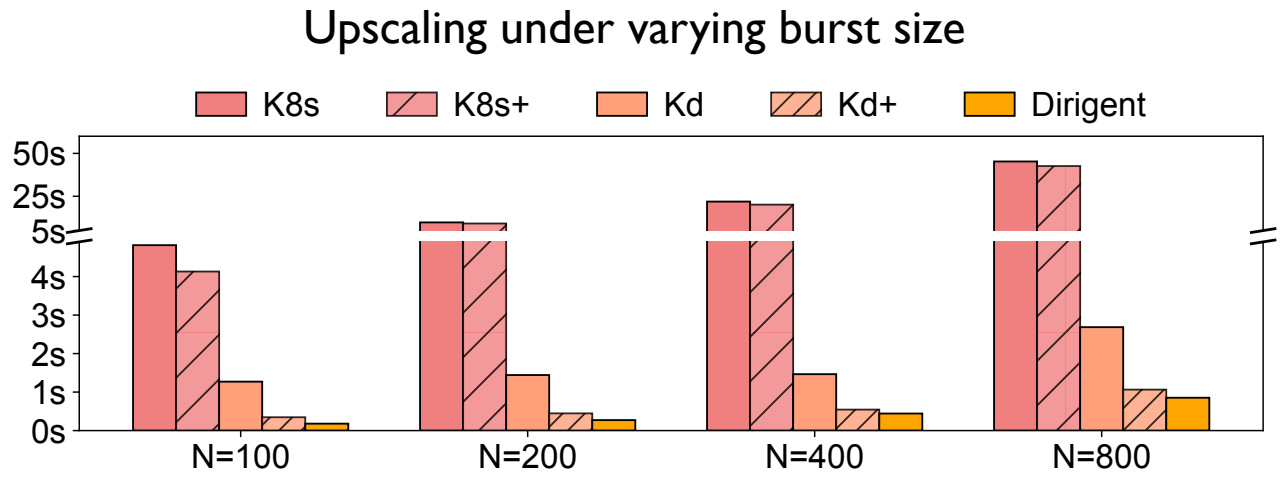
Extensions

Prometheus and Istio support

Evaluation Setup

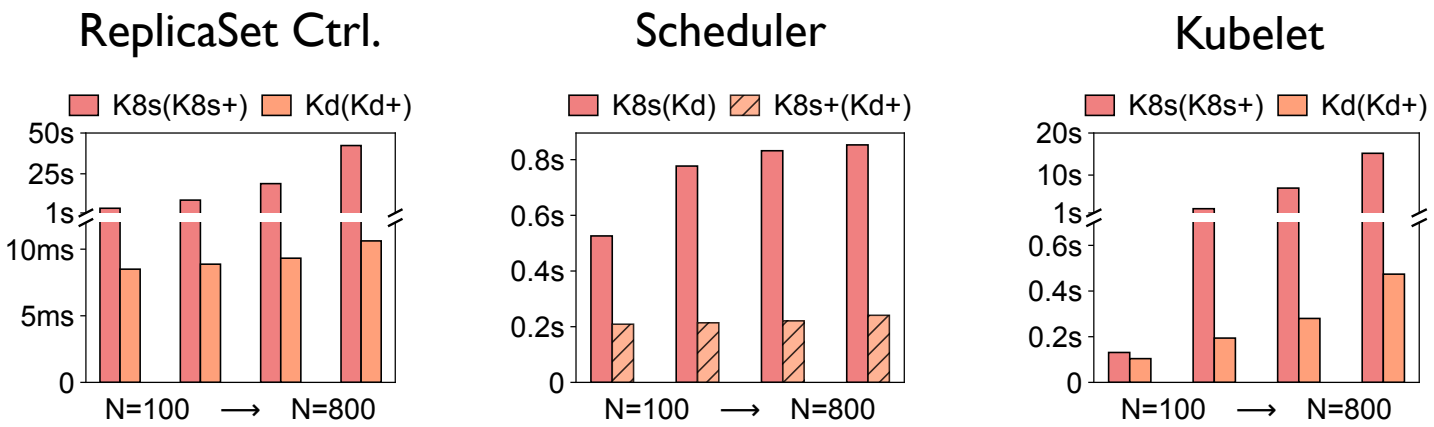
- 80-node cluster on CloudLab
  - 10 CPU cores, 64 GB RAM, 25 Gbps interconnect
- Cluster-manager-level baselines
  - K8s v1.32.0
  - Kubedirect (Kd), also based on K8s v1.32.0
- FaaS-platform-level baselines
  - Knative v1.15.0
  - Dirigent

Microbenchmarks



Total upscaling latency

3.7x to 16.9x faster



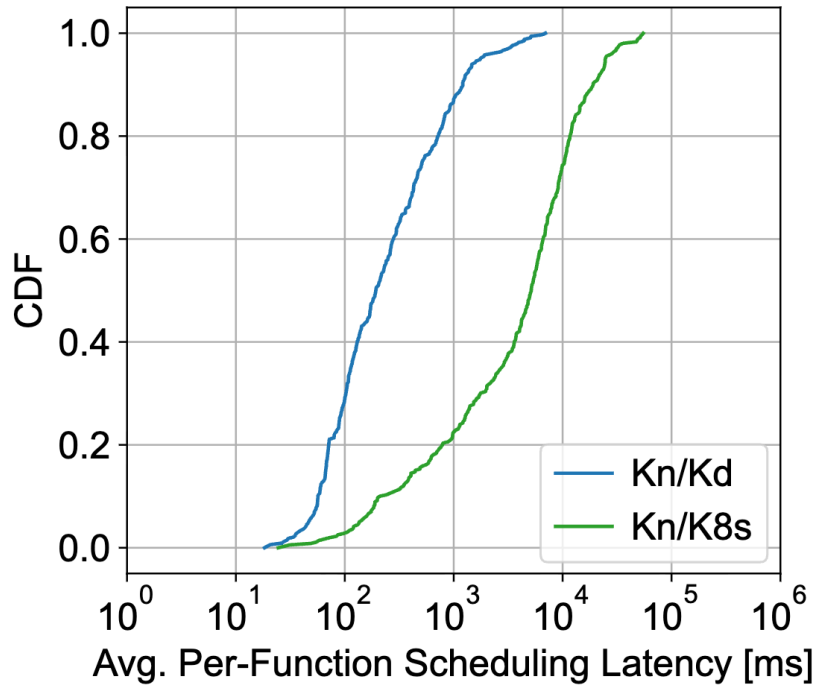
Latency breakdown

Confirms that message passing is no longer the bottleneck

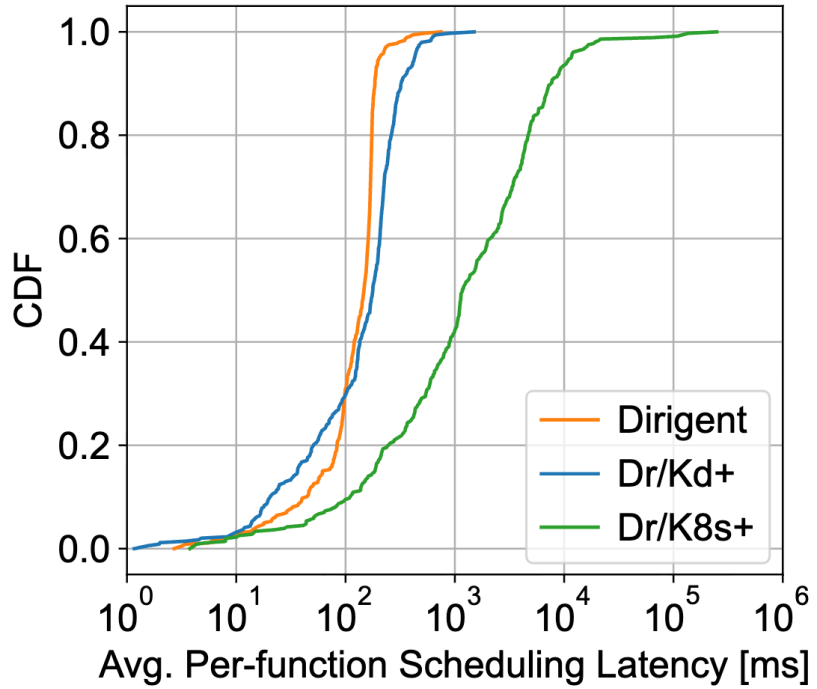
End-to-end performance

- Measured on a 30-minute clip of the Azure Functions trace

26.7x lower scheduling delay vs. Knative



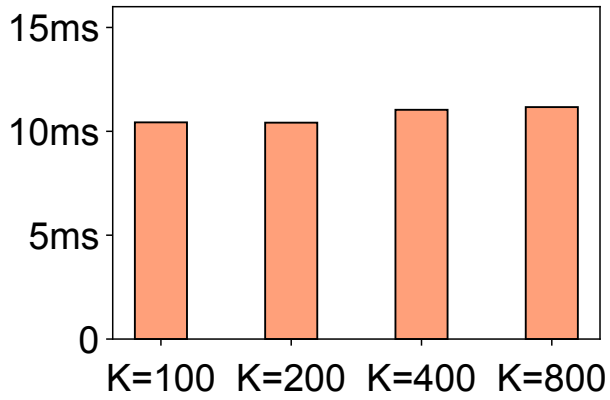
performance on par with Dirigent



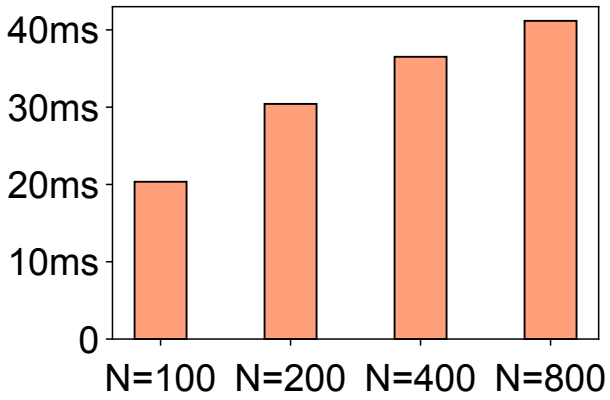
Failure recovery with cache invalidation

recovery stays cheap and scalable

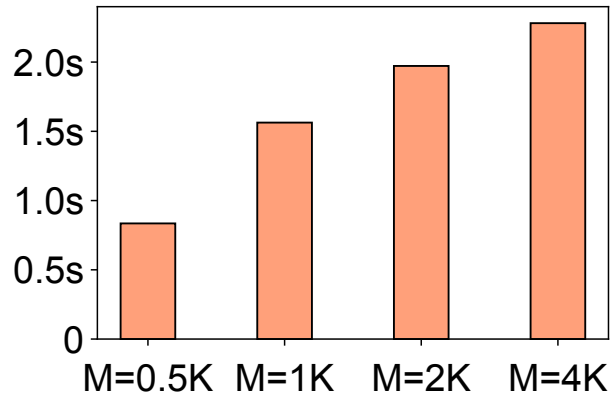
Autoscaler



ReplicaSet controller



Scheduler



Conclusion

- Kubedirect retrofits K8s for FaaS without compromising compatibility
  - Use direct message passing to avoid centralized bottlenecks
  - Use pairwise agreements to replace centralized coordination
- Limitations
  - Reduced observability within the narrow waist
  - Assume trusted tenants due to AuthN/AuthZ bypassing

Source code github.com/TomQuartz/kubedirect-ae