
ebay
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Sustainable Scaling of Kubernetes Workloads with In-Place Pod Resize and Predictive AI

Vinay Kulkarni, eBay Inc.

Haoran Qiu, UIUC

ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Agenda

- The Overprovisioning Problem
- Environmental Impact and Cost of Over-Provisioning
- eBay Sustainability Roadmap
- In-Place Pod Resize Recap
- Cluster Autoscaler Use Case
- Cluster Autoscaler with In-Place Pod Resize
- Overview of AI in Cloud & Current State of Affairs
- Reinforcement Learning (RL) & Autoscaling with RL
- Multidimensional Pod Autoscaling
- Reactive → Proactive Autoscaling
- RL Training and Results

The Overprovisioning Problem

- Estimating pod resource needs is HARD!!
- Application load-time vs. run-time resource needs
- Load unpredictability
- Expected vs. actual code paths taken
- Responsiveness of downstream dependency services
- Until recently, immutable pod resources *

Cloud Computing Environmental Impact

KubeCon CloudNativeCon North America 2023

[1,2]

The Cloud now has a greater carbon footprint than the airline industry. A single data center can consume the equivalent electricity of 50,000 homes.

According to a Lawrence Berkeley National Laboratory report, if the entire

[3]

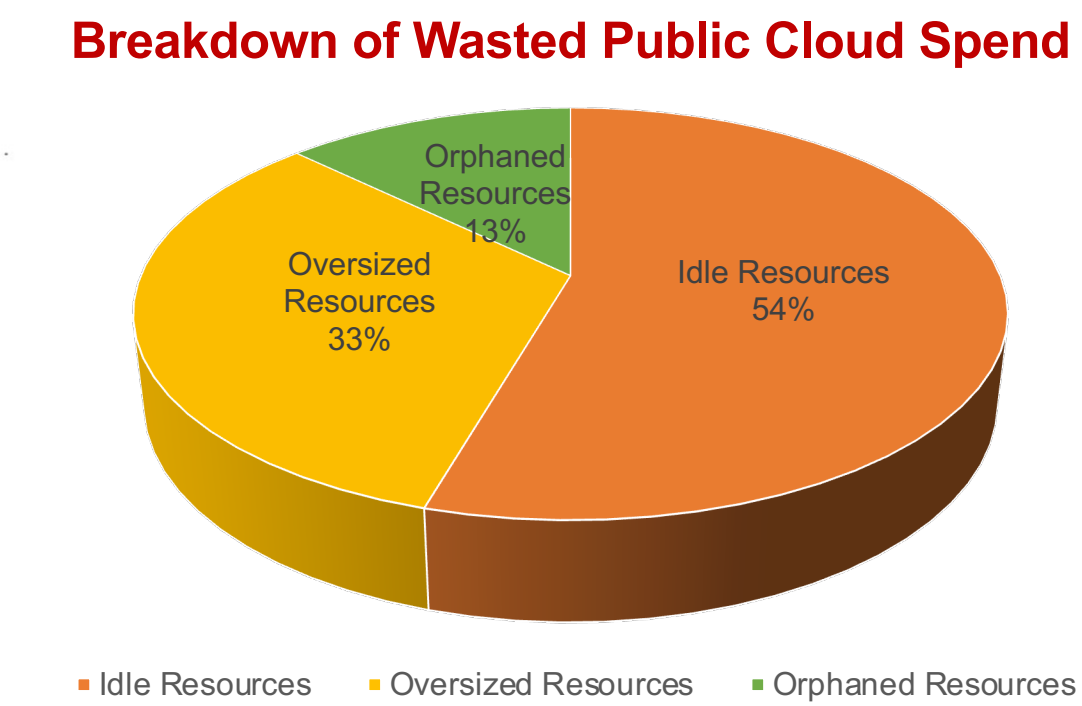
- 1.8% U.S. electricity consumption, according to W.E.F
- Estimated 300 metric tons of CO2 in 2020 from cloud computing needs
- Google reported 450,000 gallons of water per day for one datacenter in 2021
- Electronic waste and noise

ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Cost of Overprovisioning

- In 2021, an estimated $26.6 billion spent (wasted) in public cloud was due to overprovisioning and always-on resources [4]
- In a 2022 survey report from StormForge, an estimated 47% of Kubernetes and cloud waste was from overprovisioning [5]



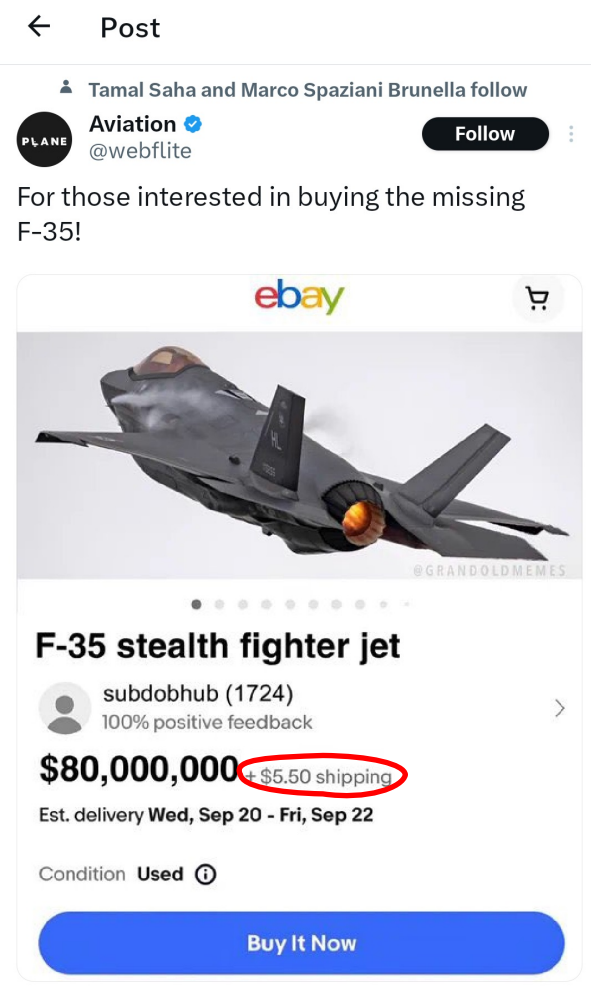
Sustainability Roadmap at eBay

- Natural side effect of the business (recommerce)
- Goal: Data centers exclusively use renewable energy

  → Predictive AI assisted horizontal pod autoscaling

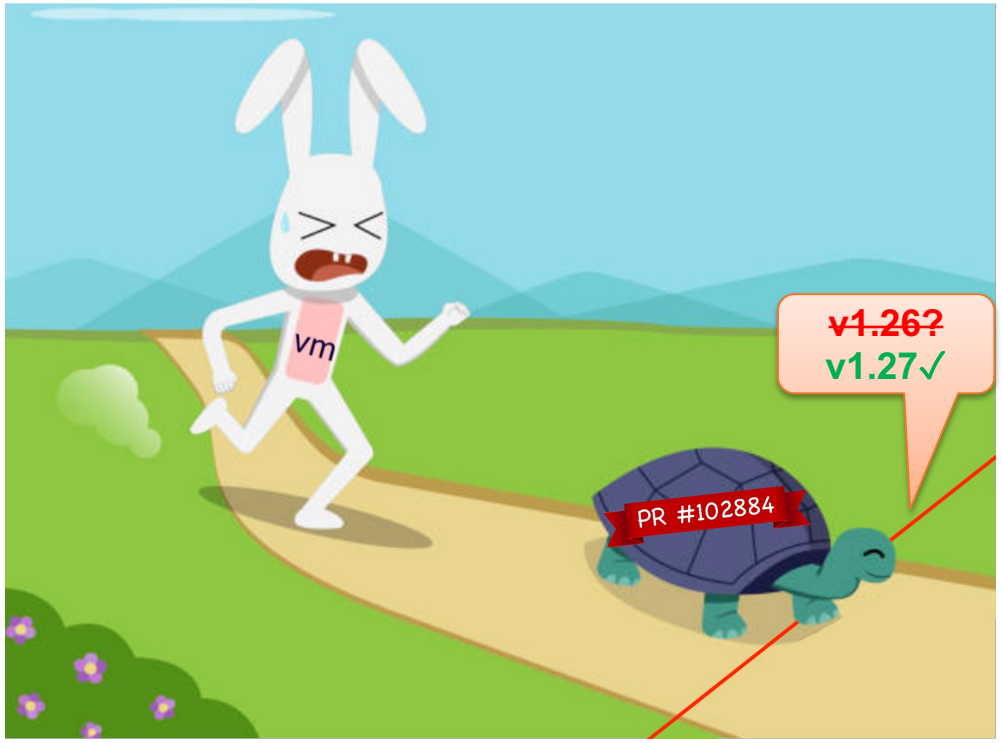
  → Vertical pod autoscaling is next

  → Multidimensional pod autoscaling?



In-Place Pod Resize Recap

- Until K8s v1.26, pod resources were immutable
- K8s v1.27+, pod spec resources represent **desired resources** for CPU and memory (alpha feature)
- Enables you to right-size over/under provisioned workloads in-place, without restarts
- Blog: https://kubernetes.io/blog/2023/05/12/in-place-pod-resize-alpha
- Talk: https://kccncna2022.sched.com/event/182HU
- KEP: https://github.com/kubernetes/enhancements/tree/master/keps/sig-node/1287-in-place-update-pod-resources



Faster Java App Startup Time

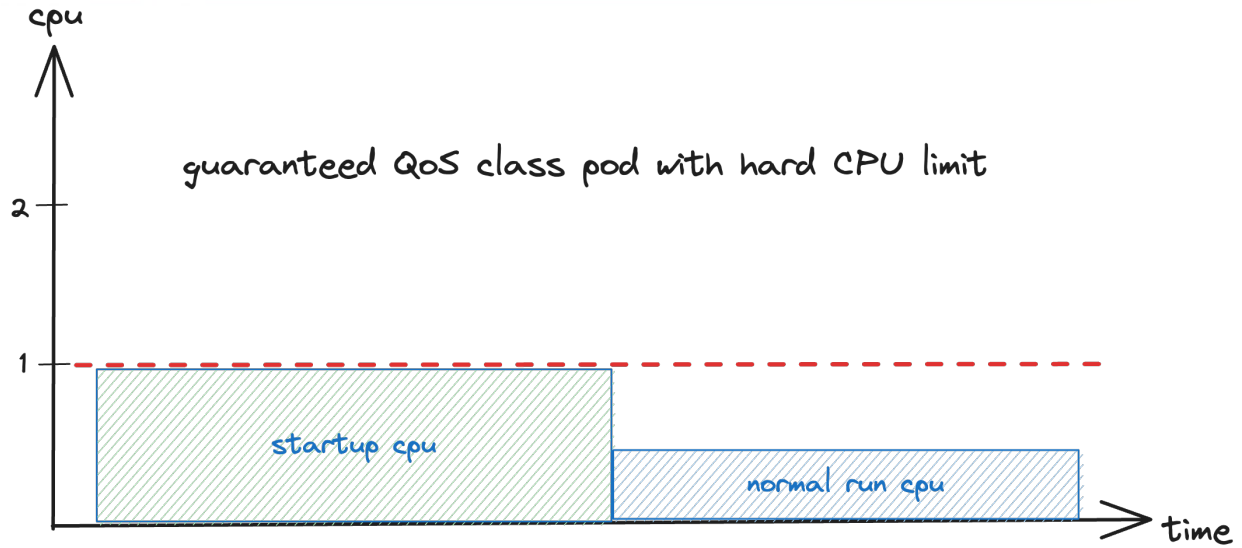
Kubernetes, Performance

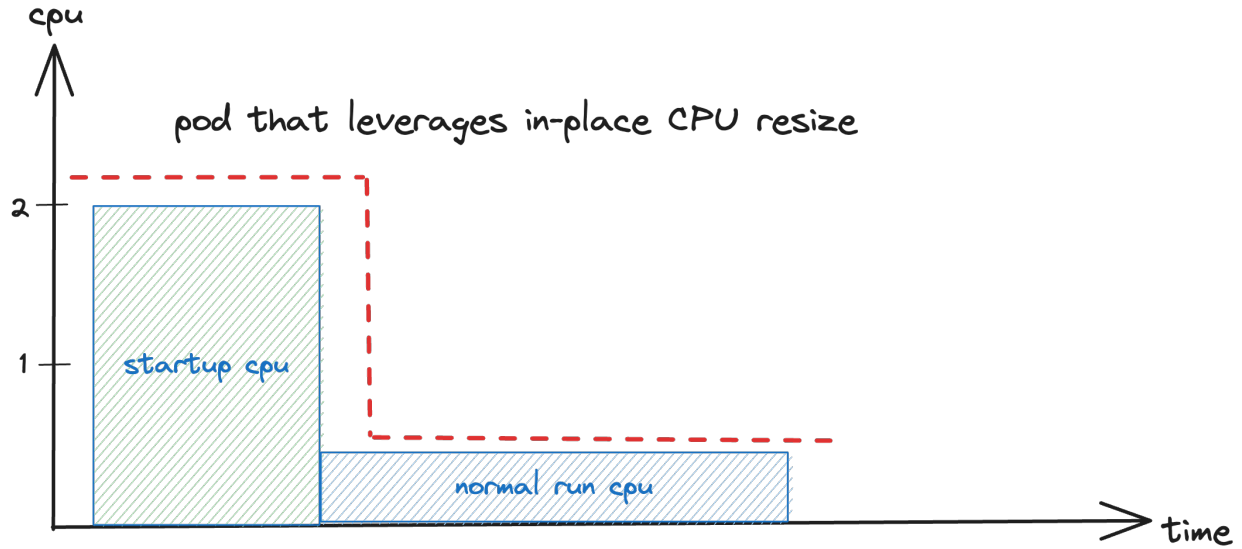
Resize CPU Limit To Speed Up Java Startup on Kubernetes

By piotr.minkowski • August 22, 2023 • 10

In this article, you will learn how to solve problems with the slow startup of Java apps on Kubernetes related to the CPU limit. We will use a new Kubernetes feature called *"In-place Pod Vertical Scaling"*. It allows resizing resources (CPU or memory) assigned to the containers without pod restart. We can use it since the Kubernetes 1.27 version. However, it is still the alpha feature, that has to be explicitly enabled. In order to test we will run a simple Spring Boot Java app on Kubernetes.

6



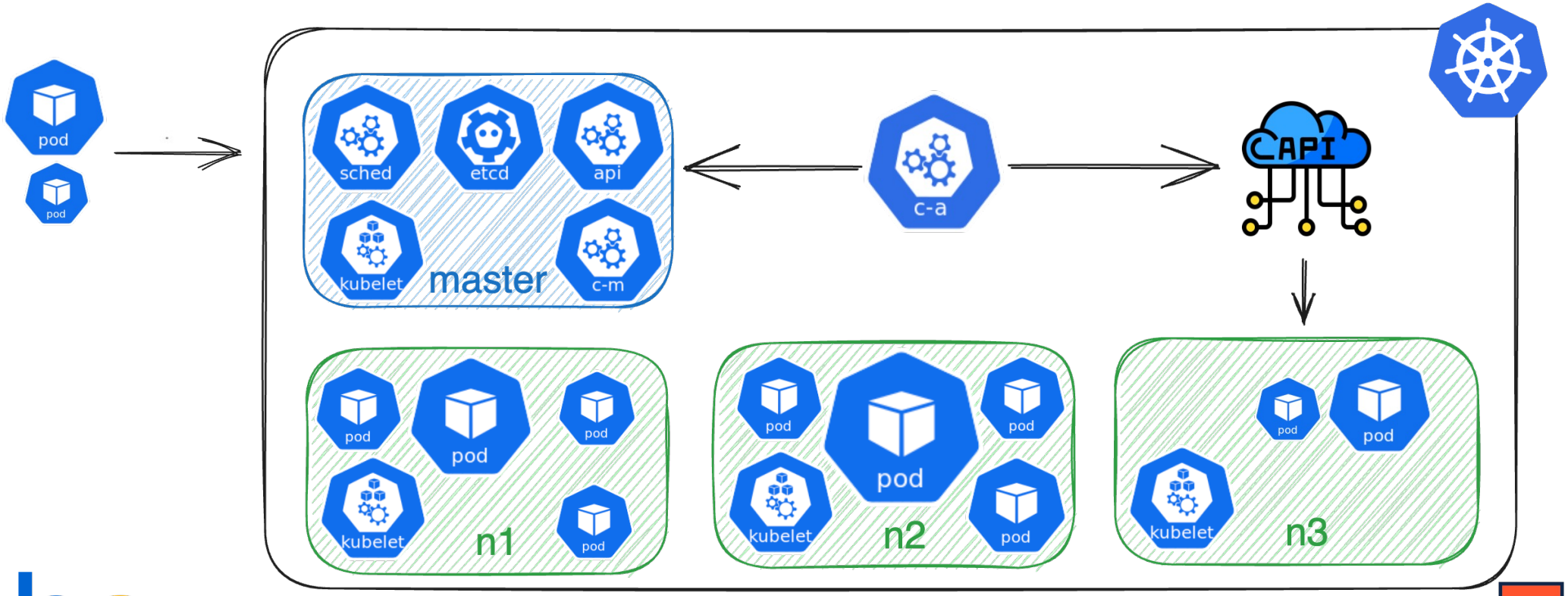


ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Cluster Autoscaler Use Case

- Schedule pods that are pending due to ‘not enough resources’
- Allocates new node(s) to create the capacity that meets pending pod requests
- Calls cloud provider API to allocate the new node(s)



Kubeadm Cluster Autoscaler Demo

KubeCon CloudNativeCon North America 2023

```
NAME     STATUS   ROLES           AGE    VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
master   Ready    control-plane   6d12h  v1.28.2   192.168.105.19   <none>        Ubuntu 22.04.3 LTS   5.15.0-87-generic   containerd://1.6.24
node1    Ready    <none>          6d12h  v1.28.2   192.168.105.16   <none>        Ubuntu 22.04.3 LTS   5.15.0-86-generic   containerd://1.6.24
```

vikulkarni — root@master: ~ — ssh root@192.168.105.19 — 132×8

```
Every 1.0s: kubectl get pods -owide                                    master: Mon Oct 30 04:48:32 2023

NAME    READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
pod1    1/1     Running   0          31m   10.0.1.214   node1   <none>           <none>
pod2    1/1     Running   0          31m   10.0.1.204   node1   <none>           <none>
```

vikulkarni — root@master: ~/demo — ssh root@192.168.105.19 — 172×29

~ — root@master: ~/demo — ssh root@192.168.105.19 | ...uster-autoscaler/cloudprovider/externalgrpc/kubeadm_extgrpc — ssh root@192.168.105.19 | ...root@master: ~/go/src/k8s.io/autoscaler/cluster-autoscaler — ssh root@192.168.105.19

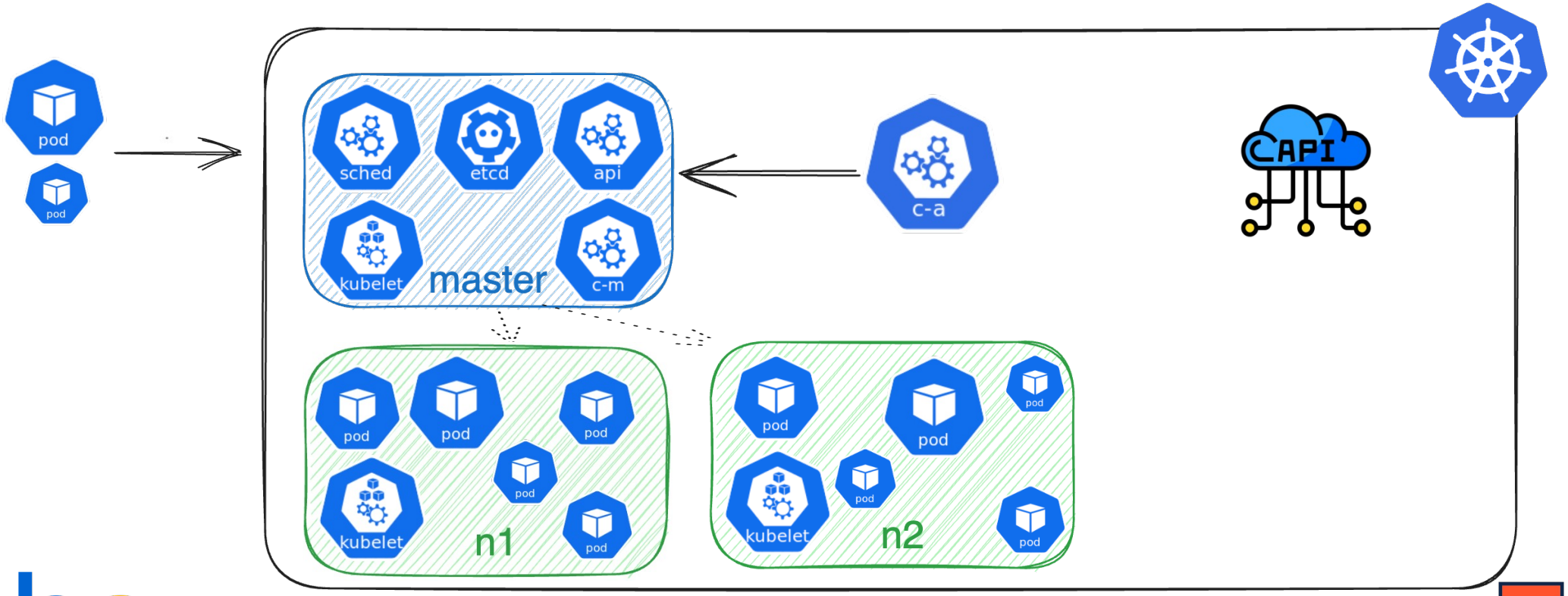
```
root@master:~/demo#
root@master:~/demo#
```

ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Cluster Autoscaler + InPlace Pod Resize

- We can leverage the ability to non-disruptively shrink underutilized pods!
- Add utilization check to the Cluster Autoscaler workflow
- Allocate new node(s) only if current pods cannot be safely resized down



Cluster Autoscaler + InPlace Pod Resize


```
86+ type Smusher interface {
87+     Smush() bool
88+ }
89+
90+ type InPlacePodSmusher struct {
91+     kubeClient kube_client.Interface
92+ }
93+
94+ func NewInPlacePodSmusher(kc kube_client.Interface) Smusher {
95+     return &InPlacePodSmusher{kubeClient: kc}
96+ }
97+
98+ func (ips *InPlacePodSmusher) Smush() bool {
99+     podsToResize := ips.getOverprovisionedPods()
100+     for pod, patch := range podsToResize {
101+         klog.Infof("DBG: Smushing pod '%s' with patch '%s'\n", pod, patch)
102+         ips.kubeClient.CoreV1().Pods("default").Patch(ctx.TODO(), pod,
103+             types.StrategicMergePatchType, []byte(patch), metav1.PatchOptions{})
104+         return true
105+     }
106+     return false
107+ }
108+
109+ func (ips *InPlacePodSmusher) getOverprovisionedPods() map[string]string {
110+     //TODO: return underutilized (from metrics-server/prometheus) pods to patch (resize down) where the resize
111+     //      recommendations are carefully crafted via supercalculus to ensure they also work in parallel universes
112+     //DEMO-HACK: return 'default/pod1' recommending resize down CPU to 200m
113+     return map[string]string{"pod1": `{"spec":{"containers":[{"name":"ctr", "resources":{"requests":{"cpu":"200m"}}}]}}`}
114+ }
```

```
601     } else {
602+         klog.Info("DBG: Checking if existing pods can be resized down before scaling up cluster...\n")
603+         if a.smusher.Smush() {
604+             klog.Info("DBG: Pods were resized down, waiting one iteration\n")
605+             scaleUpStatus.Result = status.ScaleUpInCooldown
606+             return nil
607+         }
608         scaleUpStart := preScaleUp()
609         scaleUpStatus, typedErr = a.scaleUpOrchestrator.ScaleUp(unschedulablePodsToHelp, readyNodes, daemonsets, nodeInfosForGroups)
610         if exit, err := postScaleUp(scaleUpStart); exit {
611             return err
612         }
```

Kubeadm CA + InPlace Pod Resize

KubeCon CloudNativeCon North America 2023

```
NAME      STATUS   ROLES           AGE    VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
master    Ready    control-plane   6d13h  v1.28.2   192.168.105.19   <none>        Ubuntu 22.04.3 LTS   5.15.0-87-generic   containerd://1.6.24
node1     Ready    <none>          6d13h  v1.28.2   192.168.105.16   <none>        Ubuntu 22.04.3 LTS   5.15.0-86-generic   containerd://1.6.24
```

vikulkarni — root@master: ~ — ssh root@192.168.105.19 — 132×8

```
Every 1.0s: kubectl get pods -owide                                    master: Mon Oct 30 05:35:31 2023

NAME                                  READY   STATUS    RESTARTS   AGE   IP              NODE     NOMINATED NODE   READINESS GATES
ca-ext-grpc-kubeadm-cloud-provider    1/1     Running   0          21m   192.168.105.19  master   <none>           <none>
pod1                                  1/1     Running   0          78m   10.0.1.214      node1    <none>           <none>
pod2                                  1/1     Running   0          78m   10.0.1.204      node1    <none>           <none>
```

vikulkarni — root@master: ~/go/src/k8s.io/autoscaler/cluster-autoscaler — ssh root@192.168.105.19 — 156×5

```
root@master:~/go/src/k8s.io/autoscaler/cluster-autoscaler#
```

vikulkarni — root@master: ~/demo — ssh root@192.168.105.19 — 172×22

~ — root@master: ~/demo — ssh root@192.168.105.19 | ~ — root@master: ~/go/src/k8s.io/autoscaler/cluster-autoscaler — ssh root@192.168.105.19

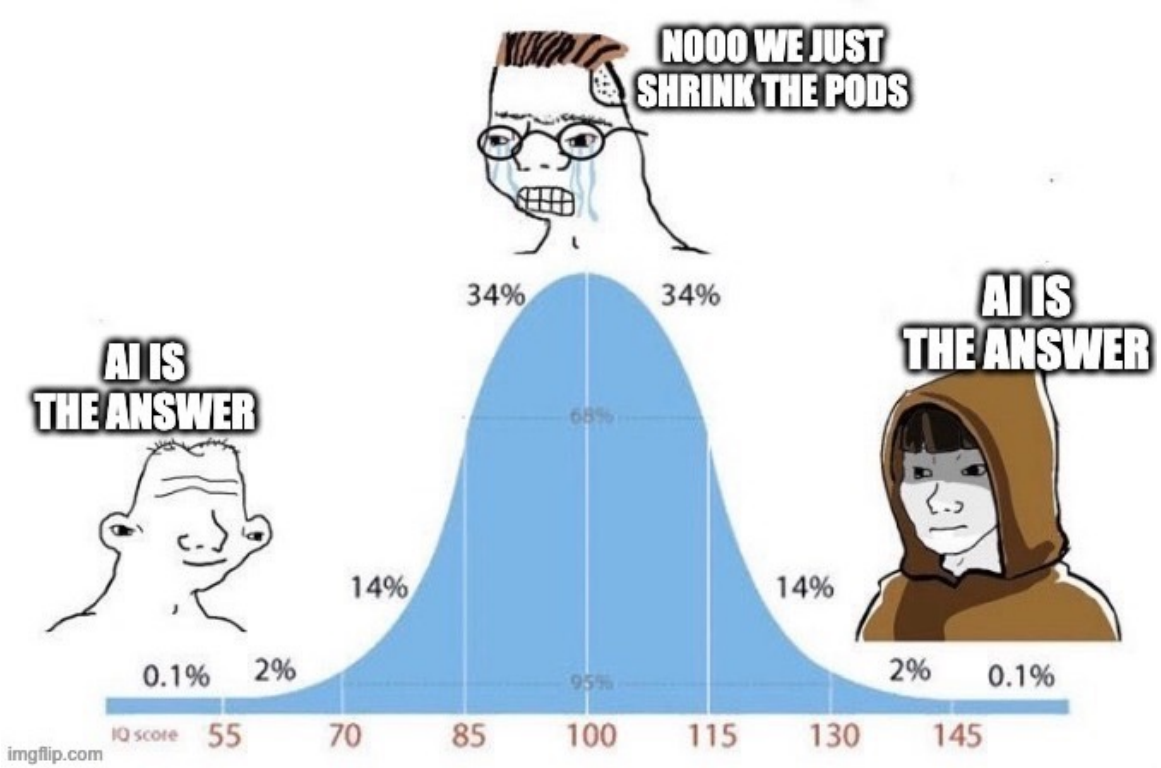
```
root@master:~/demo#
root@master:~/demo# cat onemorepod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: onemorepod
spec:
  containers:
  - name: ctr
    image: skiibum/testpod:qos
    command: ["tail", "-f", "/dev/null"]
    resources:
      requests:
        cpu: 1
root@master:~/demo#
root@master:~/demo#
```

ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

So, What Could Go Wrong?

- That critical pod you just sized down is about to be OOM-killed in a 3 am traffic spike
- CPU taken away from idle services is about to degrade weekend online shopping experience because a long running batch jobs are about to start
- We can do this all day…



Cloud Platforms: Natural Arena for RL

- Full of sequential decision-making processes on many systems management tasks
  - E.g., resource management, job scheduling, load balancing, congestion control, …
- Hard to model, currently rely on human-engineered heuristics
  - RL (reinforcement learning) enables using DNNs to express the complex dynamics with "raw" and noisy signals and decision-making policies
- Abundant data (e.g., monitoring, systems metrics, workload performance)
  - E.g., Prometheus for Kubernetes, GWP at Google, eBPF tools, …

amazon web services EC2

Google Cloud Platform

Microsoft Azure

VM Placement

Congestion Control

Cluster Scheduling

… and more!

ebay

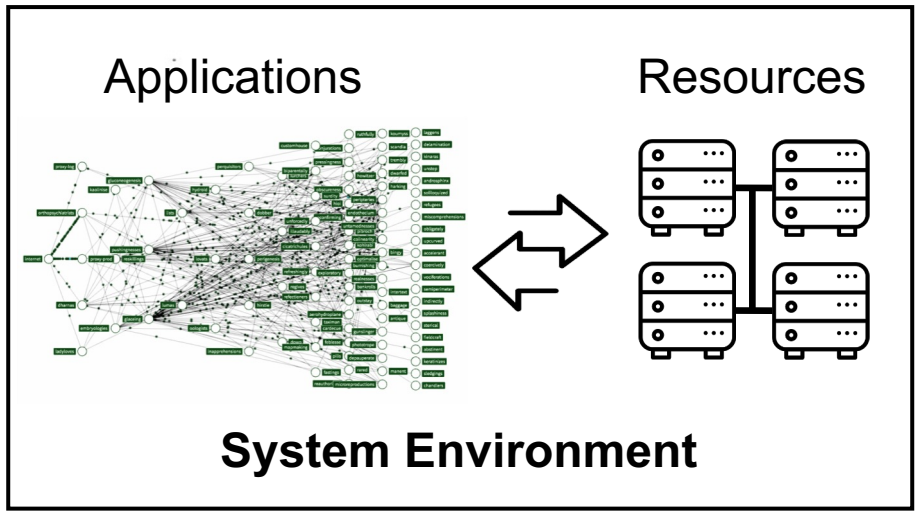
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

What Do People Do Today?

At a higher level…

Human-driven performance engineering
- Assume a simple system model
- Produce clever heuristics or parameters
- Test & tune the parameters in practice
- Redo the above steps ↺

RL-based online learning solutions
- Initialized with a random policy
- Get states, make actions, obtain rewards
- Optimize the policy based on the rewards
- Loop continues until convergence ↺



Controls

Profiling Data

Human Engineering

Heuristics / Rules

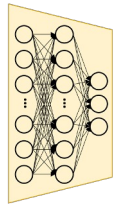
Measurements

Decisions / Predictions

Reinforcement Learning (RL)

Action

States, Rewards



DNN (Policy)

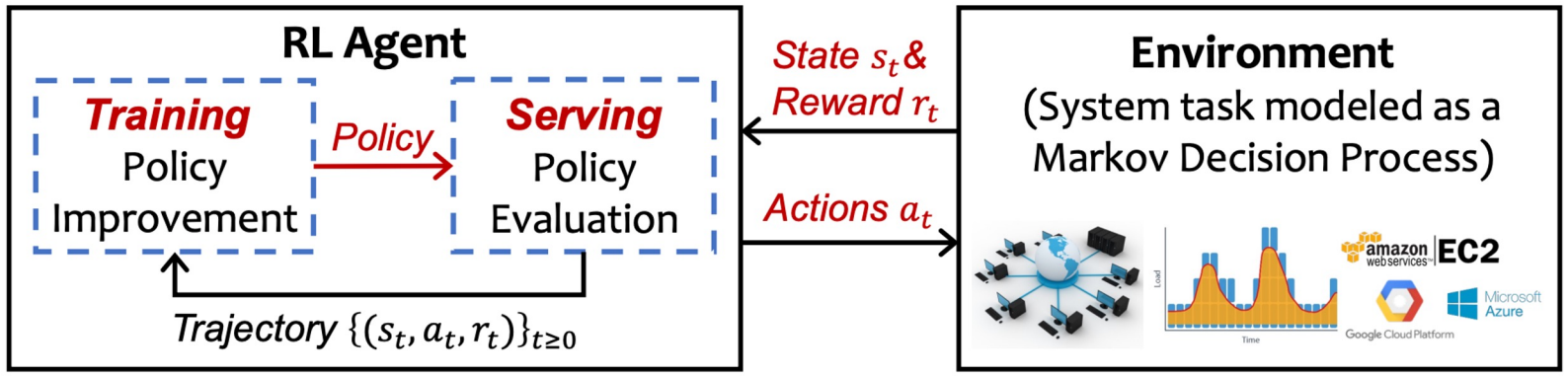
- **Systematic framework** for retraining to reduce human-guided profiling/tuning
- Reducing costly optimization (parameter search) to **O(1)** time

ebay

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

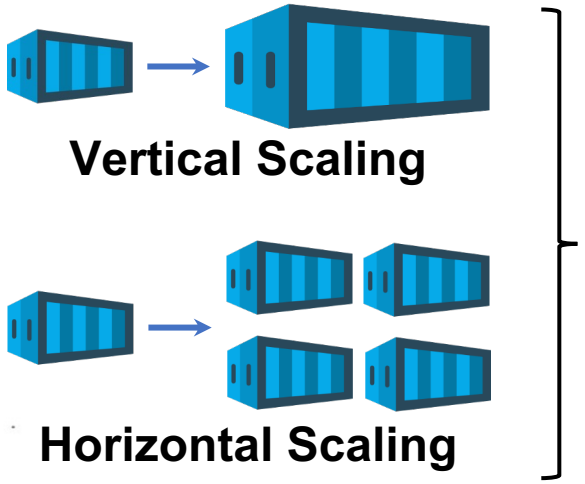
RL Primer

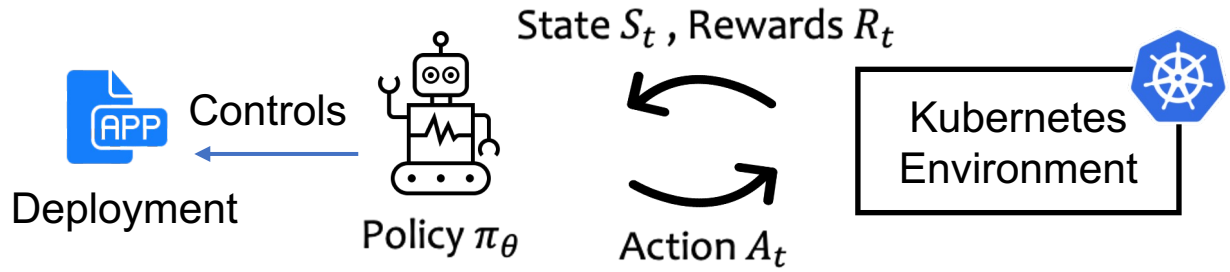
- An approach which falls between supervised and unsupervised learning
  - No labeled data but yet need a "**reward**"
- An **agent** interacts with an **environment** step by step, at each time step *t*:
  - Gets the *state* s_t, makes a decision or *action* a_t, and then receives a *reward* r_t
- Reward serves as a **feedback** directing the policy training



Goal: Maximize the expected cumulative reward in a *T*-step trajectory, i.e., $\mathbb{E}[\sum_{t=0}^{T} \gamma^t \cdot r_t]$

Autoscaling as an RL Formulation





RL-based Multidimensional Pod Autoscaling
(Modeled as a Markov Decision Process)

- With RL:
  - Get rid of human-driven application profiling and tuning of parameters in heuristics-based approach
  - E.g., in threshold-based autoscaling, the optimal threshold for CPU utilization without violating performance SLOs varies across applications and infrastructures
  - Automate policy learning with a systematic and dynamic feedback-control loop

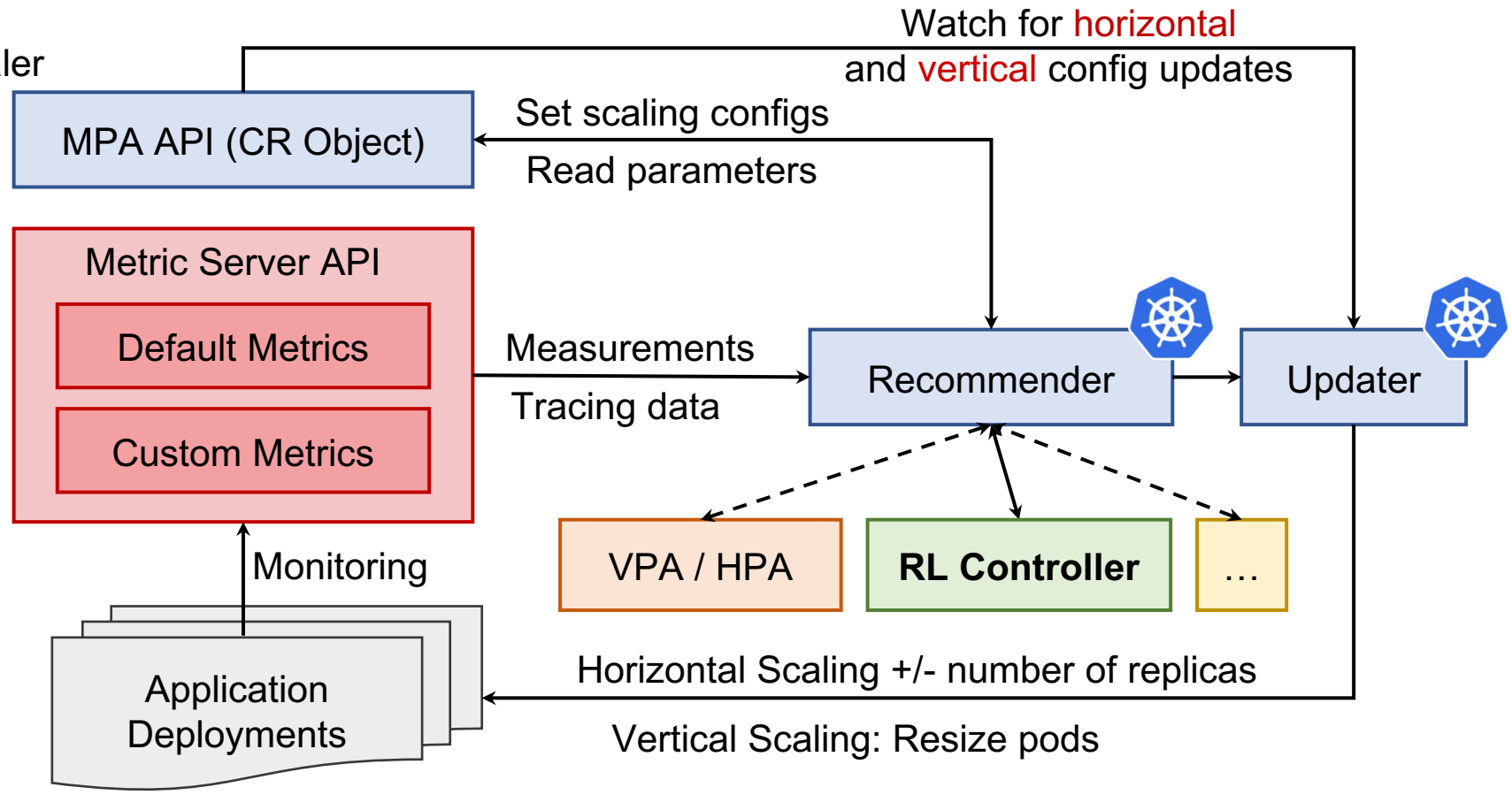
MPA Design Overview

*MPA: Multi-dimensional Pod Autoscaler

Kubernetes Operators

Principle: Decoupling scaling recommendation from scaling action execution

- Allow custom recommenders such as RL agents to plug-and-play



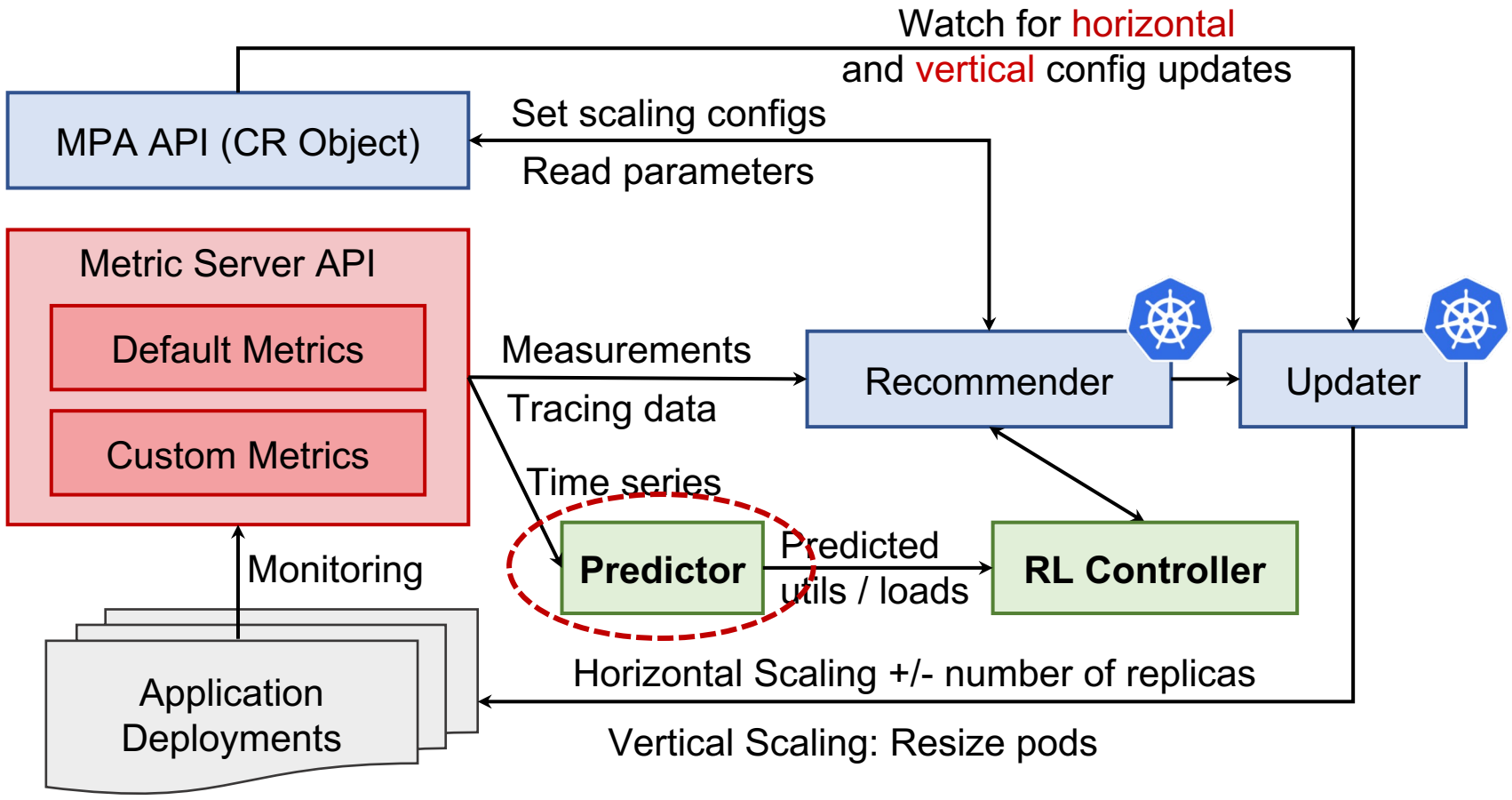
RL Agent from FIRM (OSDI 2020)

***Reactive* autoscaling**

- ***FIRM (OSDI 2020)***
- ***SIMPPO (SoCC 2022)***

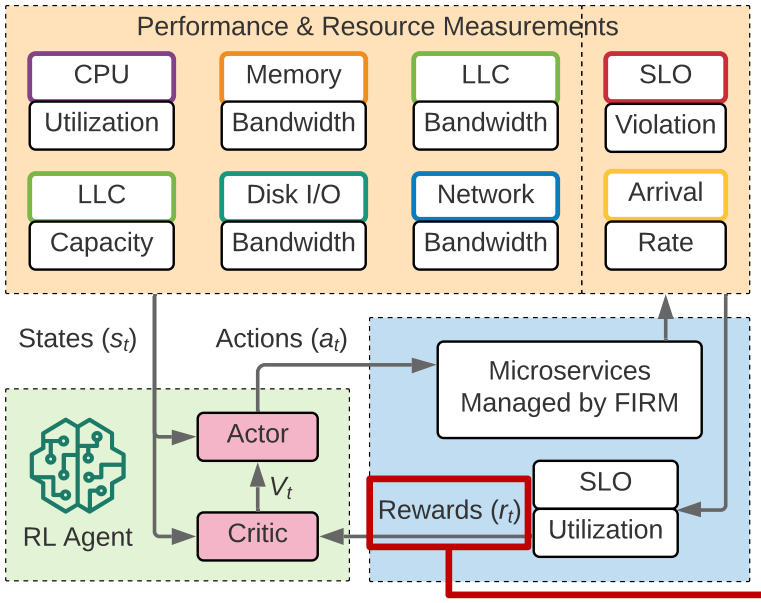
***Proactive* autoscaling (with load prediction)**

- ***DeepScaling (SoCC 2022)***



RL Training

- An RL-based resource estimation agent that learns to make autoscaling decisions *directly* from experience
- Optimizes objectives end-to-end:
  - Maximize resource utilization efficiency
  - Minimize SLO violation (regarding latency/throughput)



Reward Function:

$$r(t) = \alpha \cdot SM_t \cdot |\mathcal{R}| + (1-\alpha) \cdot \sum_i^{|\mathcal{R}|} RU_t^i / RL_t^i$$

Mitigate SLO Violation Fast

Avoid Over-provisioning

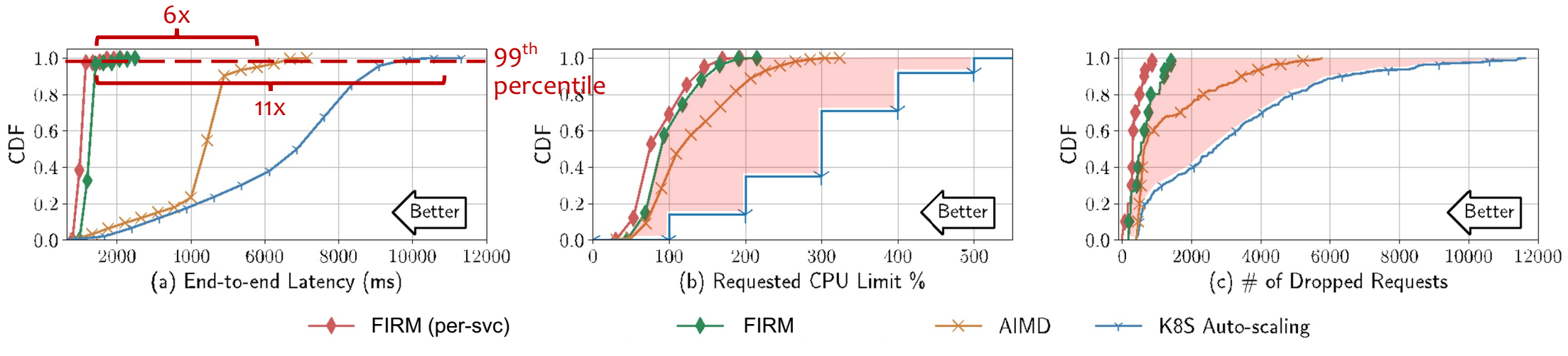
$$SM_t = \frac{Latency_{SLO}}{Latency_t}$$
(SLO maintenance)

Resource utilization of i at time t

Resource limit of i at time t

Results

- Reduces the SLO violation mitigation time by up to 9× compared with Kubernetes autoscalers
  - Reduces the average tail latencies by up to 6-11×
  - Reduces the overall average requested CPU/memory limit by 29-62%
  - Reduces the number of dropped/timed out requests by up to 8x



Summary

- Cloud Computing has significant wasted costs and environmental impact
- In-Place Pod Resize opens the doors to sustainable, multidimensional pod autoscaling
- Cluster Autoscaler based on heuristics is neither tenable nor cost-effective
- Autoscaling with Reinforcement Learning looks very promising
  - → Provides a systematic way of tuning for the best policy
  - → Gets rid of human-driven profiling or inaccurate system models
  - → Predictive autoscaling with time series forecaster
- Drive In-Place Pod Resize to GA and realize significant cost savings with holistic autoscaling

References

1. https://mit-serc.pubpub.org/pub/the-cloud-is-material/release/1
2. https://thereader.mitpress.mit.edu/the-staggering-ecological-impacts-of-computation-and-the-cloud/
3. https://www.techtarget.com/sustainability/feature/Cloud-computings-real-world-environmental-impact
4. https://jaychapel.medium.com/overprovisioning-always-on-resources-lead-to-26-6-billion-in-public-cloud-waste-expected-in-2021-da888ea68f74
5. https://www.stormforge.io/survey-report/stormforge-2022-kubernetes-cloud-waste-survey/
6. https://piotrminkowski.com/2023/08/22/resize-cpu-limit-to-speed-up-java-startup-on-kubernetes/
7. Demo code: https://github.com/kubernetes/autoscaler/commit/9dff7d95b6c6fdf085a3868115f4ddba97b2a977
8. Haoran Qiu, Subho S. Banerjee, Saurabh Jha, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer (2020). ***FIRM: An Intelligent Fine-Grained Resource Management Frameworkfor SLO-Oriented Microservices.*** In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2020).
9. Haoran Qiu, Weichao Mao, Archit Patke, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Başar, Ravishankar K. Iyer (2022). ***SIMPPO: A Scalable and Incremental Online Learning Framework for Serverless Resource Management***. In Proceedings of the 13th ACM Symposium on Cloud Computing (SoCC 2022).
10. Haoran Qiu, Weichao Mao, Chen Wang, Hubertus Franke, Alaa Youssef, Zbigniew T. Kalbarczyk, Tamer Başar, Ravishankar K. Iyer (2023). ***AWARE: Automate Workload Autoscaling with Reinforcement Learning in Production Cloud Systems***. In Proceedings of the 2023 USENIX Annual Technical Conference (ATC 2023).
11. Ziliang Wang, Shiyi Zhu, Jianguo Li, Wei Jiang, K. K. Ramakrishnan, Yangfei Zheng, Meng Yan, Xiaohong Zhang, and Alex X. Liu (2022). ***DeepScaling: Microservices Autoscaling for Stable CPU Utilization in Large Scale Cloud Systems***. In Proceedings of the 13th Symposium on Cloud Computing (SoCC 2022).

Please scan the QR Code above to leave feedback on this session