Georgia Institute of Technology

Fine-grained, Lightweight Resource Metric Collection in Mobile Edge Computing

Ke-Jou Hsu

Advisor: Ada Gavrilovska

What is Mobile Edge Computing (MEC)?

- More computing resources than client devices
- Reduced communication cost than cloud
- Flexible service deployment across edge-cloud

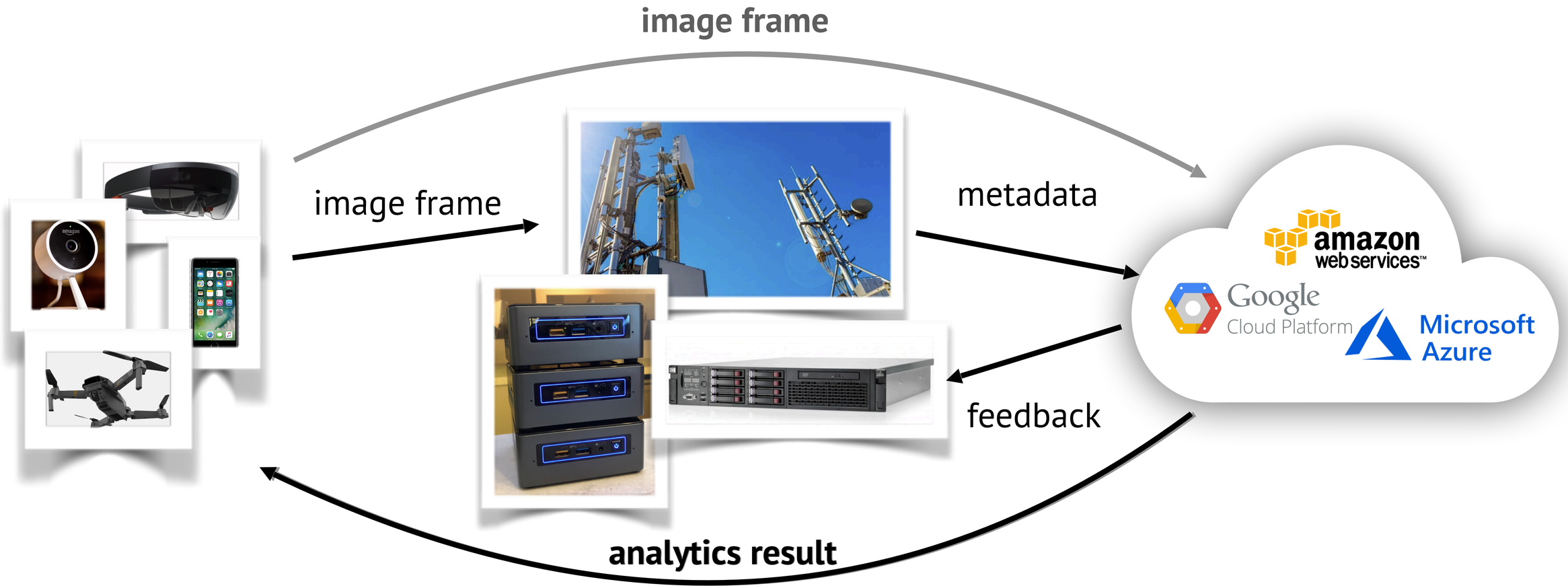
Difference between MEC and Cloud

Both of them are multi-tenancy environments, and supporting friendly deployment interfaces

	Cloud	Mobile Edge
Network distance to clients	Far	Close
Resource Heterogenity	Low	High
Resource capacity	High	Low
Strict latency request	Low	High

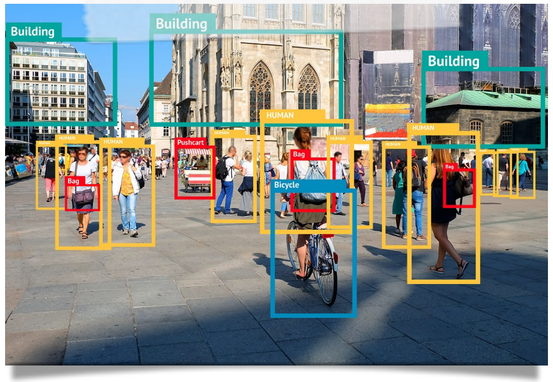
MEC provides proximity benefit to clients

For example, ML-based video analytics service

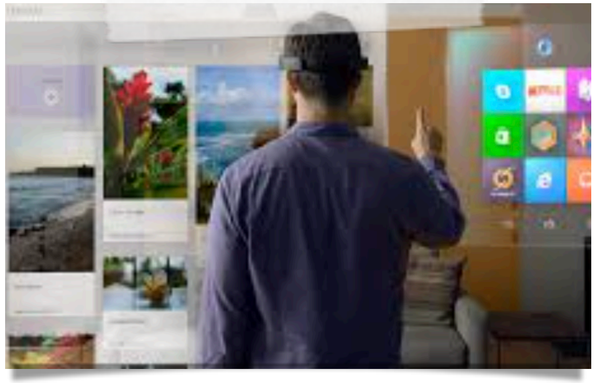


Emerging MEC-targeted applications

IoT Video Analytics (ML)

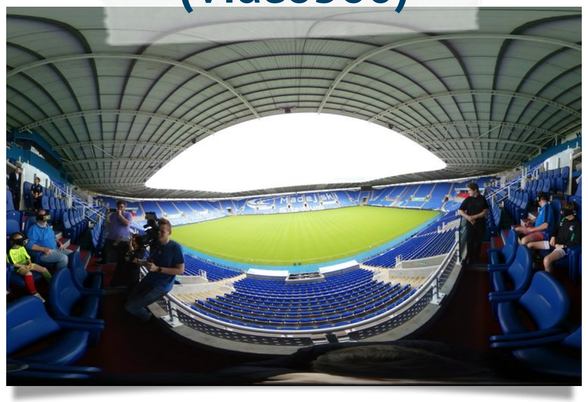


Augmented/Virtual/Extended Reality (AR/VR/XR)



Content Delivery Network (CDN)

360° Video Streaming (Video360)



Industrial and Autonomous Systems (IA)

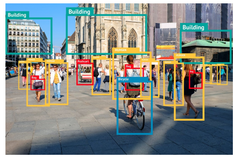
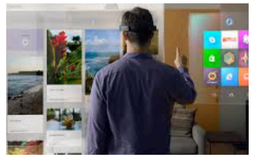
How to deploy them at edge?

Happily ever after?

Put MEC workloads at edge and get benefit right away?

Resource management is an important/popular topic on system




Application data plane

System control plane

MEC Service Provider

MEC System Provider

Resource management modules

Load Balancer

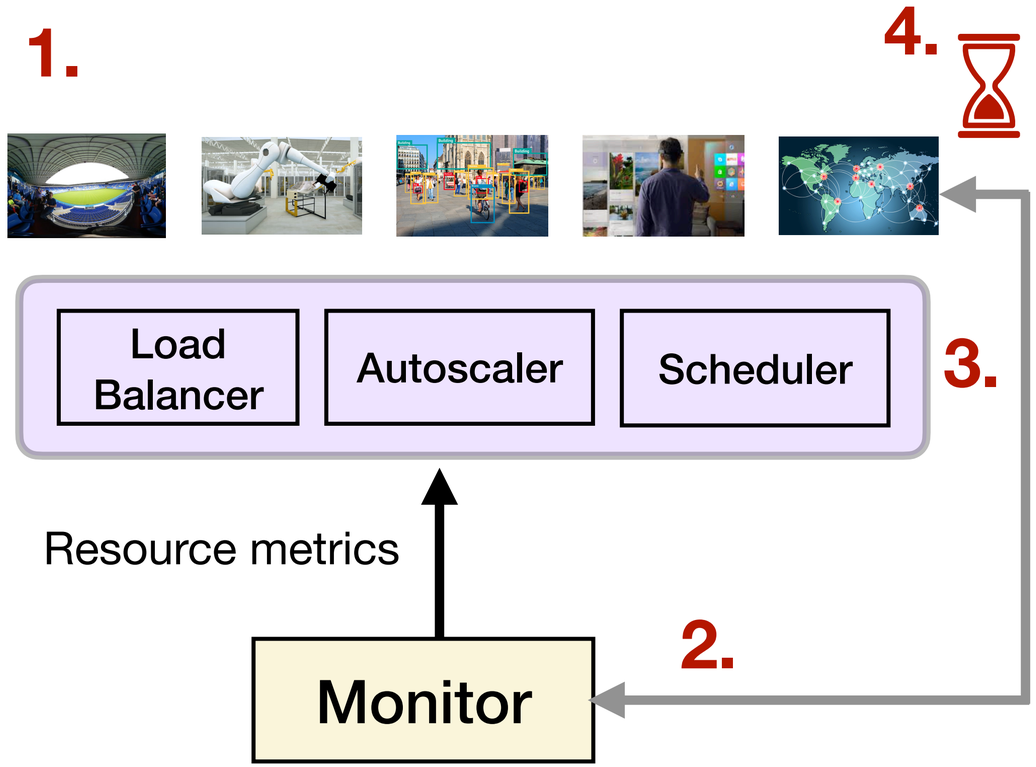
Autoscaler

Scheduler

Resource metrics

Monitor

Let’s evaluate the process at edge



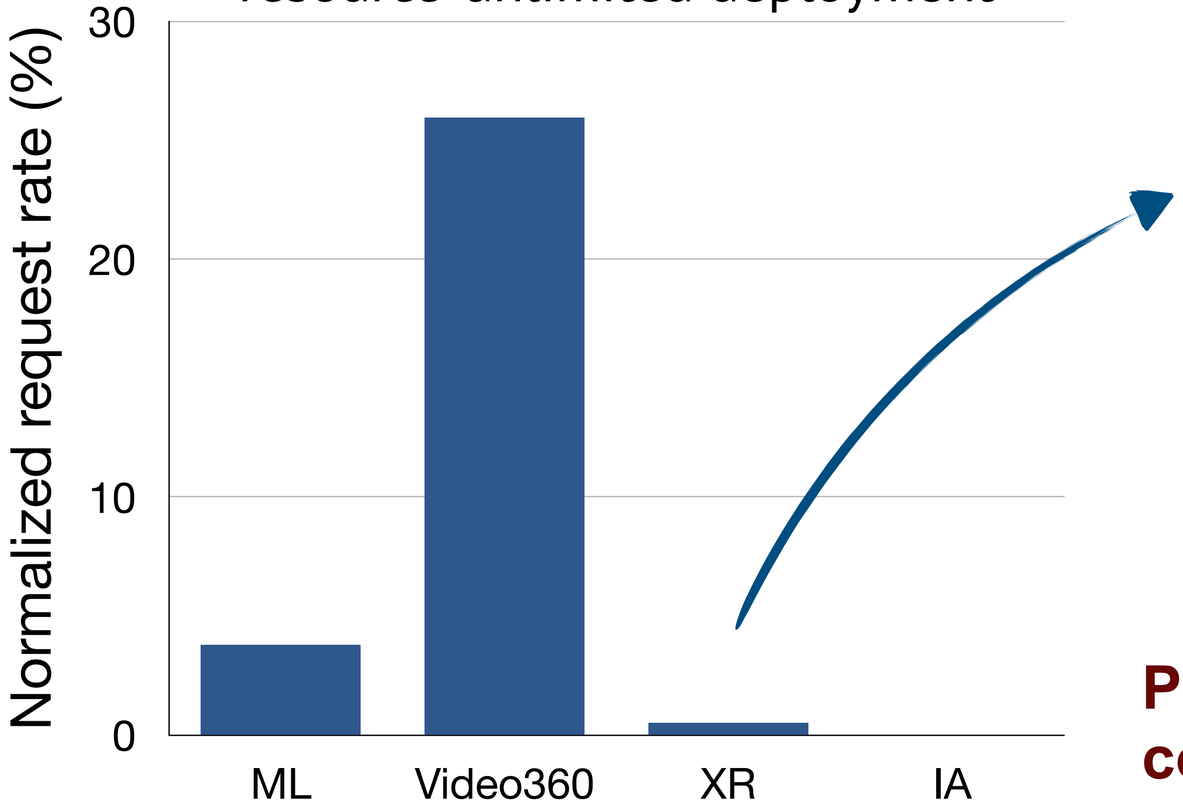
1. Deploy the workload with **full capacity** of the system
2. Measure the performance + monitor collects the metrics of resource utilization
3. **Redeploy** the workload with the metrics from the monitor
4. Verify/compare the performance from (2)

If the performance is similar ->
the metrics from monitor is reliable

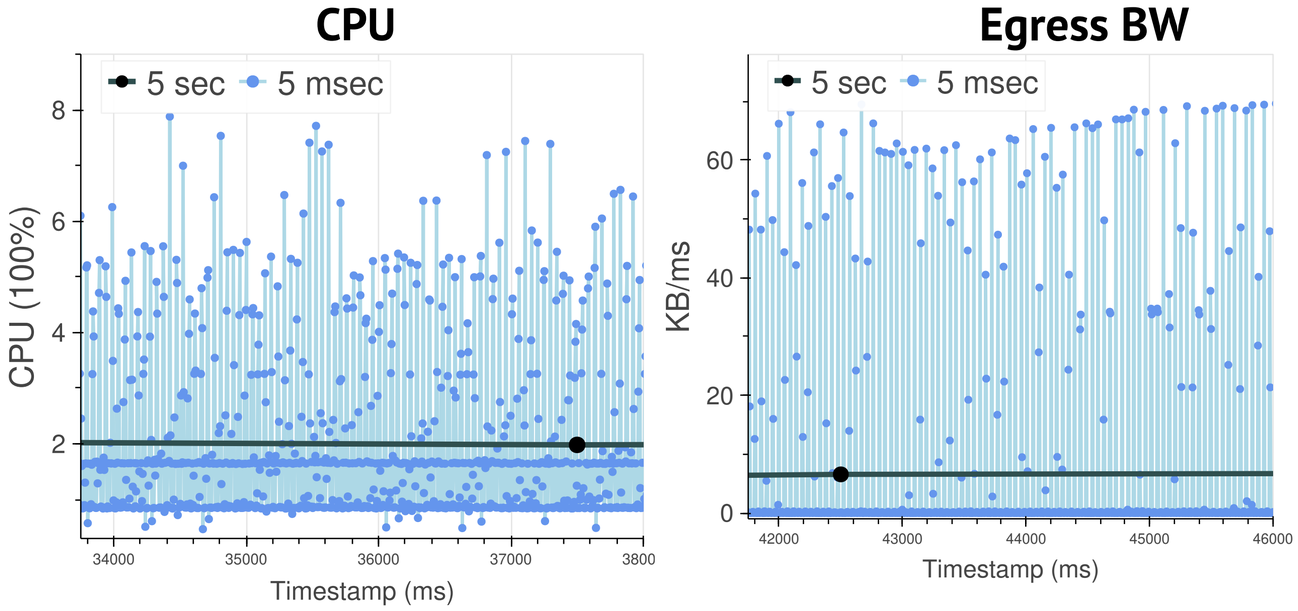
Cloud-native monitor cannot find suitable numbers for MEC workload

Using Prometheus's metrics for resource allocation

Normalized to the performance of resource-unlimited deployment



Raw data comparison between different query granularity



Problem 1: Losing information of spiky usage in **coarser granularity** metric collection.

Edge is more heterogenous

Customer owned
Private, dedicated
Distributed

Service provider owned
Sharable
Centralized, secured

Wavelength

{ t3.{medium, xlarge}
r5.2xlarge
g4dn.2xlarge

Last mile network

Telco operated network

< 5 ms

5~20 ms

Even if workload or workload-combination doesn't change, MEC environment could change

= one deployment decision cannot fit everywhere

Problem 2: A single profile for every edge is not feasible
— wasting resources or reducing performance

Require a new monitor designed for MEC

Problem 1: Losing information of spiky usage in **coarser granularity** metric collection

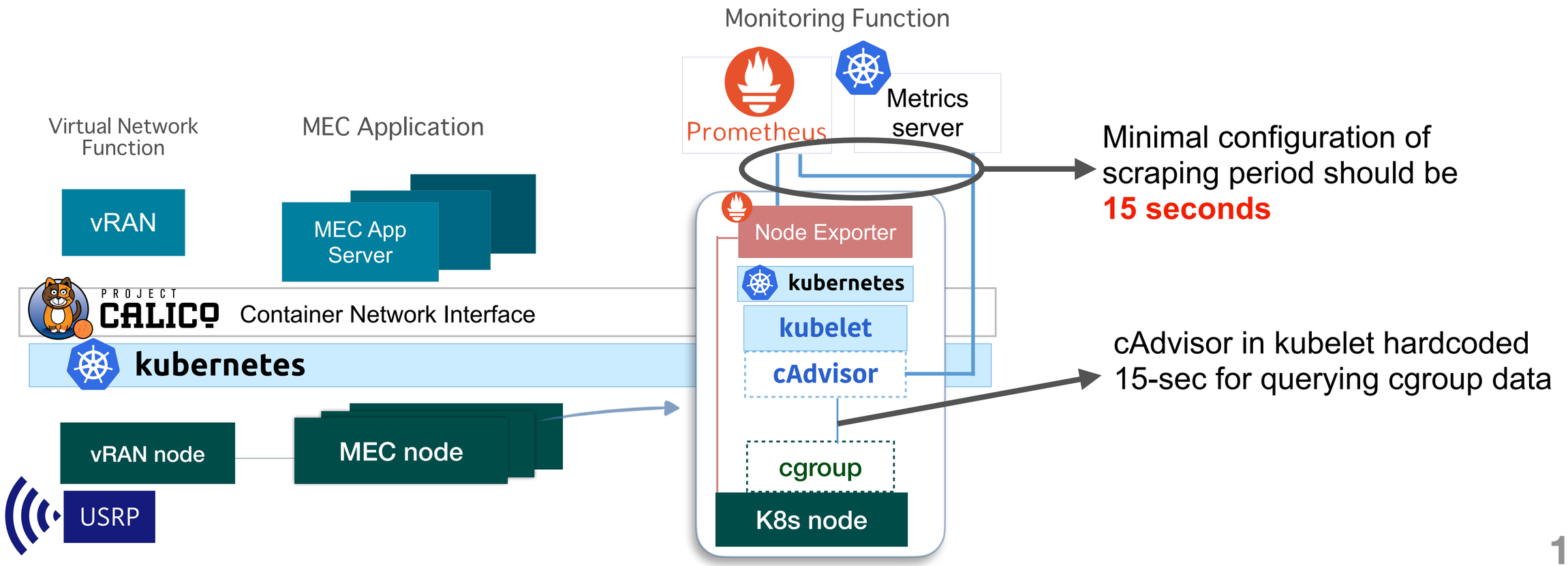
Problem 2: A single profile for every edge is not feasible

Target: fine-grained metric collection + need to profile frequently

Modifying the existed cloud-native monitoring system to solve the problem?

Target: fine-grained metric collection + need to profile frequently

Changing the scrape interval to a smaller one, then query faster? e.g. Prometheus or K8s's Metrics Server



Modifying the existed cloud-native monitoring system to solve the problem?

Target: fine-grained metric collection + need to profile frequently

1. Configure system's scrape interval to subsecond? ✗ Design limitation or depending on other system
2. Update the source code and recompile the system? ✗ No guarantee system correctness
3. Revisit existing monitoring system design? ✗ Incurring high resource consumption

- Every node needs a monitoring agent
- Covering nearly all types of metrics
  (e.g. up to 100~ in Prometheus for a single container)
- Covering all workloads
- **Monitoring system is "always-on"**

Colibri: the resource profiler for MEC

Target: fine-grained metric collection + need to profile frequently

millisecond-level query

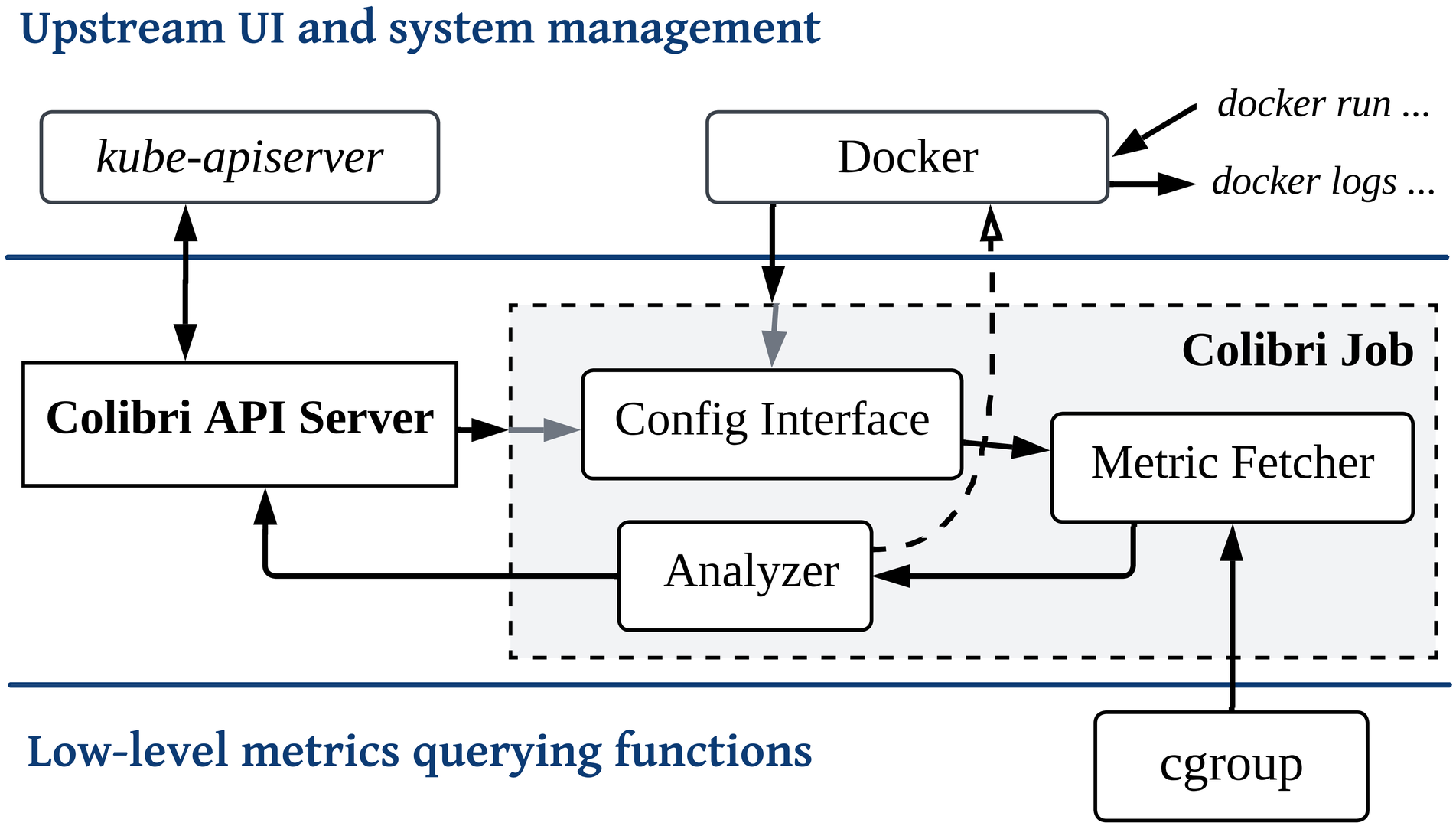
easy to use

lightweight, low-cost

1. Easily dispatch
2. Transparent, straightforward interface

3. Minimal system/package dependency
4. Running on demand
5. Customizing each run for workload

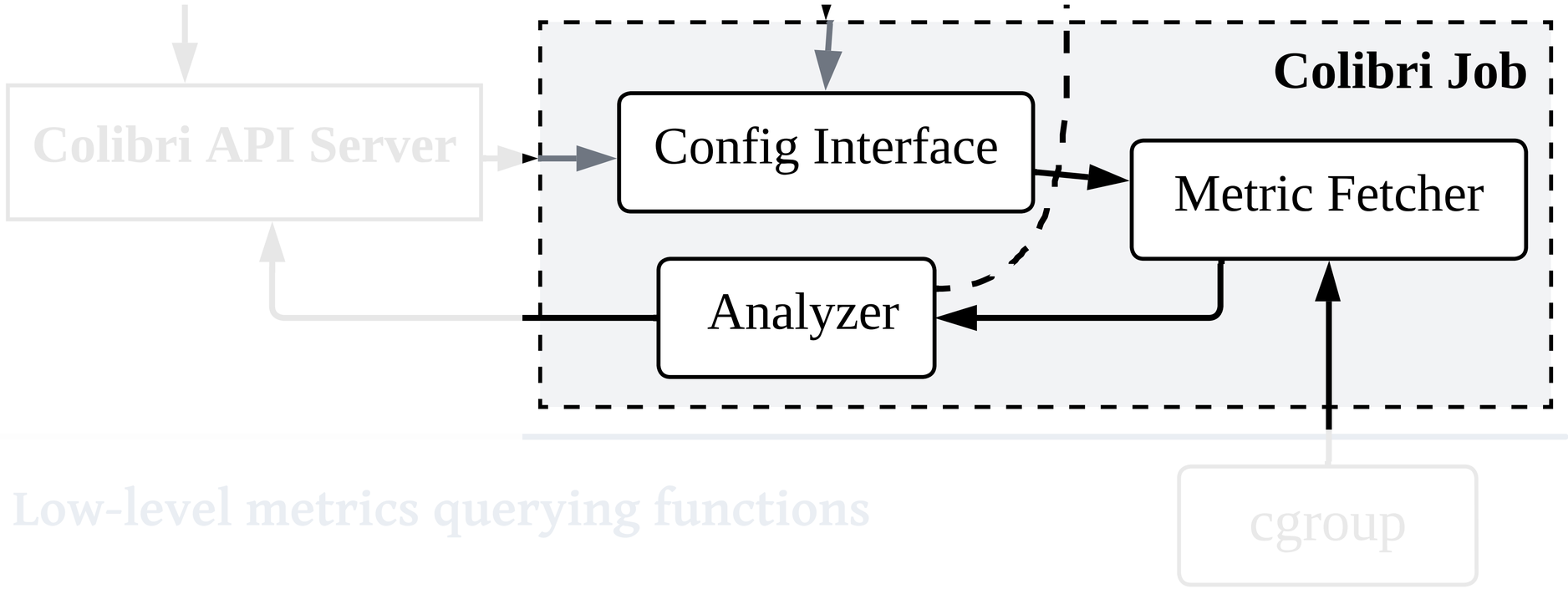
System structure of Colibri



System structure of Colibri

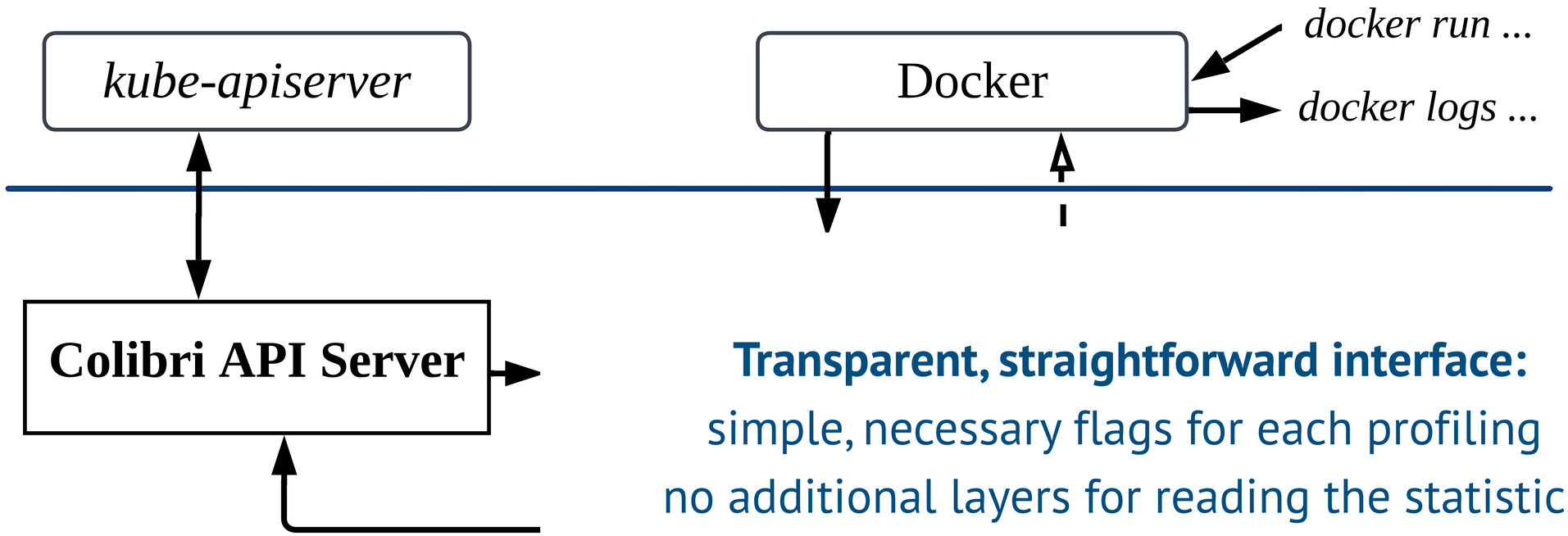
Easily dispatch: Colibri Job is containerized, and one for each workload

Running on demand: Each Job starts by request, and terminated automatically



System structure of Colibri

Upstream UI and system management



Transparent, straightforward interface:

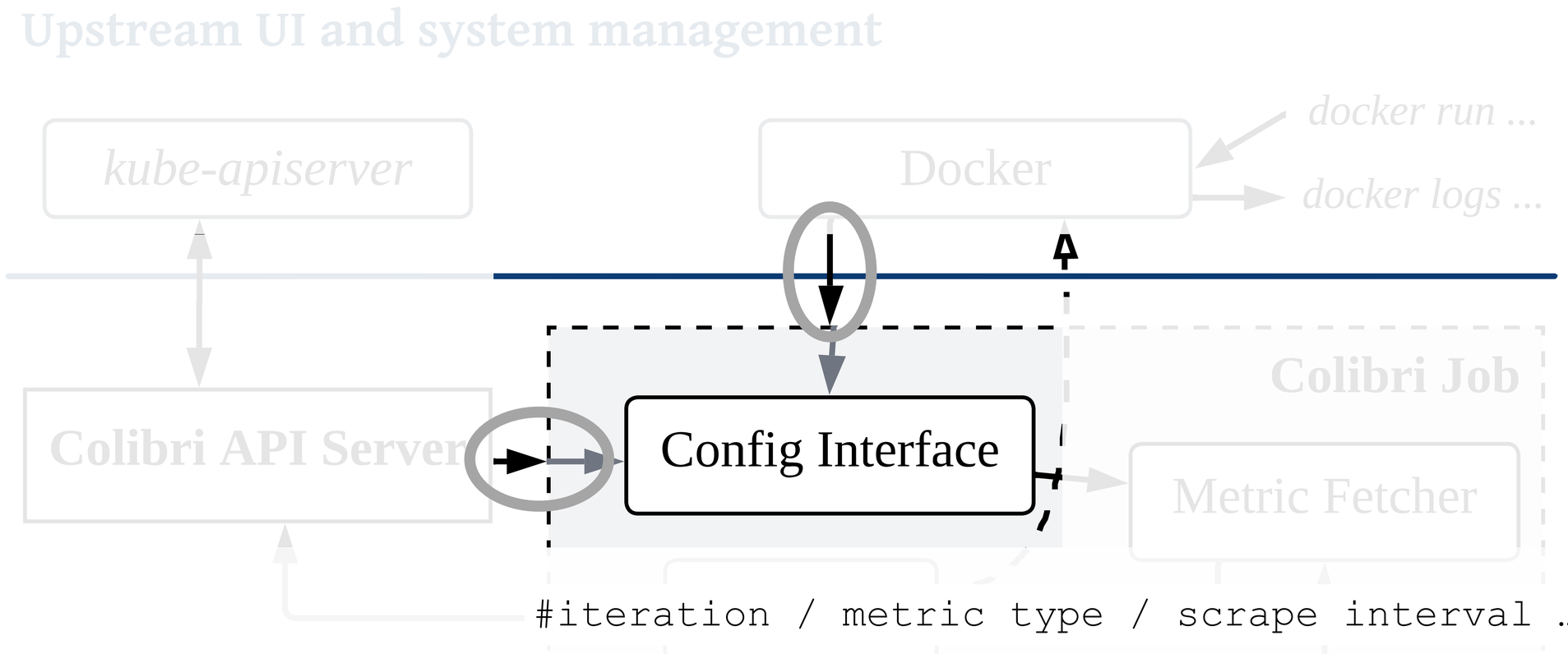
simple, necessary flags for each profiling

no additional layers for reading the statistics

Low-level metrics querying functions

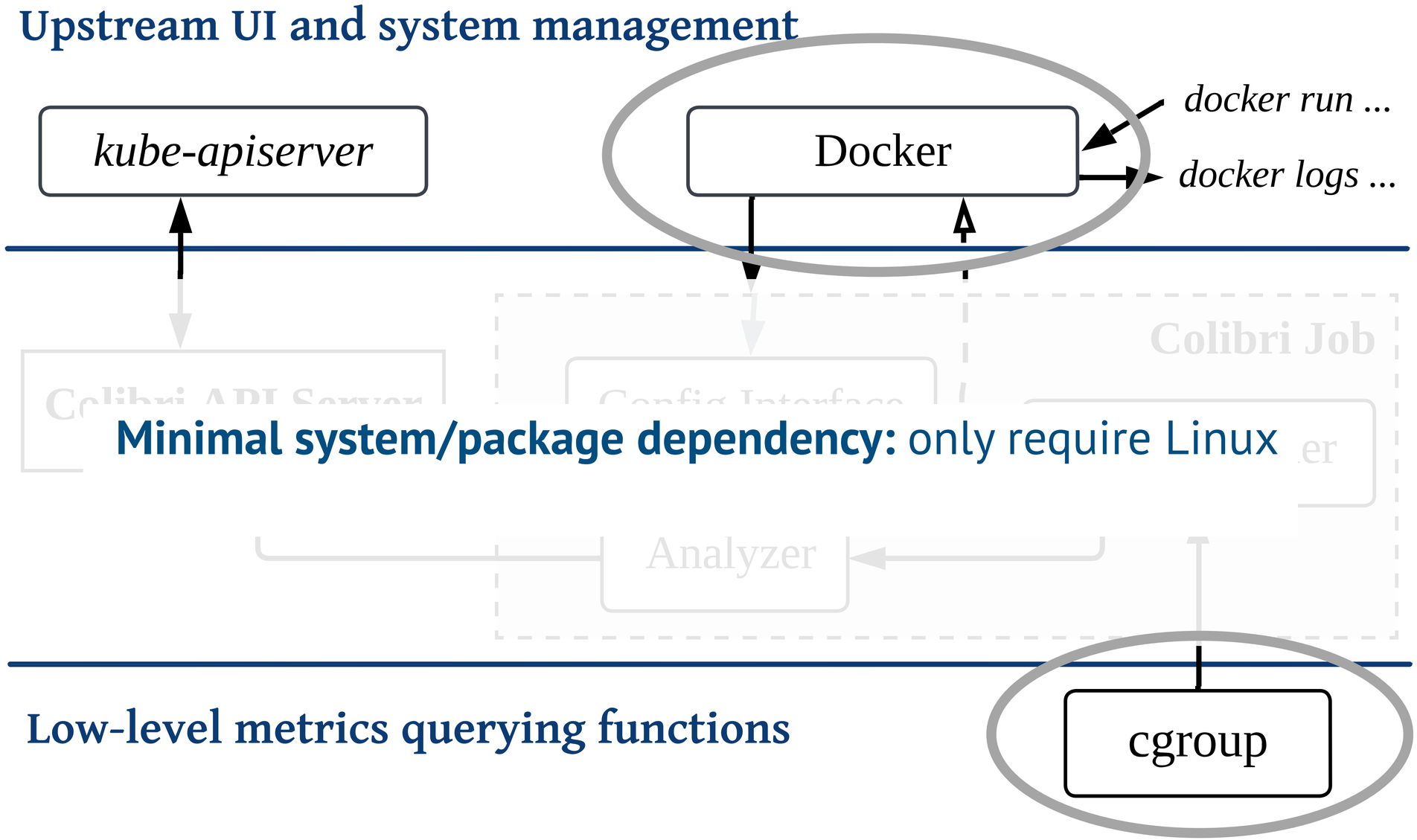
cgroup

System structure of Colibri

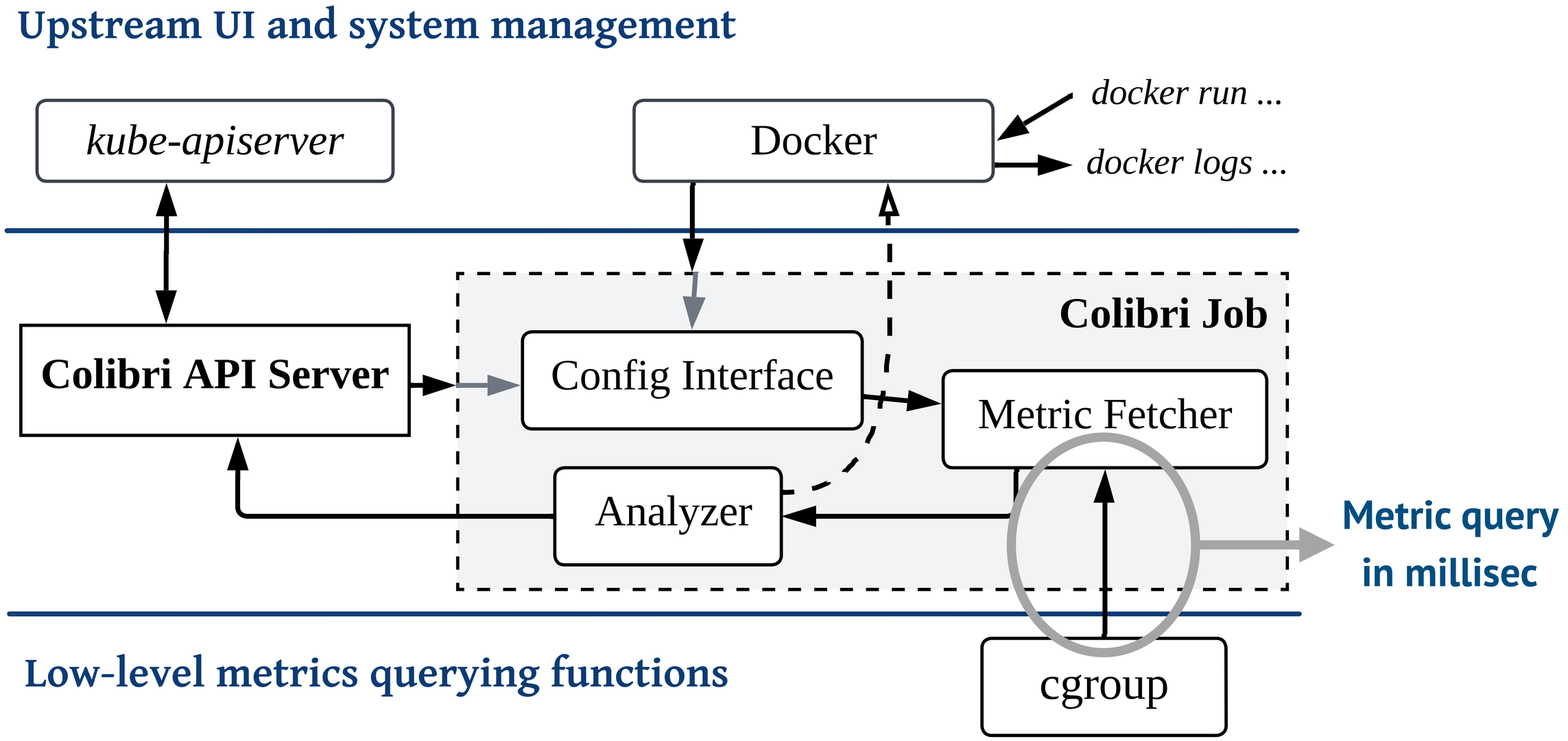


Customizing each run: further reducing query overhead

System structure of Colibri

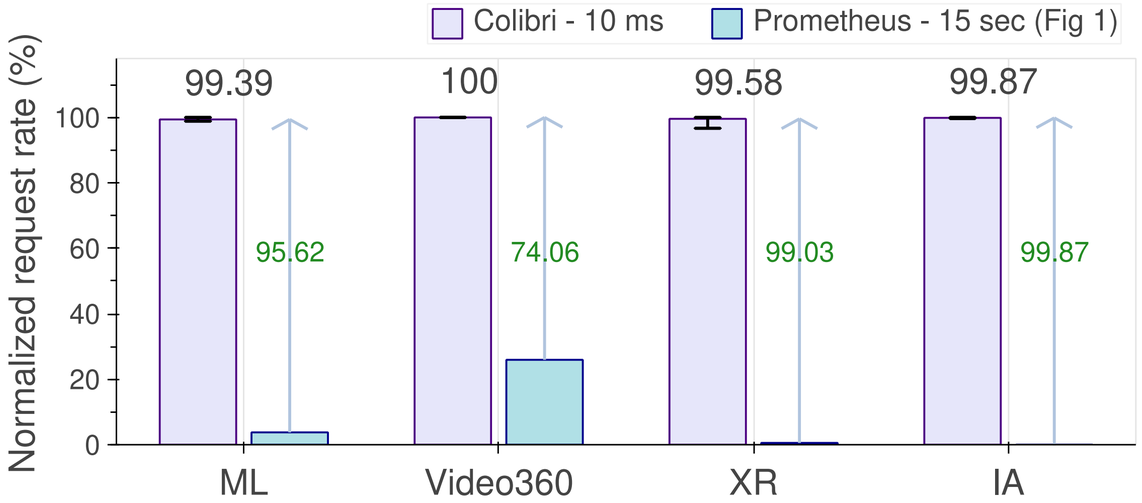


System structure of Colibri

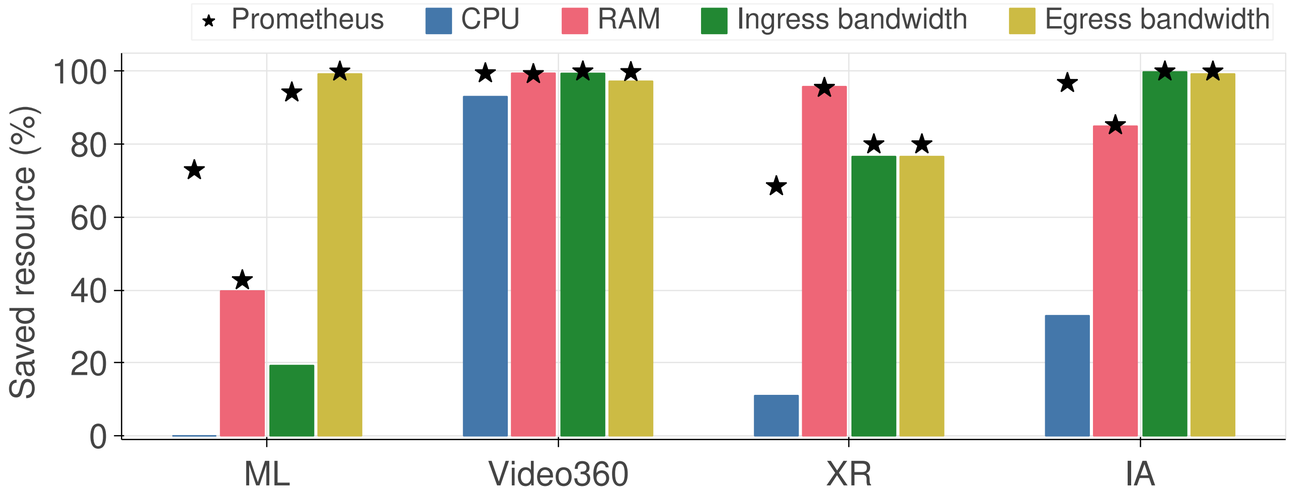


Accurate resource allocation with Colibri

Using Colibri's query data to deploy workloads



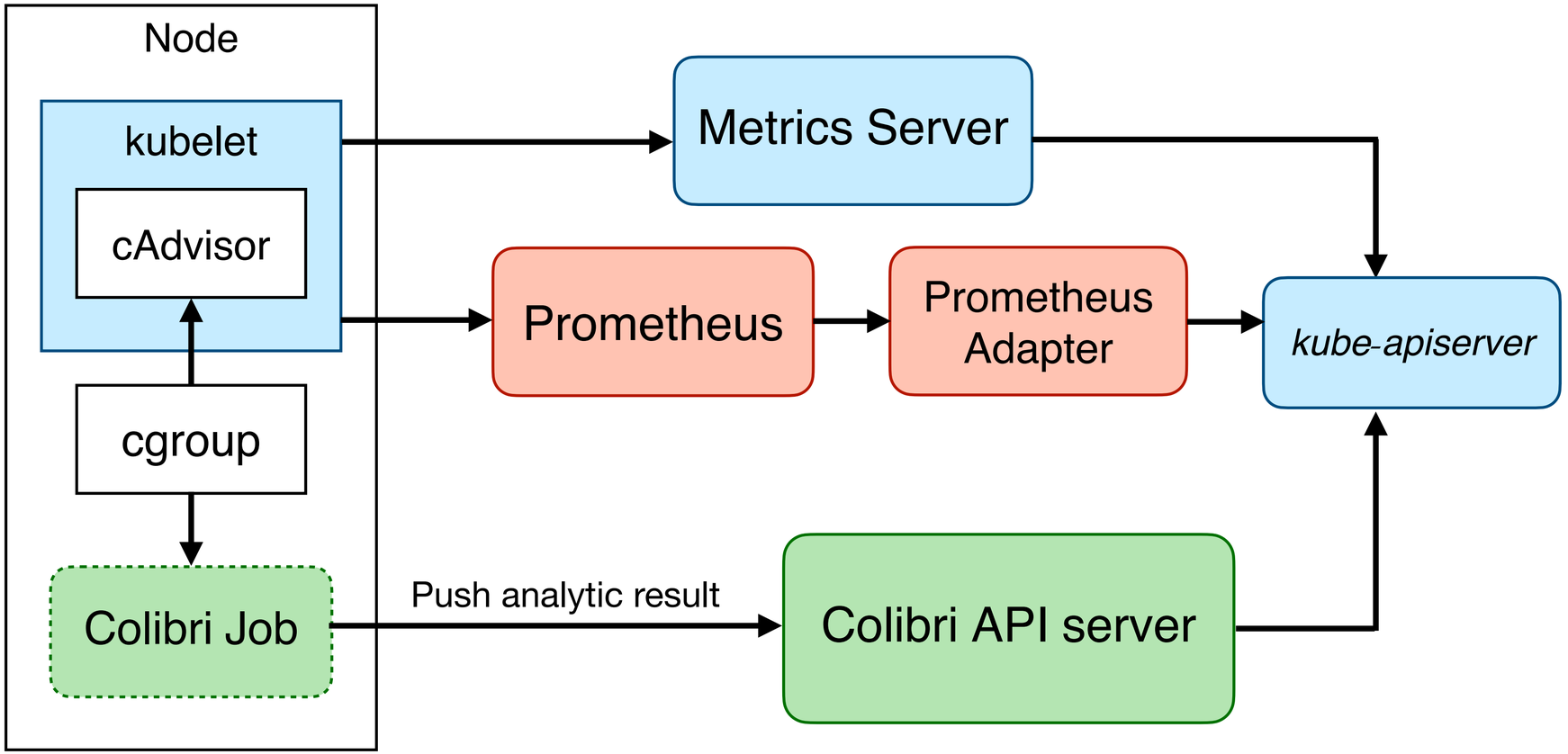
Deployment profile comparison:
Colibri vs. Prometheus



- Outperform second-level query across different workloads
- Not all metrics need high resolution query
- Some slight difference of requested resource can cause obvious performance gap

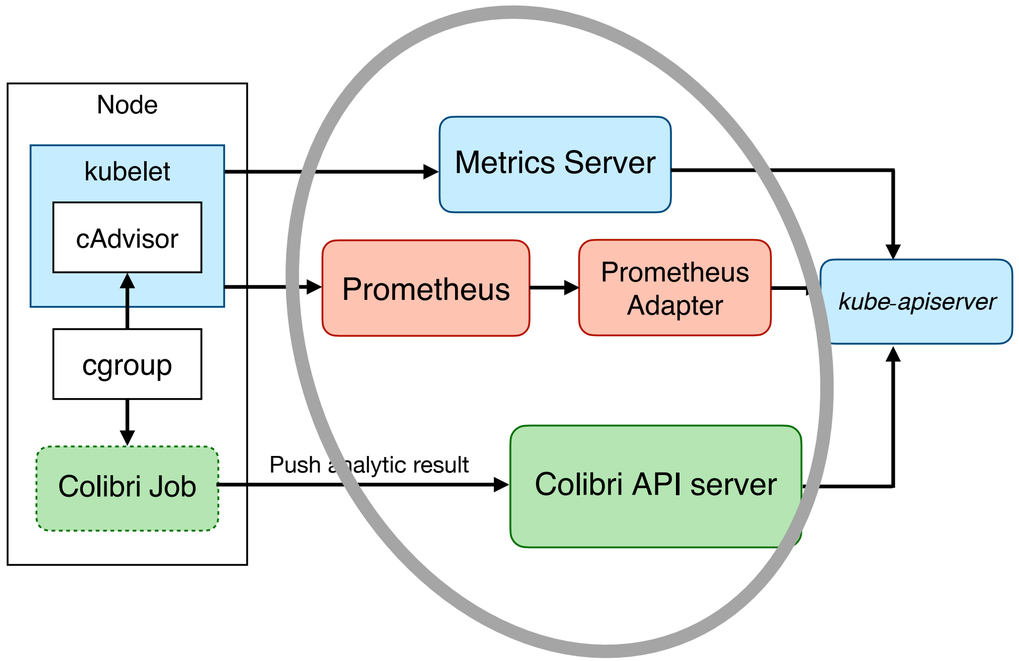
Profiling cost comparison

Overview of metric querying pipeline:
comparing with Prometheus and Kubernetes Metrics Server



Colibri's resource cost is low

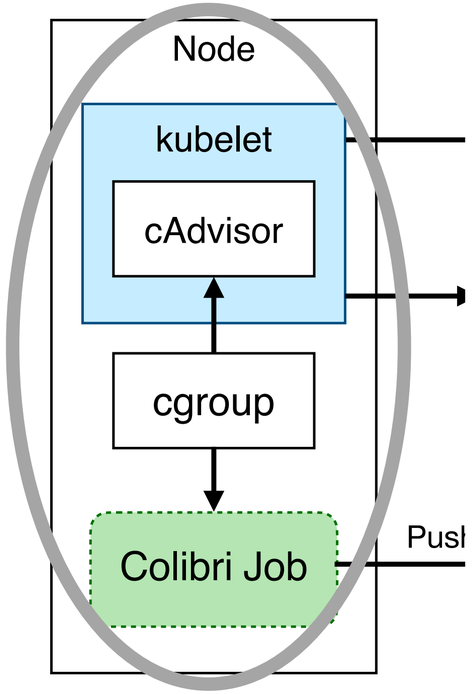
Comparing the API interface/central manager



System	Metrics server	Prometheus		Colibri API server
		Adapter	Operator	
Image size (MB)	68.8	68.8	194	107
Query interval	15 sec	N/A	15 sec / 10 ms	N/A
CPU (%)	0.6	0.8	0.9 / 139	**0.5**
RAM (MB)	18	22	207 / 250	**12**
Ingress bandwidth (Byte/s)	870	1K	12K / 703K	**191**
Egress bandwidth (Byte/s)	968	2K	372 / 117K	**184**

Colibri's resource cost is low

Comparing the querying worker



Query function	cAdvisor		Colibri Job
Image size (MB)	83.1		48.1
Query interval	15 sec	10 msec	10 msec
CPU (%)	0.59	337	6.5 **(1.93%)**
RAM (MB)	29	1005	21 **(2.09%)**
Ingress bw (KB/s)	0.05	63	1.8e-3 **(<0.01%)**
Egress bw (KB/s)	0.85	911	0 **(<0.01%)**

Summary

- MEC QoS relies on accurate, lightweight profiling process
- Colibri is fine-grained (**millisecond**-level metric query) and lightweight resource profiling system for MEC workload and environments
- Exploring more use cases to improve Colibri's system integration and scalability

Thank you !