Compute Resource Allocation

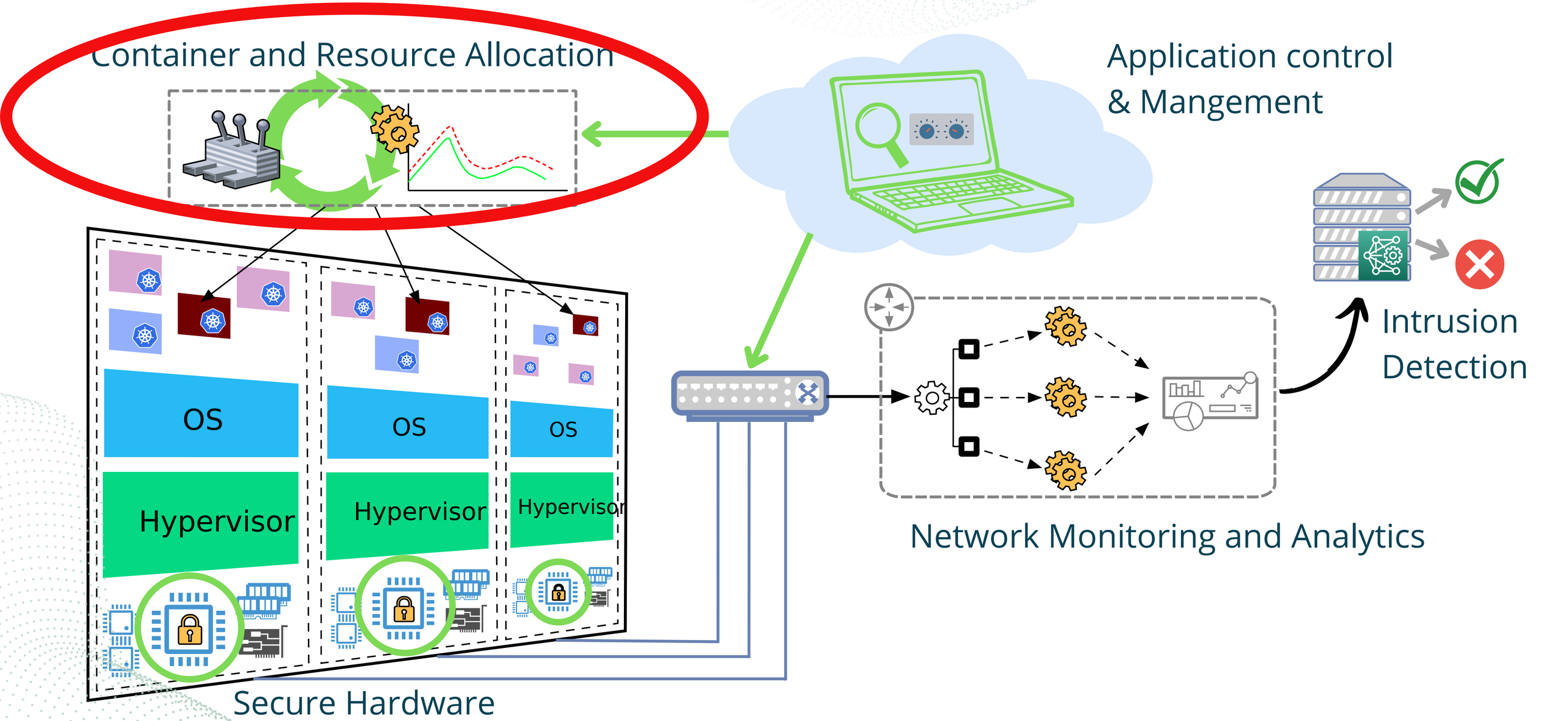
Publications:

G. Cusack, M. Nazari, S. Goodarzy, E. Hunhoff, P. Oberai, E. Keller, E. Rozner, and R. Han.
Escra: Event-driven, Sub-second Container Resource Allocation
42nd IEEE International Conference on Distributed Computing Systems (ICDCS ’22), 2022

G. Cusack, M. Nazari, S. Goodarzy, P. Oberai, E. Rozner, E. Keller, R. Han.
Efficient Microservices with Elastic Containers
CoNEXT ’19 Companion (Poster)

University of Colorado **Boulder**

Cloud Architecture Overview



Containerized Infrastructure

- Light-weight
- Rich orchestration systems
- Development workflow integration
- Strong resource isolation

Developers must make tradeoffs!

Developers must make tradeoffs

Prioritize Performance

Overallocate compute resources

Prioritize Resource Efficiency

Underallocate compute resources

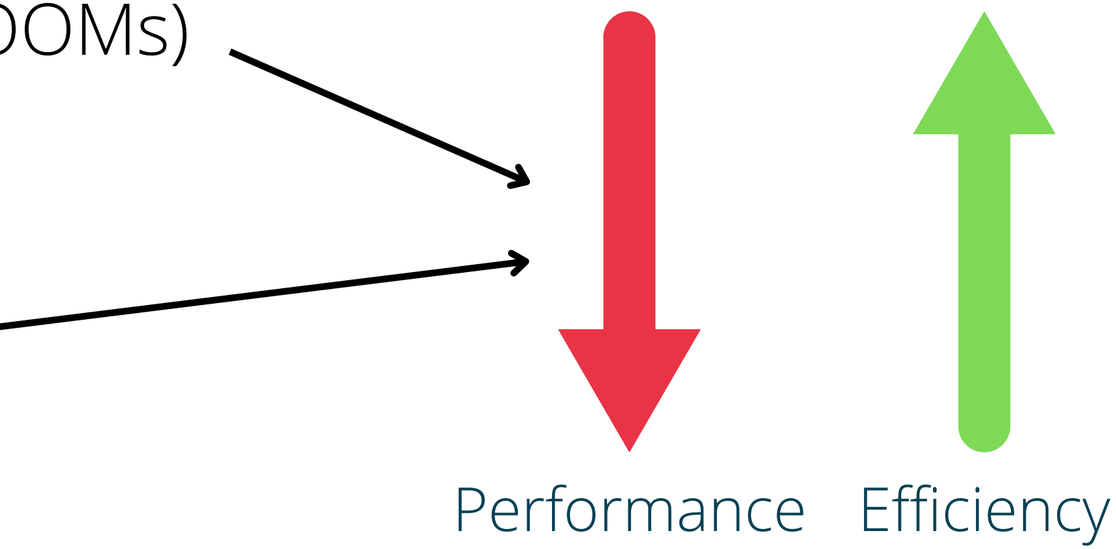
Developers must make tradeoffs

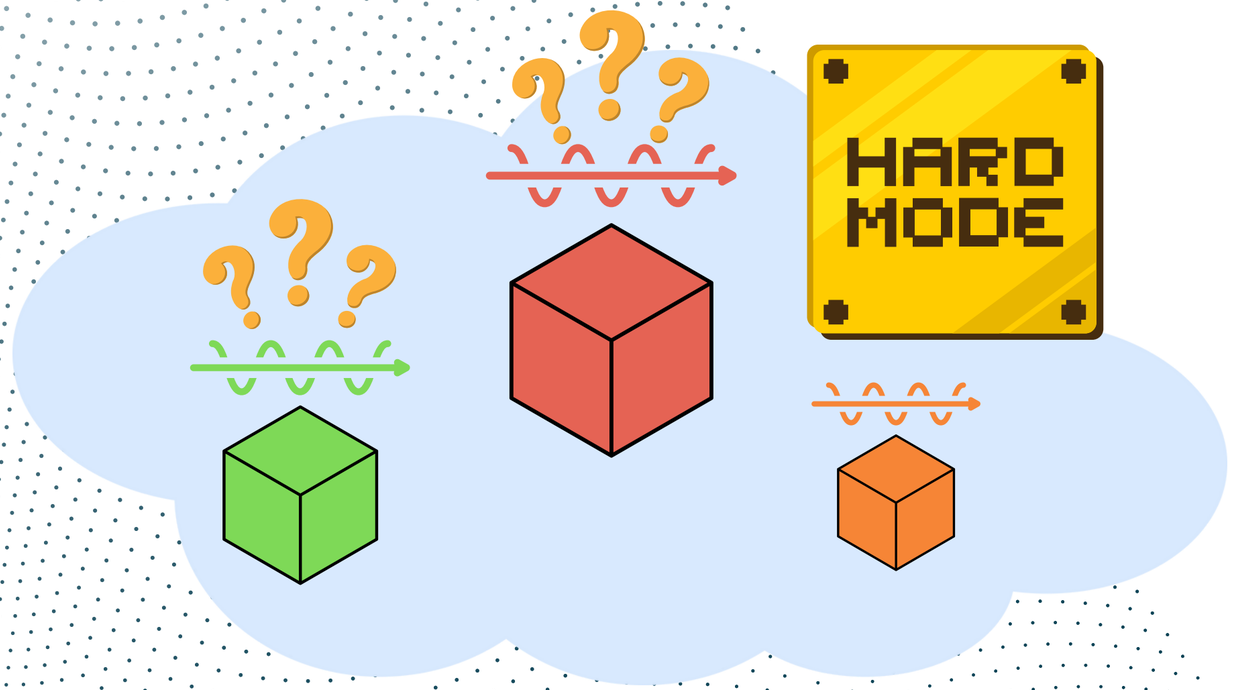
Prioritize Resource Efficiency

Underallocate compute resources

Out of Memory Events (OOMs)

CPU Throttles





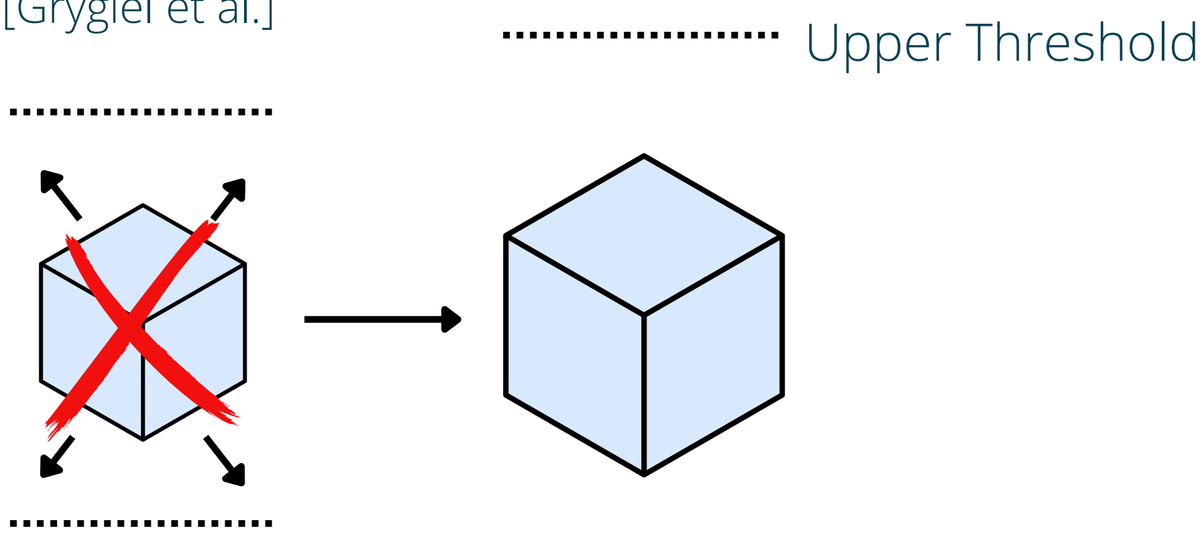
Setting Container Limits is Difficult!

- Requires representative workload
- Resource needs vary over time
- Tools aggregate usage over coarse-grained timescales
- **Result: Over provisioned containers**

Recent Work

Threshold Scaling

VPA
[Grygiel et al.]



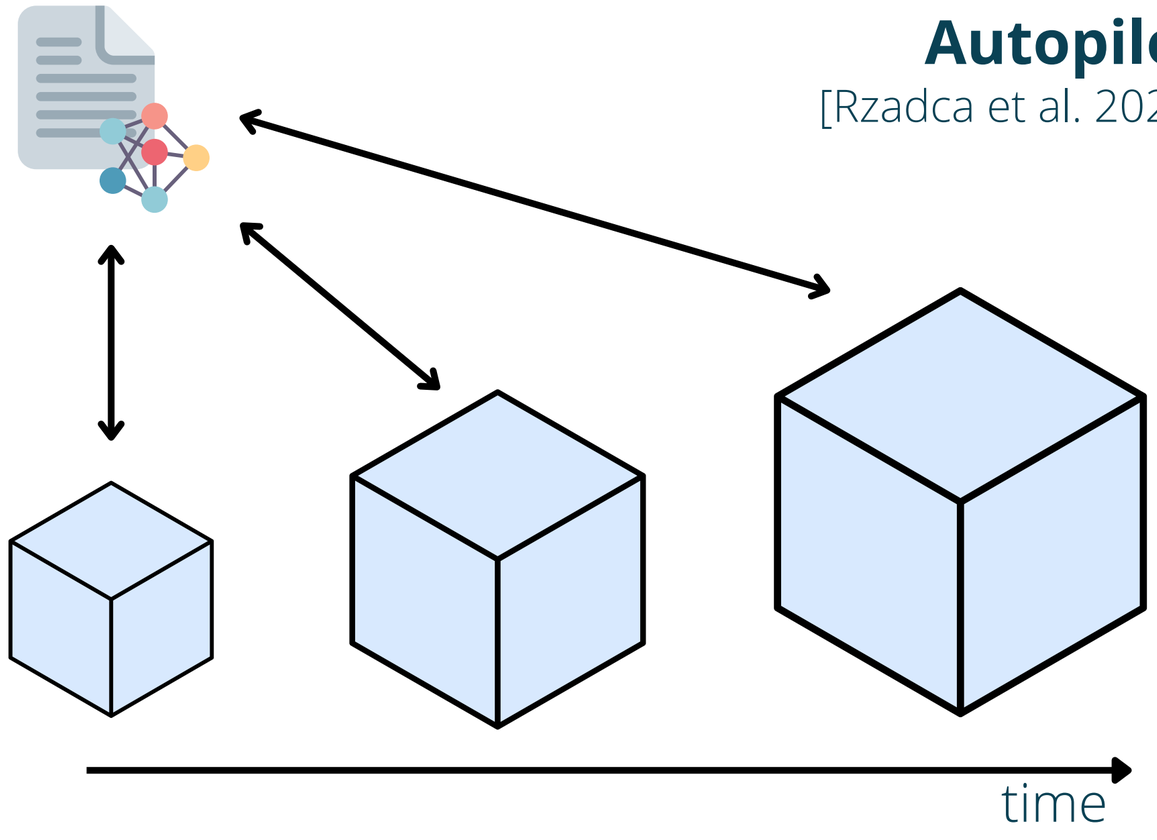
Shortcomings

- Container restart to scale
- Scale once per minute
- High slack

Recent Work

ML-based Scaling

Autopilot
[Rzadca et al. 2020]



Shortcomings

- Cannot respond to quick changes in workload
- Must allocate to maximum prediction over next 5-minute interval
- Unable to correct inaccurate predictions
- Not suitable for short-lived containers (e.g. serverless)

Key Insight

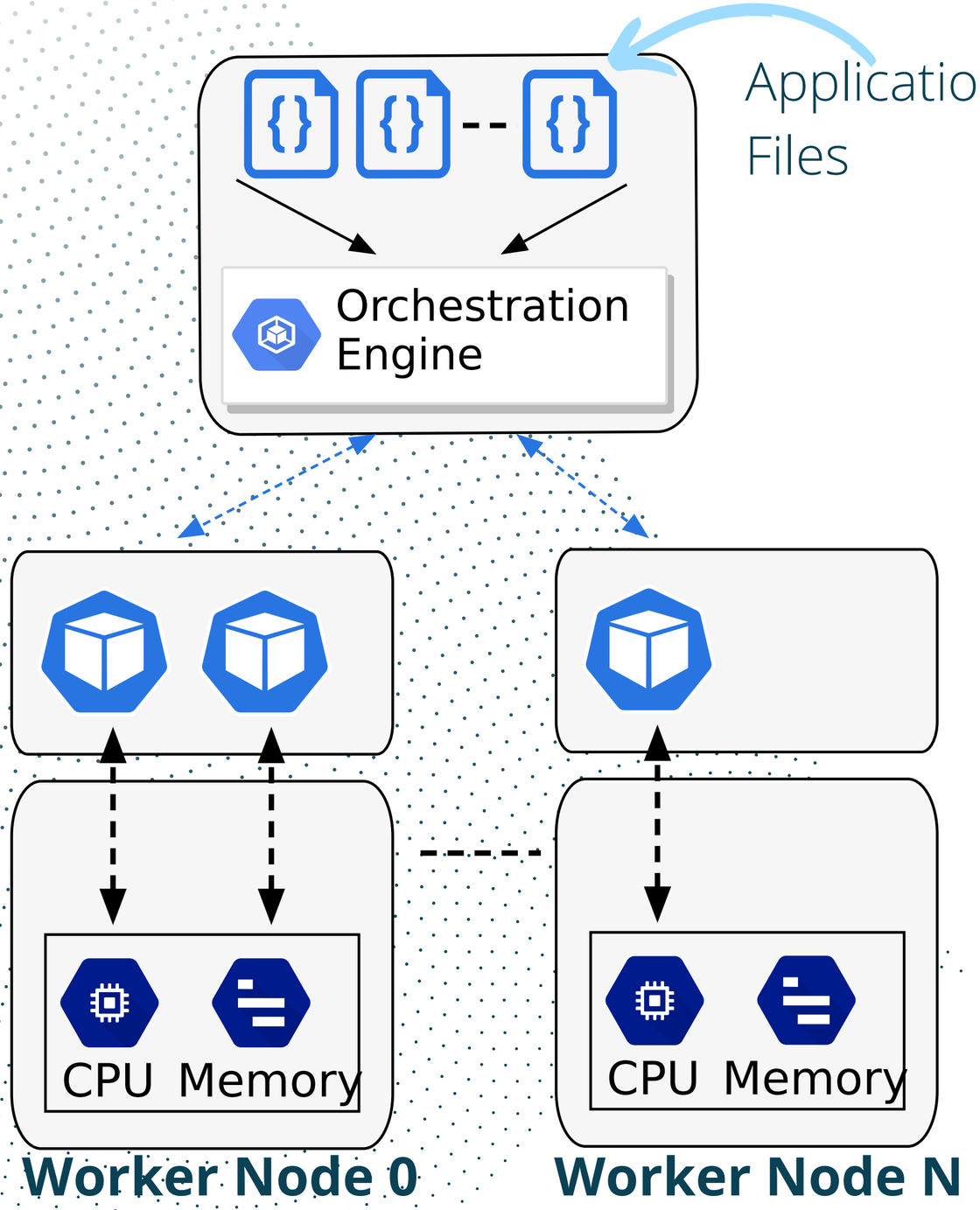
If we can make resource allocation fine grained, we don’t need to predict; we can react to workload changes.

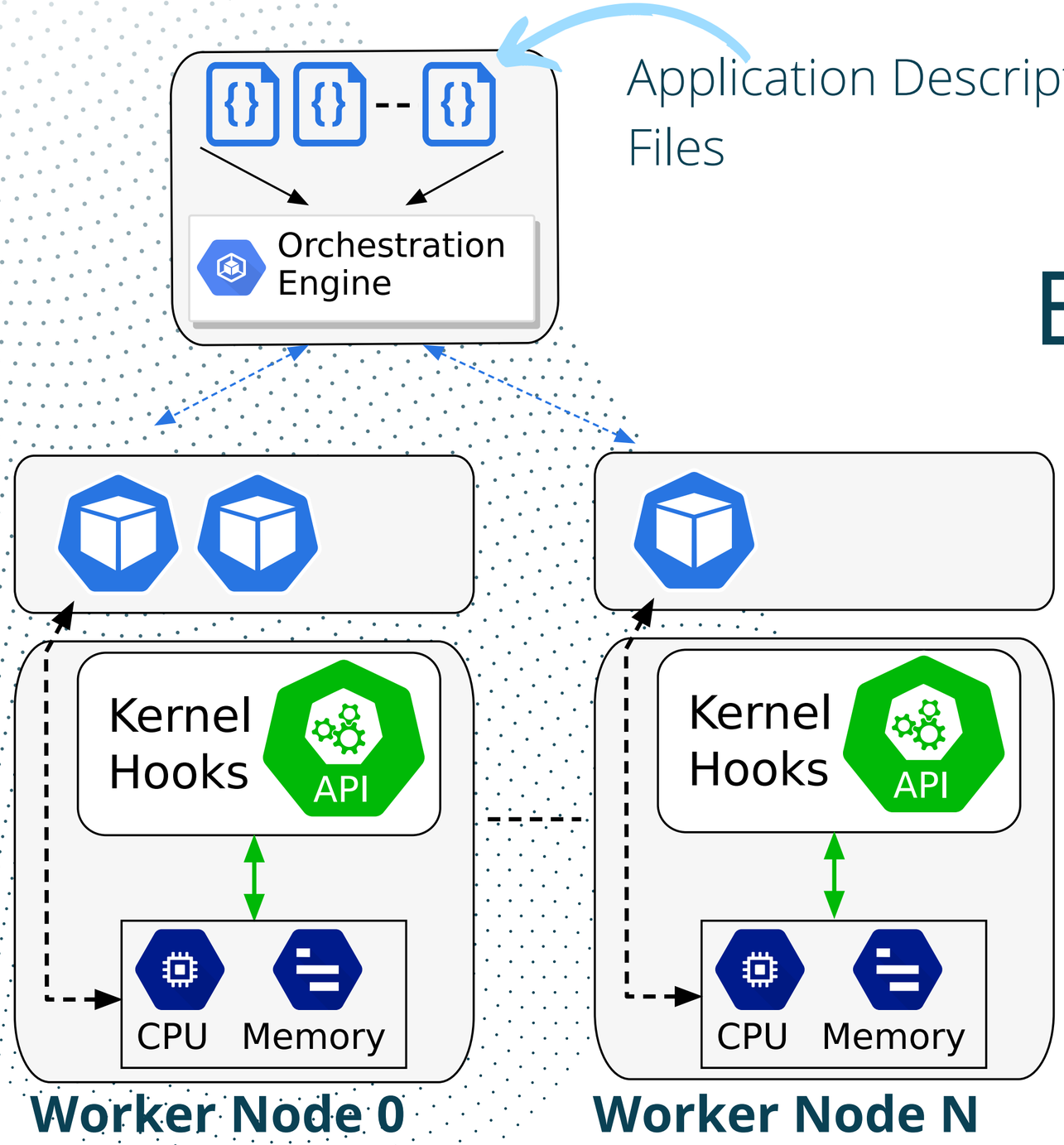
Escra

- Distributed resource allocation
- Sub-second interval scaling within and across hosts
- OOM prevention and scaling
- Immediate CPU throttle response
- No performance penalty

University of Colorado **Boulder**

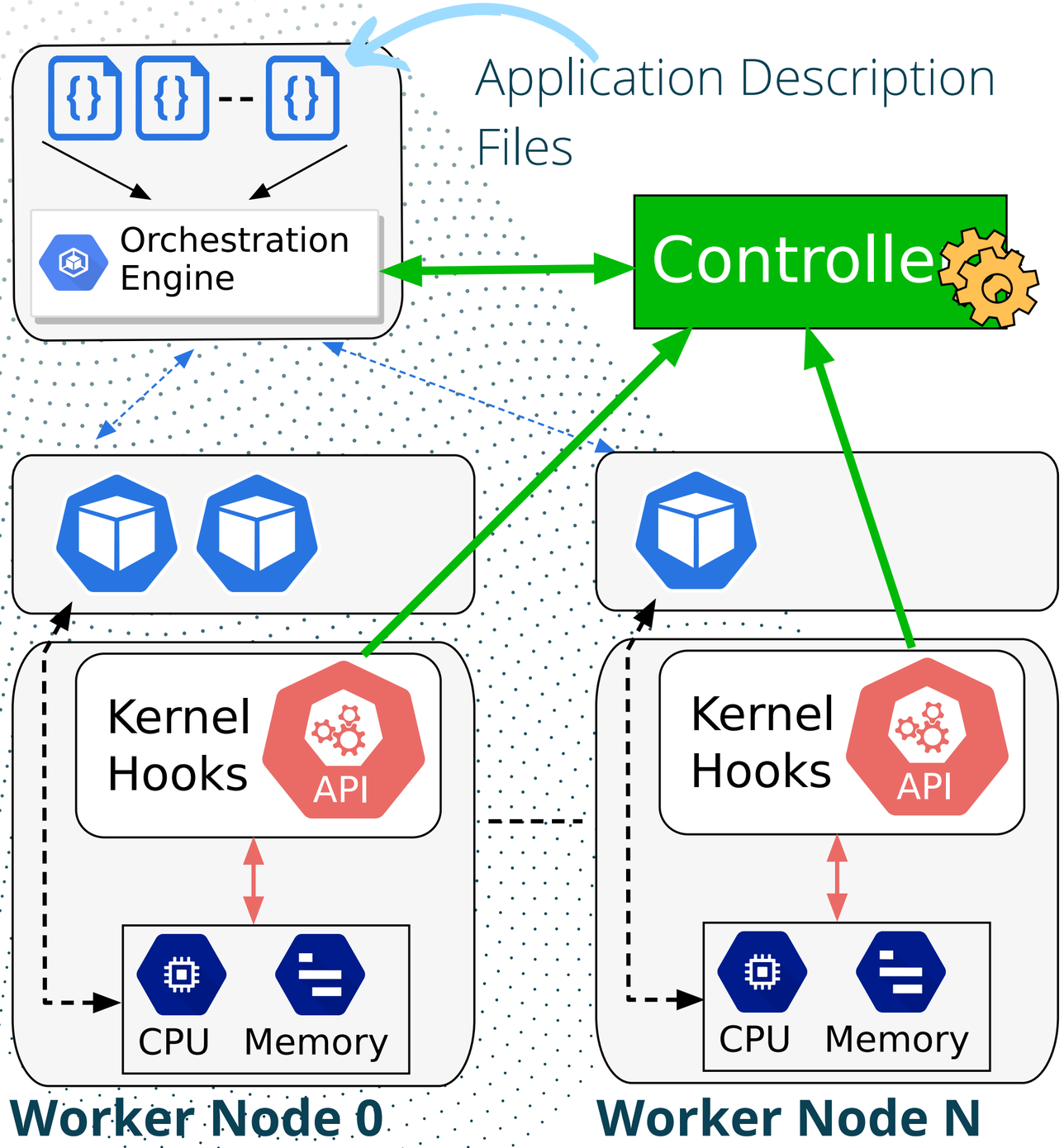
Simplified Cloud Orchestration Architecture





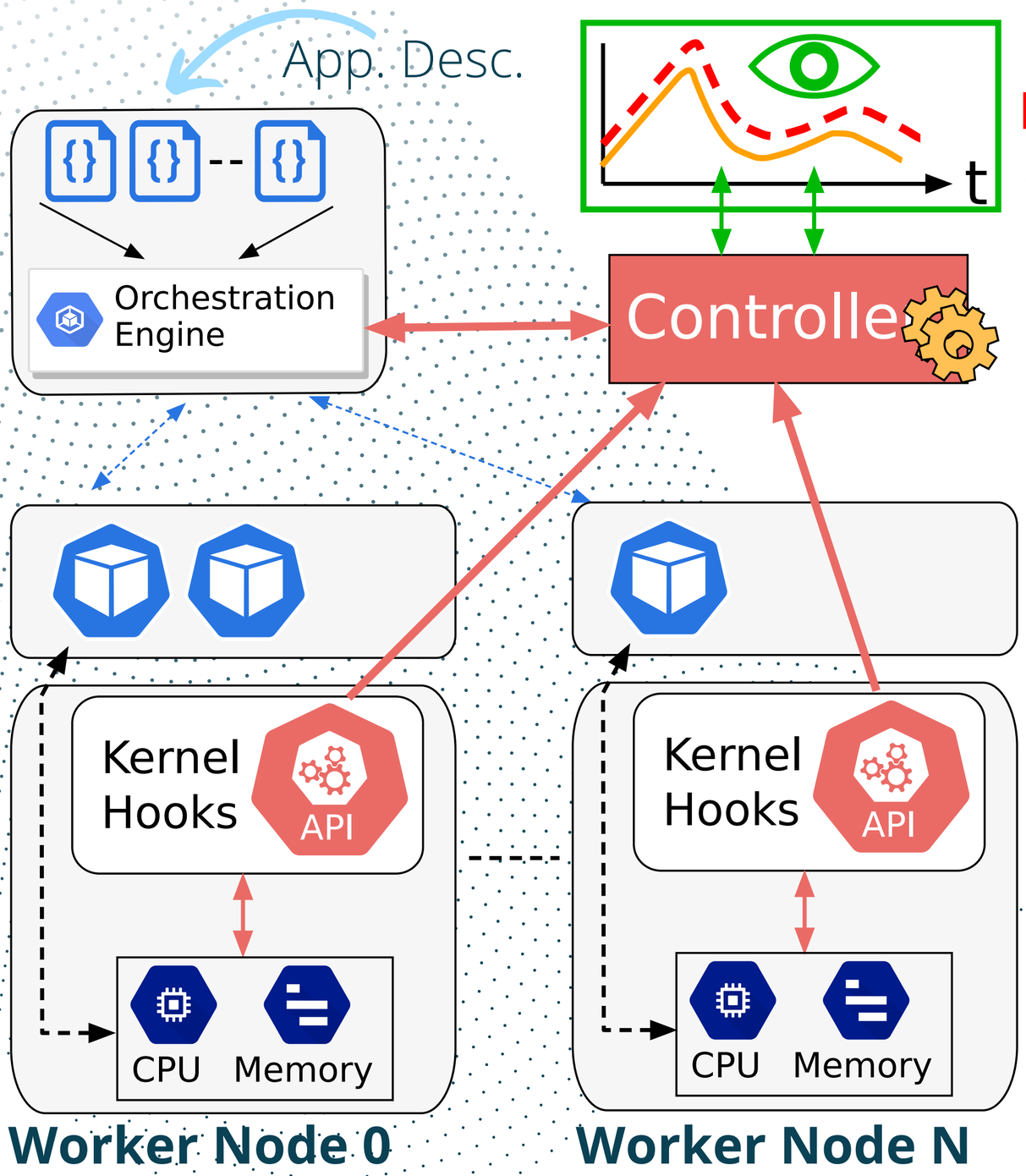
Escra Architecture

- **Kernel Hooks**



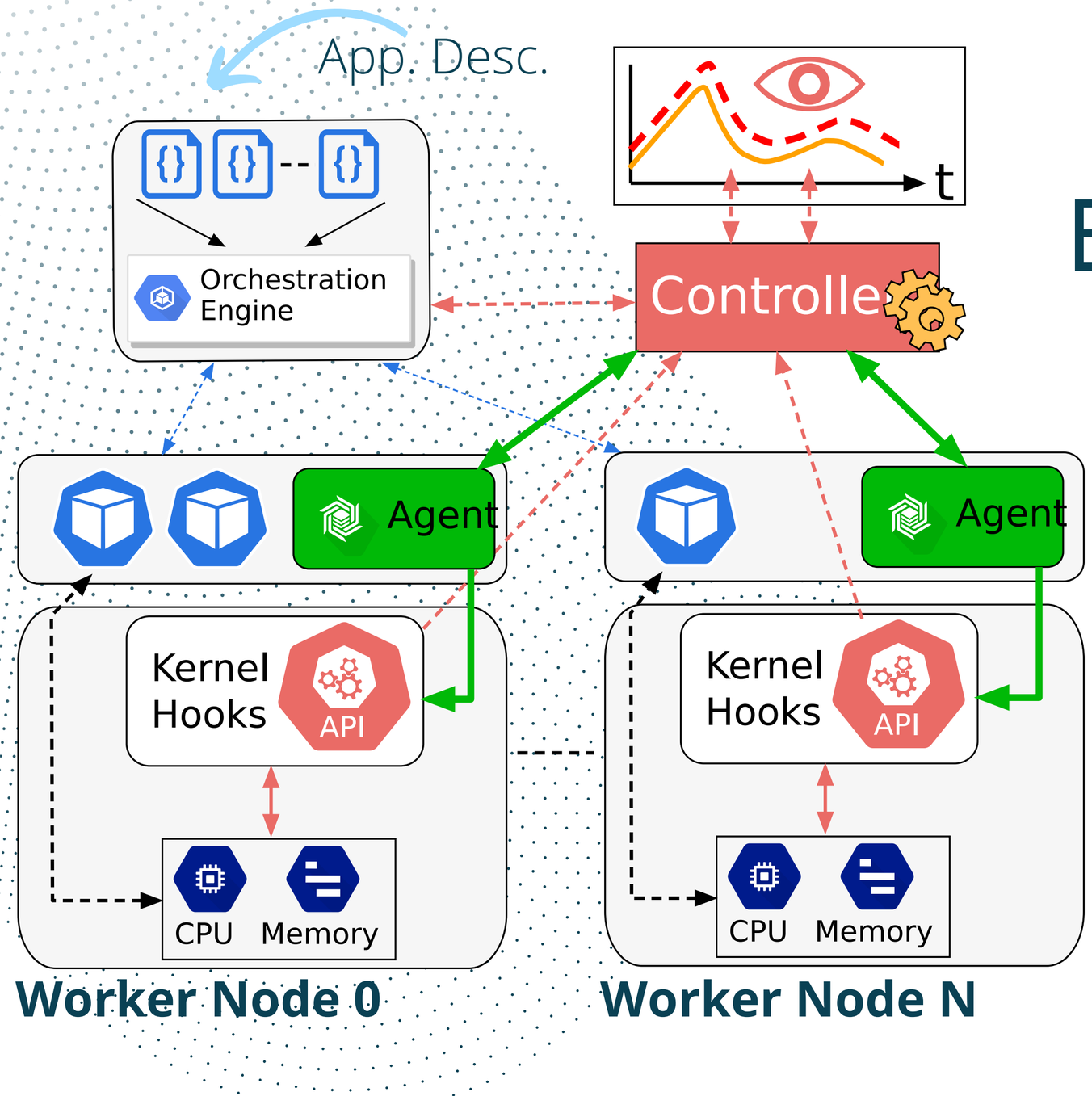
Escra Architecture

- Kernel Hooks
- **Controller**



Escra Architecture

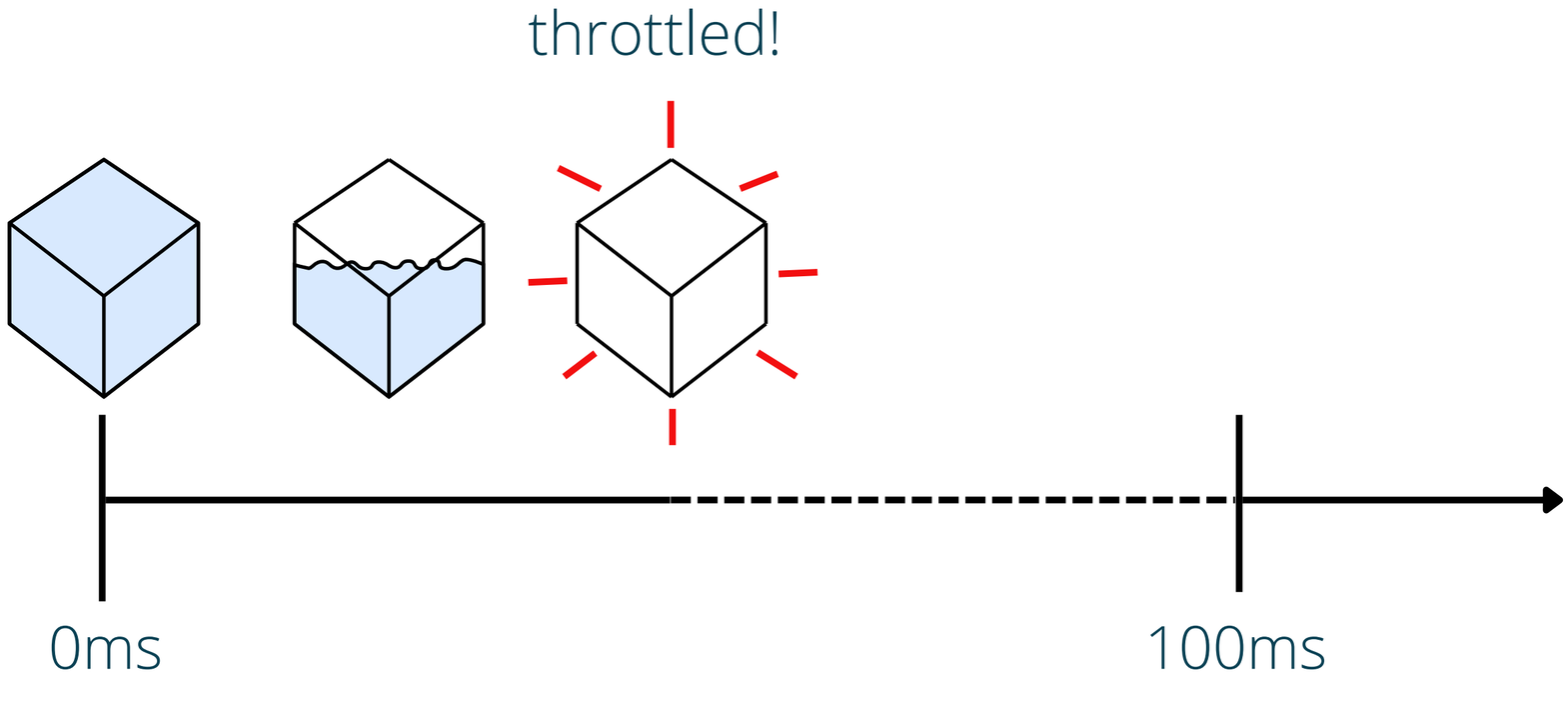
- Kernel Hooks
- Controller
- **Resource Allocator**



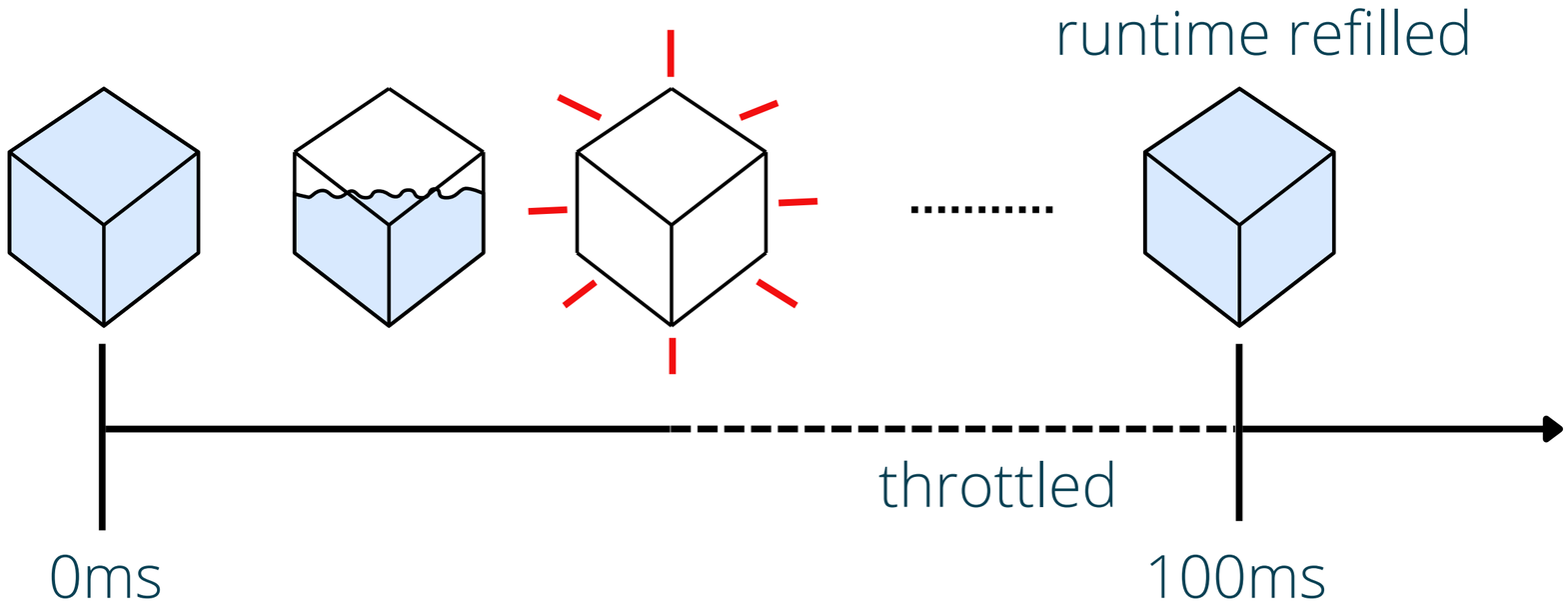
Escra Architecture

- Kernel Hooks
- Controller
- Resource Allocator
- **Agent**

CPU Telemetry and Scaling

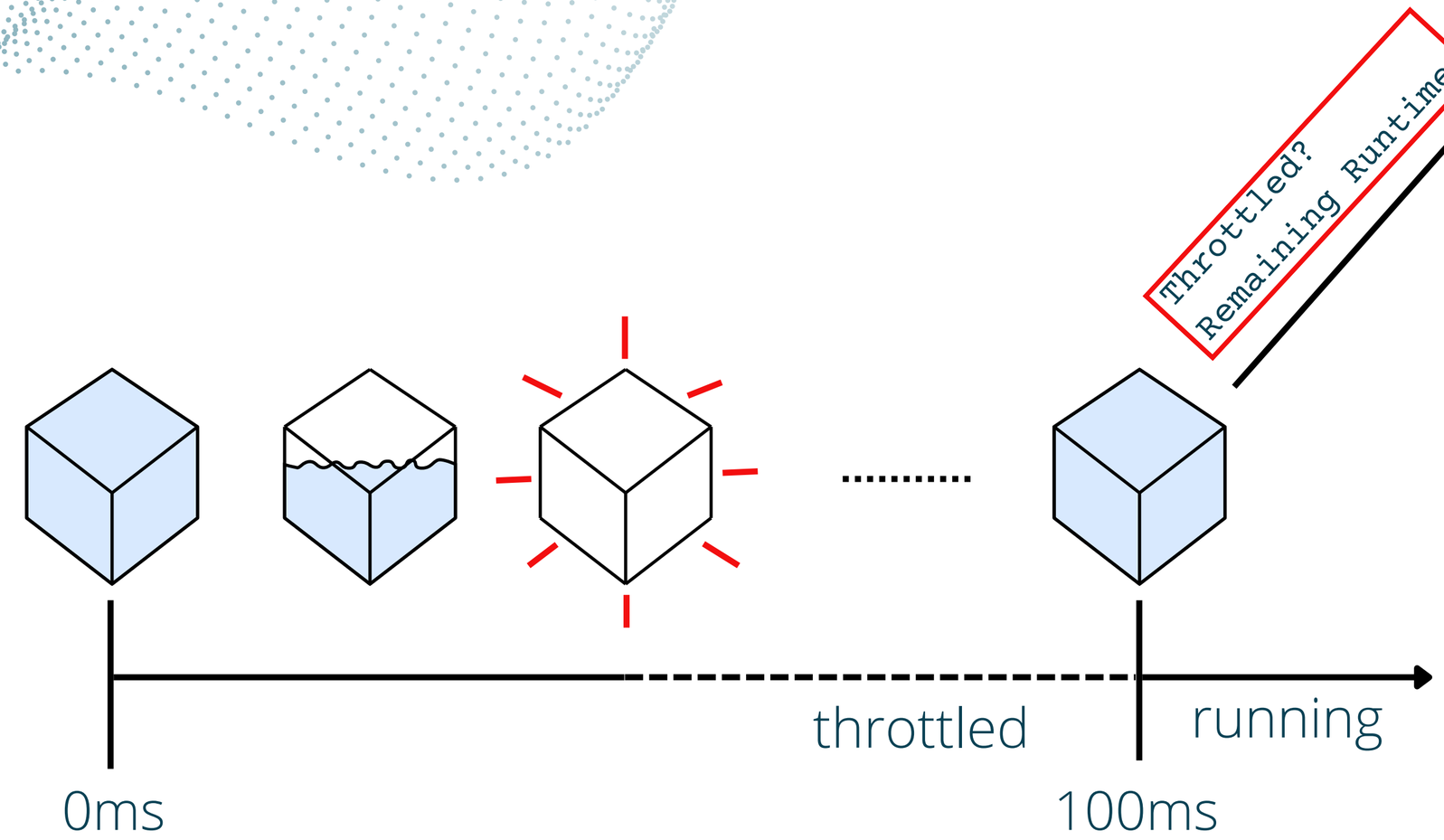


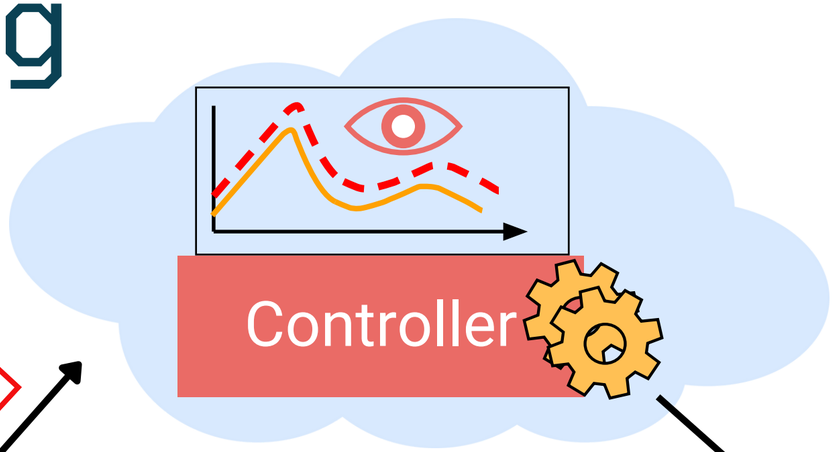
CPU Telemetry and Scaling



University of Colorado **Boulder**

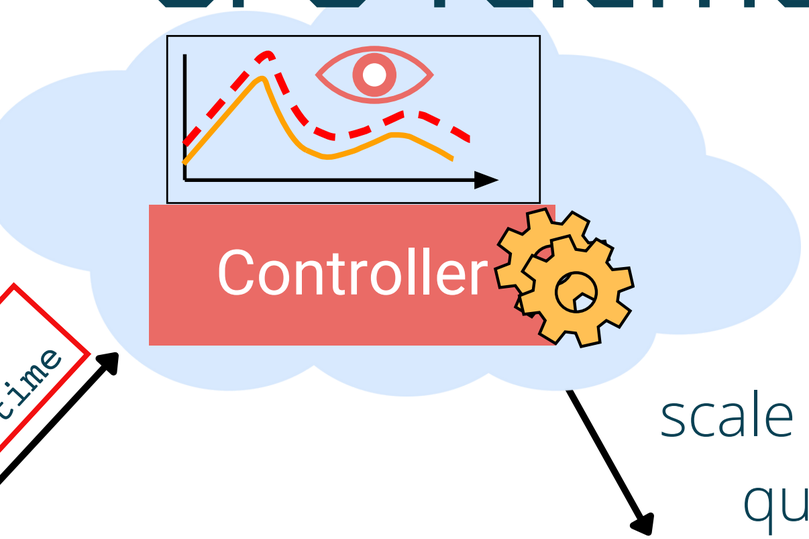
CPU Telemetry and Scaling





scale container
quota up

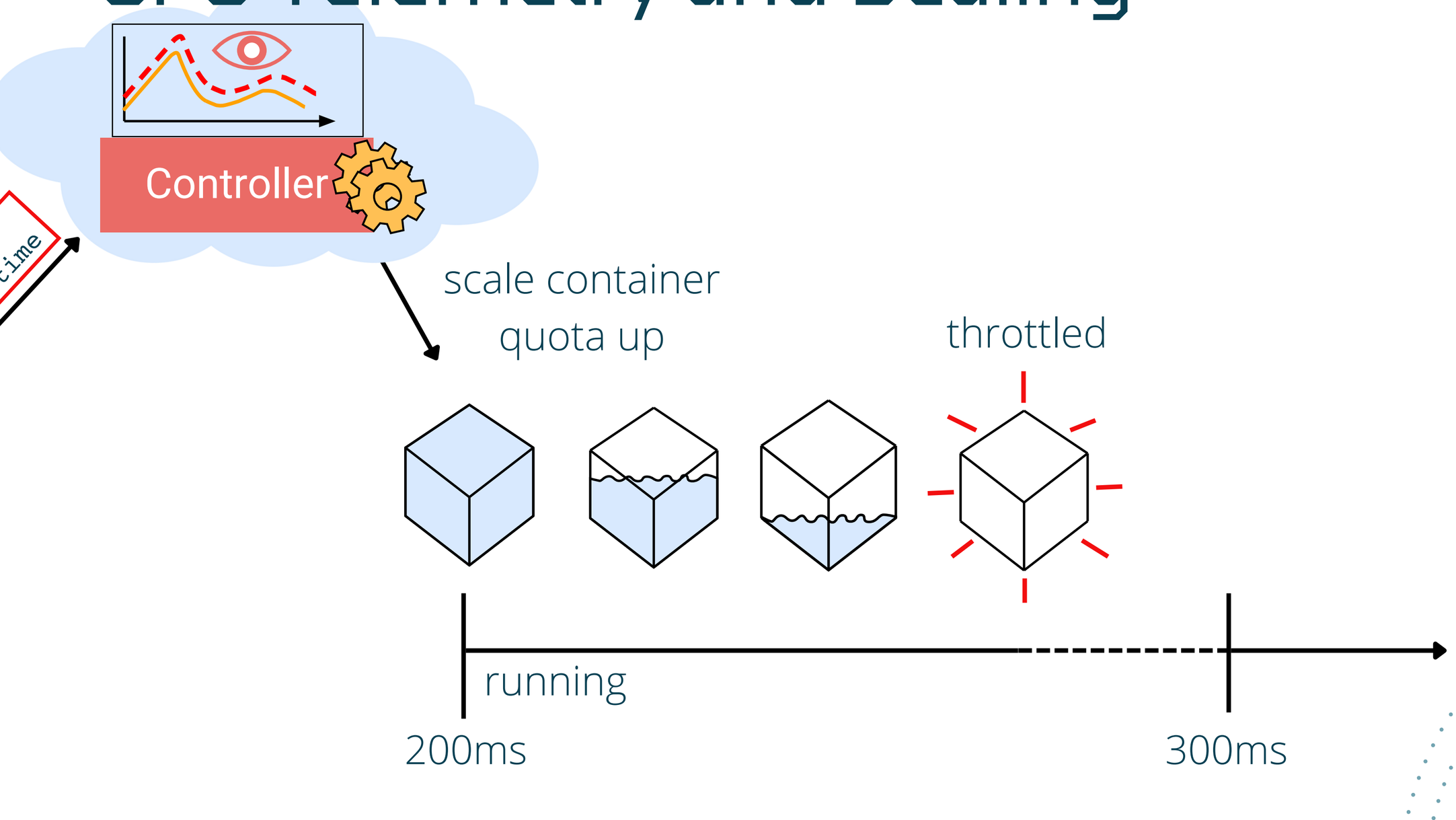
CPU Telemetry and Scaling



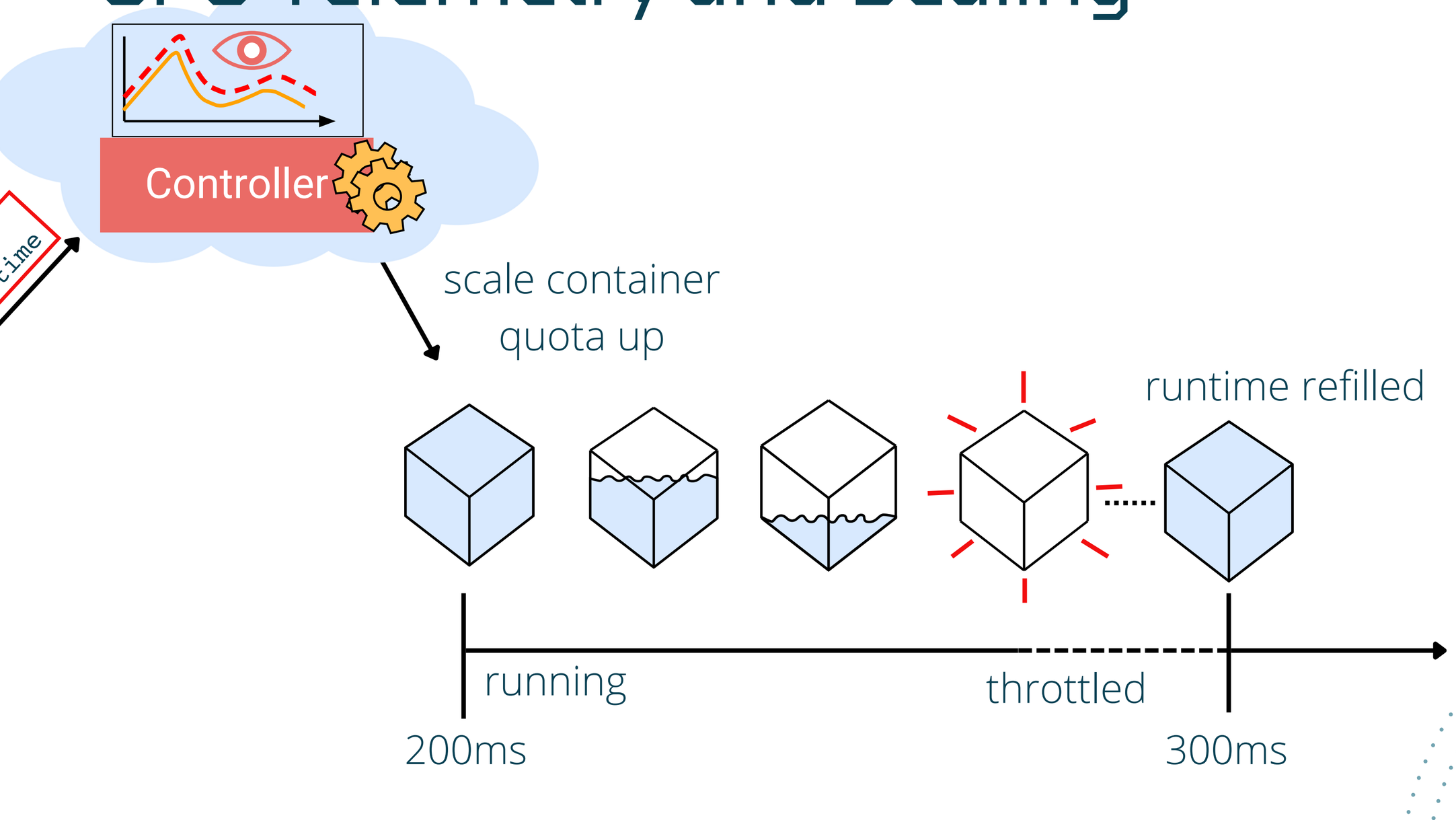
scale container
quota up

$$C(i)_q[t+1] = C(i)_q[t] + \frac{\sum_{t=0}^{n} C(i)_{th}[t]}{n} * \Upsilon(\Omega_l - \sum_{i=0}^{\lambda} C(i)_q[t])$$

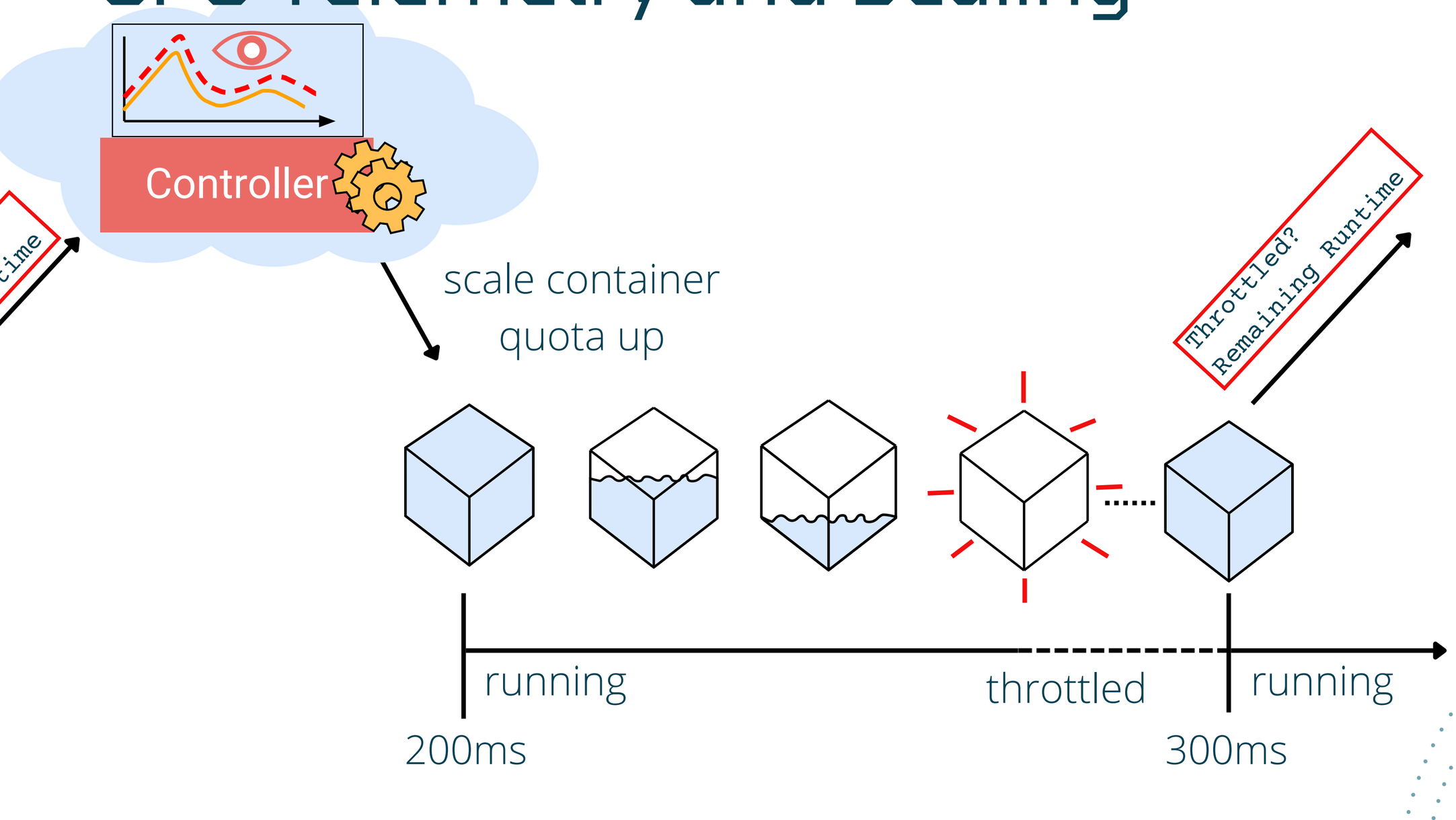
CPU Telemetry and Scaling



CPU Telemetry and Scaling

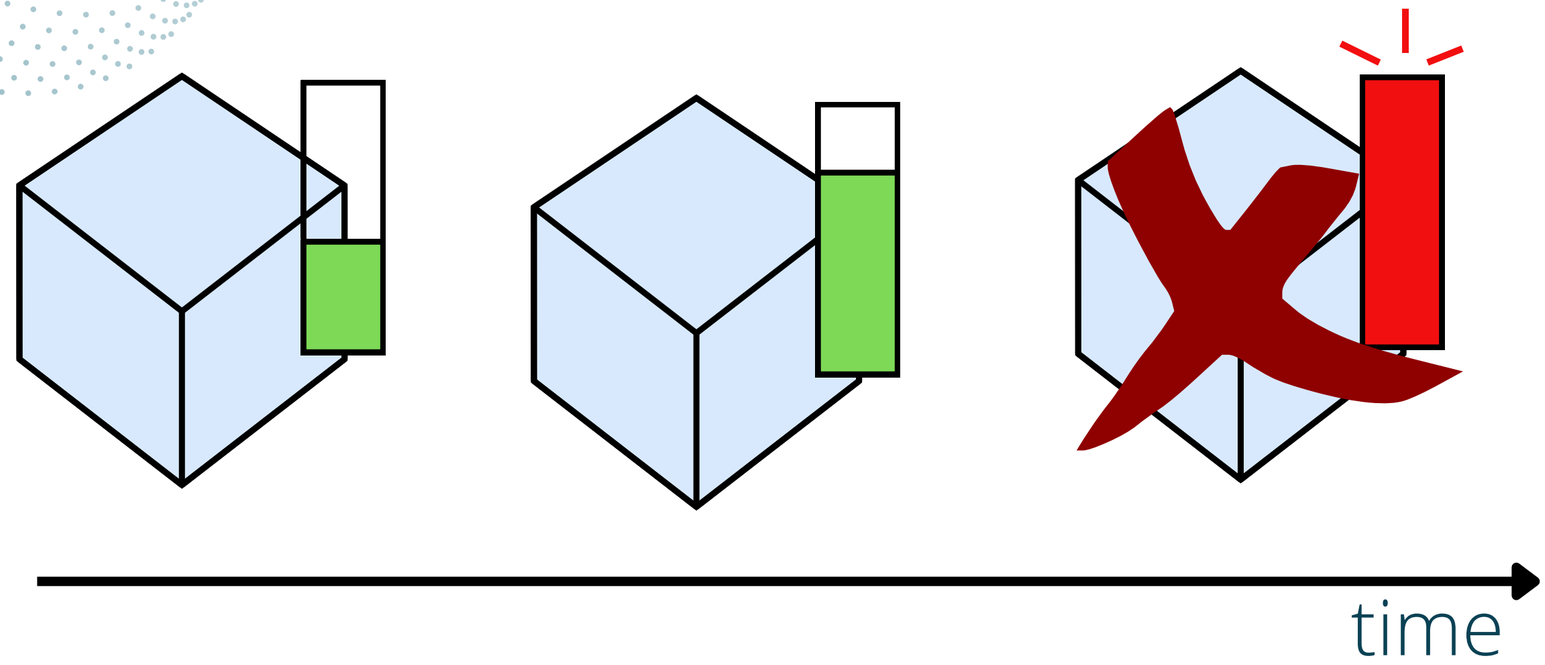


CPU Telemetry and Scaling



Event-Based Memory Scaling

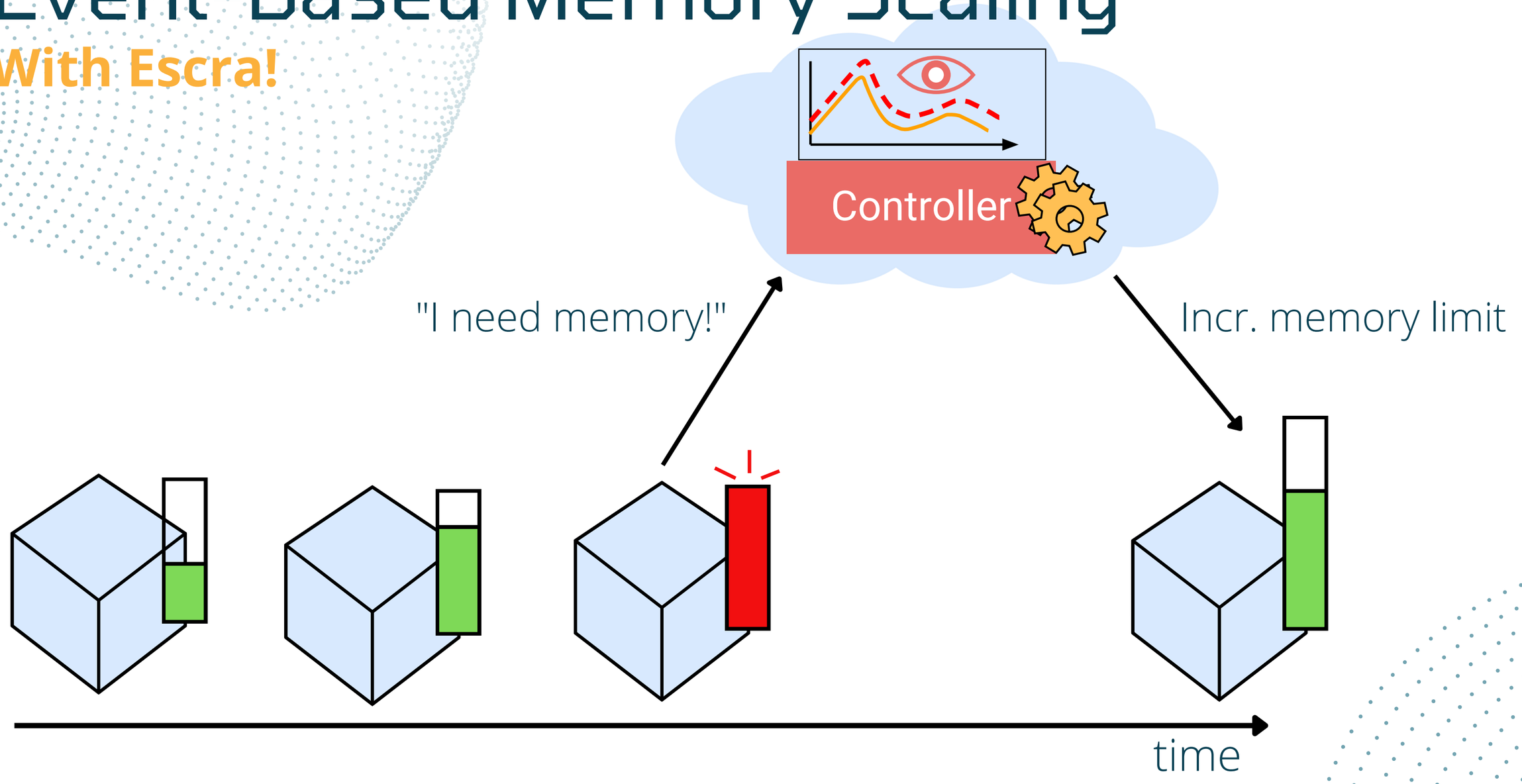
Typical Scenario



University of Colorado **Boulder**

Event-Based Memory Scaling

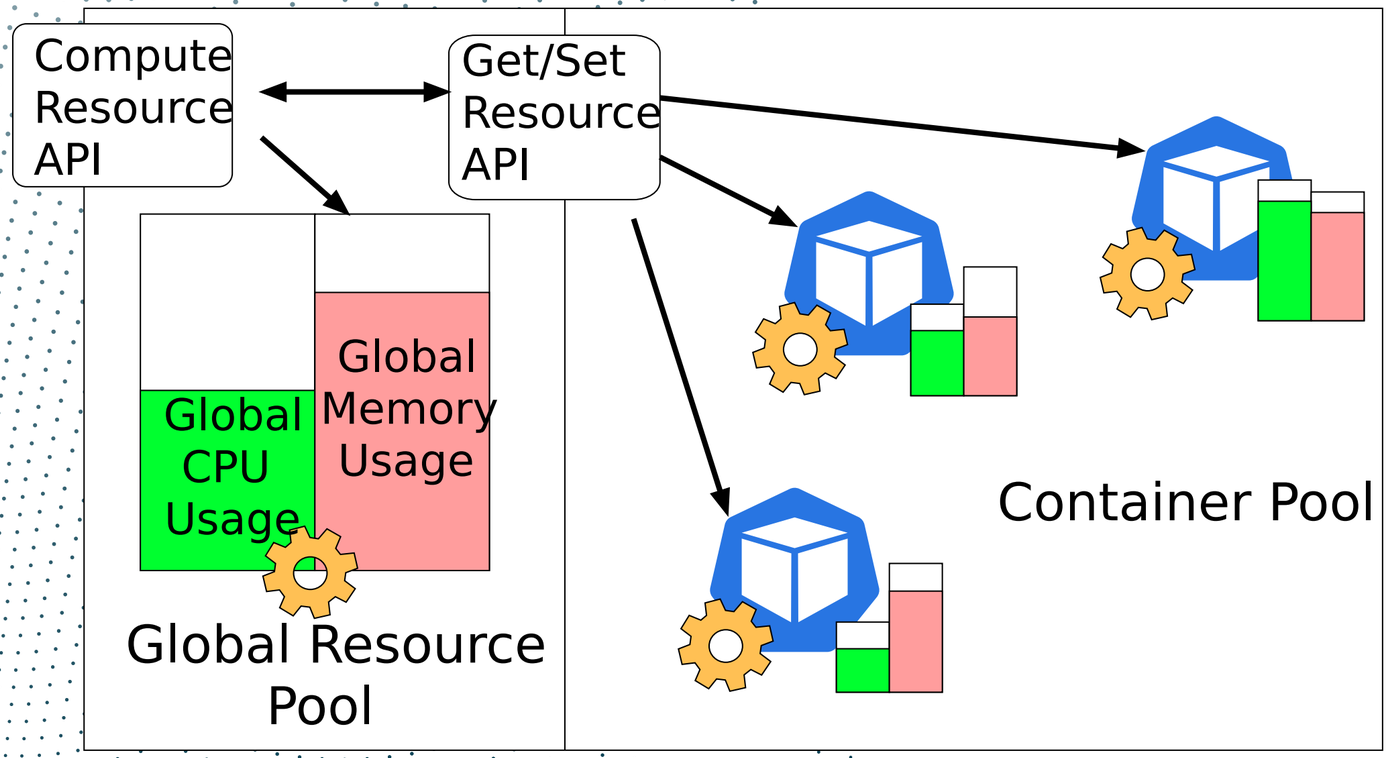
With Escra!



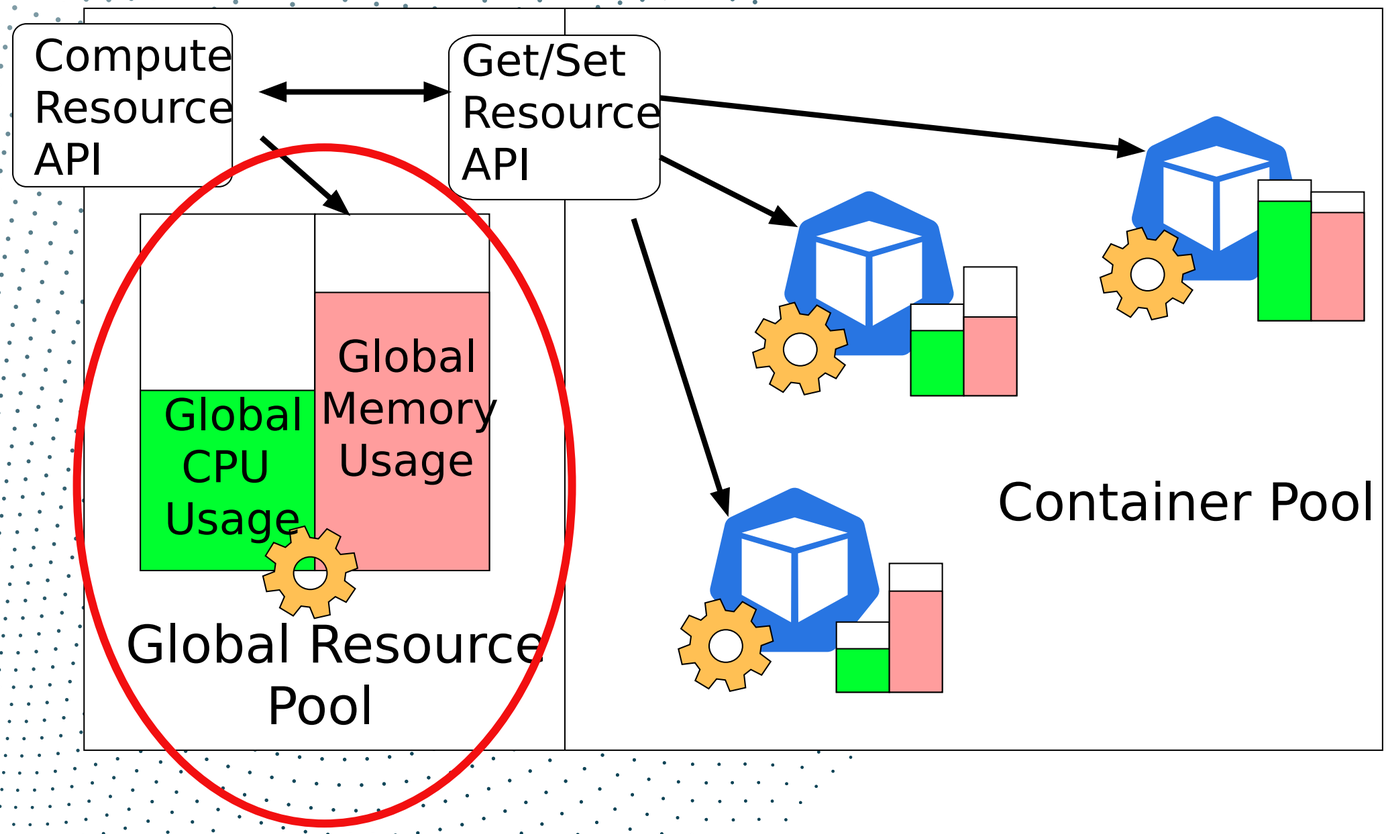
University of Colorado **Boulder**

Distributed Container

Distributed Container

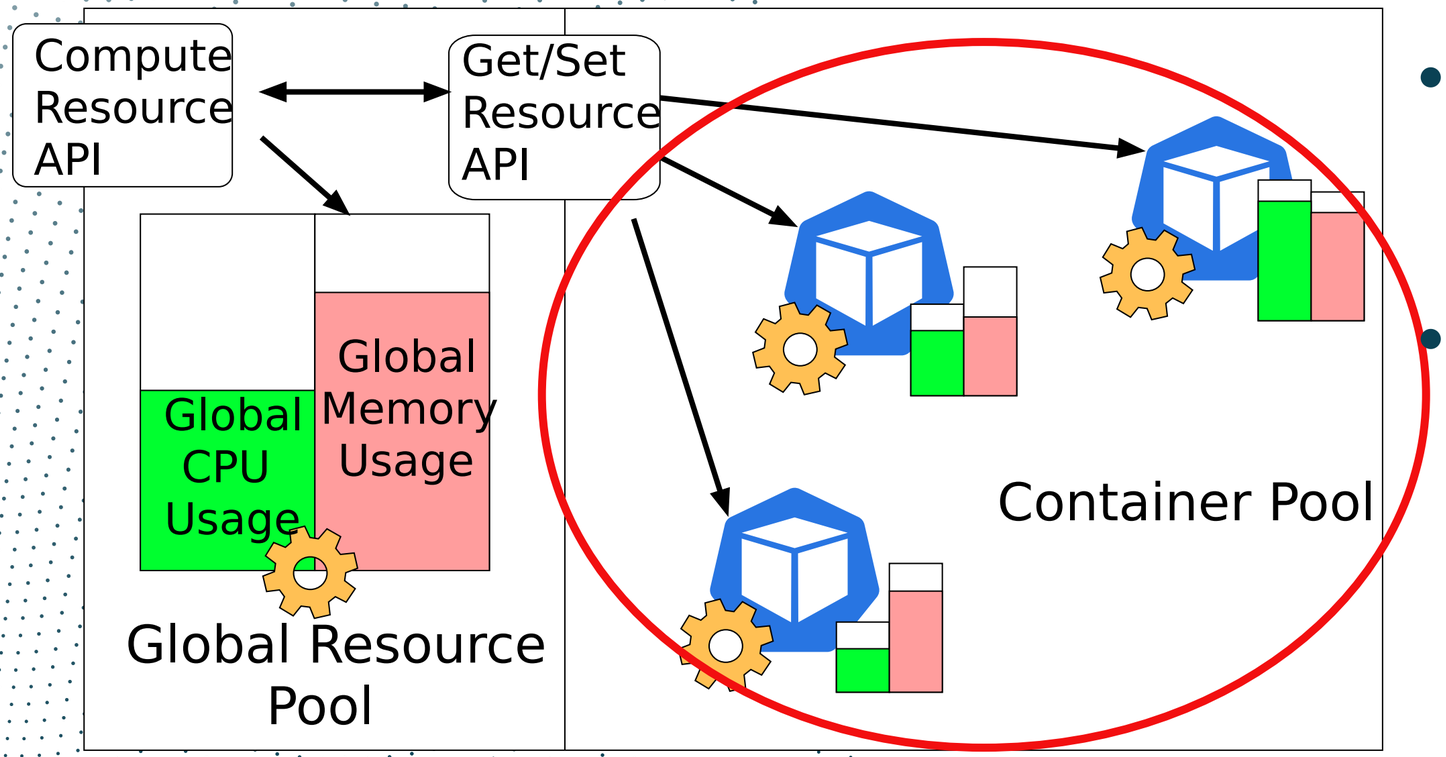


Distributed Container



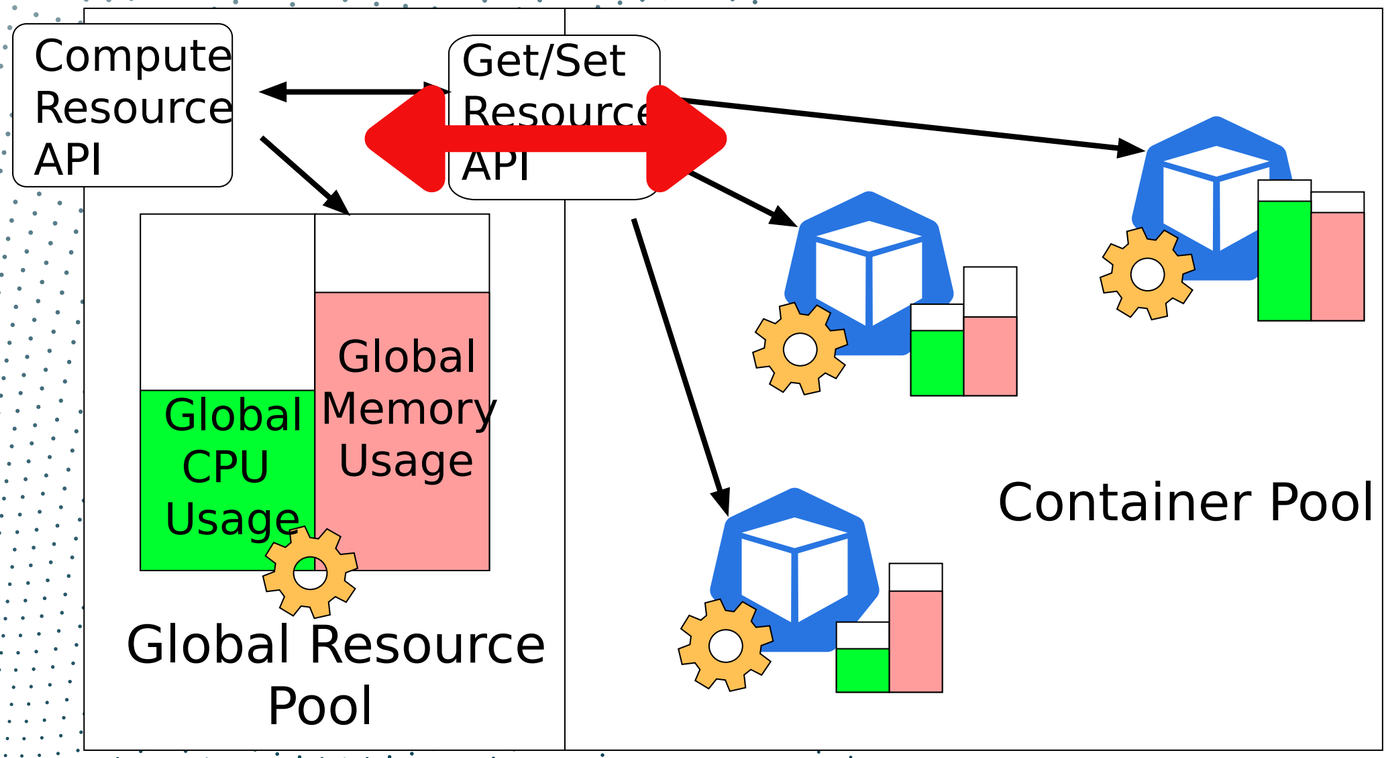
- Enforces per-application resources limits

Distributed Container



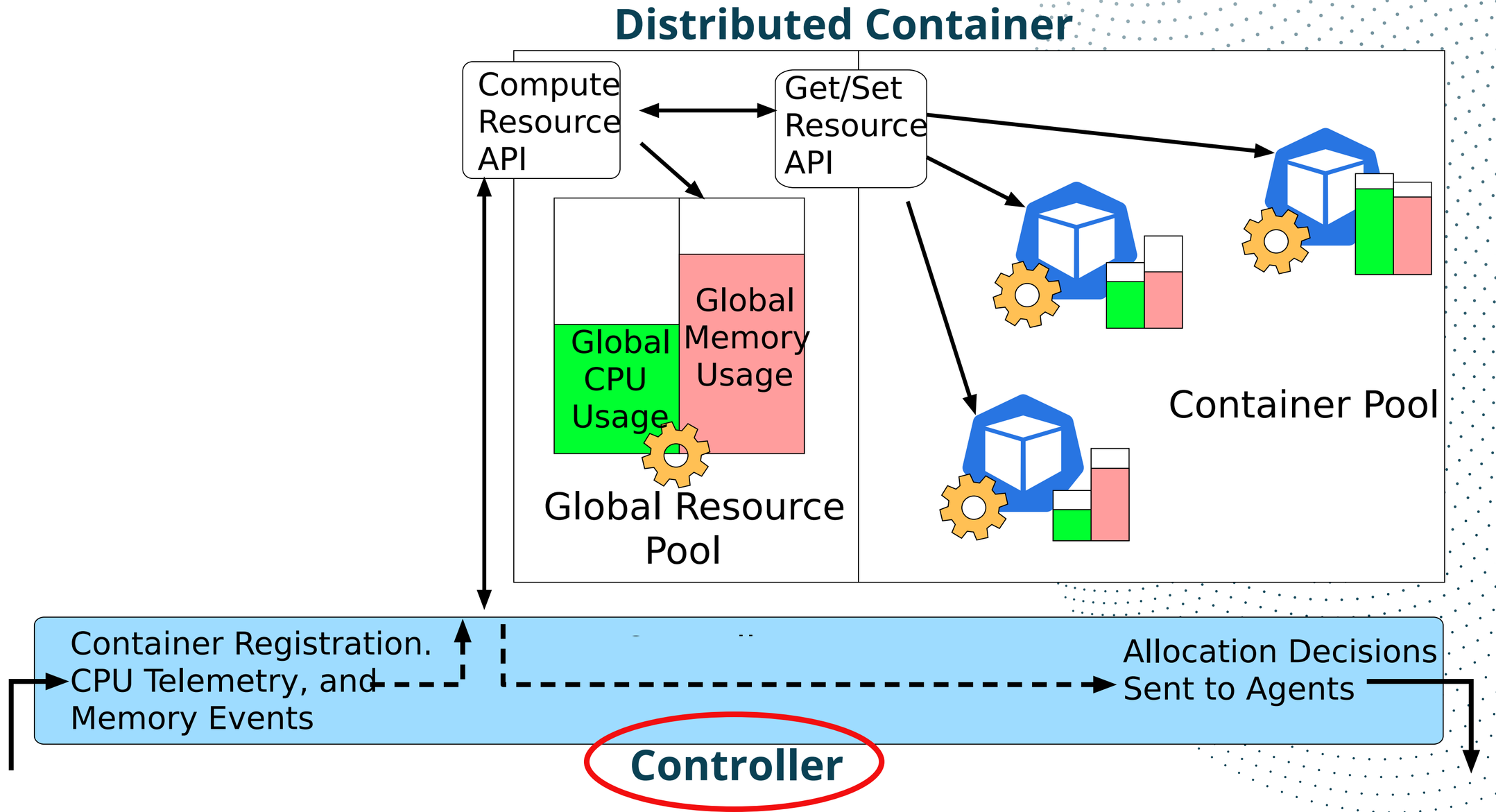
- Enforces per-application resources limits
- Per-container resource usage and limit tracking

Distributed Container

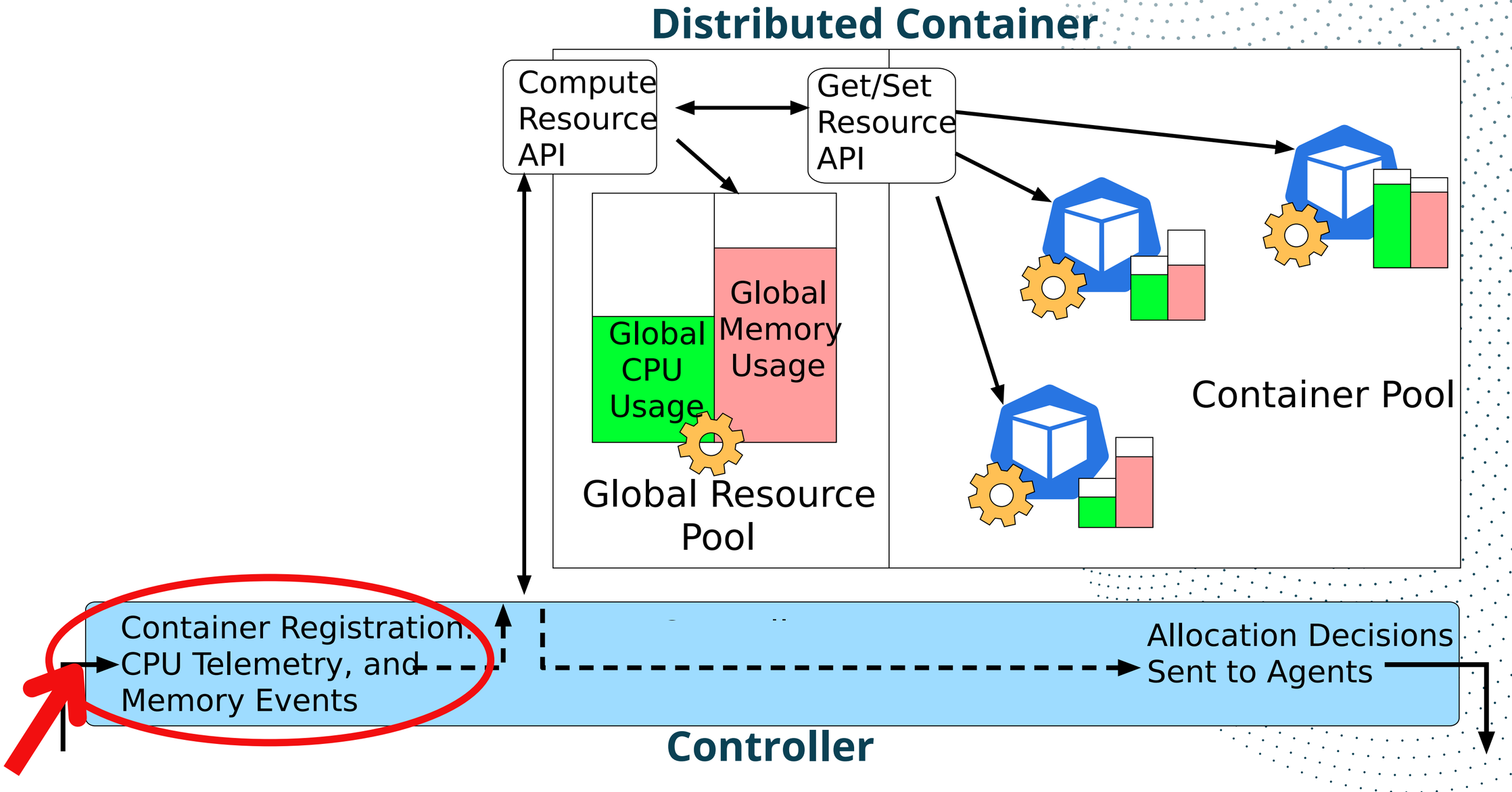


- Enforces per-application resources limits
- Per-container resource usage and limit tracking
- Containers dynamically share compute resources at runtime

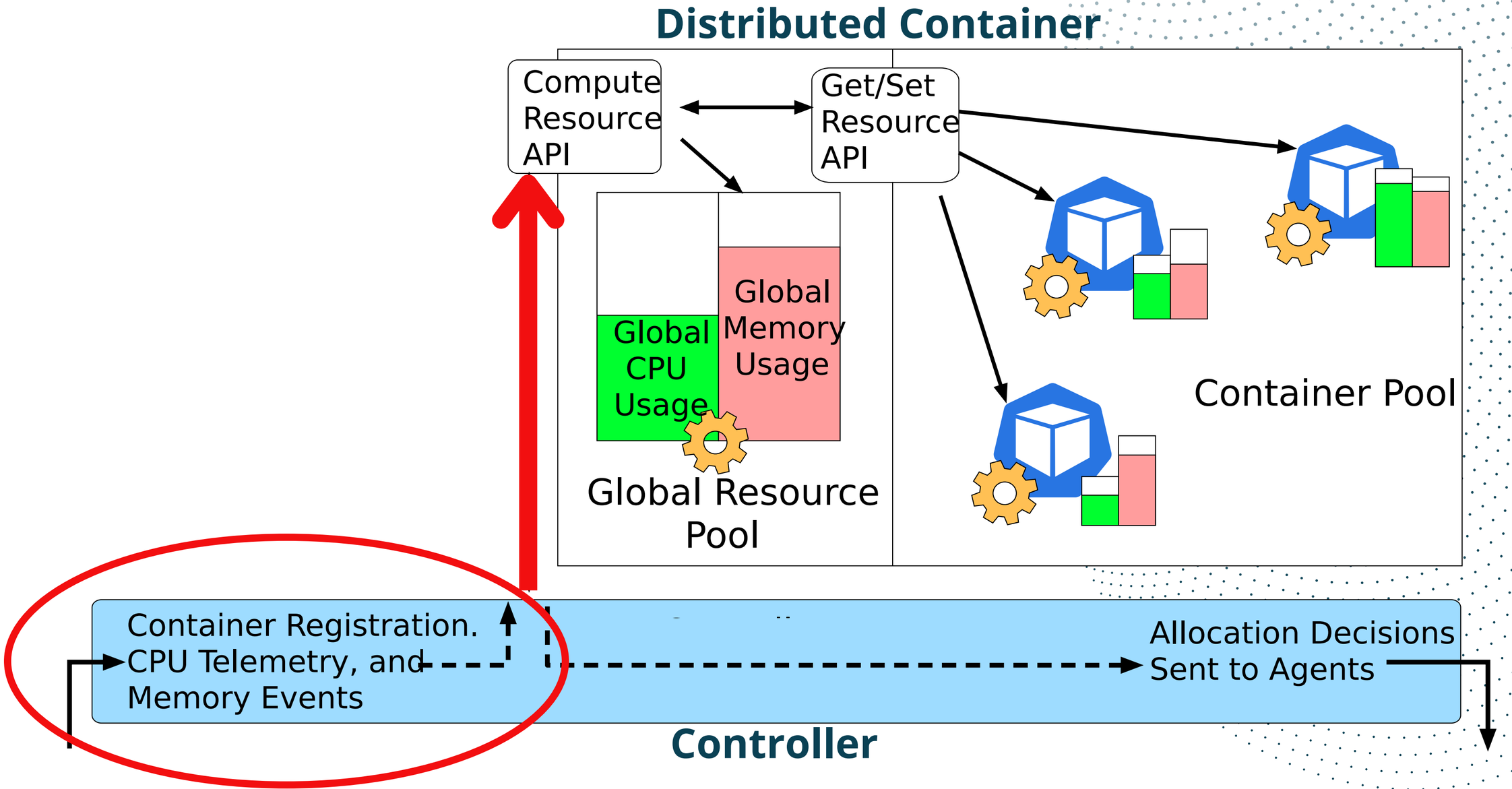
Escra Controller



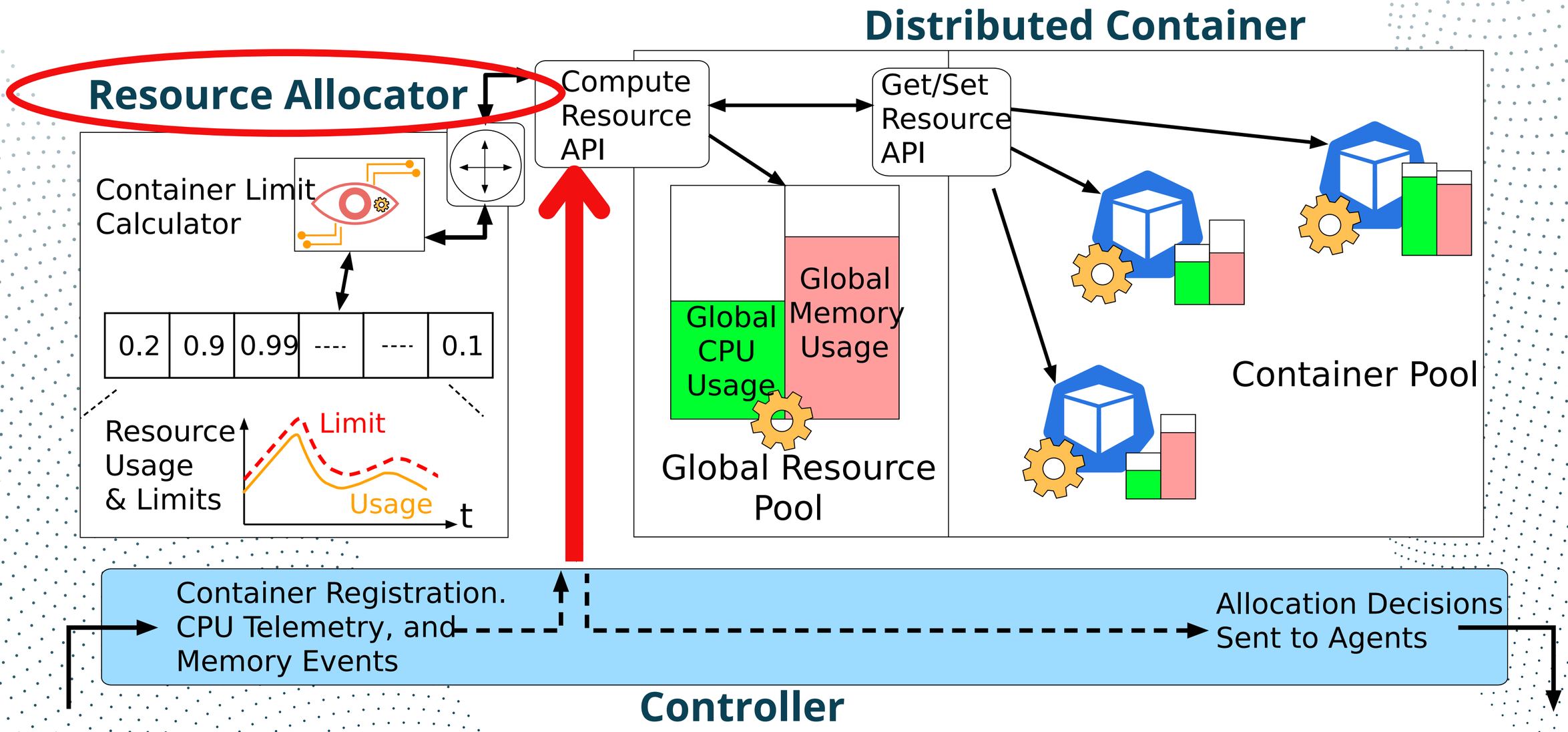
Escra Controller



Escra Controller

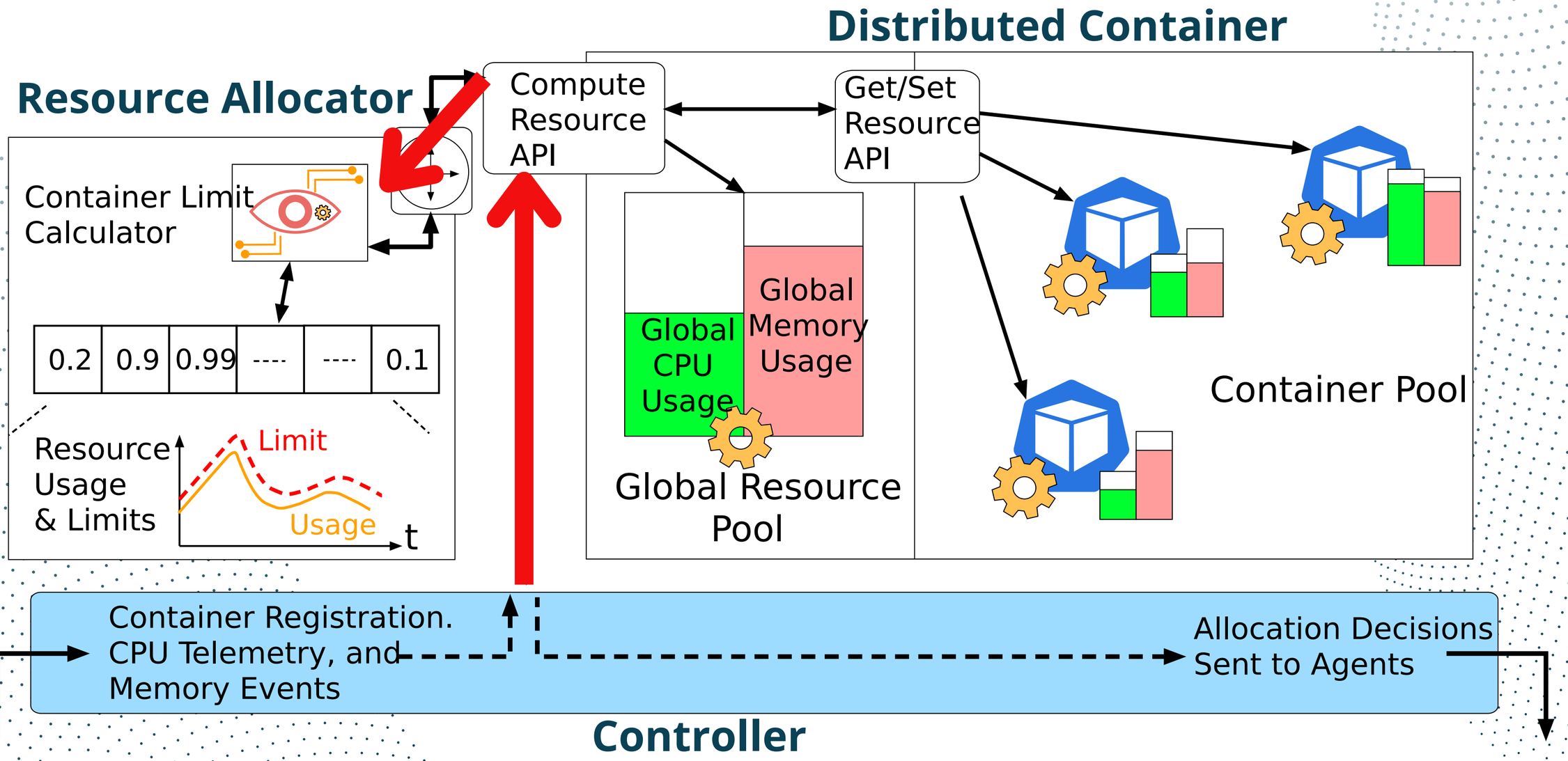


Escra - Control Node

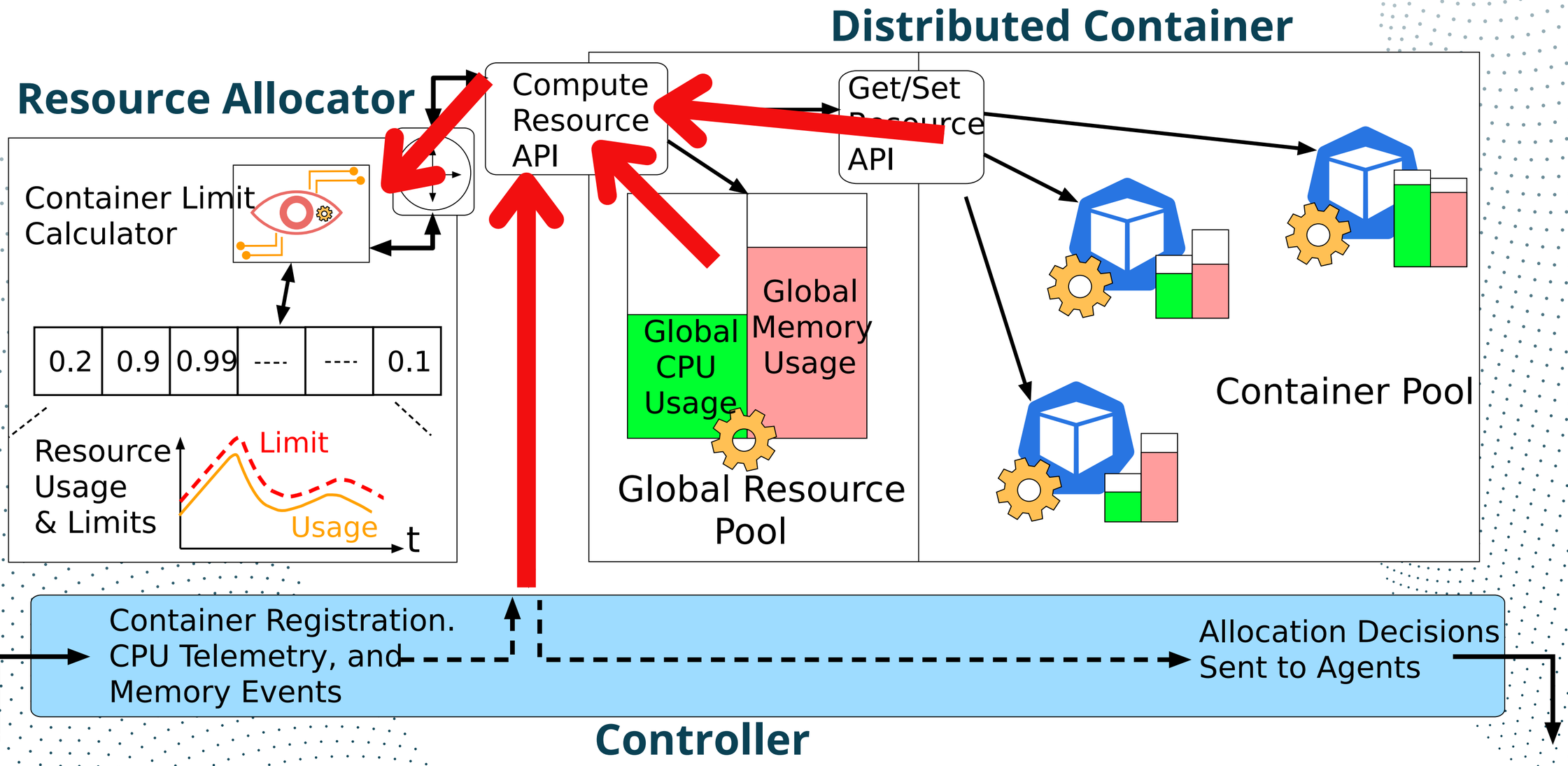


University of Colorado **Boulder**

Escra - Control Node

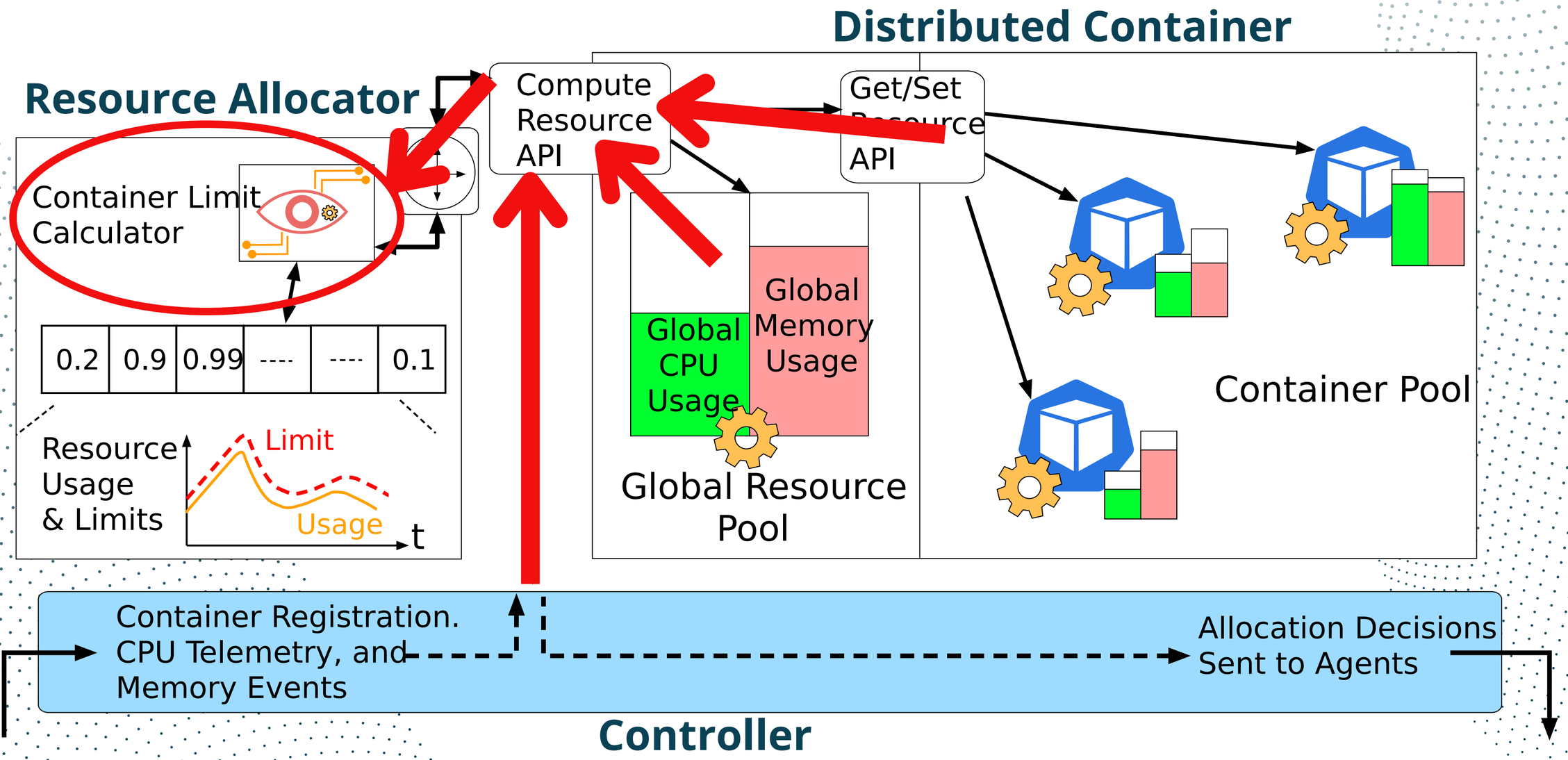


Escra - Control Node

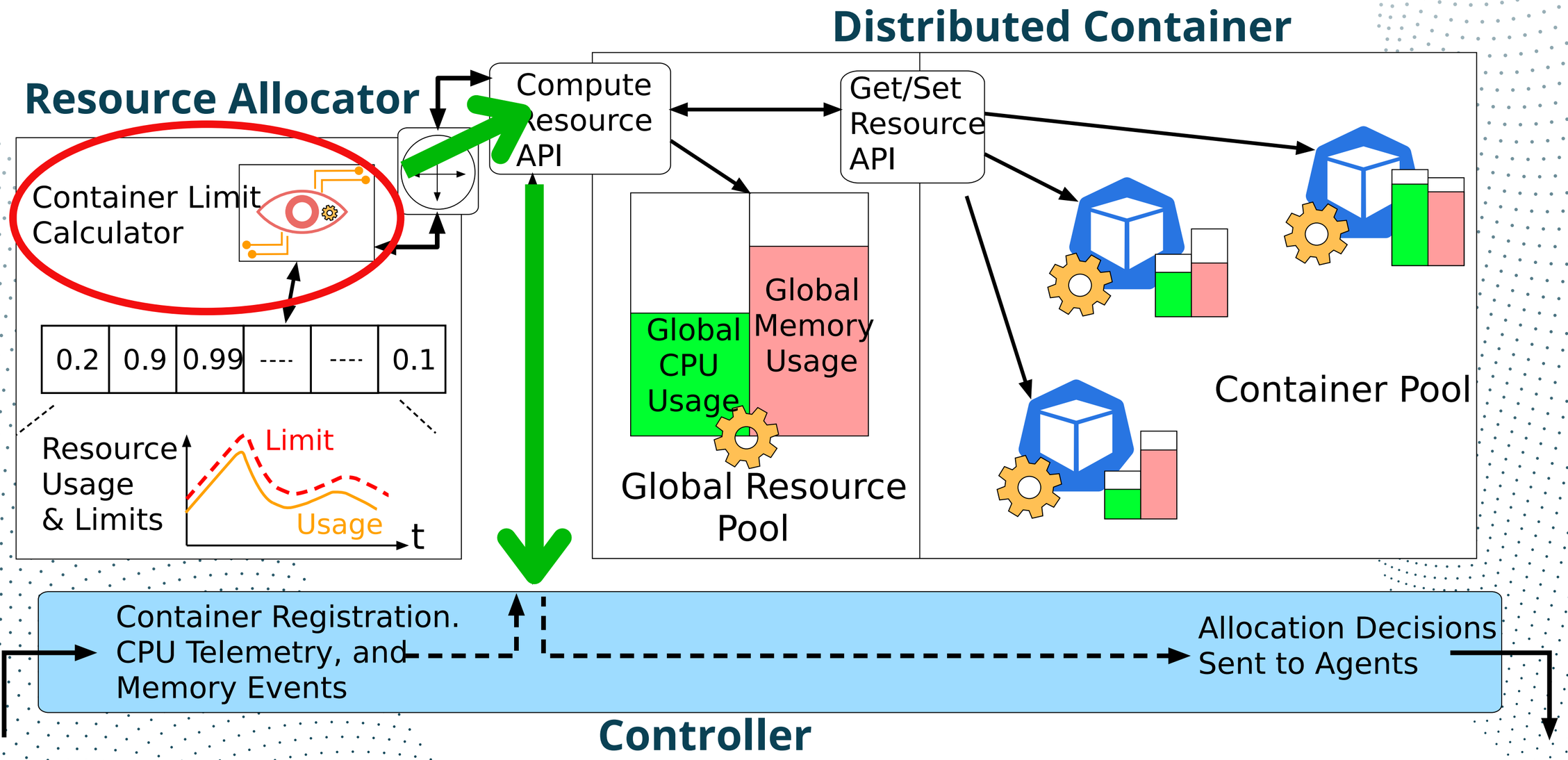


University of Colorado **Boulder**

Escra - Control Node

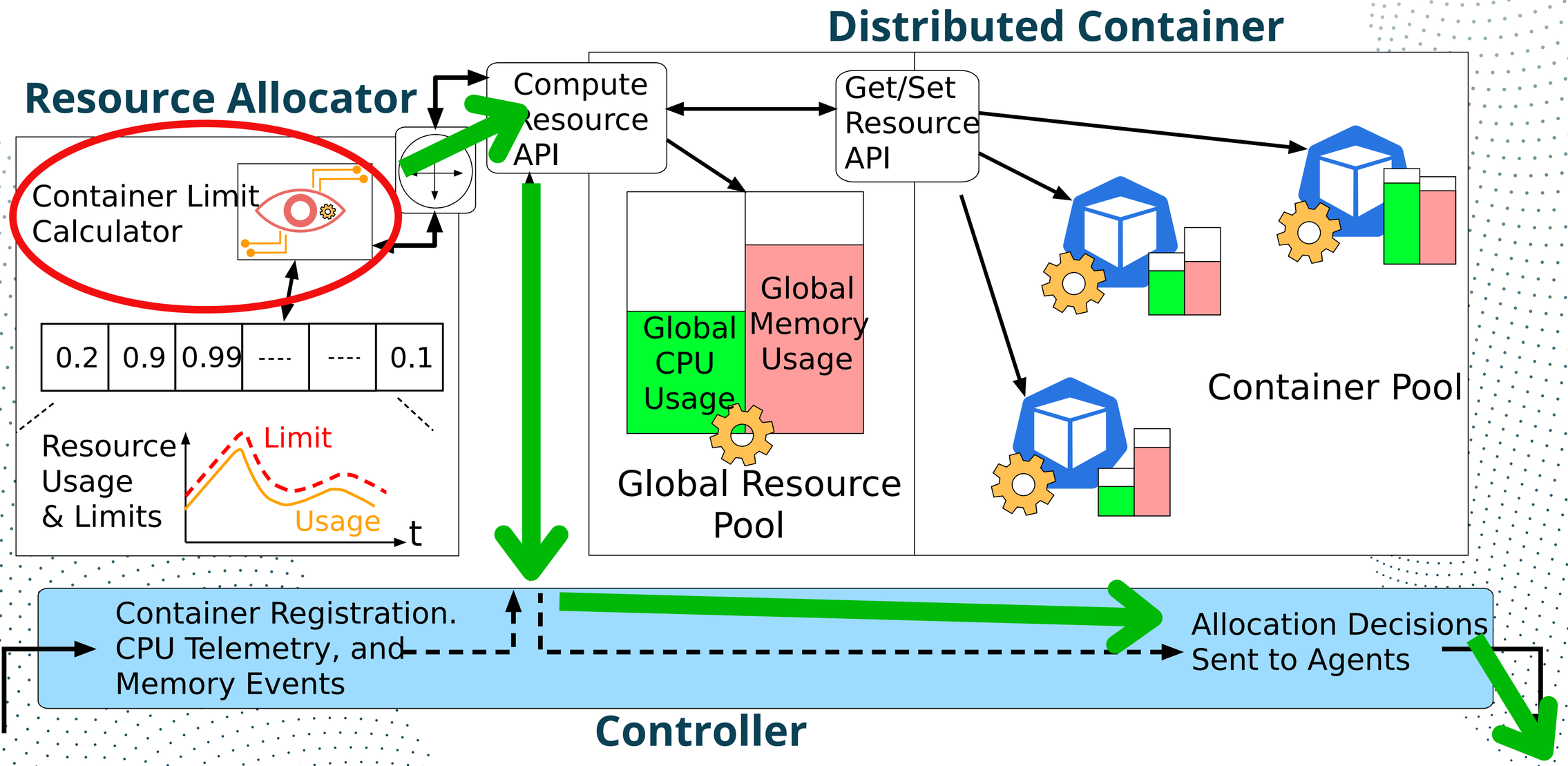


Escra - Control Node



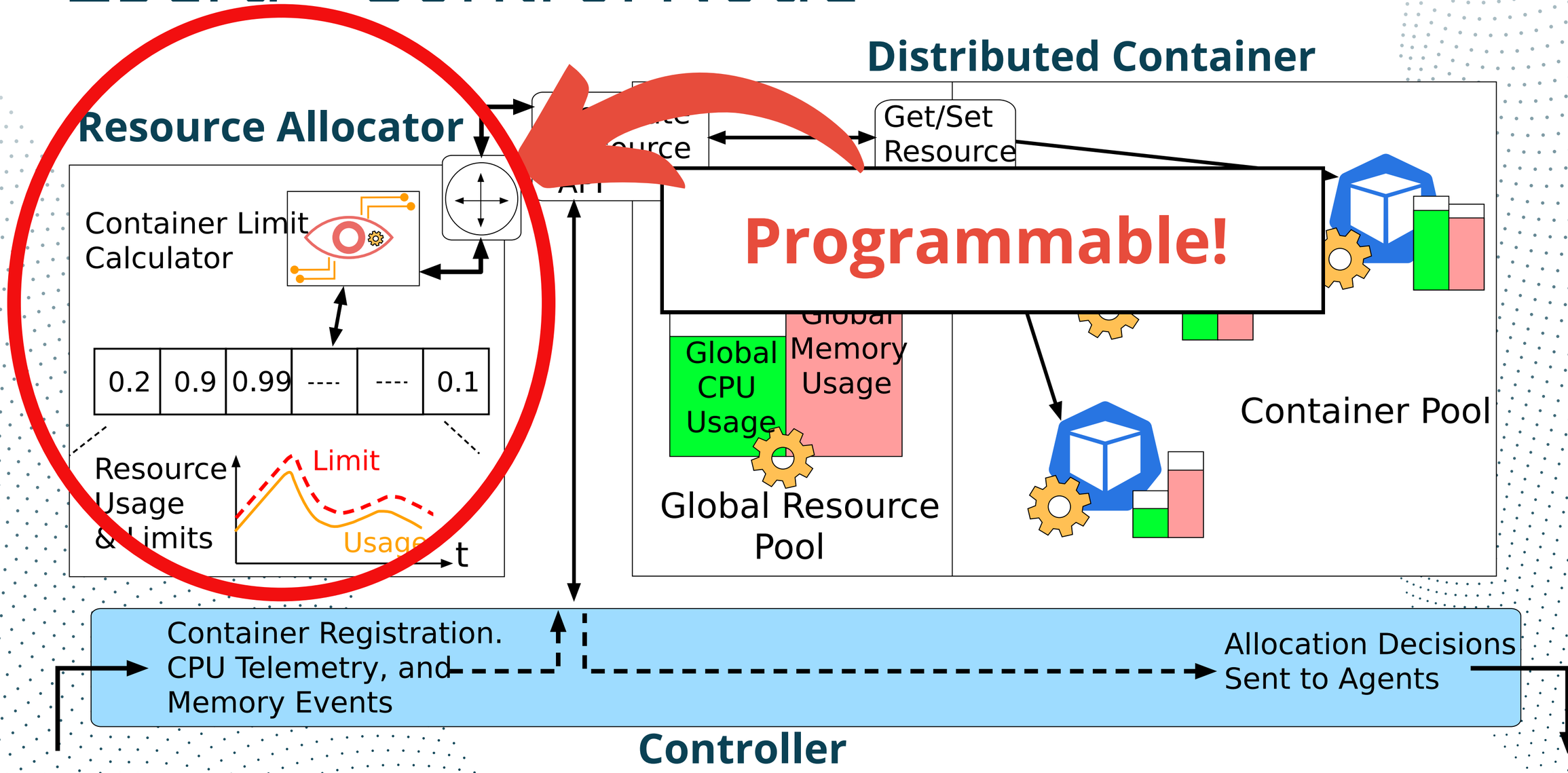
University of Colorado **Boulder**

Escra - Control Node



University of Colorado **Boulder**

Escra - Control Node



Evaluation - Microservices

Evaluation - Microservices

Metrics

Absolute Slack

Container Limit - Container Usage

Application Throughput

Successful requests/second

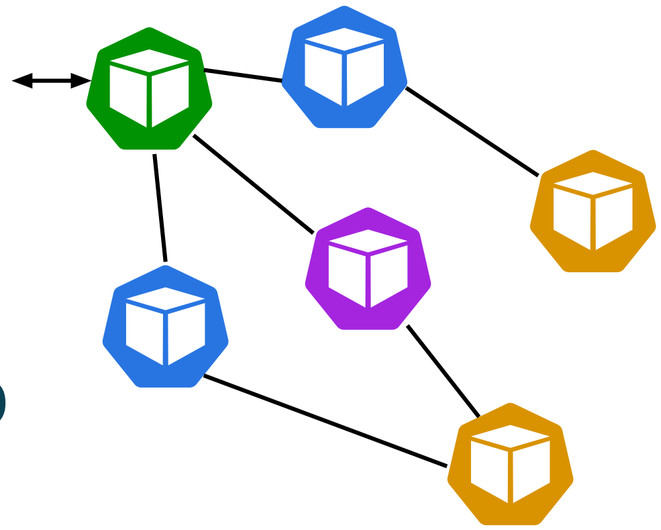
Application 99.9%ile Latency

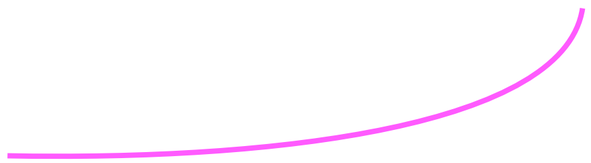
99.9%ile end-to-end latency

University of Colorado **Boulder**

Applications & Workloads

Applications	Workloads
• TrainTicket • TeaStore • HipsterShop • MediaMicroservice	• Alibaba • Burst • Exponential • Fixed

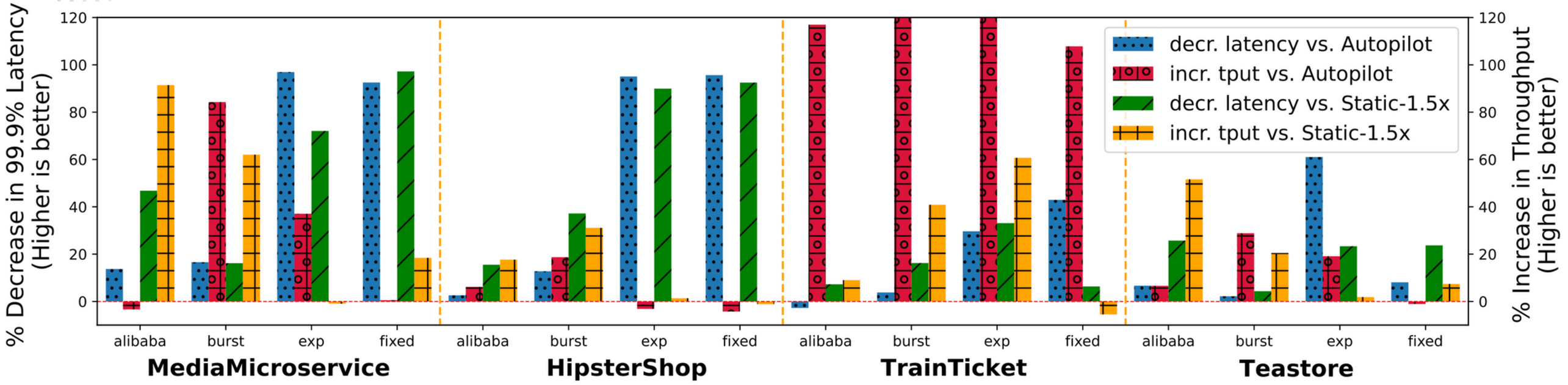




University of Colorado **Boulder**

Change in Latency and Throughput

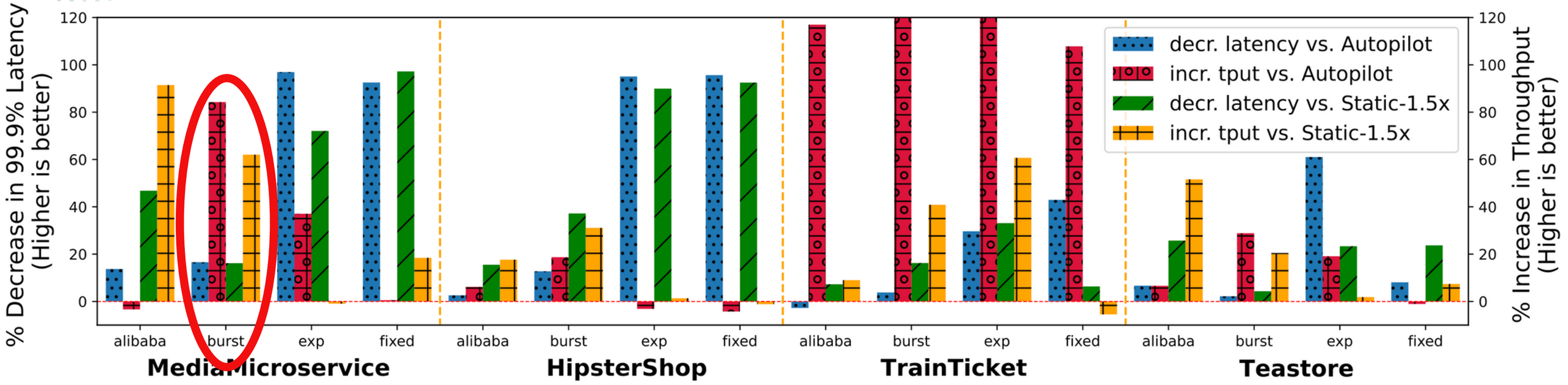
Microservices



University of Colorado **Boulder**

Change in Latency and Throughput

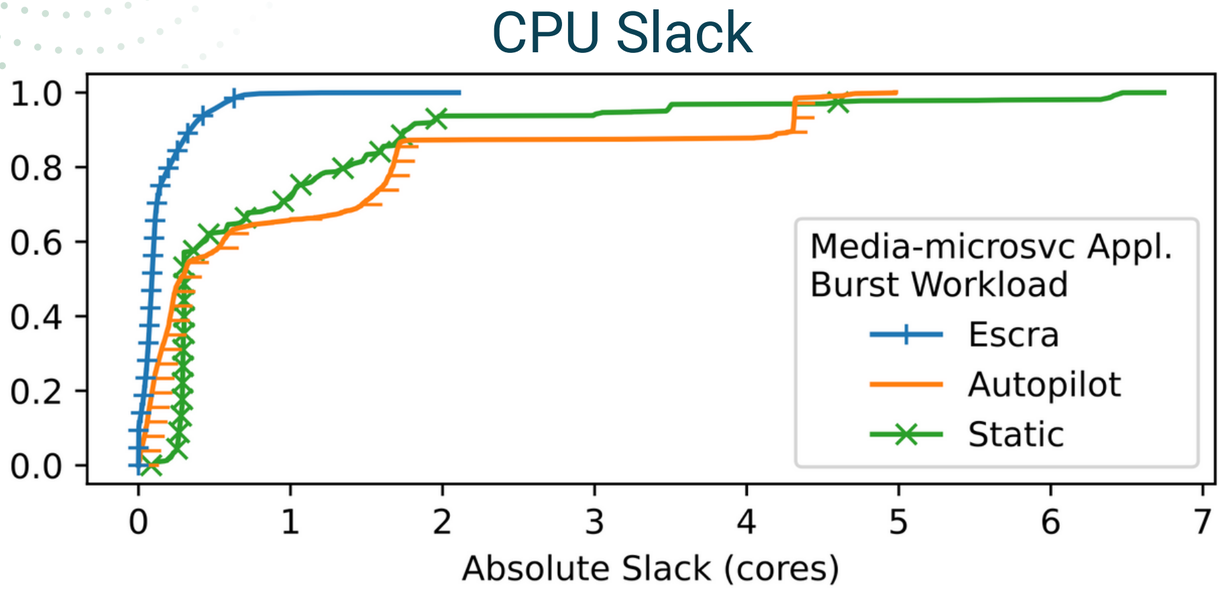
Microservices



University of Colorado **Boulder**

Escra vs. Autopilot

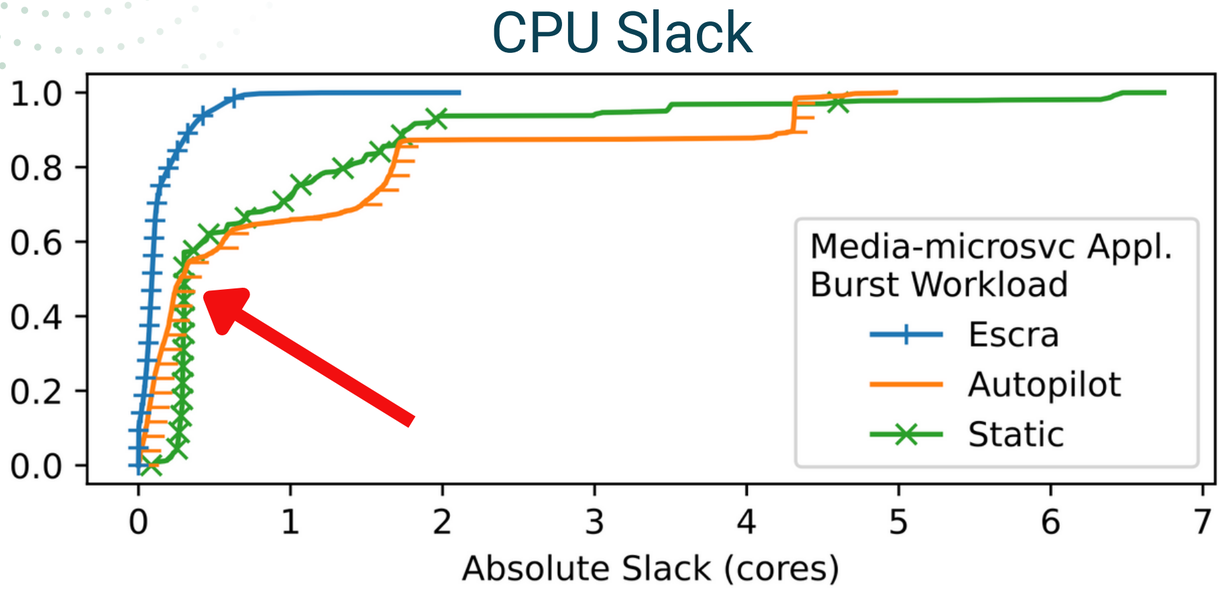
Microservices - MediaMicroservice/Burst Workload



University of Colorado **Boulder**

Escra vs. Autopilot

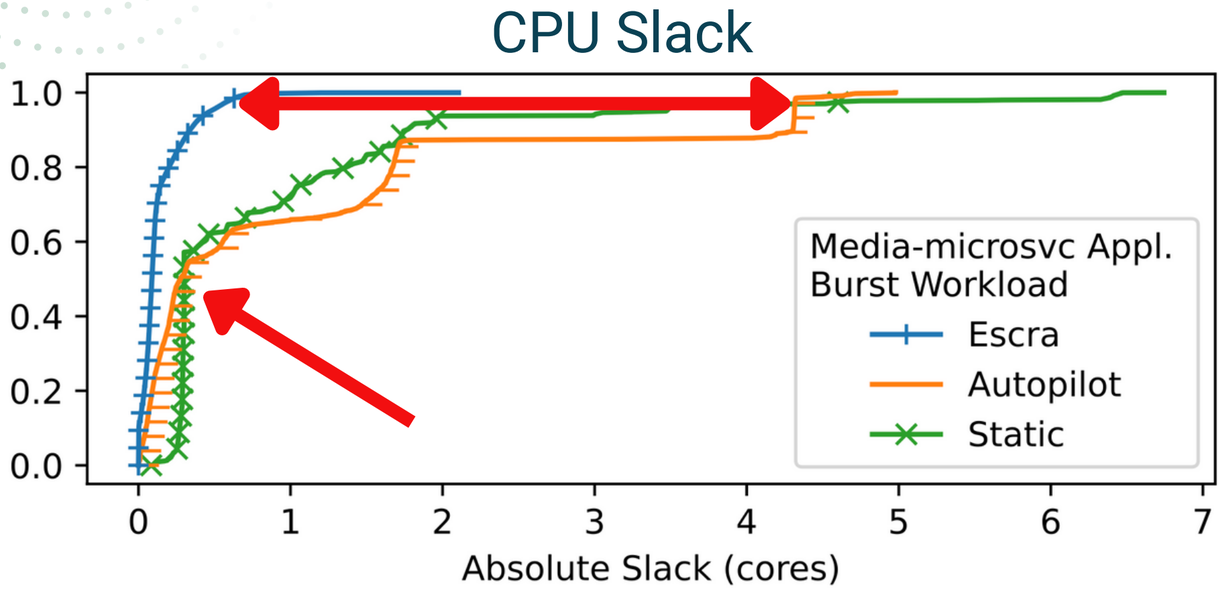
Microservices - MediaMicroservice/Burst Workload



University of Colorado **Boulder**

Escra vs. Autopilot

Microservices - MediaMicroservice/Burst Workload

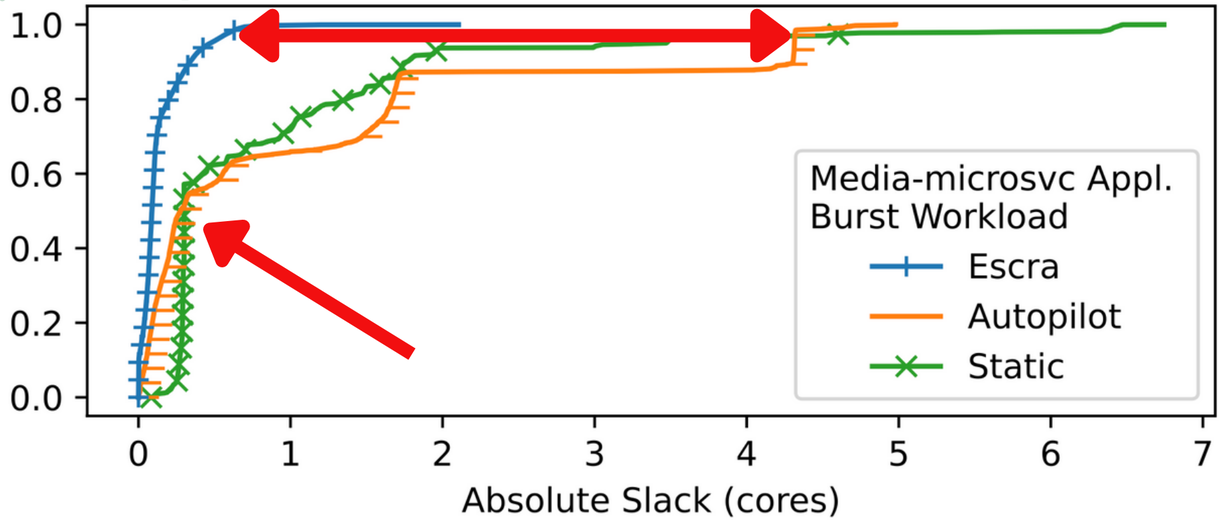


University of Colorado **Boulder**

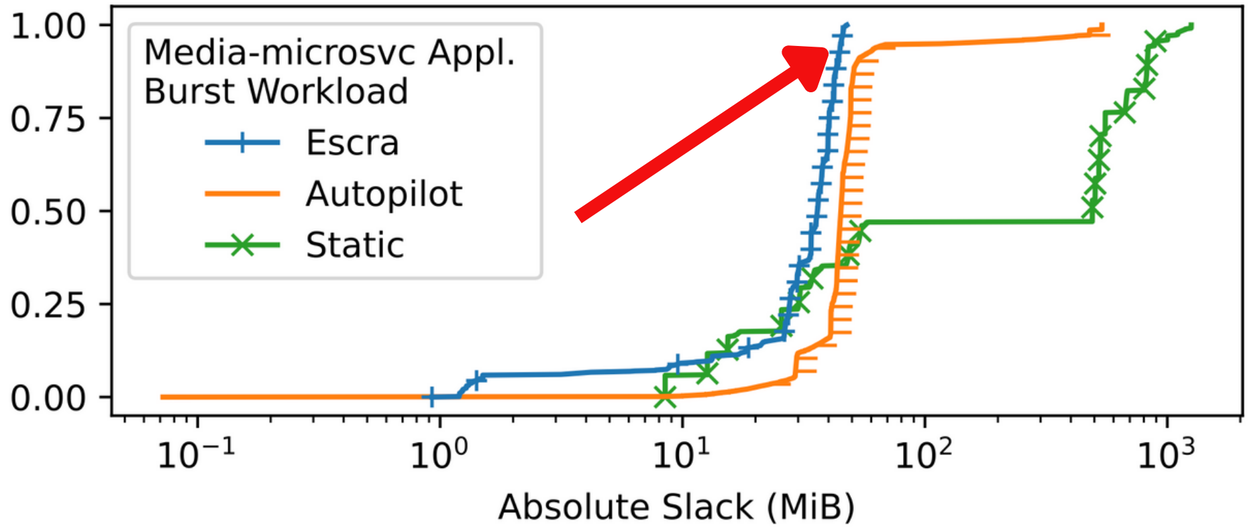
Escra vs. Autopilot

Microservices - MediaMicroservice/Burst Workload

CPU Slack



Memory Slack

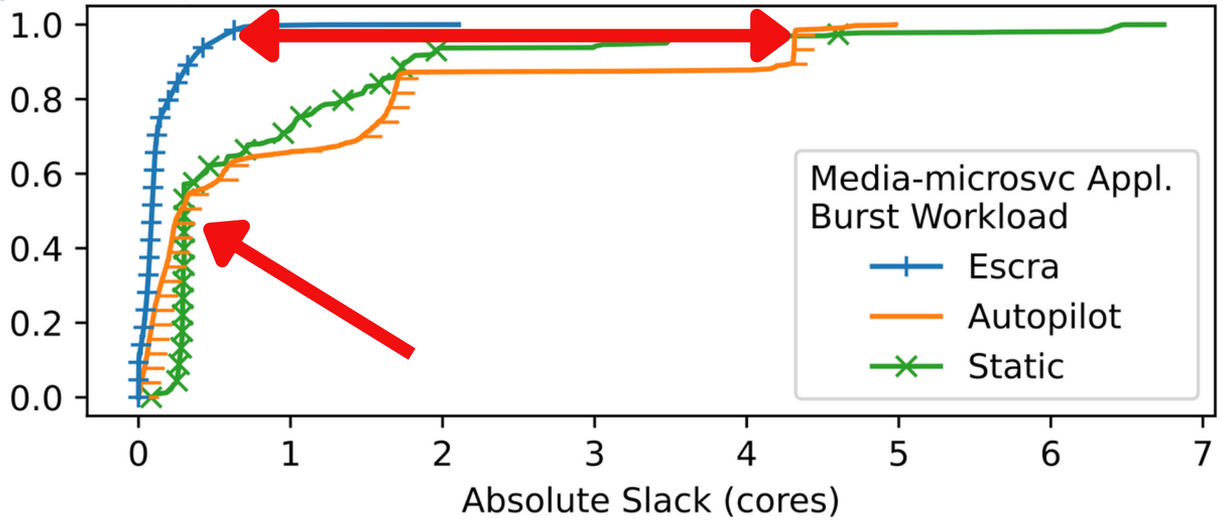


University of Colorado **Boulder**

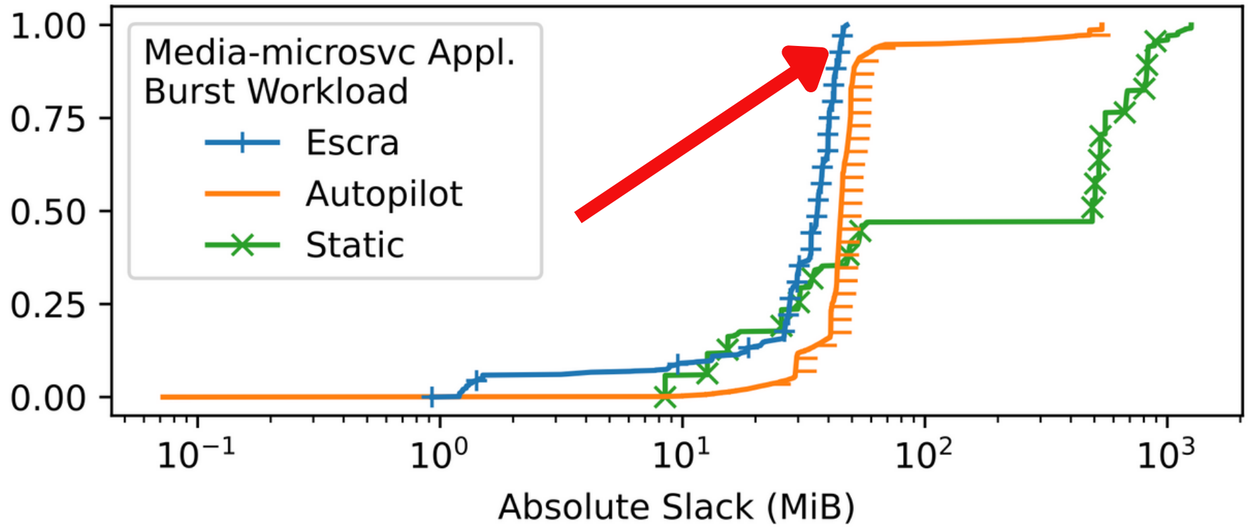
Escra vs. Autopilot

Microservices - MediaMicroservice/Burst Workload

CPU Slack



Memory Slack



- **Latency Decrease: 16.6%**
- **Throughput Increase: 84.3%**

Evaluation - Serverless

Evaluation - Serverless

Metrics

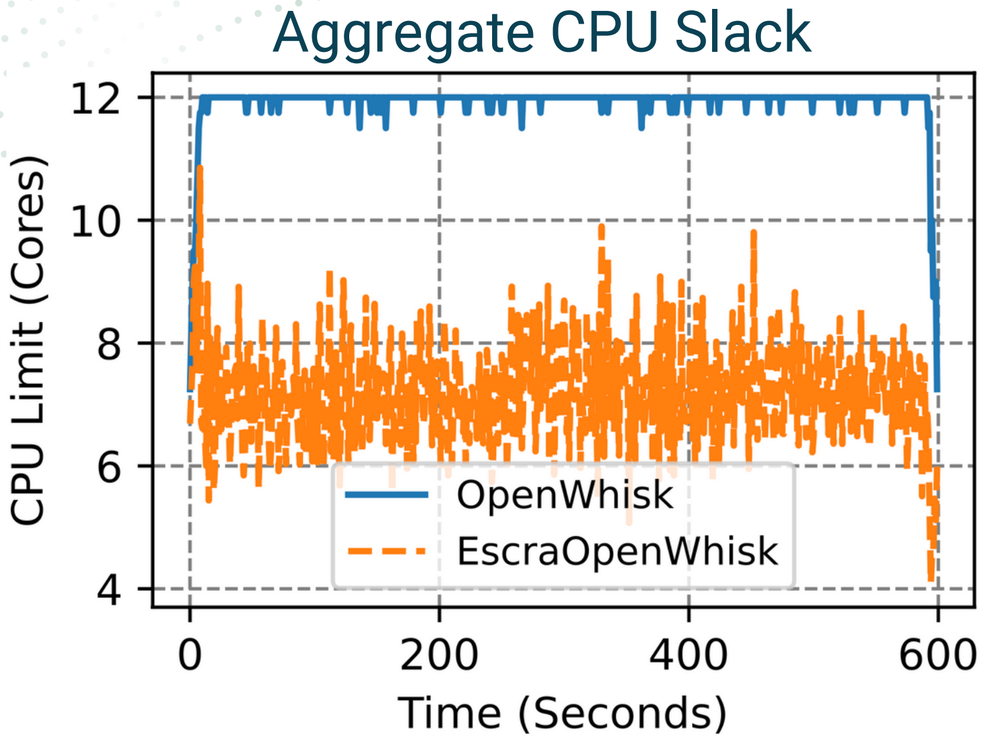
Aggregate Limits
Σcontainer limits - Σcontainer usages

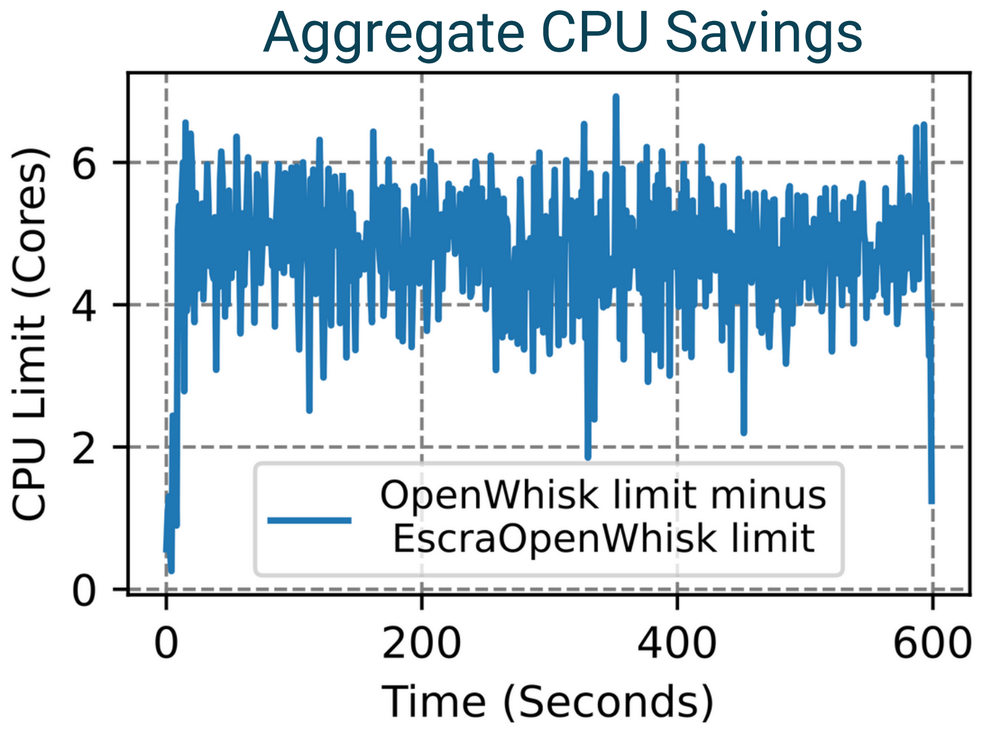
Application Latency
End-to-end latency per request or job

University of Colorado **Boulder**

ImageProcess

Serverless



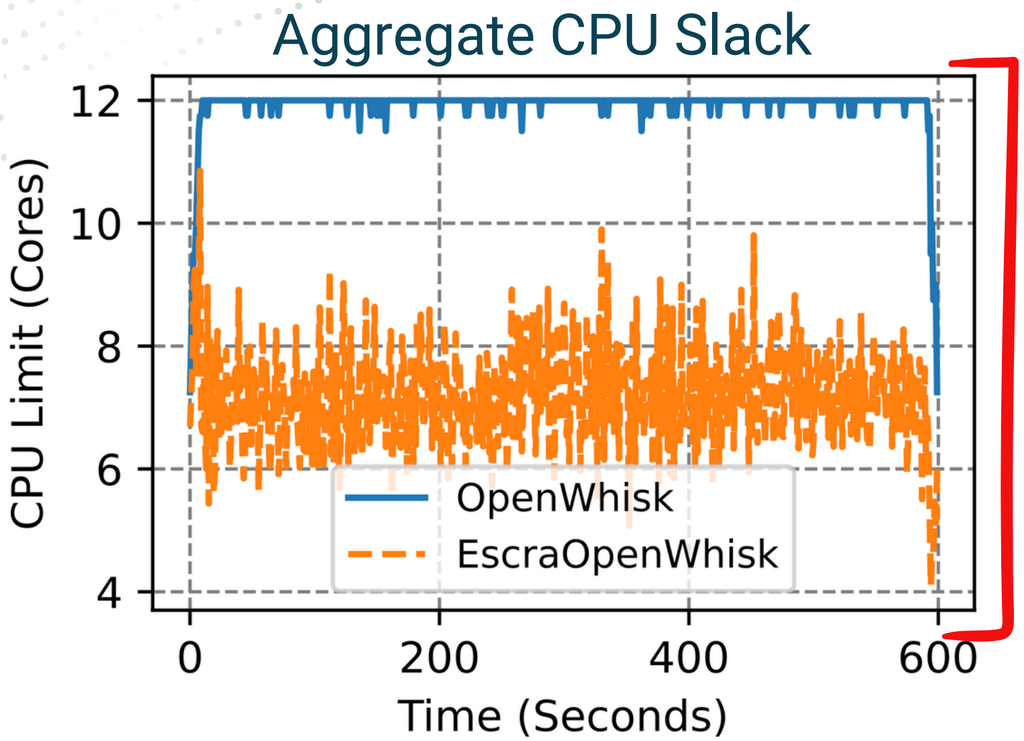


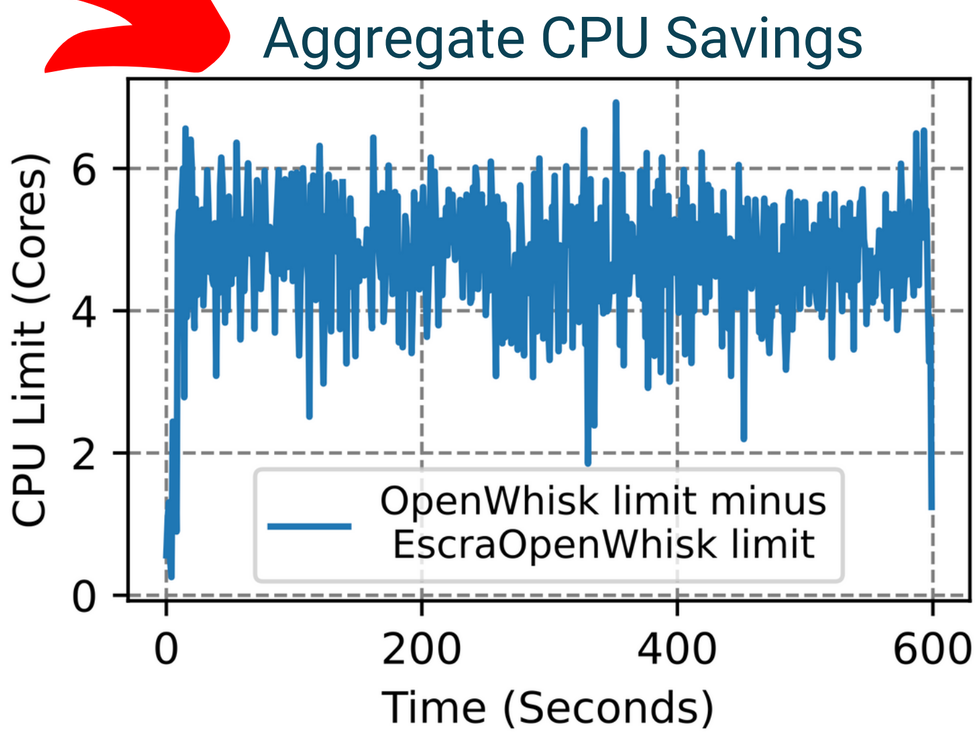
University of Colorado **Boulder**

ImageProcess

Serverless

Difference!



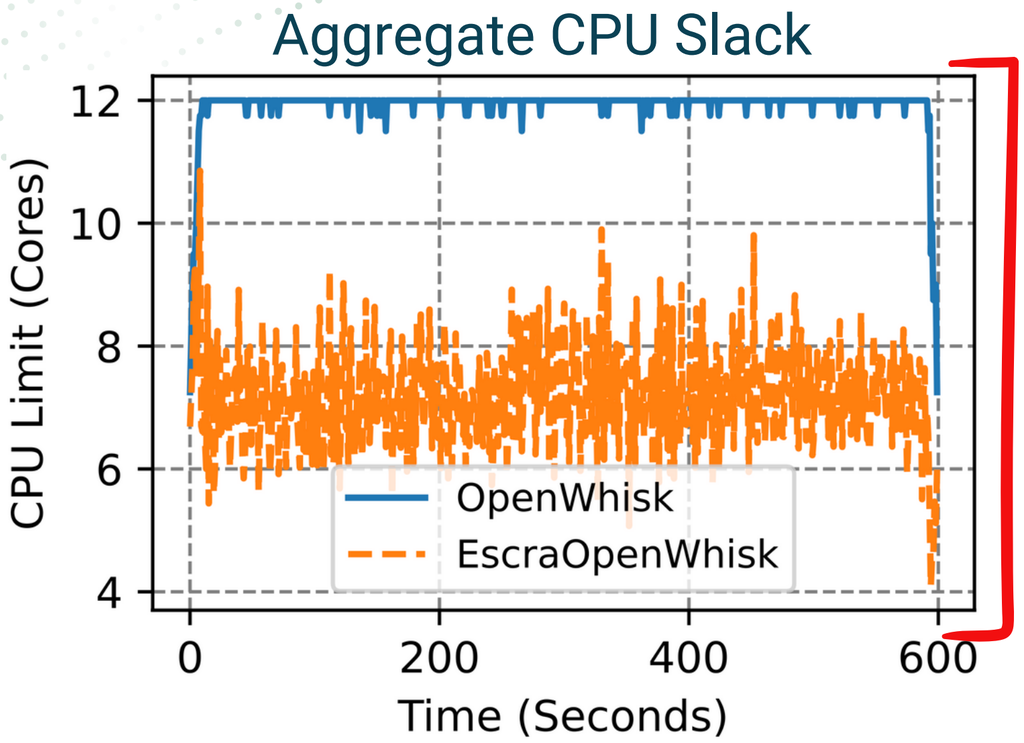


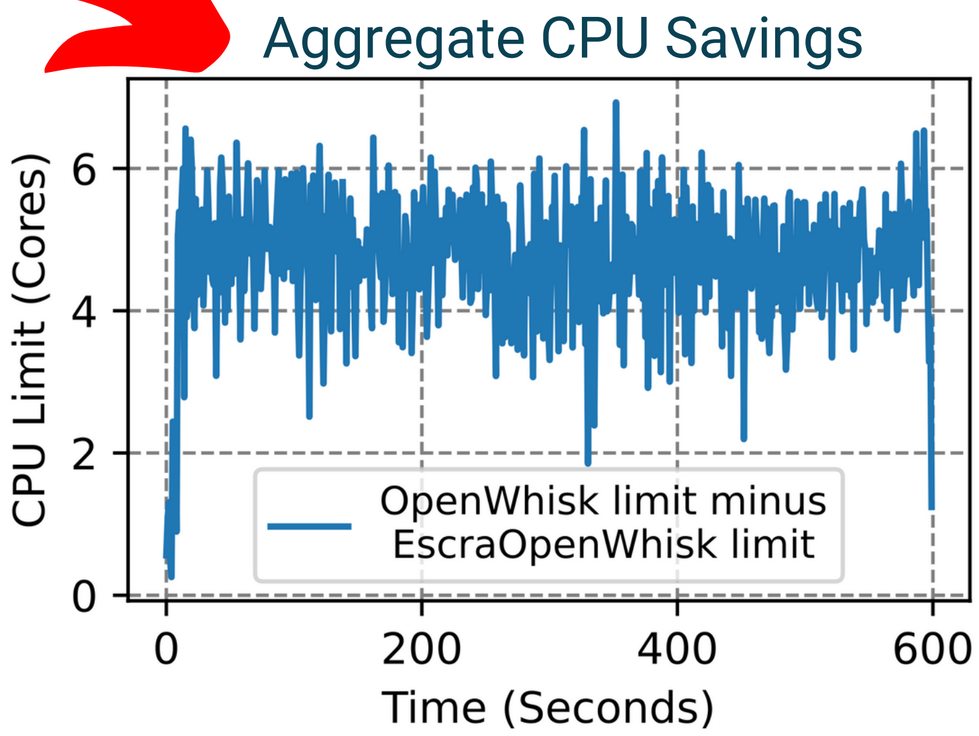
University of Colorado **Boulder**

ImageProcess

Serverless

Difference!



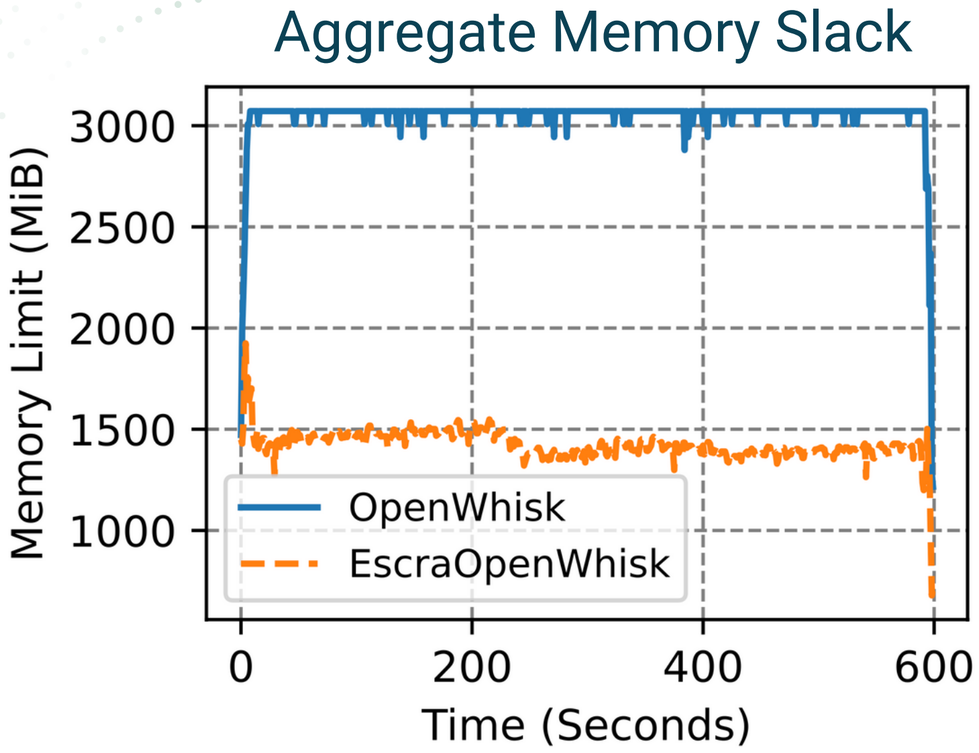


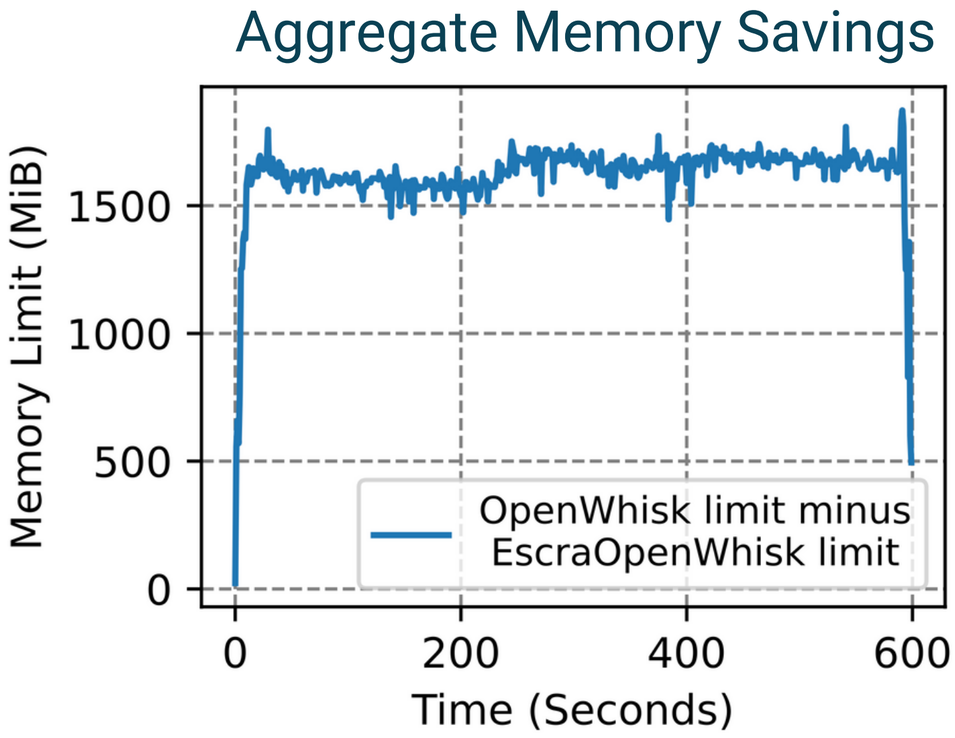
- **Escra + Openwhisk avg vCPU limit: 7 vCPU**
- **Openwhisk avg vCPU limit: 12 vCPU**

University of Colorado **Boulder**

ImageProcess

Serverless



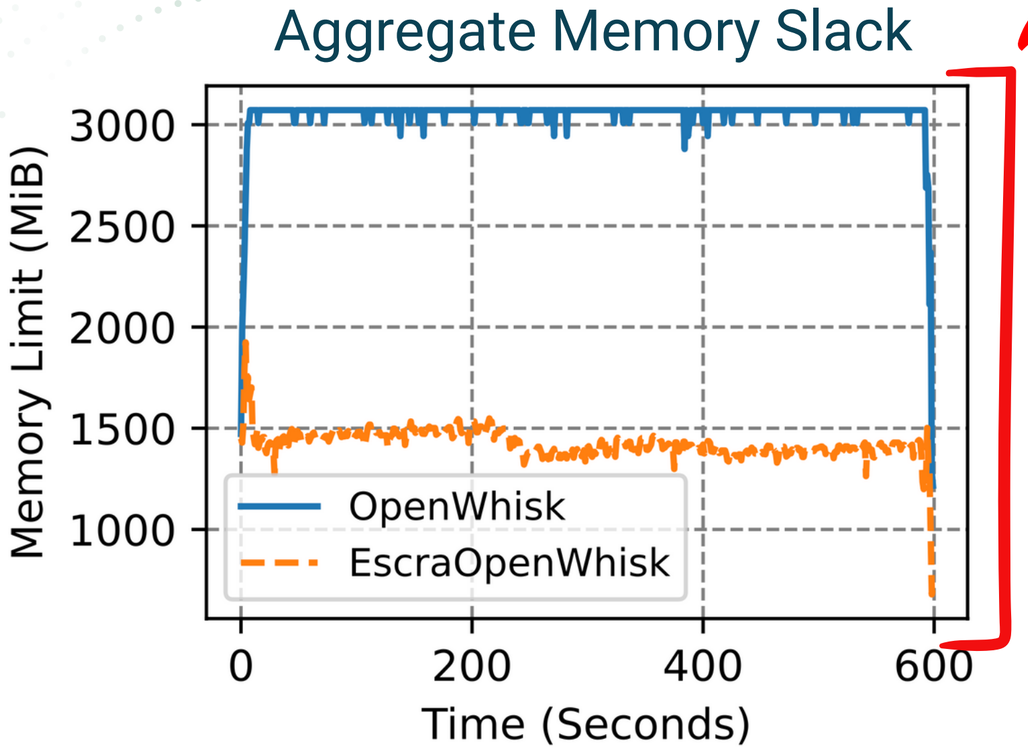


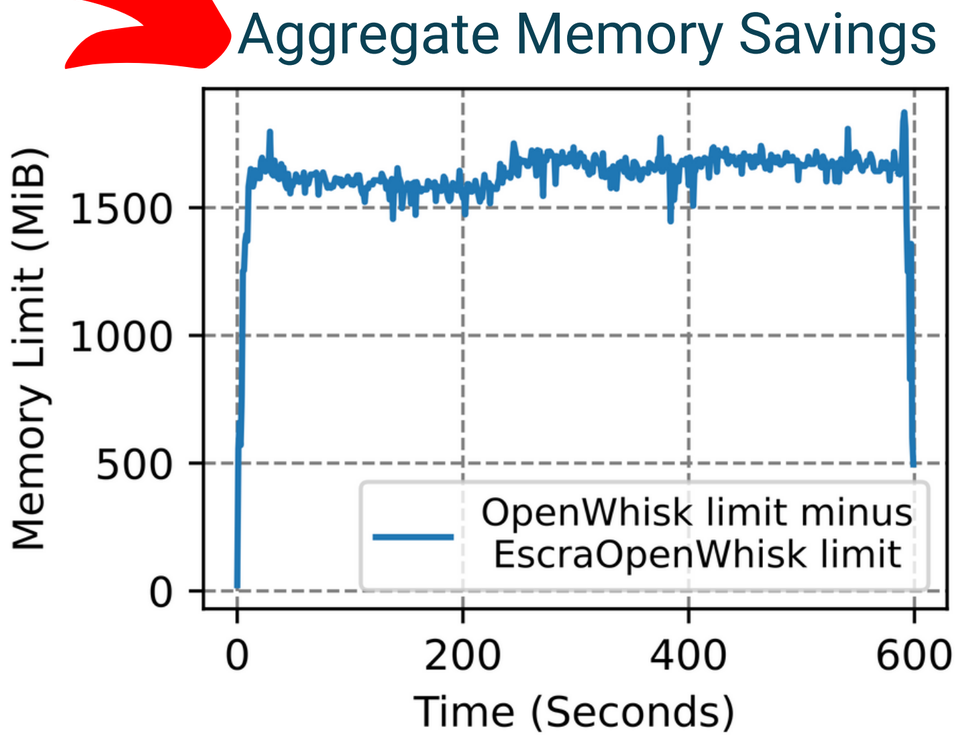
University of Colorado **Boulder**

ImageProcess

Serverless

Difference!



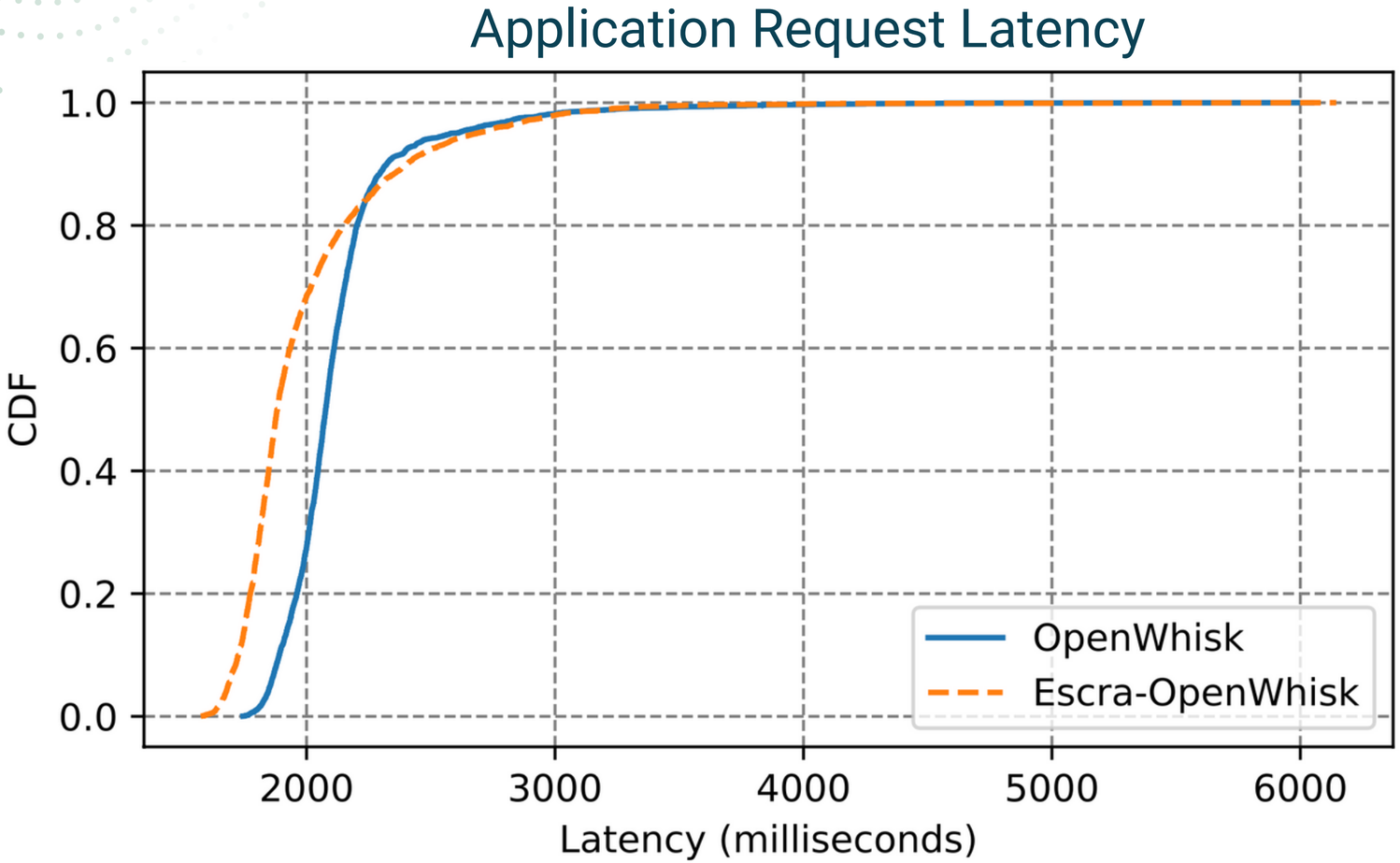


Avg. memory savings: 1550 MiB

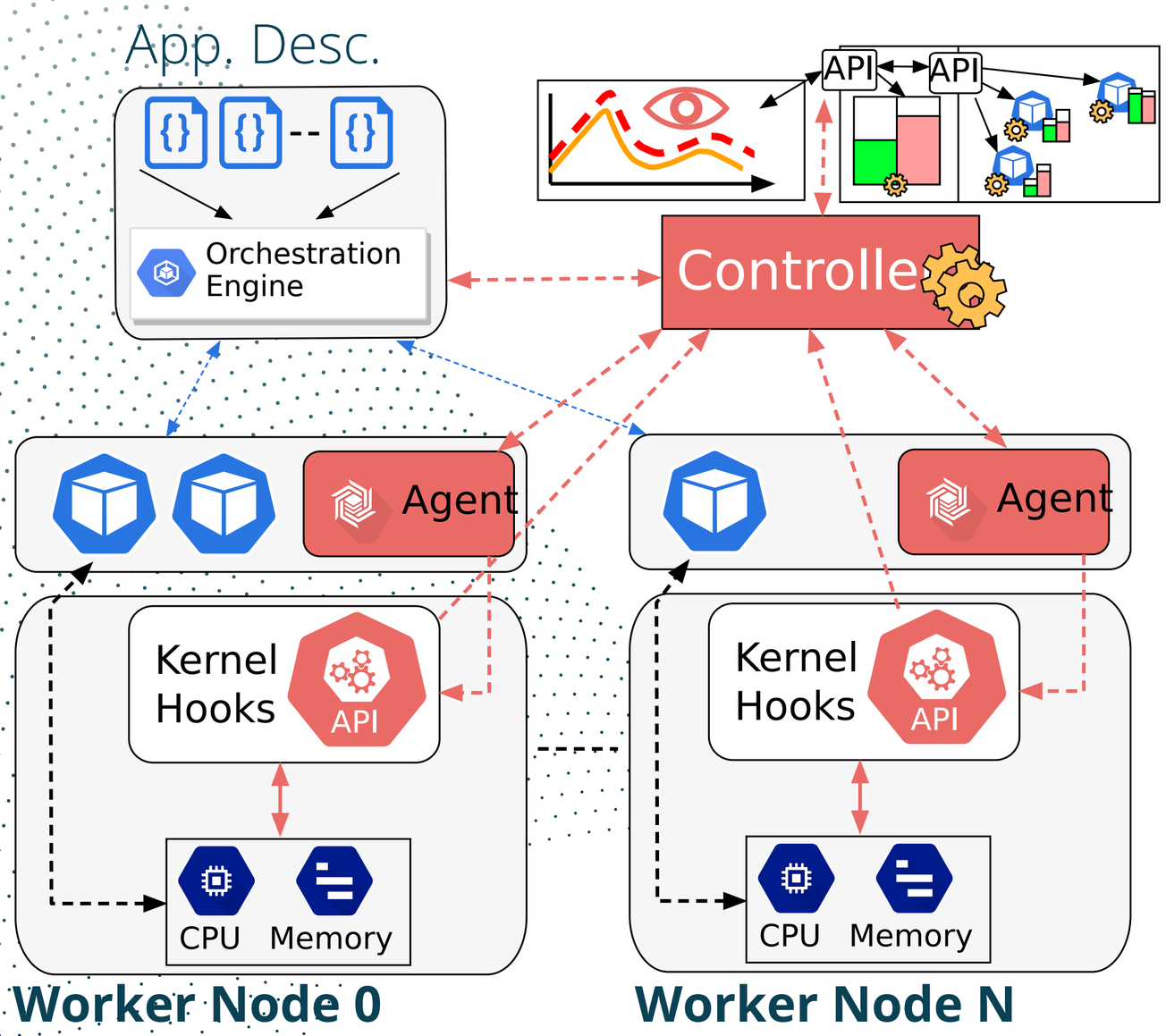
University of Colorado **Boulder**

ImageProcess

Serverless



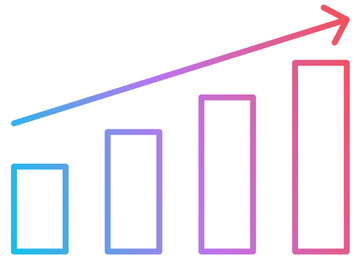
- 80% of the time, Escra + Openwhisk sees a strong performance gain
- 99th%-ile latency remains similar for both



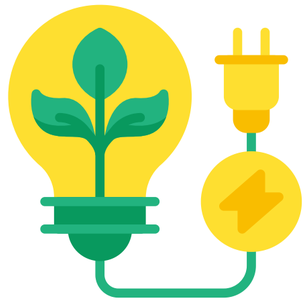
Recap

- High efficiency and high performance!
- Outperforms state of the art!
- Significant resource saving in serverless deployments

Performance



Efficiency



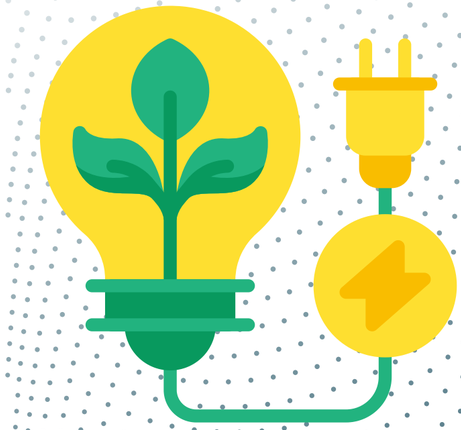
Conclusion

- Can we create and then expose a flexible underlying platform to developers that allows them to optimize their applications':

Security

Performance

Efficiency

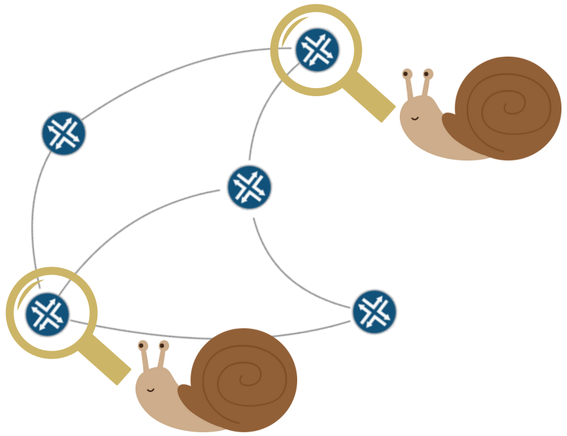


Conclusion

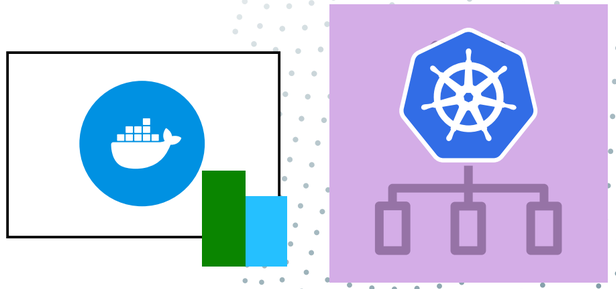
1. Identify the shortcomings and rigidity of the underlying systems that control and manage:

Secure Hardware

Network Monitoring



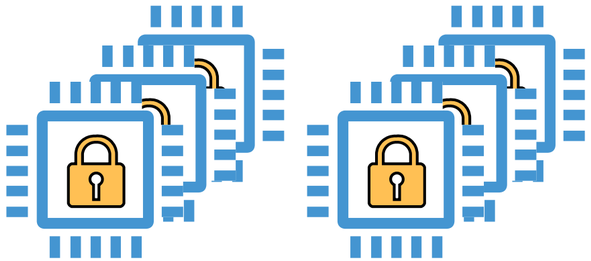
Compute Resources



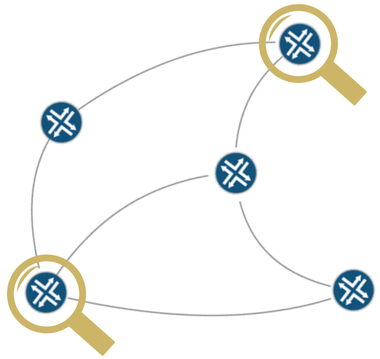
Conclusion

2) Build new, programmable platforms and systems that allow users to design and implement application-specific:

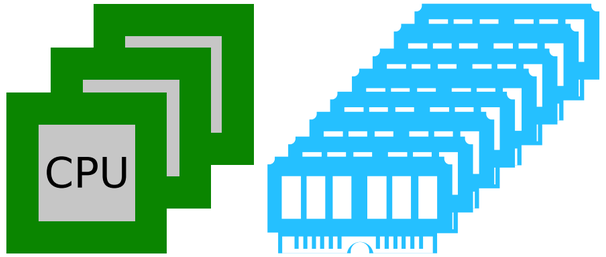
Secure Hardware Features



Network Monitoring Applications



Compute Resource Allocation Decisions



Publications

University of Colorado **Boulder**

1) G. Cusack, M. Nazari, S. Goodarzy, E. Hunhoff, P. Oberai, E. Keller, E. Rozner, and R. Han.
Escra: Event-driven, Sub-second Container Resource Allocation
42nd IEEE International Conference on Distributed Computing Systems (ICDCS '22), 2022

2) O. Michel, J. Sonchack, G. Cusack, M. Nazari, E. Keller and J. M. Smith.
Software Packet-Level Network Analytics at Cloud Scale
IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 597-610, March 2021

3) M. Hashemi, G. Cusack, E. Keller.
Towards Evaluation of NIDSs in Adversarial Setting
ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks (Big-DAMA), 2019

4) G. Cusack, M. Nazari, S. Goodarzy, P. Oberai, E. Rozner, E. Keller, R. Han.
Efficient Microservices with Elastic Containers
CoNEXT '19 Companion (Poster)

5) A. Coughlin, G. Cusack, J. Wampler, E. Keller, E. Wustrow.
Breaking The Trust Dependence on Third Party Processes For Reconfigurable Secure Hardware
International Symposium on Field-Programmable Gate Arrays, 2019

6) M. Hashemi, G. Cusack, E. Keller.
Stochastic Substitute Training: A Gray-box Approach to Craft Adversarial Examples Against Gradient Obfuscation Defenses
ACM Workshop on Artificial Intelligence and Security (AISec), 2018

7) G. Cusack, O. Michel, E. Keller.
Machine Learning-Based Detection of Ransomware Using SDN
Workshop on SDN-NFV Security, 2018

8) G. Cusack, O. Michel, & E. Keller.
Machine Learning-Based Fingerprinting of Network Traffic Using Programmable Forwarding Engines
Network and Distributed Systems Symposium, 2018 (Poster)

University of Colorado **Boulder**

Discussion/Questions?


github.com/gregcusack


@GregoryCusack


gregory.cusack@colorado.edu


linkedin.com/in/gregorycusack

Dissertation Defense
December 14, 2022
Boulder, Colorado

Huge Thank You to:

Eric Keller,
Maziyar Nazari,
Sepideh Goodarzy,
Erika Hunhoff,
Prerit Oberai,
Eric Rozner,
Richard Han,
Eric Wustrow,
Oliver Michel,
Jack Wampler,
Aimee Coughlin,
Mohammad Hashemi,
Tamara Silbergleit Lehman,
Joe Izraelevitz,
and more!