Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

U LISBOA | UNIVERSIDADE DE LISBOA

UNIVERSITY OF OXFORD

Checkpoint Coordination for Distributed Containerized Applications

Radostin Stoyanov, Viktória Spišaková, Lukáš Hejtmánek, Adrian Reber

Supervisor: Prof. Rodrigo Bruno, Prof. Wes Armour

LINUX PLUMBERS CONFERENCE Vienna, Austria / Sept. 18-20, 2024

MUNI Masaryk University

Red Hat

Kubernetes Environment

Multi-user multi-purpose Kubernetes clusters for academic users in Czechia:

- 3456 CPU, 13 TiB memory, 39 GPU (A10, A40, A100, H100, L4)
- Containers provide reproducibility – crucial for research
- Cloud provides elasticity – crucial for indecisive users and static apps

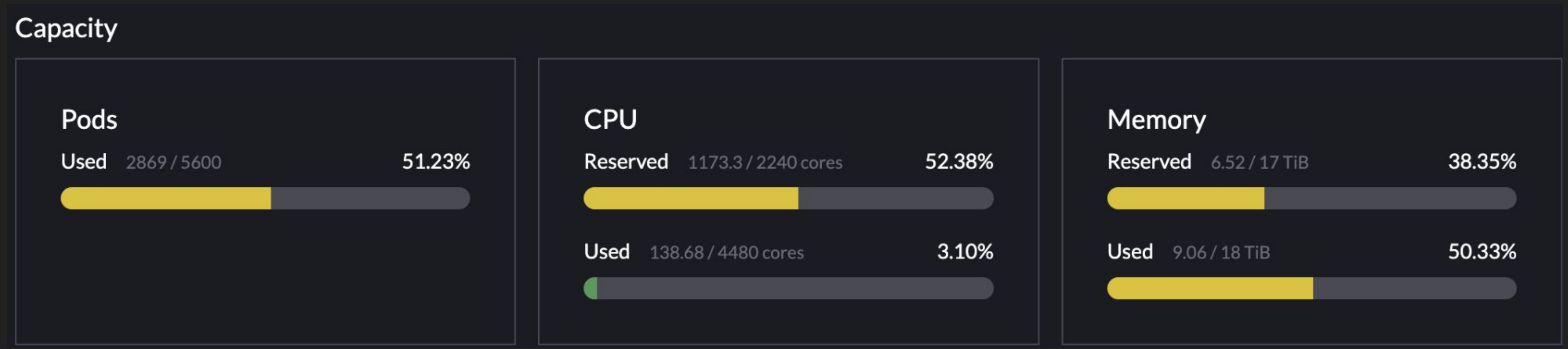
Problem:

- High variance of workload types => high variance in resource requests over time => low cluster resource utilization => ineffective

Workload Types

- Static applications (webs), runners
- HPC workflows – bioinformatics pipelines (Nextflow[1])
- Interactive workloads
  - JupyterHubs (interactive web-based computing)
  - Virtual desktops (require multiple containers to run - distributed checkpoint)
- Resource intensive computations (CryoSparc requires 80GB GPUs)
- Long-running computations (AlphaFold up to 6 months!)

[1] Viktória Spišaková, et. al., *Nextflow in Bioinformatics: Executors Performance Comparison Using Genomics Data*, Future Generation Computer Systems, 2023



Capacity

Pods

Used 2869 / 5600 51.23%

CPU

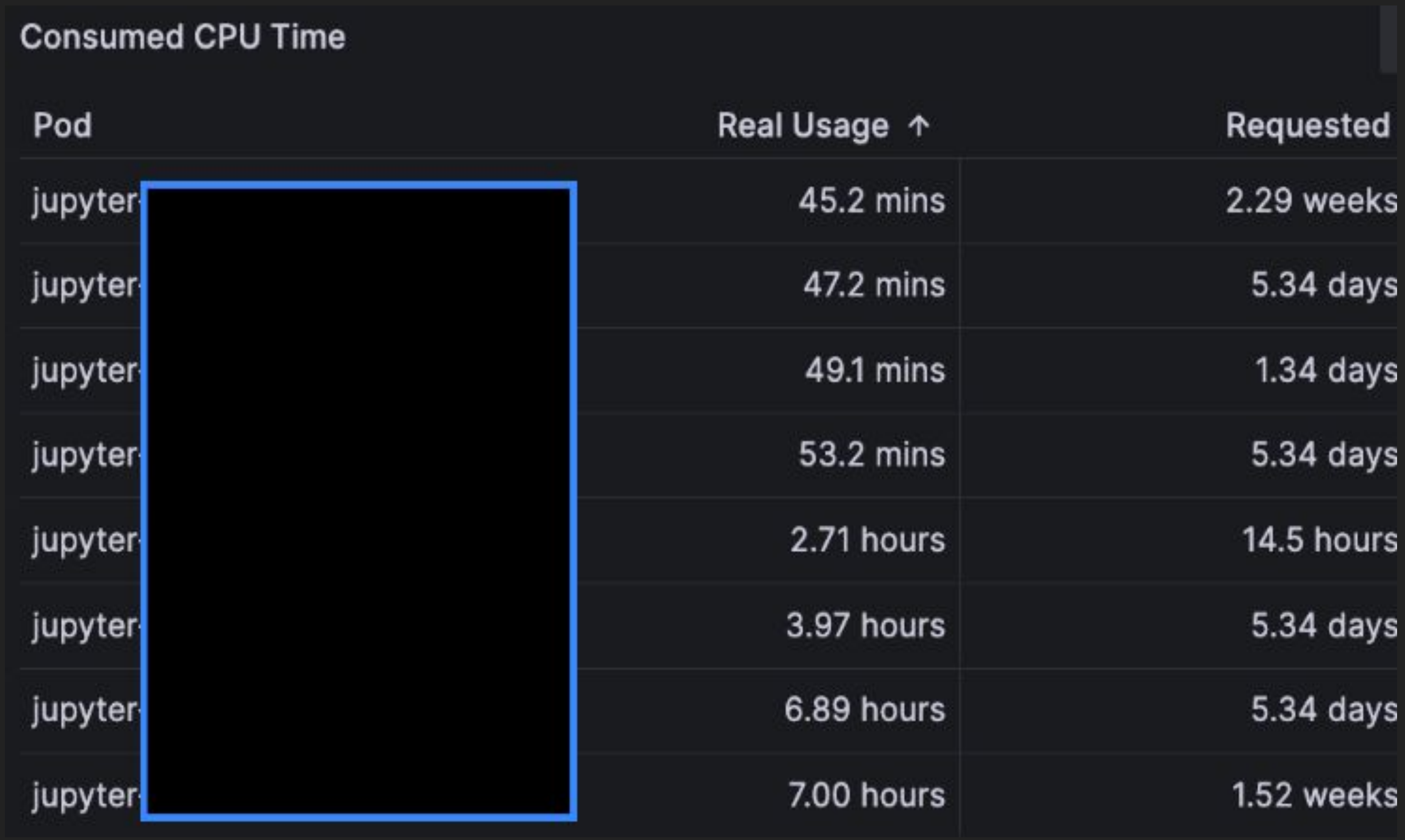
Reserved 1173.3 / 2240 cores 52.38%

Used 138.68 / 4480 cores 3.10%

Memory

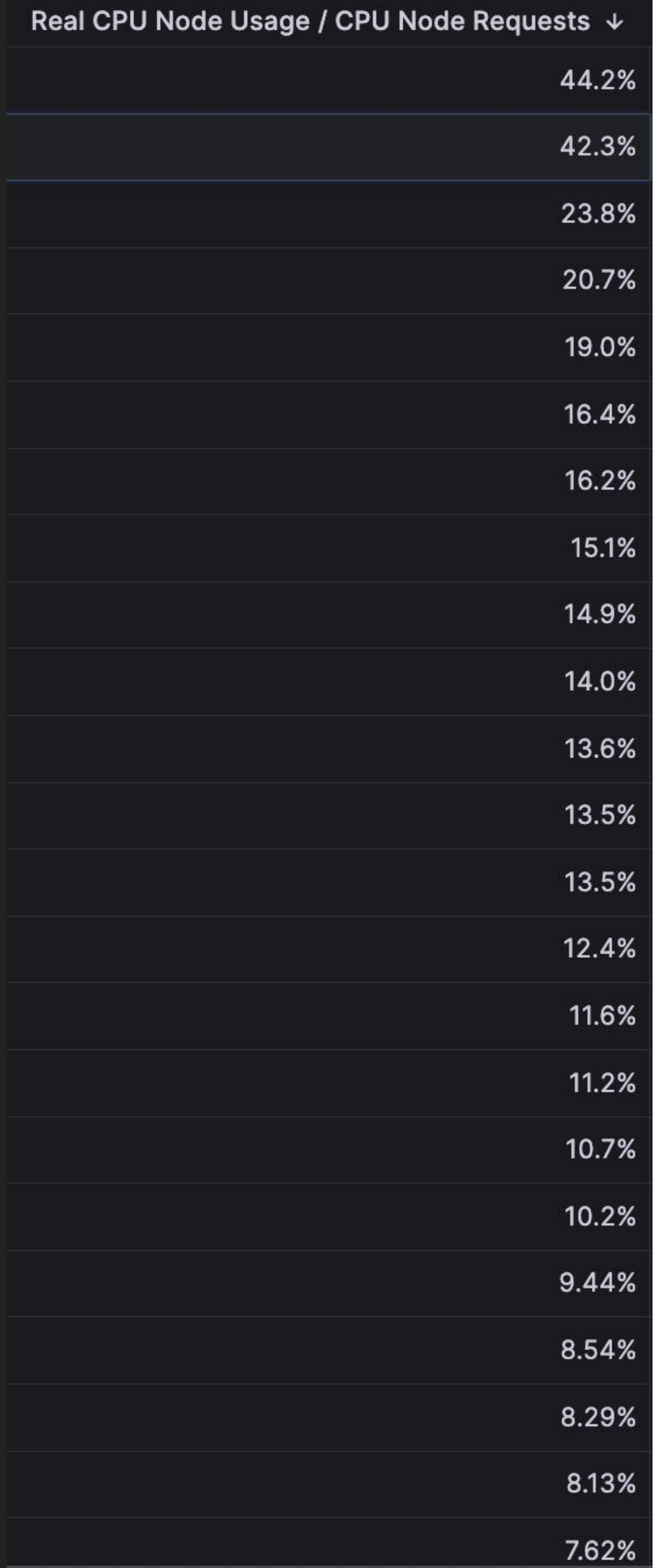
Reserved 6.52 / 17 TiB 38.35%

Used 9.06 / 18 TiB 50.33%



Consumed CPU Time

Pod	Real Usage ↑	Requested
jupyter[redacted]	45.2 mins	2.29 weeks
jupyter[redacted]	47.2 mins	5.34 days
jupyter[redacted]	49.1 mins	1.34 days
jupyter[redacted]	53.2 mins	5.34 days
jupyter[redacted]	2.71 hours	14.5 hours
jupyter[redacted]	3.97 hours	5.34 days
jupyter[redacted]	6.89 hours	5.34 days
jupyter[redacted]	7.00 hours	1.52 weeks



Real CPU Node Usage / CPU Node Requests ↓
44.2%
42.3%
23.8%
20.7%
19.0%
16.4%
16.2%
15.1%
14.9%
14.0%
13.6%
13.5%
13.5%
12.4%
11.6%
11.2%
10.7%
10.2%
9.44%
8.54%
8.29%
8.13%
7.62%

Single Container Checkpoint/Restore UC

JupyterHub (on Kubernetes):

- Manages multiple **single** instances of web-based Jupyter Notebooks
- Kubernetes checkpoint/restore works with CPU notebooks
    - GPU notebooks require small bug fixes
- Ch/R integration with JupyterHub could improve resource management
    - Better – direct Ch/R integration into Kubernetes
- Options:
    - Offer user to suspend notebook
    - Suspend user’s notebook if its resource utilization is too low for some time

Coordinated Checkpointing with CRIU

Motivation

Container checkpointing use cases:

- Podman / Docker / LXC – running containers on **individual hosts**
- Kubernetes – deployment & scaling of **distributed applications**

Problem:

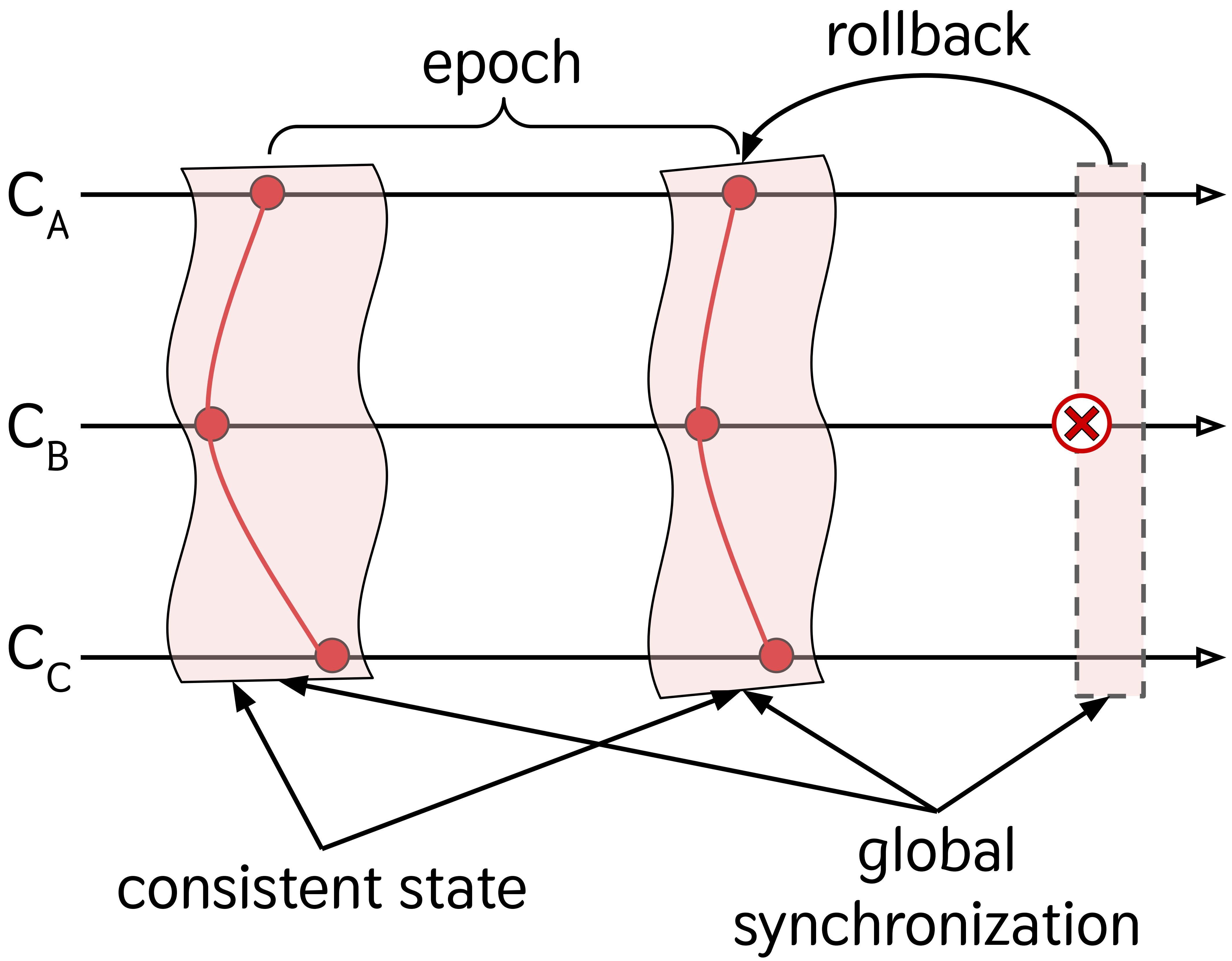
- Checkpointing with Kubernetes[1] is limited to individual containers
- How to enable support for checkpoint/restore of distributed applications?

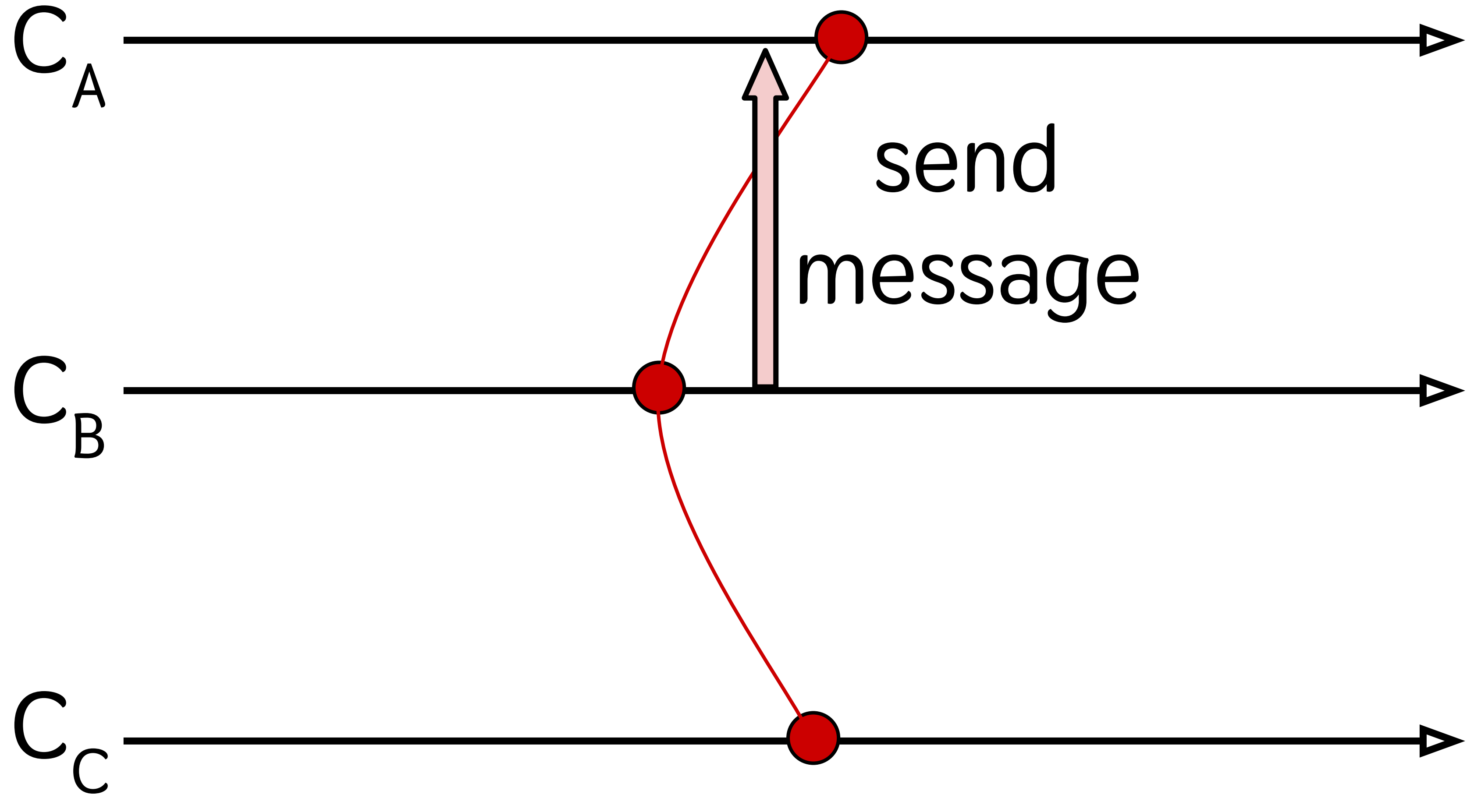
[1] https://kubernetes.io/docs/reference/node/kubelet-checkpoint-api/

Problem Statement & Research Questions

- CRIU is designed to checkpoint/restore **individual** process trees (containers)
  - How can we adapt CRIU to support checkpointing of **multiple containers** (potentially running on **different nodes**)?
  - How to enable checkpoint/restore synchronization of multiple CRIU instances?

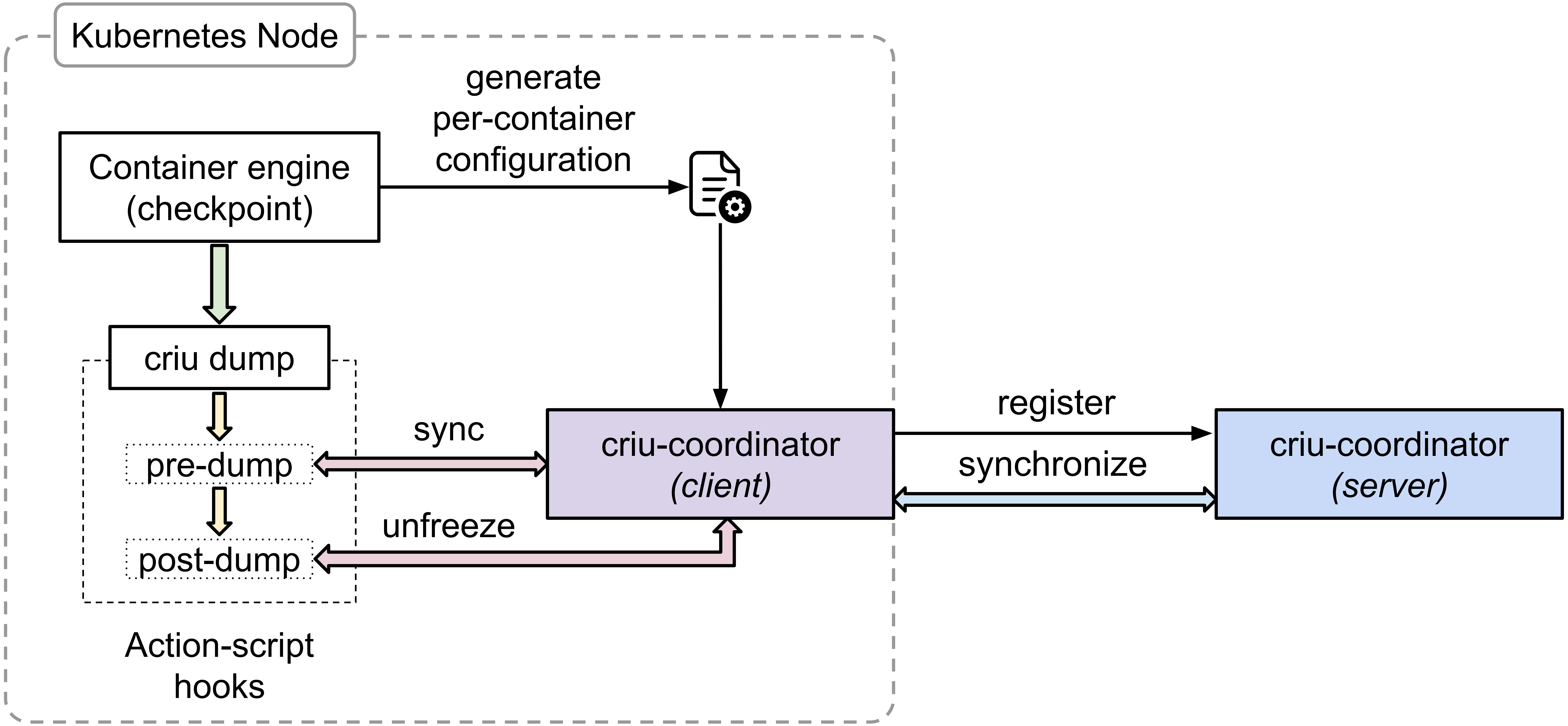
Coordinated Checkpointing Principle





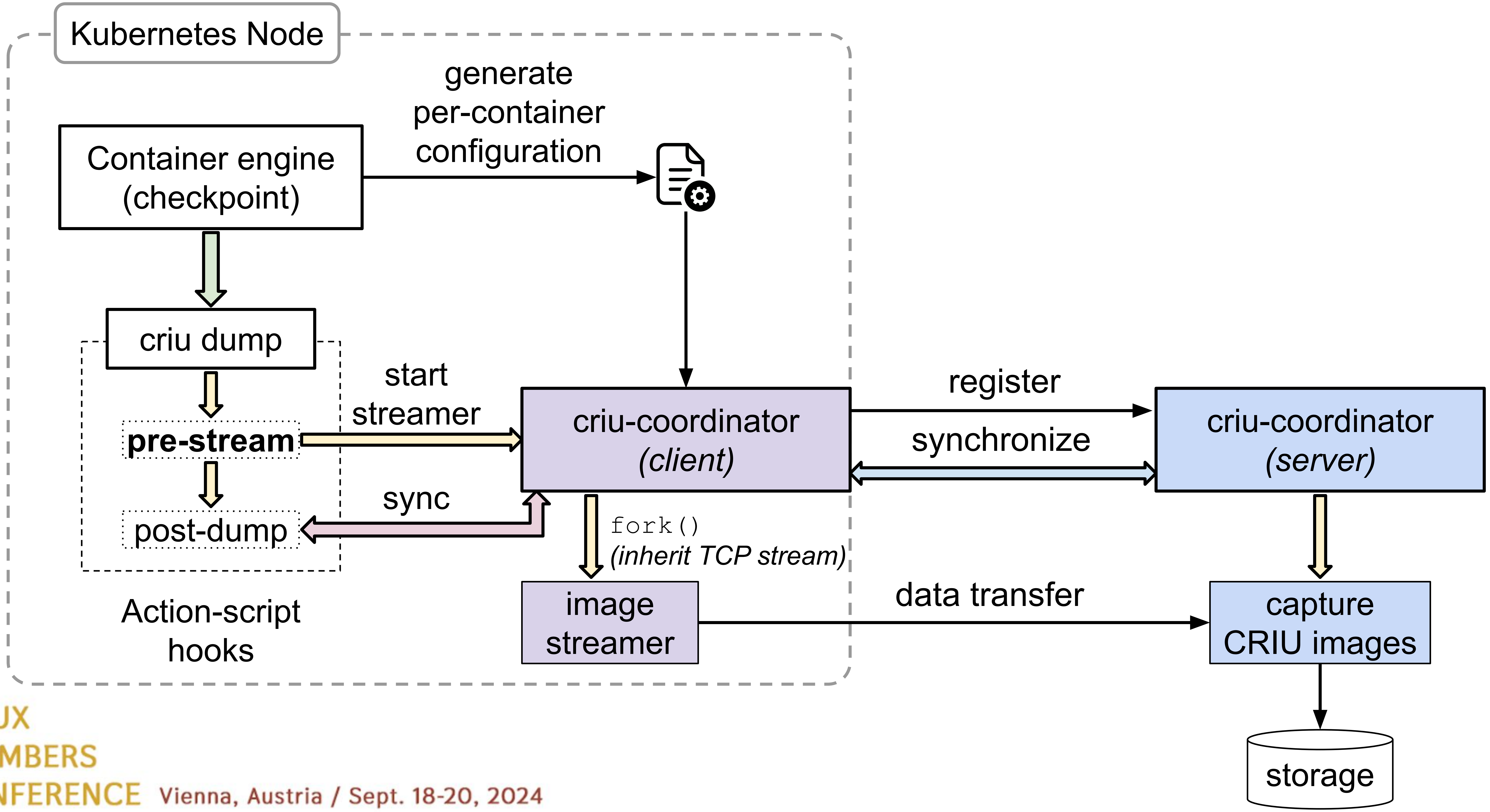
Inconsistent checkpoint: a message has been received by a process [C_A] but was never sent by the corresponding sender [C_B]
(breaks normal sequence of events)

Container Checkpointing in Kubernetes

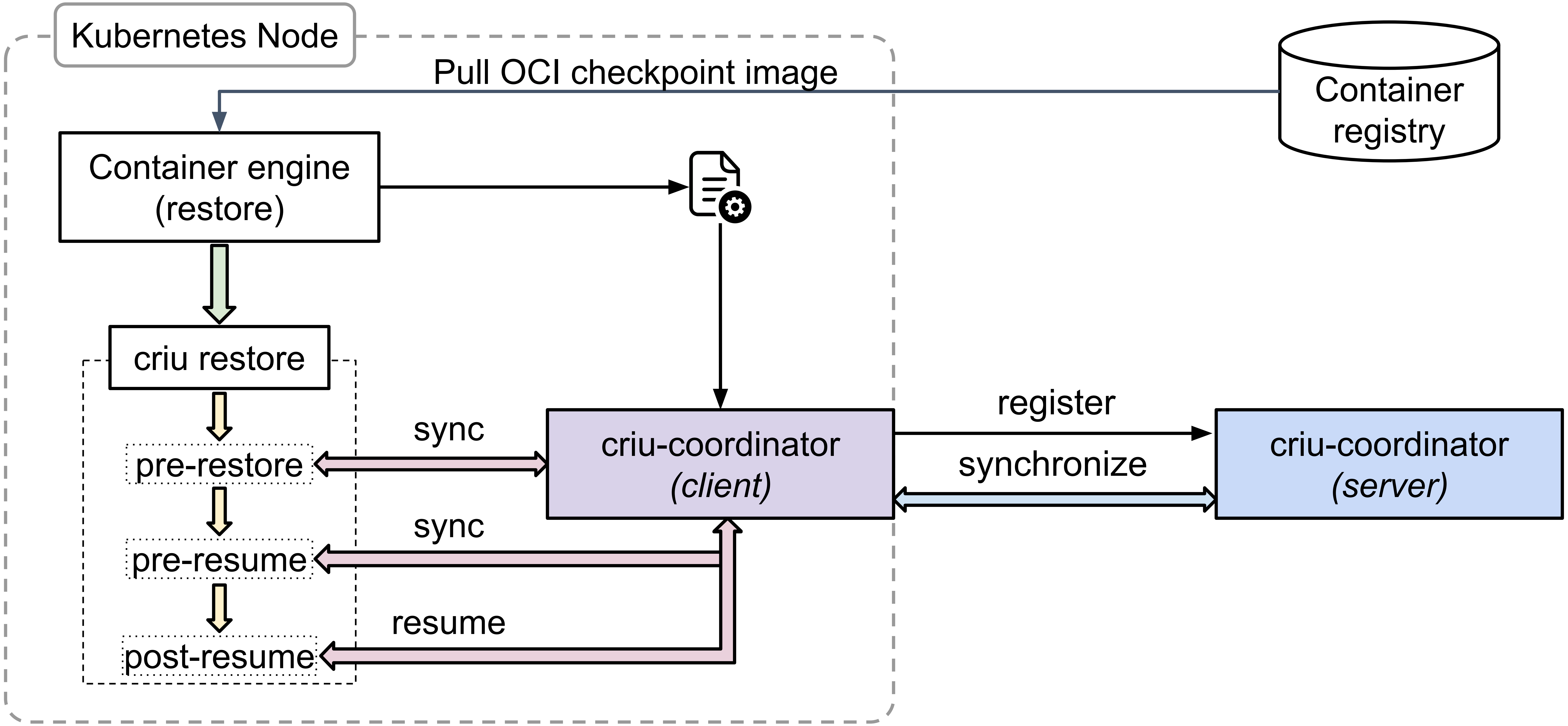


Kubernetes Node
Container engine (checkpoint)
generate per-container configuration
criu dump
pre-dump
post-dump
sync
unfreeze
Action-script hooks
criu-coordinator *(client)*
register
synchronize
criu-coordinator *(server)*

Checkpoint Streaming



Container Restore in Kubernetes



Demo

Summary & Future work

- **criu-coordinator** enables checkpointing of distributed applications with CRIU
- Checkpoint/Restore “barriers” are achieved through *action-script hooks*
- Coordinator config file is user-defined or generated by container runtimes
- Handling of established TCP connections remains an open research question

https://github.com/checkpoint-restore/criu-coordinator