SoCC' 24

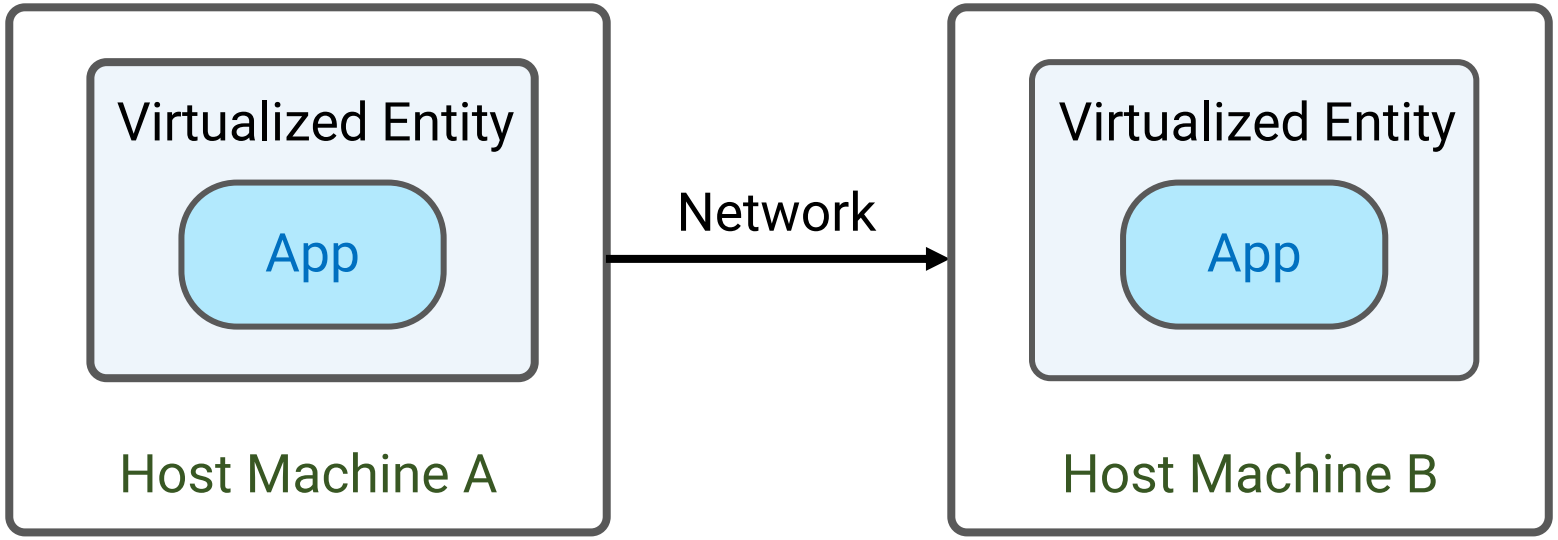
PCLive: Pipelined Restoration of Application Containers for Reduced Service Downtime

Shiv Bhushan Tripathi, Debadatta Mishra

Department of Computer Science and Engineering,
Indian Institute of Technology (IIT) Kanpur

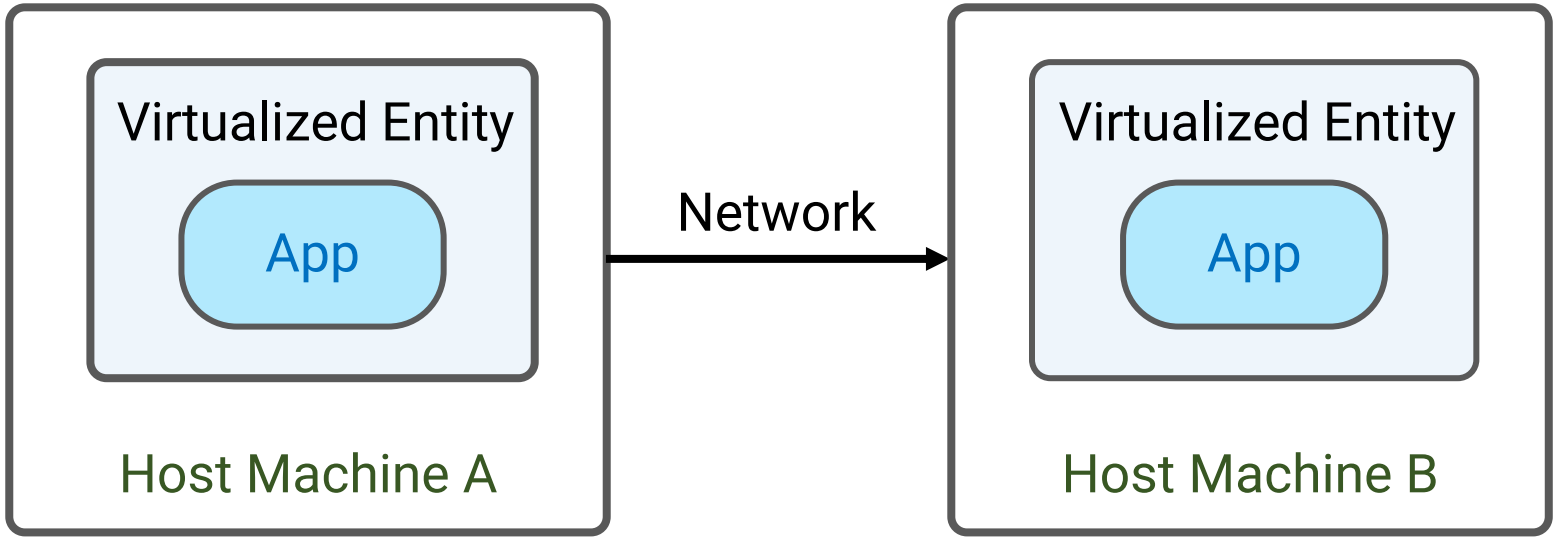
CDOS Research Group, IIT Kanpur

Live Migration



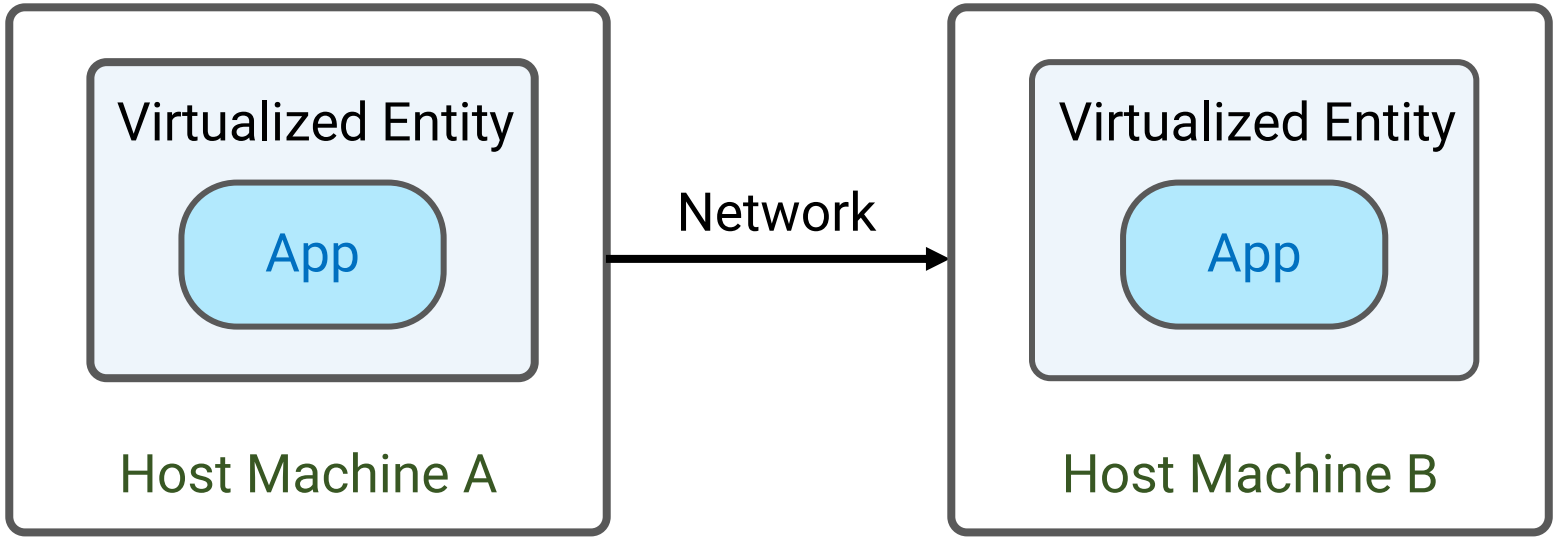
- Usage: Load balancing, system maintenance etc.

Live Migration



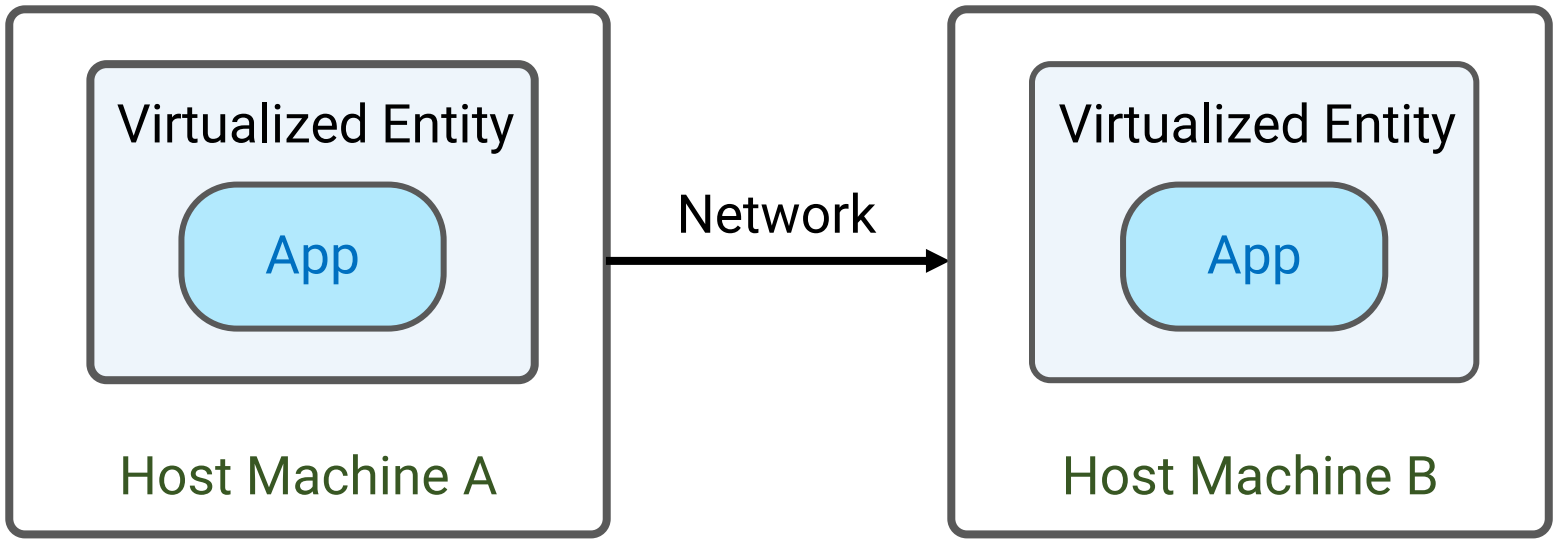
- Usage: Load balancing, system maintenance etc.
- Service downtime is crucial for liveliness of applications.

Live Migration



- Usage: Load balancing, system maintenance etc.
- Service downtime is crucial for liveliness of applications.
- Iterative pre–copy is a robust technique to reduce service downtime.

Live Migration

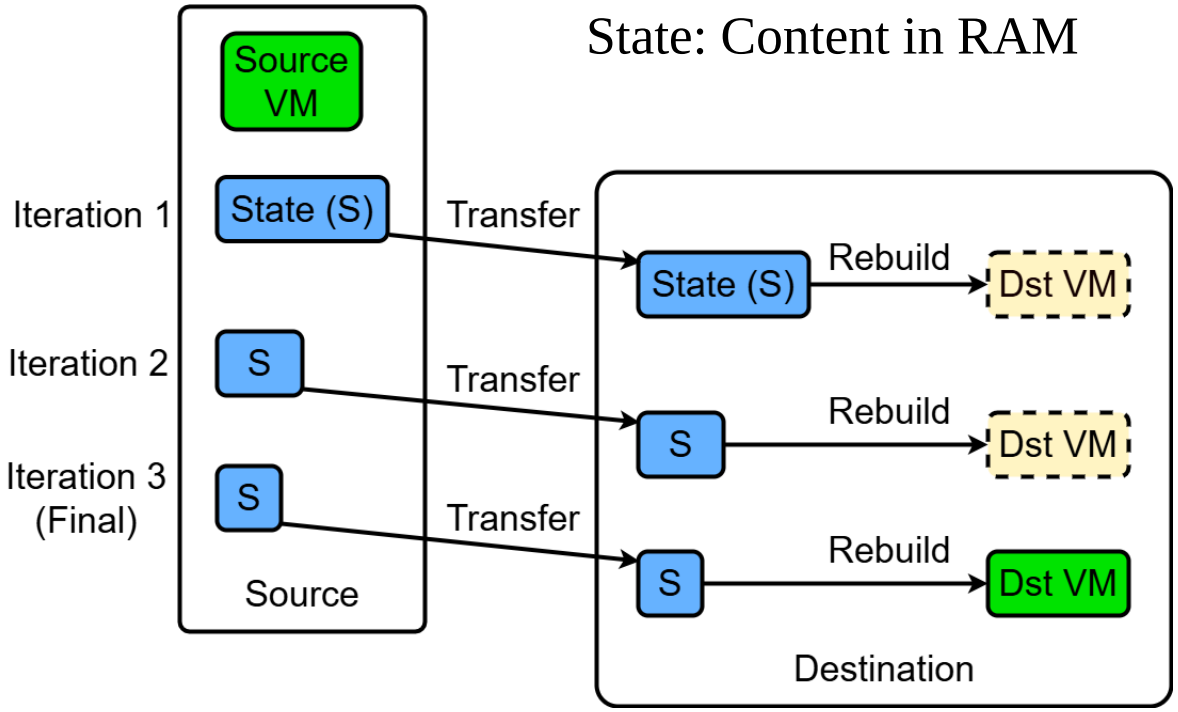


- Usage: Load balancing, system maintenance etc.
- Service downtime is crucial for liveliness of applications.
- Iterative pre–copy is a robust technique to reduce service downtime.

Paper Scope: To optimize downtime for iterative live migration of containers

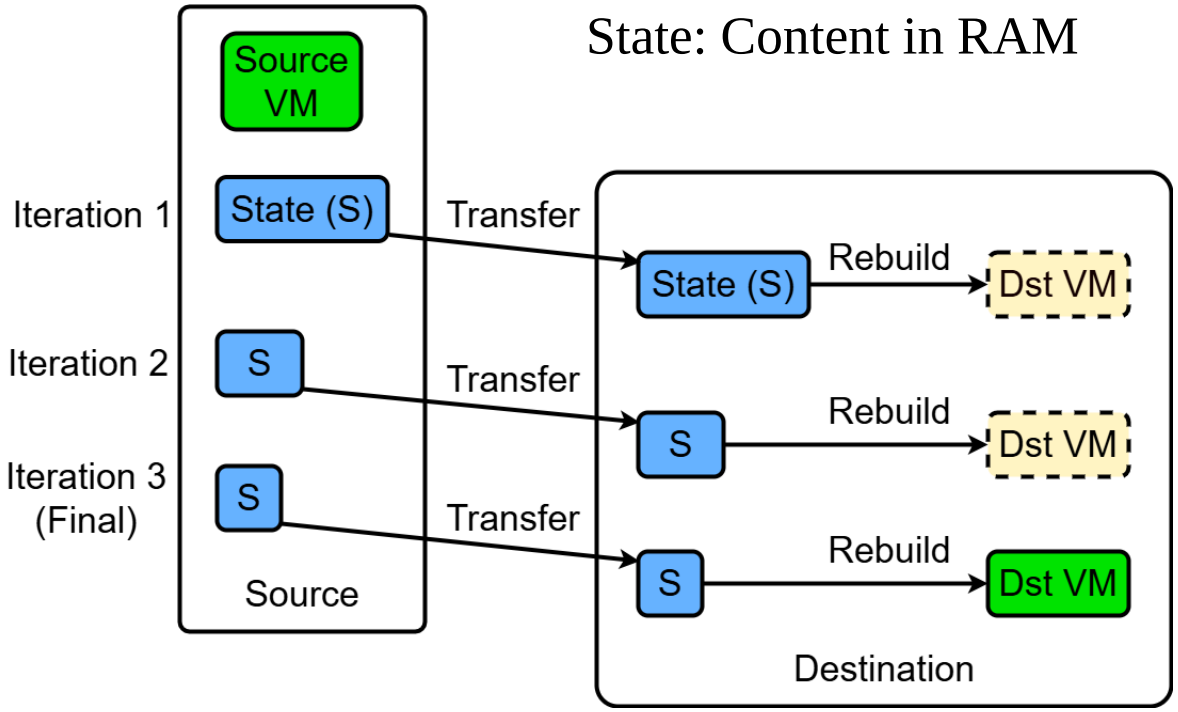
Container Migration vs VM Migration: Diskless

Container Migration vs VM Migration: Diskless



VM Migration
(Iterative rebuild)

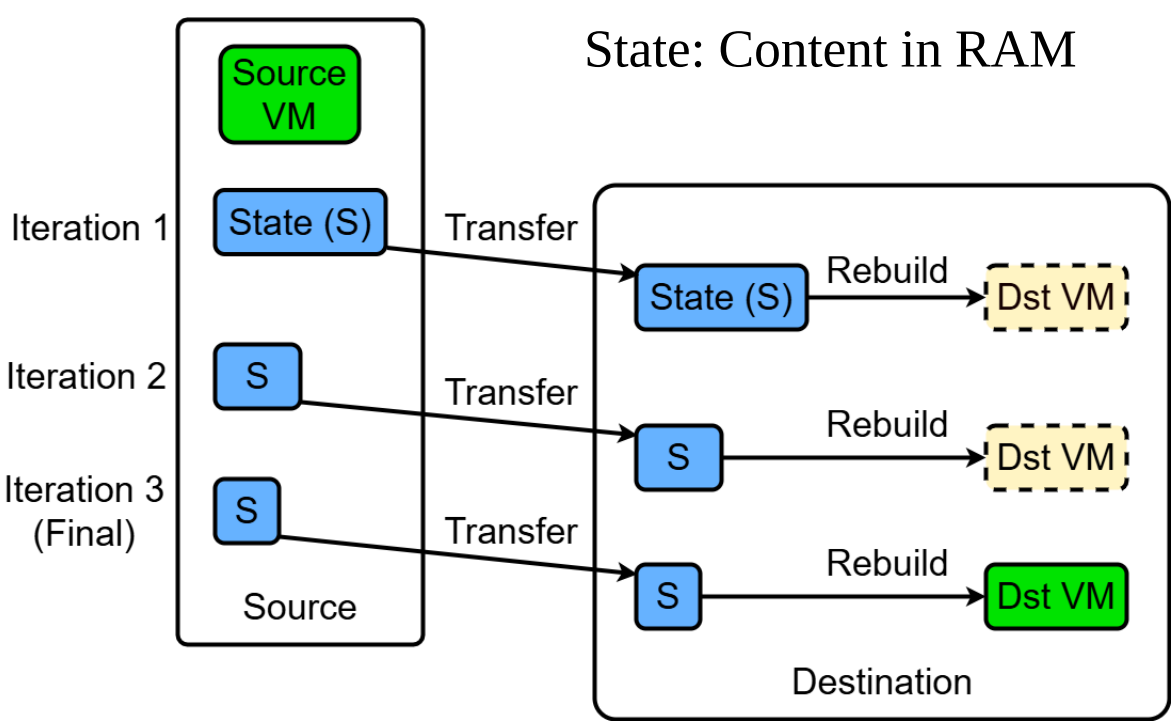
Container Migration vs VM Migration: Diskless



VM Migration
(Iterative rebuild)

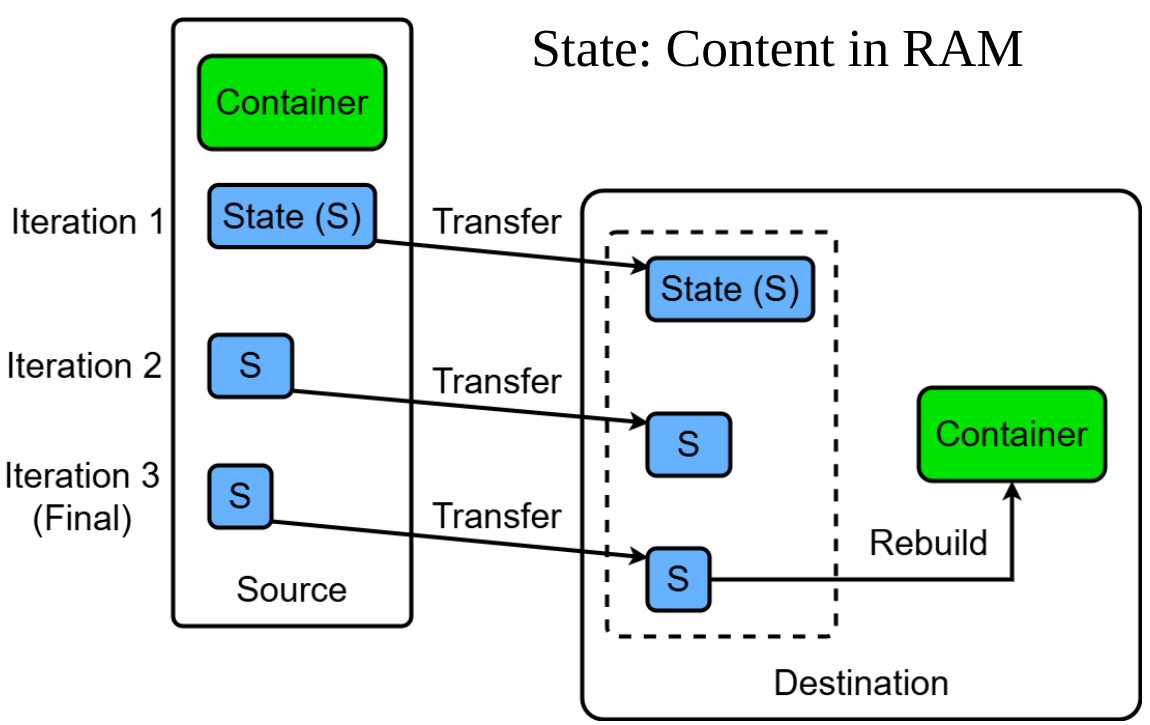
Downtime ∝ dirty rate

Container Migration vs VM Migration: Diskless



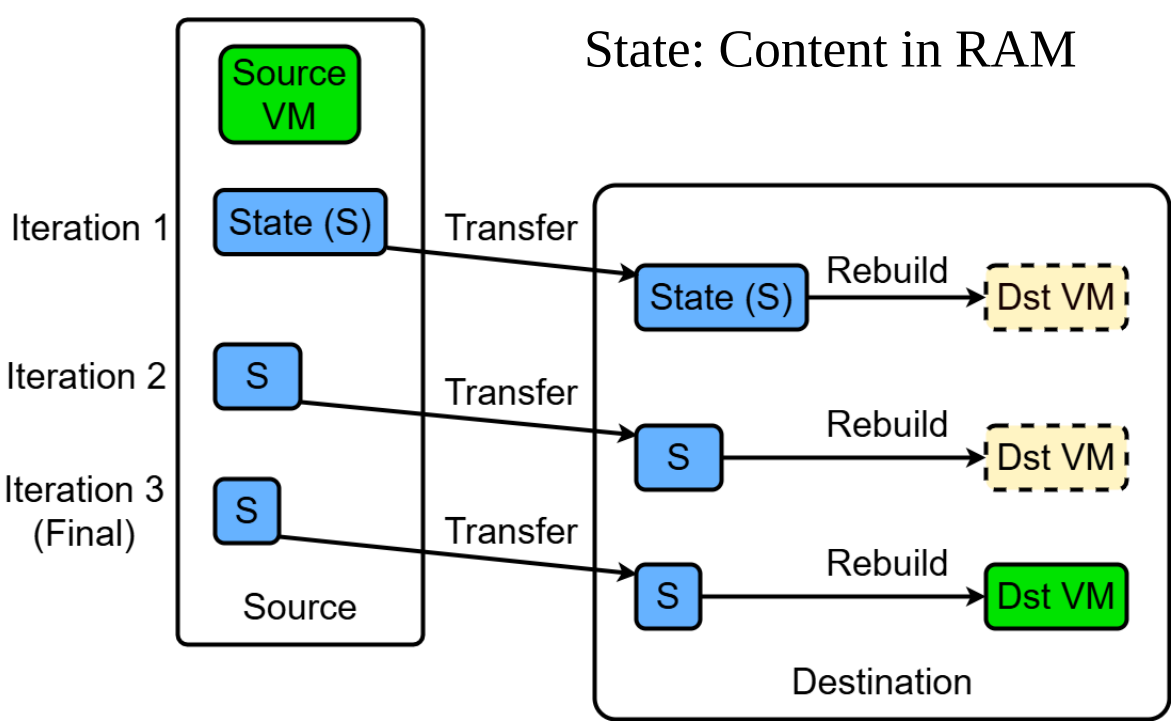
VM Migration
(Iterative rebuild)

Downtime ∝ dirty rate



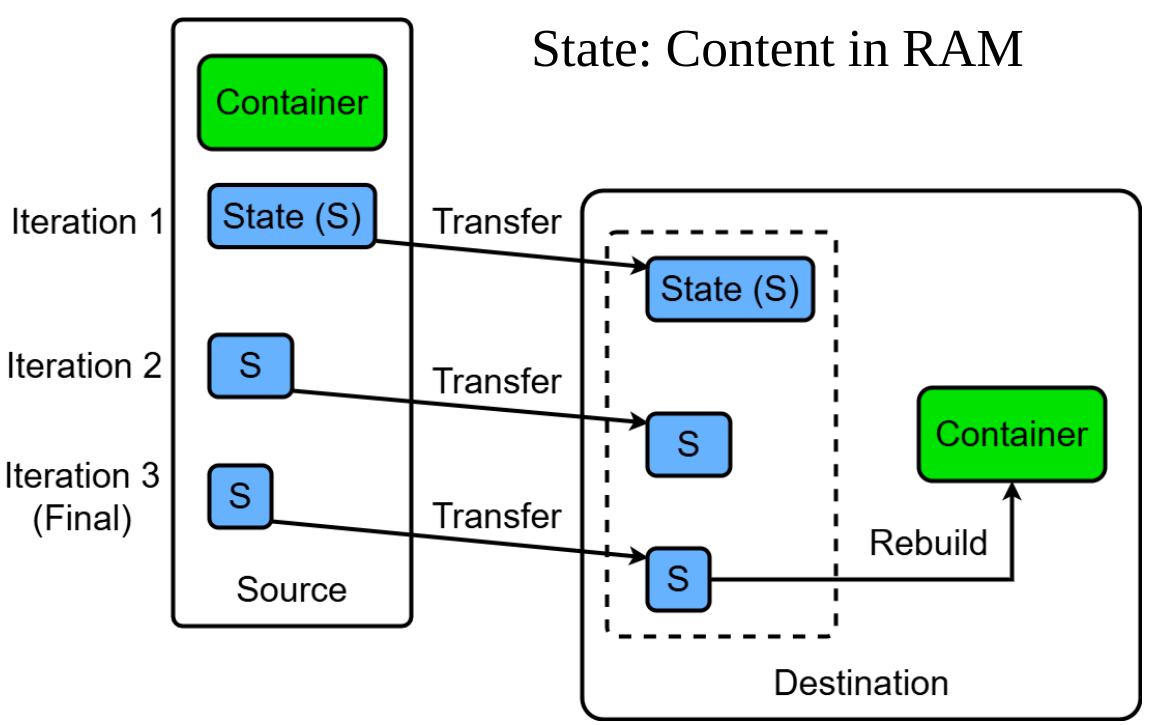
Container Migration
(One shot rebuild)

Container Migration vs VM Migration: Diskless



VM Migration
(Iterative rebuild)

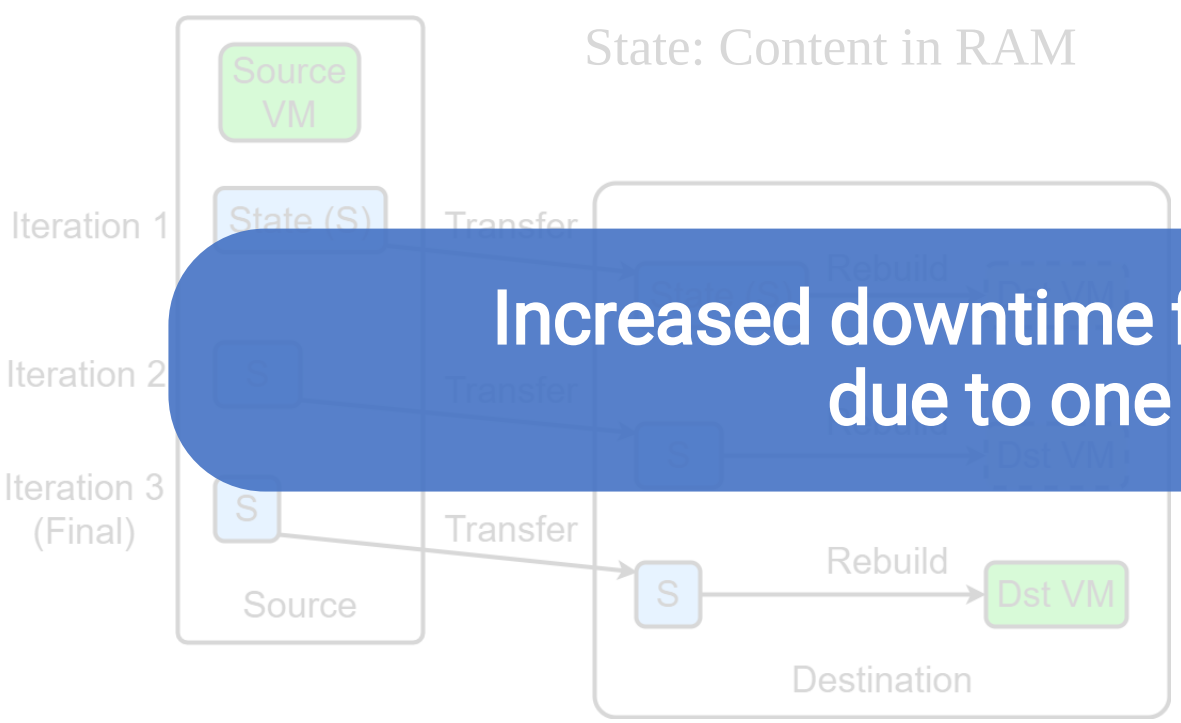
Downtime ∝ dirty rate



Container Migration
(One shot rebuild)

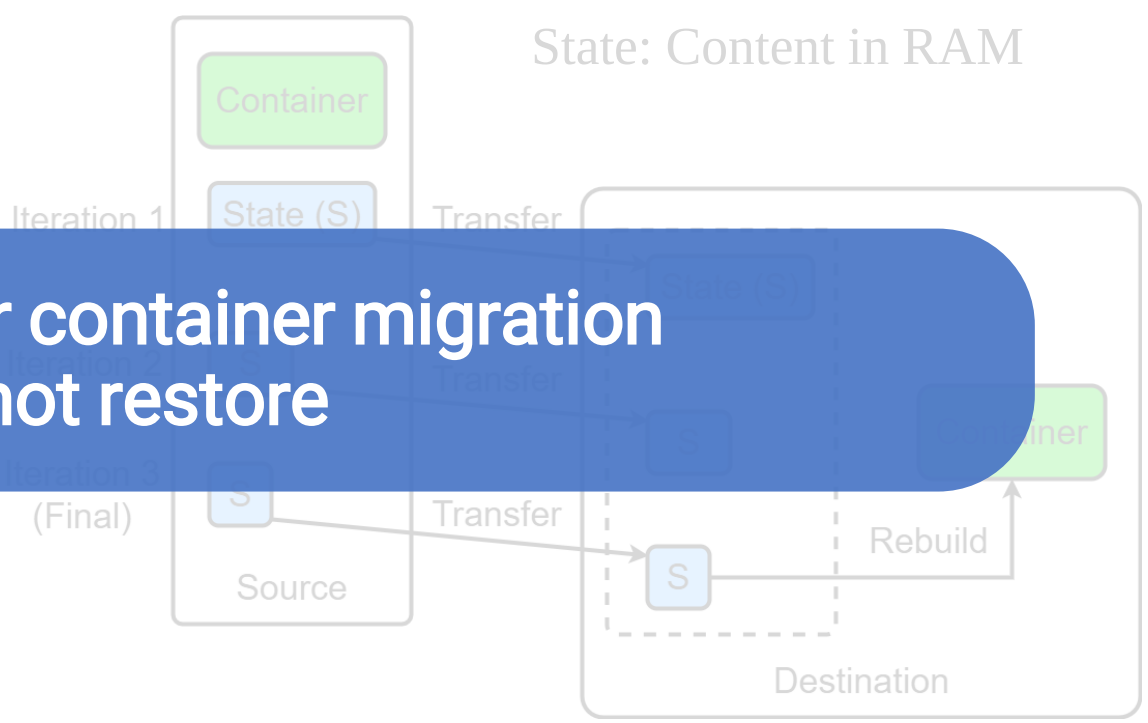
Downtime ∝ <dirty rate, container size>

Container Migration vs VM Migration: Diskless



VM Migration
(Iterative rebuild)

Downtime ∝ dirty rate

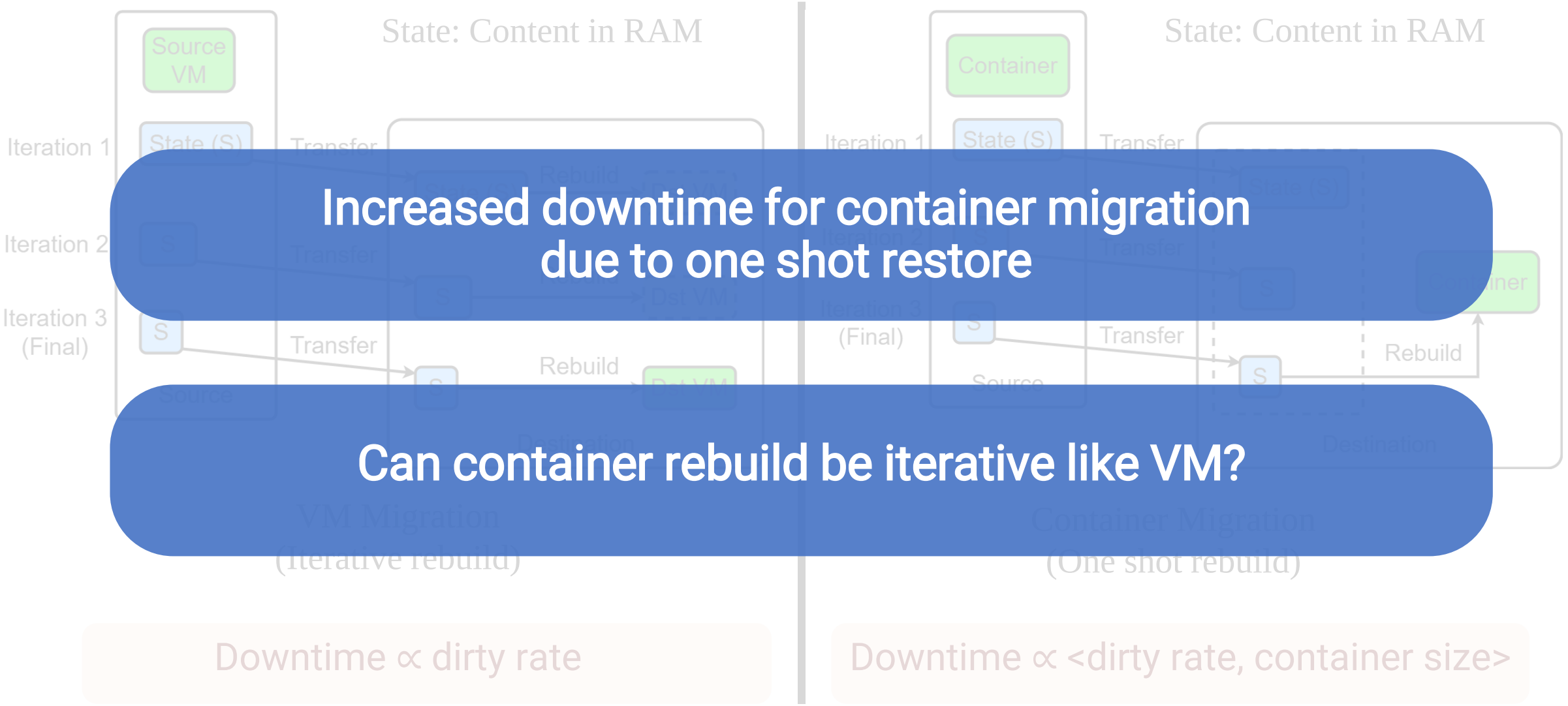


Container Migration
(One shot rebuild)

Downtime ∝ <dirty rate, container size>

Increased downtime for container migration due to one shot restore

Container Migration vs VM Migration: Diskless



Increased downtime for container migration due to one shot restore

Can container rebuild be iterative like VM?

Iterative State Rebuild: VM Migration

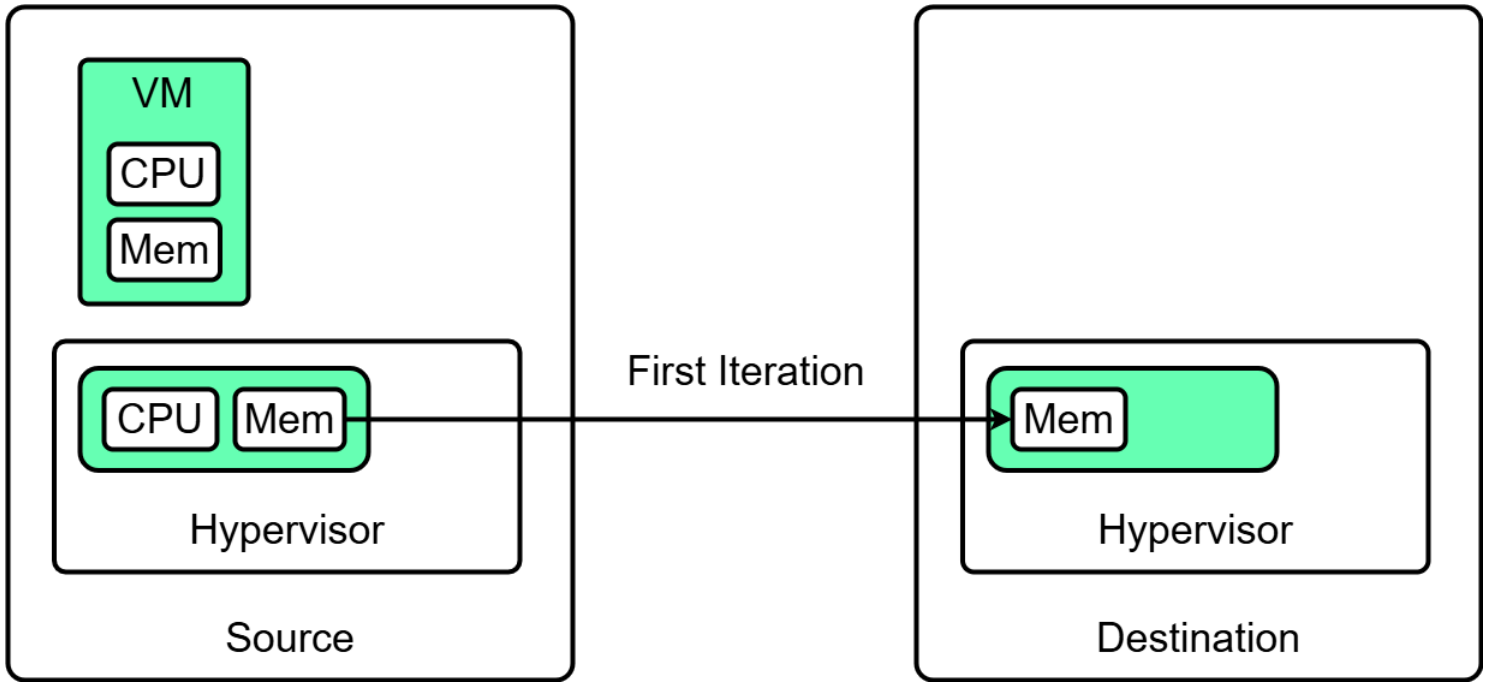
- In a diskless migration, memory state is bulky.

Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU

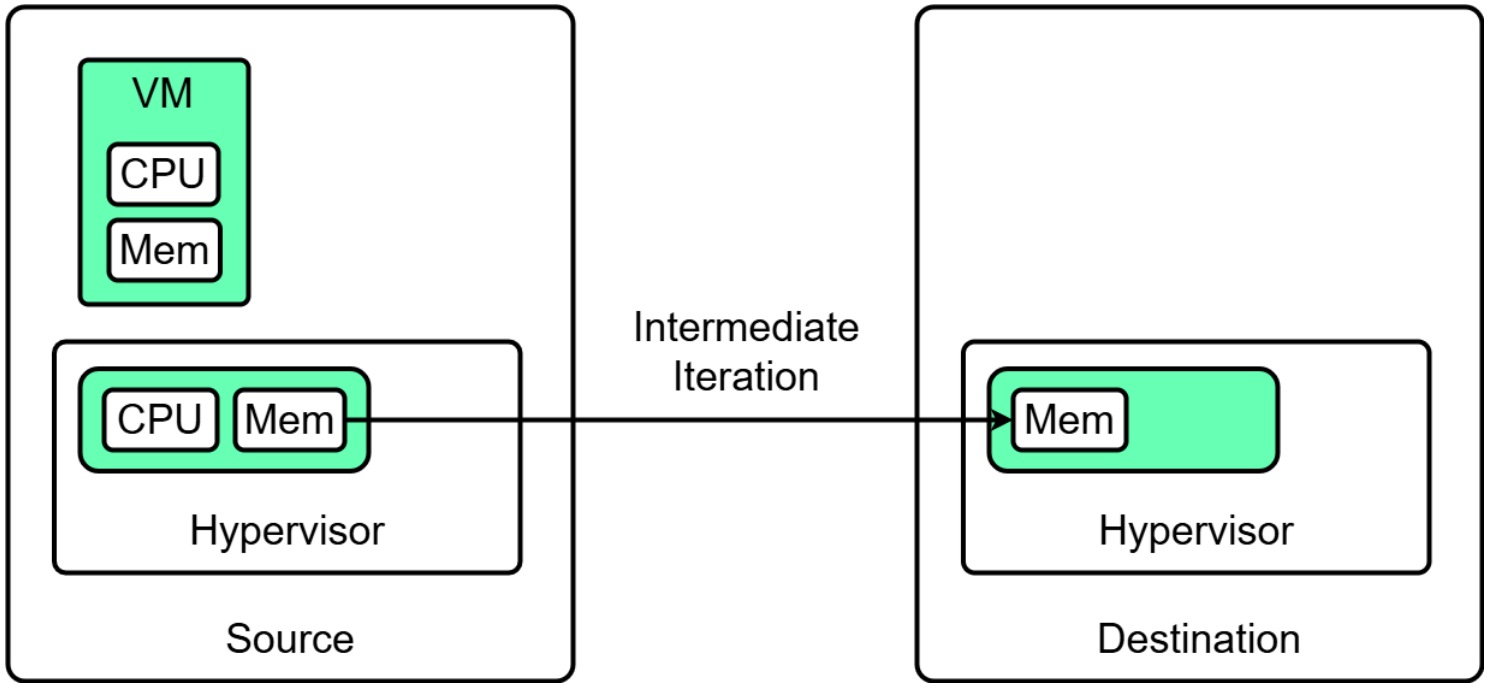
Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU



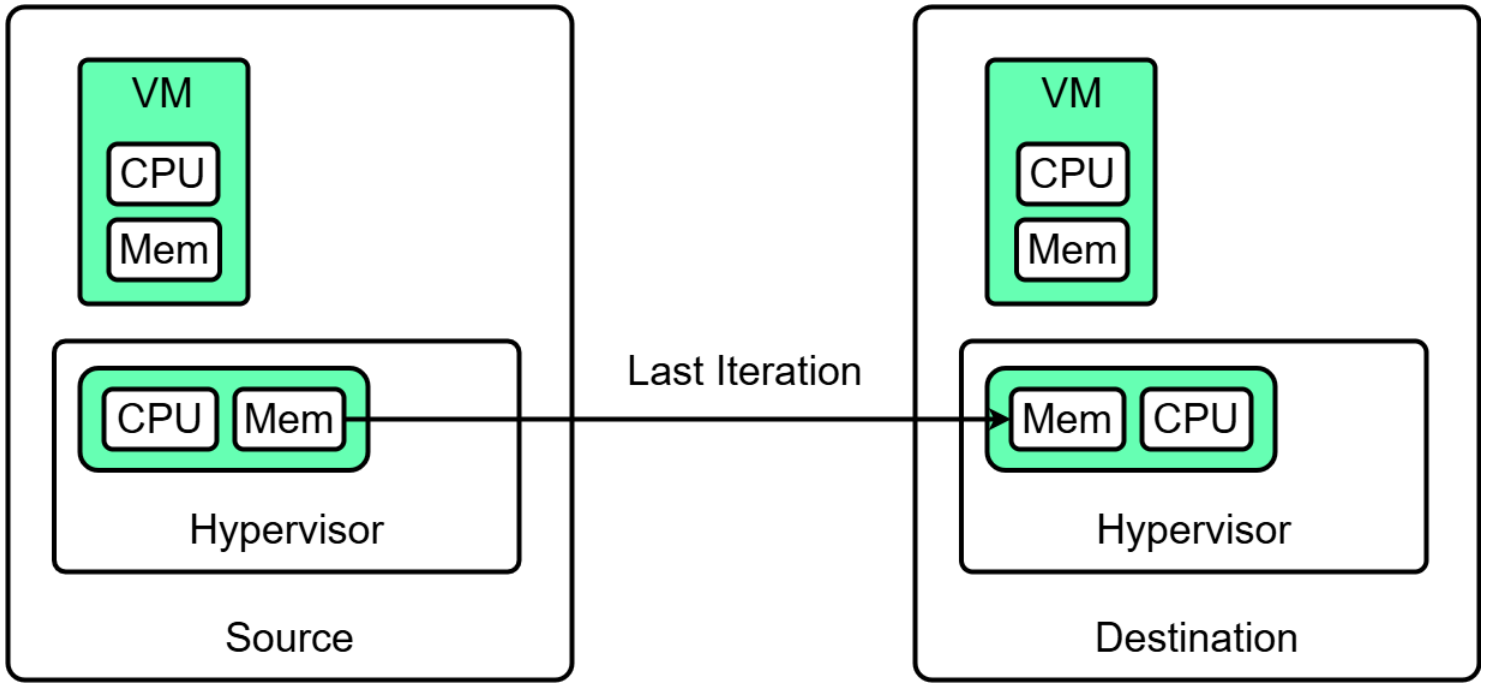
Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU



Iterative State Rebuild: VM Migration

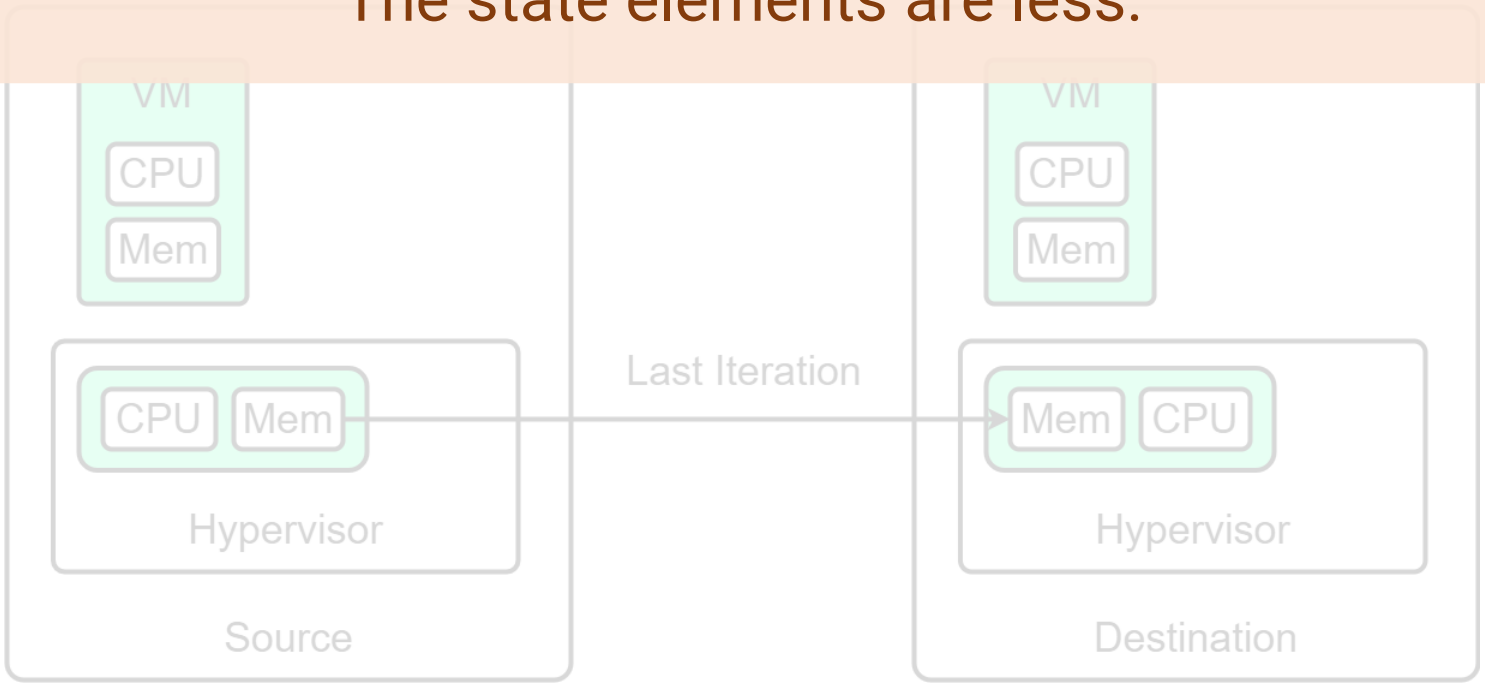
- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU



Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU

The state elements are less.

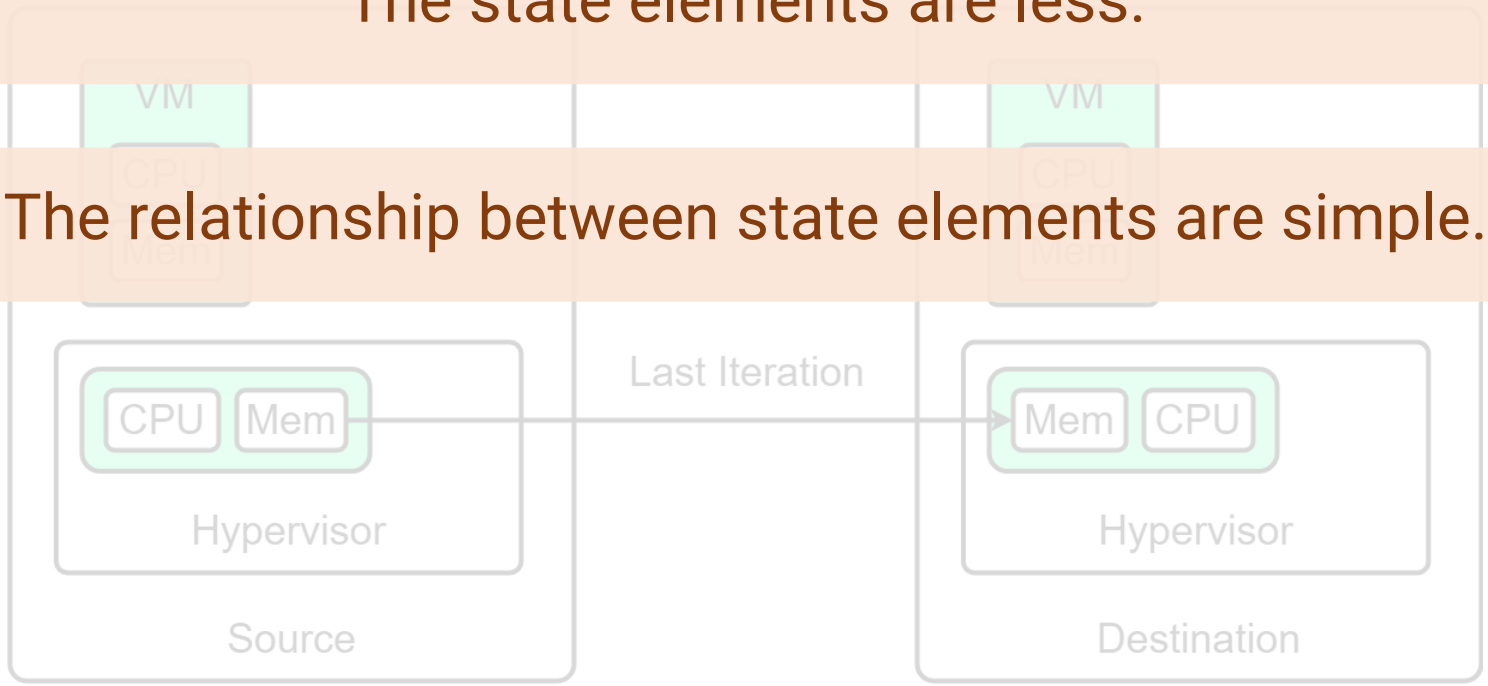


Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU

The state elements are less.

The relationship between state elements are simple.



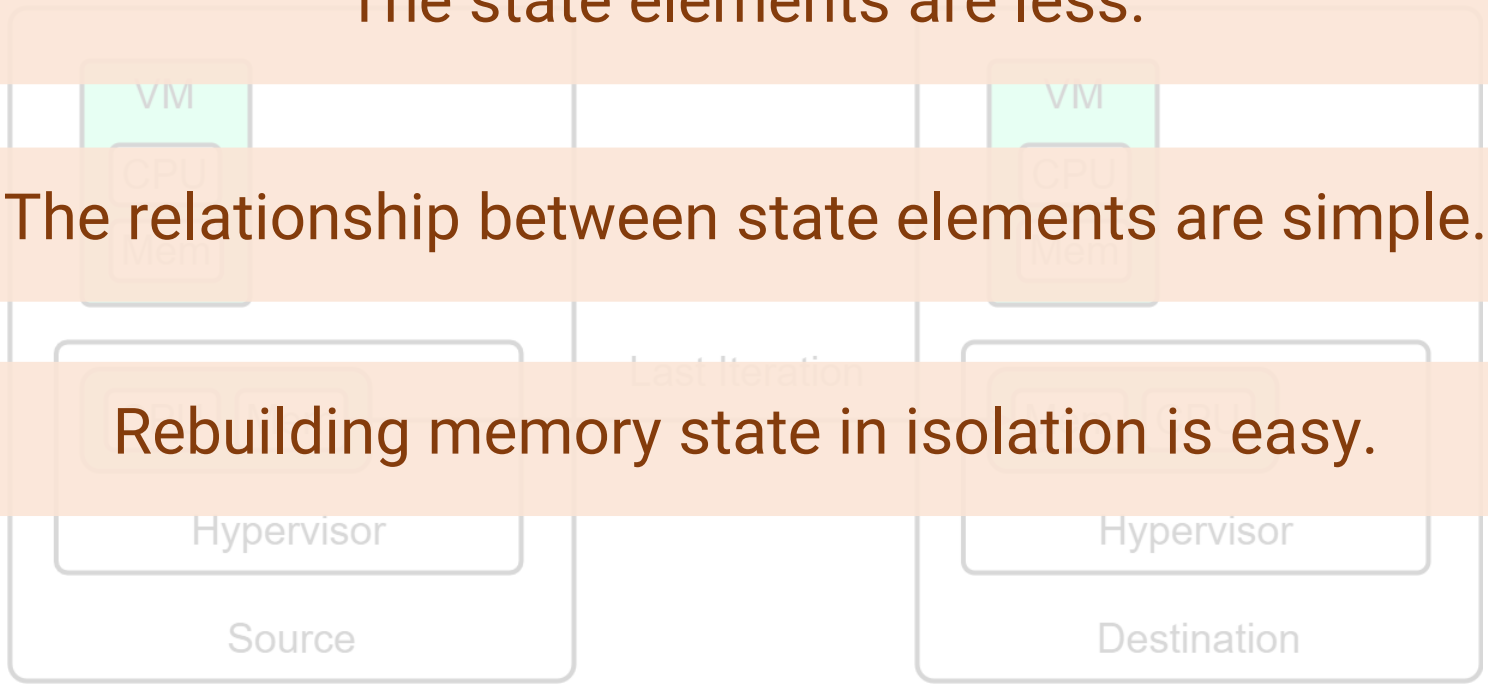
Iterative State Rebuild: VM Migration

- In a diskless migration, memory state is bulky.
- VM State: Virtual hardware resources, e.g. Physical memory, CPU

The state elements are less.

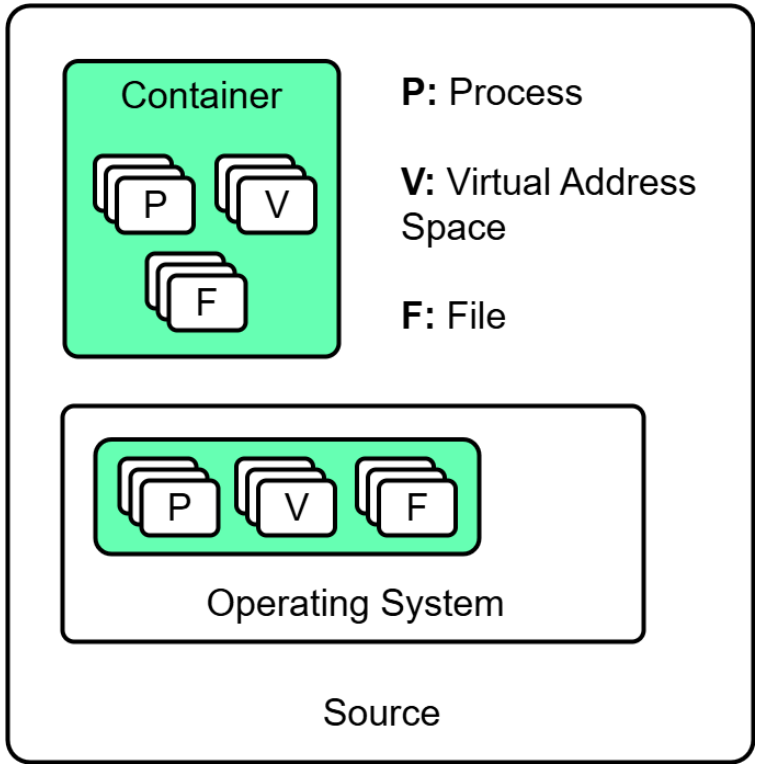
The relationship between state elements are simple.

Rebuilding memory state in isolation is easy.



Iterative State Rebuild: Container Migration

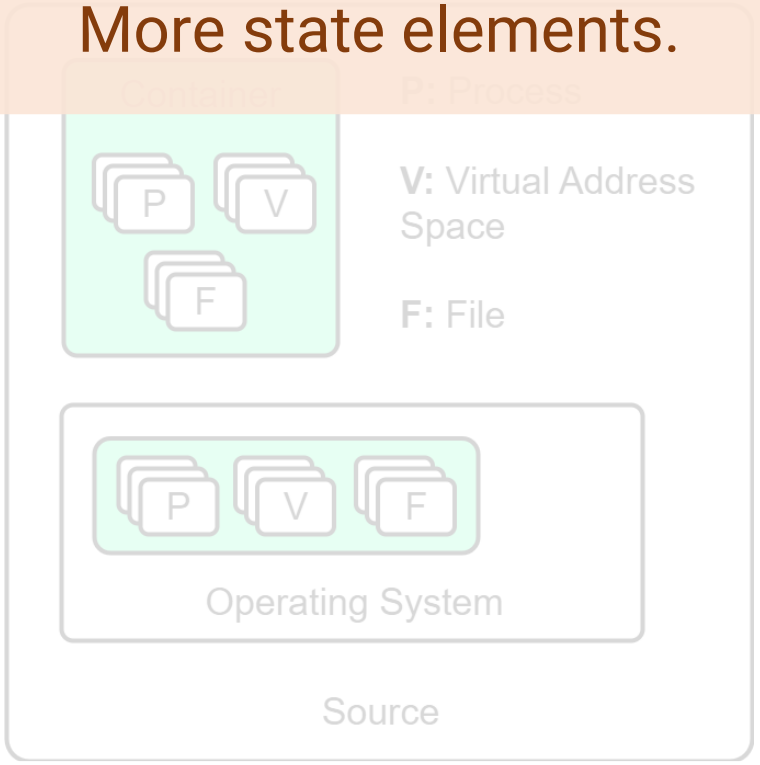
- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space, File



Iterative State Rebuild: Container Migration

- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space,

More state elements.

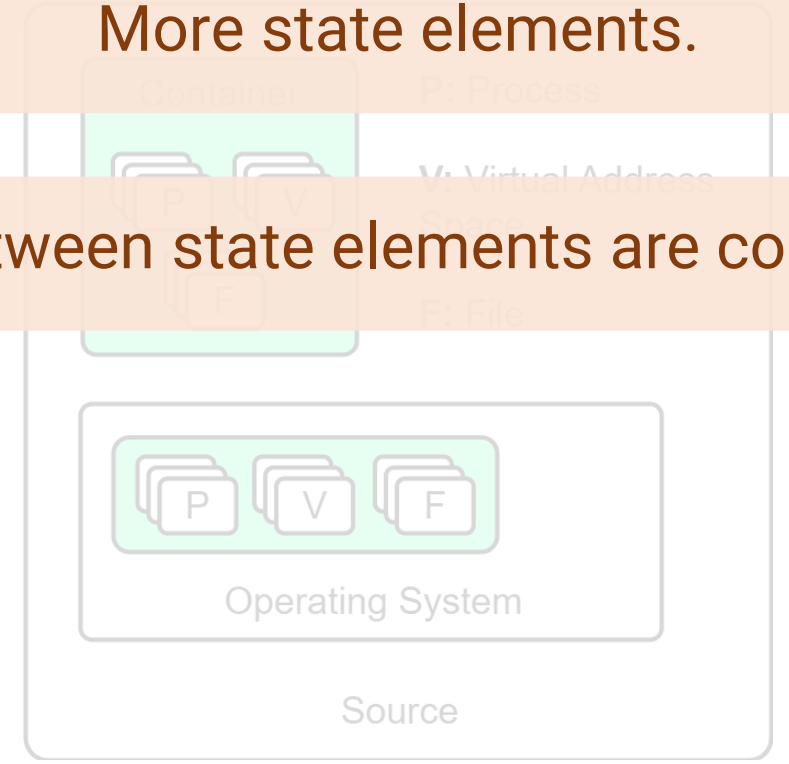


Iterative State Rebuild: Container Migration

- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space,

More state elements.

The relationship between state elements are complex and dynamic.



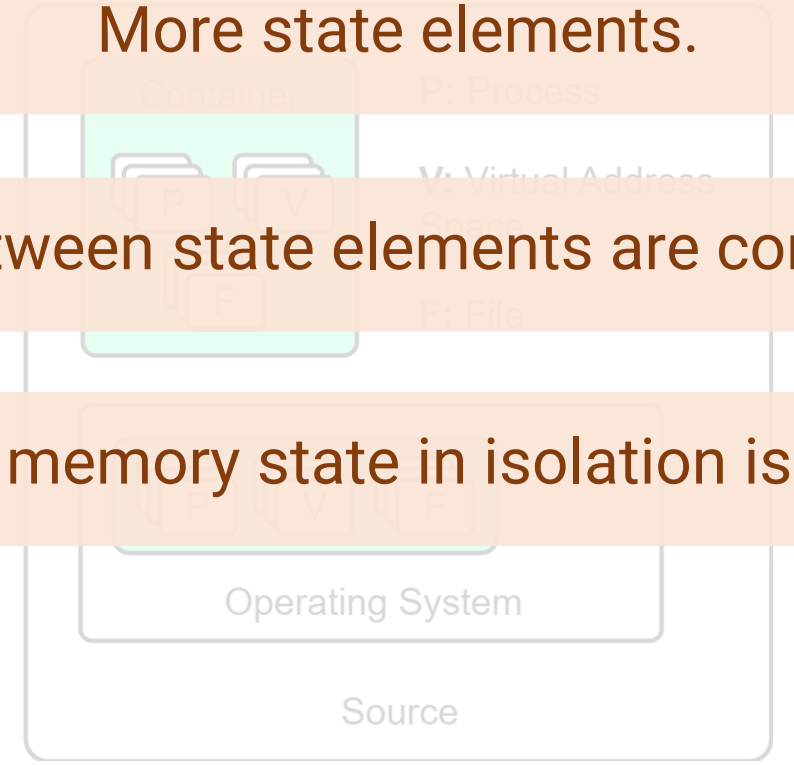
Iterative State Rebuild: Container Migration

- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space,

More state elements.

The relationship between state elements are complex and dynamic.

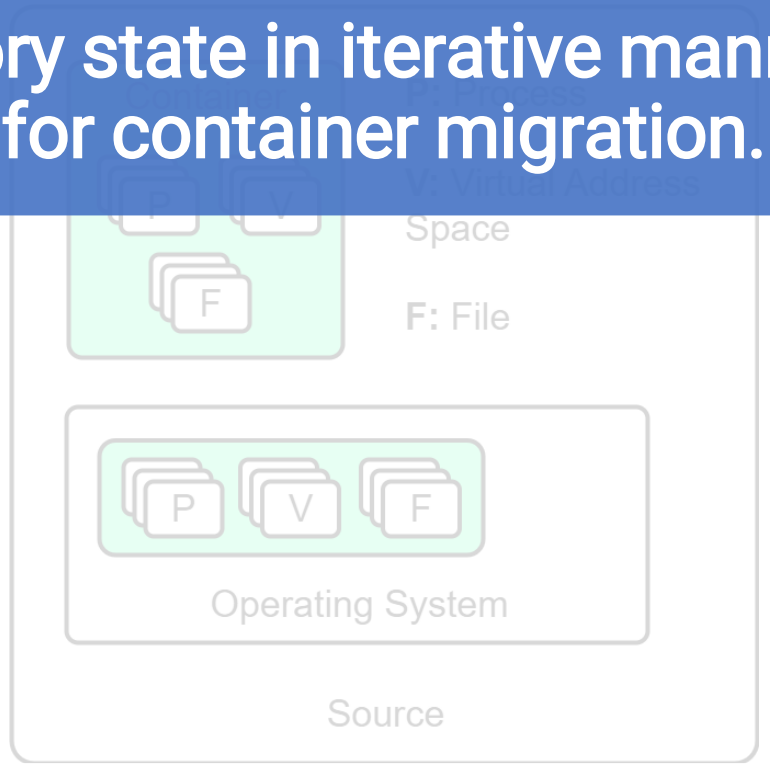
Rebuilding memory state in isolation is non trivial.



Iterative State Rebuild: Container Migration

- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space,

Rebuilding memory state in iterative manner is challenging for container migration.

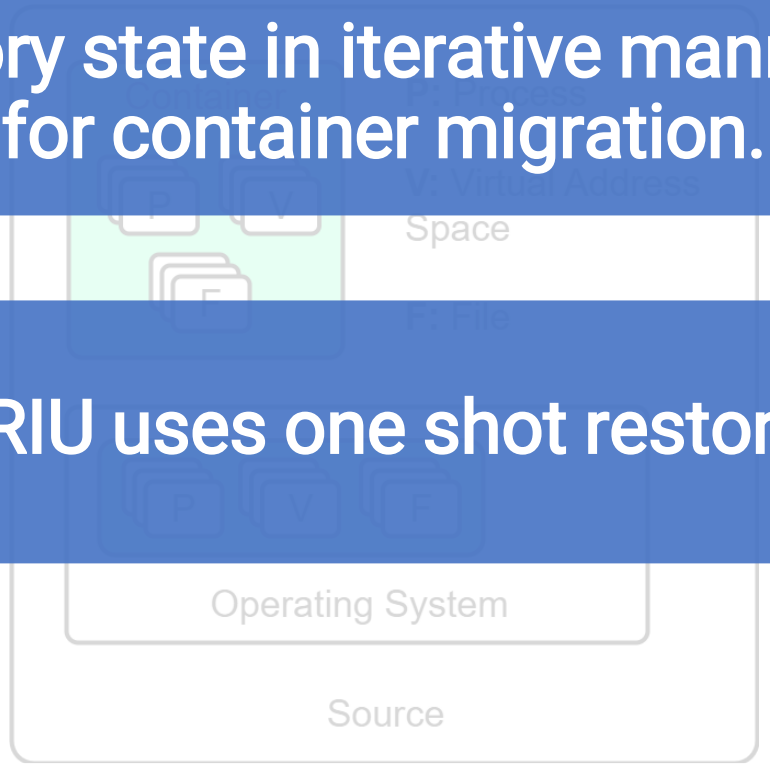


Iterative State Rebuild: Container Migration

- In a diskless migration, memory state is bulky.
- Container State: OS abstractions, e.g. Process, Virtual address space,

Rebuilding memory state in iterative manner is challenging for container migration.

CRIU uses one shot restore.



Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.

Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.

- Identify and address the challenges to design and implement iterative rebuilding using PCLive.

Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.

Contributions

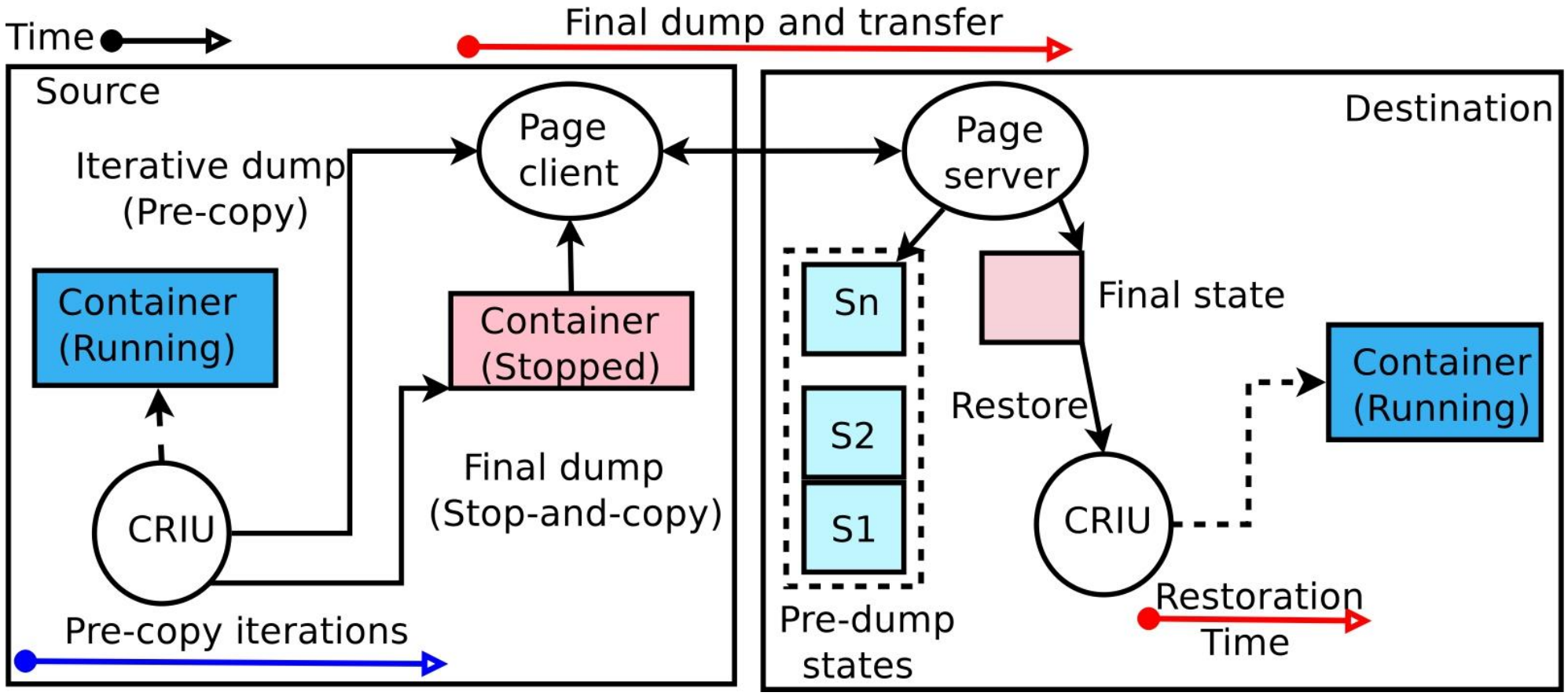
- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.
- Showcase the flexibility of PCLive to find a sweet-spot for resource overhead and downtime tradeoff.

Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.
- Showcase the flexibility of PCLive to find a sweet-spot for resource overhead and downtime tradeoff.

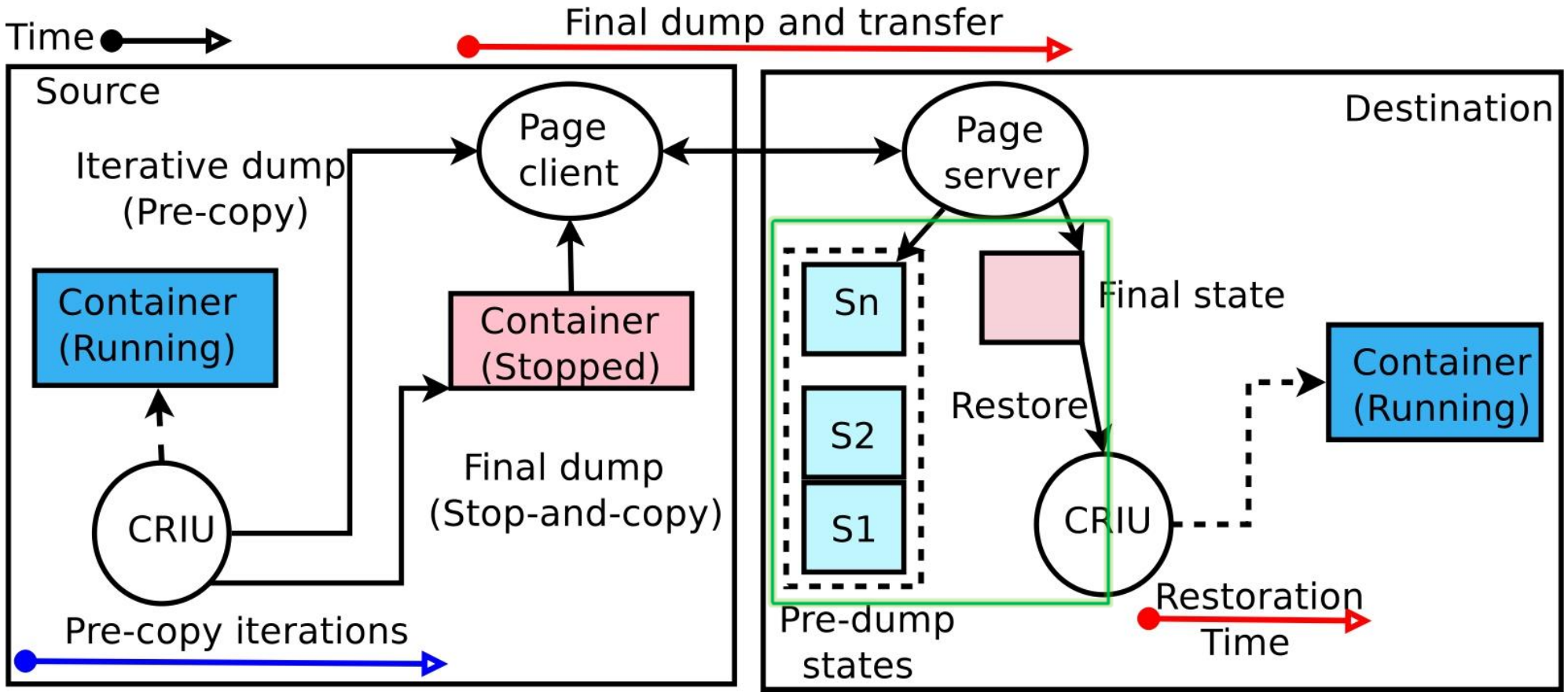
CRIU: One Shot Restore

CRIU: One Shot Restore



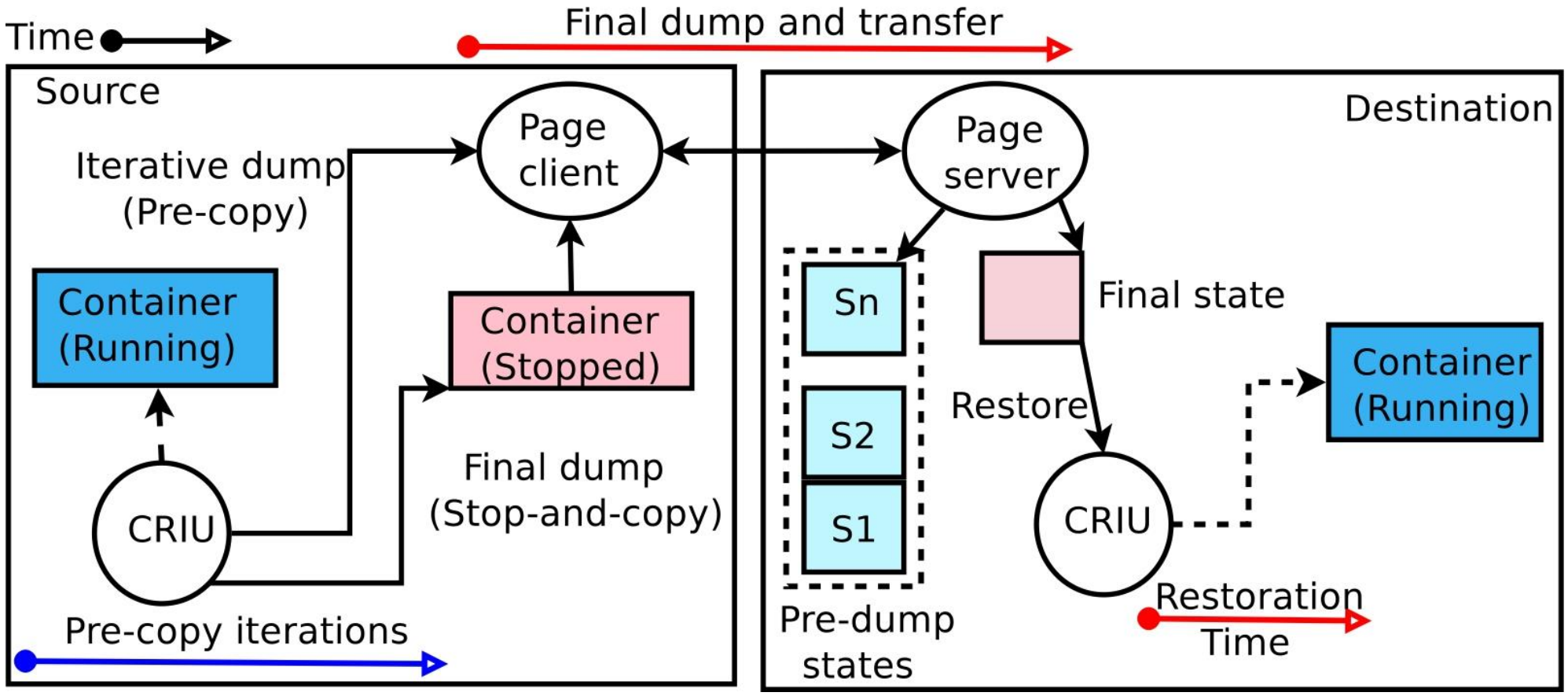
- During pre-copy, only memory states are sent and deduplication is used.

CRIU: One Shot Restore



- During pre-copy, only memory states are sent and deduplication is used.

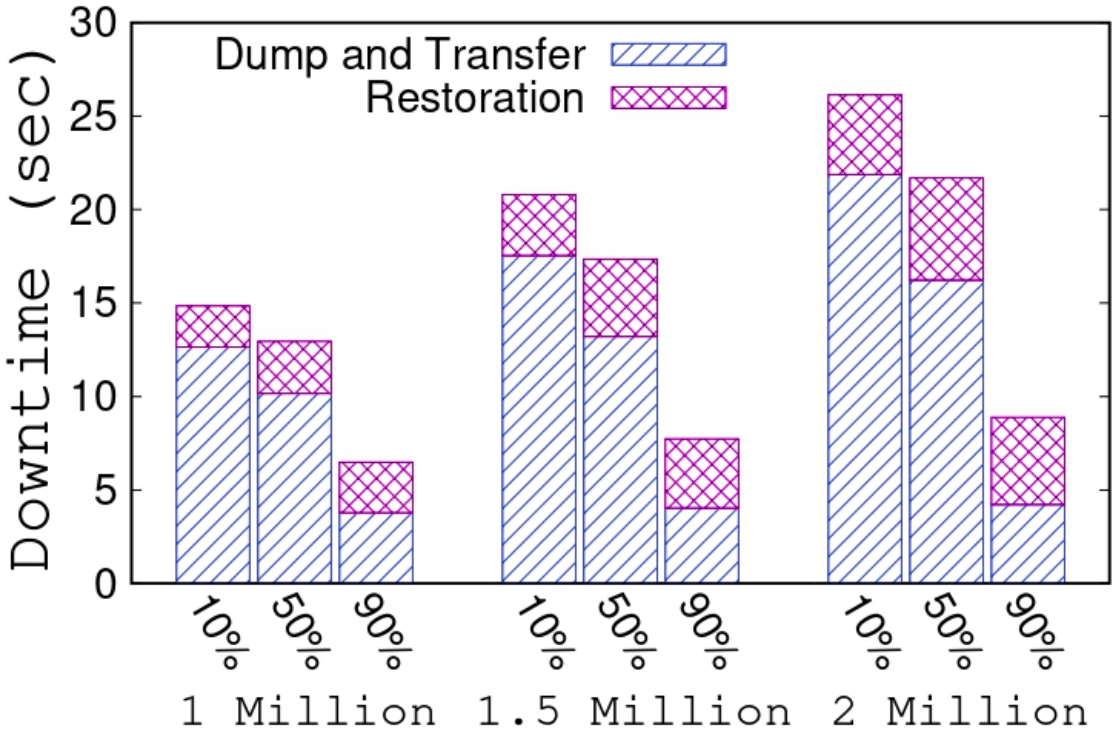
CRIU: One Shot Restore



- Downtime = Final Dump and Transfer Time + Restoration Time

Downtime: Restoration Technique Matters!

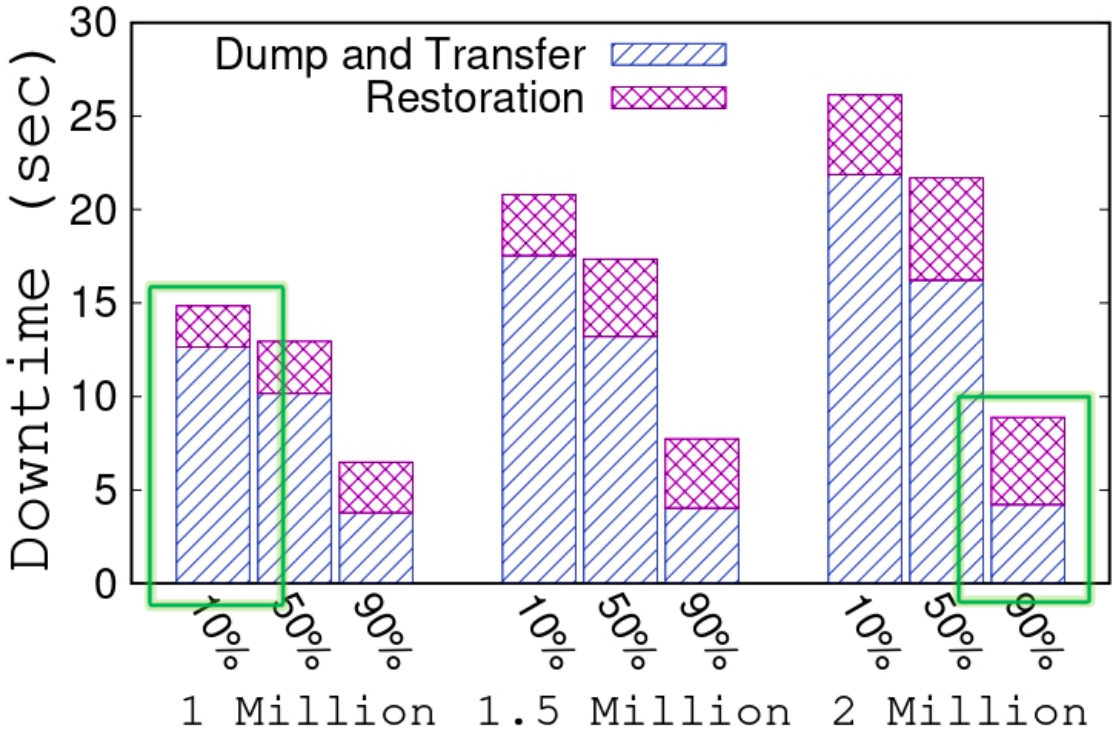
- **Setup:** Live migration with container running Redis workload with YCSB for different records and read to write ratio.



Split Cost of Downtime

Downtime: Restoration Technique Matters!

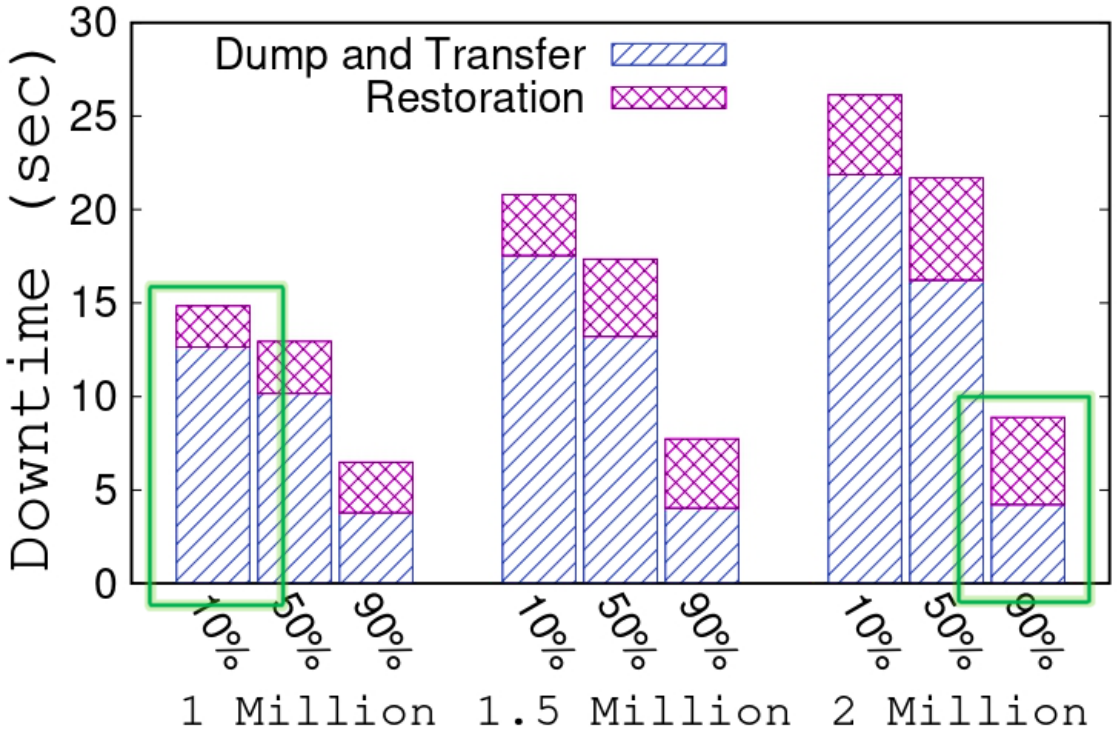
- **Setup:** Live migration with container running Redis workload with YCSB for different records and read to write ratio.
- **Restoration time:** a key contributor in downtime (14.8% - 51%).



Split Cost of Downtime

Downtime: Restoration Technique Matters!

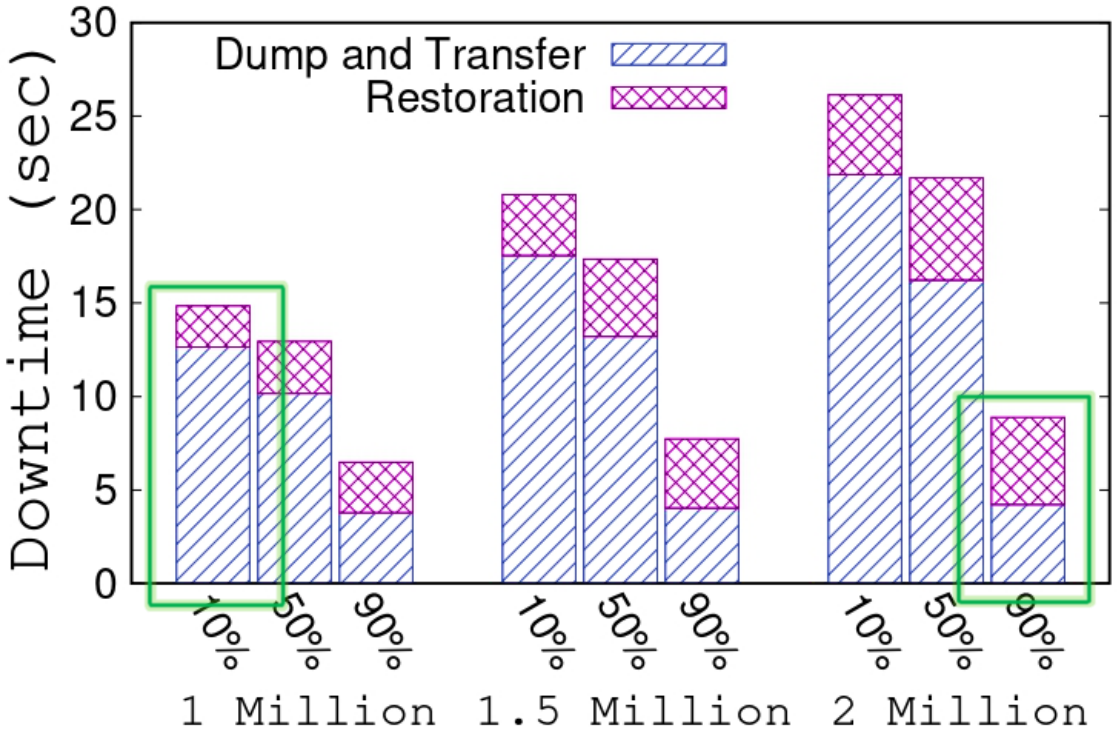
- **Setup:** Live migration with container running Redis workload with YCSB for different records and read to write ratio.
- **Restoration time:** a key contributor in downtime (14.8% - 51%).
- Restore time for write intensive workload with 1M records is 2.2s



Split Cost of Downtime

Downtime: Restoration Technique Matters!

- **Setup:** Live migration with container running Redis workload with YCSB for different records and read to write ratio.
- **Restoration time:** a key contributor in downtime (14.8% - 51%).
- Restore time for write intensive workload with 1M records is 2.2s

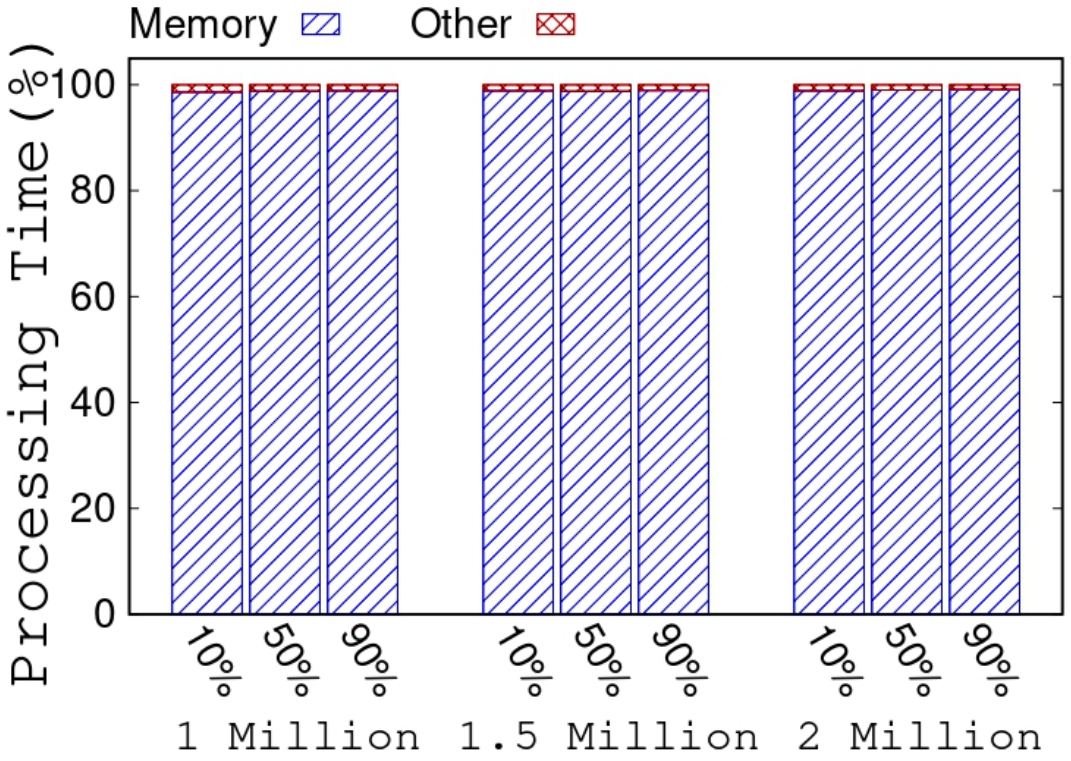


Split Cost of Downtime

Restoration has non trivial contribution towards downtime

Restoration: Significance of Memory State

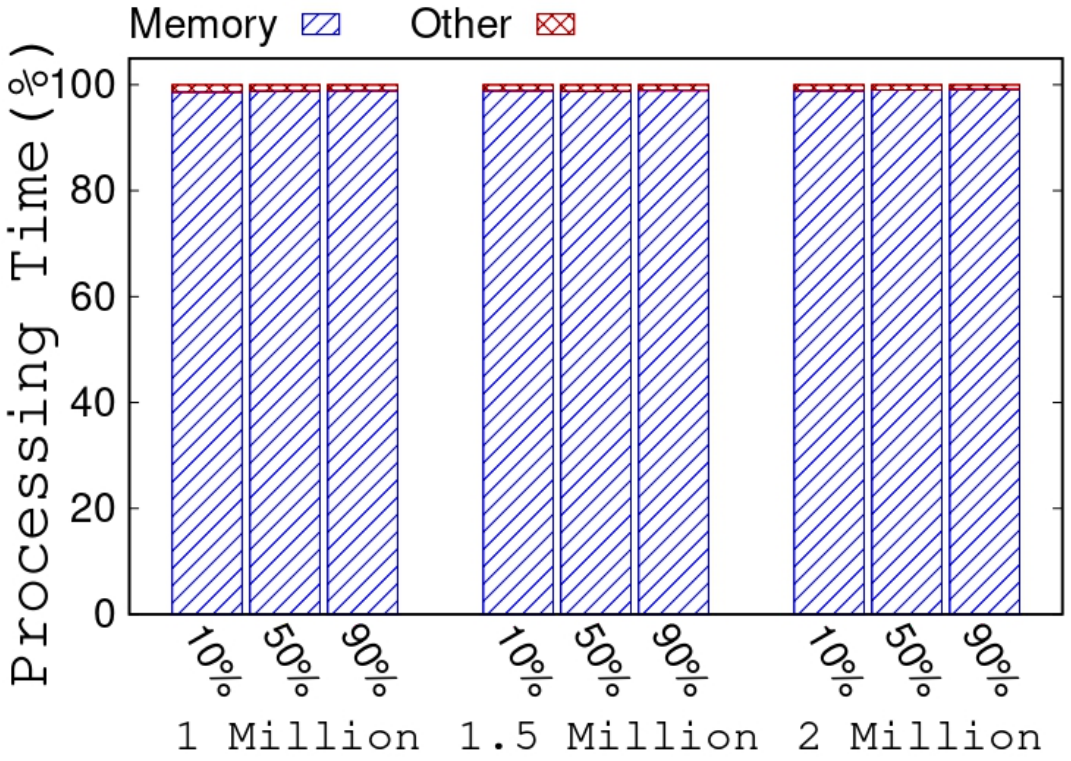
- Memory state processing dominates restoration time (99.5%) across all settings.



Restore processing cost for memory and others

Restoration: Significance of Memory State

- Memory state processing dominates restoration time (99.5%) across all settings.



Restore processing cost for memory and others

Iterative rebuild of memory state can significantly improve one shot restore

Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.
- Showcase the flexibility of PCLive to find a sweet-spot for resource overhead and downtime tradeoff.

Iterative Rebuild: Challenges

- Capture and transfer dependent container states for memory rebuilding efficiently.

Iterative Rebuild: Challenges

- Capture and transfer dependent container states for memory rebuilding efficiently.
- Maintaining relationship between different states.

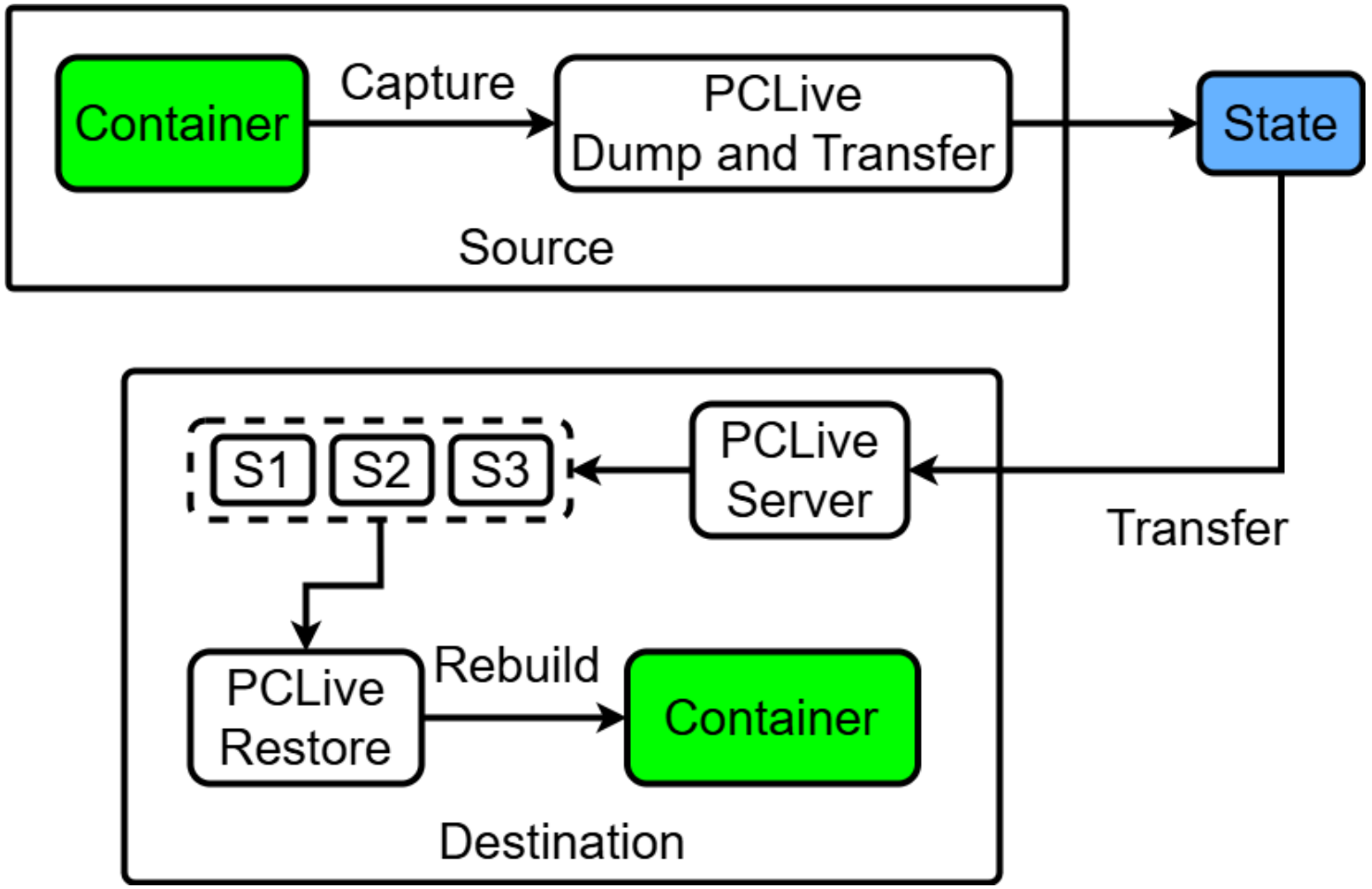
Iterative Rebuild: Challenges

- Capture and transfer dependent container states for memory rebuilding efficiently.
- Maintaining relationship between different states.
- The restoration activity should be non-intrusive and self-aware.

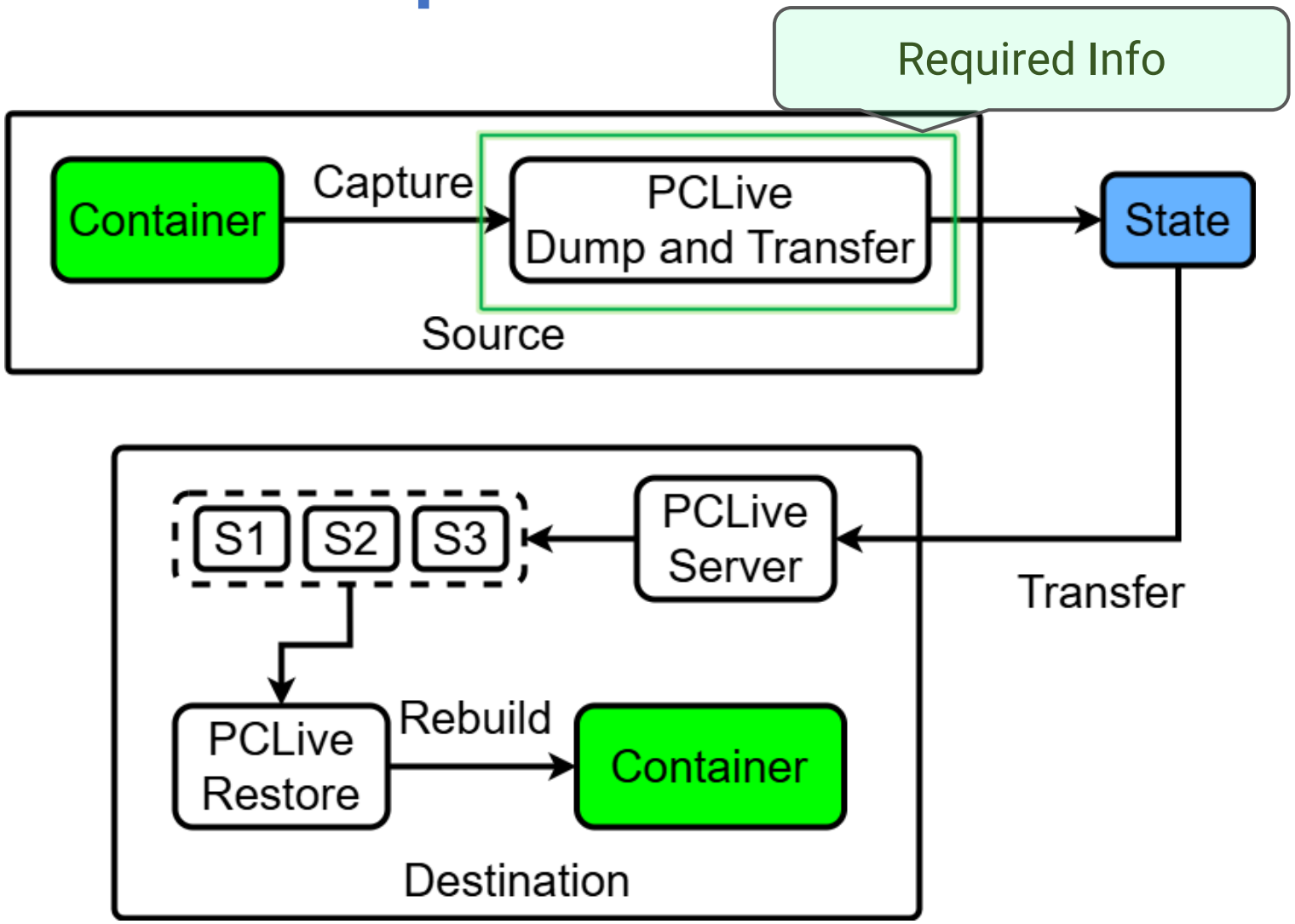
Iterative Rebuild: Challenges

- Capture and transfer dependent container states for memory rebuilding efficiently.
- Maintaining relationship between different states.
- The restoration activity should be non-intrusive and self-aware.
- The resource overheads should be comparable with one shot restore.

PCLive: Design

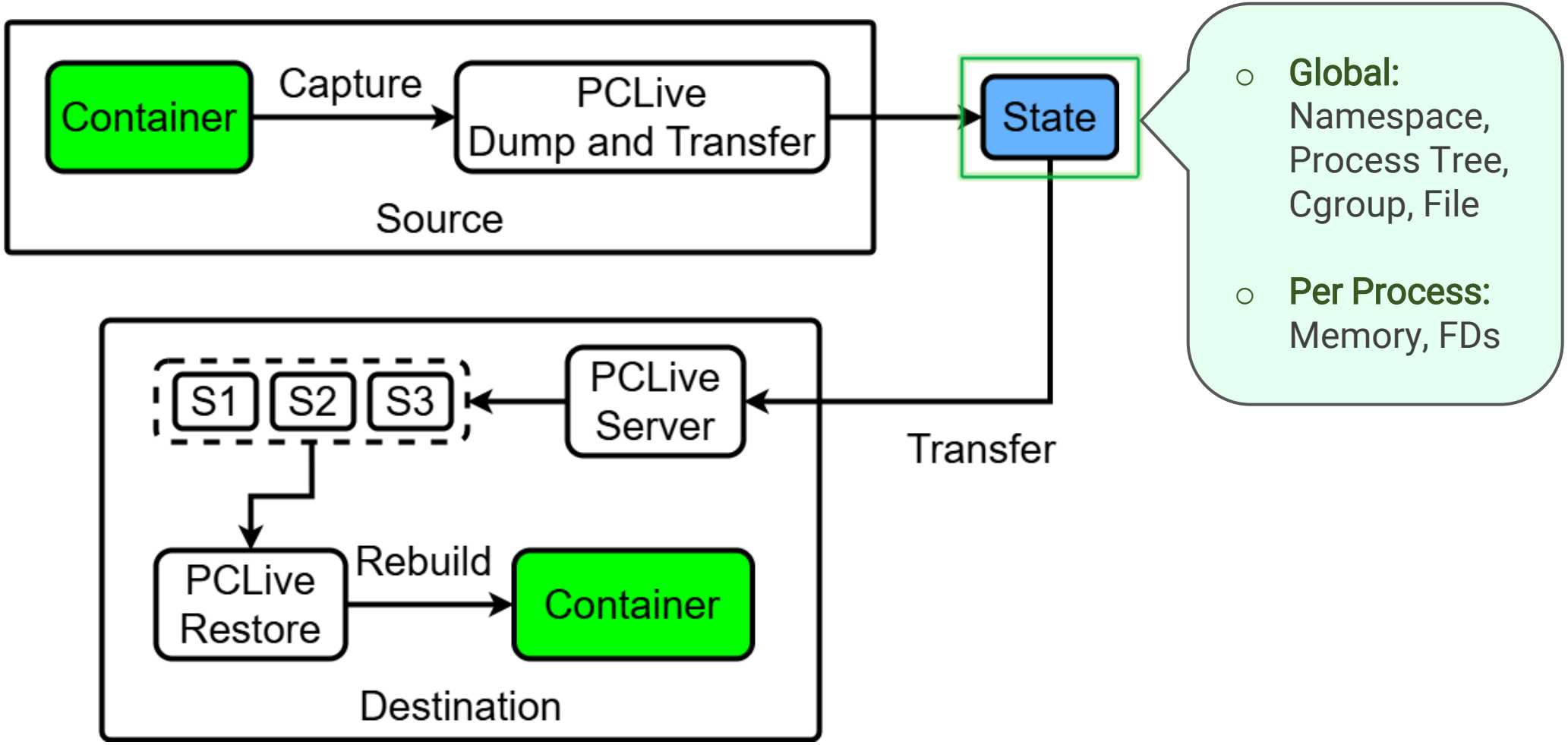


PCLive: Capture and Transfer

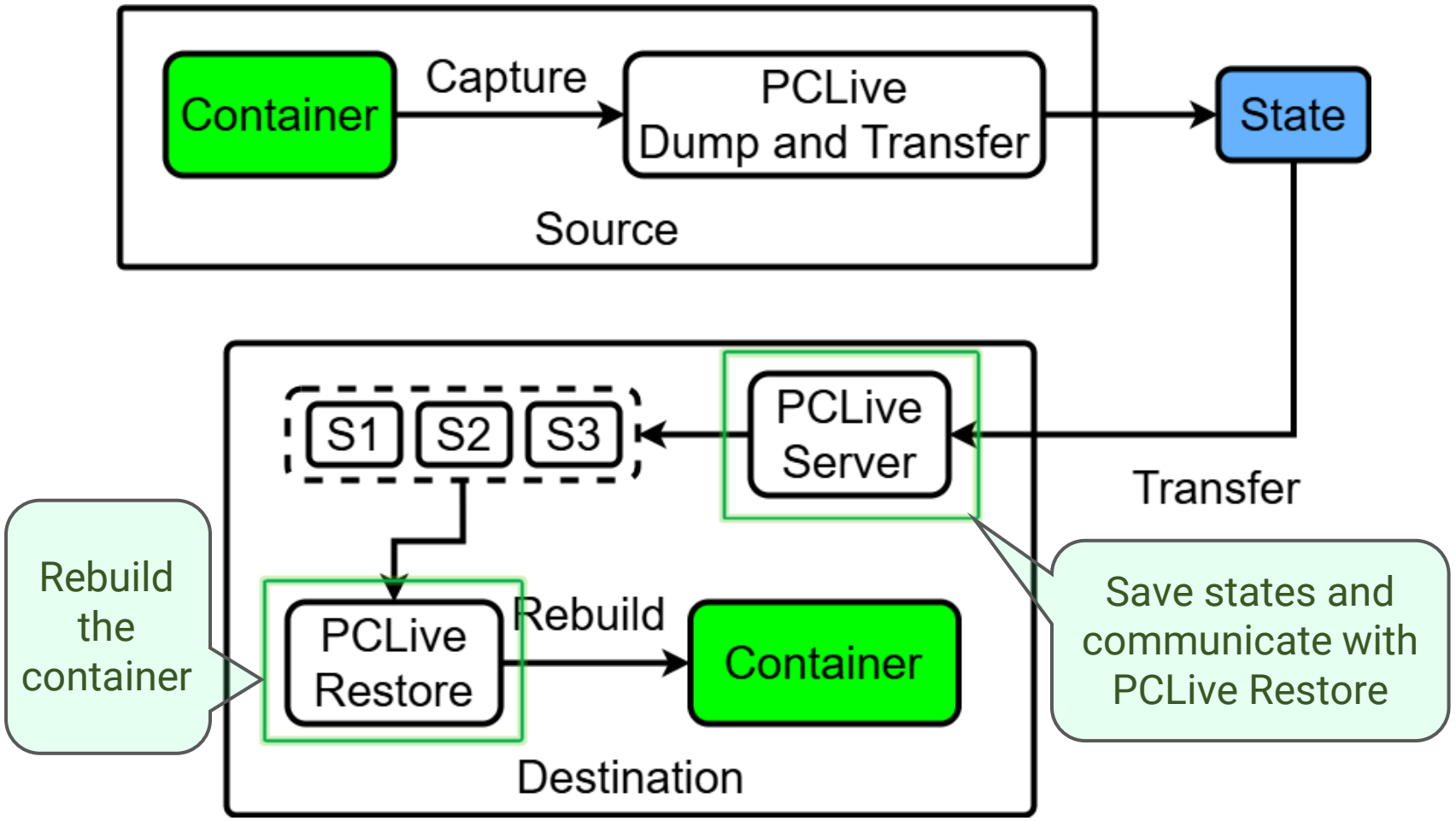


- Process Tree, Namespace, File, etc. along with memory mapping and its content.
- PCLive is configured to dump any sub-system.
- The freeze time has to be minimum.

PCLive: State

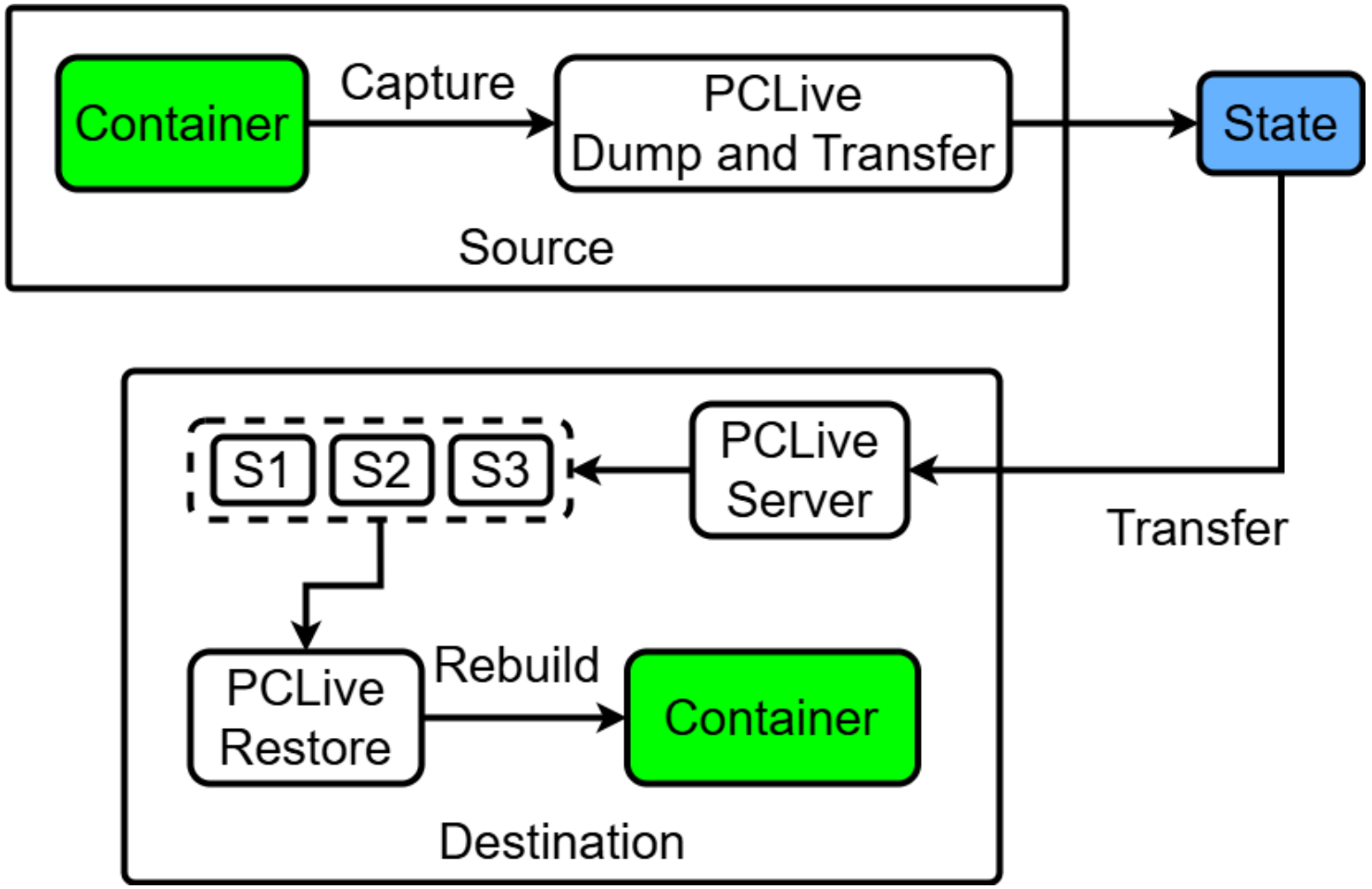


PCLive: Iterative Rebuild



- The restore can be started after any iteration (Delayed Restoration).
- Restore can be triggered after receiving global states (PCLiveG).

PCLive: Implementation

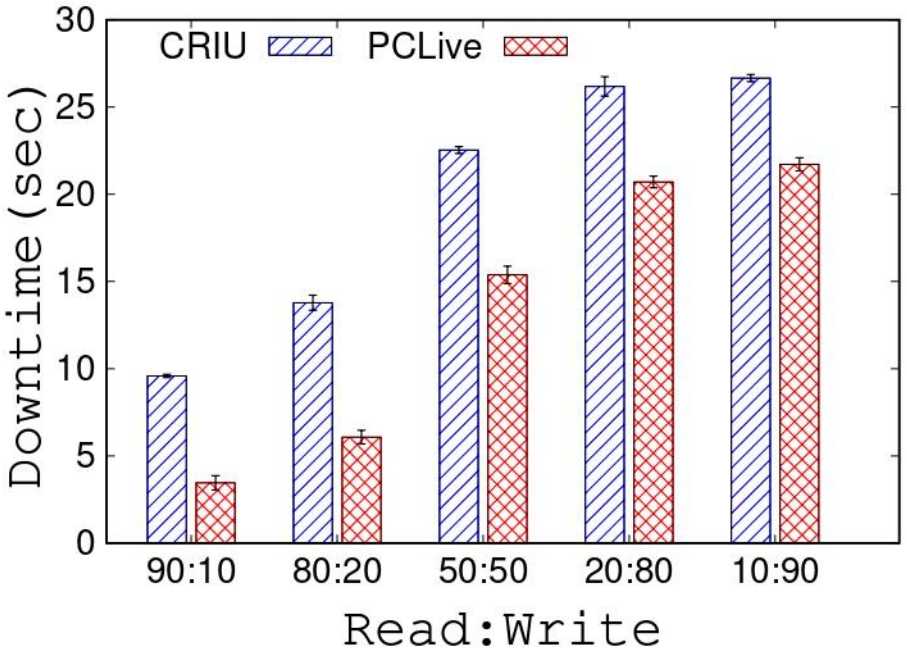


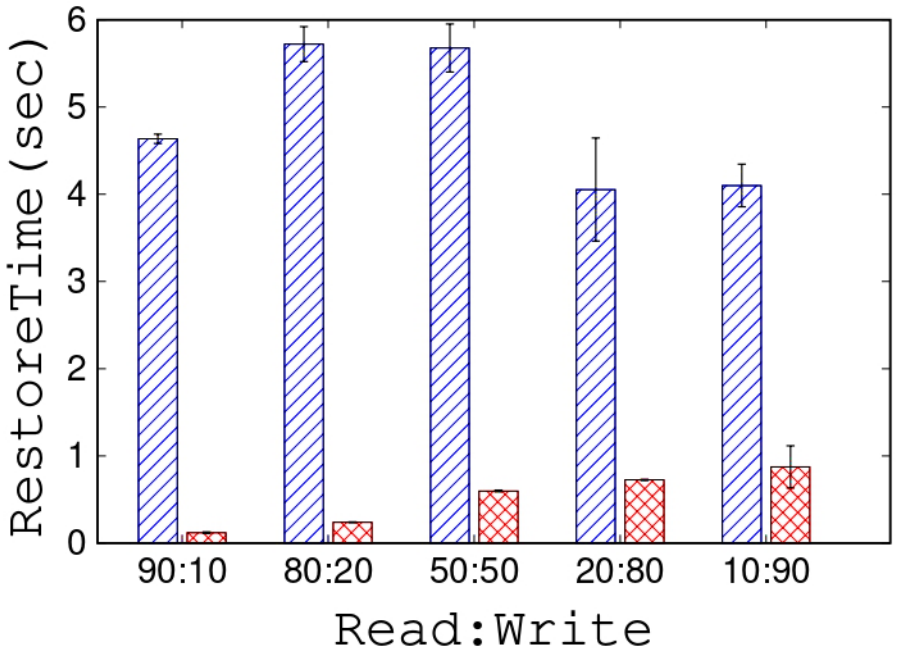
- Modified CRIU and runC to achieve pipelined restore.

Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.
- Showcase the flexibility of PCLive to find a sweet-spot for resource overhead and downtime tradeoff.

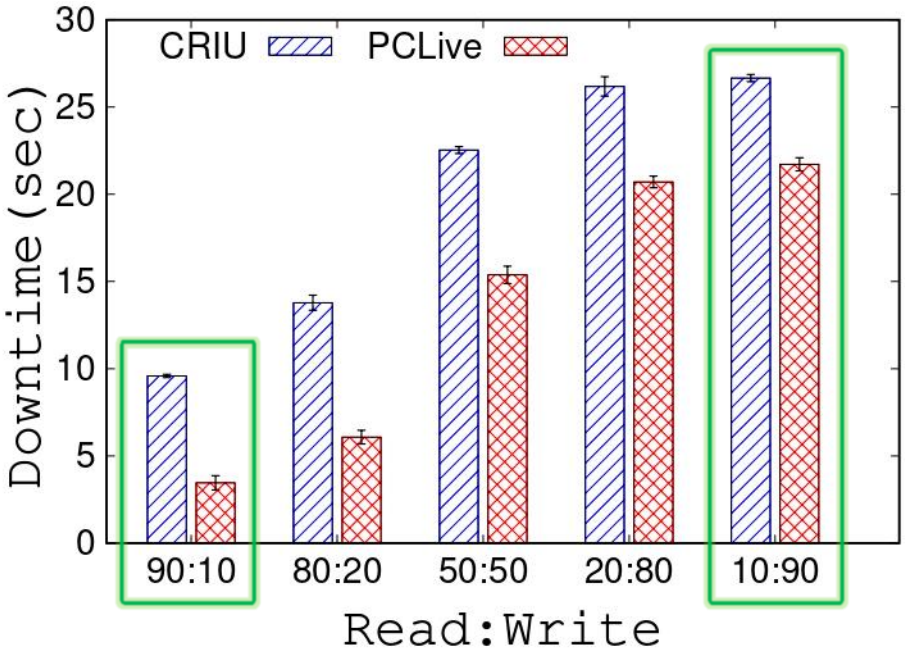
PCLive: Downtime Reduction

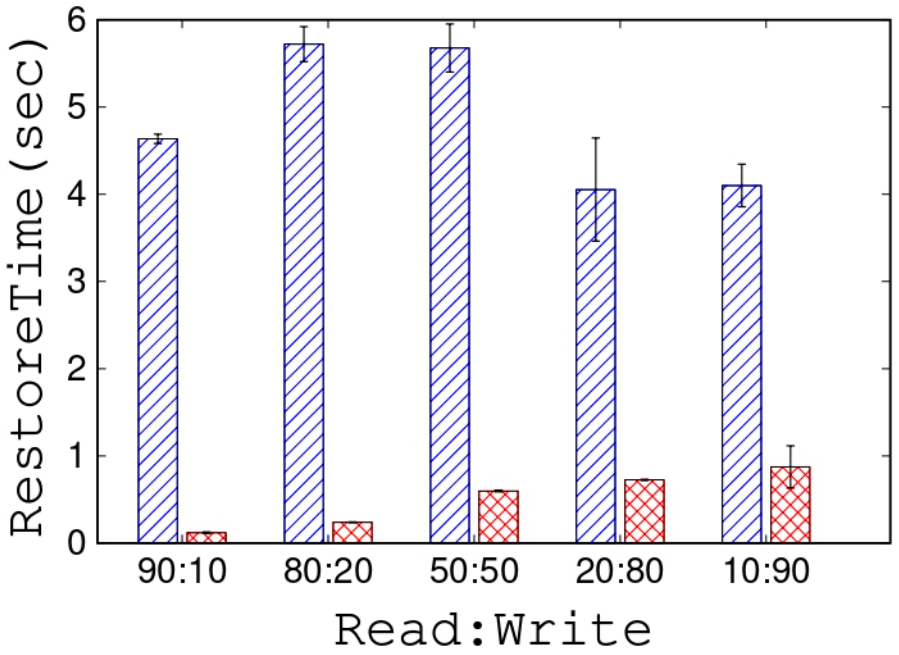




- Setup: Redis workload with YCSB for 2M records and different read to write ratio.

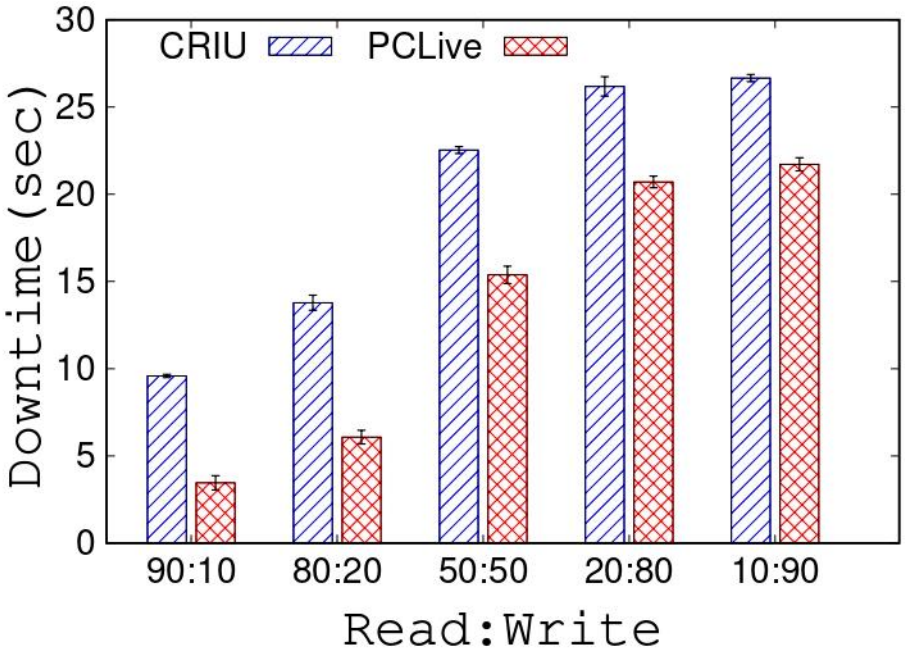
PCLive: Downtime Reduction

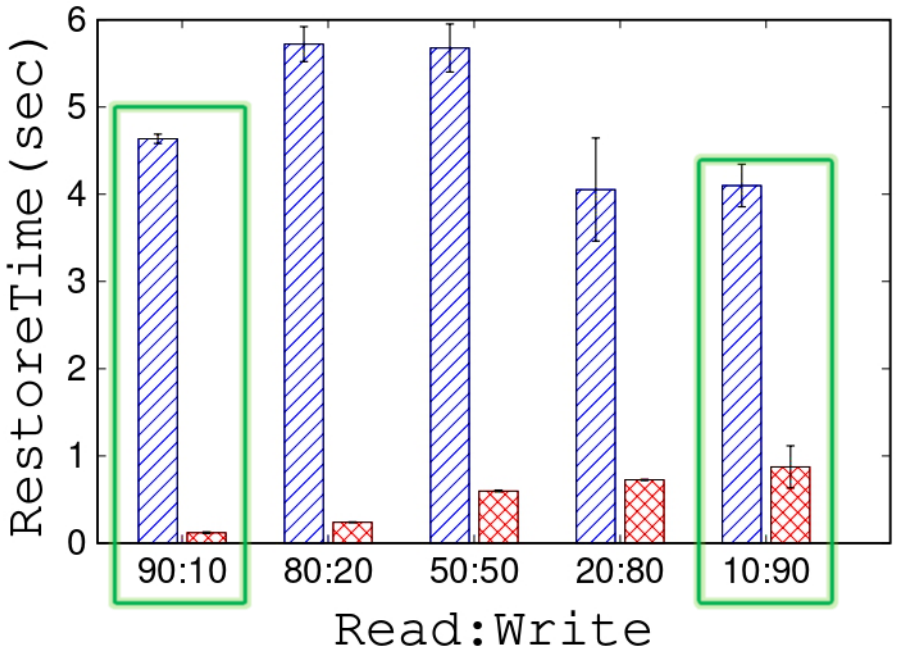




- Setup: Redis workload with YCSB for 2M records and different read to write ratio.
- Service downtime: 2.7x reduction for read intensive, 18% reduction for write intensive.

PCLive: Downtime Reduction





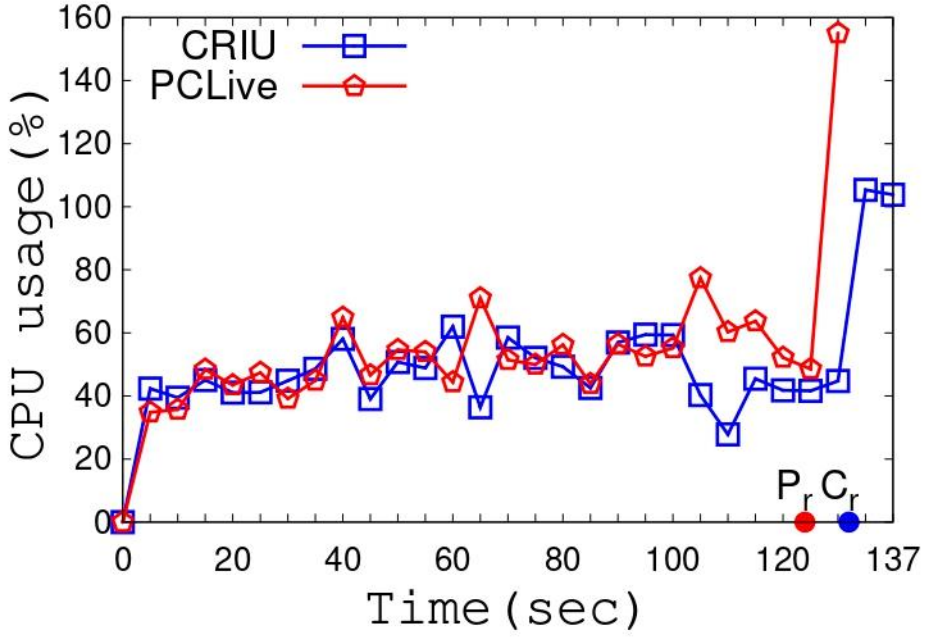
- Setup: Redis workload with YCSB for 2M records and different read to write ratio.
- Service downtime: 2.7x reduction for read intensive, 18% reduction for write intensive.
- Restore time: 38x reduction for read intensive, 5.4x reduction for write intensive.

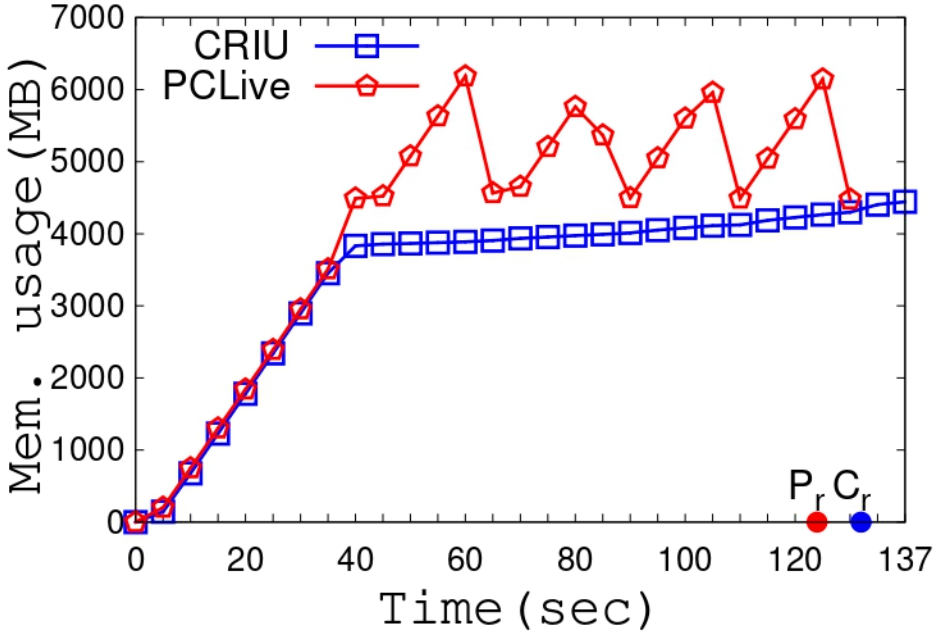
Contributions

- Empirically motivate the requirement to improve the restoration process for container migration.
- Identify and address the challenges to design and implement iterative rebuilding using PCLive.
- Demonstrate the benefits of PCLive in terms of downtime reduction.
- Showcase the flexibility of PCLive to find a sweet-spot for resource overhead and downtime tradeoff.

PCLive: CPU & Memory Utilizations

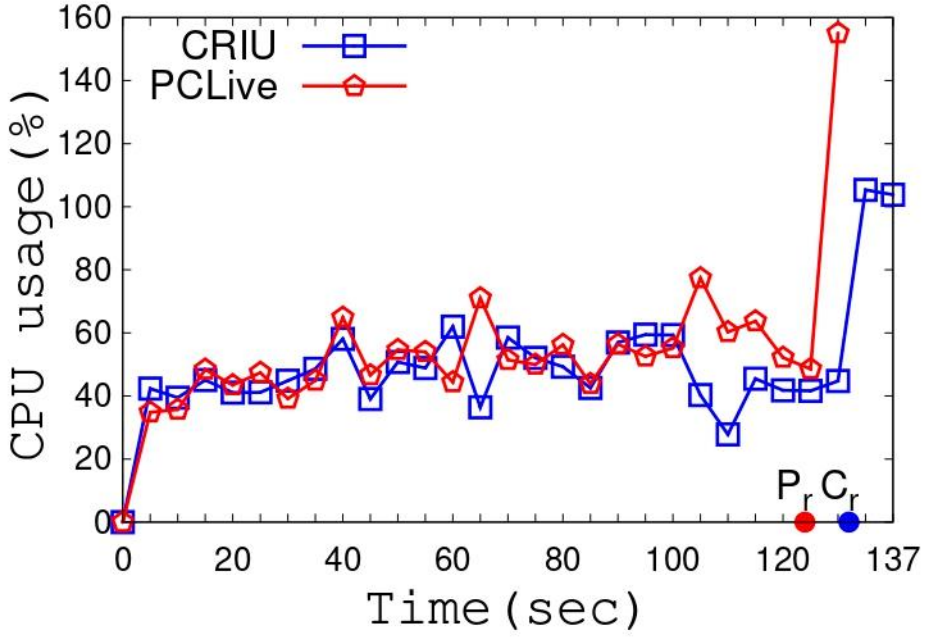
PCLive: CPU & Memory Utilizations

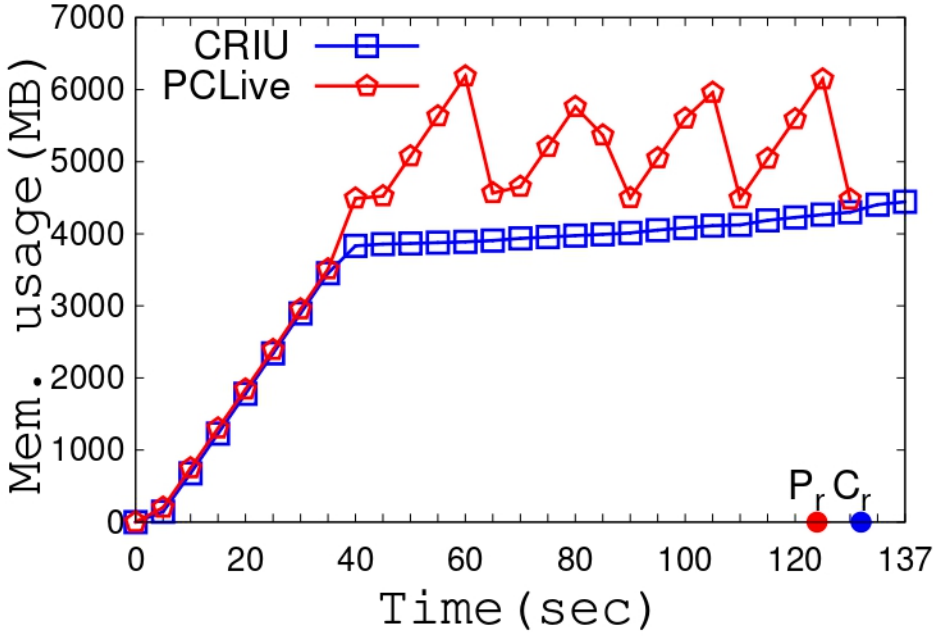




- Setup: Write intensive (10% Read) Redis workload with YCSB for 2M records.

PCLive: CPU & Memory Utilizations

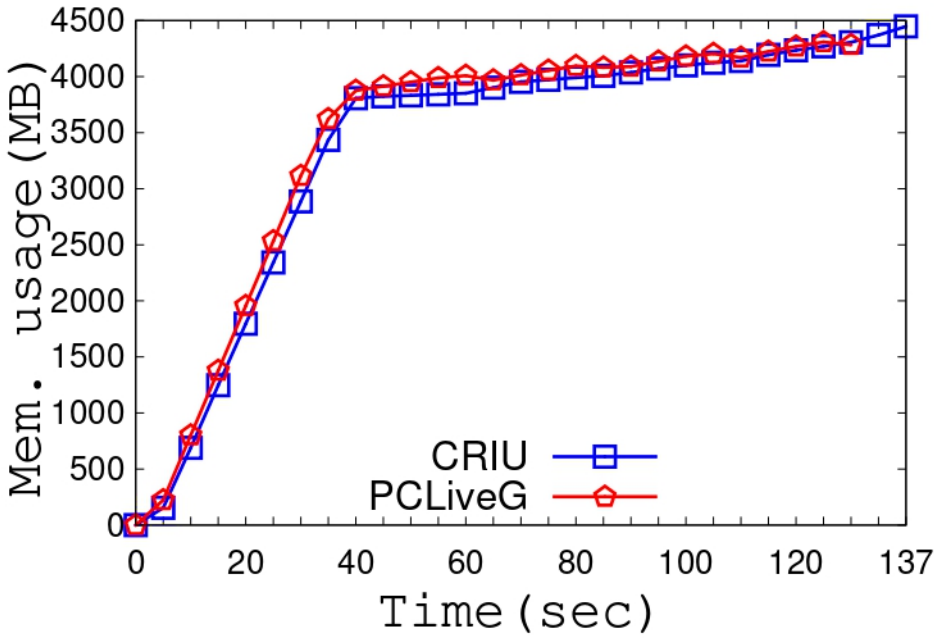




- Setup: Write intensive (10% Read) Redis workload with YCSB for 2M records.
- CPU utilization: 4% more for write intensive, similar for read intensive.
- Memory utilization: 23% more for write intensive, similar for read intensive.

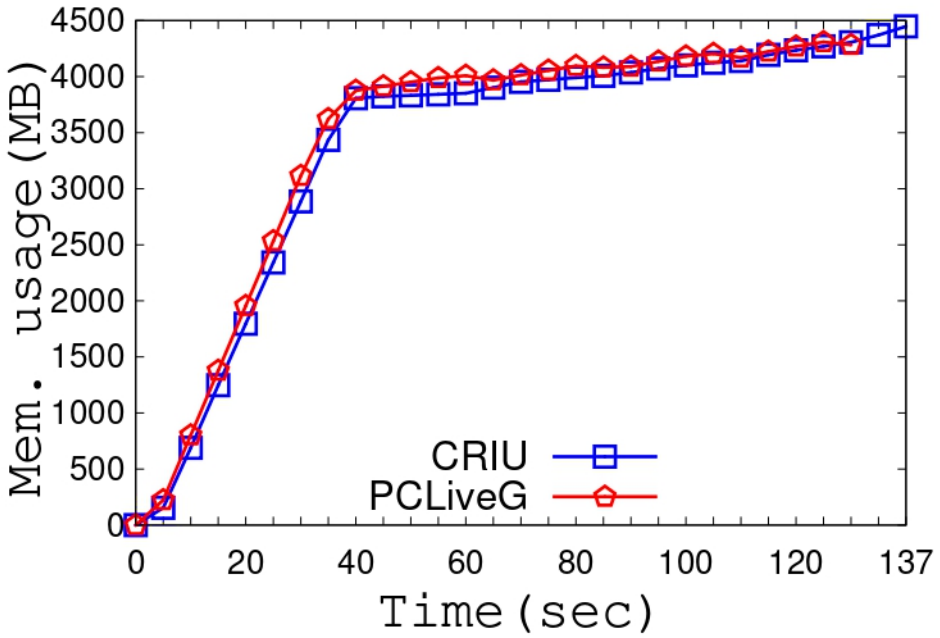
PCLive: Optimizing Memory Overhead

PCLive: Optimizing Memory Overhead



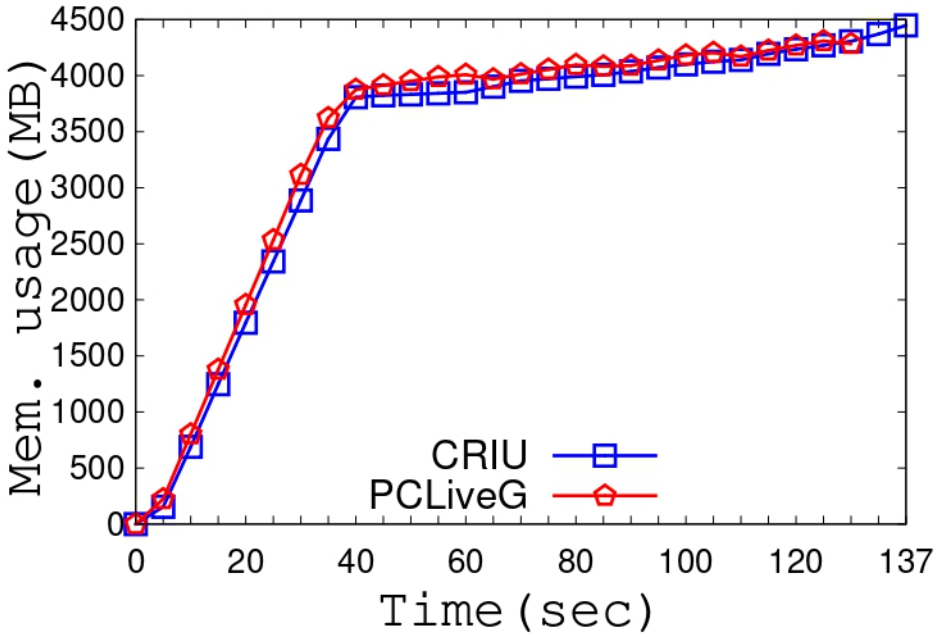
- Setup: Write intensive (10% Read) Redis workload with YCSB for 2M records.

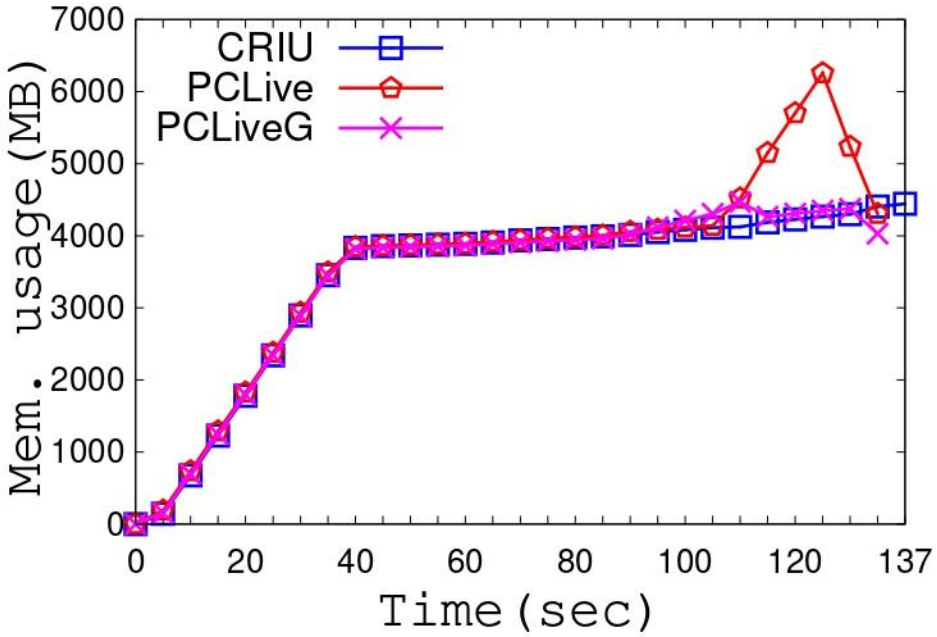
PCLive: Optimizing Memory Overhead



- Setup: Write intensive (10% Read) Redis workload with YCSB for 2M records.
- PCLiveG: Memory overhead is negligible (~100MB) for write intensive, CPU overhead is 13.5% - 21.8%. Restore time is reduced by more than 200ms.

PCLive: Optimizing Memory Overhead

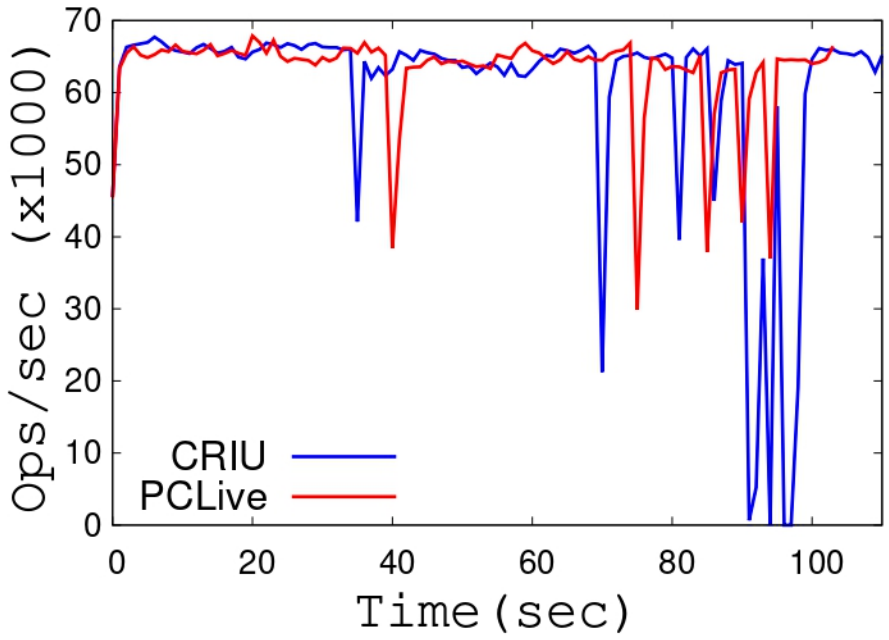


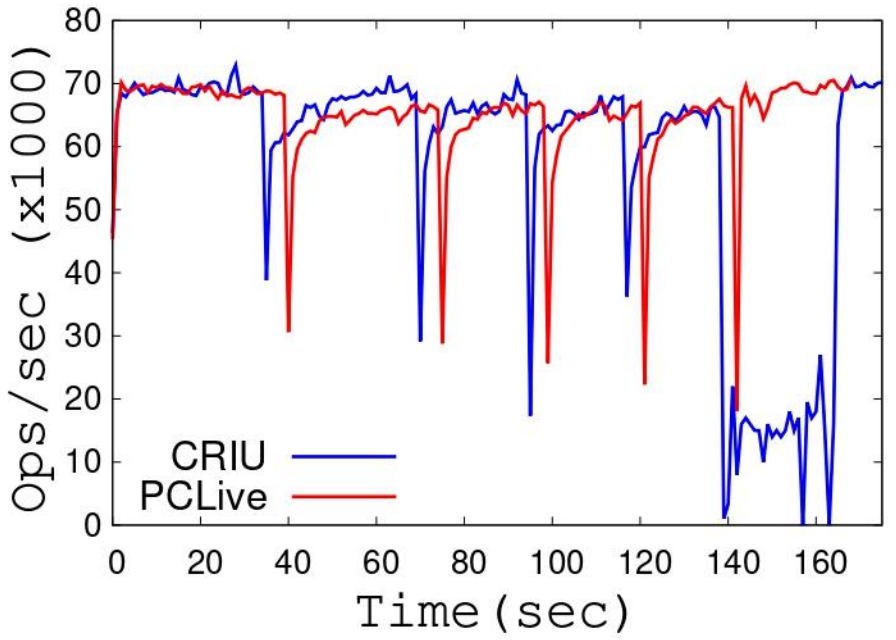


- Setup: Write intensive (10% Read) Redis workload with YCSB for 2M records.
- PCLiveG: Memory overhead is negligible (~100MB) for write intensive, CPU overhead is 13.5% - 21.8%. Restore time is reduced by more than 200ms.
- Delayed Restoration: Memory overhead is ~5% with similar CPU overheads.

PCLive: Application Throughput

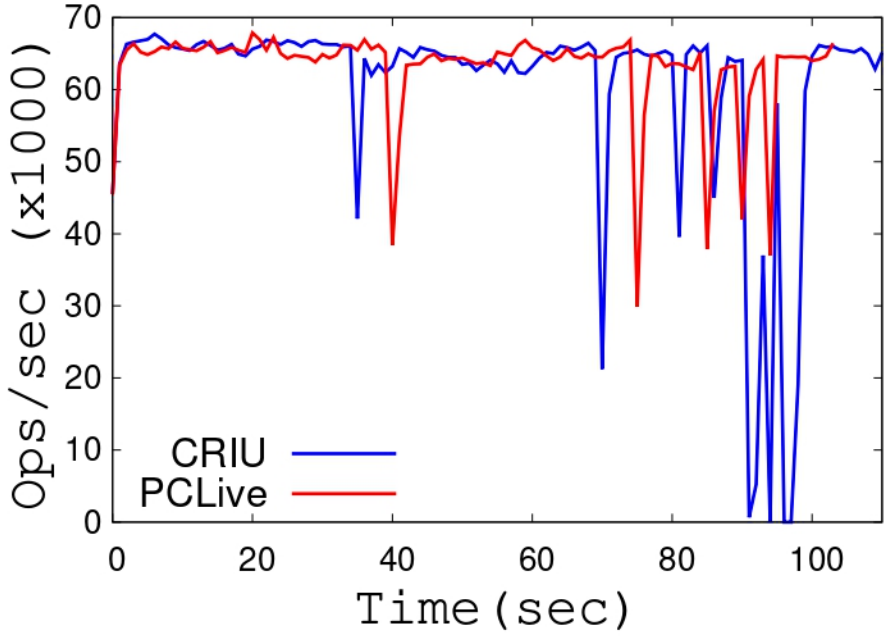
PCLive: Application Throughput

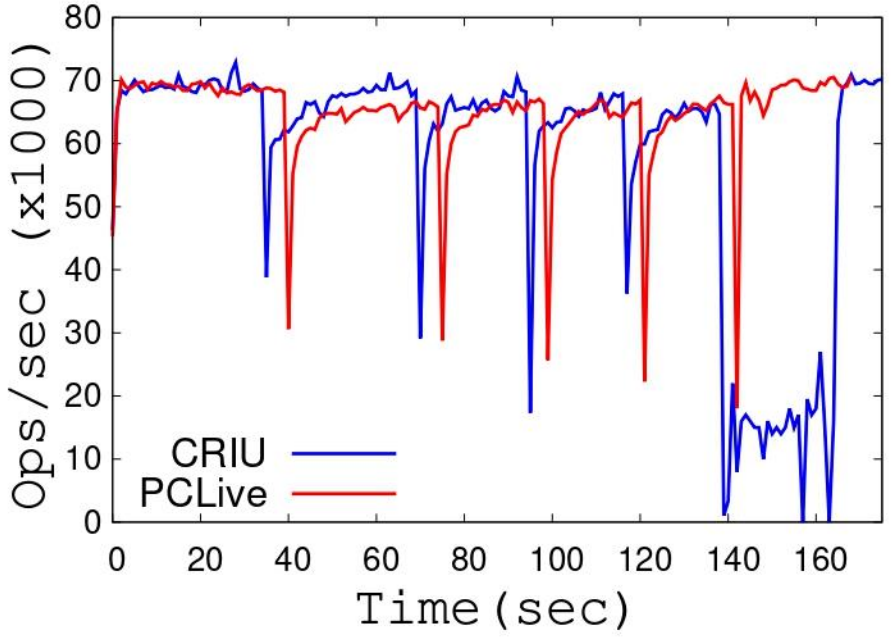




- Setup: Read and Write intensive Redis workload with YCSB for 2M records.

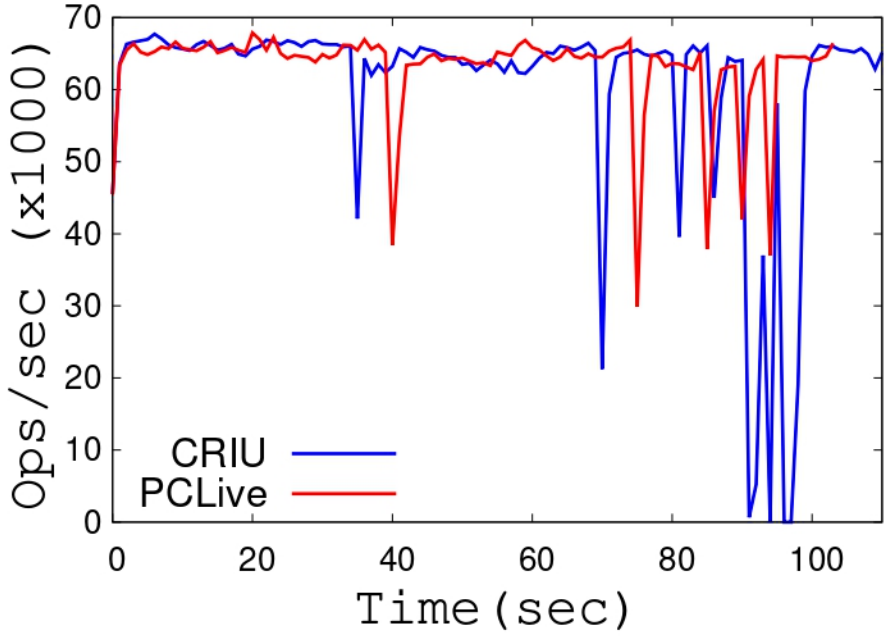
PCLive: Application Throughput

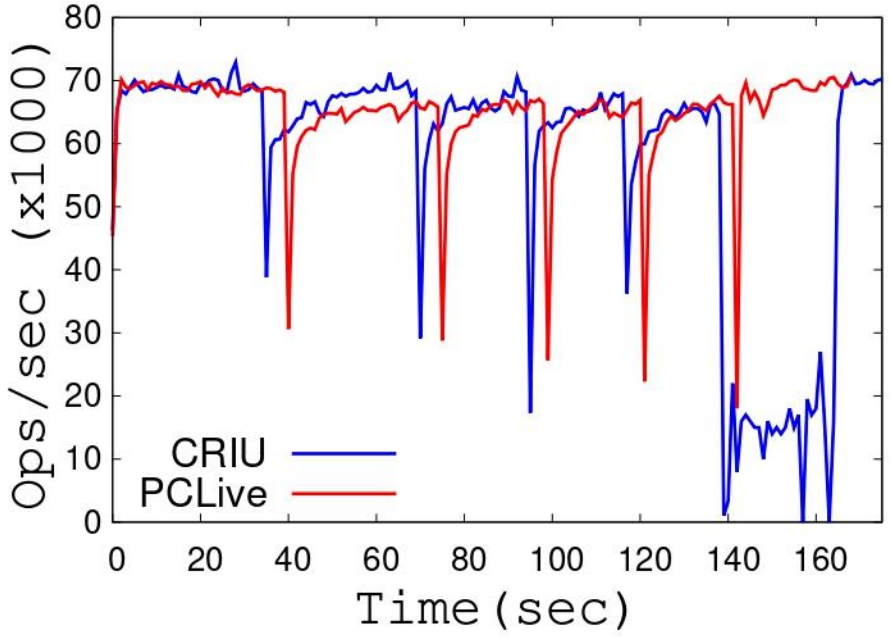




- Setup: Read and Write intensive Redis workload with YCSB for 2M records.
- PCLive improves the application throughput during stop-and-copy phase.

PCLive: Application Throughput





- Setup: Read and Write intensive Redis workload with YCSB for 2M records.
- PCLive improves the application throughput during stop-and-copy phase.
- With PCLive, sometime throughput drop is slightly more during pre-dump iterations.

PCLive: Other Evaluations

- With high–speed network, PCLiveG performs better, e.g., 5.4x downtime improvement for 10Gbps.

PCLive: Other Evaluations

- With high–speed network, PCLiveG performs better, e.g., 5.4x downtime improvement for 10Gbps.

- With PCLive, the exclusive restore time increases with increase in write intensity.

PCLive: Other Evaluations

- With high–speed network, PCLiveG performs better, e.g., 5.4x downtime improvement for 10Gbps.
- With PCLive, the exclusive restore time increases with increase in write intensity.
- With PCLiveG, the exclusive restore time remains constant and similar to VM live migration. It is also independent of write intensity.

PCLive: Other Evaluations

- With high–speed network, PCLiveG performs better, e.g., 5.4x downtime improvement for 10Gbps.
- With PCLive, the exclusive restore time increases with increase in write intensity.
- With PCLiveG, the exclusive restore time remains constant and similar to VM live migration. It is also independent of write intensity.
- PCLive is also evaluated with Benchbase (MySQL) and Graph500 workload.
- Please refer to the paper for more details.

Conclusion

- PCLive addresses container migration issue with iterative pre-copy strategy by pipelining the rebuild process at the destination machine.

Conclusion

- PCLive addresses container migration issue with iterative pre-copy strategy by pipelining the rebuild process at the destination machine.

- PCLive results in up to 38x reduction in restoration time and 2.7x reduction in service downtime as compared with baseline CRIU.

Conclusion

- PCLive addresses container migration issue with iterative pre-copy strategy by pipelining the rebuild process at the destination machine.
- PCLive results in up to 38x reduction in restoration time and 2.7x reduction in service downtime as compared with baseline CRIU.
- PCLive addresses CPU and memory overhead with techniques such as PCLiveG and Delayed Restoration.

Conclusion

- PCLive addresses container migration issue with iterative pre-copy strategy by pipelining the rebuild process at the destination machine.
- PCLive results in up to 38x reduction in restoration time and 2.7x reduction in service downtime as compared with baseline CRIU.
- PCLive addresses CPU and memory overhead with techniques such as PCLiveG and Delayed Restoration.
- With PCLiveG, the application container migration become similar to VM migration.

Thank You

Documentation and Source code:
https://www.github.com/shivbt/PCLive

Contact:
shivbt@cse.iitk.ac.in