Using Container Migration for HPC Workloads Resilience

Mohamad Sindi & John R. Williams

Massachusetts Institute of Technology
Center for Computational Engineering (CCE)

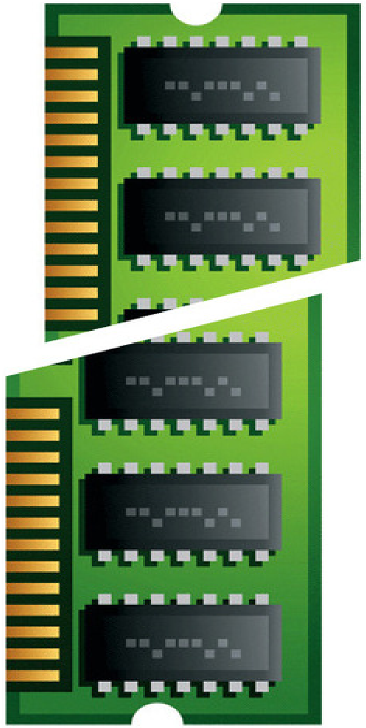
HPEC’19, Sept 26 2019

Agenda

- The issue
- Proposed mitigation method
- Demo
- Main contributions & summary

The Issue

- Today's top HPC supercomputers are running in Petascale computing power (thousands of nodes, several millions of cores).
- Mean Time Between Failures (MTBF) for some of today's top HPC Petaflop systems is reported to be several days.
- Exascale computing is expected by 2020-2021 (billion cores).
- Some studies estimate MTBF for Exascale systems to be less than 60 minutes.
- Running sustainable workloads on such systems is becoming more challenging as the size of the HPC system grows.



Current Methods to Tolerate Failures

- **Checkpoint-restart (CR)** mechanism is commonly used (application periodically saves its state, it can restart from last checkpoint incase of failure).
- Popular tool for this is Berkeley Lab Checkpoint/Restart (**BLCR**).

Limitations of CR

- High overhead (performance, storage space, etc.)
- Studies estimate that future Exascale systems could have a MTBF smaller than the time required to complete a CR process.
- CR is a reactive method, it will remedy the fault after the fact that your workload had failed.

Proposed Solution

Proactively predict failures, then remedy the situation before failure occurs, without impacting performance.

Proposed Solution

- Design a container-based proactive fault tolerance framework to improve the sustainability of running workloads on Linux HPC clusters.

- The framework mainly serves 2 objectives:

1. Predict potential compute node hardware failures.

   (not the scope of this presentation, but detailed in PhD thesis)

2. Remedy the situation once faults predicted, with minimal overhead on the running HPC workloads.

   (The focus of this presentation)

Remedy Environment

Container Technology:

OpenVZ

- We propose using the Linux container technology to perform workload migrations once failures are predicted
- Allows us to self-contain HPC application and its required libraries
- Reduces the coupling of the workload from physical hardware
- Proven its success as a scalable and lightweight technology for micro services used in large scale data centers

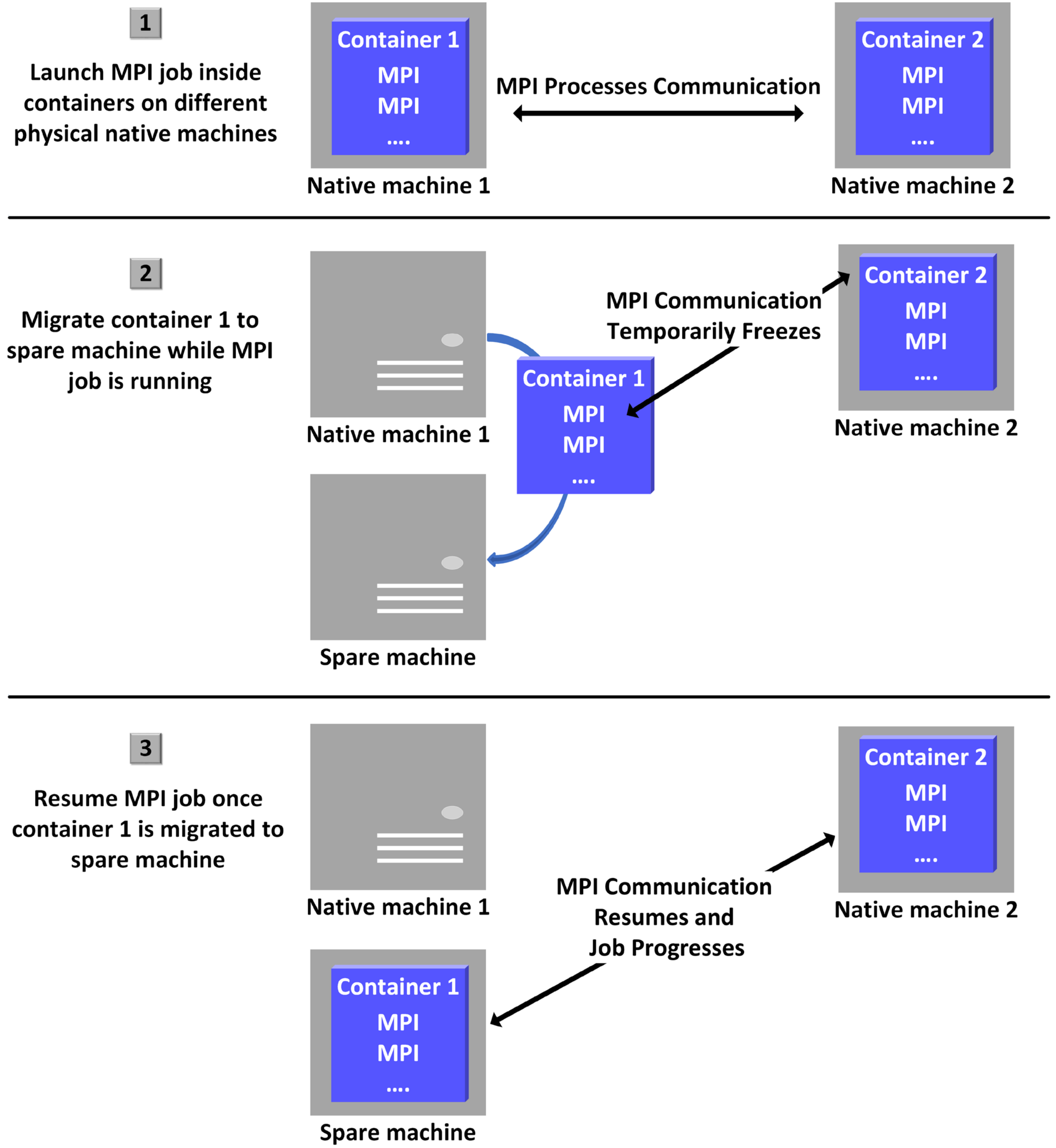
  (e.g. Google’s data centers run most of their micro services on containers)
- We adapt potential resilience capabilities of containers towards HPC workloads

Work Summary

Objective - Remedy Once Faults Predicted:

- In summary:
  - We setup a complete HPC environment that is container-based.
  - Tested it with 6 real HPC applications.
  - Applications use Message Passing Interface (MPI) de facto standard for HPC.
  - **We were able to successfully do container migration for all HPC applications (after resolving numerous technical challenges).**
  - Performed comprehensive performance benchmarks comparing container vs. native. **Container performance was almost native.**

Concept of Migration

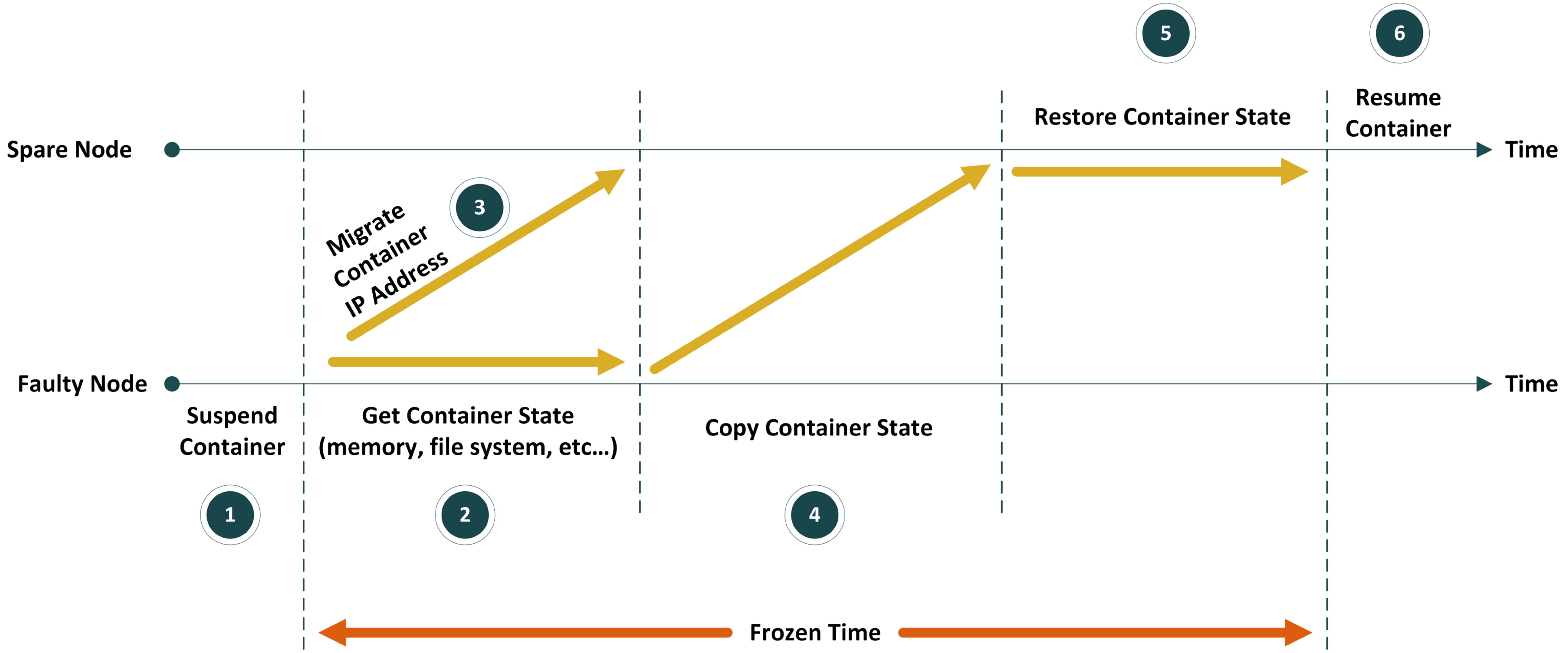


Migrating Containers

- CRIU Open-source Library:

- A tool that can be used to freeze/unfreeze processes running on Linux in user space
- May be applied to freeze/unfreeze containers
- At time of testing, was still beta with Linux RedHat 7 (no official support, buggy)
- Had to debug and modify some of the library’s source code to work in our HPC environment (code modification to fix issue with NFS mounts inside containers).

Migration Steps



Testing Real HPC Applications in Containers

Applications use MPI, **no need to modify code or binary executable**

Application	Main Developers	Language	Domain	Visualization
OSU Micro-Benchmarks	Ohio State University	C	MPI benchmark for network bandwidth and latency	NA
Palabos	- Academic: University of Geneva - Industry: FlowKit CFD	C++	CFD/Complex Physics using lattice Boltzmann method	Post-process In-situ
Flow	Open Porous Media Initiative (Oil companies + Academia)	C++	Reservoir simulation	Post-process
Fluidity	Imperial College London	Fortran, C++	- CFD solving Navier-Stokes - Geophysical fluid dynamics - Ocean Modelling - Adaptive unstructured mesh	Post-process
GalaxSee	- Shodor Education Foundation - National Center for Supercomputing Applications - George Mason University	C++	N-body galaxies simulation	In-situ
ECLIPSE	Schlumberger (commercial)	Fortran	Industry-reference reservoir simulator	Post-process

Testing Real HPC Applications in Containers

- Test using various AWS hardware platforms (# cores, memory, network):
  - Low spec nodes: (4 physical cores, 32 GB RAM, 1 Gig network)
  - Med spec nodes: (18 physical cores, 72 GB RAM, 10 Gig network)
  - High spec nodes: (32 physical cores, 256 GB RAM, 25 Gig network)
    (36 physical cores, 512 GB RAM, 25 Gig network)
- Test migration with various MPI libraries: MPICH, Open MPI, Intel MPI
- MPI job sizes ranged from 4 to 144 processes.

Testing Real HPC Applications in Containers

Important questions to answer during container testing:

1. Will application performance be impacted? (container vs. native)

2. Can we actually migrate containers with MPI processes without affecting HPC job?

3. Will produced results be intact?

(no data corruption due to migration)

Application Performance Summary

- More than 130 test runs were performed for the study using the various HPC applications and hardware platforms.
- OSU benchmarks (point-to-point and collective):
  - Network latency overhead was **~6.8%** on average with containers.
  - Network bandwidth overhead was **~3.9%** on average with containers.
- However, performance overhead of the real HPC applications was very negligible and close to native performance (**%0.034** on average).
- Worst case application performance overhead was **%0.9**
- Overall, the performance on containers was acceptable for all HPC applications tested (almost native).

Migration!

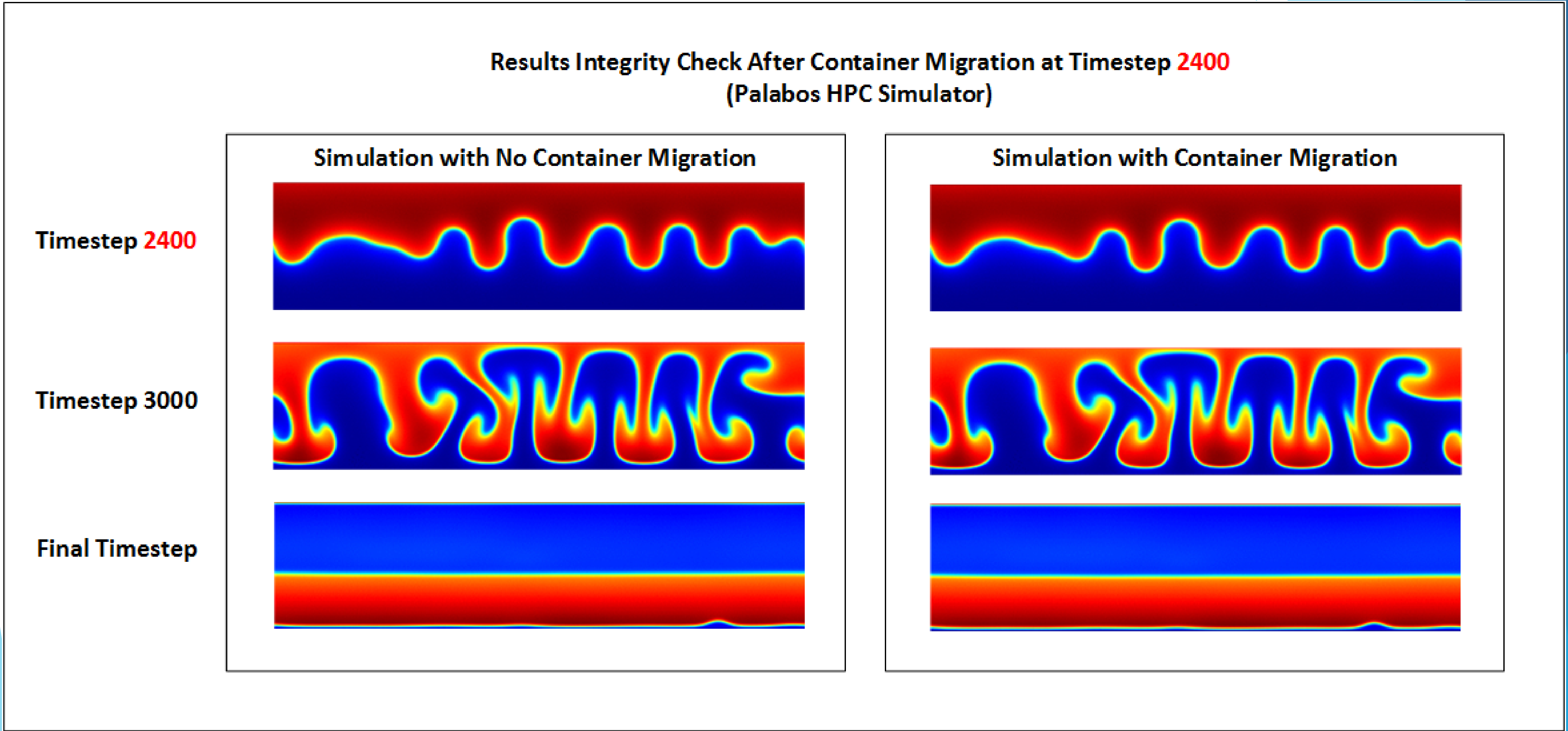
Migration Behavior

- Average container migration time was **34 seconds** (using standard SSD disk)
- Migration time mainly influenced by size of app binaries stored inside container.
- In the case of Palabos, GalaxSee, and ECLIPSE, application binaries were stored on shared NFS storage and not inside container.
- When testing with 10G/25G network, migration time was still the same!
- Bottleneck is not network, but speed of local hard disk storing container data.
- Testing with faster SSD hard disk reduced avg migration time to **22 seconds**.

Container Application	Migration Time (seconds)
Fluidity	50
Flow	35
Palabos	30
GalaxSee	29
ECLIPSE	26

Migration Behavior

Example of checking results integrity:



Demos (available on YouTube)

Palabos: Migrate container while MPI/visualization job is running.

More YouTube demo scenarios for the various applications tested are available in paper and PhD thesis.

Application	Demo Video Link
Palabos	https://youtu.be/1v73E2Ao3Mk

Main Contributions & Summary

1. To the best of our knowledge, this work is the first in the HPC domain to demonstrate successful migration of MPI-based real HPC workloads using containers and CRIU.

2. Performed comprehensive performance benchmarks on containers using real HPC workloads on multiple computing platforms.

3. Using containers in HPC is a young topic, the challenges we faced and the solutions adopted are valuable experiences to share with the HPC community.

Thank You!

Questions?