Comparison of container virtualization tools for utilization of idle supercomputer resources

Julia Dubenskaya, Stanislav Polyakov, Minh Duc Nguyen, Elena Fedotova

Skobeltsyn Institute of Nuclear Physics, Moscow State University

Motivation

- Supercomputer time is expensive
- Some of the supercomputer resources remain idle. Reasons:
  - policy
  - jobs' requirements
  - a lack of jobs
- Problem: usually supercomputer is underloaded
- Purpose: increase the load on supercomputer forcing the use of idle resources

Idea

- Create an additional queue of low-priority jobs
- Load idle resources with these jobs
- Upon arrival of a regular job:
  - interrupt the execution of low-priority jobs
  - wait for the appearance of new idle nodes
  - resume low-priority jobs there
- Low-priority jobs required characteristics:
  - small, not resource-intensive
  - not urgent

Implementation

- Container virtualization: to isolate the process
- CRIU - Checkpoint/Restore In Userspace:
  - creates a stateful checkpoint
  - stops a running process
  - the process can later be restored from the moment it was interrupted
  - restoring is possible on the same computational node or on another
- Result: a low-priority job can be completed in several stages

Container virtualization tools comparison

- Goal:
  - compare different container virtualization tools with CRIU support
  - choose one that is most suitable for our task
- Alternatives considered (virtualization tool + file system):
  - Docker + Ext4
  - LXC + ZFS

Testbed

- Computer, CPU with 2 cores:
  - product: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
  - capacity: 2401MHz
  - width: 64 bits
- Computer, memory:
  - size: 15 GiB
- Test job:
  - 100% CPU load
  - about 1.5 GiB of memory

Docker vs LXC. Timing

Container launch

	Docker	LXC	
	create/start container	create container	start container
Time, sec	1.5 ± 2	14 ± 2	1.5 ± 2

Checkpoint and restore operations

	Docker		LXC	
	checkpoint	restore	checkpoint	restore
Time, sec	70 ± 3	62 ± 3	13 ± 2	5.5 ± 0.5

Docker vs LXC. Final choice

- Unexpected LXC feature:
  - the same container can be correctly restored from a checkpoint only once
  - Otherwise - error with the loss of the container state
- Our idea assumes a multiple checkpoint and restore of the same container
- Conclusion:
  - We have chosen Docker which stably and correctly checkpoints/restores any container multiple times

Applications of the results

- We implemented a prototype system using Docker containers
- Testing of the prototype proved the reliability and stability of the proposed approach
- Hope that the results obtained can be useful to other researchers when choosing a container virtualization tool for their needs

Thank you for the attention!

Questions?