Performance Evaluation of Containers for HPC

Cristian Ruiz, Emmanuel Jeanvoine and Lucas Nussbaum

INRIA Nancy, France

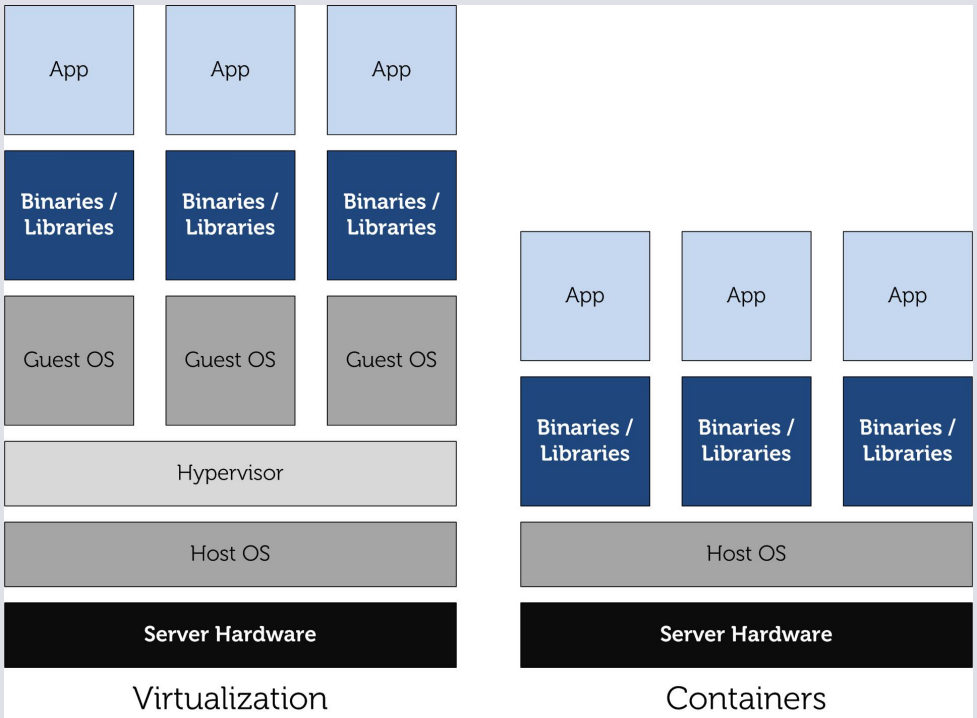
VHPC'15

Outline

Outline

Containers

Containers refers generally to Operating-system-level virtualization, where the kernel of an operating system allows for multiple isolated user-space instances.

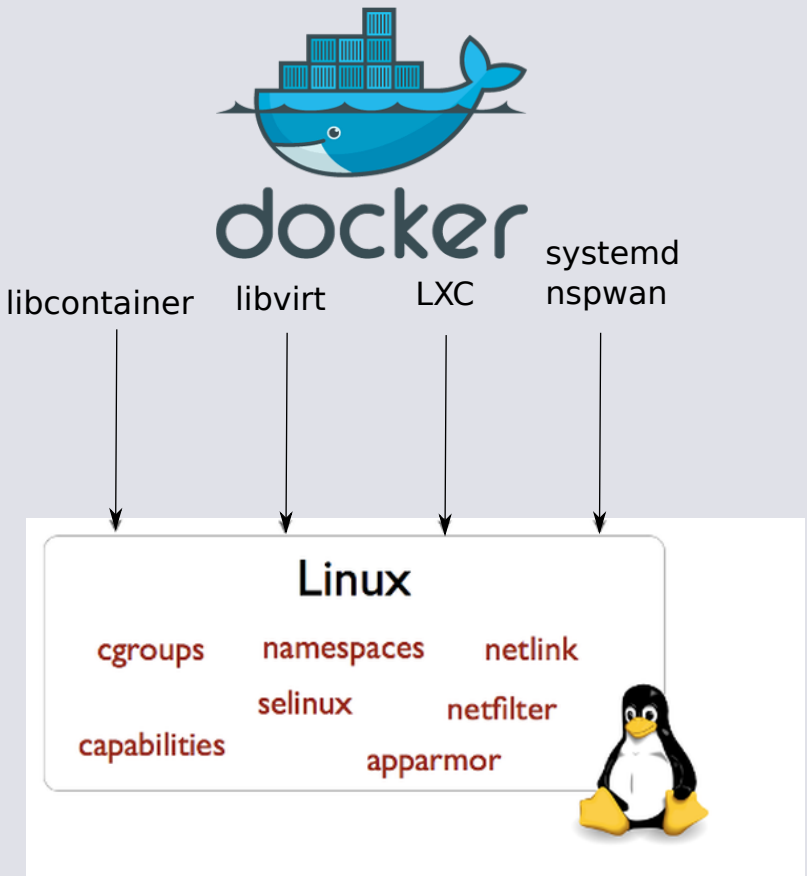


Implementations

- Chroot
- Linux-VServer
- FreeBSD Jails
- Solaris Containers
- OpenVZ

namespaces and cgroups

- Both features incorporated in Linux kernel since 2006 (Linux 2.6.24)
- Several container solutions: LXC, libvirt, libcontainer, systemd-nspawn, Docker



Outline

Benefits of using containers in HPC

- Containers allow to easily provision a full software stack. They bring:
  - portability
  - user customization
  - reproducibility of experiments
- Containers provide a lower oversubscription overhead than full vms, enabling:
  - a better resource utilization
  - to be used as a building block for large scale platform emulators

Container performance evaluation

- Matthews et al[3] compared the performance of VMWare, Xen, Solaris containers and OpenVZ using custom benchmarks
- Felter et al[2] evaluated the I/O performance of Docker using MySQL, Linpack, Stream, RandomAccess, nuttcp, netperf, fio, and Redis
- Walter et al[4] compared VMWare Server, Xen and OpenVZ using NetPerf, IOZone, and the NAS Parallel Benchmarks
- Xavier et al[5] compared Linux VServer, OpenVZ, LXC and Xen using the HPC Challenge benchmarks and the NAS Parallel Benchmarks

In this work, we answer:

- What is the overhead of oversubscription using different versions of Linux kernel?
- What is the performance of inter-container communication?
- What is the impact of running an HPC workload with several MPI processes inside containers?

Outline

Experimental setup

Hardware

- Cluster in Grid'5000 Testbed[1] where each node is equipped with two Intel Xeon E5-2630v3 processors (with 8 cores each), 128 GB of RAM and a 10 GbE adapter
- Our experimental setup included up to 64 machines

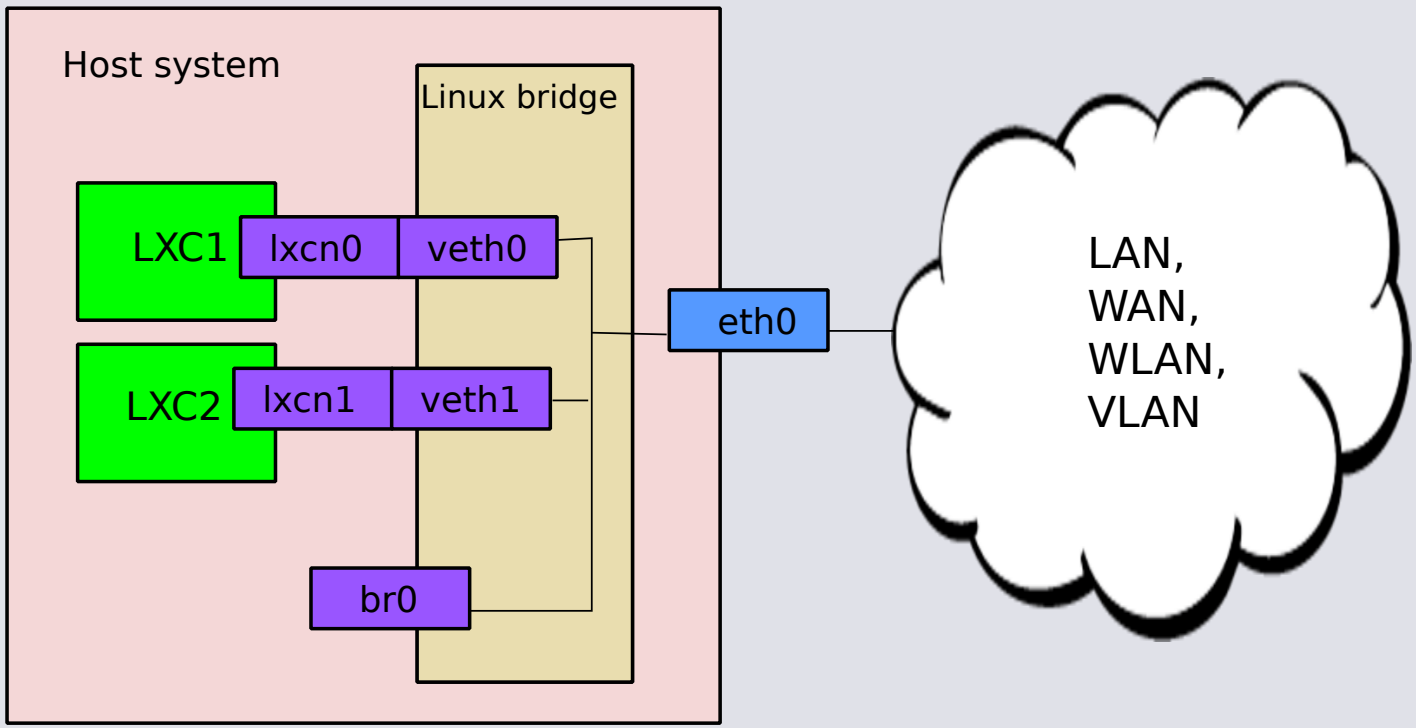
Software

- Debian Jessie, Linux kernel versions: 3.2, 3.16 and 4.0, OpenMPI and NPB. We instrumented the benchmarks: LU, EP, CG, MG, FT, IS using TAU
- We automate the experimentation processes using Distem[a] and Kameleon[b]

[a] https://distem.gforge.inria.fr
[b] https://github.com/camilo1729/distem-recipes

Network setup

- Veth pair + Linux bridge
- Veth pair + OpenvSwitch
- MACVLAN or SR-IOV
- Phys



Linux kernel version

32 containers running on: 8,16,32 physical machines.

Results

- 2/node
  - 3.2: 1577.78%
  - 3.16: 22.67%
  - 4.0: 2.40%
- Overhead present in MPI communication
- Since Linux kernel version 3.11, TSO was enabled in veth

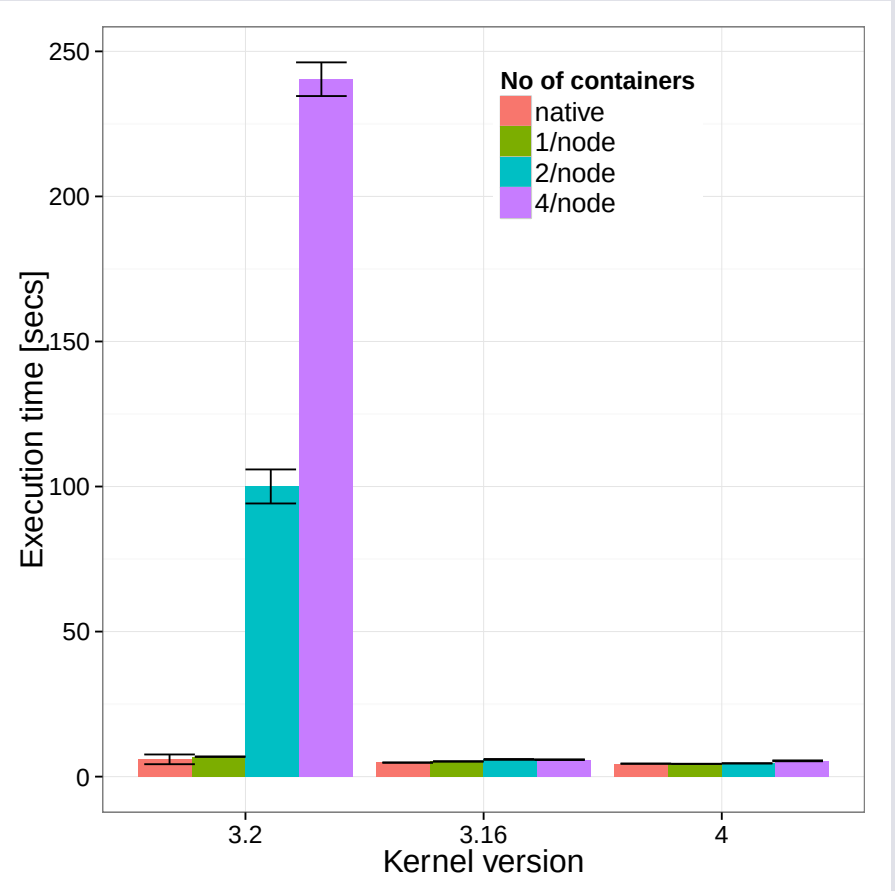


Figure: CG.B

Oversubscription Linux kernel 4.0

- There is a *veth* per MPI process
- 64 containers running over: 8,16,32,64 physical machines

Results

- Top 3 worst performance results: MG, FT, LU
- Maximum overhead (15%, 67%)
- Container placing plays an important role.

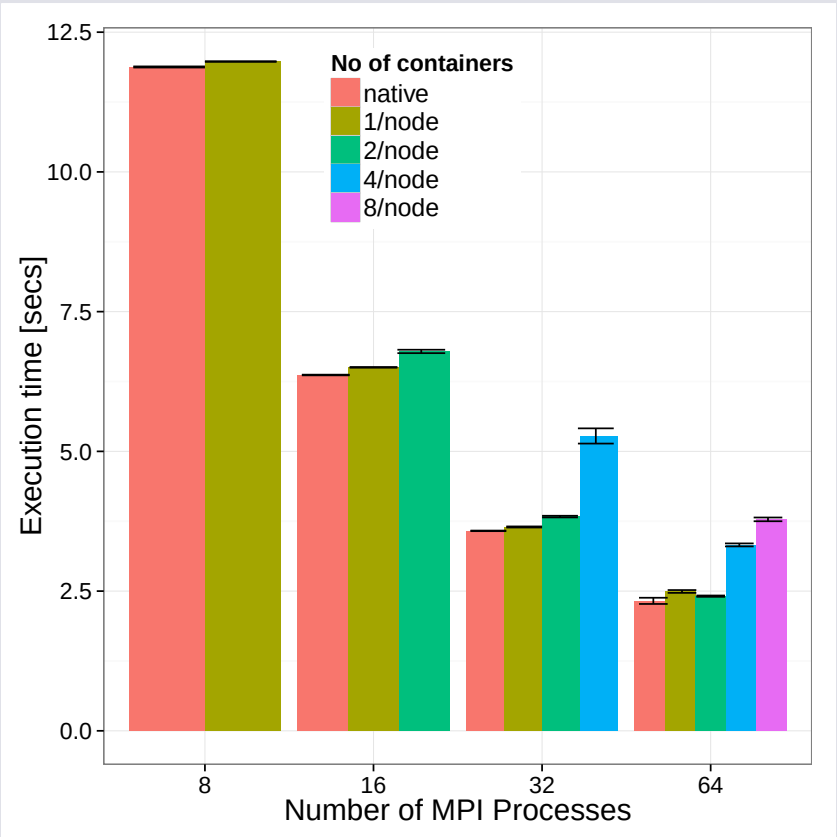
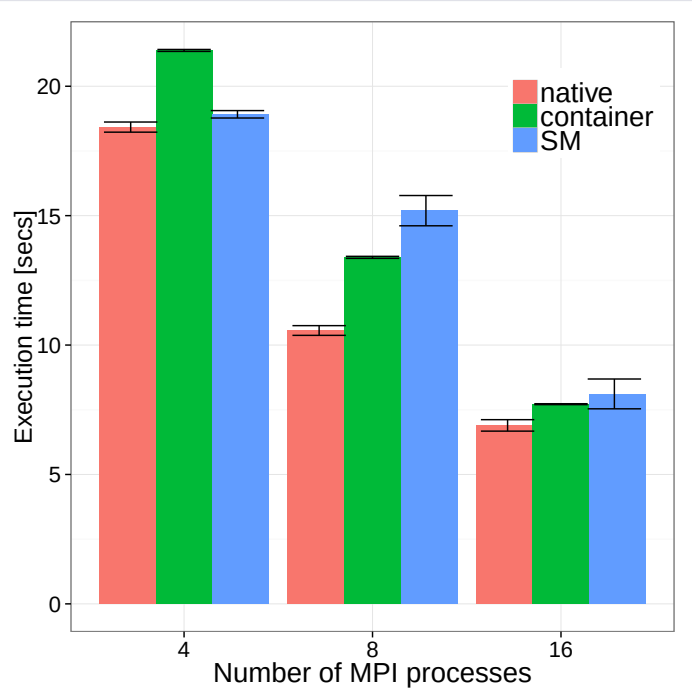


Figure: FT.B

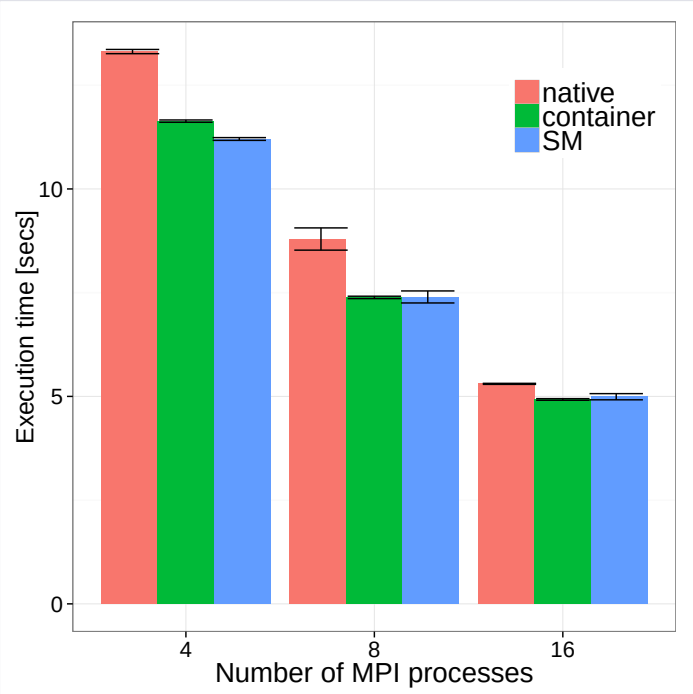
Inter-container communication

- *container* and *SM*: 1 physical node
- *native* : 2, 4, 8 physical nodes

All running the equivalent number of MPI processes.



(a) MG Class B



(b) IS Class C

Inter-container communication

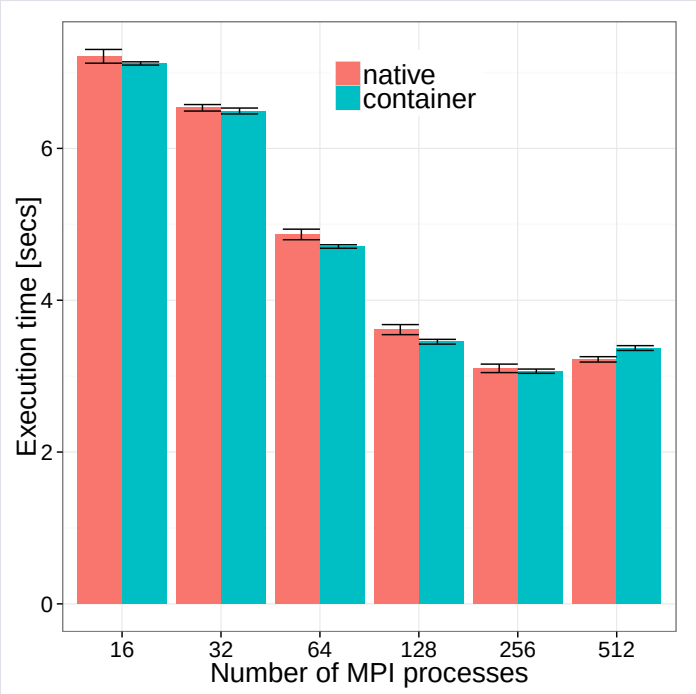
		LU.B		MG.C		EP.B		CG.B	
Native		%	time	%	time	%	time	%	time
	cpu	78	11221	70	4823	79	4342	47	3286
	comm	15	2107	15	1024	3	142	39	2721
	init	7	1050	15	1045	19	1044	15	1045
Container		%	time	%	time	%	time	%	time
	cpu	83	14621	84	6452	80	4682	71	4832
	comm	11	2015	3	206	2	141	14	935
	init	6	1056	14	1057	18	1051	15	1053
SM		%	time	%	time	%	time	%	time
	cpu	81	14989	80	6456	78	4595	70	4715
	comm	13	2350	7	602	4	258	14	938
	init	6	1040	13	1038	18	1038	16	1040

Table: Profile results. Time in *msec*

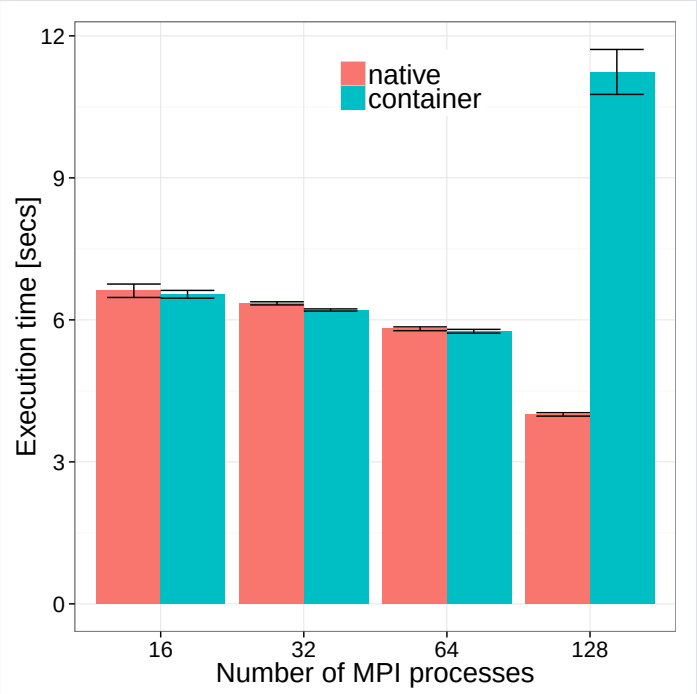
- Inter-container communication is the fastest
- Important degradation of the CPU performance for memory bound applications
- LU: 53%, MG: 53%, EP: 25%, CG: 12%, FT: 0%, IS: 0% (overheads regarding native)

Multinode inter-container communication

- 16 MPI processes were run per physical machine or container
- We used a maximum of 32 physical machines



(a) FT Class B



(b) CG Class B

Multinode inter-container communication

- Benchmarks with low MPI communication: we observed a maximum overhead of 5.97% (with 512 MPI processes)
- Benchmarks with an intensive MPI communication: we observed a higher overhead starting from 30% for the benchmark LU

Multinode inter-container communication

- A particular behavior is observed for CG benchmark. It reaches 180% of overhead when 128 MPI processes are used. The number of MPI messages sent by this benchmark increases with the number of nodes, leading to network congestion and TCP timeouts
- We found a way to alleviate the overhead by tweaking parameters of the Linux network stack
  - TCP minimum retransmission timeout (RTO)
  - TCP Selective Acknowledgments (SACK)

Outline

In the context of HPC ...

- We study the impact of using containers.
- We evaluate two interesting uses of containers:
  - portability of complex software stacks
  - oversubscription

What did we find?

- There is important performance degradation provoked by veth for Linux kernels < 3.11
- Container placing plays in important role under oversubscription
- Memory bound applications and application that use all to all MPI communication are the most affected by oversubscription
- Inter-container communication through veth has equivalent performance than communication through shared memory using OpenMPI
- Performance issues can appear only at certain scale (e.g. 180 % overhead with 128 nodes for CG benchmark)

Future work

- Measure the impact of using containers on disk I/O and other containers features like memory limitation
- The overhead observed could be diminished by integrating more advance network interconnection such as Linux's *macvlan*, SR-IOV or OpenvSwitch[1]

[1] http://openvswitch.org/

The end

Thank you

Outline

Bibliography

[1] Daniel Balouek et al. Adding virtualization capabilities to the Grid'5000 testbed. In *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer, 2013.

[2] W. Felter et al. An updated performance comparison of virtual machines and linux containers. Technical report, IBM, 2015.

[3] Jeanna Neefe Matthews et al. Quantifying the performance isolation properties of virtualization systems. In *Experimental Computer Science*, page 6, 2007.

[4] J.P. Walter, V. Chaudhary, Minsuk Cha, S. Guercio, and S. Gallo. A comparison of virtualization technologies for hpc. In *AINA 2008*, pages 861–868, March 2008.

[5] M.G. Xavier et al. Performance evaluation of container-based virtualization for high performance computing environments. In *PDP 2013*, pages 233–240, Belfast, UK, Feb 2013.