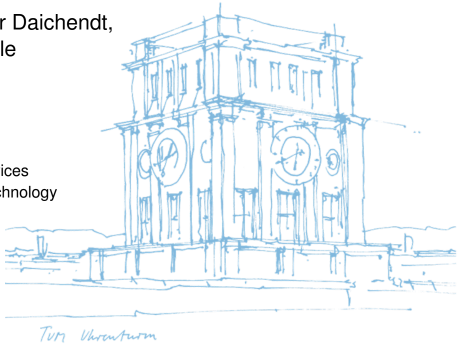
Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Containing Low Tail-Latencies in Packet Processing Using Lightweight Virtualization

Florian Wiedner, Max Helm, Alexander Daichendt,
Jonas Andre, and Georg Carle

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich



Motivation

Introduction

- Low-latency is increasingly vital in general-purpose networks

Motivation

Introduction

- Low-latency is increasingly vital in general-purpose networks
- Demand for resource sharing and on-demand provisioning is rising

Motivation

Introduction

- Low-latency is increasingly vital in general-purpose networks
- Demand for resource sharing and on-demand provisioning is rising
- One hardware machine per customer and application is expensive

Motivation

Introduction

- Low-latency is increasingly vital in general-purpose networks
- Demand for resource sharing and on-demand provisioning is rising
- One hardware machine per customer and application is expensive
- Virtualization allows to share of systems and resources between customers/applications

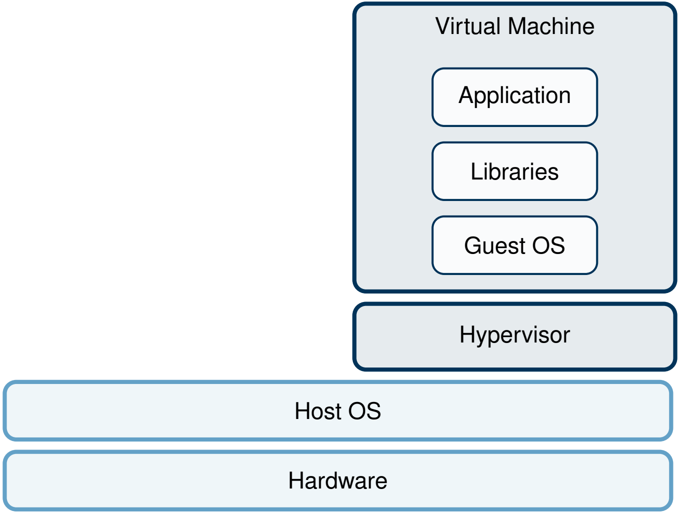
Motivation

Introduction

- Low-latency is increasingly vital in general-purpose networks
- Demand for resource sharing and on-demand provisioning is rising
- One hardware machine per customer and application is expensive
- Virtualization allows to share of systems and resources between customers/applications
- Can virtualized network applications achieve stable low latency in networking?

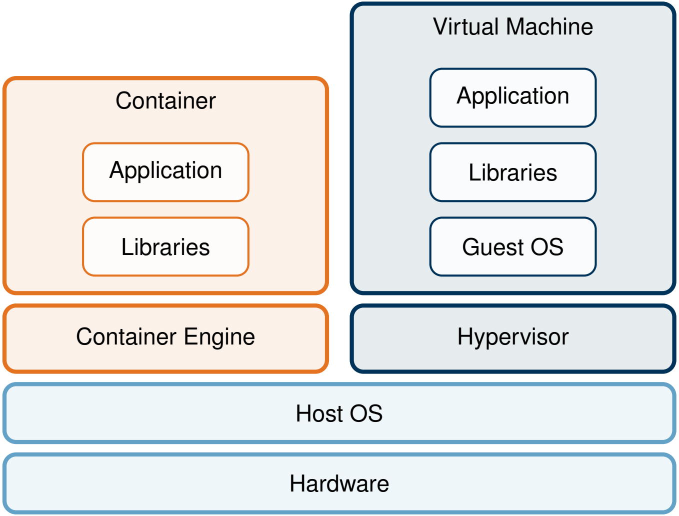
Virtualization: Virtual Machines vs. Container

Overview



Virtualization: Virtual Machines vs. Container

Overview



Research Goal and State-of-the-Art

Goal

Can container enhance resource-sharing when processing low-latency traffic?

- What is the state-of-the-art on using low-latency applications on virtualized systems?
- Are containers capable of processing traffic while holding low-latency requirements?
- Is tail-latency behavior differing between container, VMs, bare-metal, and kernel to user-mode processing?

Research Goal and State-of-the-Art

State-of-the-Art

	VMs		Container		Baremetal		Tail-latency
	Latency	Throughput	Latency	Throughput	Latency	Throughput	
Tran and Kim[1]	✗	✗	✗	✓	✗	✗	✗
Cha and Kim[2]	✗	✗	✗	✓	✗	✗	✗
Liu[3]	✓	✓	✗	✗	✗	✗	✗
Gallenmüller et al.[4]	✓	✓	✗	✗	✓	✓	✓

1 M.-N. Tran and Y. Kim, „Network Performance Benchmarking for Containerized Infrastructure in NFV environment“, in 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Jun. 2022, pp. 115–120.

2 J.-G. Cha and S. W. Kim, „Design and Evaluation of Container-based Networking for Low-latency Edge Services“, in 2021 International Conference on Information and Communication Technology Convergence (ICTC), Oct. 2021, pp. 1287–1289.

3 J. Liu, „Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support“, in 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), Apr. 2010, pp. 1–12.

4 S. Gallenmüller, F. Wiedner, J. Naab, et al., „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, et al., Eds., IEEE, 2021, pp. 537–543.

Research Goal and State-of-the-Art

State-of-the-Art

	VMs		Container		Baremetal		Tail-latency
	Latency	Throughput	Latency	Throughput	Latency	Throughput	
Tran and Kim[1]	✗	✗	✗	✓	✗	✗	✗
Cha and Kim[2]	✗	✗	✗	✓	✗	✗	✗
Liu[3]	✓	✓	✗	✗	✗	✗	✗
Gallenmüller et al.[4]	✓	✓	✗	✗	✓	✓	✓
This paper	✓	✗	✓	✓	✓	✗	✓

1 M.-N. Tran and Y. Kim, „Network Performance Benchmarking for Containerized Infrastructure in NFV environment“, in 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Jun. 2022, pp. 115–120.

2 J.-G. Cha and S. W. Kim, „Design and Evaluation of Container-based Networking for Low-latency Edge Services“, in 2021 International Conference on Information and Communication Technology Convergence (ICTC), Oct. 2021, pp. 1287–1289.

3 J. Liu, „Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support“, in 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), Apr. 2010, pp. 1–12.

4 S. Gallenmüller, F. Wiedner, J. Naab, et al., „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, et al., Eds., IEEE, 2021, pp. 537–543.

Low-Latency Optimizations

Low-latency software stack

How can we reduce latency spikes in virtualized systems?

mostly caused by interrupts

Low-latency software stack

How can we reduce latency spikes in virtualized systems?

mostly caused by interrupts

Optimizations for VMs[a]

- Polling-based IO (DPDK)
- Disable Simultaneous Multithreading
- Disable energy-saving mechanism
- Statically allocate CPU cores for processes
- Interrupt affinity to core 0
- Isolate VM and packet processing from OS kernel

[a] S. Gallenmüller, F. Wiedner, J. Naab, et al., „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, et al., Eds., IEEE, 2021, pp. 537–543.

Low-latency software stack

How can we reduce latency spikes in virtualized systems?

mostly caused by interrupts

Optimizations for VMs[a]

- Polling-based IO (DPDK)
- Disable Simultaneous Multithreading
- Disable energy-saving mechanism
- Statically allocate CPU cores for processes
- Interrupt affinity to core 0
- Isolate VM and packet processing from OS kernel

Optimizations for Container

- Polling-based IO (DPDK)
- Disable Simultaneous Multithreading
- Disable energy-saving mechanism
- Statically allocate CPU cores for processes
- Interrupt affinity to core 0

[a] S. Gallenmüller, F. Wiedner, J. Naab, et al., „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, et al., Eds., IEEE, 2021, pp. 537–543.

How can we reduce latency spikes in virtualized systems?

mostly caused by interrupts

Optimizations for VMs[a]

- Polling-based IO (DPDK)
- Disable Simultaneous Multithreading
- Disable energy-saving mechanism
- Statically allocate CPU cores for processes
- Interrupt affinity to core 0
- Isolate VM and packet processing from OS kernel

Optimizations for Container

- Polling-based IO (DPDK)
- Disable Simultaneous Multithreading
- Disable energy-saving mechanism
- Statically allocate CPU cores for processes
- Interrupt affinity to core 0
- Use control groups to limit cores and memory

[a] S. Gallenmüller, F. Wiedner, J. Naab, et al., „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, et al., Eds., IEEE, 2021, pp. 537–543.

Is the Kernel configuration influencing the latency?

Is the Kernel configuration influencing the latency?

Vanilla kernel

- Debian Bullseye
- No changes, as provided by the Debian project

Real-Time (RT) kernel

- Debian Bullseye
- Real-time patches integrated
- Targeted to achieve deterministic behavior

NoHz kernel

- Debian Bullseye
- Real patches integrated
- NoHz kernel option enabled
- Allow to isolate cores with only one application running from timer interrupts

Low-Latency Optimizations

Goal

Can container enhance resource-sharing when processing low-latency traffic?

- ✓ What is the state-of-the-art on using low-latency applications on virtualized systems?
- × Are containers capable of processing traffic while holding low-latency requirements?
- × Is tail-latency behavior differing between container, VMs, bare-metal, and kernel to user-mode processing?

Experimental Setup

Workload and Scenario

- Use container (Linux Container (LXC)) as virtual network function
- Ingress and Egress hardware interface direct-attached to the container
- DPDK-based Libmoon l2-forwarding application
- One container and simple application to reduce outside influences

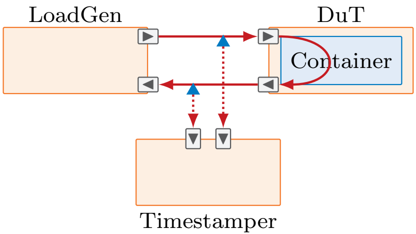
Experimental Setup

Workload and Scenario

- Use container (Linux Container (LXC)) as virtual network function
- Ingress and Egress hardware interface direct-attached to the container
- DPDK-based Libmoon l2-forwarding application
- One container and simple application to reduce outside influences
- Traffic: UDP Traffic with 64 B-sized packets
- Duration: 160 s per measurement
- Rate: 1 Mpackets/s

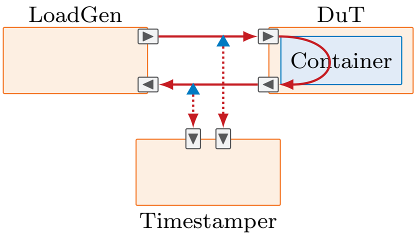
But how to measure the latency precisely?

But how to measure the latency precisely?



- Loadgen runs a packet generator (MoonGen) creating UDP packets
- Device under Test (DuT) runs Container/VMs/Packet Processing application
- Timestamper records DuT ingress/egress traffic (passive optical TAPs)

But how to measure the latency precisely?



- Loadgen runs a packet generator (MoonGen) creating UDP packets
- Device under Test (DuT) runs Container/VMs/Packet Processing application
- Timestamper records DuT ingress/egress traffic (passive optical TAPs)

DuT

- AMD EPYC 7551P 32-Core
- 2x X710 10GbE SFP+ NICs
- 128 GiB RAM

LoadGen

- Intel Xeon Silver 4116
- Intel 82599ES 10GbE SFP+ NIC
- 192 GiB RAM

Timestamper

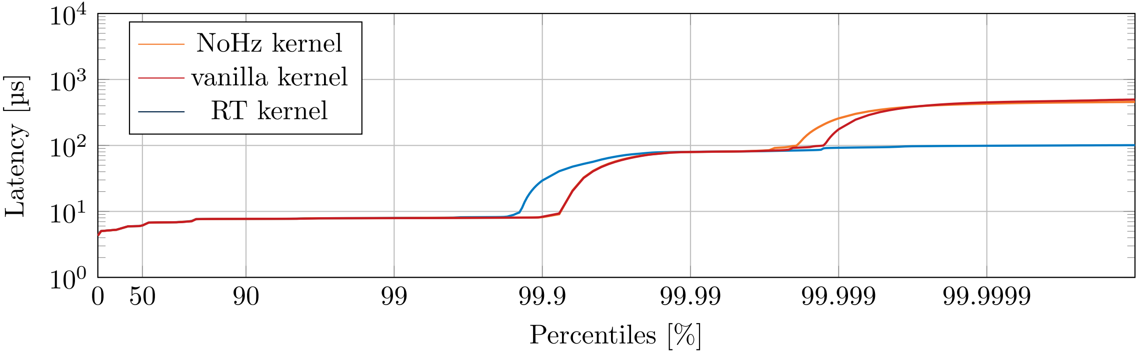
- AMD EPYC 7542 32-Core
- Intel E810-XXVDA4 NIC
- 500 GiB RAM

Kernel Variants (optimized) [1 Mpackets/s]

Which of the available kernels provides most deterministic latency behavior on container?

Kernel Variants (optimized) [1 Mpackets/s]

Which of the available kernels provides most deterministic latency behavior on container?



Evaluation

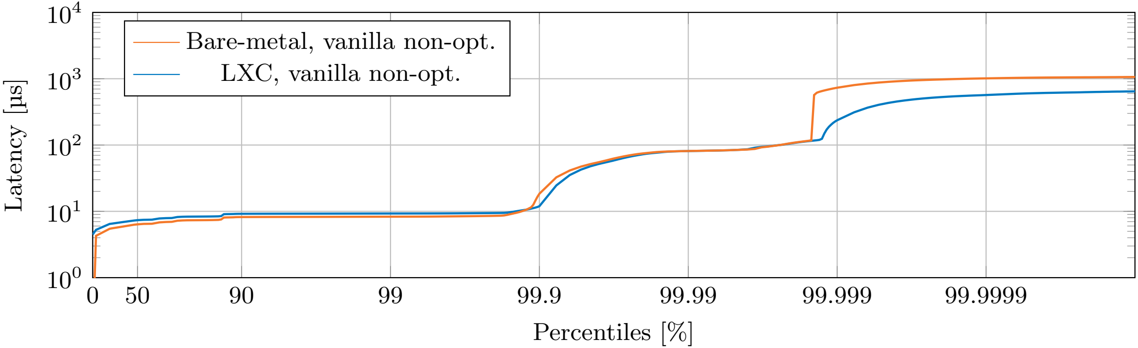
Goal

Can container enhance resource-sharing when processing low-latency traffic?

- ✓ What is the state-of-the-art on using low-latency applications on virtualized systems?
- ✓ Are containers capable of processing traffic while holding low-latency requirements?
- × Is tail-latency behavior differing between container, VMs, bare-metal, and kernel to user-mode processing?

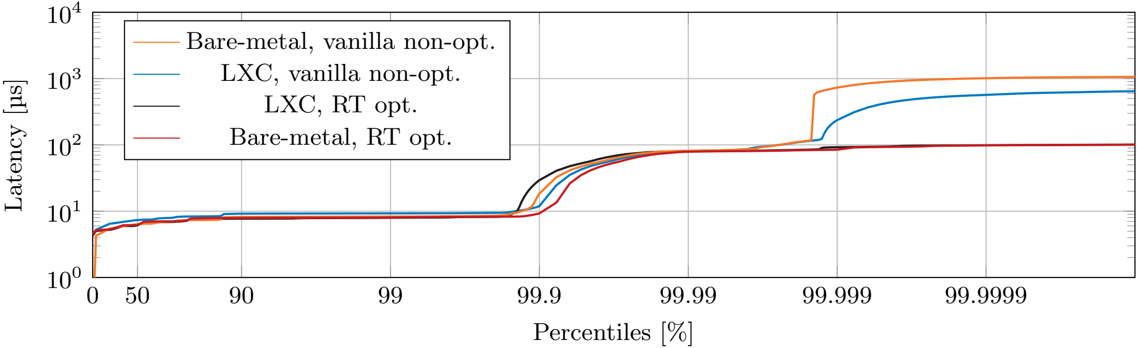
Is packet processing on containers a disadvantage compared to bare-metal?

Is packet processing on containers a disadvantage compared to bare-metal?



Container vs. Bare-metal [1 Mpackets/s]

Is packet processing on containers a disadvantage compared to bare-metal?

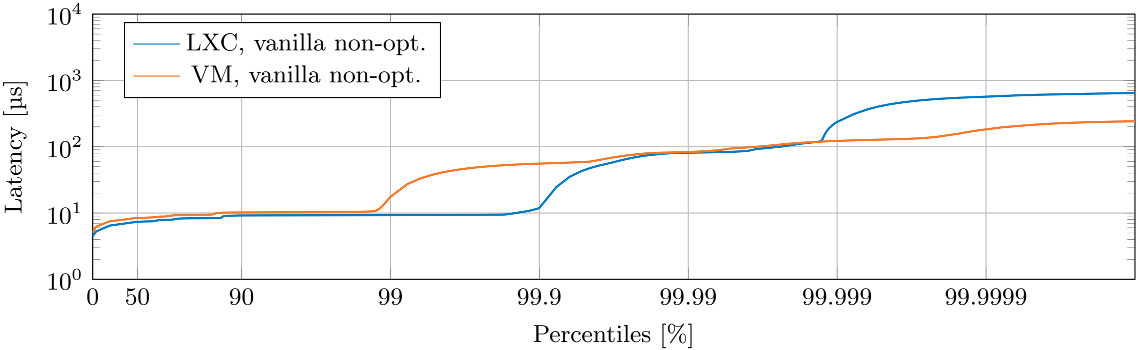


Evaluation

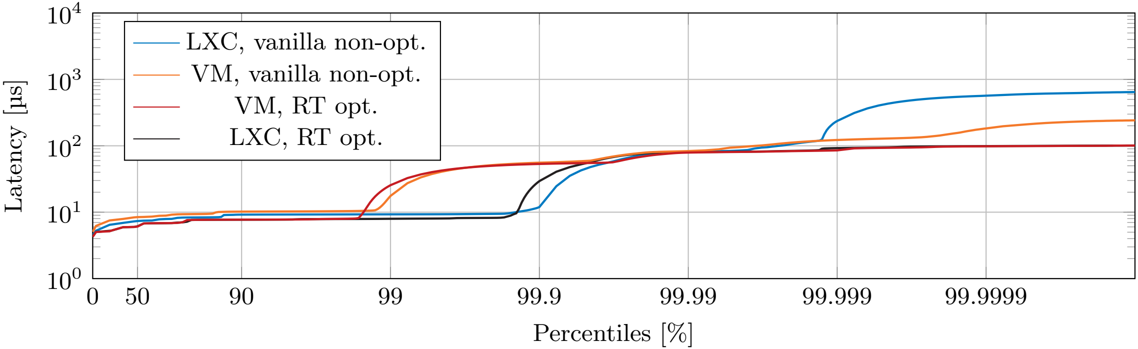
Container vs. VMs [1 Mpackets/s]

Is packet processing on containers a disadvantage compared to VMs?

Is packet processing on containers a disadvantage compared to VMs?



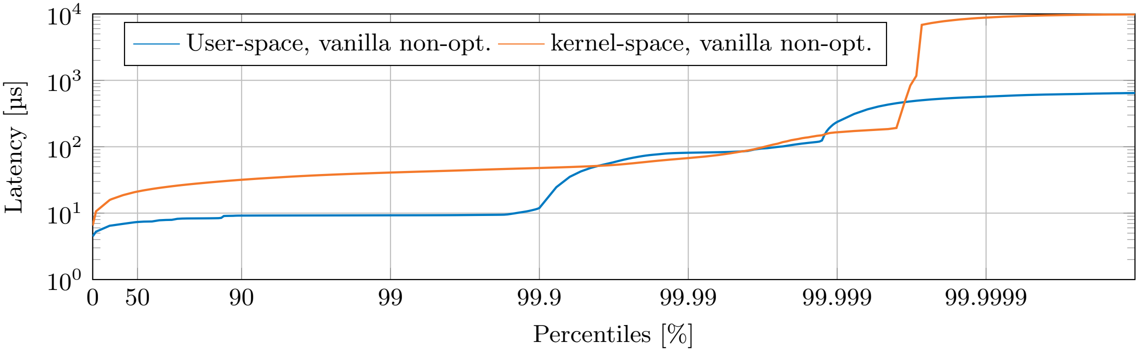
Is packet processing on containers a disadvantage compared to VMs?



Container using DPDK vs Container Linux Kernel [1 Mpackets/s]

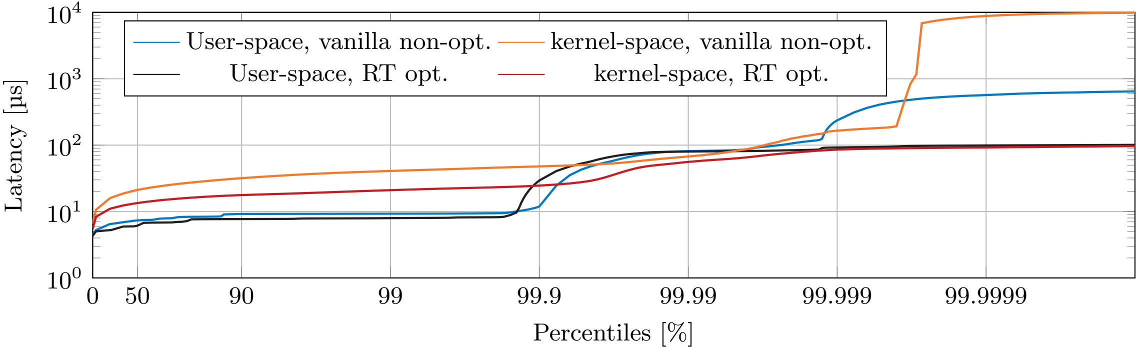
Is kernel-mode networking possible for low-latency on container?

Is kernel-mode networking possible for low-latency on container?



Container using DPDK vs Container Linux Kernel [1 Mpackets/s]

Is kernel-mode networking possible for low-latency on container?



Evaluation

Goal

Can container enhance resource-sharing when processing low-latency traffic?

- ✓ What is the state-of-the-art on using low-latency applications on virtualized systems?
- ✓ Are containers capable of processing traffic while holding low-latency requirements?
- ✓ Is tail-latency behavior differing between container, VMs, bare-metal, and kernel to user-mode processing?

Conclusion

Low Tail-Latencies in Packet Processing Systems with Lightweight Virtualization

- Using low-latency packet processing in container is possible
- Similar tail-latencies between container, bare-metal, and VMs
- More influence of shared OS in lightweight systems
- Extending state-of-the-art by adding baseline latency measurements

Conclusion

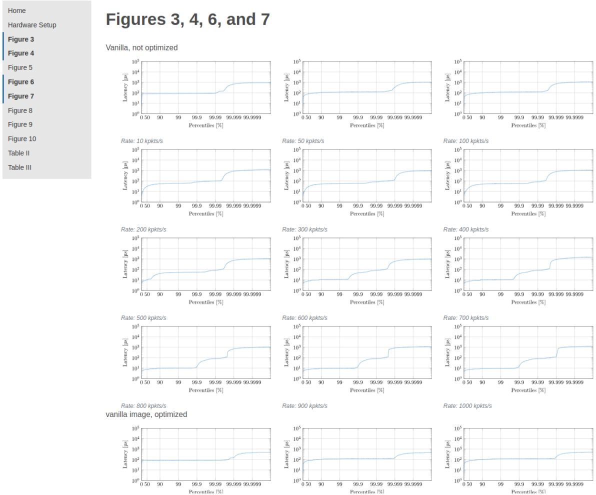
Low Tail-Latencies in Packet Processing Systems with Lightweight Virtualization

- Using low-latency packet processing in container is possible
- Similar tail-latencies between container, bare-metal, and VMs
- More influence of shared OS in lightweight systems
- Extending state-of-the-art by adding baseline latency measurements

Paper

Measurement Data

- Available artifacts:
  - Evaluation scripts
  - Measurement data
- Website for reproducibility: https://wiednerf.github.io/containerized-low-latency/



Bibliography

[1] M.-N. Tran and Y. Kim, „Network Performance Benchmarking for Containerized Infrastructure in NFV environment“, in 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Jun. 2022, pp. 115–120.

[2] J.-G. Cha and S. W. Kim, „Design and Evaluation of Container-based Networking for Low-latency Edge Services“, in 2021 International Conference on Information and Communication Technology Convergence (ICTC), Oct. 2021, pp. 1287–1289.

[3] J. Liu, „Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support“, in 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), Apr. 2010, pp. 1–12.

[4] S. Gallenmüller, F. Wiedner, J. Naab, and G. Carle, „Ducked Tails: Trimming the Tail Latency of(f) Packet Processing Systems“, in 17th International Conference on Network and Service Management, CNSM 2021, Izmir, Turkey, October 25-29, 2021, P. Chemouil, M. Ulema, S. Clayman, M. Sayit, C. Çetinkaya, and S. Secci, Eds., IEEE, 2021, pp. 537–543.

[5] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, „An updated performance comparison of virtual machines and linux containers“, in 2015 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2015, Philadelphia, PA, USA, March 29-31, 2015, IEEE Computer Society, 2015, pp. 171–172. DOI: 10.1109/ISPASS.2015.7095802. [Online]. Available: https://doi.org/10.1109/ISPASS.2015.7095802.

Bibliography

[6] L. Abeni, C. Király, N. Li, and A. Bianco, „Tuning KVM to enhance virtual routing performance“, in Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013, IEEE, 2013, pp. 3803–3808. DOI: 10.1109/ICC.2013.6655148. [Online]. Available: https://doi.org/10.1109/ICC.2013.6655148.

[7] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, „The pos framework: A methodology and toolchain for reproducible network experiments“, in CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021, G. Carle and J. Ott, Eds., ACM, 2021, pp. 259–266.

Backup

Used Frameworks

Linux Container (LXC) version 4 [5]

Kernel Virtual Machines (KVM) [6]

Backup

Plain Orchestrating Service [7]

- pos is . . .
  - a testbed orchestration service, and
  - an experiment methodology.
- Methodology makes experiments . . .
  - repeatable as everything is automated,
  - reproducible as all scripts needed to start the software can be published, and
  - easier to replicate as the experiment scripts document experiments.

Backup

Plain Orchestrating Service [7]

- pos is . . .
  - a testbed orchestration service, and
  - an experiment methodology.
- Methodology makes experiments . . .
  - repeatable as everything is automated,
  - reproducible as all scripts needed to start the software can be published, and
  - easier to replicate as the experiment scripts document experiments.

- Control hardware machines and VMs using . . . :
  - IPMI as management protocol
  - virtualBMC for controlling VMs similar to hardware machines
  - DHCP for distributing IPs

Backup

VirtualLXCBMC

Challenge: No IPMI available for LXC container

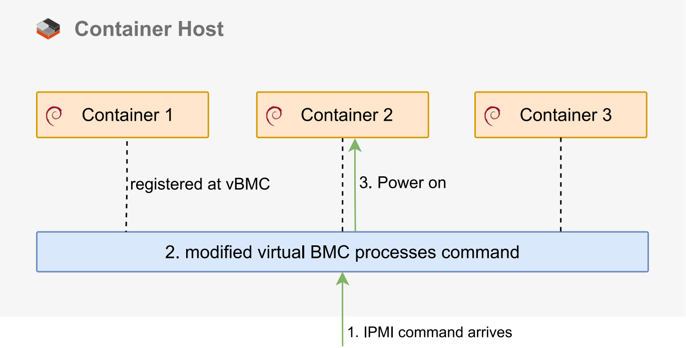
Backup

VirtualLXCBMC

Challenge: No IPMI available for LXC container

Solution: Fork VirtualBMC for LXC:

- Support starting and stopping container
- Status information of container



Backup

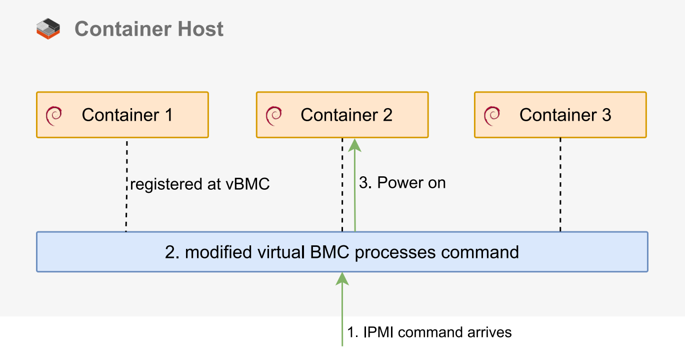
VirtualLXCBMC

Challenge: No IPMI available for LXC container

Solution: Fork VirtualBMC for LXC:

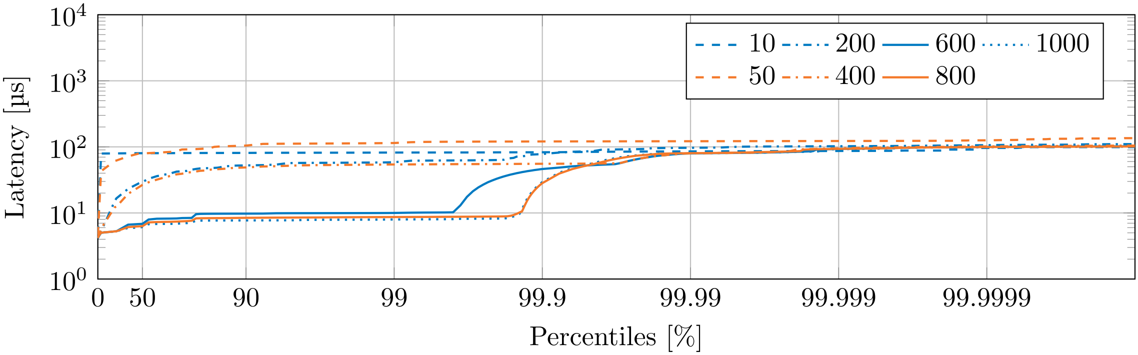
- Support starting and stopping container
- Status information of container

Published in
`https://github.com/tumi8/VirtualLXCBMC`



Backup

Rates on optimized system [kpackets/s]



Rates influence mostly the mean latency.

- The higher the rates, the lower the mean latency
- Tail-latency not significantly influenced