Dynamic Measurement of Container Network Latency with MACE

Chris Misa and Ramakrishnan Durairajan, University of Oregon
cmisa@cs.uoregon.edu ram@cs.uoregon.edu

Motivation and Challenges

Benefits of container technology for Internet research:

- Consistent and repeatable experimental interface;
- Streamlined tool deployment process;
- New vantage points at cloud-oriented data centers around the world.

Challenges for Internet measurement tools deployed in containers:

- One-way delays (OWDs) and round-trip-times (RTTs) are distorted by virtualized network stack;[1]
- No principled method to measure, quantify, and characterize the latency overheads.

Design

We develop MACE (Measure the Added Container Expense) to dynamically monitor latency overheads:

- MACE parses a stream of Linux trace events[2] to calculate egress and ingress latencies for each packet;
- MACE uses a configurable event path to determine target trace event / network device points for outbound and inbound packets;
- Different event paths measure different sources of latency in the kernel;
- We envision MACE providing key OS and network behavior observations for the dynamic optimization of virtualized networks.

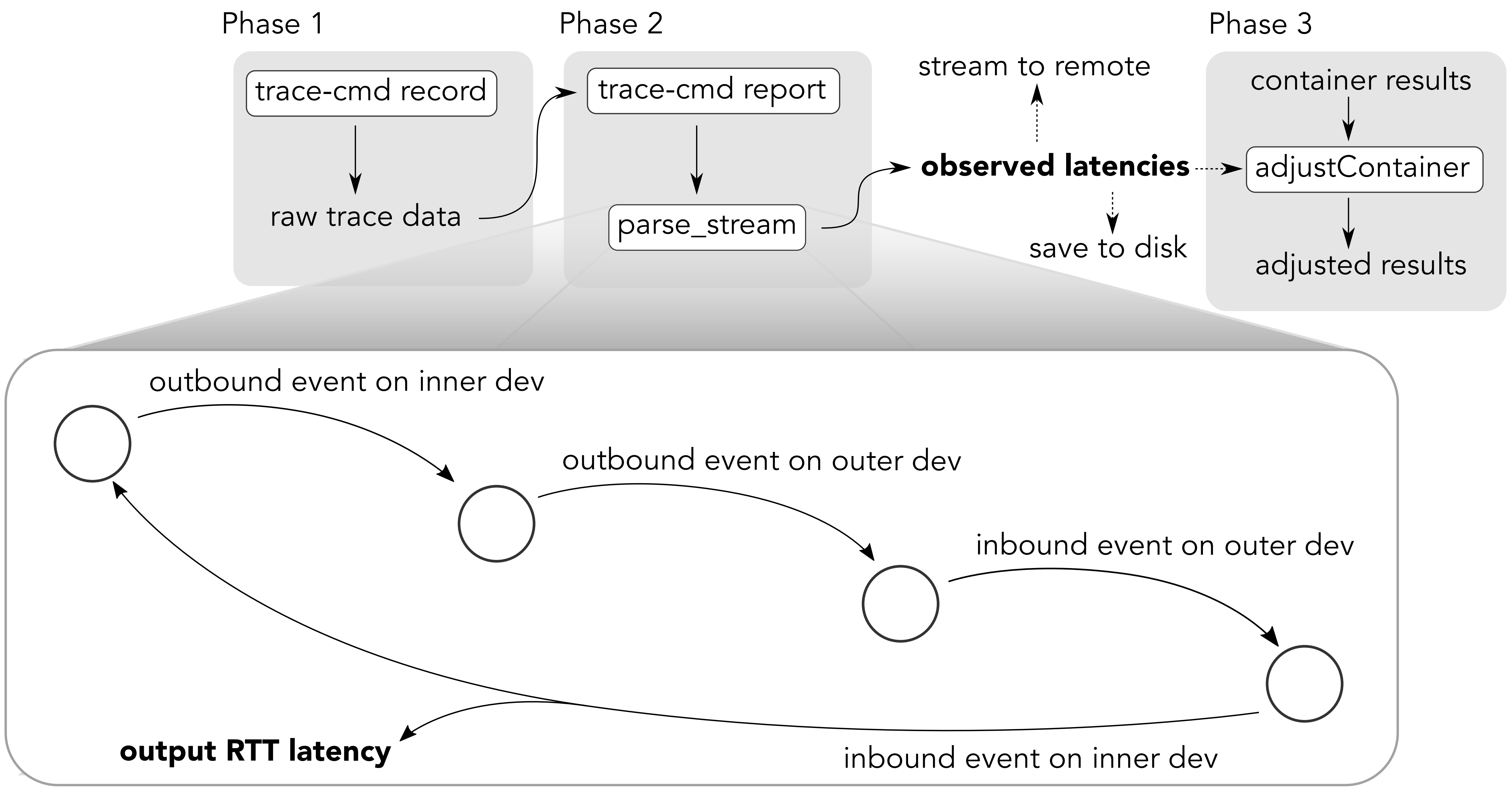

Fig. 1: MACE design overview

Path Name	Outbound				Inbound			
	Inner		Outer		Inner		Outer	
	Device	Event	Device	Event	Device	Event	Device	Event
inner-dev	veth	net_dev_xmit	physical	net_dev_xmit	physical	netif_receive_skb	veth	netif_receive_skb
max-dev	veth	net_dev_queue	physical	net_dev_xmit	physical	napi_gro_frags_entry	veth	netif_receive_skb
syscalls	NA	sys_enter_sendto	physical	net_dev_xmit	physical	netif_receive_skb	NA	sys_exit_recvmsg

Fig. 2: Table of event paths chosen for evaluation

Evaluation

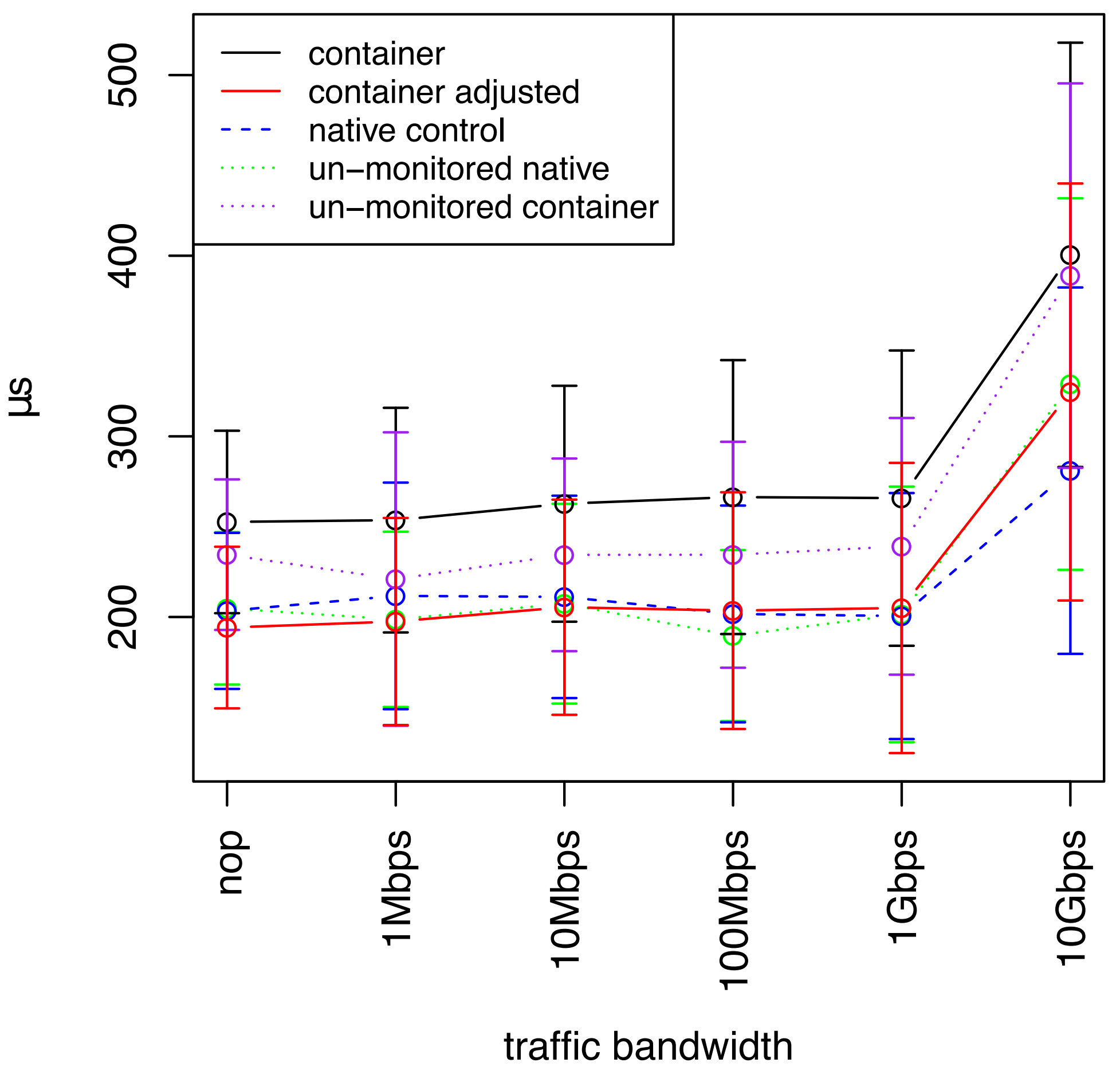

Fig. 3: Mean RTTs under the 'inner-device' event path compared with native ping using gettimeofday()

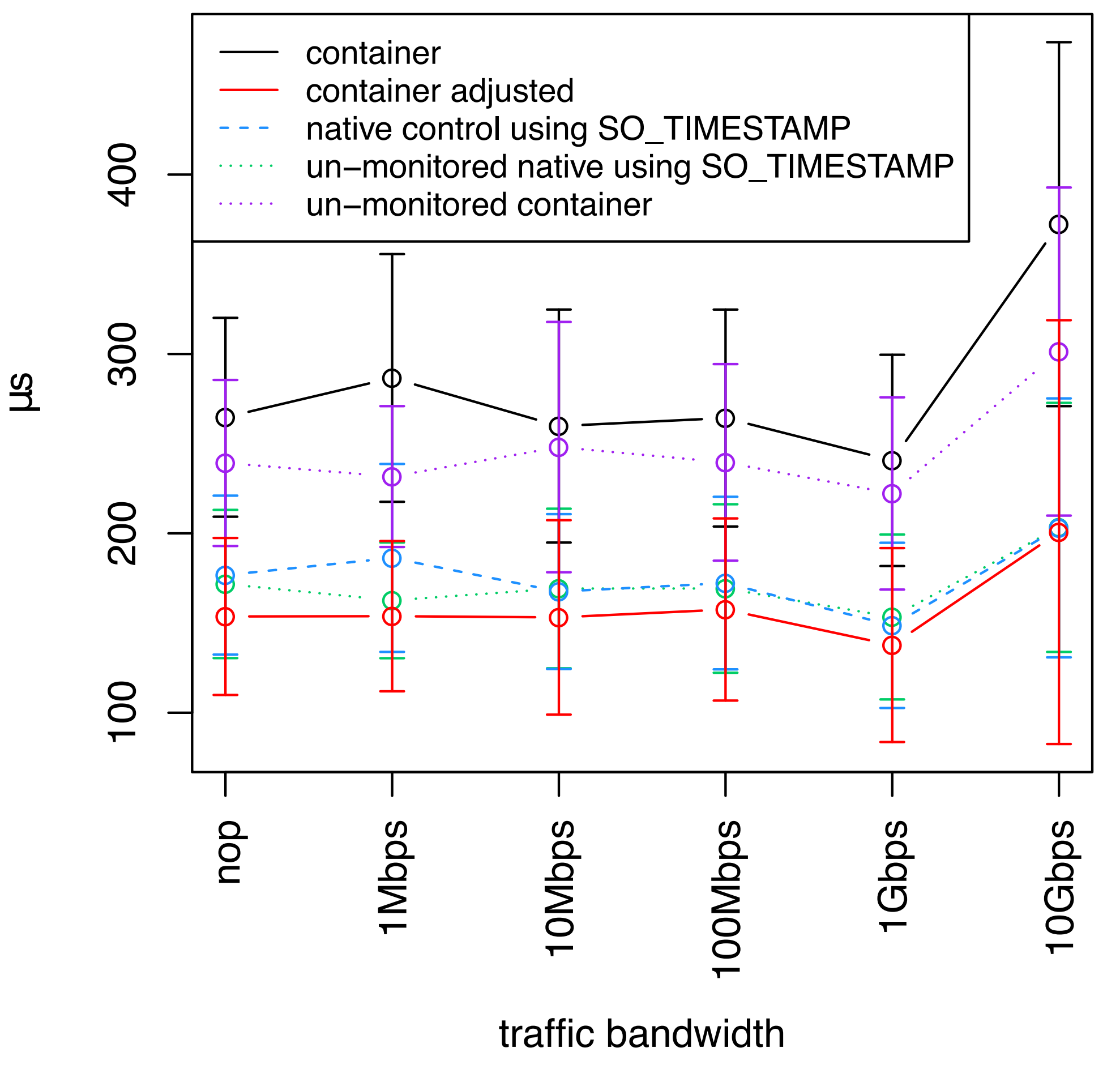

Fig. 4: Mean RTTs under the 'syscalls' event path compared with native ping using SO_TIMESTAMP on inbound packets

Key Recommendations:

- Measurement tools running inside of containers report different latencies compared to identical tools running natively in the host;
- MACE running with the 'inner-dev' method can account for and remove these overheads, yielding results consistent with an identical native installation of the same tool;
- The 'max-dev' method for MACE additionally accounts for queueing delays which may become significant in busy servers;
- Native tool installations also include latency induced by the host's network stack and the 'syscalls' method for MACE includes this latency from the container up to the kernel boundary.

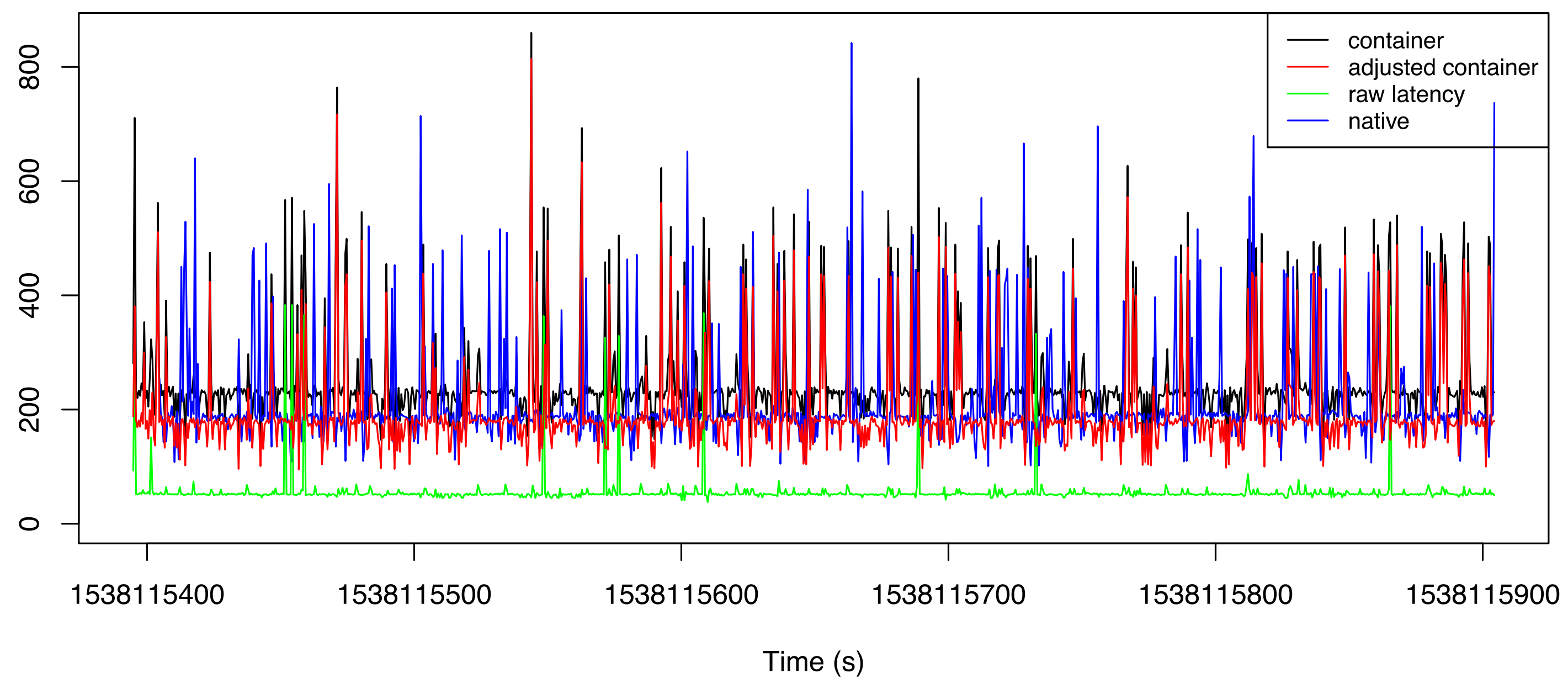

Fig. 6: Time series of 'inner-dev' event path under 10Mbps traffic

Method	Mean	Mode	Sample Deviation
native, original	169.253	156.5	65.338
native, hardware timestamping	118.793	119.6	21.620
native, no `SO_TIMESTAMP`	197.716	185.5	63.431
container, monitored, raw	275.639	231.5	110.418
container, adjusted by `inner-dev'	205.400	177.5	86.545
container, adjusted by `max-dev'	204.042	173.5	91.751
container, adjusted by `syscalls'	161.107	132.5	82.758

Fig. 5: Summary of RTT evaluation results (μs)

1. W. Felter, A. Ferreira, R. Rajamony, and J. Rubio,"An updated performance comparison of virtual machines and linux containers," Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, 2015.
2. S. Rostedt, "ftrace – function tracer," 2008. Available: https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/Documentation/trace/ftrace.txt
3. R. Ricci, E. Eide, and the CloudLab Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," ;login:, vol. 39, no. 6, pp. 36–38, 2014. Available: https://www.usenix.org/publications/login/dec14/ricci

This work is supported in part by a fellowship from the University of Oregon Office of the Vice President for Research and Innovation.