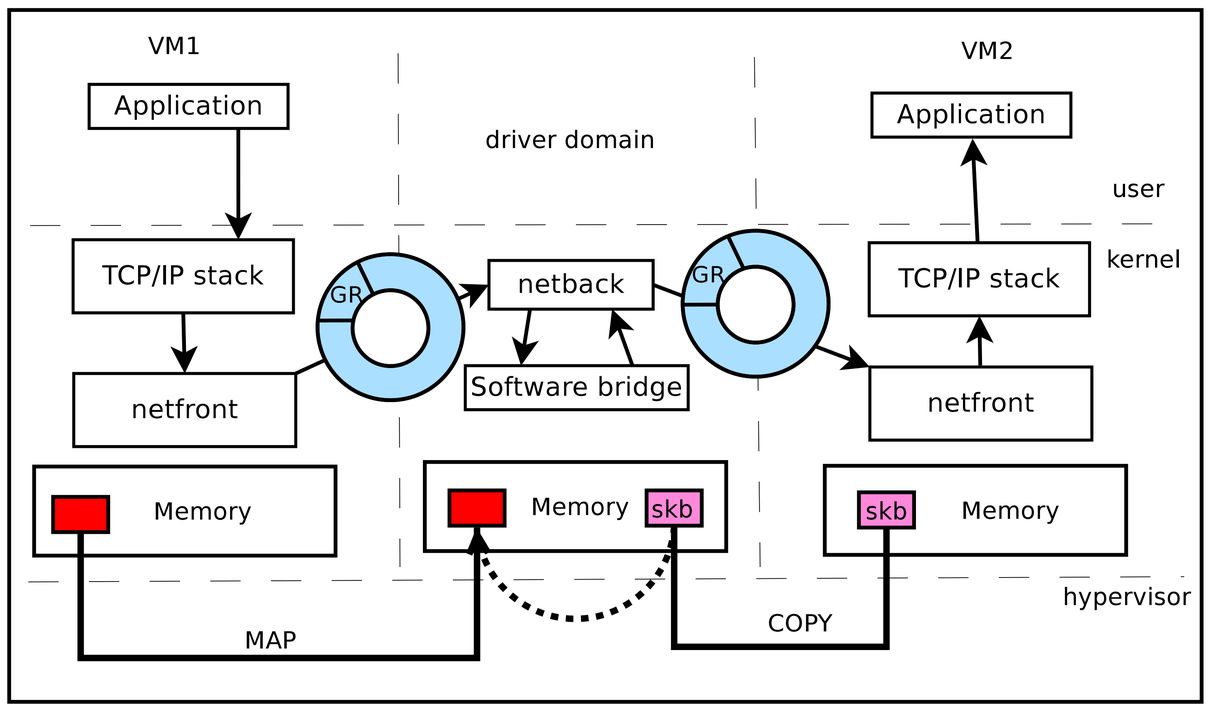
V4VSockets
low overhead intra-node communication in Xen

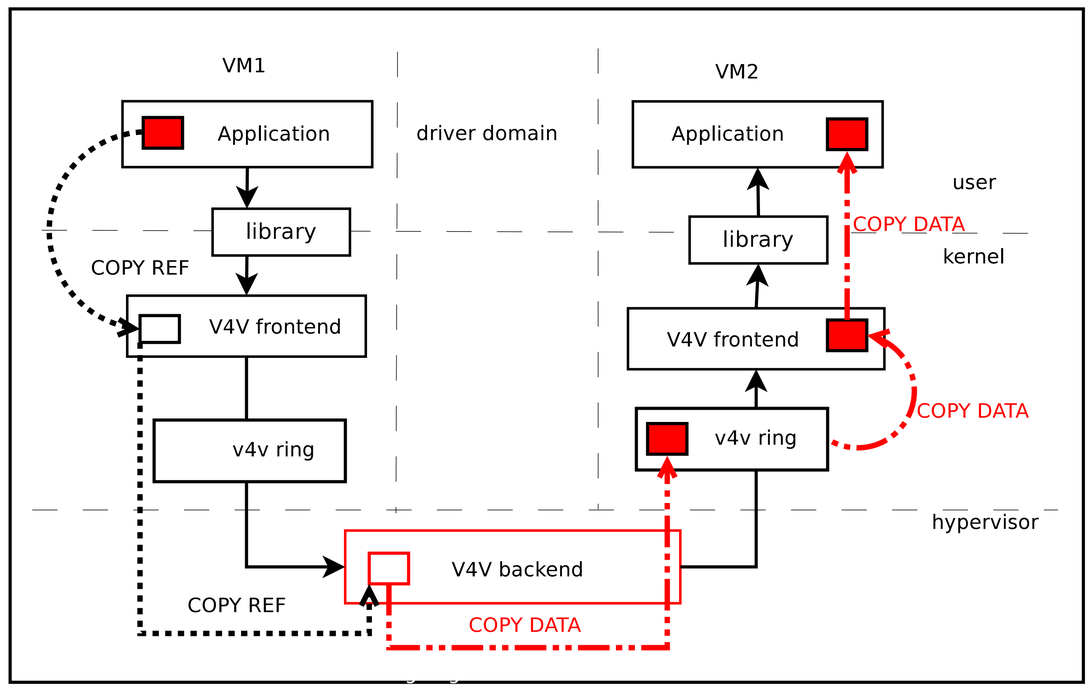
I/O Path in Xen (generic environment)



Intra-node communication suffers from severe overheads:

- ⇒ inefficient data paths
- ⇒ driver domain handles packet forwarding
- ⇒ unnecessary TCP/IP stack crossing and fragmentation

I/O Path in Xen with V4VSockets



V4VSockets is built as a full-stack protocol framework that supports p2p communication between VMs.

- ⇒ *Application layer*: the socket interface.
- ⇒ *Transport layer*: VM kernel driver.
- ⇒ *Network/Link layer*: the hypervisor, providing encapsulation of upper-layer messages to V4V messages, and packet delivery.

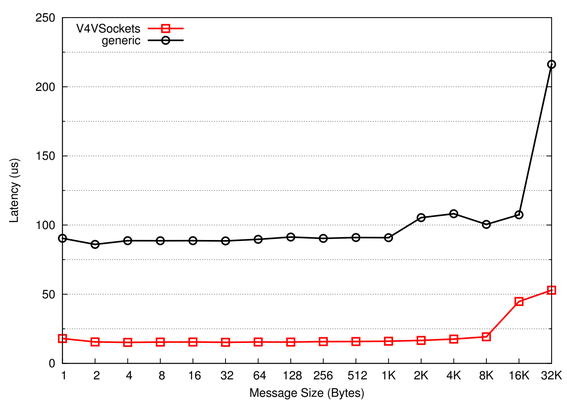
Key features

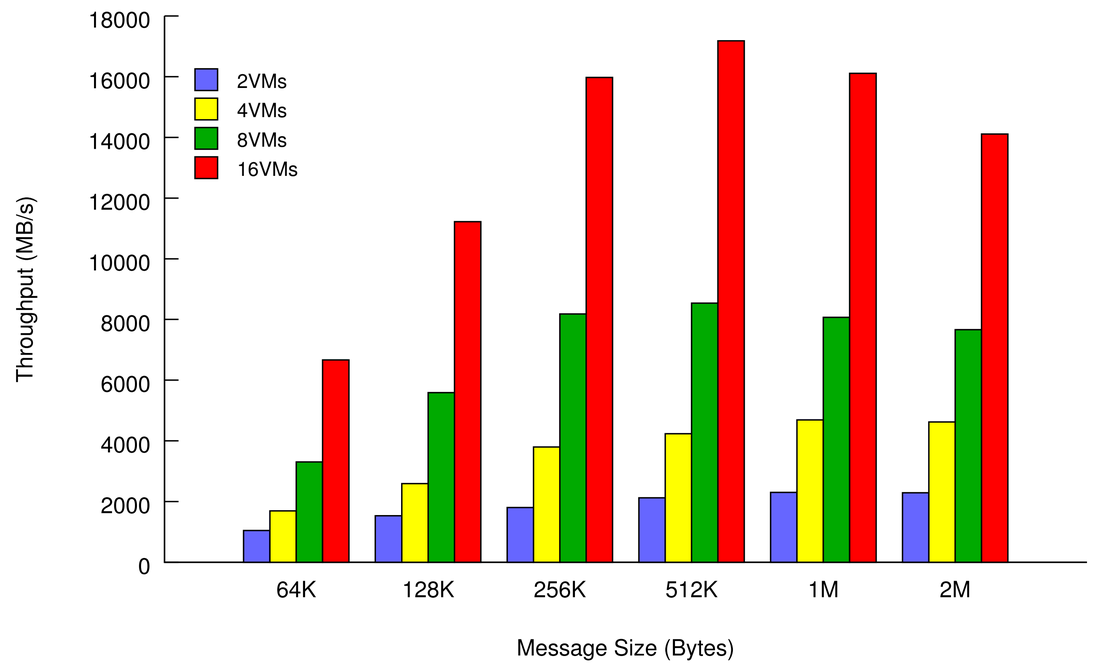
V4VSockets[a], is an efficient, socket-compliant, high performance intra node communication framework.
V4VSockets features:

- ⇒ *optimized* data path (data are copied from / to the VM kernel memory without the need to share pages between VMs)
- ⇒ no intermediary VM (driver domain), so no scheduling implications are involved.
- ⇒ no security implications (data cross the hypervisor and either get dropped or pushed forward using V4V semantics)
- ⇒ ultra *low latency* and *high bandwidth*.

[a] https://github.com/HPSI/v4v

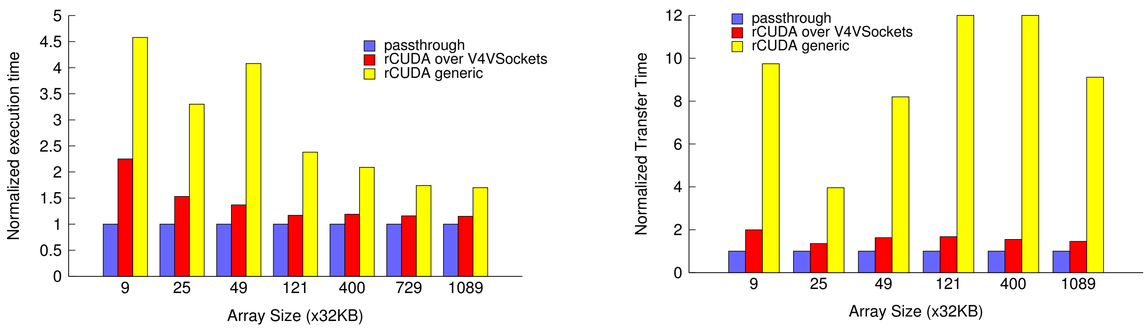
Ping-pong benchmark





- ⇒ improves latency for small messages by 81%
- ⇒ achieves 2299 MB/s for large messages (1 MB) vs. 501 MB/s
- ⇒ scales efficiently with the number of VMs, both in terms of latency and bandwidth
- ⇒ aggregate throughput ≈ 17 GB/s for 512 KB messages when 16 VMs exchange data in pairs

GPU stencil performance through rCUDA



- ⇒ matrix-matrix product benchmark run: (a) natively via GPU passthrough, (b) rCUDA over TCP/IP sockets and (c) rCUDA over V4VSockets
- ⇒ steps: 2x matrix copy, GPU compute, 1x copy back.
- ⇒ adds minimum overhead of 15% (compared to native execution)
- ⇒ Boosts transfer throughput by 7 (at best) compared to TCP/IP

Work in Progress

- ⇒ Strengthen our implementation to a more user-friendly approach,
- ⇒ Thoroughly examine the CPU utilization overheads imposed by V4VSockets,
- ⇒ Polish the peer discovery framework to adaptively use V4VSockets over generic sockets.
- ⇒ Perform an elaborate performance evaluation of the GPU sharing framework we have developed

Contact info and Acknowledgments

Anastassios Nanos[a], Stefanos Gerangelos, Ioanna Alifieraki[b] and Nectarios Koziris
{ananos,sgerag,ialif,nkoziris}@cslab.ece.ntua.gr

The authors would like to thank the members of CSLab for the stimulating conversations that led to this work, especially Dr George Goumas, Dr Konstantinos Nikas and Nikela Papadopoulou. Additionally, many thanks to the anonymous reviewers for their useful comments and suggestions.

[a] Currently with OnApp Ltd.
[b] Currently with the University of Manchester