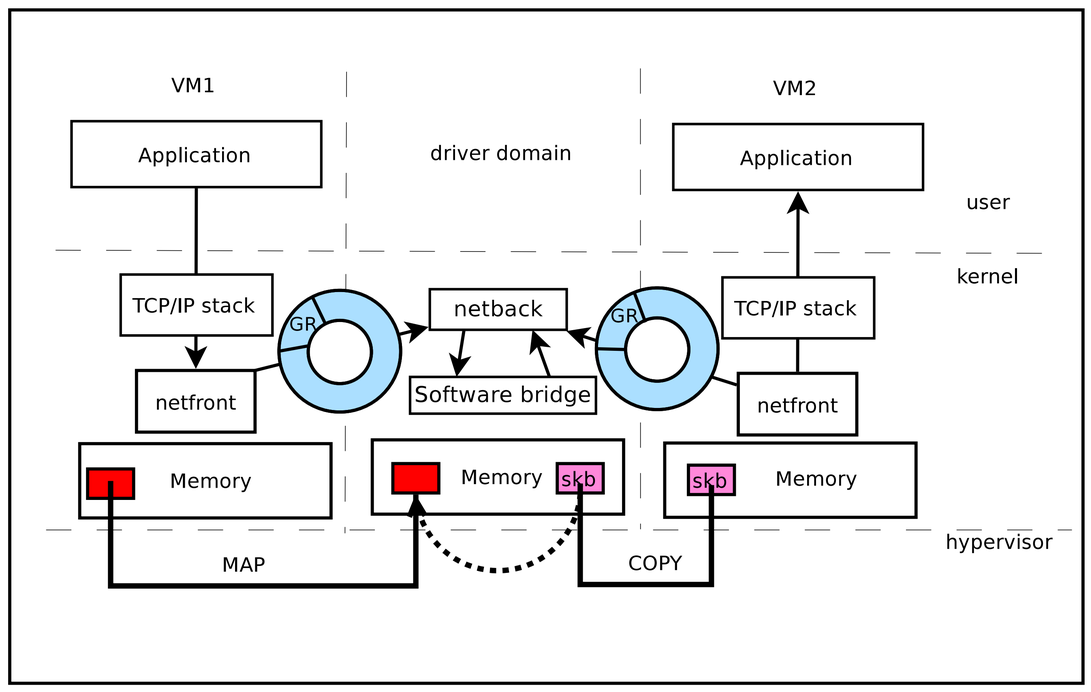
V4VSockets

Low overhead intra-node communication in Xen

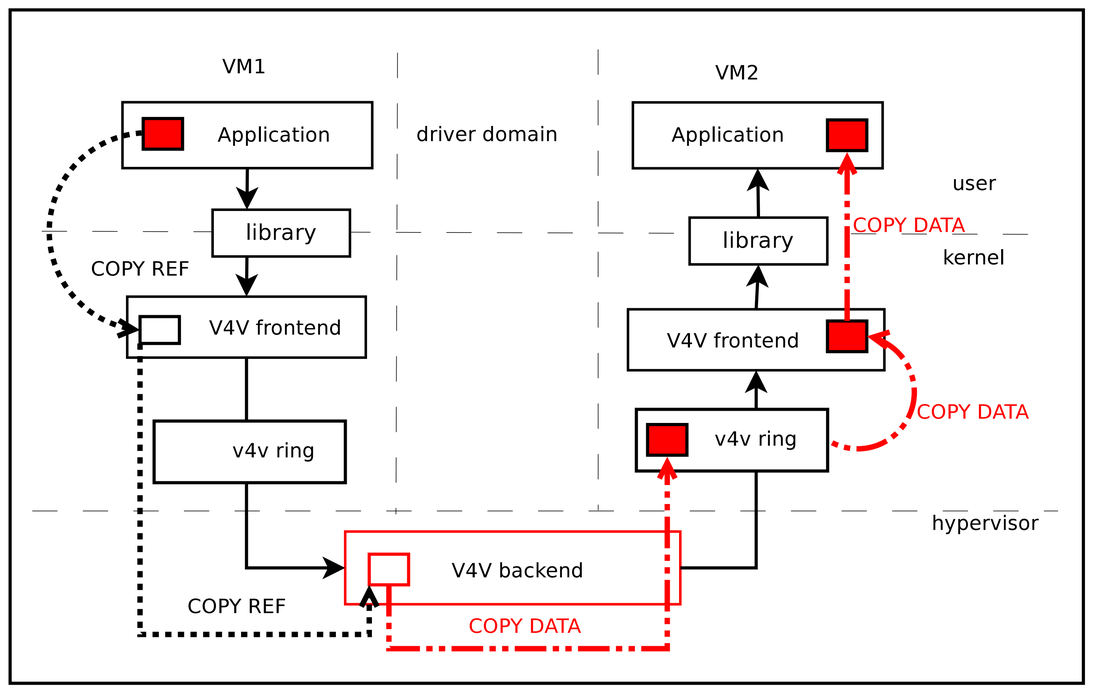
I/O Path in Xen (generic environment)



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

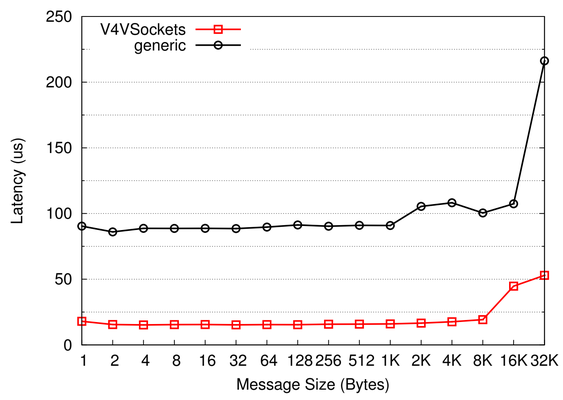
Key features

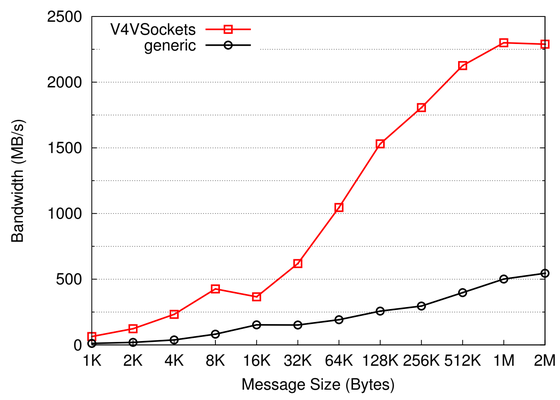
V4VSockets is an efficient, socket-compliant, high performance intra node communication framework for VMs.

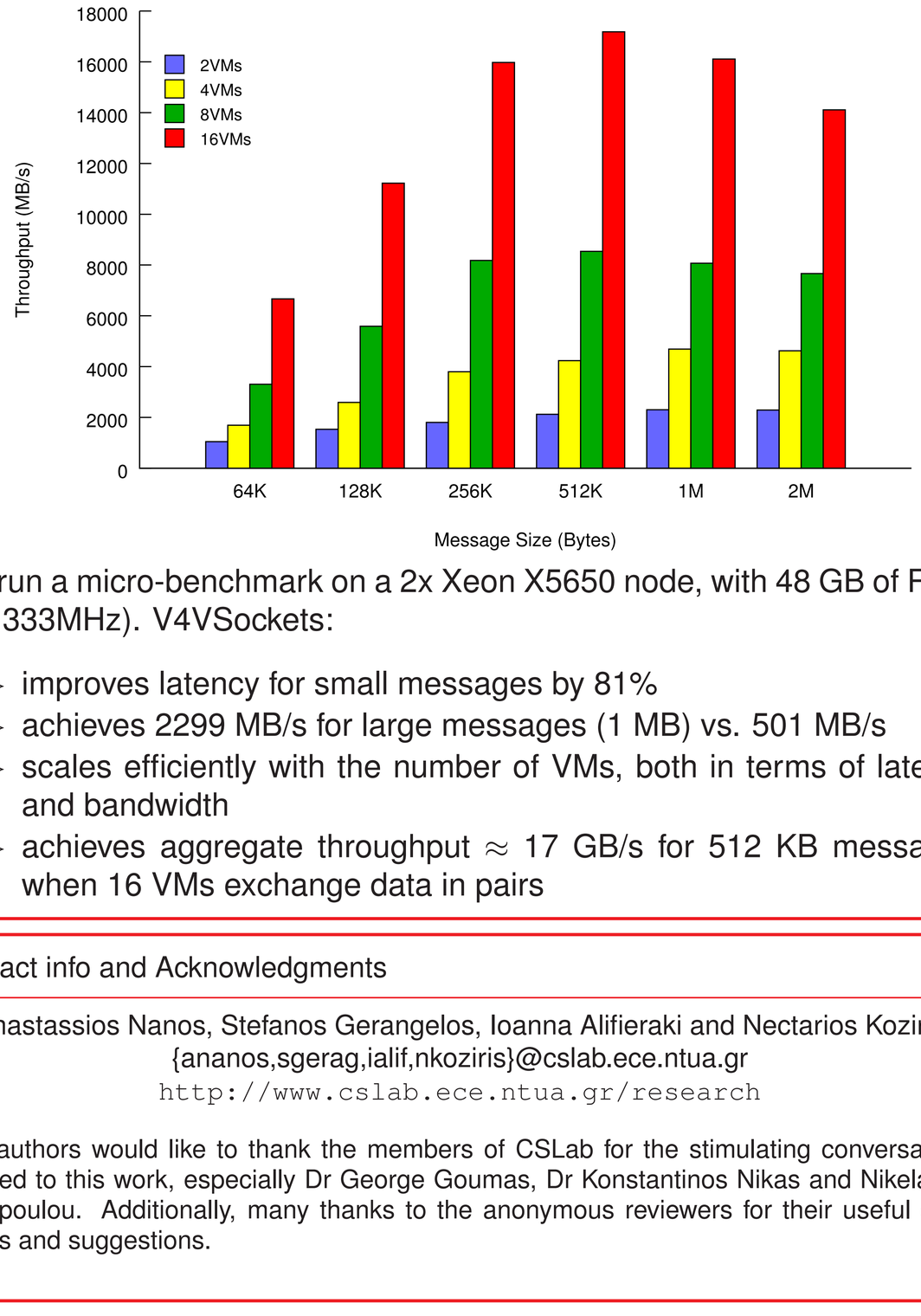
V4VSockets features:

⇒ *optimized data path* (data are copied from / to the VM kernel memory without the need to share pages between VMs)
⇒ *low-overhead* framework (no intermediary VM – driver domain – no scheduling implications are involved)
⇒ *secure* (no security implications – data cross the hypervisor and either get dropped or pushed forward using V4V semantics)
⇒ *ultra low latency* and *high bandwidth*
⇒ open-source, available @ `https://github.com/HPSI/v4v`

Ping-pong benchmark



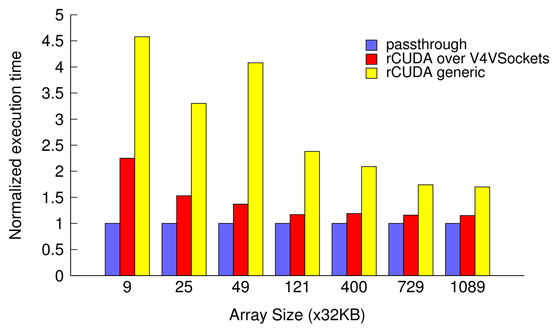


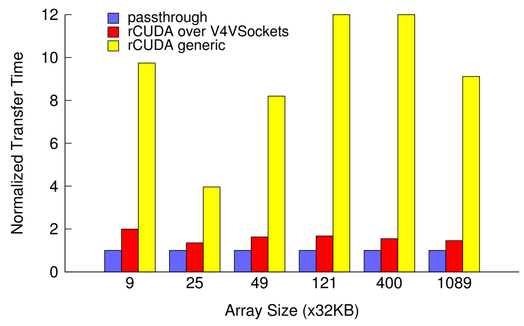


We run a micro-benchmark on a 2x Xeon X5650 node, with 48 GB of RAM (@1333MHz). V4VSockets:

⇒ 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
⇒ achieves aggregate throughput ≈ 17 GB/s for 512 KB messages when 16 VMs exchange data in pairs

GPU stencil performance through rCUDA





We run the matrix-matrix product benchmark: (a) natively via GPU passthrough, (b) via rCUDA over TCP sockets and (c) via rCUDA over V4VSockets. Steps include: (i) 2x input matrix copies, (ii) GPU product compute, (iii) 1x output matrix copy back. V4VSockets:

⇒ adds a minimum overhead of 15% (compared to native execution)
⇒ boosts transfer throughput by 7x (at best) compared to TCP/IP

Work in Progress

⇒ Strengthen our implementation towards 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 TCP sockets.
⇒ Perform an elaborate performance evaluation of the GPU sharing framework we have developed.

Contact info and Acknowledgments

Anastassios Nanos, Stefanos Gerangelos, Ioanna Alifieraki and Nectarios Koziris
{ananos,sgerag,ialif,nkoziris}@cslab.ece.ntua.gr
`http://www.cslab.ece.ntua.gr/research`

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.