V4VSockets: low-overhead intra-node communication in Xen.

Anastassios Nanos, Stefanos Gerangelos, Ioanna Alifieraki, Nectarios Koziris

High-Performance Systems and Interconnects (HPSI),
Computing Systems Lab
National Technical University of Athens
Github: `http://github.com/HPSI`
WWW: `http://cslab.ece.ntua.gr/research/`

Apr. 21st, 2015

Intro

Cloud computing

- application oriented
- fast, ease-of-use

Consolidation

- $\frac{vCPU}{physicalcores} >> 1$
- multi/many–cores

The number of co-located VMs is drastically increasing

Introduction

Applications

- stand-alone (flexibility)
- distributed (elasticity)

When it comes to down to numbers, the need for intra-node communication in small data center increases.

Introduction

Applications

- stand-alone (flexibility)
- distributed (elasticity)

When it comes to down to numbers, the need for intra-node communication in small data center increases.

Contribution

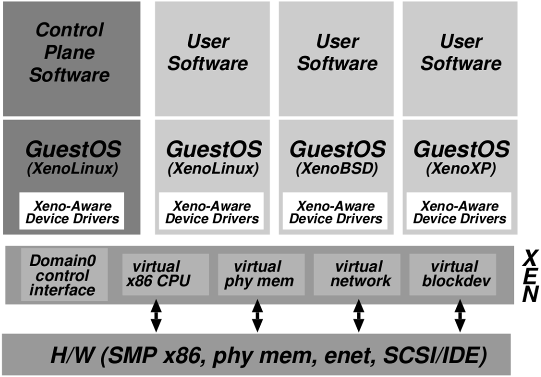
Design and Implement V4VSockets:

- efficient message exchange (one order of magnitude)
- isolation
- API compatible (Sockets)

Overview

Xen - Architecture

- hypervisor & privileged VM (driver domain) to access hardware



Overview

I/O Internals – Xen

Xen basics

- hypervisor – driver domain runs as a Linux guest
- split driver model (frontend/backend)

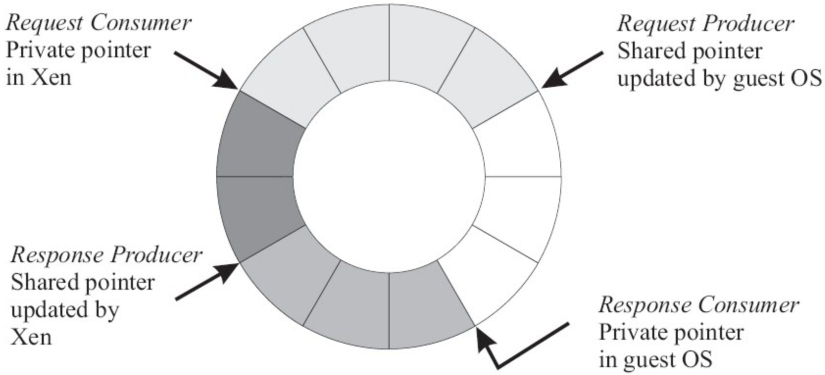
Xen – Event channels

- notify Guest/Host about a pending transaction
- easy to setup – bind to a specific "port"

Xen – Grant mechanism

- issue a page grant request
- the other end maps the grant (accept)
- this page is shared across the two domains

Xen Ring buffers



Request queue - Descriptors queued by the VM but not yet accepted by Xen

Outstanding descriptors - Descriptor slots awaiting a response from Xen

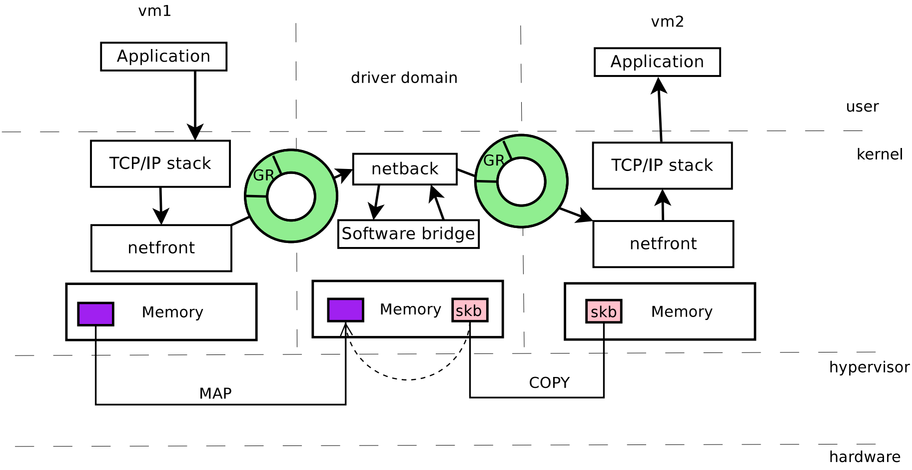
Response queue - Descriptors returned by Xen in response to serviced requests

Unused descriptors

Overview

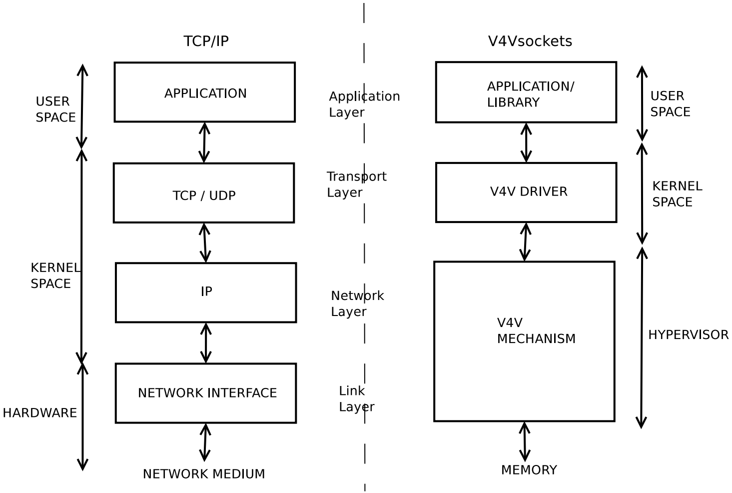
1. Communication in Virtual Environments
   - Basic Concepts
2. I/O access in Virtualized Environments
   - Xen
   - Intra-node communication in Xen
3. V4VSockets
   - Architecture
   - Experimental Evaluation

Intra-node communication in Xen



Overview

V4VSockets Architecture



V4VSockets Architecture

Application/Library layer

- forwards the relevant actions and arguments to the transport layer.

V4VSockets Architecture

Transport layer – V4V kernel driver

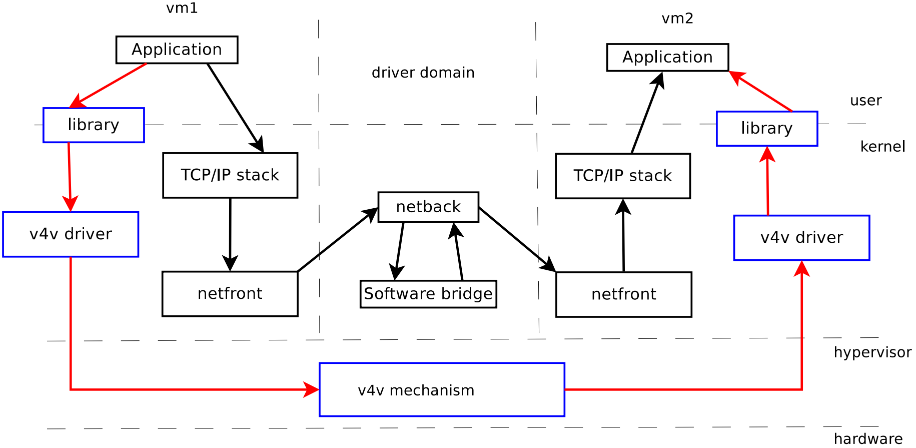
- handles the virtual connection semantics between peer VMs that need to communicate,
- is in charge of fragmenting and sending upper-layer packets by issuing hypercalls to the hypervisor (network layer), and
- provides a notification mechanism to the VM's user-space for receiving packets, as well as error control.

V4VSockets Architecture

Network/Link layer - Hypervisor

- encapsulation of upper-layer messages to packets that will be transmitted to their destination, according to V4V semantics,
- packet delivery.

V4Vsockets vs Netfront/Netback



V4VSockets

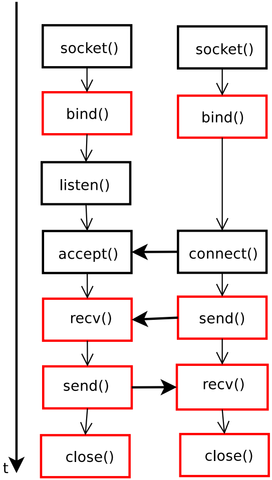
Communication mechanisms in V4VSockets

- system calls
- hypercalls
- event channels - VIRQS

V4V Rings

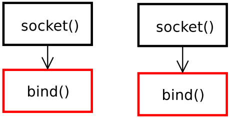
- circular buffers
- allocated in the VM address space by the VM kernel
- data exchange medium

V4VSockets – Message Exchange

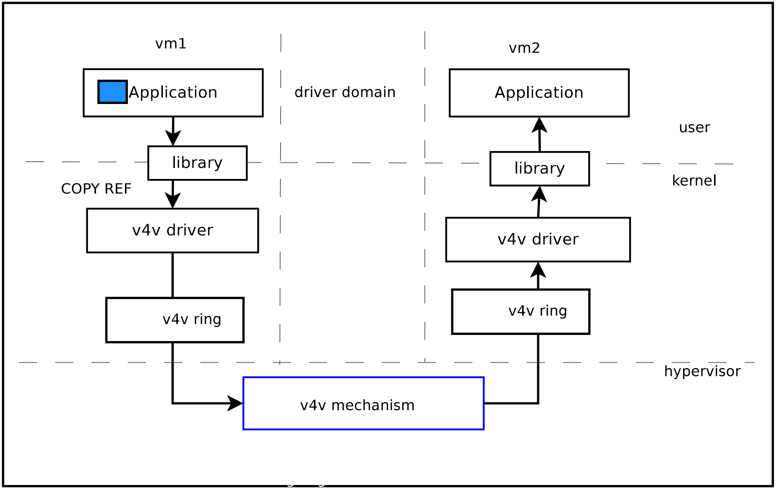


V4VSockets – Message Exchange

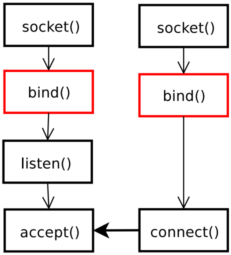
V4VSockets – Message Exchange



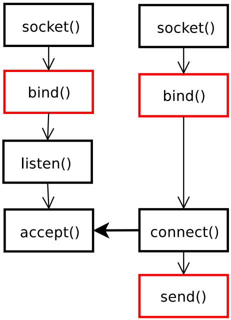
V4VSockets – Message Exchange



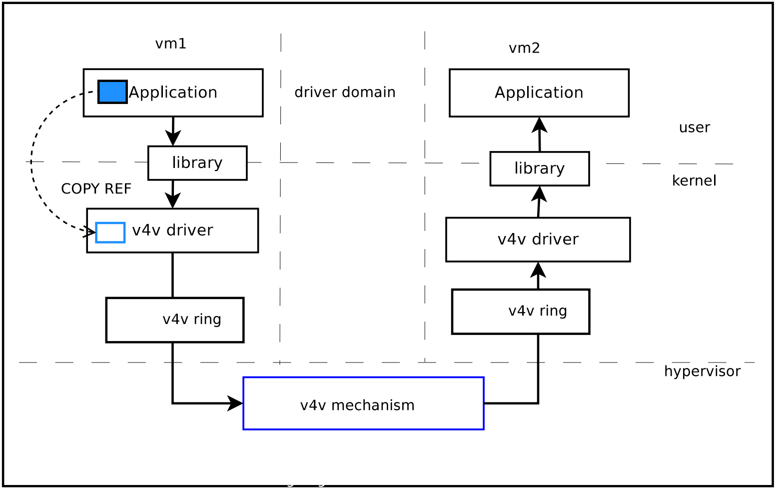
V4VSockets – Message Exchange



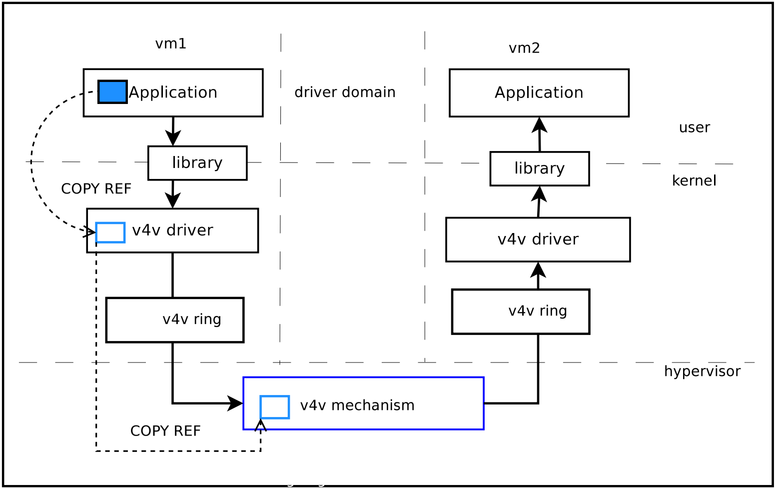
V4VSockets – Message Exchange



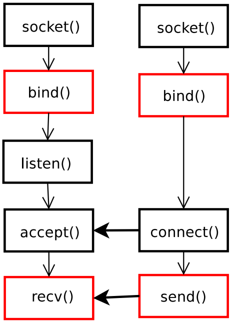
V4VSockets – Message Exchange



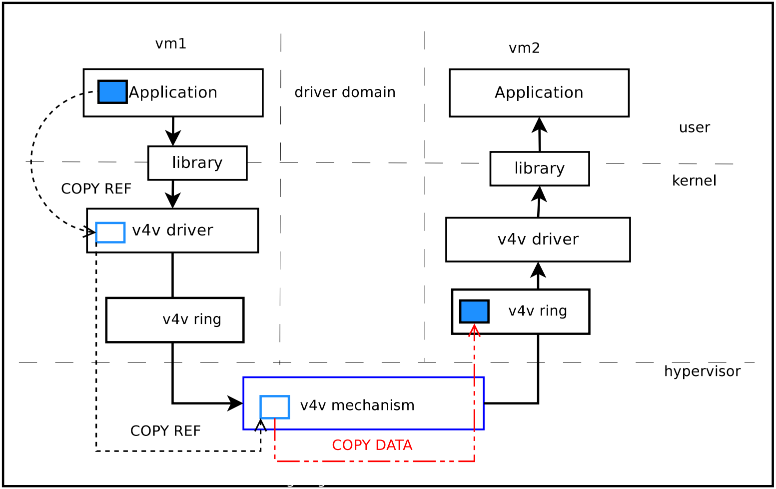
V4VSockets – Message Exchange



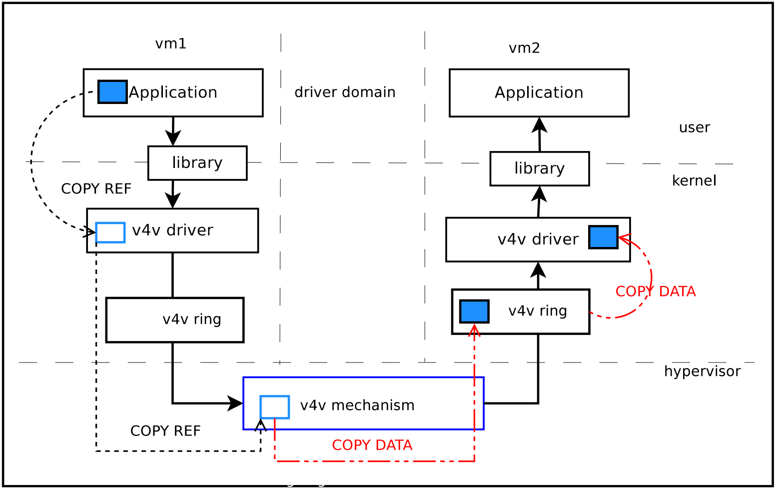
V4VSockets – Message Exchange



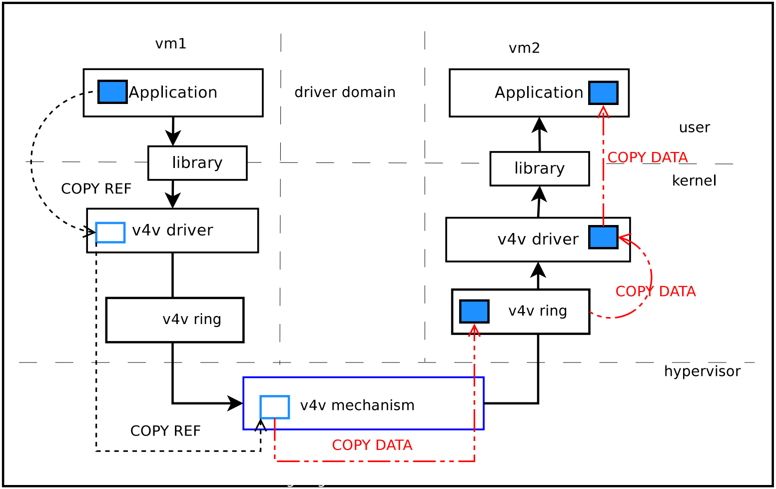
V4VSockets – Message Exchange



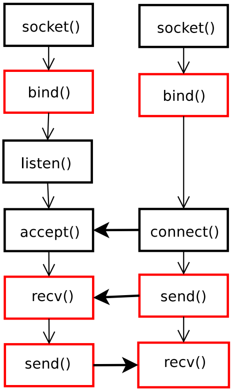
V4VSockets – Message Exchange



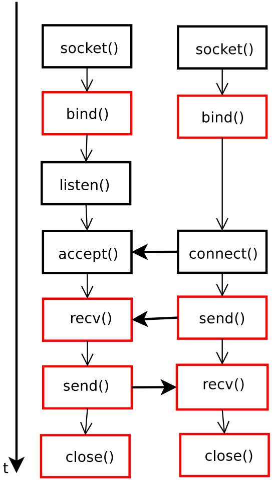
V4VSockets – Message Exchange



V4VSockets – Message Exchange



V4VSockets – Message Exchange



Overview

National Technical University of Athens CSLab

Performance Results

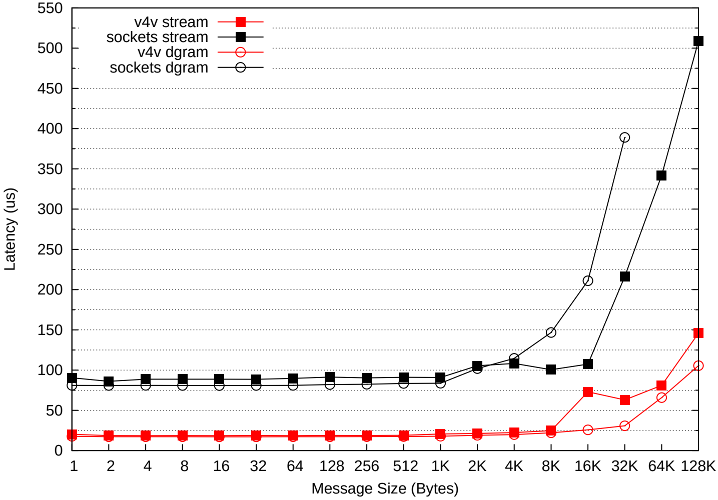
Testbed

- 2x {Intel Xeon @2.4Ghz}, Intel 5520, 48GB memory, Generic 10GbE
- Xen 4.5-unstable, Debian GNU/Linux (Linux kernel 3.14.2)
- generic micro-benchmark: `pingpong`

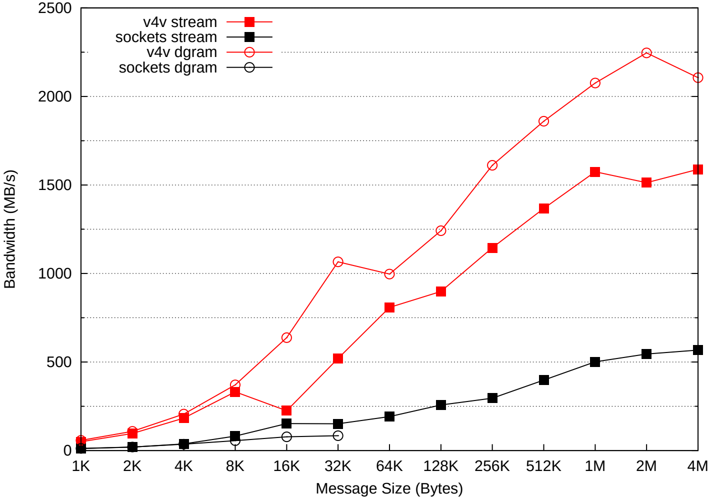
Cases:

- TCP/UDP sockets
- V4V Stream/Datagram

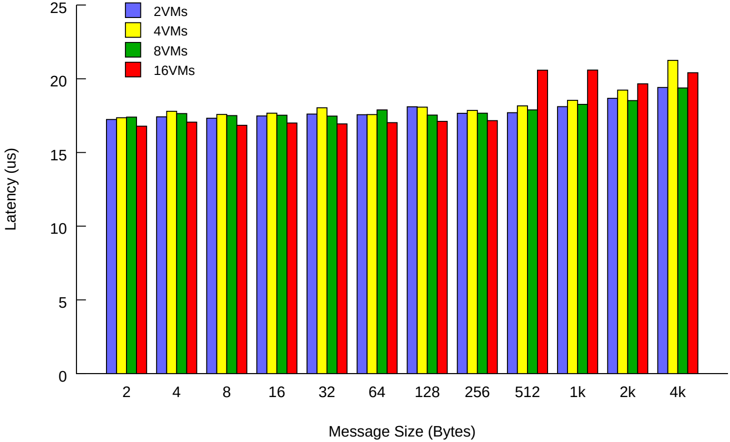
Experimental evaluation – RTT latency



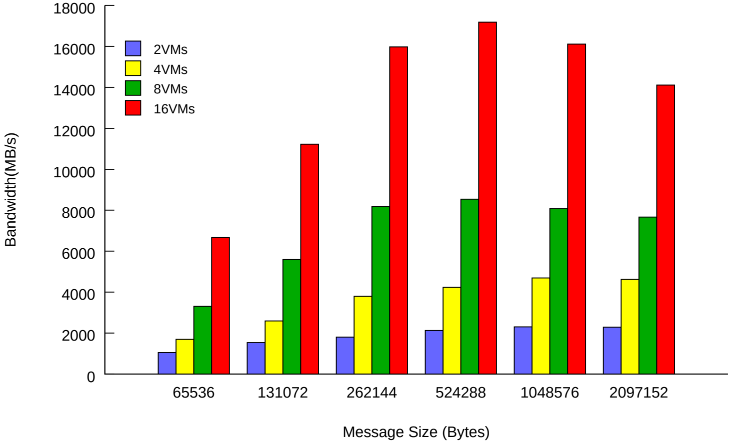
Experimental evaluation – Bandwidth



Experimental evaluation – Scaling factor (up to 16 VMs)



Experimental evaluation – Scaling factor (up to 16 VMs)



Summary

Intra-node communication in VM environments

- split driver model – generic
- V4VSockets: framework for low-overhead intra-node communication
  - is not based on a driver domain
  - does not use shared memory between guests (map/grant mechanism)
  - uses memory copies, hypercalls and event channels

- better throughput (efficient data path, bypass the complex TCP/IP stack)
- hypercall overheads (small, negligible if correctly finetuned)
- scalability (no privileged guest involved in communication)
- isolation (no shared memory)
- extensible

Future endaevors

- cpu utilization overheads
- NUMA and multihierarchical memory architectures
- map instead of copy (study the systems behavior of providing a shared memory space between VMs)
- Optimize away copies on the data path.

Available online as open-source

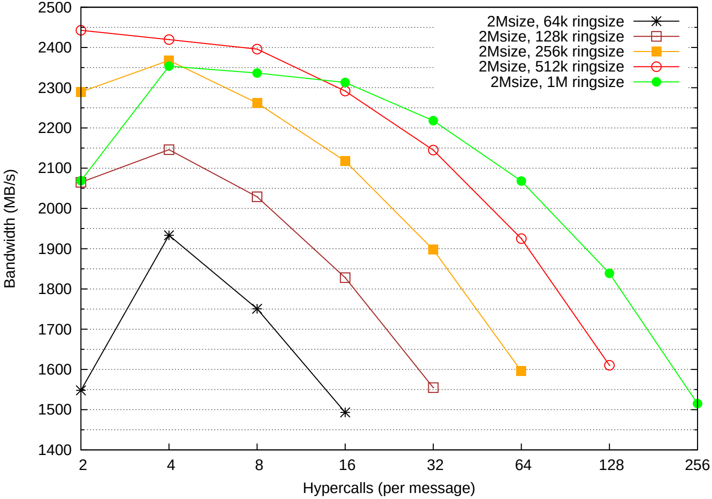
`https://github.com/HPSI/V4VSockets`

Thanks!

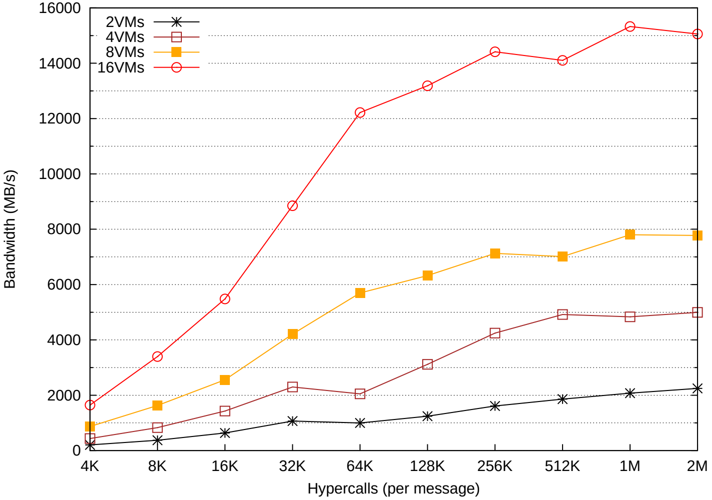
Questions?

Backup

Experimental evaluation – 2M message size vs Hypercalls



Experimental evaluation – Datagram scalability



Experimental evaluation – Hypercalls per message