virtio-vsock in QEMU, Firecracker and Linux

Status, Performance and Challenges

KVM Forum 2019 Lyon, France
31st October 2019

Andra Paraschiv, Amazon Web Services
Stefano Garzarella, Red Hat

aws

Red Hat

Agenda

State of the Art

Use Cases

vhost Backend

Firecracker and virtio-vsock

Firecracker - vsock and UNIX Domain Sockets

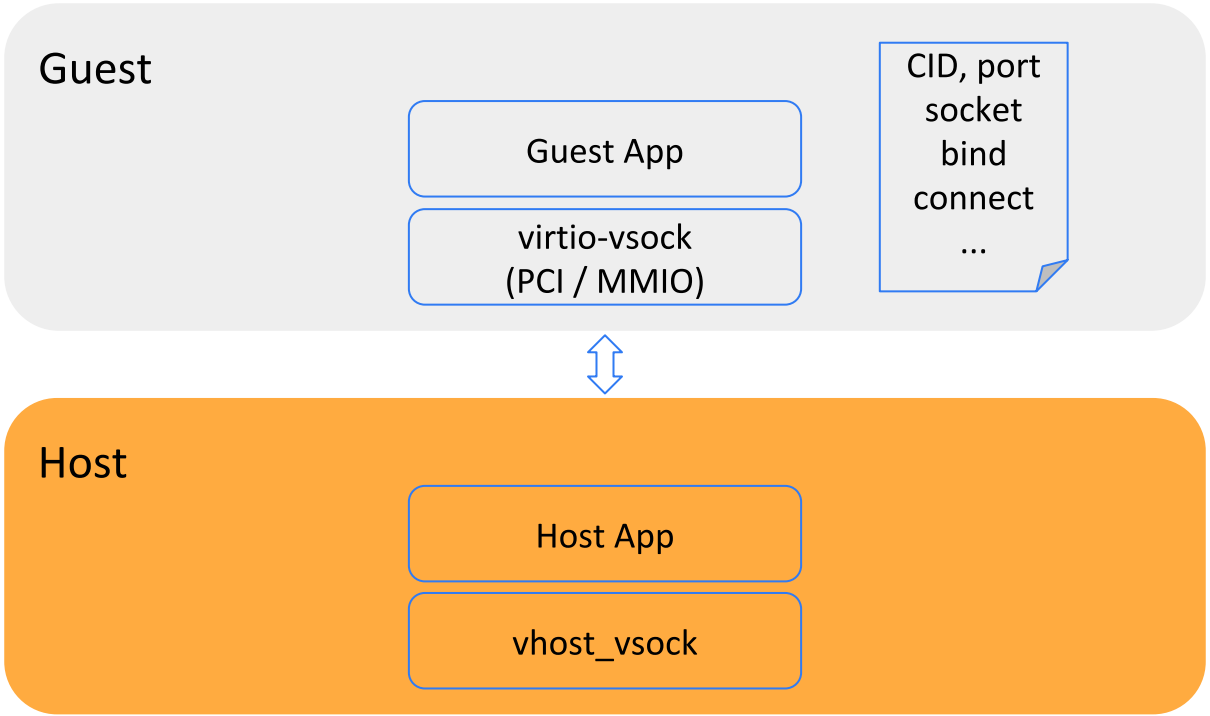
QEMU and vsock

vsock Linux Drivers - Changes in the Last Year

Tools Supported

Next Steps

State of the Art



Use Cases

Guest Agents

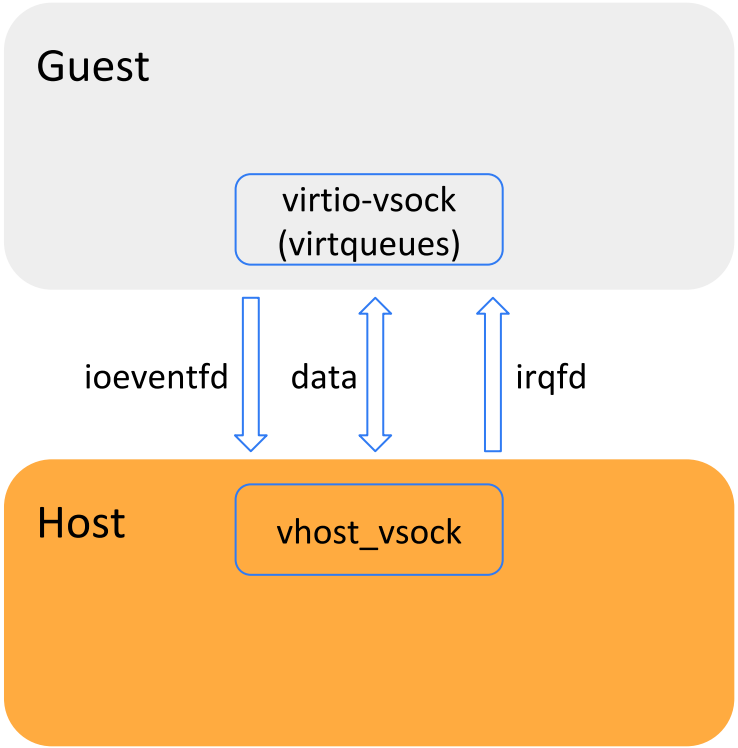
Network Applications
(SOCK_STREAM)

Hypervisor Services

aws

vhost Backend

- Linux kernel driver(s)
  - vsock
  - vhost_vsock
- virtio device emulation
  - net
  - scsi
  - vsock



Firecracker and virtio-vsock

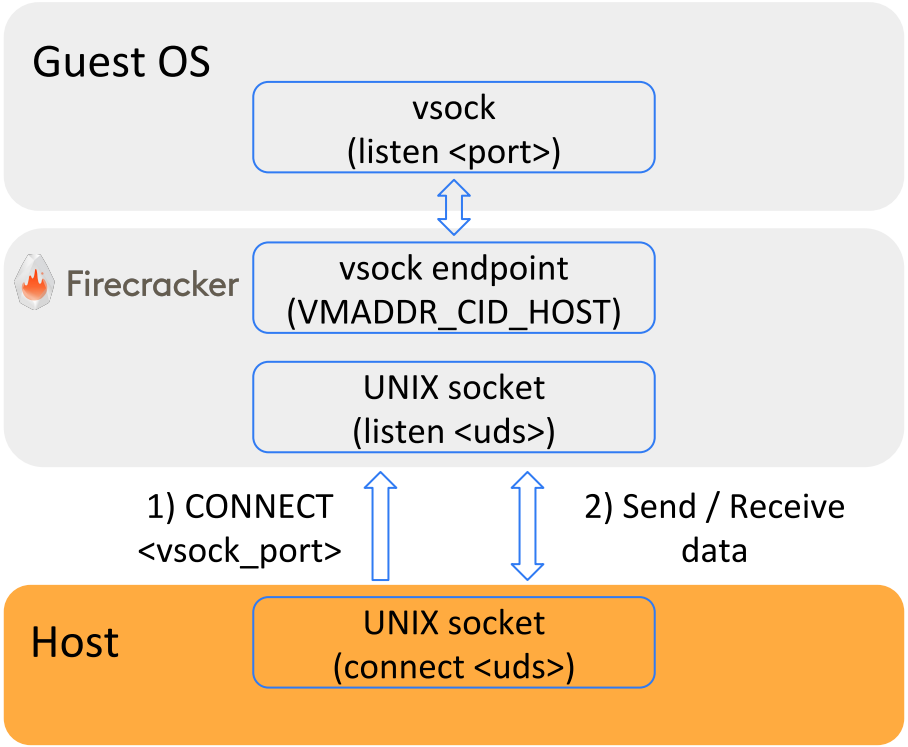
- Open source Virtual Machine Monitor (VMM)
- Experimental vhost (removed in v0.18.0)
- vhost-less solution - why? (in tree since v0.18.0)
  - Reduced security impact (e.g. privilege escalation)
  - Less dependency on host kernel features
- virtio-vsock device model over MMIO
- UNIX Domain Sockets (UDS) on host

aws

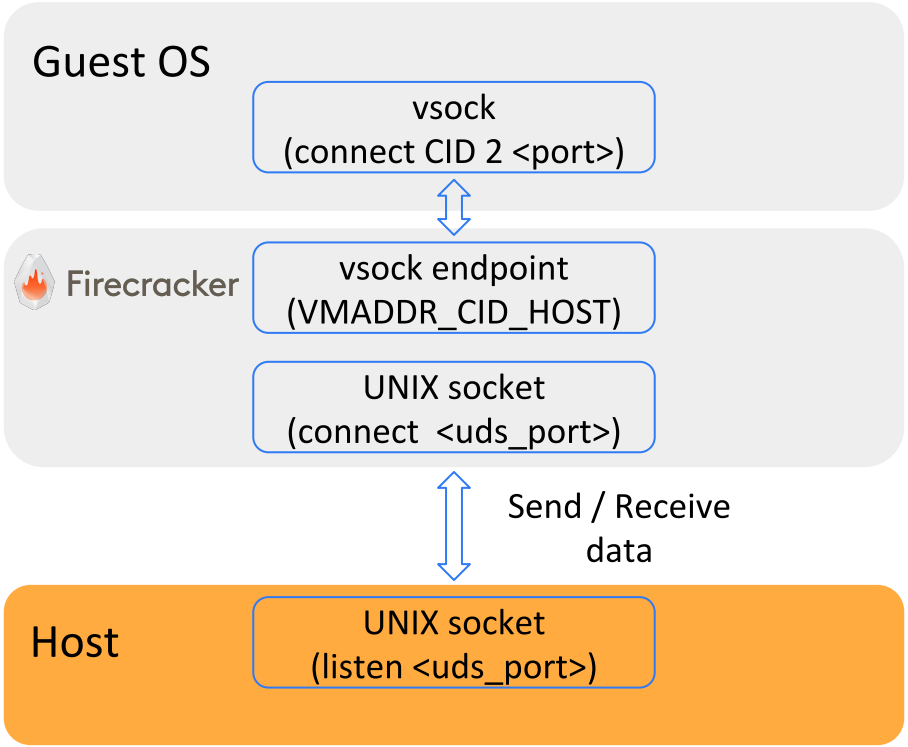
Firecracker - vsock and UNIX Domain Sockets

- One virtio-vsock device per VM
  - vsock config
    - Guest CID
    - UDS path
  - VMM listening on UNIX socket
- vsock connection
  - Host-initiated
  - Guest-initiated

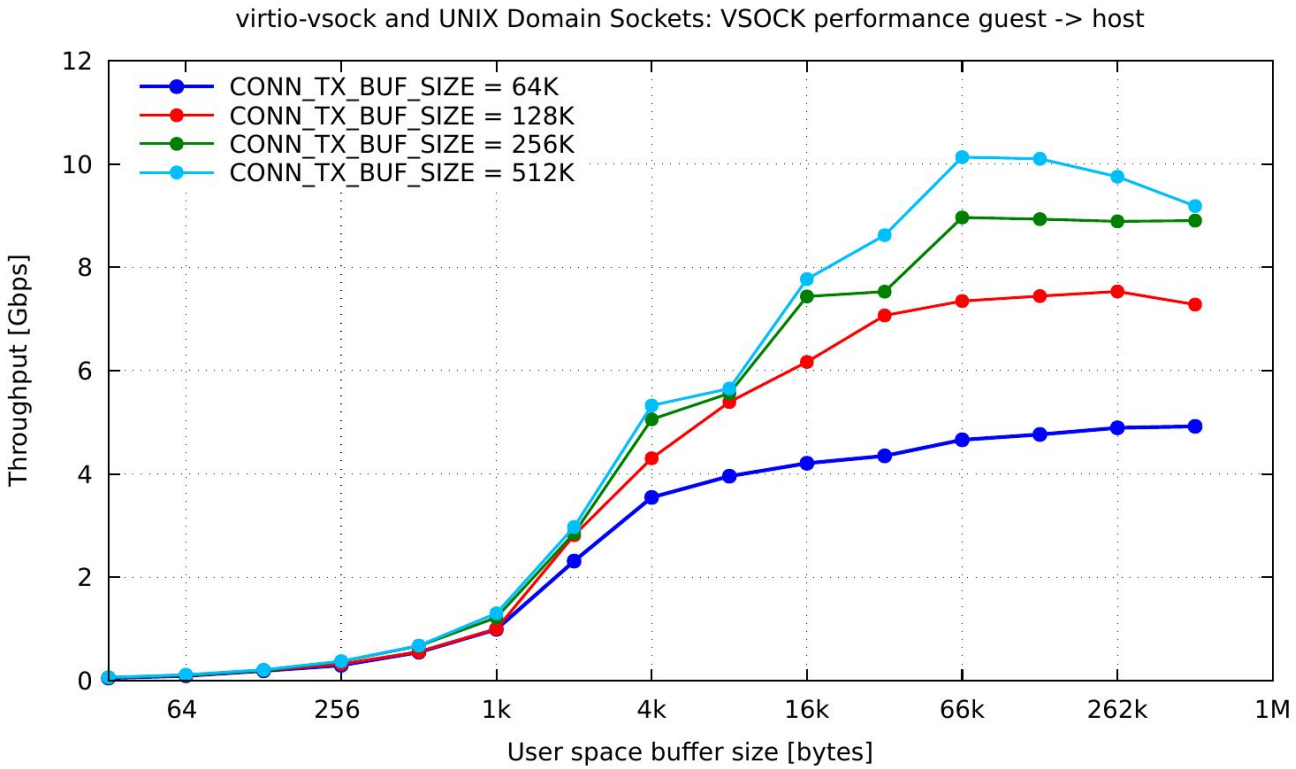
Firecracker - vsock and UNIX Domain Sockets (2)



Firecracker - vsock and UNIX Domain Sockets (3)



Firecracker - vsock and UNIX Domain Sockets (4)

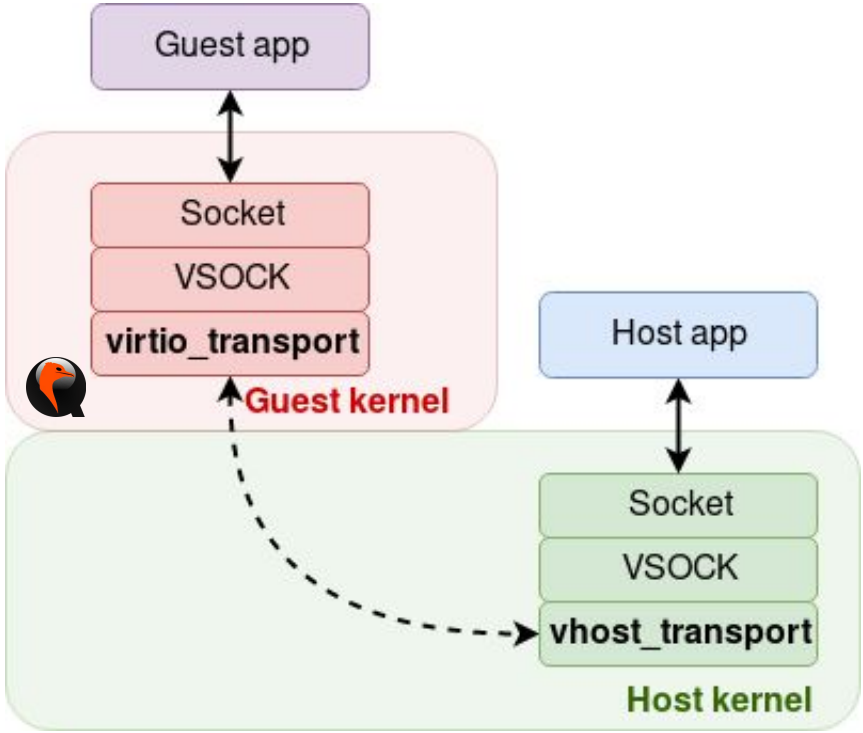


QEMU and vsock (1/2)

- `qemu-system-x86_64 … -device` **`vhost-vsock-pci,guest-cid=3`**
- QEMU vhost-vsock device
  - configuration
    - set guest CID (chosen by user or management tool)
  - live migration
    - connected SOCK_STREAM sockets become disconnected
    - guest's CID may change
      - device notifies the guest that the CID has changed

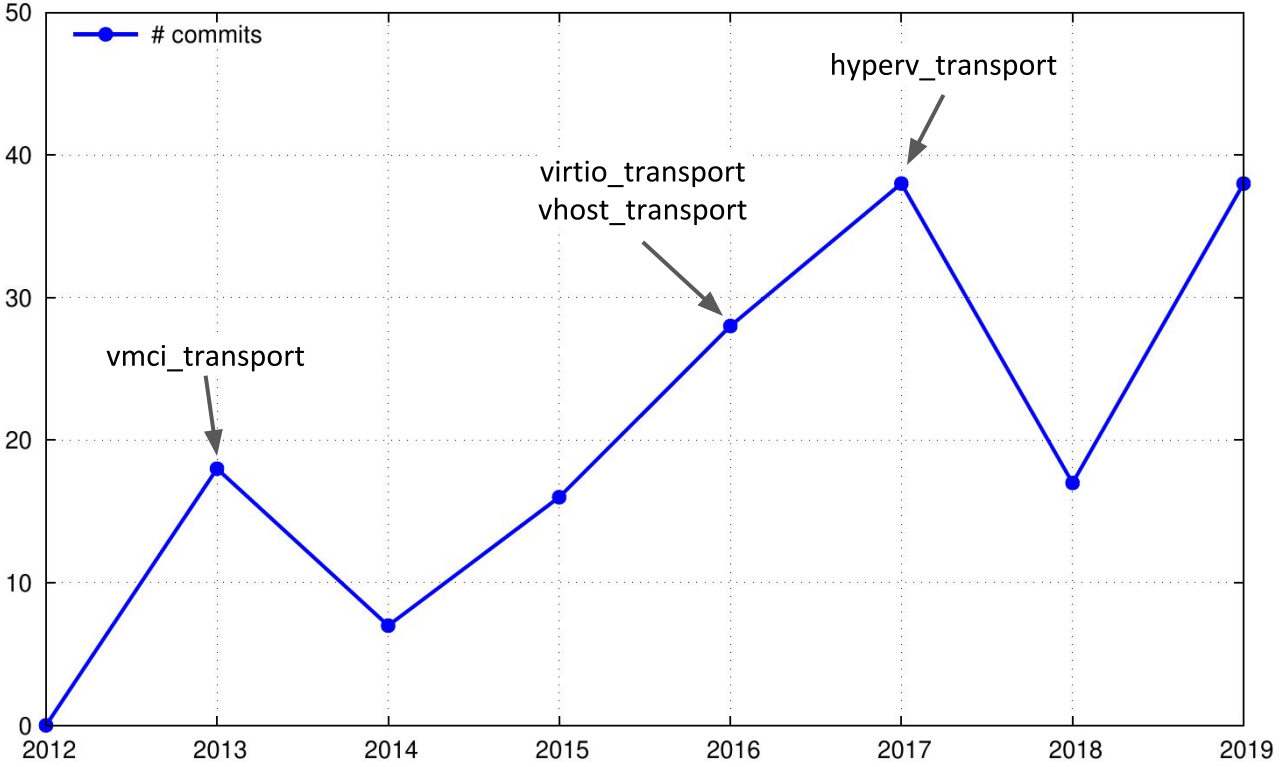
QEMU and vsock (2/2)

- vhost kernel driver
  - handles most of the host work
    - guest/host data transfer
    - interface with the host socket layer
      - socket(AF_VSOCK, ...) works for host applications
- vsock transport drivers
  - virtio_transport (guest)
  - vhost_transport (host)



Red Hat

vsock Linux drivers: commits per year



Red Hat

vsock Linux drivers: changes in the last year (1/2)

45 changesets from 16 developers

Top changeset contributors by employer

Red Hat	21	(46.7%)
Microsoft	8	(17.8%)
Linutronix	7	(15.6%)
ytht.net@gmail.com	2	(4.4%)
Intel	1	(2.2%)
VMware	1	(2.2%)
Google	1	(2.2%)
Bitdefender	1	(2.2%)
Alibaba	1	(2.2%)
The Chromium Projects	1	(2.2%)
arnd@arndb.de	1	(2.2%)

A total of 674 lines added, 426 removed (delta 245)

Top lines changed by employer

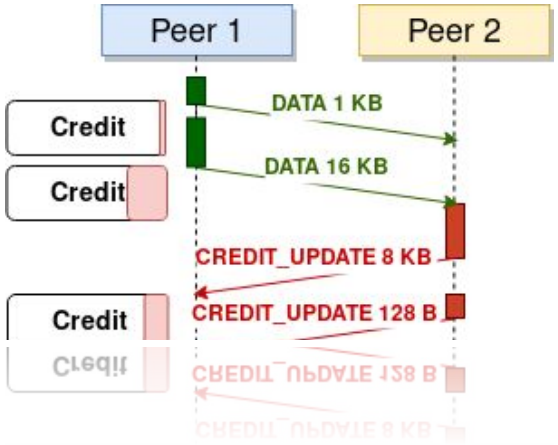
Red Hat	370	(44.6%)
Microsoft	234	(28.2%)
Linutronix	129	(15.5%)
VMware	50	(6.0%)
Bitdefender	15	(1.8%)
Intel	12	(1.4%)
ytht.net@gmail.com	7	(0.8%)
Google	7	(0.8%)
The Chromium Projects	3	(0.4%)
arnd@arndb.de	2	(0.2%)
Alibaba	1	(0.1%)

vsock Linux drivers: changes in the last year (2/2)

- Fixed several bugs, races and memory leaks in the vsock-core and transports
- virtio/vhost transports
  - **Performance improvements**
    - reduced number of credit update messages exchanged
    - allow up to 64 KB packets queued
  - Fixes
    - close and release paths
    - hot-plug and hot-unplug
    - hashing to map CID to a vsock object
    - ...

virtio-vsock: credit mechanism (1/4)

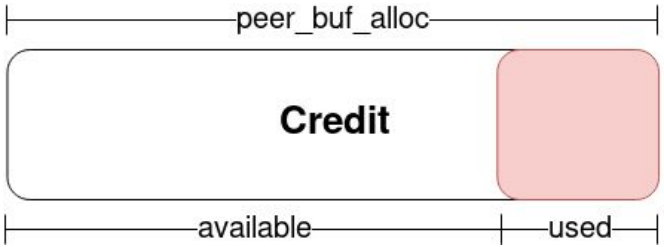
- credit-based flow control for reliable connections
- each peer stores these variables in the socket state:
  - **`buf_alloc`**
    - receive buffer size
      - number of bytes queued to be read by the user
    - configurable through `setsockopt()`
    - default value: 256 KB
  - **`fwd_cnt`**
    - number of bytes received
    - increased when bytes are consumed by the user space
  - **`tx_cnt`**
    - number of bytes sent
    - increased when bytes are sent to the other peer



Red Hat

virtio-vsock: credit mechanism (2/4)

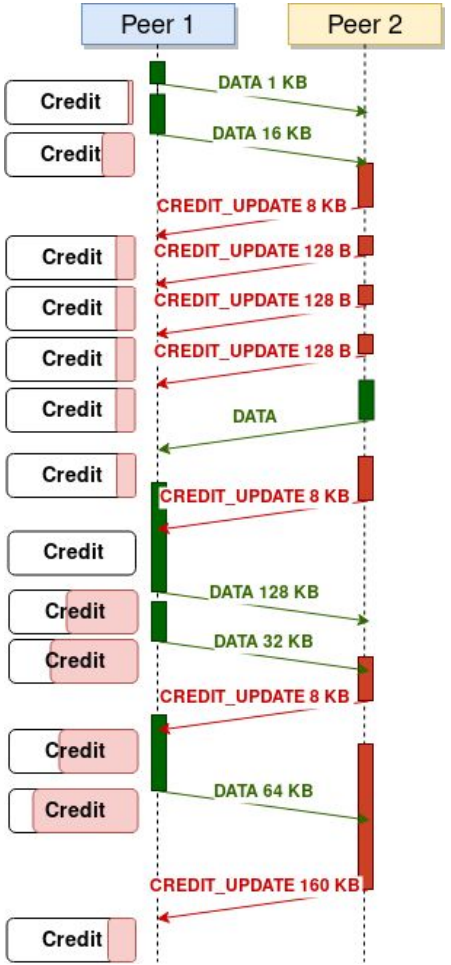
- **`'buf_alloc'`** and **`'fwd_cnt'`** are sent to the other peer:
  - in all packets header
  - with an explicit message
    - `op = VIRTIO_VSOCK_OP_CREDIT_UPDATE`
  - peer stores these values
    - **`peer_buf_alloc`**
    - **`peer_fwd_cnt`**



- `credit_used = tx_cnt - peer_fwd_cnt`
- `credit_available = peer_buf_alloc - credit_used`

virtio-vsock: credit mechanism (3/4)

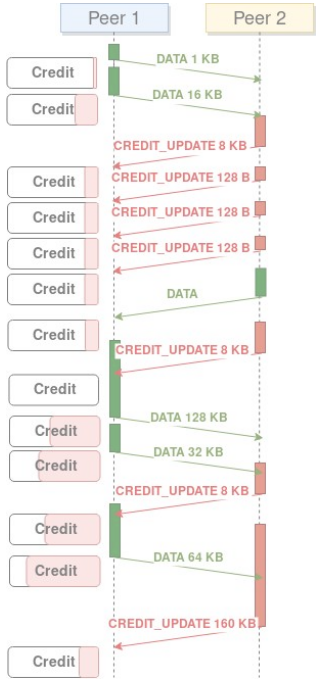
- **Improvement:** reduce the credit messages exchanged
  - **before:** send a credit update message every time the user consumes some bytes received
    - a lot of messages sent to transmitter to update the credit

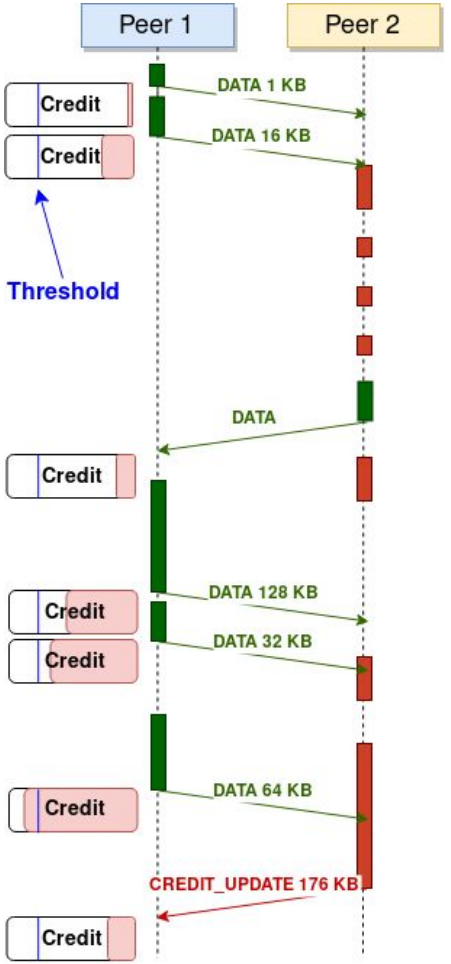


Red Hat

virtio-vsock: credit mechanism (4/4)

- **optimization**
  - send the credit update message only when the credit available seen by the transmitter is less than a threshold
    - VIRTIO_VSOCK_MAX_PKT_BUF_SIZE [64 KB]
  - DATA packets continue to carry credit information

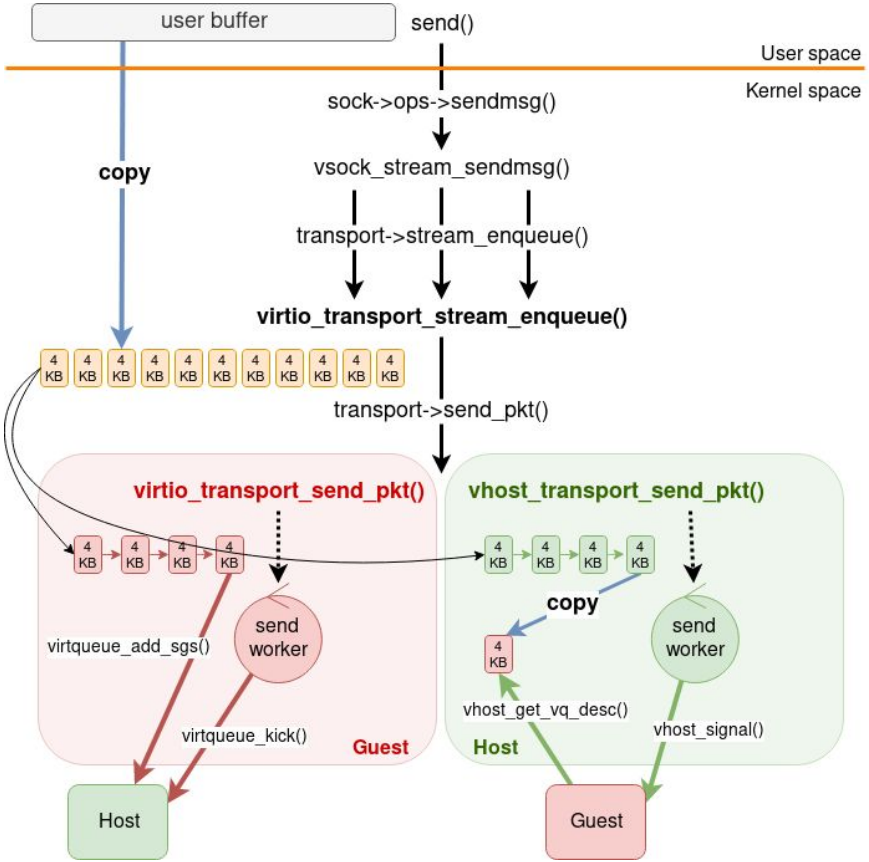




Red Hat

virtio-vsock: send() path (1/2)

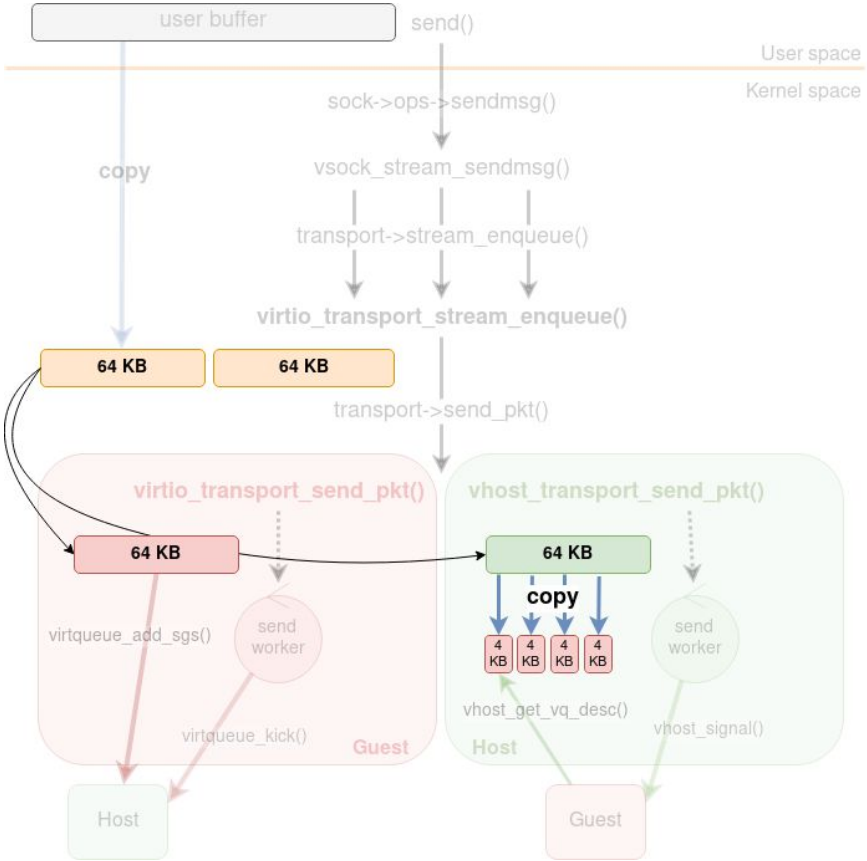
- vsock_stream_sendmsg()
  - call multiple times transport->send_enqueue() to send the entire user buffer
- virtio_transport_stream_enqueue()
  - common code between host and guest drivers
  - **improvement:** enqueue bigger packets
    - **before**: handles up to 4 KB of data to fit the RX guest buffers (4 KB)



Red Hat

virtio-vsock: send() path (2/2)

- **optimization**: handles up to 64 KB of data
  - Host can handle 64 KB packets queued and split them to fill the guest RX buffers (4 KB)
  - Guest can now send up to 64 KB packets to the Host
- other improvements to explore:
  - tunable buffer size
  - mergeable buffers
  - remove workers
  - page allocation instead of kmalloc
  - avoid double allocation per packet

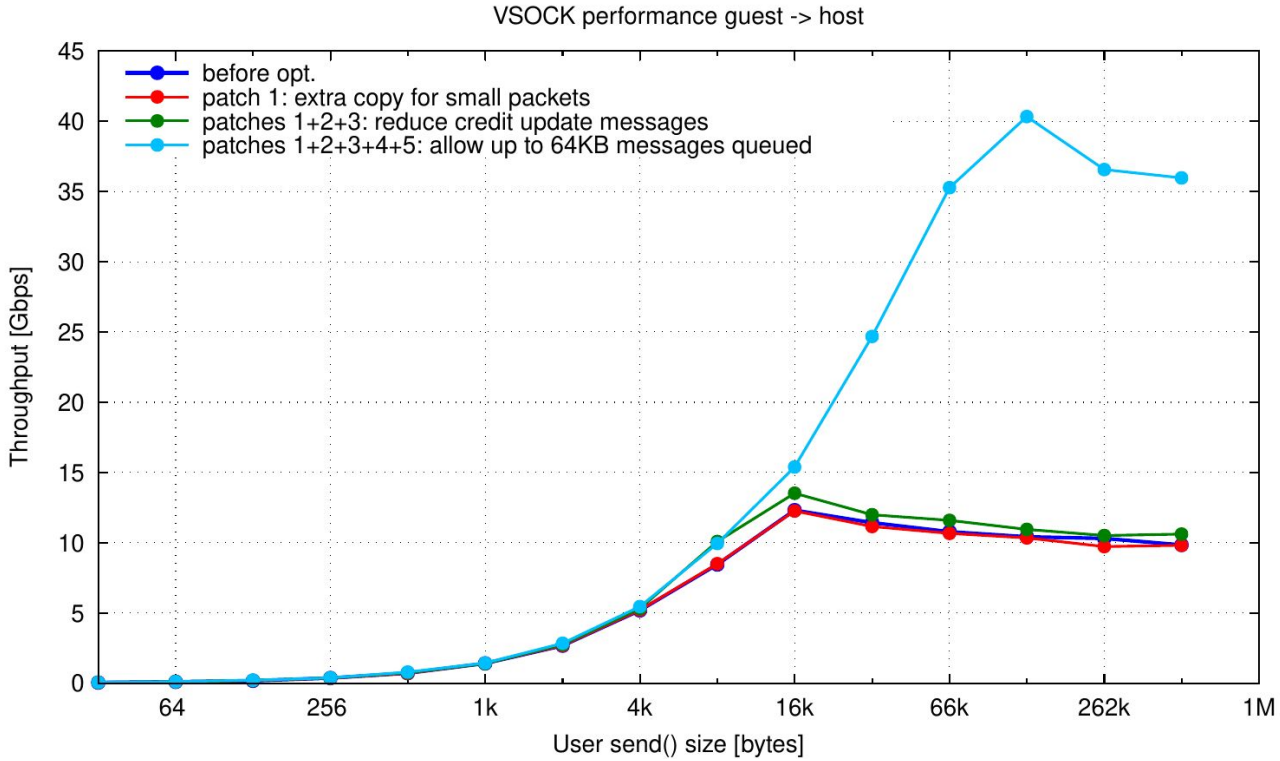


Red Hat

virtio-vsock: performance improvements (1/3)

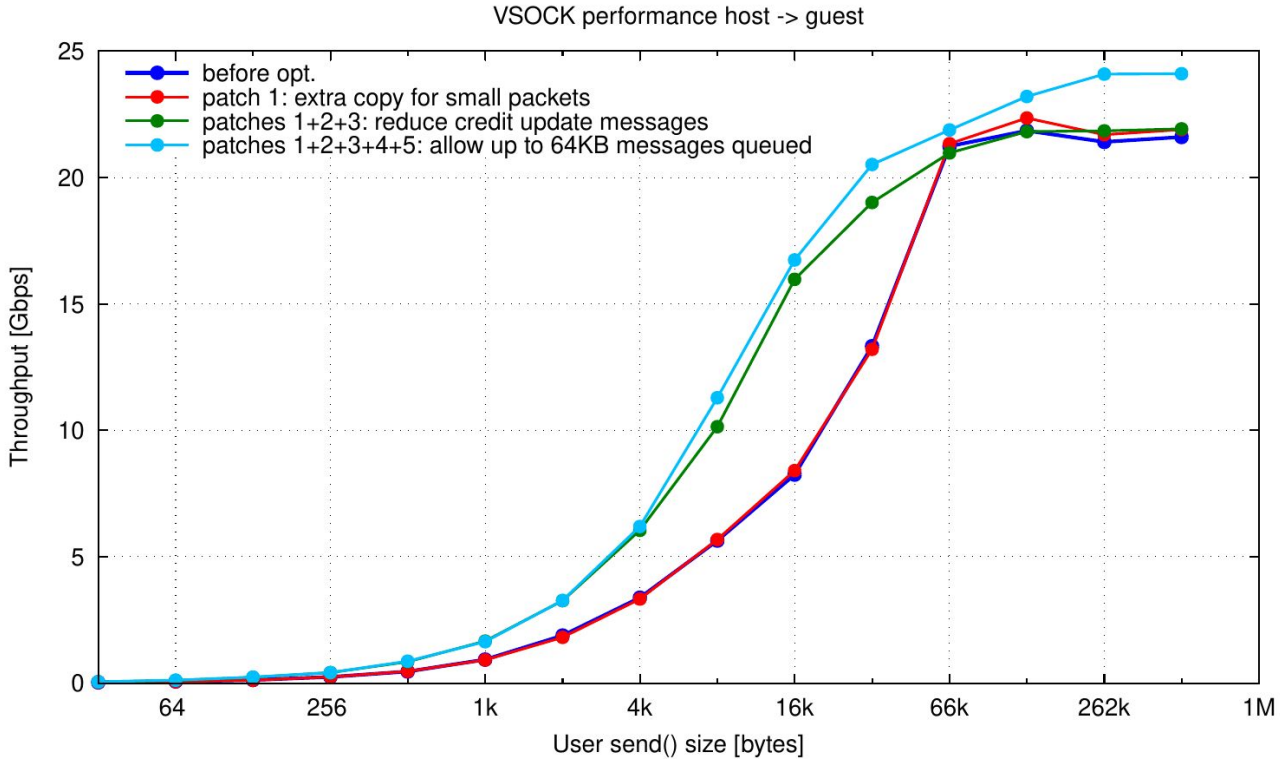
- [PATCH v5 0/5] vsock/virtio: optimizations to increase the throughput
  https://patchwork.ozlabs.org/cover/1139050/
  - **[PATCH v5 1/5] vsock/virtio: limit the memory used per-socket**
    - limit the memory usage with an extra copy for small packets
  - **[PATCH v5 2/5] vsock/virtio: reduce credit update messages**
  - **[PATCH v5 3/5] vsock/virtio: fix locking in virtio_transport_inc_tx_pkt()**
    - reduce the number of credit update messages sent to the transmitter
  - **[PATCH v5 4/5] vhost/vsock: split packets to send using multiple buffers**
  - **[PATCH v5 5/5] vsock/virtio: change the maximum packet size allowed**
    - allow the host to split packets on multiple buffers and use VIRTIO_VSOCK_MAX_PKT_BUF_SIZE as the max packet size allowed
- Upstream (Linux v5.4)
- Benchmark tool: iperf3 [9] modified with VSOCK support.

virtio-vsock: performance improvements (2/3)



Red Hat

virtio-vsock: performance improvements (3/3)



Red Hat

Tools and languages that support VSOCK

Tools:

- **wireshark** >= 2.40 [2017-07-19]
- **iproute2** >= 4.15 [2018-01-28]
  - **ss**
- **tcpdump**
  - merged in master [2019-04-16]
- **nmap** >= 7.80 [2019-08-10]
  - **ncat**
- **nbd**
  - **nbdkit** >= 1.15.5 [2019-10-19]
  - **libnbd** >= 1.1.6 [2019-10-19]
- **iperf-vsock**
  - iperf3 fork
  - https://github.com/stefano-garzarella/iperf-vsock

Languages:

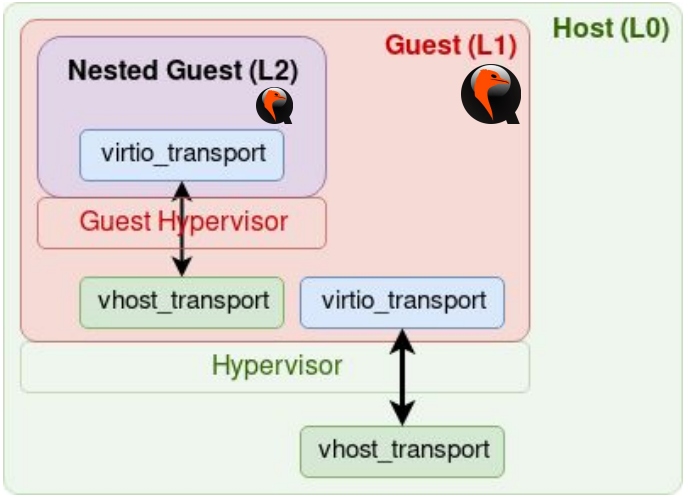
- C
  - **glibc** >= 2.18 [2013-08-10]
- Python
  - **python** >= 3.7 alpha 1 [2017-09-19]
- Golang
  - **https://github.com/mdlayher/vsock**
- Rust
  - **libc** crate >= 0.2.59 [2019-07-08]
    - struct sockaddr_vm
    - VMADDR_* macros
  - **nix** crate >= 0.15.0 [2019-08-10]
    - VSOCK supported in the socket API (nix::sys::socket)

vsock: next steps

- **multi-transports [WiP]**
  - goal: support nested VMs
    - allow multiple VSOCK transports loaded at runtime
    - RFC sent, more details in the next slide
- **network namespace**
  - goal: create independent AF_VSOCK addressing domains
    - partition VMs between domains
    - isolate host applications from guest applications bound to the same port with VMADDR_CID_ANY
- **virtio-net as a transport for the virtio-vsock**
  - goal: avoid to reimplement features already done in the virtio-net
    - mergeable buffers
    - page allocation (instead of kmalloc)
    - small packets handling

vsock-core: multi-transports support (1/2)

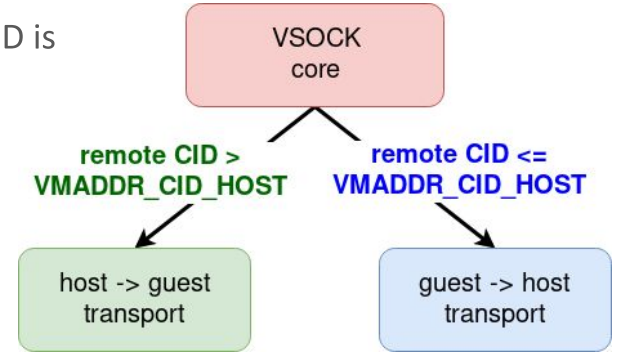
- Current implementation support only one transport loaded at runtime
  - virtio_transport or vhost_transport
  - impossible to use virtio-vsock in a nested VMs environment
- Works done:
  - vsock core module can be loaded regardless of the transports
  - vsock core handles two types of transport
    - 'host -> guest' transport
    - 'guest -> host' transport



[RFC PATCH 00/13] vsock: add multi-transports support
https://patchwork.ozlabs.org/cover/1168442/

vsock-core: multi-transports support (2/2)

- Works done (cont.)
  - each STREAM socket is assigned to a transport when the remote CID is known
    - during a connect()
    - connection request received on a listener socket
  - listener sockets are not bound to any transports
    - created also if the transports are not loaded
    - VMADDR_CID_ANY to listen on all transports.



- Works to do
  - DGRAM sockets are handled as before
  - MODULE_ALIAS_NETPROTO(PF_VSOCK) in the VSOCK core (af_vsock.ko)

[RFC PATCH 00/13] vsock: add multi-transports support
https://patchwork.ozlabs.org/cover/1168442/

Q&A

Andra Paraschiv <andraprs@amazon.com>

Stefano Garzarella <sgarzare@redhat.com>
Blog: https://stefano-garzarella.github.io/
IRC: sgarzare on #qemu irc.oftc.net

Thank you!

References

[1] http://docs.oasis-open.org/virtio/virtio/v1.1/virtio-v1.1.pdf

[2] http://man7.org/linux/man-pages/man7/vsock.7.html

[3] https://www.kernel.org/

[4] https://www.qemu.org/contribute/

[5] https://wiki.qemu.org/Features/VirtioVsock

[6] https://wiki.qemu.org/Features/VirtioVhostUser

References (2)

[7] https://github.com/firecracker-microvm/firecracker

[8] http://man7.org/linux/man-pages/man8/ss.8.html

[9] https://github.com/stefano-garzarella/iperf-vsock

[10] https://nmap.org/book/ncat-man-vsock.html

[11] https://www.tcpdump.org/linktypes/LINKTYPE_VSOCK.html

[12] https://www.wireshark.org/docs/dfref/v/vsock.html

[13] https://rwmj.wordpress.com/2019/10/21/nbd-over-af_vsock/

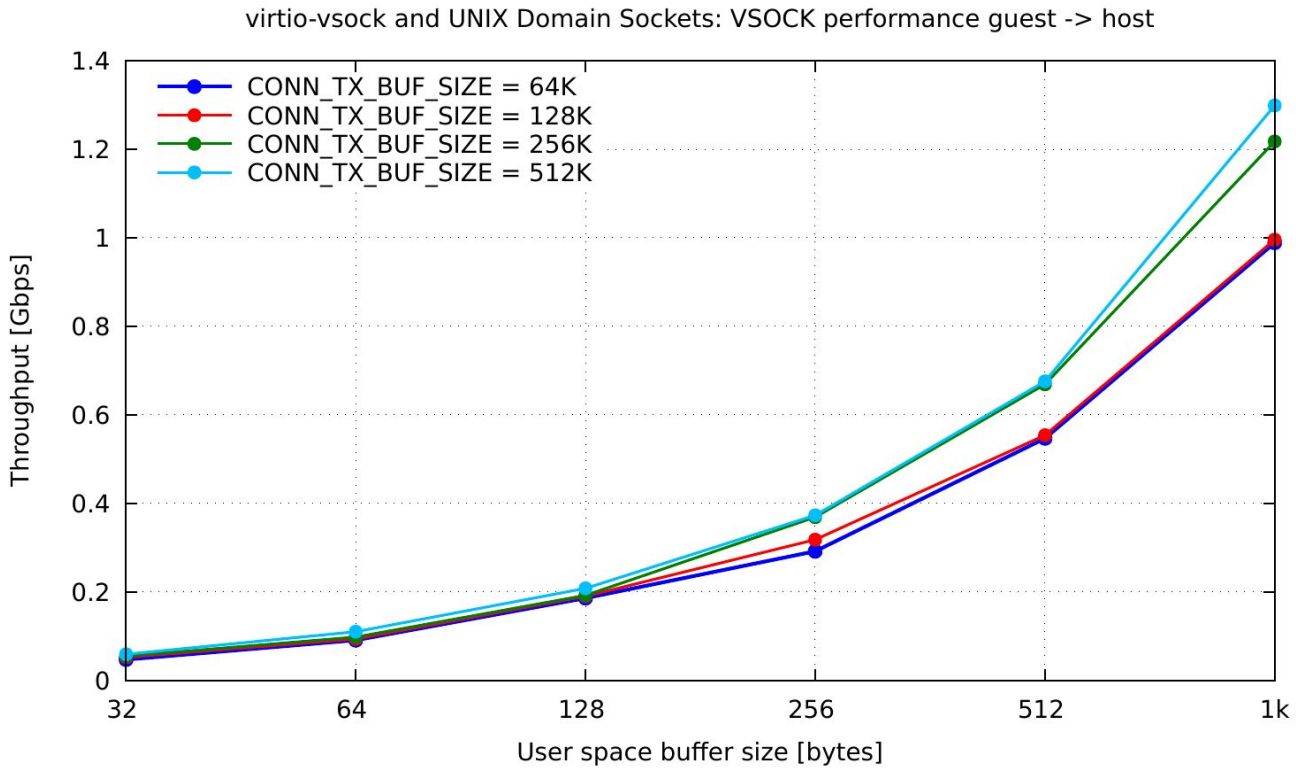
Backup Slides

Performance Metrics - virtio-vsock and UNIX Sockets

guest -> host throughput [Gbps]

user space buffer size	CONN_TX_BUF_SIZE = 64K	CONN_TX_BUF_SIZE = 128K	CONN_TX_BUF_SIZE = 256K	CONN_TX_BUF_SIZE = 512K
32	0.047	0.052	0.055	0.059
64	0.091	0.094	0.098	0.110
128	0.186	0.190	0.192	0.208
256	0.292	0.318	0.369	0.373
512	0.547	0.554	0.669	0.676
1K	0.989	0.996	1.218	1.299
2K	2.311	2.813	2.852	2.974
4K	3.546	4.301	5.057	5.325
8K	3.957	5.393	5.560	5.654
16K	4.206	6.167	7.435	7.773
32K	4.350	7.067	7.528	8.622
64K	4.660	7.347	8.966	10.133
128K	4.763	7.443	8.933	10.104
256K	4.893	7.534	8.892	9.756
512K	4.921	7.281	8.904	9.187

Performance Metrics - virtio-vsock and UNIX Sockets (2)

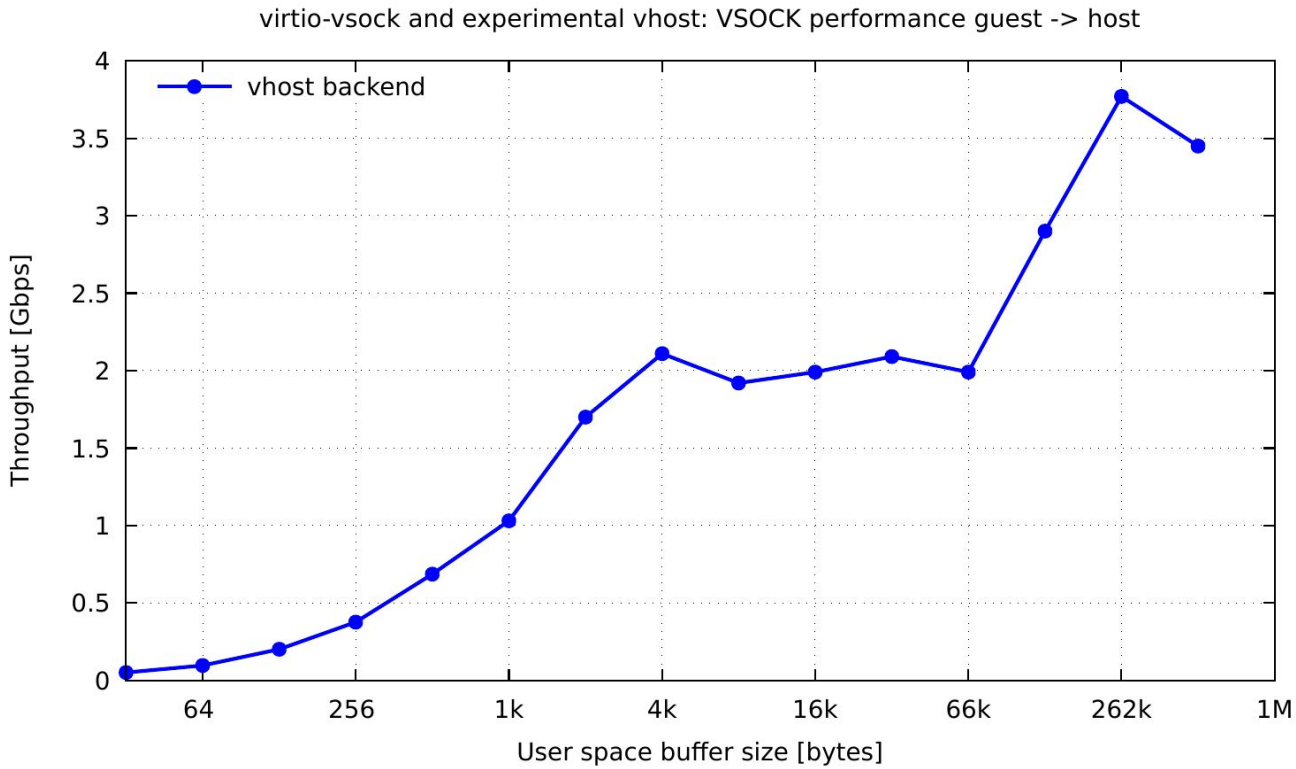


Performance Metrics - virtio-vsock and experim. vhost

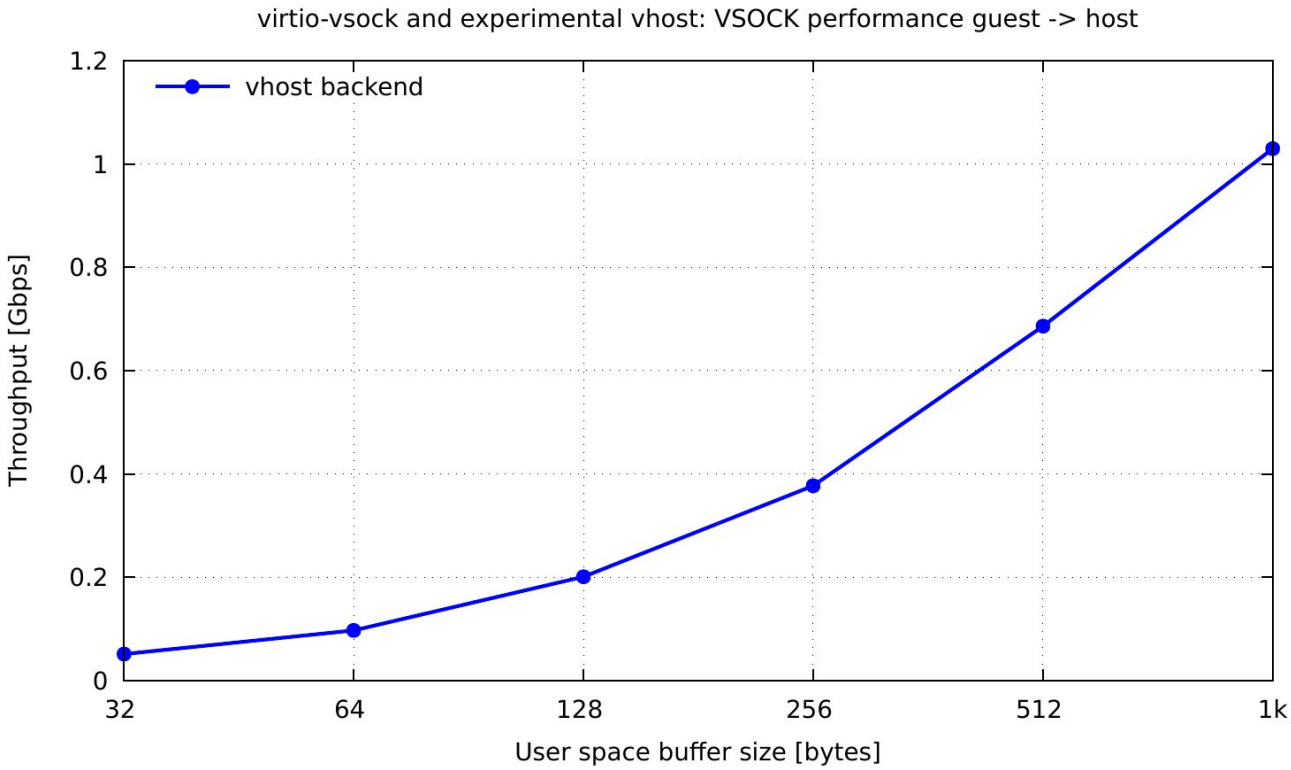
guest -> host throughput [Gbps]

user space buffer size	Kernel 5.3 (vhost support)
32	0.051
64	0.097
128	0.201
256	0.377
512	0.686
1K	1.030
2K	1.700
4K	2.110
8K	1.920
16K	1.990
32K	2.090
64K	1.990
128K	2.900
256K	3.770
512K	3.450

Performance Metrics - virtio-vsock and experim. vhost (2)

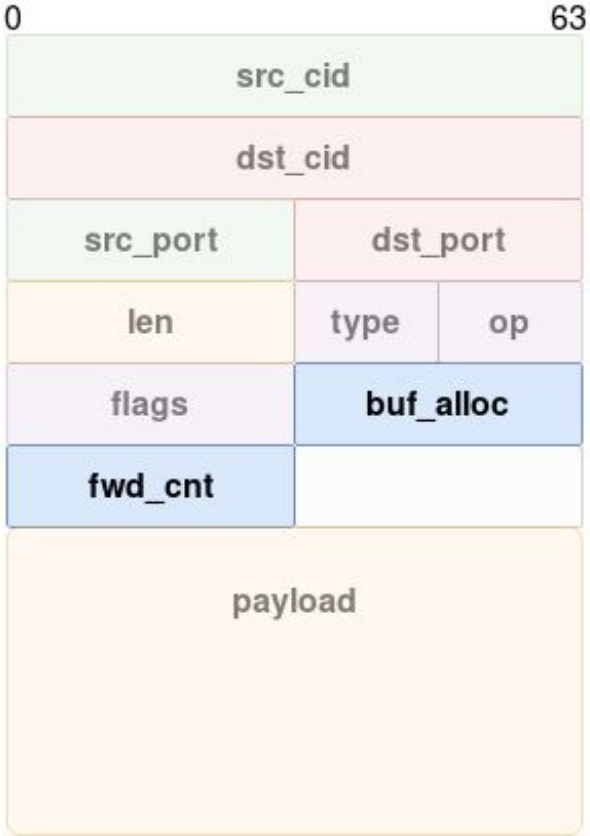


Performance Metrics - virtio-vsock and experim. vhost (3)



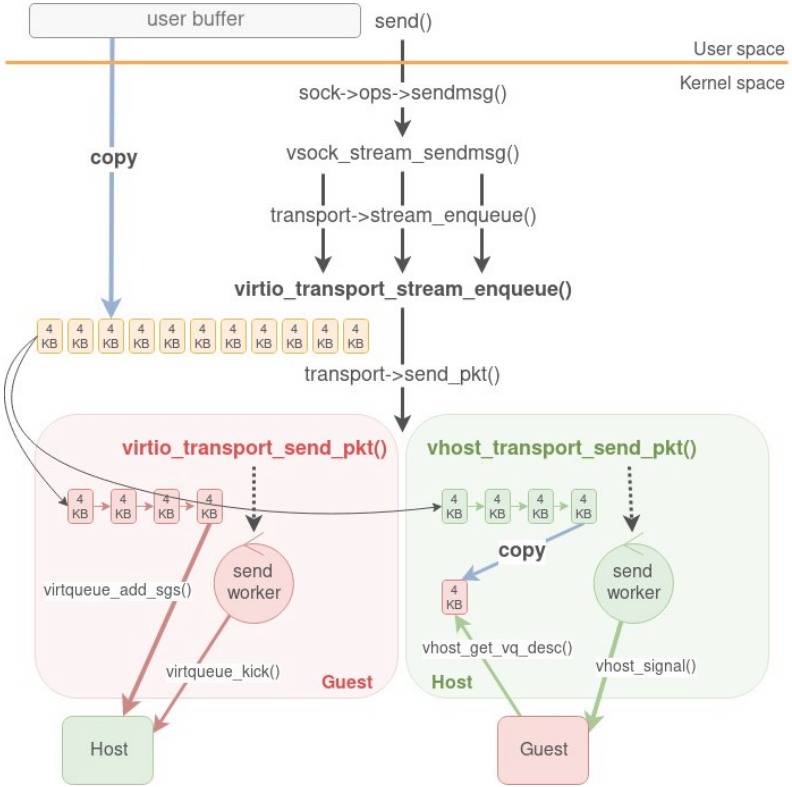
virtio-vsock: credit mechanism

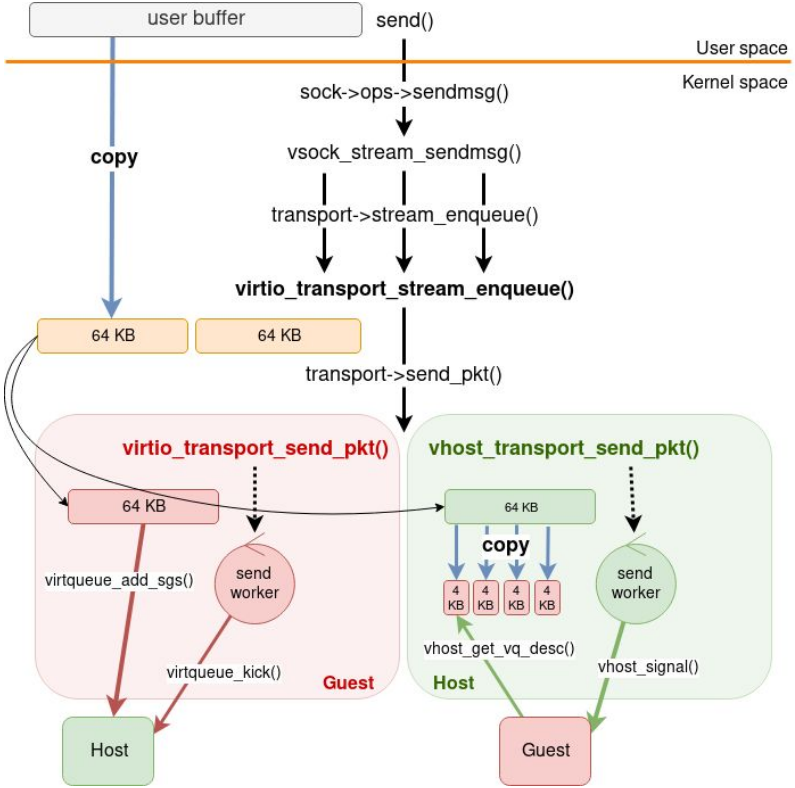
- **`'buf_alloc'`** and **`'fwd_cnt'`** are sent to the other peer:
  - in all packets header
  - with an explicit message
    - `op = VIRTIO_VSOCK_OP_CREDIT_UPDATE`



Red Hat

virtio-vsock: performance improvements





Red Hat

virtio-vsock: performance metrics (1/2)

host -> guest throughput [Gbps]

pkt_size	before opt.	patch 1	patches 2+3	patches 4+5
32	0.032	0.030	0.048	0.051
64	0.061	0.059	0.108	0.117
128	0.122	0.112	0.227	0.234
256	0.244	0.241	0.418	0.415
512	0.459	0.466	0.847	0.865
1K	0.927	0.919	1.657	1.641
2K	1.884	1.813	3.262	3.269
4K	3.378	3.326	6.044	6.195
8K	5.637	5.676	10.141	11.287
16K	8.250	8.402	15.976	16.736
32K	13.327	13.204	19.013	20.515
64K	21.241	21.341	20.973	21.879
128K	21.851	22.354	21.816	23.203
256K	21.408	21.693	21.846	24.088
512K	21.600	21.899	21.921	24.106

host -> guest efficiency
[Mbps / (%CPU_Host + %CPU_Guest)]

pkt_size	before opt.	patch 1	patches 2+3	patches 4+5
32	0.35	0.45	0.79	1.02
64	0.56	0.80	1.41	1.54
128	1.11	1.52	3.03	3.12
256	2.20	2.16	5.44	5.58
512	4.17	4.18	10.96	11.46
1K	8.30	8.26	20.99	20.89
2K	16.82	16.31	39.76	39.73
4K	30.89	30.79	74.07	75.73
8K	53.74	54.49	124.24	148.91
16K	80.68	83.63	200.21	232.79
32K	132.27	132.52	260.81	357.07
64K	229.82	230.40	300.19	444.18
128K	332.60	329.78	331.51	492.28
256K	331.06	337.22	339.59	511.59
512K	335.58	328.50	331.56	504.56

virtio-vsock: performance metrics (2/2)

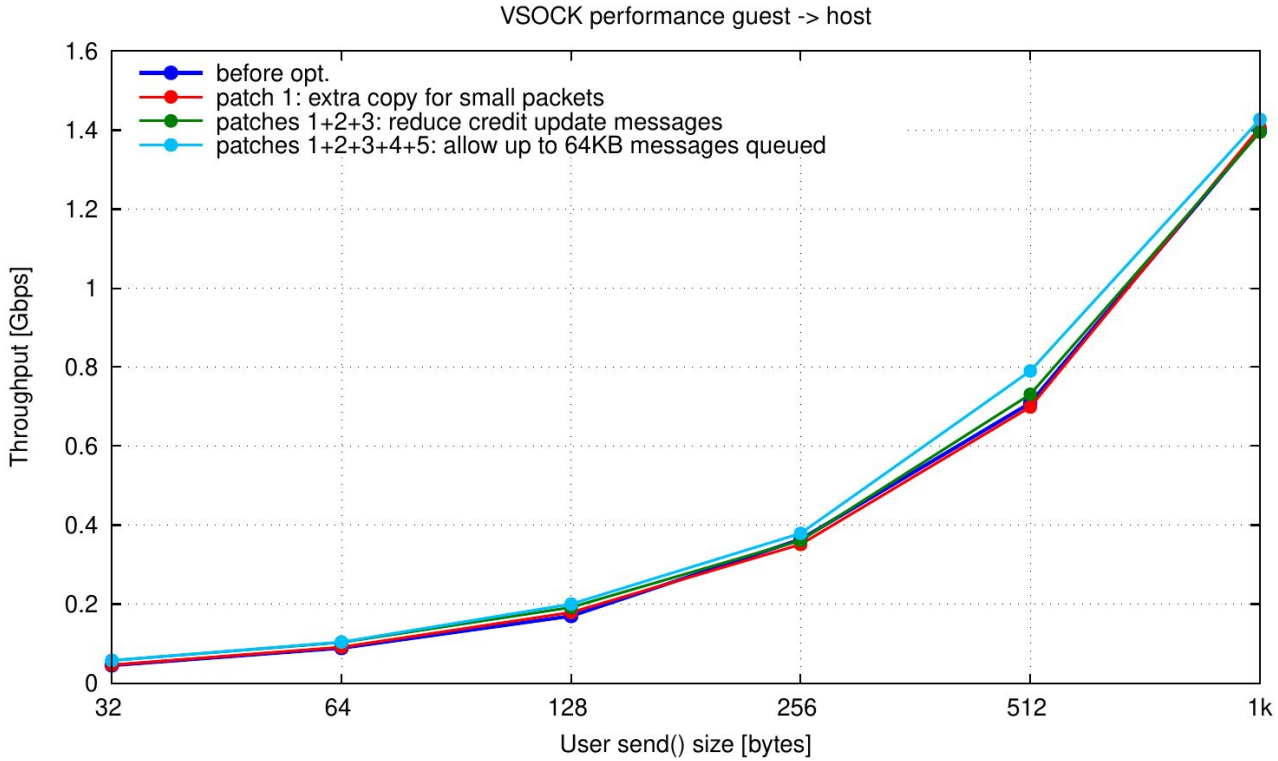
guest -> host throughput [Gbps]

pkt_size	before opt.	patch 1	patches 2+3	patches 4+5
32	0.045	0.046	0.057	0.057
64	0.089	0.091	0.103	0.104
128	0.170	0.179	0.192	0.200
256	0.364	0.351	0.361	0.379
512	0.709	0.699	0.731	0.790
1K	1.399	1.407	1.395	1.427
2K	2.670	2.684	2.745	2.835
4K	5.171	5.199	5.305	5.451
8K	8.442	8.500	10.083	9.941
16K	12.305	12.259	13.519	15.385
32K	11.418	11.150	11.988	24.680
64K	10.778	10.659	11.589	35.273
128K	10.421	10.339	10.939	40.338
256K	10.300	9.719	10.508	36.562
512K	9.833	9.808	10.612	35.979

guest -> host efficiency
[Mbps / (%CPU_Host + %CPU_Guest)]

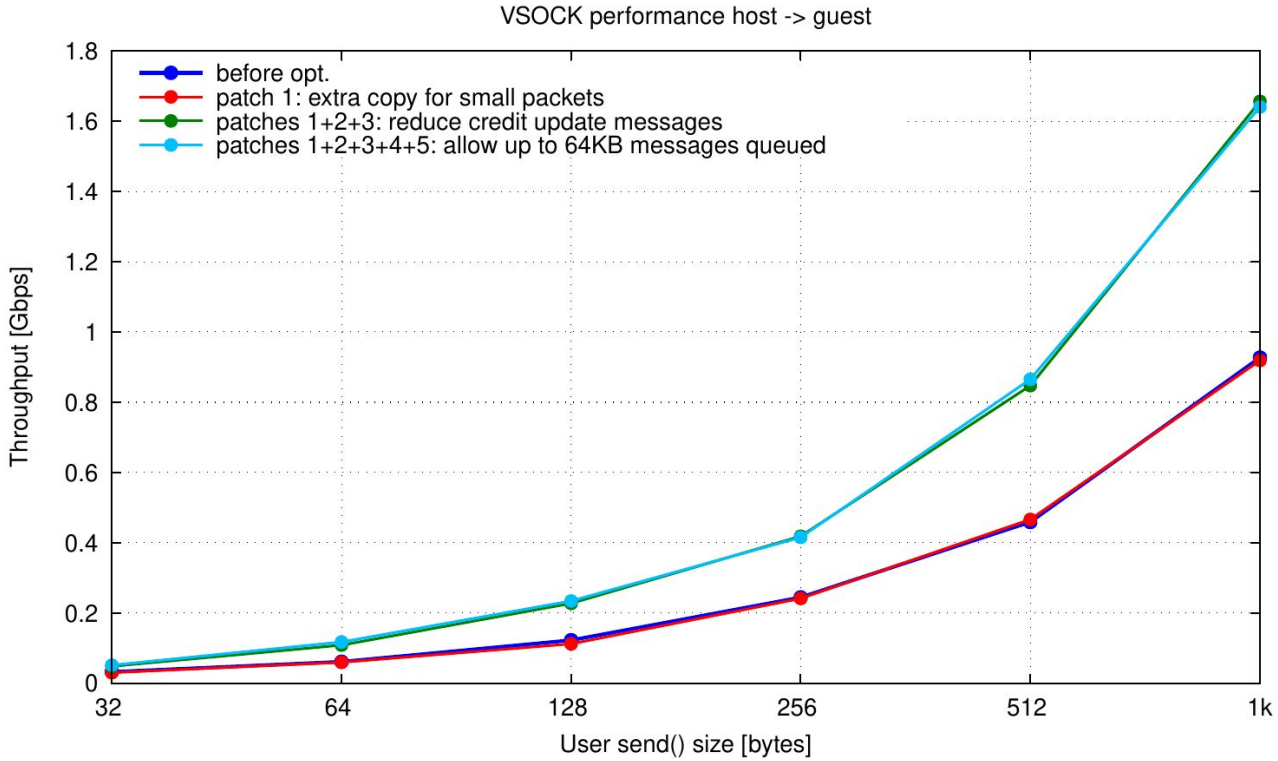
pkt_size	before opt.	patch 1	patches 2+3	patches 4+5
32	0.43	0.43	0.53	0.56
64	0.85	0.86	1.04	1.10
128	1.63	1.71	2.07	2.13
256	3.48	3.35	4.02	4.22
512	6.80	6.67	7.97	8.63
1K	13.32	13.31	15.72	15.94
2K	25.79	25.92	30.84	30.98
4K	50.37	50.48	58.79	59.69
8K	95.90	96.15	107.04	110.33
16K	145.80	145.43	143.97	174.70
32K	147.06	144.74	146.02	282.48
64K	145.25	143.99	141.62	406.40
128K	149.34	146.96	147.49	489.34
256K	156.35	149.81	152.21	536.37
512K	151.65	150.74	151.52	519.93

virtio-vsock: performance improvements (1/6)



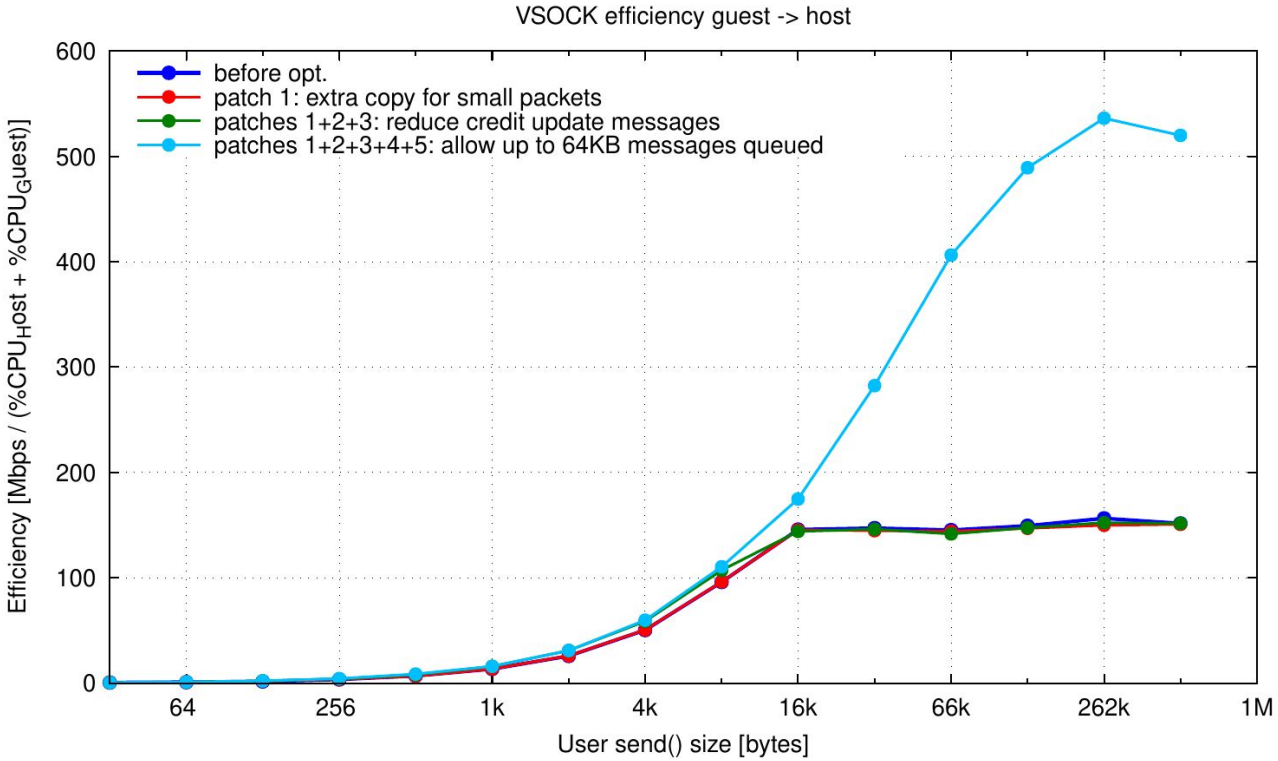
Red Hat

virtio-vsock: performance improvements (2/6)



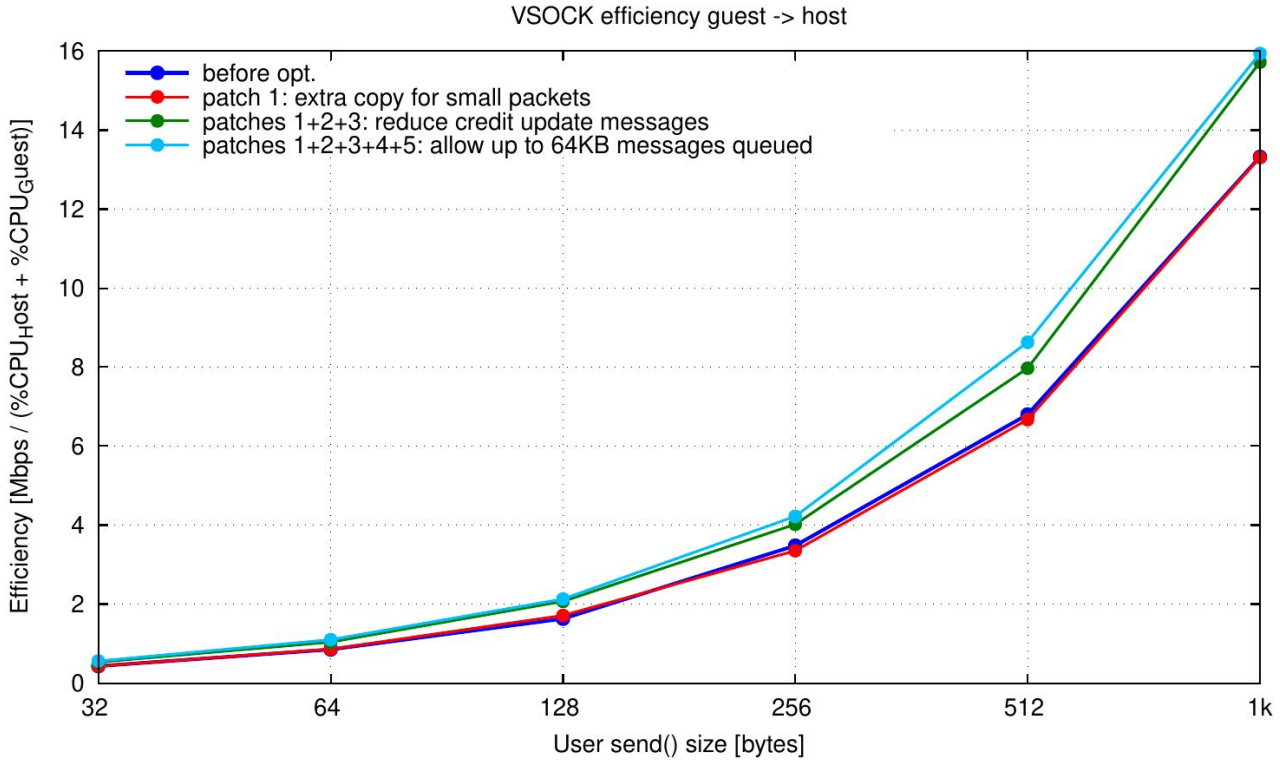
Red Hat

virtio-vsock: performance improvements (3/6)



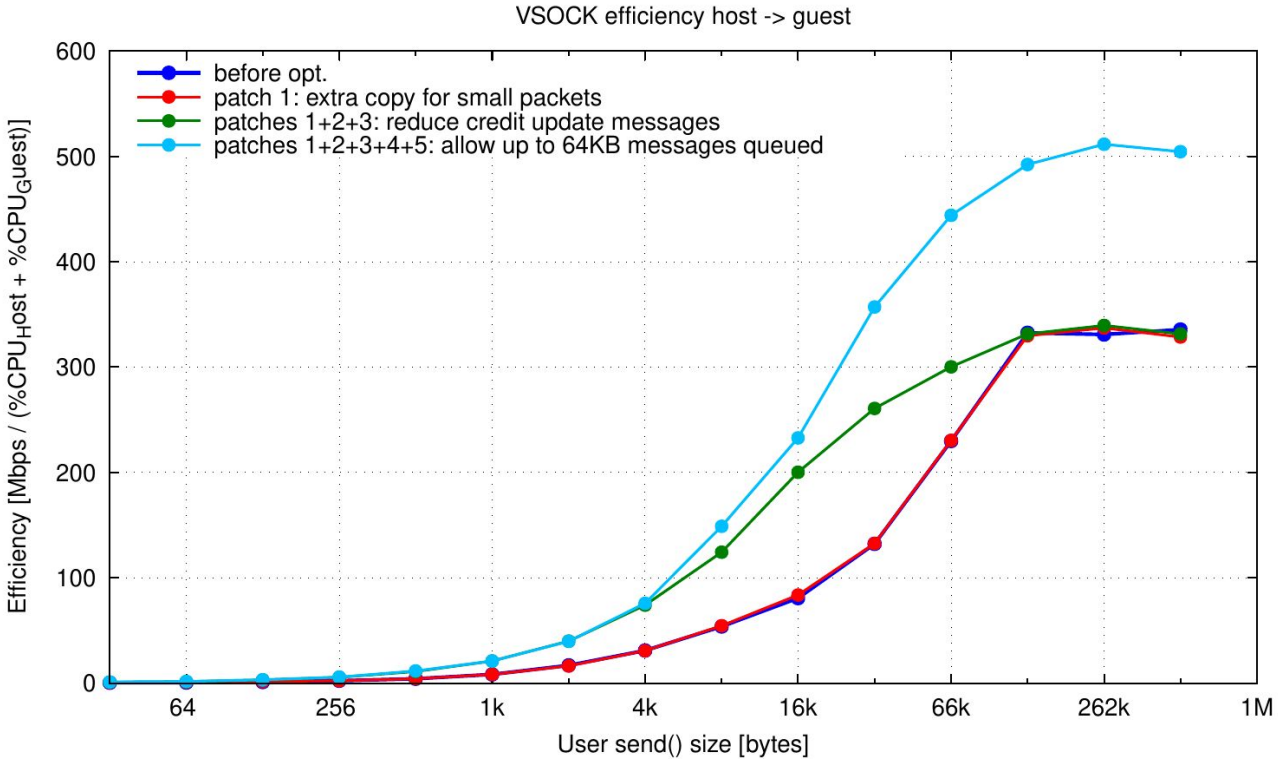
Red Hat

virtio-vsock: performance improvements (4/6)



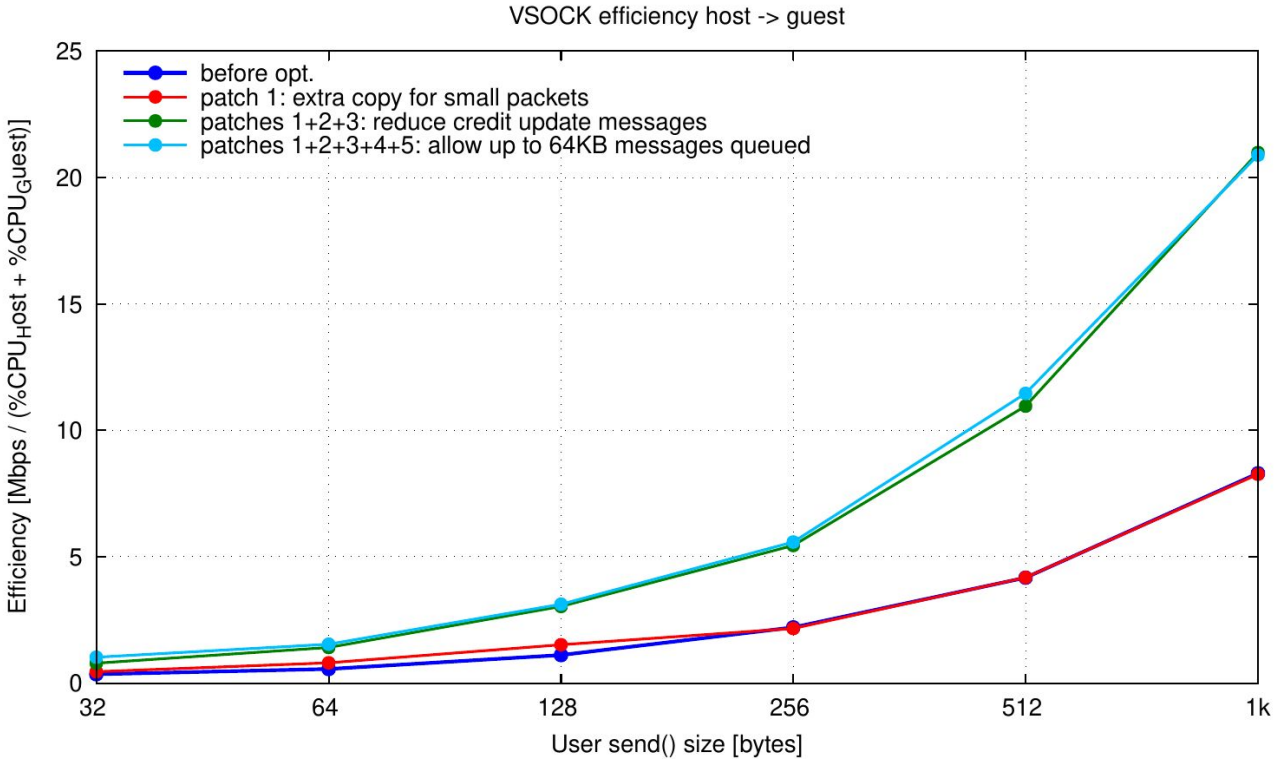
Red Hat

virtio-vsock: performance improvements (5/6)



Red Hat

virtio-vsock: performance improvements (6/6)



Red Hat