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
Agenda

State of the Art
Use Cases
vhost Backend
Firecracker and virtio-vsock
Firecracker - vsock and UNIX Domain Sockets
QEMU and vsock
vssock Linux Drivers - Changes in the Last Year
Tools Supported
Next Steps
State of the Art

Guest

- Guest App
- virtio-vsock (PCI / MMIO)
- CID, port socket bind connect ...

Host

- Host App
- vhost_vsock
Use Cases

- **Guest Agents**
- **Network Applications (SOCK_STREAM)**
- **Hypervisor Services**
vhost Backend

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

- 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-vssock device model over MMIO
- UNIX Domain Sockets (UDS) on host
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)

Guest OS

vsock
(listen <port>)

Firecracker

vsock endpoint
(VMADDR_CID_HOST)

UNIX socket
(listen <uds>)

1) CONNECT
<vsock_port>

2) Send / Receive
data

Host

UNIX socket
(connect <uds>)
Firecracker - vsock and UNIX Domain Sockets (3)

Guest OS
- `vsock (connect CID 2 <port>)`

Firecracker
- `vsock endpoint (VMADDR_CID_HOST)`
- `UNIX socket (connect <uds_port>)`

Host
- `UNIX socket (listen <uds_port>)`

Send / Receive data
Firecracker - vssock and UNIX Domain Sockets (4)

virtio-vssock and UNIX Domain Sockets: VSOCK performance guest -> host

Throughput [Gbps]

User space buffer size [bytes]

© 2019, Amazon Web Services, Inc. or its Affiliates.
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)
vsock Linux drivers: commits per year
### vsock Linux drivers: changes in the last year (1/2)

45 changesets from 16 developers

#### Top changeset contributors by employer

<table>
<thead>
<tr>
<th>Employer</th>
<th>Changesets</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat</td>
<td>21</td>
<td>46.7%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>8</td>
<td>17.8%</td>
</tr>
<tr>
<td>Linutronix</td>
<td>7</td>
<td>15.6%</td>
</tr>
<tr>
<td><a href="mailto:ytht.net@gmail.com">ytht.net@gmail.com</a></td>
<td>2</td>
<td>4.4%</td>
</tr>
<tr>
<td>Intel</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td>VMware</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td>Google</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td>Bitdefender</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td>Alibaba</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td>The Chromium Projects</td>
<td>1</td>
<td>2.2%</td>
</tr>
<tr>
<td><a href="mailto:arnd@arndb.de">arnd@arndb.de</a></td>
<td>1</td>
<td>2.2%</td>
</tr>
</tbody>
</table>

#### Top lines changed by employer

<table>
<thead>
<tr>
<th>Employer</th>
<th>Lines Added</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat</td>
<td>370</td>
<td>44.6%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>234</td>
<td>28.2%</td>
</tr>
<tr>
<td>Linutronix</td>
<td>129</td>
<td>15.5%</td>
</tr>
<tr>
<td>VMware</td>
<td>50</td>
<td>6.0%</td>
</tr>
<tr>
<td>Bitdefender</td>
<td>15</td>
<td>1.8%</td>
</tr>
<tr>
<td>Intel</td>
<td>12</td>
<td>1.4%</td>
</tr>
<tr>
<td><a href="mailto:ytht.net@gmail.com">ytht.net@gmail.com</a></td>
<td>7</td>
<td>0.8%</td>
</tr>
<tr>
<td>Google</td>
<td>7</td>
<td>0.8%</td>
</tr>
<tr>
<td>The Chromium Projects</td>
<td>3</td>
<td>0.4%</td>
</tr>
<tr>
<td><a href="mailto:arnd@arndb.de">arnd@arndb.de</a></td>
<td>2</td>
<td>0.2%</td>
</tr>
<tr>
<td>Alibaba</td>
<td>1</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

A total of 674 lines added, 426 removed (delta 245)
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
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
virdio-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
virtio-vsock: send() path (1/2)

- **vssock_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)
**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
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)
virtio-vsock: performance improvements (2/3)
virtio-vsock: performance improvements (3/3)
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


References (2)


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

## Performance Metrics - virtio-vsock and UNIX Sockets

**guest -> host throughput [Gbps]**

<table>
<thead>
<tr>
<th>user space buffer size</th>
<th>CONN_TX_BUF_SIZE = 64K</th>
<th>CONN_TX_BUF_SIZE = 128K</th>
<th>CONN_TX_BUF_SIZE = 256K</th>
<th>CONN_TX_BUF_SIZE = 512K</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.047</td>
<td>0.052</td>
<td>0.055</td>
<td>0.059</td>
</tr>
<tr>
<td>64</td>
<td>0.091</td>
<td>0.094</td>
<td>0.098</td>
<td>0.110</td>
</tr>
<tr>
<td>128</td>
<td>0.186</td>
<td>0.190</td>
<td>0.192</td>
<td>0.208</td>
</tr>
<tr>
<td>256</td>
<td>0.292</td>
<td>0.318</td>
<td>0.369</td>
<td>0.373</td>
</tr>
<tr>
<td>512</td>
<td>0.547</td>
<td>0.554</td>
<td>0.669</td>
<td>0.676</td>
</tr>
<tr>
<td>1K</td>
<td>0.989</td>
<td>0.996</td>
<td>1.218</td>
<td>1.299</td>
</tr>
<tr>
<td>2K</td>
<td>2.311</td>
<td>2.813</td>
<td>2.852</td>
<td>2.974</td>
</tr>
<tr>
<td>4K</td>
<td>3.546</td>
<td>4.301</td>
<td>5.057</td>
<td>5.325</td>
</tr>
<tr>
<td>8K</td>
<td>3.957</td>
<td>5.393</td>
<td>5.560</td>
<td>5.654</td>
</tr>
<tr>
<td>32K</td>
<td>4.350</td>
<td>7.067</td>
<td>7.528</td>
<td>8.622</td>
</tr>
<tr>
<td>64K</td>
<td>4.660</td>
<td>7.347</td>
<td>8.966</td>
<td>10.133</td>
</tr>
<tr>
<td>128K</td>
<td>4.763</td>
<td>7.443</td>
<td>8.933</td>
<td>10.104</td>
</tr>
<tr>
<td>256K</td>
<td>4.893</td>
<td>7.534</td>
<td>8.892</td>
<td>9.756</td>
</tr>
<tr>
<td>512K</td>
<td>4.921</td>
<td>7.281</td>
<td>8.904</td>
<td>9.187</td>
</tr>
</tbody>
</table>
Performance Metrics - virtio-vsock and UNIX Sockets (2)

virtio-vsock and UNIX Domain Sockets: VSOCK performance guest -> host

Throughput [Gbps]

User space buffer size [bytes]
## Performance Metrics - virtio-vsock and experim. vhost

<table>
<thead>
<tr>
<th>user space buffer size</th>
<th>Kernel 5.3 (vhost support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.051</td>
</tr>
<tr>
<td>64</td>
<td>0.097</td>
</tr>
<tr>
<td>128</td>
<td>0.201</td>
</tr>
<tr>
<td>256</td>
<td>0.377</td>
</tr>
<tr>
<td>512</td>
<td>0.686</td>
</tr>
<tr>
<td>1K</td>
<td>1.030</td>
</tr>
<tr>
<td>2K</td>
<td>1.700</td>
</tr>
<tr>
<td>4K</td>
<td>2.110</td>
</tr>
<tr>
<td>8K</td>
<td>1.920</td>
</tr>
<tr>
<td>16K</td>
<td>1.990</td>
</tr>
<tr>
<td>32K</td>
<td>2.090</td>
</tr>
<tr>
<td>64K</td>
<td>1.990</td>
</tr>
<tr>
<td>128K</td>
<td>2.900</td>
</tr>
<tr>
<td>256K</td>
<td>3.770</td>
</tr>
<tr>
<td>512K</td>
<td>3.450</td>
</tr>
</tbody>
</table>
Performance Metrics - virtio-vsock and experim. vhost (2)

virtio-vsock and experimental vhost: VSOCK performance guest -> host

Throughput [Gbps]

User space buffer size [bytes]

© 2019, Amazon Web Services, Inc. or its Affiliates.
Performance Metrics - virtio-vsock and experim. vhost (3)

![Graph showing throughput vs user space buffer size](image)

virtio-vsock and experimental vhost: VSOCK performance guest -> host

Throughput [Gbps]

User space buffer size [bytes]
virtio-vsock: credit mechanism

- 'buf_alloc' and 'fwd_cnt' are sent to the other peer:
  - in all packets header
  - with an explicit message
    - \( \text{op} = \text{VIRTIO_VSOCK_OP_CREDIT_UPDATE} \)
virtio-vsock: performance improvements
### virtio-vsock: performance metrics (1/2)

**host -> guest throughput [Gbps]**

<table>
<thead>
<tr>
<th>pkt_size</th>
<th>before opt.</th>
<th>patch 1</th>
<th>patches 2+3</th>
<th>patches 4+5</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.032</td>
<td>0.030</td>
<td>0.048</td>
<td>0.051</td>
</tr>
<tr>
<td>64</td>
<td>0.061</td>
<td>0.059</td>
<td>0.108</td>
<td>0.117</td>
</tr>
<tr>
<td>128</td>
<td>0.122</td>
<td>0.112</td>
<td>0.227</td>
<td>0.234</td>
</tr>
<tr>
<td>256</td>
<td>0.244</td>
<td>0.241</td>
<td>0.418</td>
<td>0.415</td>
</tr>
<tr>
<td>512</td>
<td>0.459</td>
<td>0.466</td>
<td>0.847</td>
<td>0.865</td>
</tr>
<tr>
<td>1K</td>
<td>0.927</td>
<td>0.919</td>
<td>1.657</td>
<td>1.641</td>
</tr>
<tr>
<td>2K</td>
<td>1.884</td>
<td>1.813</td>
<td>3.262</td>
<td>3.269</td>
</tr>
<tr>
<td>8K</td>
<td>5.637</td>
<td>5.676</td>
<td>10.141</td>
<td>11.287</td>
</tr>
<tr>
<td>16K</td>
<td>8.250</td>
<td>8.402</td>
<td>15.976</td>
<td>16.736</td>
</tr>
<tr>
<td>32K</td>
<td>13.327</td>
<td>13.204</td>
<td>19.013</td>
<td>20.515</td>
</tr>
</tbody>
</table>

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

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

© 2019, Red Hat, Inc.
## virtio-vsock: performance metrics (2/2)

### guest -> host throughput [Gbps]

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

### guest -> host efficiency

\[
\text{Mbps} / (\%CPU_{Host} + \%CPU_{Guest})
\]

<table>
<thead>
<tr>
<th>pkt_size</th>
<th>before opt.</th>
<th>patch 1</th>
<th>patches 2+3</th>
<th>patches 4+5</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.43</td>
<td>0.43</td>
<td>0.53</td>
<td>0.56</td>
</tr>
<tr>
<td>64</td>
<td>0.85</td>
<td>0.86</td>
<td>1.04</td>
<td>1.10</td>
</tr>
<tr>
<td>128</td>
<td>1.63</td>
<td>1.71</td>
<td>2.07</td>
<td>2.13</td>
</tr>
<tr>
<td>256</td>
<td>3.48</td>
<td>3.35</td>
<td>4.02</td>
<td>4.22</td>
</tr>
<tr>
<td>512</td>
<td>6.80</td>
<td>6.67</td>
<td>7.97</td>
<td>8.63</td>
</tr>
<tr>
<td>1K</td>
<td>13.32</td>
<td>13.31</td>
<td>15.72</td>
<td>15.94</td>
</tr>
<tr>
<td>2K</td>
<td>25.79</td>
<td>25.92</td>
<td>30.84</td>
<td>30.98</td>
</tr>
<tr>
<td>4K</td>
<td>50.37</td>
<td>50.48</td>
<td>58.79</td>
<td>59.69</td>
</tr>
<tr>
<td>8K</td>
<td>95.90</td>
<td>96.15</td>
<td>107.04</td>
<td>110.33</td>
</tr>
<tr>
<td>16K</td>
<td>145.80</td>
<td>145.43</td>
<td>143.97</td>
<td>174.70</td>
</tr>
<tr>
<td>32K</td>
<td>147.06</td>
<td>144.74</td>
<td>146.02</td>
<td>282.48</td>
</tr>
<tr>
<td>64K</td>
<td>145.25</td>
<td>143.99</td>
<td>141.62</td>
<td>406.40</td>
</tr>
<tr>
<td>128K</td>
<td>149.34</td>
<td>146.96</td>
<td>147.49</td>
<td>489.34</td>
</tr>
<tr>
<td>256K</td>
<td>156.35</td>
<td>149.81</td>
<td>152.21</td>
<td>536.37</td>
</tr>
<tr>
<td>512K</td>
<td>151.65</td>
<td>150.74</td>
<td>151.52</td>
<td>519.93</td>
</tr>
</tbody>
</table>
virtio-vsock: performance improvements (1/6)
virtio-vsock: performance improvements (2/6)

VSOCK performance host -> guest

- before opt.
- patch 1: extra copy for small packets
- patches 1+2+3: reduce credit update messages
- patches 1+2+3+4+5: allow up to 64KB messages queued
virtio-vsock: performance improvements (3/6)
virtio-vsock: performance improvements (4/6)
virtio-vsock: performance improvements (5/6)
virtio-vsock: performance improvements (6/6)