Persistence in Networking: Redesigning Stack, API and Networks

Michio Honda (NEC)
Lars Eggert (NetApp)
Douglas Santry (NetApp)
Munich Internet Research Retreat
May 22th 2017

Published at HotNets 2016
https://goo.gl/Ssreei
Motivation

- Non-Volatile Main Memories (NVMMs)
  - Persistent
  - Byte-addressable
  - Low latency
    - 10s-1000s of ns
- Shift from block- to byte-granularity persistency
  - OS abstractions
    - Direct access to mmap()-ed files
  - Data structures
    - Filesystems and databases

What are implications for networking?
Case Study: Careful Data Transfer

• Server persists client’s request prior to acknowledgment
• e.g., 1KB commit:
Case Study: Careful Data Transfer

- Server persists client’s request prior to acknowledgment
- e.g., 1KB commit:

2030 us
- Networking (w/o (4)) takes 40 us
Case Study: Careful Data Transfer

- Server persists client’s request prior to acknowledgment
- e.g., 1KB commit:

2000 42 us

- Networking takes 40 us
- This 2 us is not small
Case Study: Careful Data Transfer

- Parallel requests are serialized on each core

33% throughput decrease, 50% latency increase
Data Copies Matter

- **Cache Misses**
  - Persisting data (e.g., to a log) always happens to a different destination

<table>
<thead>
<tr>
<th></th>
<th>Overall cache misses</th>
<th>Largest Contributor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking only</td>
<td>0.0004 %</td>
<td>net_rx_action() (84%)</td>
</tr>
<tr>
<td>Networking + NVMM</td>
<td>4.4121 %</td>
<td>memcopy() (98%)</td>
</tr>
<tr>
<td>(read() + memcpy() + msync())</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We must avoid data copy!
Packet Store (PASTE) Overview

- Static packet buffers on a named NVMM region
  - DMA to NVMM
- Zero-copy APIs
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User

Kernel

TCP/IP input and output

NIC ring

Application

mmap() function

netmap API

metadata_header

/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries

<table>
<thead>
<tr>
<th>buf_idx</th>
<th>off</th>
<th>len</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>1135</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>932</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1024</td>
</tr>
</tbody>
</table>

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User
mmap() 
netmap API

Kernel
TCP/IP input and output

NIC ring

metadata header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
buf_idx off len
0 100 1135
1 100 932
3 100 1024

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

packet buffers (static)

unread
read or written

Application
(2) Write metadata entry
(3) Flush (buffer and metadata)
Fast Persistent Write with PASTE

Unread
Read or written

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User

Kernel

TCP/IP input and output

NIC ring

/mnt/nvmm/pktbufs

packet buffers (static)

/mnt/nvmm/myapp_metadata

/meta_header

/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries

<table>
<thead>
<tr>
<th>buf_idx</th>
<th>off</th>
<th>len</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>1135</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>932</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1024</td>
</tr>
</tbody>
</table>

/mnt/nvmm/pktbufs

Application

(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User

Kernel

TCP/IP input and output

NIC ring

metadata header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
buf_idx off len
0 100 1135
1 100 932
3 100 1024

packet buffers (static)

/mnt/nvmm/myapp_metadata

/mnt/nvmm/pktbufs

Application
(2) Write metadata entry
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User

Kernel

TCP/IP input and output

NIC ring

DMA is performed to L3 cache (DDIO)
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

mmap() → netmap API

User

Kernel

TCP/IP input and output

NIC ring

/mnt/nvmm/pktbufs
buf_ofs: 123

metadata header
/mnt/nvmm/pktbufs

metadata entries
buf_idx  off  len
0  100  1135
1  100  932
3  100  1024

/mnt/nvmm/myapp_metadata

packet buffers (static)

DMA is performed to L3 cache (DDIO)
Fast Persistent Write with PASTE

Application
1) Read data (zero copy)
2) Write metadata entry
3) Flush (buffer and metadata)

User
netmap API

Kernel
TCP/IP input and output

NIC ring

DMA is performed to L3 cache (DDIO)
Fast Persistent Write with PASTE

Unread
Read or written
Flushed

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

metadata header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
buf_idx off len
0 100 1135
1 100 932
3 100 1024

/mnt/nvmm/myapp_metadata /mnt/nvmm/pktbufs

DMA is performed to L3 cache (DDIO)

mmap() netmap API

TCP/IP input and output
NIC ring

Kernel
User
Fast Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

metadata_header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
<table>
<thead>
<tr>
<th>buf_idx</th>
<th>off</th>
<th>len</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>1135</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>932</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1024</td>
</tr>
</tbody>
</table>

/mnt/nvmm/myapp_metadata

pktbufs (static)

DMA is performed to L3 cache (DDIO)

Unread
Read or written
Flushed
Fast and Selective Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User
Kernel

TCP/IP input and output
NIC ring

/mnt/nvmm/pktbufs
buf_ofs: 123

packet buffers (static)

metadata header
/mnt/nvmm/pktbufs

 metadata entries
buf_idx  off  len
0       100  1135
1       100   932
3       100  1024

/mnt/nvmm/myapp_metadata

DMA is performed to L3 cache (DDIO)
Unnecessary data is not flushed to DIMM

Unread
Read or written
Flushed
Fast and Selective Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User
netmap API
Kernel

TCP/IP
input and output
NIC ring

metadata_header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
buf_idx off len
0 100 1135
1 100 932
3 100 1024

flushed
unread
read or written

/mnt/nvmm/myapp_metadata
/mnt/nvmm/pktbufs

DMA is performed to L3 cache (DDIO)
Unnecessary data is not flushed to DIMM
Fast and Selective Persistent Write with PASTE

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

User

Kernel

TCP/IP
input and output

NIC ring

DMA is performed to L3 cache (DDIO)

Unnecessary data is not flushed to DIMM
Fast and Selective Persistent Write with PASTE

Unread | Read or written | Flushed

Application
(1) Read data (zero copy)
(2) Write metadata entry
(3) Flush (buffer and metadata)

metadata header
/mnt/nvmm/pktbufs
buf_ofs: 123

metadata entries
buf_idx | off | len
0 | 100 | 1135
1 | 100 | 932
3 | 100 | 1024

/mnt/nvmm/myapp_metadata
/mnt/nvmm/pktbufs

DMA is performed to L3 cache (DDIO)
Unnecessary data is not flushed to DIMM

mmap() | netmap API
User | Kernel

TCP/IP input and output

NIC ring
Implementation

• Extension to netmap memory allocator
  • Exploit Linux NVDIMM stack
  • Claim packet buffers from a given file backed by NVMM
    • e.g., pkt-gen -i eth1@/mnt/pmem/bufs -f rx
Preliminary Results

- Implementation
  - Extend the netmap framework
    - Stackmap for TCP/IP

- 10-88% throughput increase, 9-46% latency reduction

10-88 % throughput increase, 9-46 % latency reduction
Related Work

• Enhanced network stacks
  • MegaPipe (OSDI’12), Stackmap (ATC’16), Fastsocket (ASPLOS’16)
  • IX and Arrakis (OSDI’14), mTCP (NSDI’13), Sandstorm (SIGCOMM’14), MICA (NSDI’14)
    No NVMM aware

• NVMM filesystems
  • BPFS (SOSP’09), NOVA (FAST’15)

• NVMM databases
  • NVWAL (ASPLOS’15), REWIND (VLDB’15), NV-Tree (FAST’15)
    No networking aware
Use cases

- **Persistent Key Value Store** (e.g., Redis)

- **Networked NVMM filesystem**
  - Merge packet metadata and inodes

- **Fault-tolerant, Distributed Middlebox** (e.g., Yoda [EuroSys16])
  - PASTE enables finger-grained state updates
  - Yoda was able to store only some vantage points

- **Software-Defined Monitoring**
  - e.g., new SDN switch action: Logging (OpenSketch [NSDI'13] Mille-Feuille [HotNets'16])
  - Operators mirror traffic for monitoring
  - PASTE enables realtime (local) logging

- **Datacenter-Wide DB**
  - Fast I/O and replication
  - Use PTP for consistency
Conclusion

PASTE: Fast data persistence with named packet buffers on NVMM and zero-copy API

• Implications
  • Network stacks are now a bottleneck for durably storing data
  • Improving network and storage stacks in isolation is not enough
  • We need new stacks design for end systems and middleboxes