FlashNet: Flash/Network Stack Co-Design

Animesh Trivedi, Nikolas Ioannou, Bernard Metzler, Patrick Stuedi, Jonas Pfefferle, Ioannis Koltsidas, Kornilios Kourtis, and Thomas R. Gross

IBM Research and ETH Zurich, Switzerland
Modern Distributed Systems

- data intensive
- run on 100-1000s of servers
- performance depends upon both network and storage
Modern Distributed Systems

- performance depends upon both network and storage
Modern Distributed Systems

- StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs, USENIX '16
- Network Stack Specialization for Performance, SIGCOMM '14
- mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems, NSDI '14
- MegaPipe: A New Programming Interface for Scalable Network I/O, OSDI '12
- ...
Modern Distributed Systems

- StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs, USENIX'16
- Network Stack Specialization for Performance, SIGCOMM'14
- mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems, NSDI'14
- MegaPipe: A New Programming Interface for Scalable Network I/O, OSDI'12
- ...
Modern Distributed Systems

- StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs, USENIX'16
- Network Stack Specialization for Performance, SIGCOMM'14
- mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems, NSDI'14
- MegaPipe: A New Programming Interface for Scalable Network I/O, OSDI'12
- ...
Modern Distributed Systems

- StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs, USENIX'16
- Network Stack Specialization for Performance, SIGCOMM'14
- mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems, NSDI'14
- MegaPipe: A New Programming Interface for Scalable Network I/O, OSDI'12
- NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs, HotStorage'16
- OS I/O Path Optimizations for Flash Solid-state Drives, USENIX'14
- Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems, SYSTOR'13
- When Poll is Better Than Interrupt, FAST'12
- ...

- performance depends upon both network and storage

SYSTOR 2017, Haifa
The Cost of the Gap
The Cost of the Gap

K IOPS

spec.  block IO  netperf  iSCSI  KV  NFS  HDFS

SYSTOR 2017, Haifa
The Cost of the Gap

K IOPS

spec. block IO netperf iSCSI KV NFS HDFS

SYSTOR 2017, Haifa
The Reason for the Gap
The Reason for the Gap
The Reason for the Gap

performance = network IO + server time + storage IO
The Reason for the Gap

\[ \text{performance} = \text{network IO} + \text{server time} + \text{storage IO} \]

application involvement
scheduling
fs lookups and overheads
...

1. request
2. request processing
3. response

flash storage
A Detailed Look: send

userspace

kernel

1. TCP/IP processing
A Detailed Look: send

1. TCP/IP processing
2. receive processing
3. receive request

userspace
---------------------
kernel
A Detailed Look: send

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
A Detailed Look: send

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
6. flash I/O completion
7. response transmission

userspace

kernel
A Detailed Look: send

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
6. flash I/O completion
7. response transmission
8. send
9. TX done
A Detailed Look: `sendfile`

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
6. flash I/O completion
7. response transmission
8. send
9. TX done

userspace
kernel
A Detailed Look: sendfile

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
6. flash I/O completion
7. response transmission
8. send
9. TX done
The FlashNet Approach

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O
The FlashNet Approach

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O

eliminate application involvement

userspace

kernel
The FlashNet Approach

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O

userspace

kernel

eliminate application involvement
reduce file system overheads
The FlashNet Approach

1. TCP/IP processing
2. receive processing
3. receive request
4. fs translation
5. block I/O

- eliminate application involvement
- reduce file system overheads
- enable direct network and storage interaction
The FlashNet Approach

1. TCP/IP processing
2. RDMA processing
3. Eliminate application involvement
4. fs translation
5. Block I/O

reduce file system overheads
enable direct network and storage interaction
The FlashNet Approach

1. TCP/IP processing
2. RDMA processing
3. simple fs layout
4. RDMA
5. block I/O

 Eliminate application involvement
Reduce file system overheads
Enable direct network and storage interaction
The FlashNet Approach

1. TCP/IP processing
2. RDMA processing
3. simple fs layout
4. use VM
5. block I/O

eliminate application involvement
reduce file system overheads
enable direct network and storage interaction
The FlashNet Approach

1. TCP/IP processing
2. RDMA processing
3. block I/O
4. response transmission
5. flash I/O completion
6. TX done
FlashNet: A Co-Designed Network and Storage Stack

- flash virtualization
- I/O management
- ...
FlashNet: A Co-Designed Network and Storage Stack

- contiguous file allocation
- supporting `mmap` & local file I/O
- ...

ContigFS

flash controller
FlashNet: A Co-Designed Network and Storage Stack

- lazy RDMA pinning
- resolving flash & file addresses
- ...

RDMA controller

ContigFS

flash controller
FlashNet: A Co-Designed Network and Storage Stack

FlashNet I/O stack

- RDMA controller
- ContigFS
- Flash controller
FlashNet: A Co-Designed Network and Storage Stack

- file
- virtual address
- LBA
- PBA
- STag
- server application
- RDMA controller
- ContigFS
- flash controller

network control setup expanding to storage
FlashNet: A Co-Designed Network and Storage Stack

file

virtual address

STag

LBA

PBA

server application

RDMA controller

ContigFS flash controller

network control setup expanding to storage
data path from a flash device to a client buffer
FlashNet: A Co-Designed Network and Storage Stack

network control setup expanding to storage
data path from a flash device to a client buffer
Performance Evaluation

How efficient is FlashNet's IO path? Does it help with applications?

...more in the paper

9-machine cluster testbed

CPU : dual socket E5-2690, 2.9 GHz, 16 cores
DRAM : 256 GB, DDR3 1600 MHz
NIC : 40Gbit/s Ethernet
3xNVMe
Flash : 6.6 GB/sec (read), 2.7 GB/sec (write)
        peak 4kB read IOPS: 1.3 M
Performance - IOPS Efficiency

**Number of clients**

**kIOPS**
Performance - IOPS Efficiency

- kIOPS
- number of clients

- socket/file
- FlashNet
Performance - IOPS Efficiency

network saturated

socket/file
FlashNet

38.6%

kIOPS

number of clients

SYSTOR 2017, Haifa
Application-level Performance: KV

<table>
<thead>
<tr>
<th>kIOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
</tr>
<tr>
<td>450</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>350</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>250</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>150</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

Aerospike | socket/file | FlashNet
Application-level Performance: KV

![Bar chart showing performance comparison between Aerospike, socket/file, and FlashNet for put and get operations in kIOPS.](image-url)
Application-level Performance: KV

Aerospike

socket/file

FlashNet

put

get

3.2x

1.8x

kIOPS

SYSTOR 2017, Haifa
Conclusion

Identified performance issues with networked flash

Apply RDMA principles and concepts by extending the path separation idea to a flash controller and a file system

FlashNet is a concrete implementation of this idea - demonstrated its capabilities in micro-benchmarks and applications

Excited to explore new use-cases for FlashNet
Thank you
Unified I/O Life Cycle

server process address space

area1

area2

1. mmap

2. reg_mr

RNIC
- save LBA offset
- generate STag

3.

4. RDMA read req.

5. - resolve STag
- request pages

6. get_LBA

7. lookup and IO

8. response transmitted

9. put_LBA

ContigFS

dir1

file1

file2

DRAM pool

virtualized FTL

LBAs

PBAs
## CPU Cycles Breakdown

<table>
<thead>
<tr>
<th></th>
<th>network</th>
<th>storage</th>
<th>device drivers</th>
<th>scheduling</th>
<th>kernel</th>
<th>request processing</th>
<th>misc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket/file</td>
<td>19.3%</td>
<td>7.3%</td>
<td>6.7%</td>
<td>15.8%</td>
<td>40.1%</td>
<td>4.7%</td>
<td>6.1%</td>
</tr>
<tr>
<td>FlashNet</td>
<td>20.6%</td>
<td>0.8%</td>
<td>6.4%</td>
<td>8.4%</td>
<td>46.7%</td>
<td>11.7%</td>
<td>5.4%</td>
</tr>
</tbody>
</table>