PostMan: Rapidly Mitigating Bursty Traffic by Offloading Packet Processing

Panpan Jin¹, Jian Guo¹, Yikai Xiao¹, Rong Shi², Yipei Niu¹, Fangming Liu*¹, Chen Qian³, Yang Wang²

¹OpenCloudNeXt Group, Huazhong University of Science and Technology  
²The Ohio State University  
³University of California Santa Cruz

OpenCloudNeXt Group: http://grid.hust.edu.cn/fmliu
Peak shopping season is going global

- Sale on Cyber Monday hits new record at $6.6 Billion in 2017[^1]
- Black Friday racks up $5.03 billion in online sales in 2017[^2]
- The 24-hour sale on Nov. 11 reaches $25 billion in sales in 2017[^3]

- Bursty traffic is arriving!
  - Conversation rate in 24h[^4]
    - 35% Cyber Monday
    - 42% Black Friday
  - Statistics on double 11[^5]
    - 325,000 orders/s at peak
    - 256,000 transactions/s at peak
Bursty traffic is a headache!

- Large volume
- Short duration
- Small packets
- Severe overhead

Payload size breakdown:
- [0, 31 bytes]
- (31 bytes, 41 bytes)

Packet processing throughput (Gbps):
- 10 Gb Linux
- 10 Gb IX
  - 64 bytes
  - 64 KB
Traditional remedy: migrating hot data for load balancing

Clients --> Memcached servers

- Time-consuming
- Exacerbate overload
- High packet processing overhead
PostMan: batching and offloading on demand

- Helper batches small packets into large ones
- PostMan offloads packet overhead from overloaded server to helpers
  - Large packets
  - No data migration
  - Rapid mitigation

Clients

PostMan helper nodes

Server experiencing bursty traffic

Memcached servers

Server with normal load
How to assemble small packets?

- Request: a packet sent by a client
- Reply: a packet sent by a server
- Connect: a command to create a connection

Server experiencing bursty traffic
**How to assemble small packets?**

Clients

<table>
<thead>
<tr>
<th>PostMan header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Length: length of the payload

Helper nodes

Memcached servers

Server experiencing bursty traffic
How to assemble small packets?

Clients

Helper nodes

Memcached servers

Server experiencing bursty traffic
Is batching in helpers efficient?

Clients

Helper nodes

DPDK

mTCP

Memcached servers

Server experiencing bursty traffic
Is batching in helpers efficient?

DPDK & mTCP based stack
Efficient packet processing
Is batching in helpers efficient?

- DPDK & mTCP based stack
- Efficient packet processing
- Remove duplicated headers
- Alleviate packet processing overhead

Clients -> DPDK -> mTCP

PostMan header

Payload of assembled packet

Helper nodes

Memcached servers

Server experiencing bursty traffic
Everything works fine, except…

Clients -> Helper nodes

Helper nodes fail, no enough information for re-transmission

Memcached servers

Server experiencing bursty traffic

Server with normal load
Stateless failover mechanism

Clients

Reconnect msg
Last received: 3

Timeout

3 2 1

Helper nodes

Memcached servers

Server experiencing bursty traffic

7 6 5 4 3 2 1

Received

7 6 5 4 3 2 1

Pending

Pending

Freely migrating connection

Stateless

No scalability bottleneck
Programming with PostMan library

Client’s library

pm_connect:
✓ Chooses a helper and connect to the helper.
✓ Sends a special “connect” packet to the helper node.

get_info:
✓ Allows the application to retrieve connection information, such as the number of sent and received packets

Helper nodes

decompose:
✓ Identifies the “connect” packet and notifies application that a new client tries to connect
✓ Disassembles the packet into small packets

Server’s library

compose:
✓ Buffers multiple replies and assemble them
How to enable helpers?

Clients \[\rightarrow\] Helper nodes \[\rightarrow\] Memcached servers

Server with normal load

Server experiencing bursty traffic

Latency is higher than SLA

Brings minimal overhead to the server side
Load balancing across helpers

Clients

Helper nodes

Memcached servers

Server experiencing bursty traffic

Server with normal load

Choose the helper with the lowest utilization

CPU: 50%

CPU: 70%
Load balancing across helpers

- Clients
- Helper nodes
  - CPU: 90%
    - Helper is overloaded
  - CPU: 60%
- Memcached servers
  - Server experiencing bursty traffic
  - Server with normal load
How to disable helpers?

Clients → Helper nodes → Memcached servers

- Server experiencing bursty traffic
- Throughput is low
- Makes the best decisions & overhead is acceptable when load is low

Server with normal load
• Positioning of PostMan
  • PostMan is an alternative solution to data migration for bursty traffic
  • Data migration is the ultimate solution to mitigate bursty traffic
Evaluation Setup

- **Testbed**: CloudLab: 15 machines
- **Machine**: 10 physical cores and hyper-threading, an Intel 82599ES 10 Gigabit NIC
- **Server side**: Memcached, Paxos and IX
- **Helper node**: DPDK 16.07.2
- **Client side**: Ping-pong benchmark and IX
- **SLA**: 500 μs (99 percentile latency, p99)
PostMan vs. Data Migration

2 helper nodes
660 client connections
Mitigation time: \textbf{550ms} vs. 13s

2 helper nodes
960 client connections
Mitigation time: \textbf{750ms} vs. 8s
Memcached vs Memcached + PostMan
- Up to 5 helper nodes
- Load range: $2000K \sim 6000K$
- Throughput: $3.3 \times$

Paxos vs Paxos + PostMan
- Up to 6 helper nodes
- Load range: $500K \sim 5000K$
- Throughput: $2.8 \times$
Performance gain of PostMan

Linux vs Linux + PostMan
- Up to 6 helper nodes
- 8 cores: turning point is **400 bytes**
- 1 core: turning point is **1460 bytes**

IX vs IX + PostMan
- Up to 6 helper nodes
- 8 cores: turning point is **260 bytes**
- 1 core: turning point is **920 bytes**
Fault tolerance

Takes about $0.4s$ to recover all 1000 connections
We propose **PostMan**, an alternative approach to *rapidly* mitigate load imbalance for services processing *small requests*

- **Rapid**: much faster than data migration
- **Efficient**: Fast I/O, user-level stack for packet processing
- **Fault-tolerant**: stateless failover design
- **Scalable**: no scalability bottleneck
Reference


Thank you!

Scan for more!
Is batching in helpers adaptive?

- Batch size ($S$) and batch interval ($T$)
- Collect and buffer packets from clients in a loop
- Record buffer size and waiting time
- Update $S$ and $T$ based on buffer size and waiting time

Lower bound of batch interval is 10us

Lower bound of batch size is $MTU$

Batch interval ($T$): $0.5T$, $T$, $1.5T$

Batch size ($S$): $0.75S$, $S$, $1.25S$
Is batching in helpers adaptive?

Initial: \( S (2000B), T (20\text{us}) \)

Buffers 3000 bytes in 80us

The batched size > \( S \), while waiting time < \( T \)…

Update: \( S (3000B), T (80\text{us}) \)

Increase batch size and decrease interval to improve capacity

Server experiencing bursty traffic
Is batching in helpers adaptive?

**Initial:** $S$ (3000B), $T$ (80us)

- 3000 Bytes
- **Buffers 1500bytes in 160us**

**The batched size < S…**

**Update:** $S$ (1500B), $T$ (160us)

- Decrease batch size and increase interval to improve latency

Server experiencing bursty traffic
Connect to helpers

Clients → Helper servers → Memcached servers

Server with normal load

Server experiencing bursty traffic

Packet #1 duplicated
Switch to other helpers

Clients → Helper servers → Memcached servers

1. Server experiencing bursty traffic
2. Packet #1 duplicated
3. Server with normal load
Migrate connection after pending packets are received

Clients

Helper servers

Memcached servers

Server experiencing bursty traffic

Pending

Pending
Further optimization

- Better load balancing strategy

- Information communication
  - Better solution to update utilization of helpers in clients
  - Better solution of informing clients to disconnect from helpers

- Management of large scale helpers
  - How to startup and shutdown helpers
Adaptive batching

Higher load → Larger batch size & smaller batch interval
Lower load → Smaller batch size & Larger batch interval
Performance of PostMan

A single helper node can process about 9.6 million small messages (2×4.8 million) per second.