KTCP

Kernel Split-TCP Proxy

Alex Markuze, VMware
TALK LAYOUT

• Background
  • Project Pathway
  • What is TCP (Kernel) Splitting is good for?

• KTCP Features.
  • Show me the benefits.

• KTCP Implementation.
  • The nuts n’ bolts.
PROJECT PATHWAY
SD-WAN
NETWORK INFRASTRUCTURE
PROJECT PATHWAY  https://arxiv.org/abs/1812.05582

• Link Health monitoring
• Management
• TCP-Split Proxy
WHAT IS SPLIT-TCP?

Single TCP connection

Leg A. TCP Connection

Leg B. TCP Connection
WHY SPLIT?

TCP Congestion Control contains a feedback loop (ACKs)

Smaller Round-Trip-Time (RTT)
  - Faster recovery from drops
  - Faster throughput ramp-up
  - Can support higher throughput with a smaller buffer (rwin)
  - Reduces unfairness due to competition with small RTT
TALK LAYOUT

• Background
  • Project Pathway
  • What is TCP (Kernel) Splitting is good for?

• KTCP Features.
  • Show me the benefits.

• KTCP Implementation.
  • The implementation drill-down.
TCP 3WH

San Francisco

TCP SYN

SYN ACK

ACK + HTTP Packet

Bangalore

TCP Three Way Handshake = 3WH

0 ms

274 ms

411 ms

Time
TTFB

San-Francisco

0 ms

274 ms

548 ms

Bangalore

411 ms

TCP Three Way Handshake = 3WH

TCP 3WH

TTFB=Responsiveness
TCP SLOW START

San-Francisco

0 ms

274 ms

548 ms

822 ms

Bangalore

411 ms

TCP 3WH

TCP ACK

Last Payload packet

Time
IDEAL TRANSMISSION

San-Francisco

0 ms

274 ms

274 ms HTTP Request

274 ms Reply Payload

548 ms TTFB

822 ms Download Time

Bangalore

411 ms

Time
SPLIT-TCP

San Francisco → Oregon → Mumbai

0 ms

274 ms

TCP 3WH

274 ms

TTFB 548 ms
Download Time

763 ms

763 ms

Time
EARLY SYN

San-Francisco

Oregon

TCP 3WH

Mumbai

Bangalore

0 ms

274 ms

488 ms

703 ms

TCP 3WH

TCP 3WH

Time
PRE-CONNECTION

San-Francisco 0 ms

Oregon

Mumbai

Bangalore

Time
TURBO START

0 ms

TCP 3WH

San-Francisco

Oregon

Mumbai

Bangalore

274 ms

302 ms

332 ms

TCP 3WH
EXPERIMENTAL RESULTS

50MB DOWNLOAD RESULTS

Time To First Byte

<table>
<thead>
<tr>
<th></th>
<th>E2E</th>
<th>Baseline</th>
<th>KTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seconds</td>
<td>0.5</td>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Download Time

<table>
<thead>
<tr>
<th></th>
<th>E2E</th>
<th>Baseline</th>
<th>KTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seconds</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
TALK LAYOUT

• Background
  • Project Pathway
  • What is TCP (Kernel) Splitting is good for?

• KTCP Features.
  • Show me the benefits.

• KTCP Implementation.
  • The implementation drill-down.
WHY THE KERNEL?

User space

Slow & Limited

User space

Kernel space

TCP Stack
Routing
Tunneling
Netfilter
NAT
Connection tracking
eBPF
Kernel Threads

User space

TCP socket

User space

Kernel space

TCP socket
CONNECTION QP

• Split connection meta-data
  ▪ 12B Key:
    ▪ struct socket *rx
    ▪ struct socket *tx

• AUX fields
  ▪ atomic_t refcount;
  ▪ struct rb_node node;
  ▪ wait_queue_head_t wait;
QP CREATION FLOW

1. SYN : Alloc TX Sock Thread on Core X
2. Connection established: Alloc RX Sock Thread on Core X
3. Lookup in QP Tree[X].
QP CREATION FLOW

Netfilter (IRQ)
- Alloc Thread: SYN handler
- Schedule on 2;
  $2 = \text{hash}(5 \text{ tuple})$
QP CREATION FLOW

**SYN handler**
- Add QP
- Wait for RX socket (90 sec)

**Connection handler**
- Lookup QP
- Remove from tree
- Wakeup waiting
- Start forwarding RX to TX

**Proxy server**
- Accept connection from source (RX socket)
- Alloc QP
- Alloc Thread: Connection handler
- Schedule on 2:
QP CREATION FLOW

SYN handler
- Start forwarding TX sock to RX

Connection handler
- Start forwarding RX to TX
THREAD POOL

- Alloc:
  - Slow: may take several milliseconds
  - Limited: Can't be called from NAPI/IRQ

```c
struct task_struct *thread_create_on_node(int (*threadfn)(void *data),
                                           void *data,
                                           int node,
                                           const char *namefmt[], ...);

/**
 * kthread_create - create a kthread on the current node
 * @threadfn: the function to run in the thread
 * @data: data pointer for @threadfn()
 * @namefmt: printf-style format string for the thread name
 * @arg...: arguments for @namefmt.
 *
 * This macro will create a kthread on the current node, leaving it in
 * the stopped state. This is just a helper for kthread_create_on_node();
 * see the documentation there for more details.
 */
#define kthread_create(threadfn, data, namefmt, arg...) \
    kthread_create_on_node(threadfn, data, NUMA_NO_NODE, namefmt, ##arg)
```
BUILDING BLOCK: POOL_ELEM

**POOL_ELEM:**
- `task_struct *task;`
- `Int (*pool_task)(void *data)`
- `void *data;`

**threadfn:**
- While (!kthread_should_stop())
  - `this->pool_task(this->data);`
  - `set_current_state(TASK_INTERRUPTIBLE);`
  - `kthread_pool_reuse(elem);`
  - `schedule();`

- **Fast alloc:**
  - `set; pool_task and data`
  - `Wake task`
  - `Can start anywhere`
MAGAZINE ALLOCATOR (ATC’01)
ZERO-COPY

- RX:
  - NEW: tcp_read_sock_zcopy:
    - based on tcp_read_sock (tcp_splice_read)

- TX:
  - do_tcp_sendpages: used in splice.
  - Modify tcp_sendmsg_locked

/**
 * This routine provides an alternative to tcp_recvmsg() for routines
 * that would like to handle copying from skbuffs directly in 'sendfile'
 * fashion.
 * Note:
 * - It is assumed that the socket was locked by the caller.
 * - The routine does not block.
 * - At present, there is no support for reading OOB data
 * or for 'peeking' the socket using this routine
 * (although both would be easy to implement).
 */

int tcp_read_sock(struct sock *sk, read_descriptor_t *desc,
                  sk_read_actor_t recv_actor)

ssize_t do_tcp_sendpages(struct sock *sk, struct page *page, int offset,
                          size_t size, int flags)
ZERO-COPY EVALUATION

- 16 core Intel Cascade Lake
- 64GB RAM
- 32Gb/s Egress
ZERO-COPY SEND EVALUATION

• 16 core Intel Cascade Lake
• 64GB RAM
• 32Gb/s Egress

Cycles / Bytes.
ZERO-COPY SPLICE EVALUATION

- 16 core Intel Cascade Lake
- 64GB RAM
- 32Gb/s Egress


CONTACT

• E-mail
  • amarkuze@vmware.com

• Code Availability
  • https://github.com/Markuze/ktcp.git