Kernel Bypass

Sujay Jayakar (dsj36)
11/17/2016
Kernel Bypass

• Background
  • Why networking?
  • Status quo: Linux

• Papers
  
Why networking?

Bigger pipes: Intel XL710 chipset (40 Gbps)

Why networking?

Faster storage: Intel P3700 (20 Gbps, 100 µs reads)
Why networking?
Why networking?

Software requirements
Status Quo: Linux
Status Quo: Linux

\[ \text{Incoming Q's} \rightarrow \text{Core} \rightarrow \text{Core} \rightarrow \text{Core} \rightarrow \text{Outgoing Q's} \rightarrow \text{NIC} \]

\[ 0.34 \mu s + 1.26 \mu s + 0.54 \mu s + 1.05 \mu s = 3.36 \mu s \]

(\(+2.23 \mu s \) to \(6.19 \mu s\) if off-core)
Status Quo: Linux

- Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
- 10 Gb/s = 1.25 GB/s = 800K to 15M packets/sec
- Service time is only 67 ns to 1.25 µs per packet!
Status Quo: Linux

- Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
- 40 Gb/s = 5 GB/s = 3.25M to 60M packets/sec
- Service time is only 17 ns to 307 ns per packet!
Status Quo: Linux

• Ethernet frame: 84 bytes to 1538 bytes (1500 MTU)
• 100 Gb/s = 12.5 GB/s = 8M to 150M packets/sec
• Service time is only 6.7 ns to 125 ns per packet!
Status Quo: Linux

- Network stack: demultiplexing, checks, scheduling synchronization needed for multi-core

- POSIX limitations
  - Copy required for `send(2)` and `recv(2)`
  - File descriptor migration among processes

- Context switch overhead
Arrakis: The Operating System is the Control Plane.
Network virtualization

- VNICS managed by “control plane” in kernel
- NIC responsible for bandwidth allocation & demultiplexing
- Intel 82599 chipset
  - Only supports filtering by MAC address, want arbitrary header fields
  - Weighted round-robin bandwidth allocation
- At most 64 VNICS
Storage virtualization

• VSIC: SCSI/ATA queue with rate limiter
• Virtual Storage Area (VSA): Persistent disk segment
• Each VSIC has many VSAs, and each VSA can be mapped into many VSICs (for interprocess sharing)
• Don’t have a device like this, emulated in software with a dedicated core
Arrakis
Arrakis: Control Plane

- VIC management: queue pair creation & deletion
- Doorbells: IPC endpoint for VIC notifications
- Hardware control
  - Demultiplexing filters (just layer 2 for now)
  - Rate specifiers for rate limiting
Arrakis: Doorbells

- SR-IOV virtualizes the MSI-x interrupt registers
- libOS driver handles interrupts in user space if on-core, otherwise goes through control plane
- API is just a regular POSIX file descriptor
Arrakis: User APIs

- Extaris
- POSIX sockets
- Arrakis/N
- Zero-copy rings

- Ethernet
  - `send_packet(queue, array)`
  - `receive_packet(queue) → packet`
  - `packet_done(packet)`
  - Doorbells on completion
IX: More protection

- Network stack in user-space unacceptable
  - Firewall, ACL management in data plane
  - Sending raw packets requires root
  - Does zero-copy send require memory protection?
- Approach: use CPU virtualization to get three-way protection between kernel, networking, and user
IX: Protection can be cheap

- FlexSC (OSDI ’10): Why are syscalls expensive?
- Direct effects: User-mode pipeline flush, mode switch, stashing registers. Only ~150 cycles!
IX: Protection can be cheap

• FlexSC (OSDI ’10): Why are syscalls expensive?

• Direct effects: User-mode pipeline flush, mode switch, stashing registers. Only ~150 cycles!

• Indirect effects: TLB flush (if no tagged TLB), cache pollution with kernel instructions/data

<table>
<thead>
<tr>
<th></th>
<th>IPC</th>
<th>i-cache</th>
<th>d-cache</th>
<th>L2</th>
<th>L3</th>
<th>d-TLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwrite(2)</td>
<td>0.18</td>
<td>50</td>
<td>373</td>
<td>985</td>
<td>3160</td>
<td>44</td>
</tr>
</tbody>
</table>
IX: Protection can be cheap

- Dune (OSDI ’12): Run Linux processes as VMX non-root ring0 with syscalls replaced with hypercalls

- Untrusted code can run in ring3, can cheaply mode switch back into ring0

- Processes can then manipulate page tables, interrupts, hardware, privilege, …
IX: More protection
Why not both?

- IX’s use of VMX non-root ring0 is clever.
- Its embedding in Linux (with Dune) is nice.
- But if they used SR-IOV, they wouldn’t have needed the Toeplitz hash hack and monopolizing a queue.
- And then they would have gotten (fast!) virtualized interrupts to not have to poll.
Arrakis: Evaluation

= 3.36 µs (+2.23µs to 6.19 µs if off-core)
Arrakis: Evaluation

0.21 µs  0.17 µs

= 0.38 µs
Arrakis: memcached
IX: Evaluation

![Graph showing Goodput (Gbps) vs. Message Size for different network configurations. The graph compares Linux 10Gbps, IX 10Gbps, mTCP 10Gbps, Linux 40Gbps, IX 40Gbps, and mTCP 40Gbps. Each network configuration shows a distinct trend with respect to message size.]
IX: Give up on POSIX

- No to POSIX: Non-blocking API with batching
- Packets processed to completion
  - “Elastic” vs. “background” threads
- No internal buffering
- Slow consumers lead to delayed ACKs
IX: Packet processing