Page Fault Support for Network Controllers

Ilya Lesokhin, Haggai Eran, Shachar Raindel, Guy Shapiro, Liran Liss, Sagi Grimberg, Muli Ben-Yehuda, Nadav Amit, Dan Tsafrir

Mellanox Technologies
Technion - Israel Institute of Technology

Compiler, Architecture and Tools Conference (CATC)
Dec 4, 2017, Haifa, Israel
Direct IO

- Provide applications / Guest OS direct access to HW
- Examples
  - SRIOV and PCIe pass-through
  - RDMA: Remote Direct Memory Access
  - User-level packet processing (DPDK)

- Eliminate overheads:
  - Context switch / Hypercall
  - Memory copies
- Customization: users can make their own tradeoffs
  - Latency vs. throughput
  - CPU vs memory
  - Custom API

*Figure 5: Average memcached transaction throughput and scalability. Top y-axis value = 10Gb/s.*

*Peter et. al., Arrakis: The Operating System is the Control Plane, OSDI’14*
IOuser and IOprovider
Address translation

- Direct I/O requires address translation
  - Provide the same isolation needed in CPU
- Most devices do not tolerate page faults
DirectIO mandates pinning

Static pinning:
- Hypervisors pin guest memory for PCI pass-through
- RDMA registration pins memory
- DPDK buffers are pinned

Benefits:
- Easier to implement
- Predictable IOuser performance
The price of pinning

No canonical memory optimizations, achieved using virtual memory and page faults:

<table>
<thead>
<tr>
<th>Demand paging</th>
<th>Over commitment</th>
<th>Page migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed allocation</td>
<td>Swapping</td>
<td>NUMA migration</td>
</tr>
<tr>
<td>Mmap-ed files</td>
<td>Deduplication</td>
<td>Compaction</td>
</tr>
<tr>
<td>Calloc with zero page</td>
<td>Copy on write</td>
<td>Transparent huge pages</td>
</tr>
<tr>
<td>Increase startup time</td>
<td>Increase memory usage</td>
<td>Decrease performance</td>
</tr>
</tbody>
</table>
Partial solutions

- **Fine-grained pinning**: pin before use, release after
  - Overhead voids benefits of DirectIO
- **Course-grained pinning**: pin-down cache
  - Common in high-performance RDMA setups
- **Cons**:  
  - Complicates programming model  
  - Untrusted IOuser makes policy decisions
- **Copying data** to/from a static pinned buffer  
  - Time wasted on copying
We need page faults!

- Allow devices to access non-pinned buffers
  - Similarly to how page faults enable CPU virtual-memory
  - Any valid IOuser buffer could be used

Have the cake and eat it:

- Provide the benefits of DirectIO
- Allow virtual memory based optimizations
- Keep memory management policy in the IOprovider
IO paging today

- State of the art IO page faults
  - PCIe ATS/PRI (Page Request Interface)
    - Intel VT-d SVM (Shared Virtual Memory)
    - AMD PPR (Peripheral Page Request)
  - NVIDIA Unified Memory
  - IBM CAPI

- Target accelerators and GPUs:
  - Execution is suspended until page fault is resolved
    - Blocking a thread
    - Suspending a GPU kernel

- Networking is different...
The receive page fault challenge

- Incoming network data cannot be stopped!
  - Buffer?
    - Expensive and rarely used (e.g. 100 Gbps x 10 ms = 125 MB)
  - Back-pressure the network?
    - Adversely affect unrelated flows
  - Drop?
    - Retransmission timeout reduces performance
RDMA receive page faults

- Network and transport properties
  - Lossless network
    - Typically large retransmission timeouts
  - Provides reliable point-to-point connections in HW
Solution: pause sender

- Page fault is detected on the device, which implements the transport

- We modified the Connect-IB firmware to:
  - Tell sender to pause
  - Ask driver to resolve page fault
  - Drop subsequent packets
Ethernet receive page faults

- Network properties
  - Lossy network
  - Typically SW transport, at IO_user

- Transport layer cannot intervene during page fault

- Just dropping packets and using TCP retransmission doesn’t cut it!
Solution: backup ring

- Direct-IO performance as long as there are no faults

- Fallback due to page faults, similar to para-virtualization:
  - Packets are stored in a pinned ring on the IOprovider
  - Copied to the IOuser, after the PF is resolved
Evaluation
Evaluation

- Implementation:
  - Page fault support for RDMA in Connect-IB firmware
  - Emulated Ethernet backup-ring by duplicating packets to both rings
Periodic IOPF

90% efficiency at $2^{-15}$ page fault frequency

Major/min or doesn’t matter
Storage evaluation

- Standard iSER initiator (iSCSI Extensions for RDMA)
  - Stock Linux kernel
  - fio benchmark
    - Random 64KB reads
- Modified iSER target
  - Based on open-source tgt target implementation
- Staging buffer: 1GB of 512 KB chunks.
Storage evaluation
HPC evaluation

- Added MPI page fault support
  - Implemented in the MXM communication library
  - Obsoleted ~10K’s of LOC of pin-down cache

- Application benchmarks
  - IMB (Intel MPI benchmark) application suite
  - B_eff (effective bandwidth) benchmark
HPC Evaluation

Intel MPI Benchmarks

$B_{\text{eff}}$ effective bandwidth (MB/sec)

<table>
<thead>
<tr>
<th>app</th>
<th>pinning</th>
<th>NPF</th>
<th>copying</th>
</tr>
</thead>
<tbody>
<tr>
<td>beff</td>
<td>16,410±45</td>
<td>16,440±10</td>
<td>8,020±20</td>
</tr>
</tbody>
</table>
Conclusions

- With network page-faults you can have:
  - Direct IO performance
  - Virtual memory canonical optimizations
  - Simplified programming model
- Receive page faults pose a unique challenge

Future work
- Better integration of IO page faults
  - Dirty and access bits
  - Eviction policy
- For more information, see our ASPLOS’17 paper

Questions?