The 7 Deadly Sins of Packet Processing

Venky Venkatesan & Bruce Richardson
The CPU Core

Unpredictable Branches

Not all branches are bad!

If branch is unpredictable, performance will suffer!

The first time a branch is encountered …

Joe Thorn / CC-BY-NC-ND [https://www.flickr.com/photos/joethorn/272181350]
Helping the branch predictor …

• Predicts conditional branches, direct & indirect calls & jumps, returns, loop iterations

➤ Guide the compiler with likely()/unlikely() on error cases, and where humans can be certain
  ➤ Wrongly structured code can waste fetch/decode bandwidth

➤ Let the branch predictor work on runtime data dependent branches

➤ Inline … gives the BTB more context, but bloats code

Sharing Cache Lines

Need to avoid the latency of having cache-lines ping-pong between different cores on a system.
Some basics …

<table>
<thead>
<tr>
<th></th>
<th>Sandy Bridge Ivy Bridge</th>
<th>Haswell</th>
<th>Skylake</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 data access (cycles)</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>L1 Peak Bandwidth (bytes/cycle)</td>
<td>2x16</td>
<td>2x32 load 1x32 store</td>
<td>2x32 load 1x32 store</td>
</tr>
<tr>
<td>L2 data Access (cycles)</td>
<td>12</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>L2 peak bandwidth (bytes/cycle)</td>
<td>1x32</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Shared L3 Access (cycles)</td>
<td>26-31</td>
<td>34</td>
<td>44</td>
</tr>
<tr>
<td>L3 peak bandwidth (bytes/cycle)</td>
<td>32</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>Data hit in L2 or L1D Dcache of another core</td>
<td>43 – clean hit 60 – modified hit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• BUT memory is ~70+ ns away (i.e. 2.0 GHz = 140+ cycles)

Incorrect Prefetching

A cache miss can use up your full packet budget, so make sure you pull in your data before you need it!

Johnmoore6 / Wikimedia Commons / CC-BY-SA-3.0 [https://commons.wikimedia.org/wiki/File:Irish_600kg_euro_chap_2009.JPG]
Prefetching …

• Hardware pre-fetcher will try to predict … and will fetch data that isn’t needed (adds overhead)

• With packed data structures, it sometimes fetches data that another core uses (inadvertently sharing cache lines)

• But in most cases, hardware prefetchers hugely improve application performance
Per-Packet Operations

Any overhead gets magnified when done per-packet.
Some of these aren’t quite obvious …

- Memory mapped I/O & UC (Uncacheable) operations
- Atomic increment/decrement/Compare-swap
- Ring enq/deq (especially those that use atomics)
- Locks
Incorrect Inlining

Trade-off: function calls have an overhead, but add flexibility
In-lining …

• Eliminates parameter passing overhead
• Increases optimization opportunities for the compiler
• More specific branch prediction context
• Mis-predicted branch penalties in a small function are higher e.g. if a branch misprediction results in a return being prematurely taken
Bad Data Structures

Remember to separate per-lcore data onto different cachelines
Making System Calls

Like flushing away cycles....