Modern Page Tables

Jake Brock - CS 256 / 456 - Fall 2013
TLB Miss Handling Cost

- TLB miss handling at 5-10% of system runtime, up to 40% runtime [Jacob.aslpos98]
- DTLB miss handling can amount to 10% of the runtime of SPEC CPU2000 workloads [Kandiraju.sigmetric02]
- As the physical and virtual addresses grow in size, the depth of page table levels increase with generations, thus the number of memory accesses increases for a single address translation, which increases TLB miss penalty
  - e.g. 2 levels in Intel 80386, 3 levels in Pentium Pro, 4 levels in AMD Optern and Intel x86-64
Memory Structure With Page Table Caches

- CPU
- L1 Cache
- L2 Cache
- L3 Cache
- TLB
- Page Table Cache

Hit: Access Address
Miss: Check L2 Cache
Miss: Check Page Table Cache
Memory Structure With Page Table Caches

- L2 is normally where you cache the page table entries, because high turnover in L1 would make it a waste of space (they would be evicted before used).
In this paper, however, for clarity, we will refer to these levels as L4 (Page Directory Pointer), PD (Page Directory) and PT (Page Table).

The x86-64 page table are named PML4 (Page Map Level 4), PDP (Page Directory Pointer), and PT (Page Table). The four levels of the page table design, this procedure requires one memory access per level in the tree. If the entry does contain a pointer, the next index is used to select an entry at the next lower level of the tree. These steps repeat until a leaf node is reached. Each leaf node in the tree contains a pointer to a data page using its physical page offset. The remainder of the 48-bit virtual address is added to the physical address of this data page to obtain the full physical address. In the latter case, the page offset from the virtual address space (or the entry instead points to a data page using its physical page offset. The page number is further split into a sequence of indices.

Figure 1: Decomposition of the x86-64 virtual address.

<table>
<thead>
<tr>
<th>63:48</th>
<th>47:39</th>
<th>38:30</th>
<th>29:21</th>
<th>20:12</th>
<th>11:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>se</td>
<td>L4 idx</td>
<td>L3 idx</td>
<td>L2 idx</td>
<td>L1 idx</td>
<td>page offset</td>
</tr>
</tbody>
</table>

Figure 1: Decomposition of the x86-64 virtual address.
4-Level Translation

Virtual Address: (0b9, 00c, 0ae, 0c2, 016)
(L4 index, L3 index, L2 index, L1 index, Page Offset)
3.2 Translation caches

3.2.1 Split Translation Cache (STC)

A partial translation of a virtual address.

With this tagging scheme, data from one entry is not needed to

Figure 5: An example of the contents of a STC. Each index is 9-bits,
and the data holds a 40-bit physical page number of the next page table
level. An entry in the L2 cache must match on all three indices, an entry
in the L3 must match on two and the L4 on one.
Split Translation Cache

<table>
<thead>
<tr>
<th>L4 index</th>
<th>L3 index</th>
<th>L2 index</th>
<th>Next Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b9</td>
<td>...</td>
<td>0ae</td>
<td>508</td>
</tr>
<tr>
<td>L2 entries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b9</td>
<td>00c</td>
<td>...</td>
<td>508</td>
</tr>
<tr>
<td>L3 entries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b9</td>
<td>00c</td>
<td>...</td>
<td>125</td>
</tr>
<tr>
<td>L4 entries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b9</td>
<td></td>
<td></td>
<td>042</td>
</tr>
</tbody>
</table>

### Figure 5

An example of the contents of a STC. Each index is 9-bits, and the data holds a 40-bit physical page number of the next page table level. An entry in the L2 cache must match on all three indices, an entry in the L3 must match on two and the L4 on one.

- Example - Virtual Address (0b9, 00c, 0dd, 0c3, 929)
- Can locate L3 and L2 page table pages, but not L1, since 0dd not in L2 page table cache.