Online Superpage Promotion
Revisited

Zhen Fang, Lixin Zhang, John Carter, Sally McKee, and Wilson Hsieh

http://www.cs.utah.edu/~impulse
Department of Computer Science
University of Utah

1 Introduction

The amount of data that a typical translation lookaside buffer (TLB) can map has not kept pace with the growth in cache sizes and application footprints. As a result, the cost of handling TLB misses limits the performance of an increasing number of applications. The use of superpages, multiple adjacent virtual memory pages that can be mapped with a single TLB entry, extends a TLB's reach without significantly increasing its size or cost. Previous studies have shown that simple online policies that decide to create superpages dynamically can be effective in reducing TLB penalties.

We reevaluate online superpage policies in the context of the Impulse memory controller, which supports no-copy superpage construction [4]. Our results show that the presence of Impulse changes the tradeoffs in choosing an appropriate policy, and that the most aggressive policy becomes desirable.

2 Background

We evaluate two of the online superpage promotion policies, asap and approx-online, that were developed by Romer et al. [2] asap is a greedy policy that promotes a superpage as soon as all of its component pages have been referenced. The algorithm does not consider reference frequency for the potential superpages, which minimizes bookkeeping overhead. The

price for this simplicity is that the asap policy may build superpages that are rarely referenced later, in which case the benefits of these superpages would not offset the costs of building them.

approx-online uses a competitive strategy to determine when superpages should be coalesced. If a superpage P accrues many misses, we expect that it will be referenced again in the future, and that promoting it will prevent many future TLB misses. Such promotions effectively prefetch the translations for the non-resident base pages in the new superpage. To track this reference information, the approx-online algorithm maintains a counter P.prefetch for each potential superpage P. On a TLB miss to base page p, P.prefetch is incremented for each potential superpage P that contains the referenced page p and at least one current TLB entry. When the miss charges for a superpage P reach a pre-set threshold for superpages of size P.size, the pages that constitute P are promoted into a superpage. For all superpages P that contain P, P.prefetch is incremented by P.prefetch, since whenever P.prefetch was incremented, P.prefetch was, too.

The choice of threshold value is critical to the effectiveness of approx-online. The ideal threshold is small enough for useful superpages to be promoted early, thereby eliminating future TLB misses, but large enough so that the cost of promotion does not dominate TLB overhead. We found that the promotion thresholds used by Romer et al.'s [2] approx-online simulations tend to be too high.

No-copy superpage creation relies on hardware support provided by the Impulse memory controller [1]. Such hardware provides an extra level of address remapping at the memory: unused physical addresses are remapped into "real" physical addresses. We refer to these remapped addresses as shadow addresses, or the shadow address space. From the point of view of the processor and OS memory management system, shadow addresses are used in place of real physical addresses. The existence of shadow memory is transparent to user programs and the processor.

The Impulse memory controller recognizes shadow addresses and translates them to physical addresses through a shadow-to-physical memory controller page table. The operating system is responsible for managing this new level of address translation. Building superpages from base pages that are not physically contiguous can be accomplished by simply remapping the virtual pages to contiguous, aligned shadow pages. The memory controller then maps the shadow pages to the original physical pages.
3 Results and Conclusions

We use the execution-driven simulator Paint [3], which models a variation of a 240 MHz, HP PA-RISC 1.1 processor running a BSD-based microkernel. The 64-kilobyte L1 data cache is non-blocking, virtually indexed, and direct-mapped with 32-byte lines. The 512-kilobyte L2 data cache is non-blocking, physically indexed and 2-way set-associative with 128-byte lines. The memory system has a total memory latency of 60 cycles. The simulated remapping memory controller is based on the HP controller used in high-end workstations.

The TLB holds both instruction and data translations. The base page size is 4096 bytes. Superpages are built in power-of-two multiples of the base page size, and the biggest superpage that the TLB can map contains 1024 base pages. Kernel code and data structures are mapped using a single block-TLB entry that is not subject to replacement. Our results include measurements for two TLBs, a small one with only 32 entries, and a larger one with 128 entries, which let us examine how scaling the TLB affects the applications that we study. The smaller TLB size also is close to the TLB size that Ronger et al. used in their study. They generate their traces using ATOM on a DEC Alpha 3000/700 running DEC OSF/1 2.1, a system that contains a 225 MHz Alpha 21064 processor with a 32-entry DTIB and an 8 entry ITLB, a 2-megabyte offchip cache, and 160 megabytes of main memory.

To evaluate the different superpage promotion approaches on real-world problems, we use nine programs from a mix of sources. Our benchmark suite includes three SPEC95 benchmarks (compress, gcc, and vortex), three image processing benchmarks (raytrace, rotate, and filter), two scientific benchmarks (cga and matmul), and one SPLASH-2 benchmark (radix).

<table>
<thead>
<tr>
<th></th>
<th>compress</th>
<th>cga</th>
<th>raytrace</th>
<th>rotate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remap ASAP</td>
<td>1.84</td>
<td>1.13</td>
<td>1.39</td>
<td>1.30</td>
</tr>
<tr>
<td>Remap AOL</td>
<td>1.83</td>
<td>1.11</td>
<td>1.38</td>
<td>1.47</td>
</tr>
<tr>
<td>Copy ASAP</td>
<td>1.82</td>
<td>0.82</td>
<td>1.20</td>
<td>0.64</td>
</tr>
<tr>
<td>Copy AOL</td>
<td>1.69</td>
<td>1.10</td>
<td>1.46</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Figure 1: Normalized speedups for the different promotion policies with a 32-entry TLB

Figures 1 and 2 present a representative subset of our experimental results. To summarize these results, we find that when creating superpages dynamically:

- Remapping-based promotion outperforms copying-based promotion by up to 30%.

We also found that Remapping-based superpage promotion has better cache performance than copying-based promotion. Depending on the application, the difference in cache performance can significantly affect the speedup of superpage promotion.

Although our results for copying-based promotion are qualitatively similar to Ronger et al.'s, they differ quantitatively. Ronger et al. use trace-driven simulation, so their cost model for promotion is quite simple. Based on our measurements, the costs for copying-based promotion are significantly higher in a real system, largely due to cache effects. We also find that the promotion thresholds used in Ronger et al.'s approx-online simulations tend to be too high.

As applications consume larger amounts of memory, the necessity of using superpages will grow. Our most significant result is that, given relatively simple hardware at the memory controller, a straightforward greedy policy for constructing superpages works well.

References


