CS 5600 Final Project - Memory Efficiency with Huge Pages

Yufei Zhang, Lu Yan

Introduction

Our VM design dynamically allocates different page sizes per memory allocation, from 4KB, 8KB, etc. to 1GB, in order to reduce address translation cost and increase TLB hit rates. However, this approach does not take into account that using a huge page can waste precious fast-tier memory, as some of the subpages of a huge page are rarely accessed but still placed in the fast-tier memory. As an extension to our VM simulator design, this final project focuses on improving the efficiency of memory use in tiered memory systems when huge pages are involved.

Firstly, we need to enable dynamic determination regarding to whether a page is hot, warm, or cold by considering the overall access frequency distribution of pages. We then need to implement functions that dynamically decides a threshold for hot, warm, and cold pages and places pages in the appropriate memory tier. Therefore, we should be able to properly fill the fast tier with the hottest pages.

Secondly, an approach to tackle the fast-tier memory waste issue is breaking up such a huge page into multiple base pages and migrates only the hot subpages into the fast tier. Since splitting a huge page adds additional CPU overheads due to the data copying and TLB shutdown it causes, we must carefully choose the huge pages that need to be split.

Questions

Therefore, the following questions can be raised (may add more):

1. What kinds of huge pages are to be split to effectively save fast-tier memory within reasonable CPU overheads? We can determine a workload and set different standards to decide which huge pages are to be split, and then measure the latency of each experiment.
2. Since some huge pages (that has only a few hot subpages) are split and the efficiency of overall memory use is increased, we expect an overall reduction in the latency of a running workload, because of, for example, less swapping happening between RAM and secondary storage. We can measure the latencies of different workloads both with and without huge page splitting to quantify the performance improvement.

References

https://lpc.events/event/11/contributions/967/attachments/811/1529/Optimize%20Page%20Placement%20in%20Tiered%20Memory%20System.pdf) P3
https://multics69.github.io/pages/pubs/memtis-lee-sosp23.pdf) P2