Jefferson Chien, Cheng Hung He, Heidi Cheng
Professor Amy Ousterhout
CSE291 Final Project Proposal
Oct 20 2022

Dynamic Page Sizing for Remote Memory in Datacenters

Motivation

The conventions to have a fixed set of memory page sizes can be traced back to the early days of kernel development, which in our opinion, cannot keep up with rapidly evolving datacenter workloads. Past research has proven that page size is critical to application performance. We, however, believe that it is impossible to find a single fixed page size that would suit the evolving modern datacenter workloads. Furthermore, emerging network technologies such as RDMA and DPDK make it feasible to deploy remote memory clusters in datacenters. This results in the increased importance of page size since network latency dominates memory access overheads. While many researchers believe that huge page sizes can leverage high spatial locality, researchers such as Calciu and Voelker argued otherwise. We consider arguments from both sides and determine that fine-tuned page sizing may be the best strategy for datacenter applications. Inspired by Shenango's simplistic scheduling policy, we aim to develop a strategy to dynamically determine page size with low overhead for both local and remote memory accesses.

Goals

1. Demonstrate page sizing heavily impacts system performance
2. Show that dynamic page sizing is a NEED for modern datacenter workload
3. Implement a dynamic page sizing algorithm that can deduce the most suitable page size with minimum overhead
4. Provide a good definition of "suitable" page size (Inspired by Shenango)
5. Address virtual address translation issues to provide compatibility for existing software applications
6. Specify a clear and concise application programming interface to delegate page sizing decision policies to user-level applications.

Expected Challenges

1. Existing hardware supports only a limited set of page sizes. This makes flexible page sizes infeasible without specialized hardware support. Therefore, we might need to emulate this system to demonstrate its benefits with the variable page sizes.
2. The huge overhead introduced by emulation can prevent us from identifying the benefits of dynamic page sizing.
3. It is difficult to define the metrics for evaluating the performance of a real-time page sizing strategy.
4. Dynamic page sizing would require the operating system to update page tables for applications and even TLB entries. This might introduce prohibitive high overhead that can compromise the benefits of the proposed strategy. Thus, we must minimize such overheads.

Potential Solutions

1. To address G1, we selected three real workloads, YCSB, Yahoo Streaming, and Redis, and planned to evaluate their performance with page sizes of 4KB, 1MB, and 1GB and identify their memory access patterns. We envision the results of the three benchmarks can help us identify the potential benefits of different page sizes.
2. To address G2 and G3, we propose two solutions
    a. We would like to emulate the whole memory paging system with pure software for both the fixed page size approach and the dynamic page size approach. Pure software emulation can showcase dynamic page sizing performance improvement without introducing the dynamic overhead from instrumentation.
    b. As a proposal, we define both the page fault rate and remote page fetch overhead as the metrics for evaluating the performance of a strategy. We expect a strategy to find the optimal page size that can strike a balance between both metrics.

We'll leave unaddressed challenges and goals to future work.

Timeline

Date	Deliverables
~10/27/2022	- Conduct analysis and investigation on three different types of workloads: YCSB, Yahoo Streaming, and Redis on - Memory access pattern
~11/10/2022	- Design the dynamic page sizing algorithm
~12/1/2022	- Emulate the algorithm with the memory access patterns

	obtained in step 1 to verify its effectiveness.

Future Work

After examining the effectiveness of the proposed algorithm, we would like to emulate a memory management system with dynamic paging capability for both local and remote memory accesses under minimal hardware assistance. Additionally, identify the required hardware features to support the proposed method.