Disaggregate Paging for efficient Large Page Management

Yeonjin Noh
Hanyang University
maguyn@hanyang.ac.kr

Youjip Won
Hanyang University
yjwon@hanyang.ac.kr

Abstract

The number of applications that use large workloads is becoming increasingly diverse, and demand for memory continues to grow. However, the size of the translation lookaside buffer(TLB) can not catch up with the increase in the DRAM capacity, resulting in frequent TLB misses. This TLB miss eventually leads to a hardware address translation overhead, which degrades system performance.

To solve this problem, operating systems such as Linux and FreeBSD support the huge page mechanism that can change the page size to 2MB or 1GB. Linux's huge page technique, transparent huge page (THP), constructs a huge page through a set of continuous 4 KB pages(called base pages). For example, When THP need to allocate a 2MB-size huge page, THP allocates continuous 512 base pages and treat them as one huge page. The use of huge page technologies can greatly reduce the TLB miss rate by increasing the size of the memory area managed by the TLB.

However, the huge page supported by the operating system causes many problems due to inefficient allocation of memory. In the case of the Linux THP, huge page-sized memory is allocated regardless of the size of the data, causing internal fragmentation problems. In addition, since there is no memory reclaim technique for huge page area, inefficient memory reclaim operation is inevitable. For example, when some data access is biased in KB-size area of huge page, even if the remaining area of huge page is not accessed and can be reclaimed, whole huge page area remaining unreclaimable.

In this paper, we develop Dual Page Management technique that solves the internal fragmentation of THP and the lack of an efficient page replacement technique.

In order to solve the internal fragmentation problem of THP, Dual Page Management allocates huge page when the page size of the requested page is equal to or larger than a certain ratio(e.g. 75%) of the huge page, In the other case, use an existing allocator(e.g. slab) to allocate base pages. By this technique, Dual Page Management can prevents internal memory fragmentation and allocates memory area with granularity.

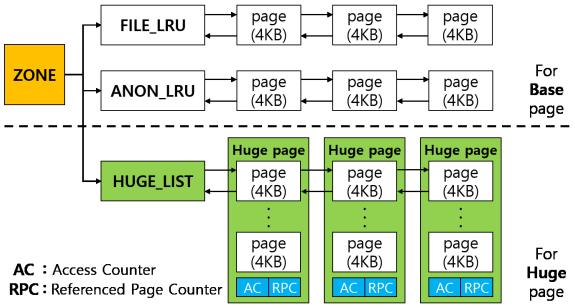

Figure 1: Page list of Dual Page Management

In the Dual Page Management, HUGE_LIST, which is a separate data structure like figure 1, is used for detect biased data in huge page and efficient memory reclaim. HUGE_LIST maintains access counter(AC) and referenced page counter(RPC) variables for each huge page. The AC variable records the number of process accesses and the RPC variable records the number of base pages referenced within one huge page. These two variables are used when the memory reclaim daemon(e.g. kswapd) examined that huge page is reclaimable. When one of the two variables is below the threshold that predefined, the memory reclaim daemon split huge page into individual base pages and insert them into the existing LRU list. Based on the existing LRU policy, this split base pages can be classified into frequently used base pages and non-used base pages. If both variables are below the threshold, to speed up memory reclaim operation, the memory reclaim daemon reclaim huge page directly without split. These techniques of Dual Pages Management can identify biased memory access areas within huge pages and enable efficient memory reclaim over existing LRU schemes.