The Super Advantage: Optimizing Memory Management with Transparent Superpage Support

Akhilesh Boppana
Stony Brook University
Stony Brook, NY, USA
aboppana@cs.stonybrook.edu

Thejesh Arumalla
114963271
Stony Brook University
Stony Brook, NY, USA
tarumulla@cs.stonybrook.edu

Nikhil Deshakulkarni Bhaskar
Stony Brook University
Stony Brook, NY, USA
ndeshakulkar@cs.stonybrook.edu

Abstract

In our work, we present the implementation of super page management for multi-process workloads. The approach includes reservation concept, promotion/demotion logic, and alignment with virtual address space. We have implemented a simulator that takes the virtual address and processId from the Apache Web Server Benchmark's memory trace using Intel's Pin tool as input. By aligning super pages with virtual address space, we reduced the TLB misses, resulting in better TLB coverage and complexity time. We evaluated the performance on various benchmarks, and the experimental results indicated that using super pages can effectively reduce TLB misses and improve time complexity. Our code can be found here: Memory Management Simulation using Superpages

1 Introduction

The exponential increase in available memory in modern computer systems has presented new challenges for operating system designers, who must balance the need for efficient memory usage with the need for high system performance. Super pages, which are larger-than-normal page sizes, have emerged as a promising solution to these challenges. In our study, we present an implementation to super page management that includes a reservation concept, promotion/demotion logic, and alignment with virtual address space.

We utilize the concept of spatial locality to optimize memory access patterns by grouping together data that is likely to be accessed together. By doing so, we are able to improve overall system performance by minimizing the number of TLB misses that occur during memory access. This approach is particularly effective when applied to the management of superpages, as superpages are inherently designed to be benefited by spacial locality. We used this concept of reservation, by reserving contiguous pages as a superpage in advance, the operating system ensures that a contiguous block of physical memory is available to accommodate the superpage. This avoids the overhead of searching for and allocating multiple individual physical pages when a superpage is requested.

Additionally, this can simplify superpage management by ensuring that only one superpage size is used. This eliminates the need to manage multiple superpage sizes, which can be complex and inefficient. We also used promotion/demotion logic to maintain a balance between the number of super pages in use and the number of free pages available, ensuring that the system always has enough free memory available for new requests and reducing fragmentation

In terms of system performance, our approach helps to reduce the number of TLB misses. TLB misses can be a significant bottleneck in system performance, as they result in a lengthy process of retrieving requested pages from main memory. By aligning super pages with virtual address space, we can reduce the number of TLB misses, resulting in better TLB coverage and reduced complexity time. Overall, our methodology provides a comprehensive and effective approach to super page management in modern operating systems, with significant benefits for both end-users and system designers alike.

2 Related Work

In this section, we discuss the previous work conducted on different superpage solutions with a focus on the following aspects: reservations, relocation, eager superpage creation, hardware support, and demotion issues.

In recent years, several studies have explored reservations in super page management that allow processes to reserve a certain amount of memory in advance. This reservation approach [4] ensures that the future memory access is part of a superpage readily stored in a TLB for faster memory access which is particularly useful for large-scale systems. Researchers in this domain have also studied page relocation-based schemes, which involve moving pages to a new location when they are promoted to superpages. This relocation approach can effectively improve memory performance but must also recover the costs associated with copying the pages.

Some operating systems like IRIX [2] and HP-UX [3] have adopted an eager superpage creation approach. In this approach, the size of the Superpages is specified by the user, but this approach is not transparent and can result in reduced flexibility. More recently, hardware support for super page management [1] has also been explored, such as the use of a contiguous virtual superpage that is mapped to discontiguous physical base pages. This hardware approach can help improve memory performance, but it does not address demotion issues, such as large pages that are partially dirty or referenced.

Our approach improves upon the previous works by implementing the concept of reservation. We also used relocation, which runs as a background process in identifying the contiguous memory locations and o selects those pages as a superpage that contribute the most to contiguity. We also implemented the promotion and demotion techniques with algorithms in place to tackle the issues during demotion during cache and TLB eviction without the need for any hardware support.

3 Methodology

We assume the memory address is of 48 bits, with the first 36 bits representing the page number and the rest representing the offset.Super page is of size 4 and is aligned and starts at multiples of 4. There are 2^{36} base pages and 2^{34} super pages initially.

3.1 Free Page Lists

We maintain two free page lists both representing the physical memory

1. Free Base Page List - Free Base Page list is a list of free continous base pages where each element is a tuple showing the start and end base page, its initial value is $[(0, 2^{36})]$
2. Free Super Page List - Free super page list is a list of contiguous available super pages, with initial value $[(0, 2^{34})]$

Example - To illustrate how it works, let's assume there is a request for superpage 2(base pages: 8,9,10,11), there free base page list is updated to $[(0, 7)(12, 2^{36})]$, the free super page list is updated to $[(0, 1), (3, 2^{34})]$

3.2 Promotion Logic

The system maintains a trace of the previous 10,000 instructions, which helps in analyzing the memory access patterns of the running program. By examining these instructions, the system can identify which continuous four base pages (a group of four adjacent pages) are being accessed most frequently.

Once the frequently accessed four base pages are determined, the system performs a process known as promotion. Promotion involves replacing the smaller individual pages with larger super pages that encompass those pages. This consolidation of pages into super pages reduces the overhead of memory management by treating them as a single entity and increases the TLB coverage.

3.3 Demotion Logic

The system keeps track of the number of untouched base pages within each super page. An untouched base page refers to a page that has not been accessed or modified since it was loaded into memory.

The purpose of measuring the number of untouched base pages is to identify super pages that are not being fully utilized. If a super page has at least k untouched base pages, where k is a predetermined threshold, it indicates that a significant portion of the super page is not actively used or modified by the running program.

When such underutilization is detected, the system performs a process known as demotion. Demotion involves breaking down the super page into smaller individual pages, effectively reversing the promotion process. By demoting the underutilized super page, the system frees up memory space and allows for more efficient memory management.

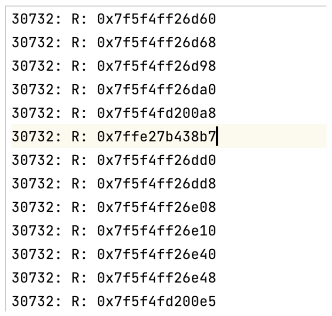
Demotion of a super page with a sufficient number of untouched base pages helps optimize memory utilization by returning the memory resources to the pool of available pages. This allows the system to allocate memory more effectively to other active processes or data that are in greater demand and reduces internal fragmentation.

3.4 Reservation

To facilitate easier promotions and optimize memory management, we adopt a strategy where, instead of assigning a single base physical page, we allocate a super page comprising four base pages when an initial memory request is made. By doing so, we reserve the remaining three base pages within the super page, which streamlines the promotion process. This is to take advantage of the spatial locality.

4 Experiments

We took the trace of apache benchmarks by using 1000 requests, 2 processes. The trace contains the processID, virtual address, R/W flag.



```
30732: R: 0x7f5f4ff26d60
30732: R: 0x7f5f4ff26d68
30732: R: 0x7f5f4ff26d98
30732: R: 0x7f5f4ff26da0
30732: R: 0x7f5f4fd200a8
30732: R: 0x7ffe27b438b7
30732: R: 0x7f5f4ff26dd0
30732: R: 0x7f5f4ff26dd8
30732: R: 0x7f5f4ff26e08
30732: R: 0x7f5f4ff26e10
30732: R: 0x7f5f4ff26e40
30732: R: 0x7f5f4ff26e48
30732: R: 0x7f5f4fd200e5
```

Figure 1. sample trace output of the PIN tool for apache benchmark program

4.1 Impact of TLB size on TLB Misses

In Figure 1 below it shows the impact of TLB size on TLB misses, as the number of instructions increase which is expected.

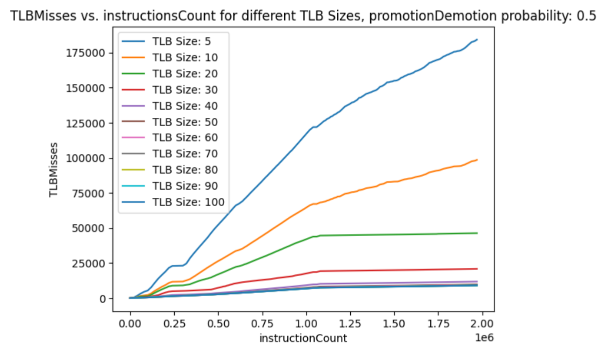


Figure 2. TLB Misses vs TLB size

4.2 Using promotions and demotions to reduce TLB misses

We introduced a parameter: promotion_demotion_probability serves as a configurable setting that allows the system to control the frequency of promotion and demotion operations based on specific requirements or performance considerations. By adjusting this parameter, the system can fine-tune the behavior of memory management to achieve an optimal balance between memory utilization and performance.

When the parameter is set to 0, it indicates that the promote/demote logic is disabled, and the system does not actively perform any promotions or demotions of super pages. This setting may be useful in certain scenarios where the benefits of promotion and demotion are not deemed significant or where the system prioritizes other aspects of memory management.

On the other hand, if the parameter is set to a value between 0 and 1, it signifies that the promote/demote logic is enabled, and happens with a probability equal to the parameter value.

As we see in Figure 3 as we increase the frequency of number of promotions, the number of TLB misses decreases

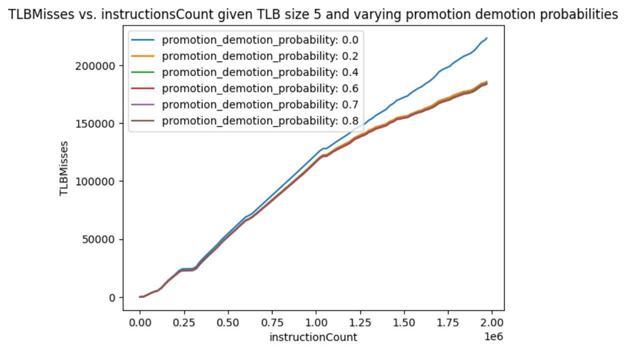


Figure 3. TLB Misses vs promotion/demotion probability

4.3 Reservations

We are not seeing a noticable difference though, we assume it could be that our specific traces which we generated from the benchmarks could be causing it

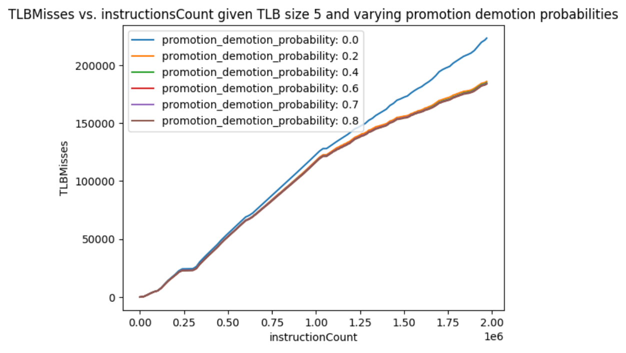


Figure 4. TLB Misses vs reservation

5 Conclusion

This work presents an implementation of super page management for multi-process workloads. We evaluated the performance by running simulations on various benchmarks i.e., Apache Web Server Benchmark, and Siege http benchmarks, and generated the memory trace using Intel's Pin tool. We simulated these memory traces and observed that as the size of TLB increases, TLB misses are reduced. Also, when promotions & demotions were performed frequently, we observed a decrease in TLB misses when compared to a system without superpages. And the trend is such that the more often promotion/demotion was performed, the lesser the TLB misses. However, we could not notice any significant change in the TLB values when the reservation was done, this could be attributed to the memory traces of the particular benchmarks used.

6 Future Work

As a future direction of this work, we plan to investigate the costs associated with copying data to the disk when the RAM is full. This will allow us to better understand the performance impact of superpages and how to optimize it. Additionally, we intend to introduce the concept of a dirty bit in the TLB tables. By keeping track of which pages have been modified, we can avoid unnecessary writes to disk and improve overall performance.

We could also explore the impact of varying the size of superpages on performance and identify optimal values for these parameters.

Finally, we plan to evaluate our approach on a wider range of workloads and system configurations to ensure its effectiveness in diverse settings. This will involve testing our approach on larger systems with more complex workloads and verifying that it can scale to meet the needs of modern computing environments.

References

[1] Zhen Fang et al. "Reevaluating online superpage promotion with hardware support". In: *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*. IEEE. 2001, pp. 63–72.

[2] Narayanan Ganapathy and Curt Schimmel. "General Purpose Operating System Support for Multiple Page Sizes." In: *USENIX Annual Technical Conference*. 98. 1998, pp. 91–104.

[3] Indira Subramanian et al. "Implementation of Multiple Pagesize Support in HP-UX." In: *USENIX Annual Technical Conference*. 1998, pp. 105–119.

[4] Madhusudhan Talluri and Mark D Hill. "Surpassing the TLB performance of superpages with less operating system support". In: *ACM SIGPLAN Notices* 29.11 (1994), pp. 171–182.