NAME: ANNU REG NO:24088

REPORT ON ASSIGNMENT 2 (HPCA)

Introduction

Memory management is a crucial component affecting overall system performance in contemporary computer designs. An important memory management problem is dealing with Translation Lookaside Buffer (TLB) misses, which happen when a requested virtual address is not present in the TLB. Performance deterioration may result from this since the system will have to visit the page table in memory, adding to the delay.

Our goal in this project is to use huge pages (2MB in size) to optimize a workload and decrease TLB misses. Because the workload is supplied through a shared library, memory access is not consistent, which presents chances for performance enhancement through careful mapping of virtual memory areas. The main objective is to locate memory areas that frequently have TLB misses and use huge pages to remap them in order to enhance application performance.

Methodology

2.1 Data collection:

The methodology for gathering and examining memory access statistics, the application of big page mappings, and an assessment of the performance gains made possible by this strategy will all be covered in the study.

Gathering memory access data and determining which locations are most frequently the cause of TLB misses is the initial stage in this approach. We accomplished this by utilizing the perf mem tool, which is intended for memory access profiling and TLB miss detection.
To get the RAM trace data, the following commands were run:

$ perf mem record -o parsed_perf_data.data make run SRN0=24088

$ perf mem -i parsed_perf_data.data> parsed_perf_data.txt

During the workload's execution, this command logs a trace of memory accesses, including the addresses of accessible memory and the associated TLB misses. The data is written in file parsed_perf_data.txt.

2.2 Data Analysis

Finding the most frequently accessed memory areas that were responsible for the bulk of TLB misses was the next step after gathering the memory access data. To automate this study, we created the analyze.py Python script.

The following tasks are carried out by the script:

1. Memory Access Data Parsing: Memory addresses and the quantity of TLB misses linked to each address are extracted from the memory trace file.
2. Aggregating by Regions: 2MB virtual memory regions are used to aggregate memory accesses. Large pages in the system are 2MB in size, hence this grouping is required.
3. Regions Ranked by TLB Misses: The total number of TLB misses is used to rank the combined regions.

Command used to run the python script file:

$python3 analyze.py 8

2.3 Modifying the main.c file:

The next step was to alter the given C program to allocate huge pages at the best memory locations after they had been determined. This was accomplished by modifying the main.c file to allocate memory using the MAP_HUGETLB flag, which guarantees that the designated memory region uses huge pages, using the mmap system call.

2.4 Evaluation of Performance

Performance metrics were collected when the program was run with and without large pages in order to assess how well the large page mappings worked. Prior to and following the optimization, the number of TLB misses was the main statistic of interest.

In order to determine the speedup obtained by lowering the quantity of TLB misses, we additionally measured the program's execution time.

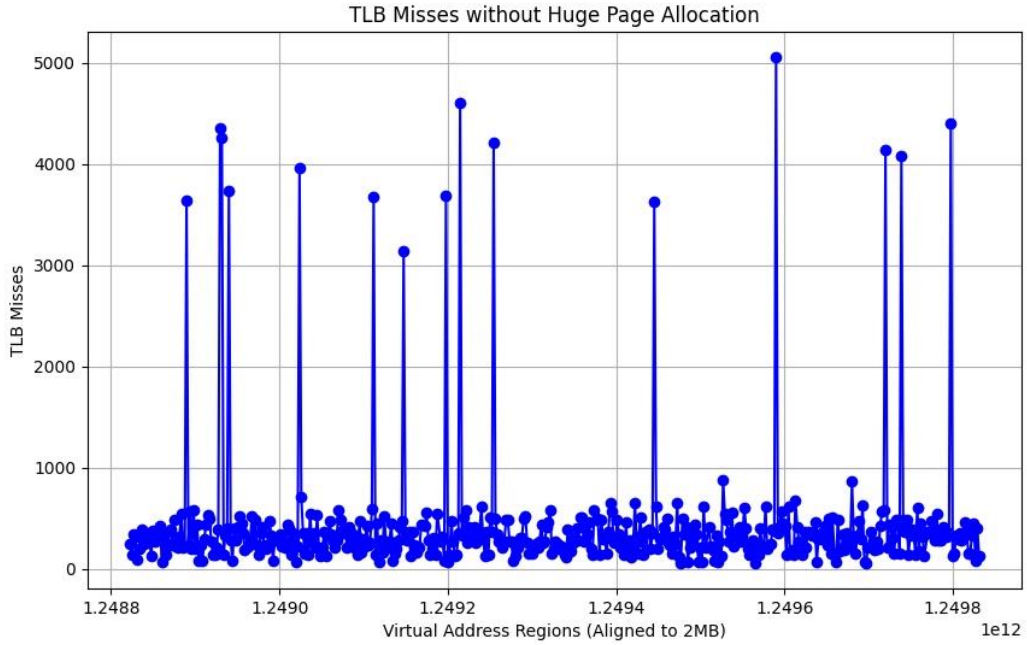
The following command was used to evaluate no of tlb misses:

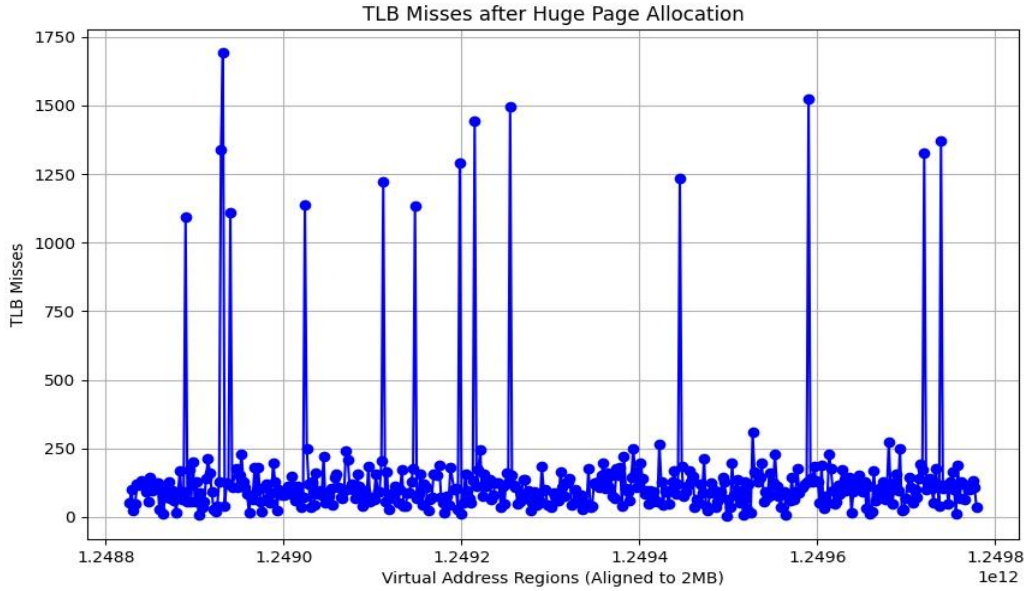
$ perf stat -e dTLB-load-misses,dTLB-store-misses,iTLB-load-misses make run SRN0=24088

3. Findings and Interpretation

3.1 TLB misses:

We created a graph that displays the quantity of TLB misses for various 2MB memory locations within the address space of the program. Different 2MB memory regions are shown by the x-axis, while the number of TLB misses in each zone is shown by the y-axis.





1. Before optimization: The graph indicates that a notably greater number of TLB misses occur in specific memory areas.
2. After Optimization: The number of TLB misses in the chosen locations dramatically decreased after the huge pages were distributed, suggesting that the optimization was successful.

	dTLB load misses	dTLB store misses	iTLB load misses
Before huge pages	7,43,59,75,672	73,600	10,508
After huge pages	6,93,01,34,712	65,961	13,211

3.2 Speedup:

The program's execution time was recorded both before and after optimization. The execution time before allocating huge pages was 48.3176s and after deploying huge pages was 44.5387s. we achieved a speedup of 1.0848. This improvement resulted from fewer TLB misses, which decreased the overhead of page table access during memory operations.

4. Conclusion

This task illustrated how crucial memory management optimization is, especially for workloads that experience frequent TLB misses. We were able to reduce TLB misses significantly and increase program performance by remapping memory locations that are responsible for the majority of TLB misses to large pages.
The main points of this assignment are:

1. The usefulness of memory bottleneck detection profiling tools such as perf mem.
2. How using huge pages sparingly might result in considerable performance advantages.
3. Understanding memory access patterns is crucial for maximizing system performance.
4. We were able to increase the workload's efficiency and show how specific memory optimizations might result in noticeable speed gains by using this method.