Assignment 2

E0-243

High Performance Computer Architecture (HPCA)

Ayush Kumar Singh

(24851)

DIFFERENT STEPS FOLLOWED :

Step 1 : Collection of sample of memory accesses

- Download the zip file provided in the assignment.
- Extract the zip file, It comprised of 4 files – libwork.so, main.c, makefile and work.h
- Compile the main.c file (provided) by linking the libwork.so file with it.

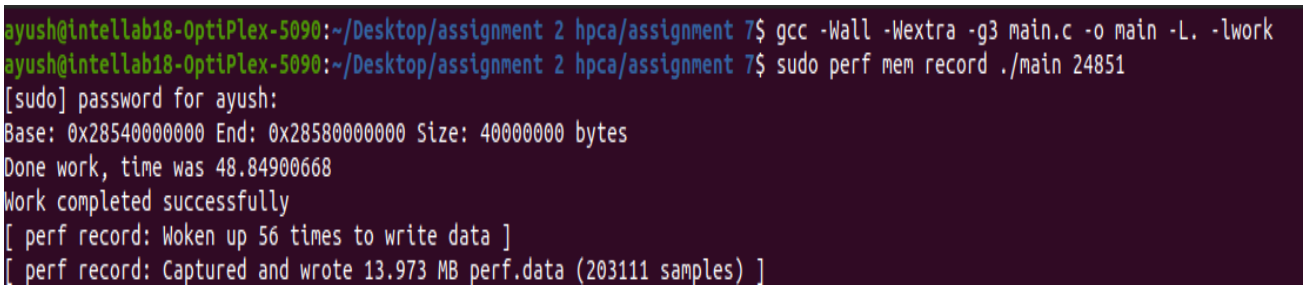
  gcc -Wall -Wextra -g3 main.c -o main -L. -lwork

  - here -wall and -wextra are used to handle warning and -L. -lwork is used to link the libwork.so file.
  - It will generate the 'main' compiled file.
- Install the perf tool :
  - **sudo apt update**
  - **sudo apt install linux-tools-common linux-tools-$(uname -r)**
  - **sudo apt install linux-perf**
- record the samples –

  sudo perf mem record ./main 24851

  This generates a file named 'perf.data'
- convert the collected samples in 'perf.data' to text file named 'report.txt'

  sudo perf mem report -i perf.data > report.txt



Step 2 : Write a Python script to find the most optimal locations for Large pages

- Install python
  - **sudo apt update**
  - **sudo apt install python3**
  - **sudo apt install python3-pip**
- Create a python file named ‘analyze.py’

  touch analyze.py
- Write the script
- Run the script

  python analyze.py 8
  - Here 8 represents the number of large pages
  - Running this script will generate ‘largepages.txt’ which will list down the base addresses of the 8 large pages.

```
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$ touch analyze.py
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$ python analyze.py 8
Optimal large page regions saved to 'largepages.txt'.
```

Step 3 : Use of Large Pages in the program

- Modify the ‘main.c’ file to allocate the huge pages on the addresses present in the largepages.txt (hardcore the addresses in hexadecimal format from the largepages.txt file which contains addresses in decimal format)
- Compile the main.c file

  gcc main.c -o output -L. -lwork
- Run the file

  sudo ./output 24851

```
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$ gcc main.c -o output -L. -lwork
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$ sudo ./output 24851
Base: 0x28540000000 End: 0x28580000000 Size: 40000000 bytes
Mapped 2MB huge page at address 0x2854de00000
Mapped 2MB huge page at address 0x28579c00000
Mapped 2MB huge page at address 0x28550800000
Mapped 2MB huge page at address 0x28549a00000
Mapped 2MB huge page at address 0x2857b400000
Mapped 2MB huge page at address 0x2854ac00000
Mapped 2MB huge page at address 0x28578e00000
Mapped 2MB huge page at address 0x28552800000
Done work, time was 29.503591867
Work completed successfully
```

Step 4:Again use the command to collect the memory sample after allocating huge pages

- Run the command to collect data

  `sudo perf mem record ./main 24851`
- Again convert it to text file named ‘report2.txt’

  `sudo perf mem report -i perf.data > report2.txt`

```
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$ sudo perf mem record ./main 24851
Base: 0x28540000000 End: 0x28580000000 Size: 40000000 bytes
Done work, time was 45.339496724
Work completed successfully
[ perf record: Woken up 53 times to write data ]
[ perf record: Captured and wrote 13.144 MB perf.data (191047 samples) ]
ayush@intellab18-OptiPlex-5090:~/Desktop/assignment 2 hpca/assignment 7$
```

ANALYSIS:

A TLB miss (Translation Lookaside Buffer miss) occurs when the CPU tries to translate a virtual memory address to a physical address, but the required page table entry is not found in the TLB.

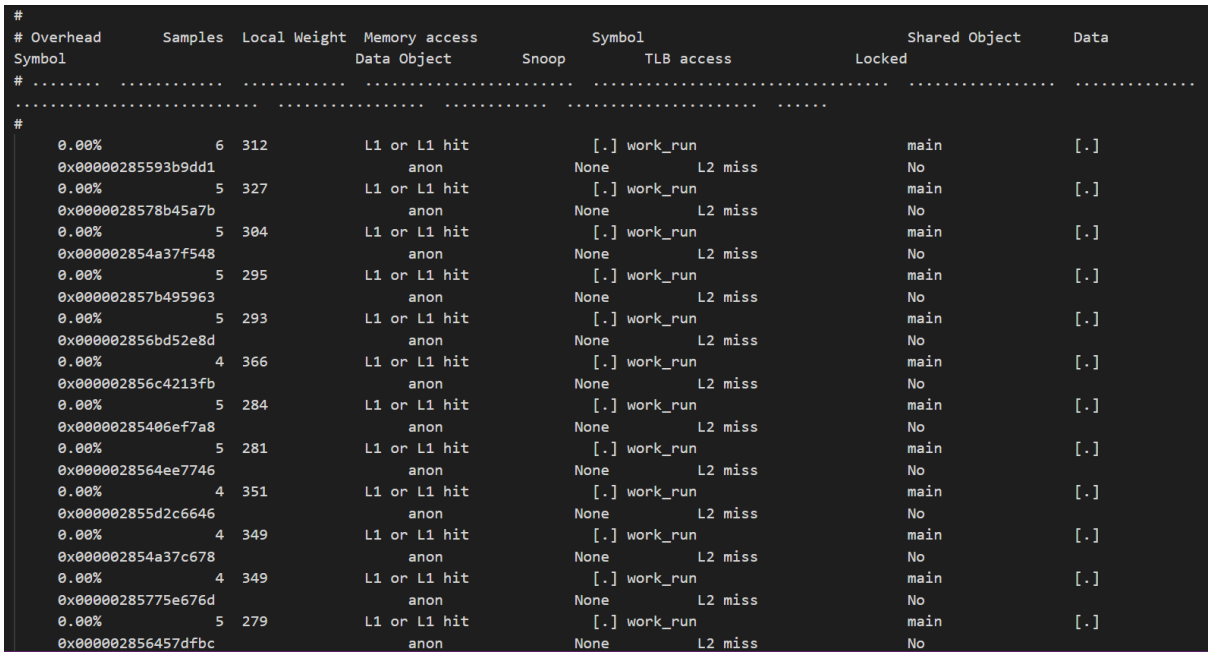
After performing the full analysis, It is found that using huge pages reduces the TLB misses. With 2MB pages, fewer entries need to be maintained in the TLB since each entry covers a larger portion of the virtual address space compared to the default 4KB pages. Programs often exhibit spatial locality, where memory accesses are concentrated in nearby regions. Huge pages take advantage of this by mapping larger contiguous regions, reducing the need for frequent TLB lookups for smaller regions.

When I first collected the samples without using huge pages, it gave around **203111** samples consuming about **13.973 MB** of ‘perf.data’ file. Total time taken to perform the mem record task took around **48.849 seconds**.

After using the huge pages of size 2 MB, when the data was collected, it gave around **191047** samples consuming around **13.144 MB** of ‘perf.data’ file. Total time taken to perform mem record task took around **45.339 seconds**.

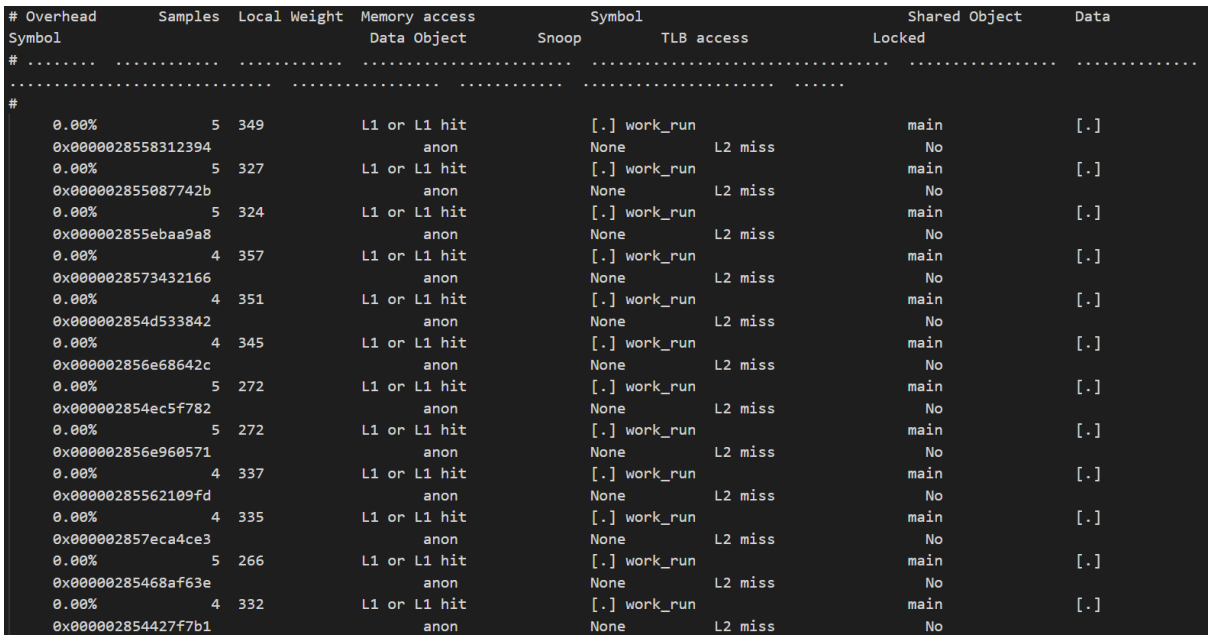
So the **speedup** achieved was around = 48.849/45.339 = **1.077**

First sample was collected in the report.txt file, and the second sample (i.e. sample collected after using the huge pages) was collected in the report2.txt file.



# Overhead	Samples	Local Weight	Memory access	Symbol	Shared Object	Data Symbol
			Data Object	Snoop TLB access	Locked	
0.00%	6	312	L1 or L1 hit	[.] work_run	main	[.]
0x00000285593b9dd1			anon	None L2 miss	No	
0.00%	5	327	L1 or L1 hit	[.] work_run	main	[.]
0x0000028578b45a7b			anon	None L2 miss	No	
0.00%	5	304	L1 or L1 hit	[.] work_run	main	[.]
0x000002854a37f548			anon	None L2 miss	No	
0.00%	5	295	L1 or L1 hit	[.] work_run	main	[.]
0x000002857b495963			anon	None L2 miss	No	
0.00%	5	293	L1 or L1 hit	[.] work_run	main	[.]
0x000002856bd52e8d			anon	None L2 miss	No	
0.00%	4	366	L1 or L1 hit	[.] work_run	main	[.]
0x000002856c4213fb			anon	None L2 miss	No	
0.00%	5	284	L1 or L1 hit	[.] work_run	main	[.]
0x00000285406ef7a8			anon	None L2 miss	No	
0.00%	5	281	L1 or L1 hit	[.] work_run	main	[.]
0x0000028564ee7746			anon	None L2 miss	No	
0.00%	4	351	L1 or L1 hit	[.] work_run	main	[.]
0x000002855d2c6646			anon	None L2 miss	No	
0.00%	4	349	L1 or L1 hit	[.] work_run	main	[.]
0x000002854a37c678			anon	None L2 miss	No	
0.00%	4	349	L1 or L1 hit	[.] work_run	main	[.]
0x00000285775e676d			anon	None L2 miss	No	
0.00%	5	279	L1 or L1 hit	[.] work_run	main	[.]
0x000002856457dfbc			anon	None L2 miss	No	

report.txt file

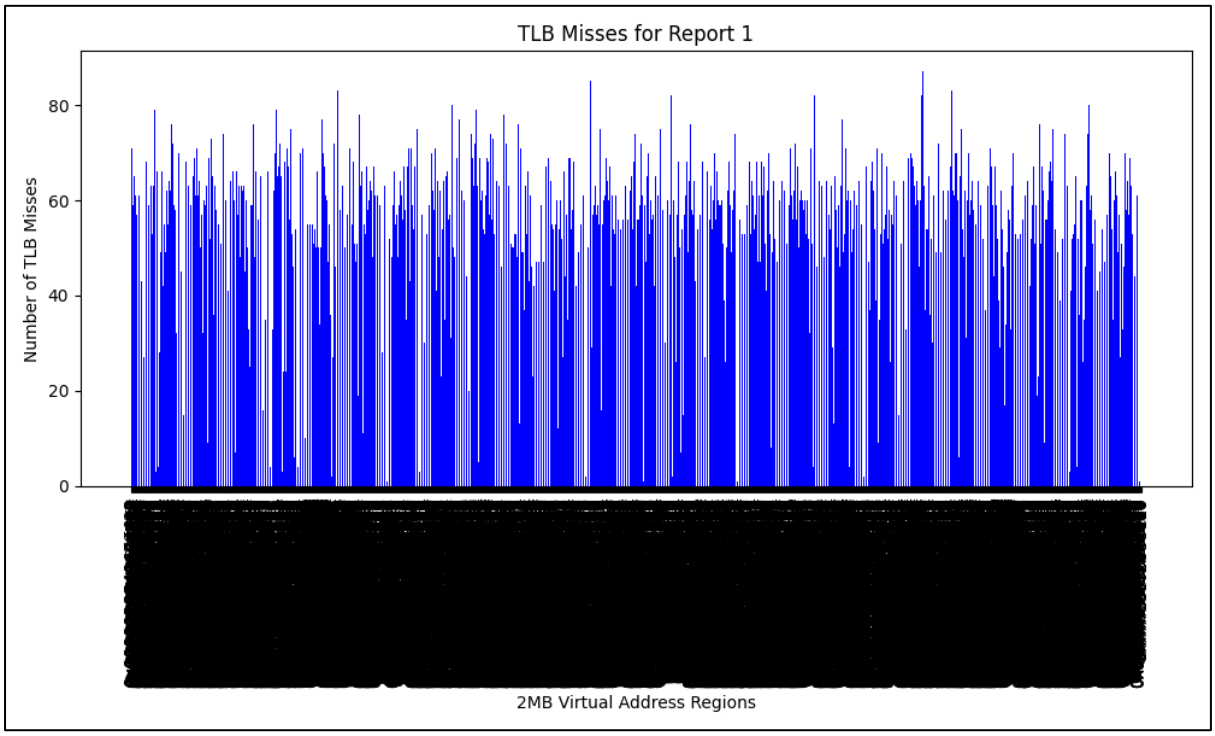


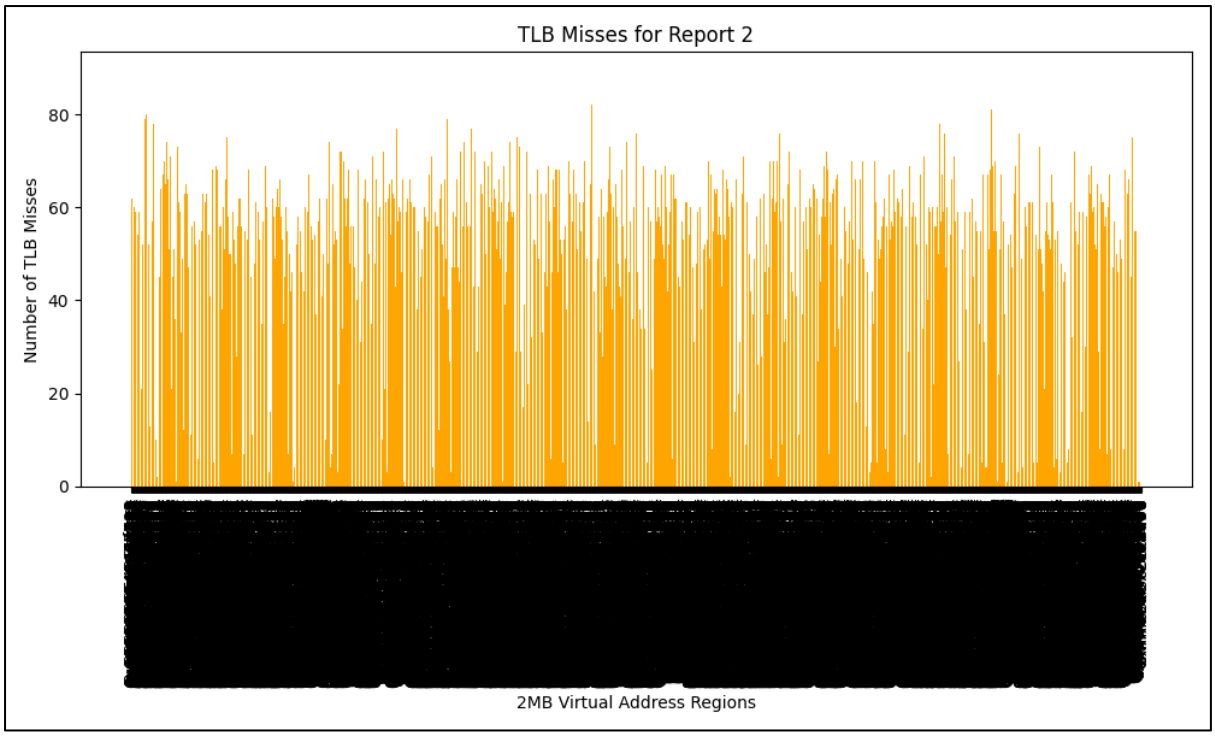
# Overhead	Samples	Local Weight	Memory access	Symbol	Shared Object	Data Symbol
			Data Object	Snoop TLB access	Locked	
0.00%	5	349	L1 or L1 hit	[.] work_run	main	[.]
0x0000028558312394			anon	None L2 miss	No	
0.00%	5	327	L1 or L1 hit	[.] work_run	main	[.]
0x000002855087742b			anon	None L2 miss	No	
0.00%	5	324	L1 or L1 hit	[.] work_run	main	[.]
0x000002855ebaa9a8			anon	None L2 miss	No	
0.00%	4	357	L1 or L1 hit	[.] work_run	main	[.]
0x0000028573432166			anon	None L2 miss	No	
0.00%	4	351	L1 or L1 hit	[.] work_run	main	[.]
0x000002854d533842			anon	None L2 miss	No	
0.00%	4	345	L1 or L1 hit	[.] work_run	main	[.]
0x000002856e68642c			anon	None L2 miss	No	
0.00%	5	272	L1 or L1 hit	[.] work_run	main	[.]
0x000002854ec5f782			anon	None L2 miss	No	
0.00%	5	272	L1 or L1 hit	[.] work_run	main	[.]
0x000002856e960571			anon	None L2 miss	No	
0.00%	4	337	L1 or L1 hit	[.] work_run	main	[.]
0x00000285562109fd			anon	None L2 miss	No	
0.00%	4	335	L1 or L1 hit	[.] work_run	main	[.]
0x000002857eca4ce3			anon	None L2 miss	No	
0.00%	5	266	L1 or L1 hit	[.] work_run	main	[.]
0x00000285468af63e			anon	None L2 miss	No	
0.00%	4	332	L1 or L1 hit	[.] work_run	main	[.]
0x000002854427f7b1			anon	None L2 miss	No	

Report2.txt file

The report.txt file comprised of around 203k samples while the report2.txt file comprised of approximately 190k samples.

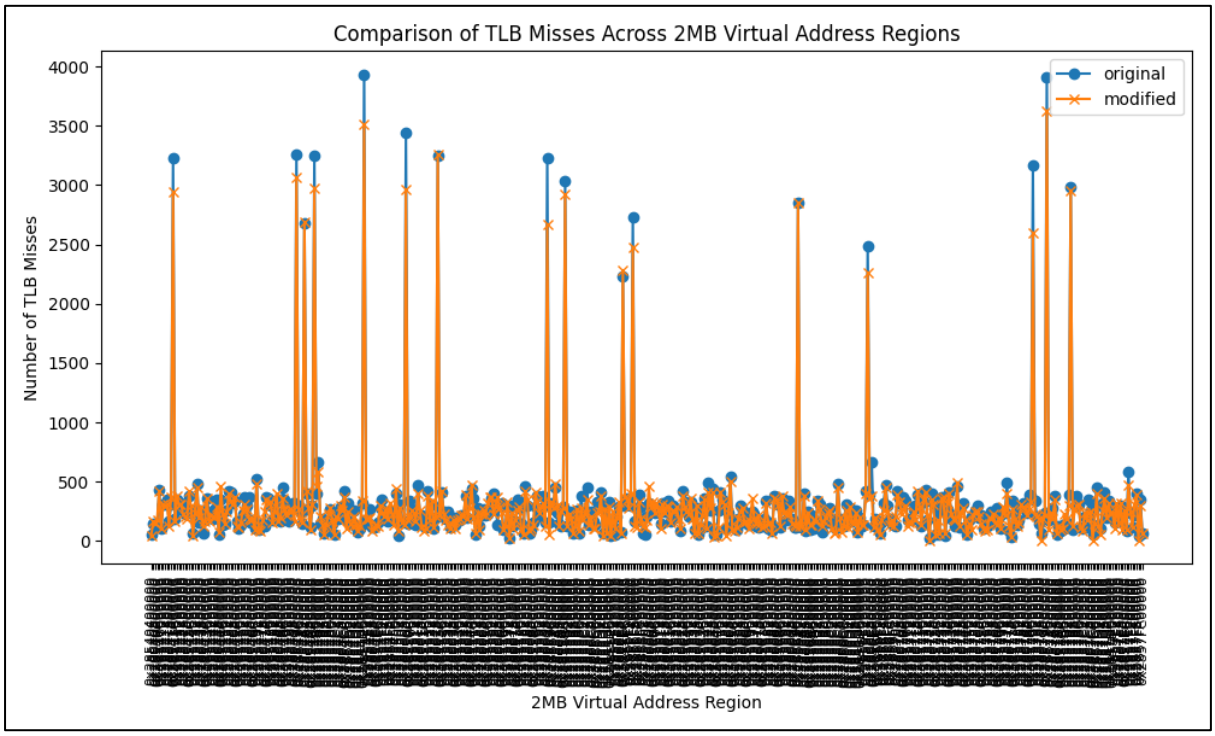
I have used python libraries pandas and matplotlib libraries of the python to draw the graph of the following samples from the 2 files.



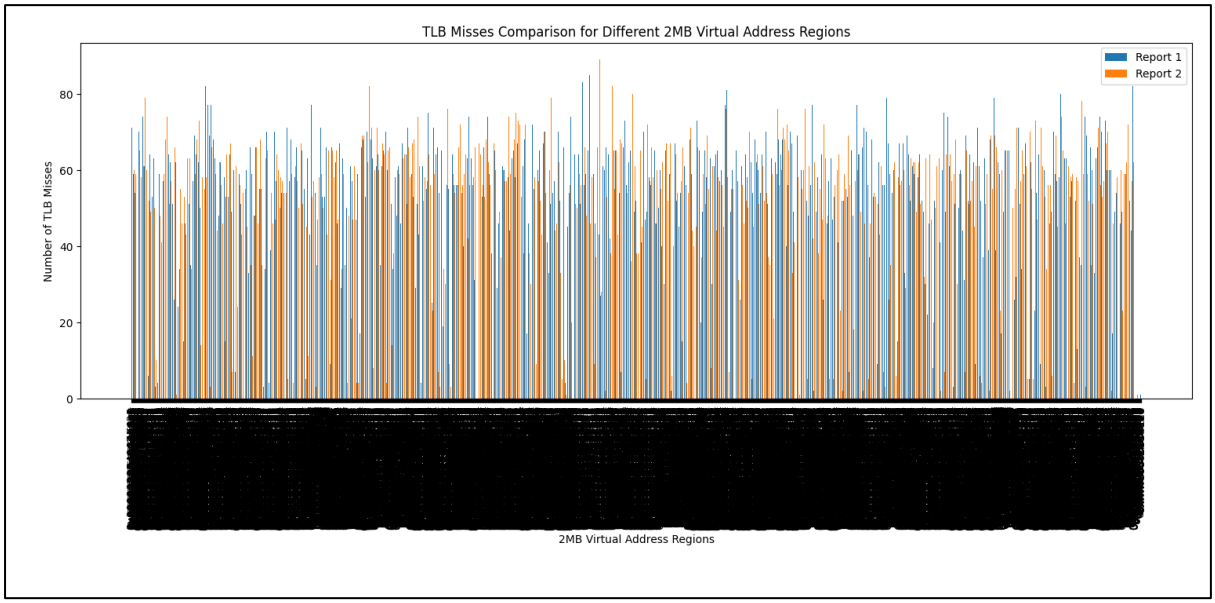


The above bar graph shows the TLB misses for report 1 (when no huge pages were used) and TLB misses for the report 2 (when huge pages were used) seperately.

In order to compare then together, a Line graph and a Bar graph is drawn to capture their performance side by side.



Line graph showing the comparison



Bar graph showing the comparison

The graph clearly shows that by using huge pages of size 2MB, the TLB misses can be controlled in a much better way.

SUMMARY:

In this analysis, I explored the impact of using large pages (specifically 2MB huge pages) on memory access patterns and TLB (Translation Lookaside Buffer) misses. After collecting memory access samples both with and without the use of huge pages, I observed a significant reduction in TLB misses, indicating improved memory management efficiency. The initial sample collection yielded approximately 203,111 samples and a data size of 13.973 MB, taking about 48.849 seconds. In contrast, the post-optimization collection resulted in roughly 191,047 samples, with a smaller data size of 13.144 MB and a reduced recording time of 45.339 seconds. This demonstrates a speedup factor of approximately 1.077. Visualizations generated using Python's pandas and matplotlib libraries clearly illustrated the reduction in TLB misses when utilizing huge pages, confirming that larger memory pages enhance spatial locality and minimize TLB lookup frequency. Overall, the findings underscore the effectiveness of using huge pages in optimizing memory access performance.