Assignment 2

E0-243 HPCA

This assignment should be done **individually** (no groups). Do not discuss your solutions with others. While analyzing your code if our tool finds cases of cheating, a “F” grade will be assigned for the course.

Due Date: 19th October, 2024, 11:59 PM

Problem

Context: As discussed in class, it is not often easy to find enough physical memory contiguity to allocate large/huge pages. Thus, it is important to be judicious while choosing the virtual address region (2MB long here) that you may want to map with large pages to minimize the number of TLB misses. The following assignment is about judiciously choosing virtual address regions for an unknown workload to be mapped using large pages.

You are provided with a shared library (.so) that has some unknown workload. It, however, exposes two functions as API: work_init() for initialization & memory allocations and work_run() for executing it. You are also provided with a main program that calls these functions appropriately and a Makefile to build it. To execute the program, considering the last five digits of your serial number is 12345, you can execute:

```
SRNO=12345 make run
```

You are allowed to use up to a maximum number of 2MiB large pages (say, 8) to minimize the number of TLB misses in the workload and improve its performance. We have observed that the given workload does ***not*** access all parts of its allocated memory in a uniform manner. Thus, your goal would be to deploy large pages for virtual address regions that witness most TLB misses.

You may follow the following approach to achieve this goal.

Step 1: Collect a sample of the memory accesses

Use the “perf mem” tool to collect a sample of the memory access by running the workload and saving it to a file. Your collected data should at least include the addresses of the accessed memory and TLB misses.

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

Your script must:

1. Be named analyze.py
2. Take one integer argument “n”, that is the maximum number of deployable large pages
3. It would use the perf mem tool to collect the sampled memory access trace and analyze tool’s output to find virtual address regions (2MB), which would be most beneficial to be mapped using 2MB large pages.
4. Save the “n” most optimal base address for 2MB large pages (one per line in decimal format) in a file named ‘largepages.txt’

Step 3: Use large pages in the program

Modify the marked section in the main.c file to use 8 huge pages of size 2 MB each at the virtual address regions that you chose in the previous step. Test the program to make sure it works for any given “n” large pages.

Submission

1. Bundle your python script (analyze.py) and the modified main.c file in a zip file named Assignment2_<last 5 digits of your serial no>.zip
2. Submit a report containing at least:
    a. The command you used to collect the memory access data
    b. Your analysis methodology.
    c. A graph that shows the number of TLB misses for different 2MB virtual address regions in the program’s address space. The x-axis should represent different 2MB virtual address regions, while the y-axis should report the number of TLB misses for the given virtual address region corresponding to the x-axis. You should provide this graph before (the original program) and after allocating large pages (your modified program).
    d. Report speedup after deploying the large pages.

Notes

- Some of the programming parts might be evaluated using automation. Do not output (from the code) anything additional/more than asked for.

- The perf mem tool works smoothly on Intel machines. You can use the machines in Well Fargo Lab. It works on AMD machines, too, but some kernel/driver version issues may come up.

Useful references that will help with the assignment.

- https://man7.org/linux/man-pages/man1/perf.1.html
- https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/monitoring_and_managing_system_status_and_performance/getting-started-with-perf_monitoring-and-managing-system-status-and-performance#introduction-to-perf_getting-started-with-perf
- https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/monitoring_and_managing_system_status_and_performance/recording-and-analyzing-performance-profiles-with-perf_monitoring-and-managing-system-status-and-performance
- https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/monitoring_and_managing_system_status_and_performance/monitoring-application-performance-with-perf_monitoring-and-managing-system-status-and-performance#monitoring-application-performance-with-perf_monitoring-and-managing-system-status-and-performance
- https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/monitoring_and_managing_system_status_and_performance/profiling-memory-accesses-with-perf-mem_monitoring-and-managing-system-status-and-performance
- https://www.man7.org/linux/man-pages/man2/mmap.2.html
- https://www.man7.org/linux/man-pages/man2/madvise.2.html