Fall 2020 - CS 519 - Homework 1 (100 points, Due October 6th, 11:55pm)

This homework will consist of two parts: (1) using shared-memory for IPC, and (2) test your knowledge of caching. This is an individual homework.

Part 1 - IPC Shared Memory (50 points)

In this part, you will implement a pipe-based and shared memory-based IPC communication between two processes. The "HW-1" folder contains (1) IPC-pipe.c and (2) IPC-shmem.c.

Part 1.a IPC Using Pipes (20 points)

First, in IPC-pipe.c, you will add a code that forks a child process (using `fork()`). The parent and child processes must communicate using Linux pipes and share work to perform matrix multiplication. The size of the matrix must be configurable using an input argument. You will get a bonus of 5% (for Part 1.a) if your code allows multiple child processes to communicate and share work with parent and other child processes.

Recall that for IPC pipes uses `pipe(pipefd)` to create a pipe, and use `read(pipefd, buf, size)` and `write(pipefd, buf, size)` for communication.

A tricky part is to coordinate between the parent and child processes. Your code must synchronize (less frequently) between parent and child processes. To synchronize between parent and child processes, you can use semaphore (`semctl` and `semop`) system calls. We have added some dummy functions to initialize, reserve, release semaphores. Here's a reference

The matrix sizes should be large enough. We will test sizes up to 1000 x 1000. So, make sure to allocate and release memory dynamically.

Your code must also check the correctness of the matrix multiplication output, and print the time taken to perform the multiplication.

Part 1.b IPC Using Shared Memory (20 points)

Next, in IPC-shmem.c, you will add a code to perform matrix multiplication when parent and child processes communicate using shared memory by using calls like `shmget()` and `shmat()`.

You will get a bonus of 5% for this part if your code allows multiple child processes to communicate and shared work with the parent process.

Part 1.c IPC Scalability Points (10 points)

A good solution must scale with the increase in the number of CPU cores. Remember the class discussions about CPU scaling, the related issues, and ways to address them. As one increases the number of CPUs, the cost of matrix multiplication should ideally reduce. Of course, there is a limit to CPU scaling.

You must clearly show a graph that shows the core count (in x-axis) and the performance in the y-axis (similar to Linux scalability paper). If your solution does not scale beyond a certain core count, you must explain why.

Reporting Part 1 Results

Your report will compare the matrix mucltiplication performance when using pipes and shared memory for different matrix sizes. If your code supports multiple child processes, then your report must show the performance for different child process count.

Part 2 - Cache Size Test (50 points)

Write a C/C++ program to determine the (a) L1, L2, LLC (last-line) cache size, (b) cacheline size (c) TLB size, and (c) memory latency (approximate) of a computer system.

You can use the starter file cache-tlb.c to get started. Note that you are not allowed to use existing Linux scripts or tools to extract this information.

You can use the function (currently empty) to add your code. Feel free to extend or change the arguments of the function.

- i. Report the results from iLab.
- ii. Clearly explain the limitations of your current code.

A submission that reports the correct numbers without explaining the insights used in obtaining them will not get significant credit. While writing the code you must be careful about microarchitectural enhancements such as out-of-order execution and prefetching that may affect memory system behavior.

Submission Instructions

- All executables must be compiled using a single Makefile. If you do not know how to write a Makefile, refer here
- Your submission must be compressed and submitted using Sakaai. tar -zcvf HW-1-.tar.gz HW-1

Computing Resource

Please use iLab for this assignment.

Starting Early

This an essential homework for understanding the basics. Please start working on this homework early. If you have questions, make sure to ask them during office hours.