PERFORMANCE COMPARISON FOR IPC USING PIPES AND SHARED MEMORY`

PROJECT REPORT

CS220 – Operating Systems

18K-1198	Mubeen
18K-1184	Anas Siddiqui
18K-1294	Jaafar bin Farooq

Project description:

In this project we explored two methods of interprocess communication pipes and shared memory and we implemented 3 problems with each method . the problems we implemented were Matrix addition , Matrix multiplication, Fibonacci series.

Project objective:

The main goal of this project was to compare the efficiency of these 2 interprocess communication methods when applied to all three problems

Comparison result:

Fibonacci Series

Using pipe:

Execution time Data:

Child process time : 0.464ms

Parent process : 0.328 ms

Total time = 0.792 ms

Using shared memory:

Execution time Data:

Child process time : 0.028ms

Parent process : 0.034 ms

Total time: 0.062 ms

The result shows that when this problem was done using pipes it took 0.034 milliseconds while using shared memory time taken was 0.792 millisecond which shows Shared Memory method was faster

Matrix multiplication

Using pipe:

Execution time Data:

Child process time : 0.212ms

Parent process : 0.199 ms

Total time: 0.411

Using shared memory:

Execution time Data:

Child process time : 0.003 ms

Parent process : 0.006 ms

The result shows that when this problem was done using pipes it took 0.411 milliseconds while using shared memory time taken was ~0.01 millisecond which shows Shared Memory method was faster

Matrix Addition

Using pipe:

Execution time Data:

Child process time :0.219ms

Parent process : 0.125 ms

Total time =0.344 ms

Using shared memory:

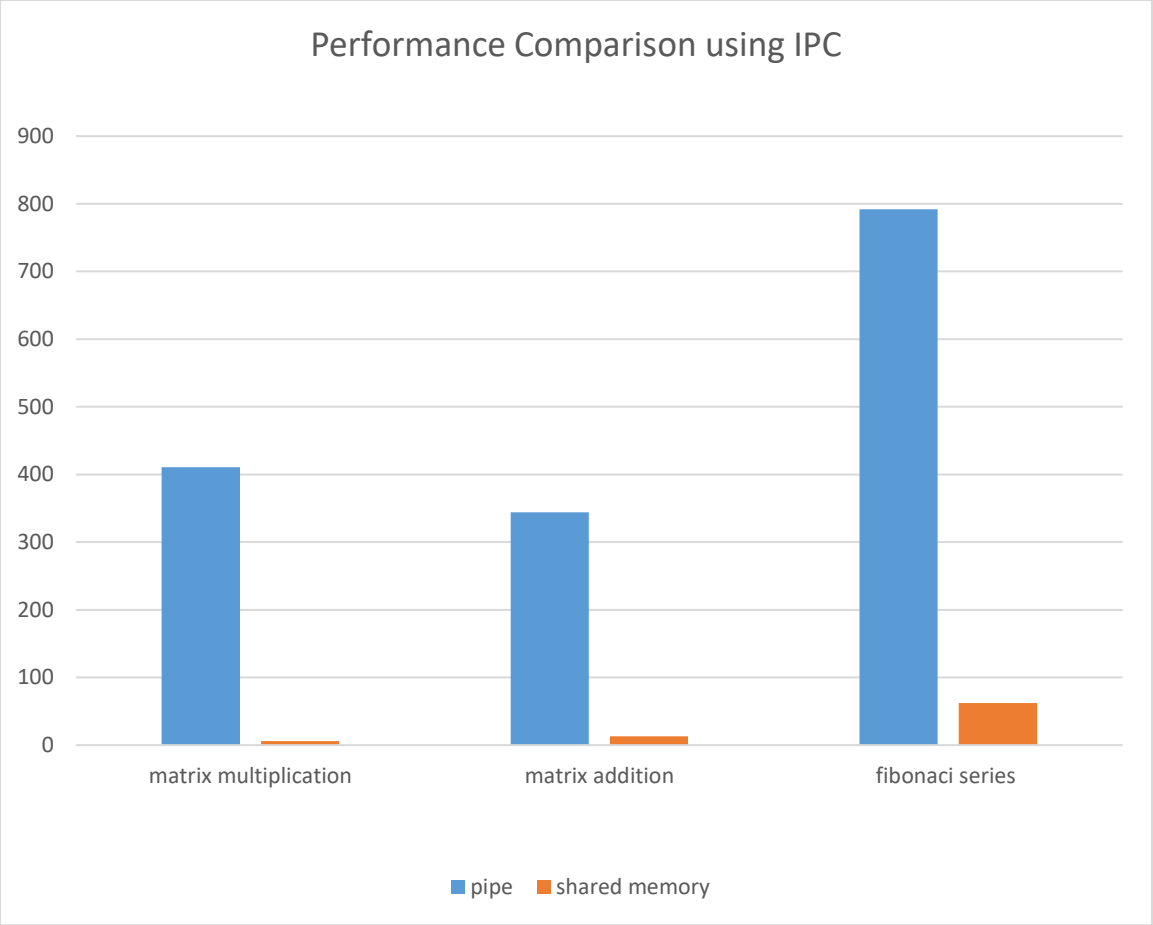
Execution time Data:

Child process time : 0.007ms

Parent process : 0.006 ms

Total time = 0.013ms

The result shows that when this problem was done using pipes it took 0.007 milliseconds while using shared memory time taken was 0.013 millisecond which shows Shared memory method was faster in this problem



1/100th of 1ms on vertical axis.

Conclusion:

With pipes the synchronization is simple and built into the pipe mechanism itself - your reads and writes will freeze and unfreeze the app when something interesting happens. With shared memory, it is easier to work asynchronously and check for new data only once in a while - but at the cost of much more complex code.

Shared memory also gives you more control over buffering and resource use - within limits allowed by the OS it's you who decides how much memory to allocate and how to use it. With pipes, the OS controls things automatically, so once again you lose some flexibility but are relieved of much work.

As seen above, in the time comparison, we can easily conclude that there is a huge difference in time complexities of two methods. Shared memory is way faster than pipe for inter process communication.