Parallel and HPC

HW3 Report

Link to github repository:

https://github.com/DMelkonyan111/Parallel-and-HPC-Homeworks

Assignment 1

In assignment 1, I created a multi-threaded dice game using POSIX threads. I set up multiple player threads, with each thread representing a player. Each player rolls a six-sided die using the rand() function for several rounds. After each round, all threads wait at a POSIX barrier. This ensures that every player finishes rolling before we determine the winner. Once all players reach the barrier, one thread compares the dice results, figures out the round winner, and updates the win counter. After all rounds are completed, the program calculates which player has the most wins and prints the overall winner. The barrier ensures synchronization among all players during every round.

```
Round 1 winner: Player 3
Round 2 winner: Player 1
Round 3 winner: Player 2
Round 4 winner: Player 0
Round 5 winner: Player 1
Overall winner: Player 1
```

Assignment 2

In assignment 2, I simulated a multiplayer game lobby using threads and a POSIX barrier. I created multiple player threads, where each player pretends to get ready by sleeping for a random amount of time. Once they finish preparing, each player prints that they are ready and waits at a barrier. This barrier makes sure the game does not start until all players are ready. Once all threads reach the barrier, they move on and print "Game Started!" This shows how multiple threads can synchronize before proceeding to the next phase.

```
Player 1 ready
Player 2 ready
Player 3 ready
Player 0 ready
Player 4 ready
Game Started! Player 4
Game Started! Player 2
Game Started! Player 3
Game Started! Player 1
Game Started! Player 0
```

Assignment 3

In assignment 3, I set up a weather data collection system using sensor threads. Each thread acts as a sensor that generates a simulated temperature value. After collecting the temperature, all sensor threads wait at a barrier to ensure that data collection is done by every sensor. Once all threads finish, the program calculates the average temperature from the collected values and prints it. The barrier makes sure that the processing phase starts only after all sensors have provided their data.

Assignment 4

In assignment 4, I created a three-stage pipeline simulation using multiple worker threads. Each thread goes through three sequential stages of work. After completing each stage, all threads wait at a POSIX barrier before moving to the next stage. I used separate barriers for each stage to ensure strict synchronization. This approach guarantees that no thread begins stage 2 until all threads finish stage 1, and the same goes for stage 3. The program shows how barriers can coordinate execution phases in parallel systems.