|  |  |  |  |
| --- | --- | --- | --- |
| **Solution Type** | **Number of Slaves** | **Execution time** | **Explain the result** |
| Sequential Solution | 1 | 8.680009 | This “control” result is of the “regular” sequential execution. MPI was not used in this solution so we get the straightforward time it takes to execute “Heavy” for all inputs. |
| Static Task Pool | 2 | 5.937294 | 2 processes execute the program statically with a predetermined portion of the tasks (50% each) and then reduce the results to sum the result of each task. We see here an improvement to the “sequential” execution. In this solution, the final sum might be delayed in case one of the processes is slower from the other. Execution time will be limited by the time the slowest process takes to execute. |
| Static Task Pool | 4 | 4.618956 | 4 processes run a predetermined portion of the tasks (25% each). We see a slight improvement over 2 processes but not a significant one. A reason for this could be that most of the work is done in one process which takes a long time to execute their given tasks. |
| Dynamic Task Pool | 2 | 9.028125 | 2 processes execute the program dynamically with one process as the master and one as a slave. When using 2 processes there is not much sense in it, the master will delegate the work to a single process that will execute all tasks. We see that even though we have 2 processes this takes even longer than the sequential execution. One process executes similarly to the sequential program and the master has the overhead of handling the process. Therefore, the single slave in sequential execution has a better time, we save the time it takes to create the master. |
| Dynamic Task Pool | 4 | 3.032981 | A great improvement of the execution time, where each slave executes a very small task and gets a new one upon completion. This is why this is better than the 4 slaves during static execution, there might be a slave that is faster when comparing to other slaves, and his quick execution time can be utilized as opposed to the 4 tasks in static where each gets a predetermined amount of tasks. |
| Dynamic Task Pool | 20 | 1.277951 | Here we can see the true power of dynamic task pool execution where a single master oversees 20 slave processes. Each slave process executes a single task and gets a new one upon completion, this allows the fastest slave to execute more tasks. |