Reeju Bhattacherji
OpenMP Report December 11, 2023

Introduction

The purpose of this report is to compare the speed of different scheduling methods for the given code using OpenMP. The code calculates the energy of water molecules and is parallelized using OpenMP to run on 12 threads. The different scheduling methods that will be compared are ***`static`, `dynamic`, `guided`, and `auto`***, with varying **chunk sizes of 0, 1, 100, and 10000** for each method. The execution times(Wall Time) will be measured, and the speed will be compared with the MPI version and the serial version on the same machine.

Methodology

The code was parallelized using OpenMP (water1.c), and the energy calculation loop was parallelized using different scheduling methods and chunk sizes. The execution times were measured using the ***custom*** function, and the speed was compared with the MPI version (water2.c) and the serial version (water.c) on the same machine.

Results

The following table shows the execution times for each scheduling method, chunk size, and the MPI and serial versions:

Scheduling Method	**Chunk Size**	**Execution Time (sec)**	**MPI Time (sec)**	**Serial Time (sec)**
static	-	5.557778		
static	1	3.948599		
static	100	3.665986		
static	10000	7.768753		
dynamic	-	3.531517		
dynamic	1	3.684899	**6.426475**	**42.870873**
dynamic	100	3.253475		
dynamic	10000	6.646248		
guided	-	3.359954		
guided	1	3.652835		
guided	100	3.015933		
guided	10000	6.849948		
auto	-	3.462537		

Discussion

From the table, it can be observed that the `guided` and `auto` scheduling methods perform better than the `static` and `dynamic` scheduling methods. The `guided` and `auto` methods show similar execution times, indicating that they are effective in utilizing the available resources and providing good speed. The MPI version performs better than the serial version, and the difference is significant, indicating that the parallelization using OpenMP is effective in improving the performance of the code.

Conclusion

In conclusion, the `guided` and `auto` scheduling methods provide better performance than the `static` and `dynamic` scheduling methods for the given code. The MPI version performs better than the serial version, the difference is significant, indicating that the parallelization using OpenMP is effective in improving the performance of the code.