Muratcan YILDIZ 16050111036
Berkan YILDIRIM 16050111025
Gürkan ALTINTAŞ 16050111019

CENG316 Parallel Computing Project-1

Parallel Calculation of Gravitational Force Interaction of Stars

Project Summary

In this project, we implemented parallel calculation of gravitational force interaction of stars in a galaxy. We utilized multi thread parallelism via OpenMP library. We used Newton's universal law of attraction when making calculations.

Project Pseudocode

- **Sequential Algorithm Pseudocode**

```
get variables from input
Allocate stars array pointer and graviatational force array
Read_file(Argument one, Argument two,Argument tree); //Read From File
    calculate_gravitational_force(Argument one, Argument two,Argument three, Argument four); //Calculate Gravitational Force{
        Start time
        for i 0 to Count
            for k 0 to Count
                if i not equal k do
                    Calculate Distance, Calculate force between two stars, Add to total
        Finish time
}
    calculate_2Norm(Argument one, Argument two,Argument three);//Calculate 2 Norm{
        for i 0 to Count
            total+=pow(Argument one, Argument two);
}
write_file(Argument one, Argument two); //Write to file
```

- **Parallel Algorithm Pseudocode**

```
get variables from input
Allocate stars array pointer and graviatational force array
Read_file(Argument one, Argument two,Argument tree); //Read From File
    calculate_gravitational_force(Argument one, Argument two,Argument three, Argument four); //Calculate Gravitational Force{
        Start time
        # pragma omp parallel for num_threads(thread_count) // do parallel
        for i 0 to Count
            for k 0 to Count
                if i not equal k do
                    Calculate Distance, Calculate force between two stars, Add to total
        Finish time
    }
calculate_2Norm(Argument one, Argument two,Argument three);//Calculate 2 Norm{
        # pragma omp parallel  for num_threads(thread_count)
        for i 0 to Count
            # pragma omp critical
            total+=pow(Argument one, Argument two);
    }
write_file(Argument one, Argument two); //Write to file
```

Parallel Algorithm in Foster Methodology

Foster's Design Methodology has four steps which are Partitioning, Communication, Agglomeration and Mapping. If we explain Foster's Methodology in terms of these steps; firstly, we stored data in pointer array and divide them into more small equal groups to operate on them. After dividing the problem, we used OpenMP library to communication among processors. It does communication in the background via shared memory. Step of agglomeration to our project, there isn't any thread like thread 'A' must be executed before thread 'B' can be executed.Also we does not need reduce communication. Because of these reason aggregate them into a single composite thread does not make sense for us. Lastly, in mapping steps our data mapped to each threads.

We adopted **Data parallelism** because it focuses on distributing the data across different threads to operate on the data in parallel. We applied it on a pointer array because a data parallel job on an array of *n* elements can be divided equally among all the processors/threads and performed synchronous computation on data. Also, in data parallelism speedup is more as there is only one execution thread operating on all sets of data.

SCHEDULING DATAS

SEQUENTIAL TIME

Computer	**Time (Minute)**
Computer 1	1.9431
Computer 2	5.9221

OMP DEFAULT SCHEDULING

Computer 1

Thread Count	Time (Minute)	Speedup
1	1.9010	1.0224
2	1.9397	1.0015
4	1.6852	1.1530

Computer 2

Thread Count	Time (Minute)	Speedup
1	6.0499	0.9788
2	3.0992	1.9108
4	2.8171	2.1021

OMP STATIC SCHEDULING

Computer 1

schedule (static)

Thread Count	Time (Minute)	Speedup
1	1.7455	1.1132
2	1.8889	1.0286
4	1.6434	1.1823

schedule (static, 8)

Thread Count	Time (Minute)	Speedup
1	1.6524	1.1759
2	1.7974	1.0810
4	1.4938	1.3007

Computer 2

schedule (static)

Thread Count	Time (Minute)	Speedup
1	5.4121	1.0942
2	2.9464	2.0099
4	2.3737	2.4948

schedule (static, 8)

Thread Count	Time (Minute)	Speedup
1	6.0303	0.9820
2	3.1772	1.8639
4	2.5310	2.3398

OMP DYNAMIC SCHEDULING

Computer 1

schedule (dynamic)

Thread Count	Time (Minute)	Speedup
1	1.6927	1.1479
2	2.6740	0.7266
4	2.4436	0.7951

schedule (dynamic, 8)

Thread Count	Time (Minute)	Speedup
1	1.6424	1.1833
2	2.2429	0.8663
4	1.6842	1.1537

Computer 2

schedule (dynamic)

Thread Count	Time (Minute)	Speedup
1	5.6080	1.0560
2	3.1572	1.8757
4	2.5784	2.2968

schedule (dynamic, 8)

Thread Count	Time (Minute)	Speedup
1	5.3066	1.1159
2	2.7593	2.1462
4	2.1074	2.8101

OMP GUIDED SCHEDULING

Computer 1

schedule (guided)

Thread Count	Time (Minute)	Speedup
1	1.6568	1.1728
2	1.8083	1.0745
4	1.8202	1.0675

schedule (guided, 8)

Thread Count	Time (Minute)	Speedup
1	1.6853	1.1529
2	2.2429	1.0860
4	1.6842	1.3466

Computer 2

schedule (guided)

Thread Count	Time (Minute)	Speedup
1	5.2722	1.1232
2	2.6697	2.2182
4	1.9313	3.0663

schedule (guided, 8)

Thread Count	Time (Minute)	Speedup
1	5.6740	1.0437
2	2.8816	2.0551
4	2.1811	2.7151

OMP AUTO SCHEDULING

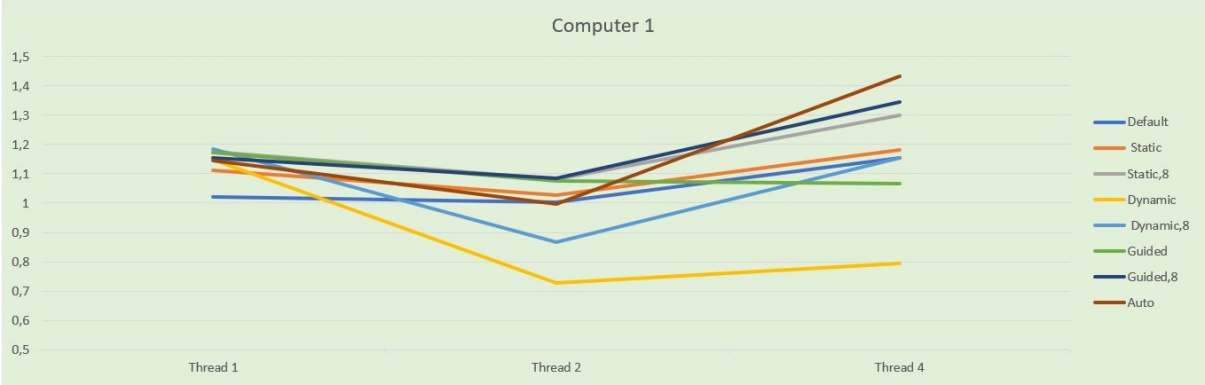
Computer 1

Thread Count	Time (Minute)	Speedup
1	1.6945	1.1467
2	1.9506	0.9961
4	1.3562	1.4327

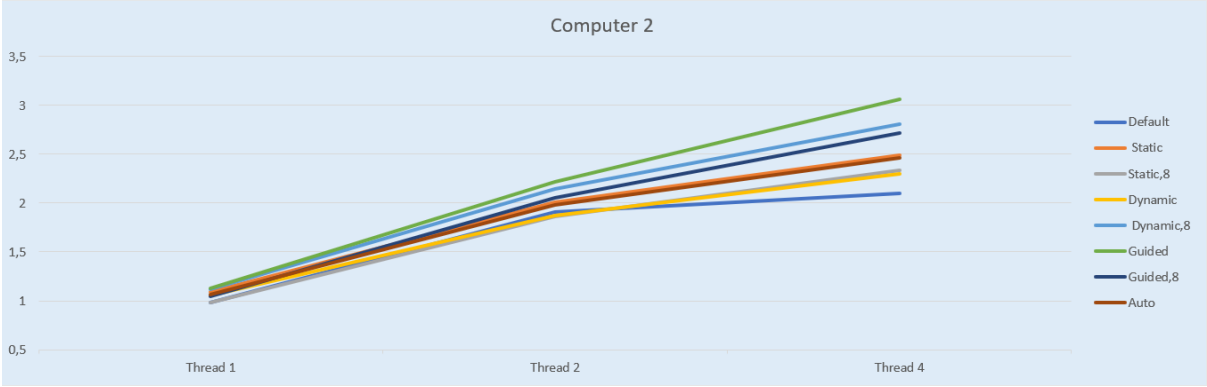
Computer 2

Thread Count	Time (Minute)	Speedup
1	5.5506	1.0669
2	2.9826	1.9855
4	2.4047	2.4627

Speedup Chart For Computer 1



Speedup Chart For Computer 2



Evaluation

In our project, we had the opportunity to test different scheduling algorithms on our computer. One of our computers was running on a very fast processor like the intel core i7 7700hq and the virtual machine while the other was running on a dualboot with a slower processor like the intel core i5 7200u. Due to these differences, computer 1 got very fast results, while computer 2 got much later results. Since Computer 1 works very fast and on the virtual machine, there was not much difference between thread 2 and thread 1. On average, a 10 second leap occurred in Threads4. As computer 2 is slow, computer speed has increased very much as the number of threads increased as seen in the graph. Although there is not much difference in Computer 1, we got the best result with Auto. In Computer 2, we got the best result with **guided schedule**. With this method, the process completion speed of the program increased by nearly 3 times. This is a really good result.