Project 2 Write Up

Andrew Johnson

CS 475

When going through this project I decided to use the Kelly ENGR lab computer terminal. For last couple projects I was using the flip server OSU offers, but when looking at its uptime when I was using it concerned me. I was right to switch to it because I had an uptime of “load average: 0.07, 0.05, 0.01”. While computing this project I was using 8 processors the whole time.

This is the table of data I gathered for Project 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| COARSE | | | | FINE | | | |
| STATIC | | DYNAMIC | | STATIC | | DYNAMIC | |
| Threads | MegaBodies | Threads | Megabodies | Threads | MegaBodies | Threads | MegaBodies |
| 16 | 66.401 | 16 | 95.308 | 16 | 3.45 | 16 | 3.376 |
| 8 | 202.909 | 8 | 181.9 | 8 | 36.901 | 8 | 20.058 |
| 4 | 202.398 | 4 | 176.59 | 4 | 52.603 | 4 | 23.126 |
| 2 | 126.611 | 2 | 131.311 | 2 | 65.479 | 2 | 26.234 |
| 1 | 85.679 | 1 | 83.526 | 1 | 66.928 | 1 | 27.722 |

This is the graph we got from the data above.

I noticed that the sweet spot when using Coarse-Grained parallelism is between 4 and 8. Then with Fine-Grained parallelism the more threads you use the worse it gets. Fine-Grained is also significantly slower than Coarse-Grained. Lastly, I can see that Static scheduling is proving faster than dynamic scheduling.

I believe that Coarse-Grained is much faster because while it goes through fine grained it has to go through an entire for loop instead of a for loop that is spilt up and computed at the same time.