A comparison of versions of the N body problem.

PDC Assignment 3 Part 2 2023

Gia Bao Hoang – a1814824

I. Objective:

The goal of this report is to present a comparative performance analysis of two parallel solvers for the N-body problem, as discussed in our lectures and textbook. We aim to replicate the results shown in Table 6.3 of the textbook using the USS system, which possesses four hardware processors. As such, we can only recreate the first three rows of the table.

II. Tabulation:

We are examining two versions of the N-body solvers: the basic version and the reduced version with default scheduling. From these, we will derive two additional variations. The first, known as "reduced forces cyclic," applies a cyclic schedule during the initial phase of force computations, while subsequent inner loops retain the default schedule. The second variation employs a cyclic schedule throughout all inner loops.

Each version requires the same set of parameters: the number of threads, the number of particles, the number of timesteps, the size of the timesteps, and the output frequency. Since our primary concern lies in the runtime of the parallel N-body solvers, we can exclude input and output data. The output frequency can be fixed at a constant value of 1 to simplify the tabulation process. It is suggested to use a timestep of 0.01 as it performs well with automatically generated data.

The program will be executed using the following command to measure the elapsed time:

```
gcc -g -Wall -fopenmp -o <program name> <program name>.c -lm -DNO_OUTPUT
./<program name> <number of threads> <number of particles> <number of
timesteps> 0.01 1 g
```

The key adjustable parameters are the number of threads, particles, and timesteps. Given the four-processor limit of the USS hardware, we will test with 1, 2, and 4 threads. The particle and timestep counts will be tested at 10, 100, 1000, and an additional high value of 10000 timesteps for extensive testing.

The tables below present the results for two sets of conditions: one for 1000 particles with 1000 timesteps, and another for 1000 particles with 10000 timesteps.

Threads	Basic	Reduced Default Sched	Reduced Forces Cyclic	Reduced All Cyclic
1	14.69	9.03	9.07	8.99
2	8.36	7.14	4.80	4.83
4	5.43	4.70	3.20	3.26

Table1: Run-Times of the n-Body Solvers with 1000 particles and 1000 timesteps (in seconds)

Threads	Basic	Reduced Default Sched	Reduced Forces Cyclic	Reduced All Cyclic
1	143.92	90.73	90.50	89.82
2	79.17	72.32	49.88	49.10
4	51.59	47.27	34.20	34.14

Table2: Run-Times of the n-Body Solvers with 1000 particles and 10000 timesteps (in seconds)

For each combination of threads, particles, and timesteps, each program was run ten times, yielding ten distinct runtimes. To limit the influence of outliers, we selected the minimum runtime from the ten trials for our calculations, rather than the median or average. Any anomalies could arise from occasional delays in the computing environment. Selecting the minimum runtime allows us to achieve a more accurate reflection of the parallel program's performance under optimal conditions.

III. Discussion:

Before delving into the data tables, we must acknowledge a few underlying assumptions. Firstly, in the reduced code, the workload of the initial force computation phase decreases with each loop iteration. We anticipate that a cyclic schedule would distribute this diminishing workload equitably among threads. On the other hand, subsequent parallel for loops require a consistent amount of work for each iteration. At first glance, these loops seem more suited to a block schedule. However, we must consider that the scheduling approach of one loop could impact the performance of others. Thus, a mix of cyclic and block schedules could potentially undermine efficiency.

A close examination of the tables reveals a consistent hierarchy in the runtimes of the different n-Body solver versions. Increasing the number of threads reliably reduces runtime across all programs, as evidenced in both tables. The basic version consistently takes the longest, followed by the reduced default scheduling version. The reduced forces cyclic and reduced all cyclic versions show the fastest runtimes, with only slight differences between them. Notably, in Table 1, the reduced forces cyclic version achieves marginally quicker runtimes than the reduced all cyclic version when utilizing 2 or 4 threads. This pattern generally persists under conditions of 1000 particles and 10000 timesteps, with minor exceptions when operating with 2 threads.

Upon ramping up the timestep count to 10000 (as in Table 2), we notice a slight shift in performance dynamics. The reduced forces cyclic version now trails behind the reduced all cyclic version, although the disparity remains minimal. This shift invites further inquiry into the performance of these programs under more substantial particle and timestep counts when run on USS hardware.

The trends observed in these tables are consistent with our initial assumptions. The reduced versions of the n-Body solvers, particularly those applying cyclic scheduling, outperform the basic version. This performance boost is largely attributable to the cyclic schedule's ability to evenly distribute workloads among threads, especially during the first phase of force computation where the workload diminishes per loop iteration.

IV. Conclusion :

In summary, under conditions of 1000 particles and 10000 timesteps, the reduced forces cyclic version of the n-Body solver offers a considerable speedup without resorting to extensive cyclic scheduling. However, as we increase the number of particles and timesteps beyond this limit, further scrutiny is warranted. It's plausible that under such high-load conditions, the reduced all cyclic version could prove to be the more efficient solver. This conjecture aligns with the notion that a cyclic schedule could be more effective when handling larger, more intricate computations. As we proceed, rigorous testing and analysis will be crucial to confirm this hypothesis.