University of South Bohemia České Budějovice

Name	**Tushar Rewatkar**
Personal Number	**B21462**
Course	**UFY/505 – Parallel Programming and Computing**
Assignment	**Parallelization of Water Energy Calculation Serial Code**

Table 1 – Reading (100k.gro)

Number of threads	Time of reading from HD (Option 1)	Time of reading from HD (Option 2)	Time of reading from HD + broadcasting data (Option 3)	Time of calculation
1-parallel	0.264948	0.262917	0.245592	32.76845333
4	0.250063	0.942844	0.233048	8.229865
8	0.246666	1.925631	0.253552	4.937179667
24 (1 node)	0.268183	6.325915	0.76159	2.008157

For the tables below, the third option of reading (one process reads and broadcast to the others) was used.

Table 2 – For the 10k.gro file

Number of threads	Total maximum wall time of a thread	Theor. time (T1/nproc)	Speed up	Overhead	Load Imbalance
1 – serial	0.362414				0
1 – parallel	0.41042	0.41042	1	0	0
2	0.21814	0.20521	1.881452278	6.300862531	0.000200
4	0.126768	0.102605	3.23756784	23.54953462	0.000600
8	0.08103	0.0513025	5.06503764	57.94551922	0.001400
24 (1 node)	0.101668	0.017100833	4.03686509	494.5207349	0.004600
24 (2 nodes)	0.144114	0.017100833	2.847884314	742.7308611	0.004600

48 (2 nodes)	0.114903	0.008550417	3.57188237	1243.829248	0.009400
72 (3 nodes)	0.156466	0.005700278	2.623061879	2644.883778	0.014201

Table 2 – For the 100k.gro file

Number of threads	Total maximum wall time of a thread	Theor. time (T1/nproc)	Speed up	Overhead	Load Imbalance
1 – serial	40.952741				0
1 – parallel	32.88941	32.88941	1	0	0
2	16.57818	16.444705	1.983897509	0.811659437	0.000020
4	8.444882	8.2223525	3.894596751	2.706396983	0.000060
8	5.183468	4.11117625	6.345058945	26.08235903	0.000140
24 (1 node)	2.810813	1.370392083	11.70103098	105.1101312	0.000460
24 (2 nodes)	2.268626	1.370392083	14.49750201	65.54576078	0.000460
48 (2 nodes)	3.284924	0.685196042	10.01222859	379.4137444	0.000940
72 (3 nodes)	2.475611	0.456797361	13.28537076	441.9494968	0.001420

Effectiveness of parallelization

From the table above, we see that increasing the number of threads in parallelization reduces the total time needed to run the program hence parallelization reduces.

Method of reading the data

The results in the table above were obtained via the third option of reading data – one thread reads and broadcasts to all other threads.

Overhead

It is defined as excessive total CPU time of parallel job with respect to CPU time of parallel job on 1 thread. In other words, it is just excessive CPU consumption with respect to job time on single thread.

As you can see from above tables (10k and 100k), the Overhead % increases as the total job time decreases with increases in the number of threads, that clearly implies that the CPU utilization is more as compared to single thread.

In some cases the value can go in negative as well, that implies that CPU utilization(Total Job time of that particular number of threads) is lesser as compared to single thread of parallel job time.

Load imbalance

From the tables above, the first observation is that the load imbalance value is the same for the same number of threads in both files (10k.gro and 100k.gro).

The load imbalance is proportional to the number of threads, it increases as the thread increases which clearly means that there is an uneven distribution of the tasks among threads (clearly seen as load imbalance values is not equal to 1).

Load imbalance is not a property of input file, it clearly depends on the number of pairs calculated by each thread.

If the load imbalance is equal to 1 then it clearly shows that task assigned among threads is most unevenly distributed. Theoretically, when minpairs is $1/4^{th}$ of maxpairs, then the load imbalance will be 1. It can also go above 1 theoretically, when minpairs is less than $1/4^{th}$ of maxpairs, then load balance will be go beyond 1.

When load imbalance is 0 it implies that tasks assigned among threads is distributed equally as you can see in table for single thread, because minpairs and maxpairs will have the same value and it will compute to 0.