Performance Evaluation of T_L511L64

Jairo Panetta

January 22, 2007

1. Purpose

On late 2006, Saulo Barros finished a new version of the global model, reaching the main 2006 objective (concerning the global model) of the NEC-CPTEC Agreement: produce an MPI with OMP version that forecasts 10 days on a Semi-Lagrangean T_L511L64 formulation over a Linear and Reduced Grid with Kuo Convection and Lacis Radiation using at most 2 hours and 30 minutes of 4 nodes of the SX-6 (32 processors), maintaining binary reproducibility.

The goal was fully achieved. Under these options and using a time step of 450 seconds, program execution takes 1 hour and 31 minutes (5499s), running at an effective speed of 83 GFlops (32.5% of the four nodes top speed), with 99.4% average vectorization ratio and average vector length of 207, requiring 49.8 GBytes of memory, using 4 MPI tasks (one at each SX-6 node) with 8 OMP threads per task.

While conducting experiments on execution time variation with OMP threads and MPI tasks, a few program characteristics draw my attention. First, memory increases with MPI tasks. Second, program does not execute on a single node since it requires more than 32 GBytes of memory (although each node has 64 GBytes, OS kernel was configured at 32 GBytes – kernel will be re-configured by NEC personal). Third, parallel speed-up seems limited, although that was a fragile conclusion due to the small number of experiments.

On January 19, 20 and 21, I conducted extensive performance experiments to obtain parallel speed-up data, after modifying source to reduce memory requirements.

This report shows code modifications and performance evaluation results.

2. Code Modifications

One reason for high memory usage was the overestimated size of arrays *g* and *per* at subroutine *nlnmi*, file *NonLinearNMI.f90*. These arrays read data from the NMI file. A tighter memory estimative was given by José Paulo Bonatti, but not precise enough for parallel runs. Since precise array sizes are scattered over the file, I end up reading the file twice, once to obtain array sizes and once to store data. Arrays were precisely allocated before the second file read.

Memory reduction was significant, allowing single node runs.

3. Experiment Setting, Reports and Main Results

Experiments were run on the Empi file, under favorable machine load. Program was lightly instrumented with FTRACE calls, dividing source in just two regions: initialization and integration, allowing separate accounting for these two execution phases.

To reduce experiment execution time, integration was performed for a single day, starting at 00Z of February 20th, 2006.

There are three sources of data for Performance Evaluation: the “Program Information” report and the two parts of the “FTRACE” report. Program Information reports on entire MPI processes, while the FTRACE reports on selected execution

phases. The first part of the FTRACE report deals with CPU time information, while the second part deals with wall clock information and MPI communication. Both reports were used.

Table 1 below shows measured wall clock time and computed speed-up for the total run and for each phase of the run. Total execution time is the maximum Real Time over all MPI processes, as reported by Program Information. Execution time of each phase was computed as the maximum value of the "Elapsed" column of the second part of the FTRACE report. Since data is taken from distinct sources, the sum of the execution times of the phases is not identical to the total execution time, but discrepancy is negligible (lower than 4 seconds).

Processors	MPI	OMP	Execution Time (s)			Speed-up		
			Total	Initialization	Integration	Total	Initialization	Integration
1	1	1	12.285,96	1.093,28	11.192,42	1,00	1,00	1,00
2	1	2	6.401,10	666,25	5.733,19	1,92	1,64	1,95
3	1	3	4.401,88	517,87	3.880,98	2,79	2,11	2,88
4	1	4	3.415,22	445,80	2.966,13	3,60	2,45	3,77
5	1	5	2.816,03	440,77	2.374,85	4,36	2,48	4,71
6	1	6	2.421,56	370,75	2.048,58	5,07	2,95	5,46
8	1	8	1.946,12	332,29	1.611,25	6,31	3,29	6,95
14	2	7	1.264,10	337,20	924,12	9,72	3,24	12,11
16	2	8	1.218,78	341,25	874,40	10,08	3,20	12,80
21	3	7	1.006,02	351,69	653,01	12,21	3,11	17,14
24	3	8	953,32	350,94	600,93	12,89	3,12	18,63
28	4	7	896,28	364,89	529,51	13,71	3,00	21,14
32	4	8	858,72	384,47	471,08	14,31	2,84	23,76

Table 1: Wall Clock Execution Time and Speed-up

Data shows that:

- Initialization time does not scale well with increasing processor count;
- Integration time scales reasonably within a node (OMP), but requires some work;
- The combined OMP-MPI integration scaling is reasonable, but requires some work.

Table 2 below shows memory requirements of each run, in MBytes. Data for each node is extracted from the "Memory Size" row of "Program Information" reports. Total is the sum over all nodes.

Processors	MPI	OMP	Total	Node 1	Node 2	Node 3	Node 4
1	1	1	27.296	27.296			
2	1	2	27.584	27.584			
3	1	3	27.600	27.600			
4	1	4	27.744	27.744			
5	1	5	28.016	28.016			
6	1	6	28.160	28.160			
8	1	8	28.448	28.448			
14	2	7	30.208	15.136	15.072		
16	2	8	30.496	15.280	15.216		
21	3	7	32.448	11.408	9.696	11.344	
24	3	8	32.880	11.552	9.840	11.488	
28	4	7	34.672	9.456	7.904	7.904	9.408
32	4	8	35.216	9.568	8.048	8.048	9.552

Table 2: Memory Requirements (MBytes)

Data shows that:

- Memory increases at about 160 Mbytes per OMP thread;
- Memory increases at about 1GByte per MPI process in excess of the OMP increase.

We proceed by pointing out a few details.

4. Initialization

Table 3 below reports execution times (wall clock) of each thread during the initialization phase, for executions under a single MPI process. Data is taken from the "Elapsed" column (first column) of the second part of the FTRACE report.

OMP	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8
1	1.093,28							
2	666,25	47,82						
3	517,87	32,54	32,54					
4	445,80	24,21	24,21	24,21				
5	440,77	19,71	19,71	19,71	19,71			
6	370,75	16,79	16,79	16,79	16,79	16,79		
8	332,29	13,31	13,31	13,31	13,31	13,31	13,31	13,31

Table 3: Thread Wall Clock Execution Time during Initialization

Data shows very poor OMP parallelism. Maybe some OMP directive is missing in the control path during initialization. But data itself is inconsistent, since the sum of execution times of all threads decreases as OMP thread count increases.

Further investigation is required.

5. OMP Parallelism on Integration

Table 4 below reports execution times (wall clock) of each thread during the integration phase, for executions under a single MPI process. Data is taken from the "Elapsed" column (first column) of the second part of the FTRACE report.

OMP	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8	Speed-up
1	11.192,42								1,00
2	5.733,20	5.733,20							1,95
3	3.880,98	3.880,98	3.880,98						2,88
4	2.966,14	2.966,14	2.966,14	2.966,14					3,77
5	2.374,85	2.374,85	2.374,85	2.374,85	2.374,85				4,71
6	2.048,59	2.048,59	2.048,59	2.048,59	2.048,59	2.048,59			5,46
8	1.611,26	1.611,26	1.611,26	1.611,26	1.611,26	1.611,26	1.611,26	1.611,26	6,95

Table 4: Thread wall clock execution time during integration

Data shows no variation of wall clock among threads of each run. But since speed-up is not perfect, further investigation is required. We examine CPU time during integration.

OMP	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8	Sum
1	11.096,86								11.096,86
2	5.595,24	5.595,29							11.190,53
3	3.785,55	3.780,31	3.784,15						11.350,01
4	2.889,64	2.885,15	2.885,22	2.885,04					11.545,06
5	2.343,20	2.349,48	2.342,59	2.342,80	2.342,54				11.720,61
6	1.986,99	1.992,48	1.986,90	1.987,66	1.987,94	1.990,52			11.932,49
8	1.553,50	1.553,59	1.553,37	1.553,37	1.553,27	1.558,89	1.555,76	1.558,06	12.439,81

Table 5: Thread cpu time during integration

Table 5 above contains the CPU time of each thread during integration. Data is taken from the first part of the FTRACE report. Last column shows the sum of CPU time over all threads of a run.

Again, CPU time variation among threads is negligible. Either load is fully balanced among threads or load is unbalanced and threads busy wait for the other threads.

But the sum of CPU times over all threads steadily increases with thread count. While this is consistent with loosing speed-up, it does not expose its cause. One possible cause is a repeated computation (a sequential computation) over all threads.

Table 6 below shows the amount of computing per thread, in GFLOP. Data is obtained by multiplying column "Exclusive Time" by column "MFLOPS" of the first part of the FTRACE report. Consequently, it is approximated data. Last column is the sum over all threads. Variation of the sum among threads is negligible, indicating that if there is a repeated computation, it is negligible.

OMP	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8	Sum
1	41.372,42								41.372,42
2	20.674,96	20.696,99							41.371,95
3	13.741,54	13.820,44	13.809,87						41.371,84
4	10.299,54	10.358,86	10.357,65	10.355,86					41.371,91
5	8.230,97	8.292,48	8.286,67	8.282,04	8.279,94				41.372,10
6	6.861,67	6.904,75	6.904,09	6.899,97	6.896,35	6.905,12			41.371,95
8	5.147,21	5.168,94	5.172,88	5.172,88	5.173,31	5.180,98	5.172,12	5.183,37	41.371,69

Table 6: GFLOP per thread during integration

Further investigation is required. Current available data rules out repeated computations, but does not indicate the source of parallelism leak.

6. MPI Parallelism on Integration

Table 7 below summarizes communication and computation wall clock time during the integration phase on MPI runs with 8 OMP threads each. Data is taken from the second part of the FTRACE report.

Data shows that MPI processes wait idle on communication. It also shows that communication time, number of messages and total amount of data transfer varies among processes of a single MPI run. These are typical of load unbalancing.

MPI	ELAPSE [sec]	COMM.TIME [sec]	COMM.TIME / ELAPSE	IDLE TIME [sec]	IDLE TIME / ELAPSE	AVER.LEN [MByte]	COUNT	TOTAL LEN [GByte]
2	874,37	76,70	0,09	11,44	0,01	26,30	13235	340,20
2	874,40	69,52	0,08	12,73	0,02	26,30	13235	340,20
3	600,94	67,27	0,11	14,42	0,02	21,80	14797	314,90
3	600,94	76,61	0,13	20,07	0,03	11,10	26223	283,00
3	600,94	71,45	0,12	15,63	0,03	21,80	14763	314,40
4	471,08	56,85	0,12	10,37	0,02	17,60	16359	281,70
4	471,07	70,20	0,15	21,34	0,05	8,60	27751	234,40
4	471,05	68,48	0,15	21,38	0,05	8,60	27751	234,40
4	471,05	55,38	0,12	10,58	0,02	17,70	16291	281,40

Table 7: Computation and Communication during integration

7. Conclusions

Memory increases with processor count. The increase is more stringent on MPI processes than on OMP threads.

Parallelism of the initialization phase requires attention.

Parallelism of the integration phase requires further analysis, mainly on OMP threads. MPI data indicates a load unbalancing problem.

The impact of these imperfections on execution time is presented at Figure 1 below, that shows how the parallel efficiency (% of perfect parallelism) of the integration phase varies with processor count.

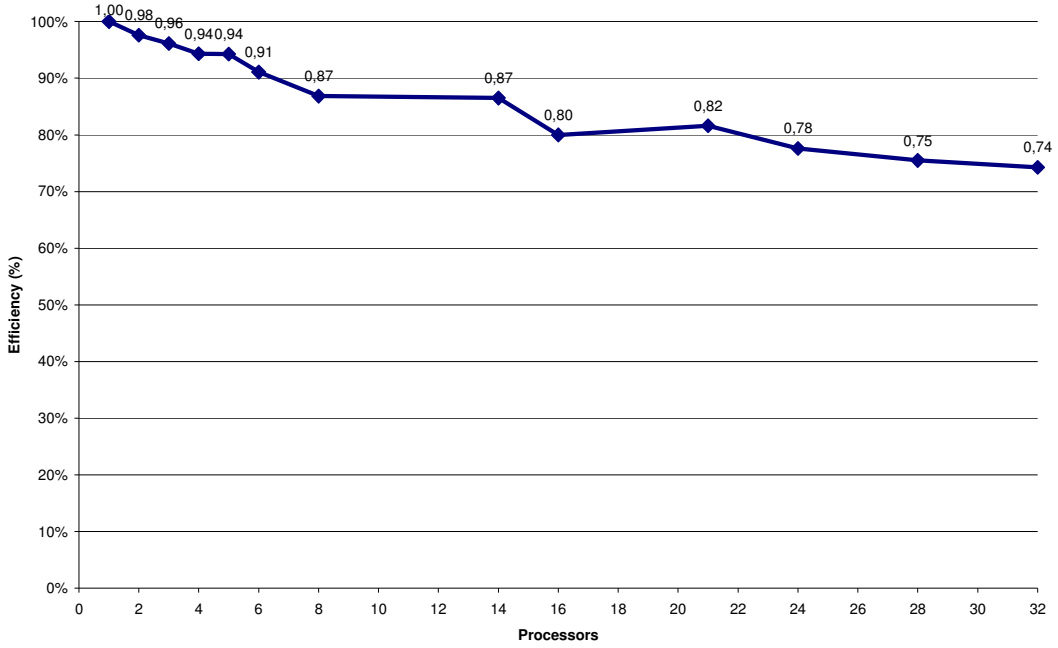

Figure 1: Parallel Efficiency of the Integration Phase

Figure shows the damage of combining small scale imperfections on parallelism. Their constructive interference reduces parallelism gain very fast.

8. Sources

For the 10 days run, source at the SX-6 /gfs/dk15/panetta/gcm/SLSaulo/glinear. Execution at subdirectory run, results at subdirectory ResNoGrellNoClirad. On the lap top, /home/panetta/CPTEC/MPIOMP/ExperimentosJan2007/ResTL511L64Reduzida, source at subdirectory Src, results at subdirectory Results.

For the single day run, see the SX-6 /gfs/dk15/panetta/gcm/SLSaulo/SpeedUp directory. Source at subdirectory Src, results at Results, executions by ZZmit.sh of subdirectory Run. At the laptop, see all subdirectories under the directory /home/panetta/CPTEC/MPIOMP/ExperimentosJan2007/ResTL511L64UmDia.

This report is at subdirectory Results of the lap top directory described above.