Investigation of Load Balancing Scalability in Space Plasma Simulations

Ata Turk[1], Gunduz V. Demirci[1], Cevdet Aykanat[1], Sebastian von Alfthan[2], and Ilja Honkonen[2,3]

[1] Bilkent University, Computer Engineering Department, 06800 Ankara, Turkey
{atat,gunduz,aykanat}@cs.bilkent.edu.tr
[2] Finnish Meteorological Institute, PO Box 503, FI-00101, Helsinki, Finland
{Sebastian.von.Alfthan,ilja.honkonen}@fmi.fi
[3] Department of Physics, University of Helsinki, PO Box 64, 00014, Helsinki, Finland

Abstract. In this study we report the load-balancing performance issues that are observed during the petascaling of a space plasma simulation code developed at the Finnish Meteorological Institute (FMI). The code models the communication pattern as a hypergraph, and partitions the computational grid using the parallel hypergraph partitioning scheme (PHG) of the Zoltan partitioning framework. The result of partitioning determines the distribution of grid cells to processors. It is observed that the initial partitioning and data distribution phases take a substantial percentage of the overall computation time. Alternative (graph-partitioning-based) schemes that provide better balance are investigated. Comparisons in terms of effect on running time and load-balancing quality are presented. Test results on Juelich BlueGene/P cluster are reported.

Keywords: partitioning, petascaling, space plasma simulation.

1 Introduction

The dynamics of near Earth space environment have gained immense importance since many mission critical global technological systems depend on spacecraft that traverse this space and even small dynamical events can cause failures on the functionalities of these spacecraft. Hence performing accurate space weather forecasts are of utmost importance. Space weather forecasting is performed by modeling the electromagnetic plasma system within the near Earth space including the ionosphere, magnetosphere, and beyond.

At the Finnish Meteorological Institute (FMI), two simulation models are being developed to tackle this issue: a magnetohydrodynamic simulation code for real-time forecasting and a hybrid Vlasov simulation code for very accurate space weather forecasting. In a hybrid Vlasov model, electrons are modeled as a fluid and ions as six-dimensional distribution functions in ordinary and velocity space, enabling the description of plasma without noise.

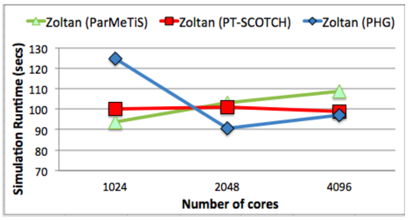
Both codes need to exhibit excellent parallel scalability to reach the required level of performance. The simulation models are designed to run on a parallel grid. In the hybrid-Vlasov code, the parallel grid contains cells in ordinary space, and each spatial grid cell contains a three-dimensional velocity distribution function, which is implemented as a simple block-structured grid. A major bottleneck for this code is the need for efficient load balancing at scale, as the target is to run it on more than 10.000 cores. For determining the distribution of spatial cells to processors, the grid uses the PHG partitioning mode of the Zoltan partitioning framework.

2 Investigations in Jugene

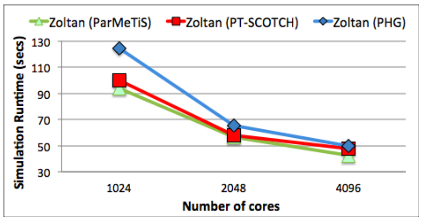
In this study the effects of load balancing tools in the performance of hybrid Vlasov simulation code have been analyzed in detail. The scalability of the code itself is also tested to some extent. The reported findings can be listed as follows:

- Porting of the hybrid Vlasov code to Juelich BlueGene/P (Jugene) system is performed and profiling of the performance of the code up to 10^4 cores (previous tests were performed for less than 10^3 cores due to limited resources) is achieved to reveal that it successfully scales up to 10^4 cores.
- Analysis of the load-balancing (partitioning) scheme is performed. It is observed that the time spent on preprocessing constitutes a significant portion of the overall runtime. Further analysis revealed that when the number of cores reach to 10^4, the determinant factor in simulation runtime tends to be the balancing performance instead of the overall communication cost.
- An alternative load-balancing scheme (based on graph-partitioning), which is known to have better load balancing performance, is embedded in the code. Experiments show that the alternative scheme has simulation time performances that are more scalable than PHG.

We should note here that graph partitioning models cannot exactly model the communication overheads associated with the communication patterns in the hybrid Vlasov code and the usage of the hypergraph modeling scheme is more correct theoretically. However, as a general observation we can state that, although the communication metrics optimized by graph partitioning schemes are not exact, if the problem domain is regular enough, the error made by graph partitioning method for estimating the communication overhead of a partition is more or less the same for all possible partitions in the solutions space. This property enables the graph partitioning schemes to improve its solutions over regular computational domains successfully since the error made while moving through different partitions in the solution space cancel each other. Since the subject problem domain exhibits such features, we believe that the usage of graph partitioning tools might yield good results as well and thus investigate such alternatives.



(a) Weak Scaling



(b) Strong Scaling

Fig. 1. Simulation runtimes of the weak and strong scaling experiments for hybrid Vlasov code utilizing different partitioning libraries available in Zoltan

Table 1. Communication overheads and balancing performances under weak scaling experiments for various parallel partitioning libraries that are called within Zoltan

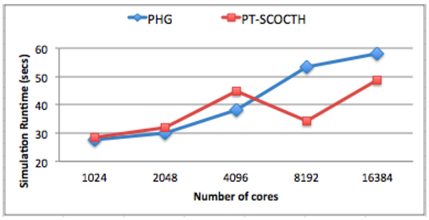
	Zoltan (PHG)			Zoltan (ParMeTiS)			Zoltan (PT-SCOTCH)		
# of cores	Total volume	Comp. imb.(%)	Comm. imb.(%)	Total volume	Comp. imb.(%)	Comm. imb.(%)	Total volume	Comp. imb.(%)	Comm. imb.(%)
1024	70.206	99,9	44,1	78.538	50,0	34,2	77.824	0,1	5,3
2048	143.538	99,9	42,9	161.318	31,3	41,0	159.744	0,1	2,6
4096	289.292	99,9	60,0	327.812	62,5	40,0	323.568	6,2	12,8

3 Experiments

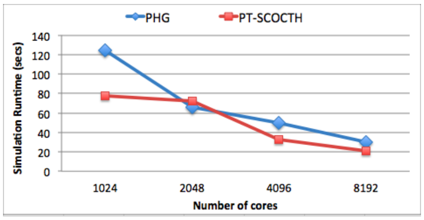
In our experiments, we first compared the performance of calling Zoltan PHG [1] with the performance of calling ParMeTiS [2] and PT-SCOTCH [3] within Zoltan in the initial load balancing step of the simulation code to see which of these three possible partitioning options available in Zoltan is best for this particular code. In these experiments we measured the weak and strong scaling performance of these three schemes. In the experiments for weak scaling, 3D grid size is arranged such that under perfect load balance, each process would have to process 16 spatial cells, and for strong scaling, total number of spatial cells is set to $16 \times 32 \times 32$. Since the memory in a Jugene node is small, we could only perform weak-scaling experiments up-to 4K cores with 16 spatial cells per core.

In Table 1, we present the total communication volume, computational imbalance, and communication imbalance values observed in the weak scaling experiments. As seen in the table, in terms of total communication volume, PHG performs the best, whereas in terms of communication and computation load balancing, PT-SCOTCH performs the best. We should note here that strong scaling experiments provided similar communication and computation balancing results but are not reported due to space constraints.

As seen in Table 1 and Fig. 1, even though PHG produces lowest overall communication overheads, the graph-based partitioning libraries produce as good as, if not better, running time results. This is probably due to PHG's poor load-balancing performance. After these observations we decided to remove the



(a) Weak scaling



(b) Strong scaling

Fig. 2. Weak scaling simulation runtimes of hybrid Vlasov code using PT-SCOTCH and Zoltan PHG in preprocessing.

overhead of calling Zoltan for running the PT-SCOTCH library, which has nice load-balancing features and rewrote the initial load balancing code such that it calls PT-SCOTCH library directly.

Table 2. Communication overheads and balancing performances under weak scaling experiments for Zoltan PHG and PT-SCOTCH

	Zoltan (PHG)			PT-SCOTCH		
# of cores	Total volume	Comp. imb.(%)	Comm. imb.(%)	Total volume	Comp. imb.(%)	Comm. imb.(%)
1024	27.726	99,6	76,9	28.672	0,9	14,4
2048	59.608	99,8	92,9	61.440	0,9	6,7
4096	123.294	99,9	73,3	126.976	1,5	6,7
8192	250.718	99,9	26.1	258.048	1,2	6,7
16384	505.008	99,9	93,3	520.192	1,4	6,7

In Fig. 2 we compare the performance results of directly calling PT-SCOTCH and Zoltan PHG in the preprocessing step of simulation. In the weak-scaling experiments reported, the number of grid cells per core is fixed to four and in the strong-scaling experiments the total number of spatial cells is set to $16 \times 32 \times 32$. As seen in Fig. 2(a), as the number of cores reaches to 8K and beyond, PT-SCOTCH starts to perform considerably better then PHG, probably again due to its better load-balancing capability as noted in Table 2. We again note here that strong scaling experiments provided similar communication and computation balancing results but are not reported due to space constraints. Similarly, as seen in Fig. 2(b), PT-SCOTCH generally produces better results in strong-scaling experiments as well.

4 Conclusions

We showed that the hybrid Vlasov simulation code developed at FMI can scale up to 16K cores. We also showed that by replacing the initial load balancing scheme based on Zoltan parallel hypergraph partitioning tool with PT-SCOTCH parallel graph partitioning tool increases the overall communication volume but still improves the simulation runtime since PT-SCOTCH produces better balanced partitions. These results indicate that for the hybrid Vlasov code, minimizing imbalance is as important as, if not more important than, minimizing the overall communication volume.

References

1. Devine, K., Boman, E., Heaphy, R., Hendrickson, B., Vaughan, C.: Zoltan Data Management Services for Parallel Dynamic Applications. Computing in Science and Engineering 4(2), 90–97 (2002)
2. Schloegel, K., Karypis, G., Kumar, V.: Parallel static and dynamic multi-constraint graph partitioning. Concurrency and Computation: Practice and Experience 14(3), 219–240 (2002)
3. Chevalier, C., Pellegrini, F.: PT-Scotch: A tool for efficient parallel graph ordering. Parallel Computing 34(6-8), 318–331 (2008)