Evaluation of Dynamic Scheduling Methods in Simulations of Storm-time Ion Acceleration

Ioannis Riakiotakis, Georgios Goumas, Nectarios Koziris
National Technical University of Athens
School of Electrical and Computer Engineering
Computing Systems Laboratory
{iriak, goumas, nkoziris}@cslab.ece.ntua.gr

Fiori-Anastasia Metallinou and Ioannis A. Daglis
National Observatory of Athens
Institute for Space Applications and Remote Sensing
{ametal, daglis}@space.noa.gr

Abstract

In this paper we investigate the applicability of classic dynamic loop scheduling methods on a numerical simulation code that calculates the trajectories of charged particles in the earth's magnetosphere. The numerical application under consideration investigates the influence of substorm-induced electric fields that cause magnetospheric disturbances, responsible for severe hazards in human activities and technology infrastructures in the near-earth space environment. The computational time to simulate the motion of each particle is dependent on the inital conditions applied and may greatly vary between different particles. This fact leads to great load imbalances in parallel execution scenarios and thus to degraded overall performance. For this reason we apply dynamic scheduling techniques to load-balance the tasks in homogeneous, heterogeneous and loaded distributed-memory parallel platforms and select the most appropriate among the available strategies.

1. Introduction

This work evaluates the use of dynamic scheduling methods in improving the performance of a numerical simulation code for storm-time ion acceleration in the terrestrial magnetosphere. The scientific problem investigated is the influence of substorm-induced electric fields on the storm-time ring current development. The ultimate goal is to clarify the storm-substorm relationship and consequently contribute to the understanding of Sun-Earth connection. Any sudden change in the Earth's magnetic field is defined as geomagnetic activity. Such perturbations are caused by electric currents flowing in the Earth's magnetosphere and ionosphere, which are driven by the solar wind. The main types of geomagnetic activity in the near-Earth space are geospace *magnetic storms* and *magnetospheric substorms*. Magnetic storms cause a reduction of the strength of the horizontal component of the geomagnetic field, they are observed globally at mid-latitudes and last for a few days. During a magnetic storm, the large-scale magnetospheric convection is responsible for the transport and acceleration of energetic ions into the inner magnetosphere. During intense magnetic storms, a succession of substorms tends to occur. Magnetospheric substorms are short intervals of intense magnetic disturbance, lasting for a few hours. A specific sequence of events, such as auroral arc brightening and auroral electrojet development, occur in the magnetosphere and ionosphere during a substorm. The storm-substorm relationship still remains an open question in magnetospheric physics. Clarification of this problem contributes to the understanding of Sun-Earth connection. Our aim is to contribute to the above problem by using a numerical simulation code for storm-time ion acceleration in the terrestrial magnetosphere.

It has been observed that the geomagnetic field disturbances drive the Space Weather and are able to affect human activities and technology infrastructures in the near-earth space environment, the ionosphere and on ground. Important impacts of Space Weather may be summarized as follows: Spacecraft charging, penetration of charged particles to satellite electronics, astronaut health hazards, dis-

This work has been supported by the Greek Secretariat for Research & Technology and the European Commission through the Operational Programme "Information Society".

ruption of communications, radio black-outs, GPS signal scintillation. Because of the flow of the Geomagnetically-Induced-Currents (GICs) to the ground, there are reports on damages on pipelines, increased heating and burning-out of transformers, electric power problems and electrical black-outs.

Using the particle code of D. Delcourt [6] we are able to calculate trajectories of charged particles moving under the scenario of a storm-time substorm occurrence. We simulate the reconfiguration of the geomagnetic field during a substorm, while storm-time conditions are set to the magnetosphere. We choose the ion species we want to trace and after setting its initial conditions (energy, coordinates) we follow its transport and energization in the magnetosphere. The exact trajectory of the particle depends on the initial conditions and the time instant at which its motion starts under a specific magnetic field configuration. Depending on the above parameters, when we trace a large number of ions their trajectories may vary dramatically. Some of them may terminate their motion earlier than others, by reaching the magnetopause, while others may drift around the Earth in the inner magnetosphere and contribute to the ring current population. As a result, when running the code for a large number of particles we face the problem of different calculation volume per particle.

To deal with the above problem we apply classic dynamic loop scheduling schemes in order to efficiently load balance the computation tasks. Dynamic scheduling algorithms [8] attempt to use runtime information of the system in order to make informative decisions for balancing the workload. This makes them applicable to a large spectrum of applications. In [7] different dynamic load balancing algorithms with different complexities were compared. An important class of dynamic scheduling algorithms is the self-scheduling class: Chunk Self-Scheduling (CSS) [12], Guided Self-Scheduling (GSS) [14], Trapezoid Self-Scheduling (TSS) [18], Factoring Self-Scheduling (FSS) [15]. These algorithms were devised for scheduling parallel tasks, i.e. loops without dependencies, executed on homogenous systems. Self-scheduling algorithms divide the total number of tasks into chunks, which are then assigned dynamically to the processing nodes. Also some variations of the above algorithms were presented that improve their performance when applied to heterogeneous systems like Weighted Factoring (WF) and [9] and Distributed trapezoid-scheduling (DTSS) [3],or that use runtime statistics to adapted to highly irregular workloads like adaprive factoring (AF) [2] and adapted weighted factoring (AWF) [10].

We experimentally evaluate the efficiency of the above scheduling schemes for various particle numbers on a medium-scale distributed-memory machine. We take into consideration the scenarios of homogeneous, heterogeneous and loaded computing nodes. Our experimental results show that dynamic scheduling techniques can significantly improve performance especially in the cases when cpu or system heterogeneity is added to the application's imbalanced nature.

The rest of the paper is organized as follows: Section 2 presents information concerning the simulated phenomena and the relevant simulation code, Section 3 provides a brief presentation of dynamic scheduling methods and Section 4 presents experimental results on the efficiency of the above methods on the parallel execution of the aforementioned numerical application. Finally, Section 5 summarizes our work and discusses directions for future work.

2 A model for storm-time substorms

In this section we provide information concerning the models used for storm-time substorm simulations and the characteristics of the corresponding numerical code.

2.1 Physics of the problem

Figure 1 schematically describes the evolution of the magnetosphere during a substorm. Magnetospheric substorms are more localized phenomena since their effects are observed at high latitudes. On the other hand magnetic storms are more global phenomena observed at mid-latitudes also. In Figure 1 we have plotted two geomagnetic indices SYM-H and AL versus time, which represent the magnetic storm and substorm activity respectively. Geomagnetic indices measure components of the geomagnetic field, as measured by the magnetometers (instruments located on the ground). Figure 1 shows that while SYM-H index decreases (this indicates magnetic storm activity), sudden decreases of the AL index also occur. Decreases of AL index represent substorm activity.

The two main drivers responsible for the energization and transport of particles from the magnetotail to the near-Earth region are the convective drift and the inductive drift [13]. The convective drift is imposed by the large scale convection electric field in the nightside magnetosphere. As the solar wind flows towards the earth, the solar wind and the earth's magnetic field create a natural generator. The flow of the magnetized solar wind with a velocity represents an electric field, referred as *convection electric field*:

$$\mathbf{E} = -\mathbf{v}_{sw} \times \mathbf{B}$$

This convection electric field cannot penetrate the magnetosphere, since the solar wind cannot penetrate the magnetopause. But during periods that the Interplanetary Magnetic Field (IMF) has a southward component, the northward oriented terrestrial field lines reconnect with the IMF

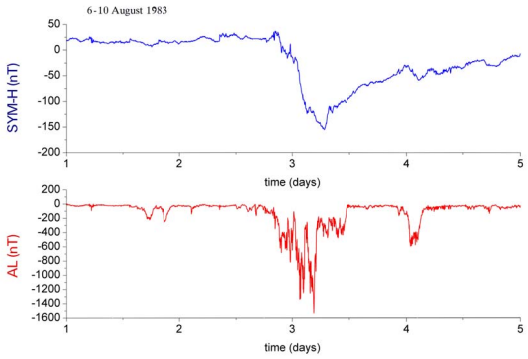

Figure 1. The evolution of SYM-H and AL geomagnetic indices during a period of magnetic storm activity.

field lines at the dayside magnetopause. As a result, the field lines and the plasma drift as a whole, setting on a large-scale circulation inside the magnetosphere, known as convection (see Figure 2). According to the Lorentz law, the equation of motion of a particle of charge q and velocity v, under the influence of an electric field E and a magnetic induction B, is given by:

$$m\frac{dv}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

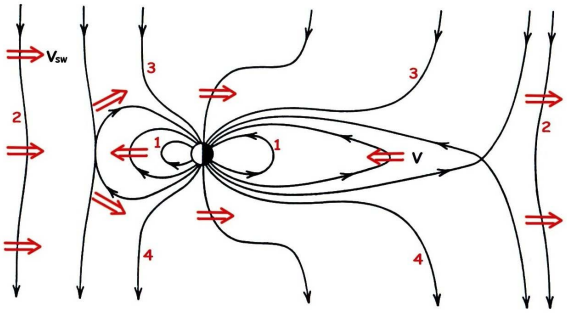

Figure 2. A pattern of the large-scale convection process as spread across the entire width of the magnetosphere.

The vertical component of the velocity, which is usually referred as E-cross-B drift, equals to:

$$v_{\perp} = \frac{\mathbf{E} \times \mathbf{B}}{B^2}$$

$\mathbf{E} \times \mathbf{B}$ is perpendicular to both the electric and magnetic fields and it is the basic transport and acceleration process for ions moving from the magnetotail plasma sheet to the inner magnetosphere.

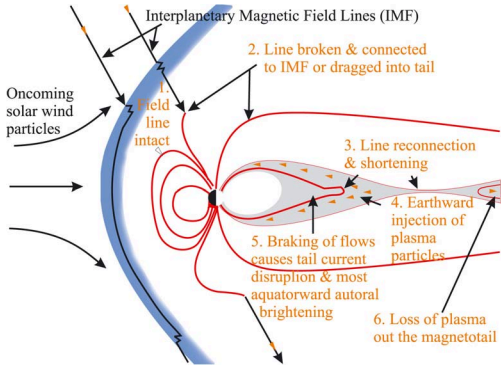

Figure 3. During substorms the reconfiguration of the geomagnetic field, induces electric fields in the magnetotail, which are responsible for plasma injections to the inner magnetosphere.

During magnetospheric substorms, the time variations in the *geomagnetic field*, as the magnetotail collapses, give rise to intense electric fields that drive the inductive drift. When a particle experiences time-varying magnetic fields, it is subjected to electrical forces, according to the *Faraday's law*:

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

The *substorm induced electric fields* accelerate particles and inject them to the inner magnetosphere where they become trapped. They are localized to the midnight-dusk sector and play an important role in the storm-time ring current formation [4] (see Figure 3).

Our study aims at quantifying the influence of substorm-induced electric fields on the build-up of particle radiation during geospace magnetic storms. In order to investigate the relative influence of magnetospheric convection and substorm injections on ring current development, we need to examine temporal and spatial variations of ion energy densities in the inner magnetosphere during storms both with and without substorm occurrence. For this reason we need to model a substorm event under storm-time conditions. The geomagnetic field used is simulated through the *Tsyganenko model* [17], which gives a description of the average magnetic field configuration for 6 different levels of geomagnetic activity. It includes contributions from external sources such as the ring current, the magnetotail current system, the magnetopause currents and the large-scale system of field-aligned currents. If $\mathbf{B_1}$ and $\mathbf{B_2}$ correspond to initial and final configurations of the geomagnetic field, according to the Tsyganenko model, the magnetic field at

position r and time t is given by:

$$\mathbf{B}(\mathbf{r},t) = \mathbf{B_1}(\mathbf{r}) + f(t)(\mathbf{B_2}(\mathbf{r}) - \mathbf{B_1}(\mathbf{r}))$$

where $f(t)$ is a polynomial of degree 5, that smoothly varies between 0 at $t = 0$ and 1 at $t = \tau$, where τ is the time scale of the magnetic transition [5]. The large-scale steady convection electric field in the magnetosphere is calculated by the *Volland-Stern model* [19, 16]. It has been arranged to fit most general features of electric fields observed by polar orbiting satellites. A basic free parameter of the Volland-Stern model is the cross polar-cap potential drop (Φ_{pc}) which characterizes the strength of the convection and as a result the intensity of the magnetic storm activity. It is calculated from the electric fields measured by polar-orbiting spacecraft on a pass over the high-latitude ionosphere. It typically ranges from 20kV, during geomagnetically quite periods, to 150 kV, during highly disturbed conditions.

The electric field induced by a transition of the geomagnetic field, from an initial level to a final one, more or less disturbed, is derived by the vector potential technique of Delcourt [6]. The 3D particle code of *Delcourt* computes the guiding center (GC) equation in the near-Earth region and the full equation of the particle motion (FP) at distances larger than 3RE.

We model a storm-time substorm according to the following pattern:

1. We primarily set convection conditions in the magnetosphere by the cross-polar cap potential drop. pc is set to the value of 40 kV and at some time before the onset of the substorm it changes to the value of 80kV, and remains on that level for about one hour.

2. During the growth phase of a magnetospheric substorm the magnetospheric magnetic field is stretched to a less dipolar configuration, forming an elongated magnetotail (e.g. [1]). The growth phase of a substorm can be simulated by the evolution of a given magnetic field level-a, which is represented by a given value of Kp geomagnetic index, to a more disturbed one, level-b (a higher value of Kp). The time scale of the magnetic transition is of the order of 30 minutes, which is regarded to be a typical value for a substorm growth phase [11].

3. As the expansion phase of a substorm is characterized by a collapse of the geomagnetic tail, this dipolarization can be modeled by an evolution of the magnetic field level-b to the initial level-a disturbance. The time scale of this second transition is in the order of a few minutes.

4. The magnetic field remains at the ground level for the next few hours. Under the above sequence of transitions the magnetic field is calculated by the Tsyganenko model.

2.2 Numerical code sketch

The numerical code that simulates the motion of a single particle in the magnetosphere for a specific time window of K time steps performs the three following major operations:

1. *Initialization*
   a. Set storm-time conditions to the magnetosphere (polar cap potential drop)
   b. Set field transitions (levels of geomagnetic activity, duration of the field transitions)
   c. Set the particles initial conditions (position, energy, pitch angle, species of ions, time at which it starts its trajectory during the evolution of the magnetic field).
   d. Set observation window size to K and current time step $k = 0$.

2. *Calculation of particle position*
   a. Use the Tsyganenko model to calculate the *magnetic field*.
   b. Use the Volland-Stern model to calculate the *convection electric field*.
   c. Use the vector potential technique of Delcourt to calculate the *induced electric field*.
   d. Use the Lorentz law to calculate the new position of the particle.
   e. k=k+1

3. *End of motion control*
   a. If particle is still in motion, and $k < K$ go to step 2.

Clearly step 3 in the above sketch clarifies the great differences that may occur in the computation time of various particles. There exist cases where a particle may end its motion after a couple of steps (e.g. by colliding with the earth's surface, or reaching the boundary of the magnetopause) or cases where the particle may remain in motion during all the observed time window and can reach up to several hundreds of time steps. The trajectory of each particle depends strongly on its initial conditions and the exact time at which it starts its motion, during the evolution of the magnetic field we need to simulate. In this code interactions between particles are not taken into account, so a particles' trajectory does not depend on other particles. In order to produce a more representative and realistic view of the ring current, a large number of particles, is required. The initial conditions for the particles used are taken from a range of values that describe the plasma population we are interested in examining. It is obvious that the more particles used for each simulation, the more computational time is needed. The investigation of transport and the mechanism of energization for

a wider range of initial conditions (such as energies, pitch angles, and coordinates) is an important issue of our study, which also depends strongly on the available computational power.

3 Dynamic scheduling methods

We give below an overview of the most well-known self-scheduling schemes (SS, CSS, TSS, FSS and GSS), employed to speedup the execution process of a storm-time substorm model. Self-scheduling algorithms work by partitioning iteration spaces into *chunks*, thus creating a pool of tasks which are dynamically assigned to processing nodes upon request. In our case, each iteration in this iteration space is a new independent execution of the simulation code, using different initial conditions.

The simplest self-scheduling algorithm, called *pure Self-Scheduling* (SS for short) assigns just one iteration to each worker per request. This algorithm achieves almost perfect load balance. All workers are expected to finish at nearly the same time, with maximum difference of one iteration execution time. On the other hand, SS can suffer from excessive scheduling overheads (T_{sched}), since the allocation of tasks to workers is done in M scheduling steps, where M is the total number of tasks.

Chunk Self-Scheduling (CSS) [12] assigns constant size chunks to each worker, i.e., $C_i = constant$ and $1 \leq C_i \leq \frac{M}{P}$, where P is the number of workers. The chunk size is usually chosen by the user. If $C_i = 1$ then CSS is identical to (pure) Self-Scheduling. On the other hand, if $C_i = \frac{M}{P}$ the CSS is identical to static scheduling. A large chunk size reduces scheduling overhead, but at the same time increases the chance of load imbalance. As a compromise between load imbalance and scheduling overhead, other schemes start with large chunks to reduce the scheduling overhead, which are gradually reduced in size throughout the execution to improve load balancing. These schemes are known as reducing chunk size algorithms.

In *Guided Self-Scheduling* (GSS) [14], each worker is assigned a chunk given by the number of remaining iterations divided by the number of workers P, i.e., $C_i = R_i/P$, where R_i is the number of remaining iterations. According to GSS, R_0 is the total number of iterations, i.e. M, and $C_i = \lceil R_i/P \rceil$, where $R_{i+1} = R_i - C_i$. Thus, $C_i = \lceil (1 - \frac{1}{P})^i \cdot \frac{M}{p} \rceil$ and the number of scheduling steps is $N \simeq \frac{1}{\ln\lceil \frac{P}{P-1} \rceil} \ln\lceil \frac{M}{P} \rceil$. GSS allocates most of the work in the first few scheduling steps and the amount of the remaining work is not adequate to balance the workload, so that in some cases the load balancing achieved by GSS is poor.

The *Factoring Self-Scheduling* (FSS) [15] scheme schedules tasks in batches of P equal chunks. In each batch j, a worker is assigned a chunk size given by a subset of the remaining iterations (usually half) divided by the number of workers. The chunk size for batch j is $C_j = \lceil \frac{R_j}{\alpha \times P} \rceil$ (thus $C_i = \lceil (\frac{1}{\alpha})^{j+1} \frac{M}{P} \rceil$) and $R_{j+1} = R_j - (P \times C_j)$, where the parameter α is computed (by a probability distribution) or is sub-optimally chosen $\alpha = 2$. The total number of steps is approximately equal to $N \simeq P 1.44 \ln\lceil \frac{M}{P} \rceil$ [20]. The weakness of this scheme is the difficulty to determine the optimal parameters. However, tests show improvement on previous schemes due to fewer adaptations of the chunk-size.

The *Trapezoid Self-Scheduling* (TSS) [18] scheme linearly decreases the chunk size C_i. In TSS the first and last chunk size pair (F, L) may be set by the programmer. In a conservative selection, the (F, L) pair is determined as: $F \frac{M}{2 \times P}$ and $L = 1$. This ensures that the load of the first chunk is less than $1/P$ of the total load (i.e., the total number of tasks) in most task workload distributions and reduces the chance of imbalance due to a large first chunk. Still excessive synchronization may occur. One can improve this by choosing $L > 1$. The proposed number of steps needed for the scheduling process is $N = \frac{2 \times M}{(F+L)}$. Consequently, the decrement between consecutive chunks is $D = (F - L)/(N - 1)$, and the chunk sizes are $C_1 = F, C_2 = F - D, C_3 = F - 2 \times D, \ldots$. TSS improves on GSS by decreasing the chunk-size linearly, in a less dramatic way, achieving in this way better load balancing.

All the aforementioned dynamic scheduling methods provide flexibility concerning the tradeoff between load-balancing and scheduling overheads. Depending on the nature of a particular application (i.e. the minimum and maximum computation times of chunks, the distribution of loads among chunks, the scheduling overhead times, etc.) and the features of the underlying computational platform, a particular method may outperform the rest in terms of total parallel execution time. The goal of this paper is to select the most proper of the above methods for the numerical code that simulates the trajectories of charged particles moving under storm-time substorm conditions. In the next section we provide relevant experimental data on medium-scale cluster platforms, homogeneous, heterogeneous and loaded.

4 Experimental evaluation

In this section we evaluate the impact of dynamic scheduling methods on the total parallel execution time of the ion acceleration models. The ion acceleration model is written in Fortran 90 and compiled with the ifort v. 8.1 compiler. The dynamic scheduling methods are implemented in C++ with calls to the MPI library (MPICH v. 1.2.7) and compiled with the icc v. 8.1 compiler. We performed three sets of experiments. The first uses a homogeneous platform which is a 16-node Linux cluster (kernel 2.6.23.1).

Each node includes two quad-core Xeon chips based on Intel's Core 2 microarchitecture (E53352GHz). Two cores per package share a 4MB L2 cache. The interconnection network is Gigabit Ethernet. We experimented with 64 processes running in the above cluster. Our second set of experiments is run on a heterogeneous platform using 48 processes from the above cluster and 16 processes from an 8-node Linux cluster with a 2-way SMP Intel Pentium 4 Xeon processor with 1MB L2 cache. The third set involved 64 processes from the first cluster artificially loaded with stochastic loads. In all cases, we have run the experiments 10 times and the results presented in the following sections, are the average of these runs.

4.1 Homogeneous platform

In the first series of experiments we run the application with the dynamic algorithms SS, TSS, FSS, GSS, and we compared their performance to that of a static algorithm. The static algorithm calculates the chunk size by dividing the number of tasks (the number of studied particles in our case) by the number of the available processing nodes, and assigns these chunks to the processing nodes in a single step. For this experiment we used 64 processes from the 16 node, quad-core, homogenous cluster. We consider that the processing capabilities of all processes are the same, i.e. that all processes can process the same amount of data at the same time, but the completion time for the computation of each particle is variable. The computation times of a sample run of 6400 particles is given in Figure 4. We expect that this variation will make the dynamic algorithms more suitable than the static for the scheduling of this application.

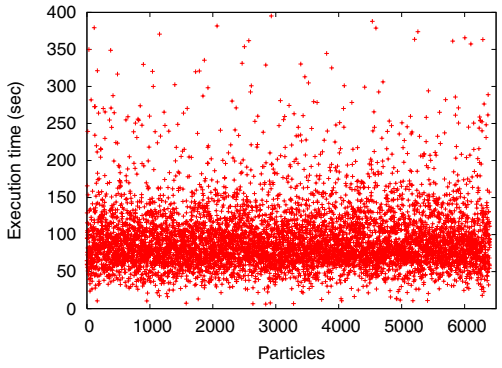

Figure 4. Distribution of execution times per particle for 6400 particles.

As we can see from the results given in Figure 5, all dynamic algorithms outperform the static algorithm. The performance improvement ranges from about 10% to 20%. We can see that the performance improvement drops as the number of particles increases. This is explained by the fact that as we can see from Figure 4, the completion time of the particles is uniformly distributed. As the number of particles increases, the computation time of all chunks of particles assigned by the static algorithm approaches an average value, and so all processing nodes finish at roughly the same time. This in not true when we have a smaller number of particles, so the performance gap between the static and the dynamic schemes is broader. Moreover, we can see that in all cases, the simple self-scheduling (SS) has better or similar performance to that of the other self-scheduling algorithms. As it is mentioned in Section 3, the SS algorithm can achieve the best load balancing from all other members of the self-scheduling class, with the penalty of increased scheduling overhead. In our case the scheduling overhead, which is in the order of miliseconds, is insignificant with respect to the task completion time, which is in the order of tens-hundreds seconds, so we expected that the simple self-scheduling algorithm will outperform the other dynamic schemes. That means that pure SS is more suitable for the combination of computation and scheduling-overhead times in the platforms under consideration and thus we will employ this algorithm for the dynamic scheduling in the forthcoming experiments presented in the following paragraphs.

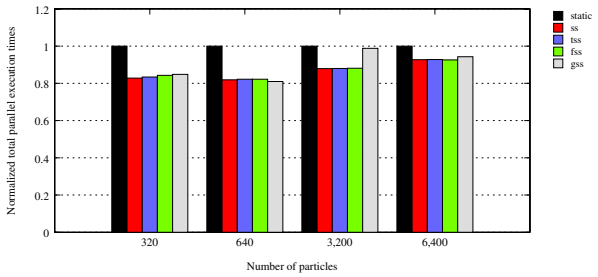

Figure 5. Comparison of static and dynamic scheduling schemes in the homogenenous platform for different numbers of particles.

4.2 Heterogeneous platform

In this set of experiments we add system heterogeneity on top of workload variability. In this case we do not deal with just variable task completion times, but also with uneven processing capabilities of the processing nodes. We use again 64 processes, 48 from the homogeneous cluster and 16 from the 2-way SMP cluster. We have run a small-size problem consisting of 10 particles to quantify the difference on the processing capabilities of the two types of processing nodes, which resulted to a normalized ratio of $1.9 : 1$, i.e the first type of processing nodes can perform

1.9 times more computations than the second type of processing nodes at the same time. In this series we tested the performance of the best dynamic algorithm of the previous set of experiment, i.e., simple SS, to that of static and weighted-static (wstatic) algorithms. The weighted-static algorithm is similar to the static algorithm, but it assigns more particles to the faster processing nodes, in a ratio of $1.9 : 1$. In Figure 6 we can see the normalized parallel execution times for each of these algorithms. The performance gain of the SS compated to the static algorithm in the heterogeneous case ranges from 34% to 46%. The gap between the static and the dynamic algorithm is broader than in the homogeneous case and this is due to the increased load imbalance cased by the difference in the processing capabilities of the processing nodes. The weighted-static (wstatc) algorithm that takes into consideration the computational power of each cpu, as expected performs significantly better than the static one, but the dynamic algorithm is still better, in the range of 13% to 26%. That is because even if we balance the processing capabities of the processing nodes there is still imbalance caused by the workload variability.

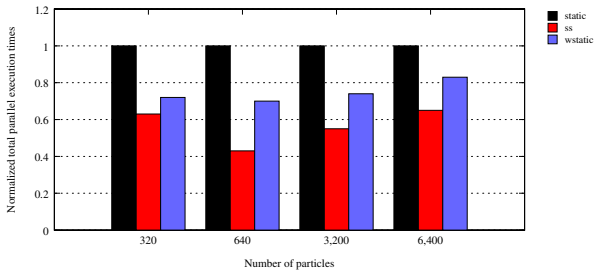

Figure 6. Comparison of static and dynamic scheduling schemes in the heterogeneous platform for different numbers of particles.

4.3 Homogeneous platform with loads

In the third set of experiments we use again the homogeneous system of the first experimental set but this time we inject external loads to the system, i.e. we start a number of processes on the processing nodes. We make the simple assumption which holds in practice, that the processing capability of a processing node is inversely proportional to the number of processes existing in its run-queue. So, we run a load generator at each node that creates loads at intervals and with lifetimes following a poisson distribution. The mean arrival times and the lifetime of the loads is different for each node to simulate a completely unpredictable environment, starting for 1 load coming every 10 seconds with lifetime of 5 seconds, up to 10 loads coming every 30 seconds with lifetime of 20 seconds. We do not consider a weighted-static algorithm this time since the system is homogeneous and we have no indication of the perceived processing capability of the system, i.e,. we do not know how much the performance of each node is deteriorated under the influence of the external load. The results of this set of experiments are given in Figure 7. We observe that the dynamic algorithm is always better than the static algorithm, and the performance difference is no-less than 40%, with maximum value of 50%. This is the case where we can observe the greatest difference between the static and the dynamic algorithm, and this is explained by the fact that in this case the existence of the external loads creates the most unpredictable and dynamic environment, with the greatest probability of load imbalance.

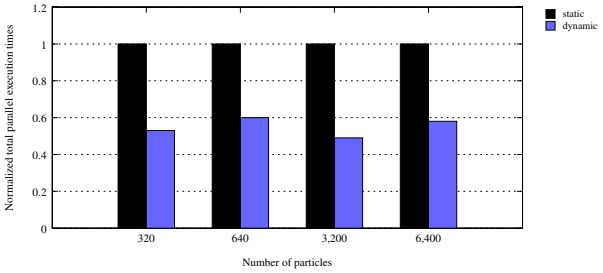

Figure 7. Comparison of static and dynamic scheduling schemes in the homogenenous platform with loads for different numbers of particles.

5 Conclusions – Future work

In this paper we investigated the performance impact of dynamic scheduling strategies on a numerical code that simulates the trajectories of charged ion particles moving in the magnetosphere under the scenario of storm-time substorm occurence. Such an application is important to predict severe Space Weather phenomena that may create important impacts such as spacecraft charging, astronaut health hazards, radio black-outs etc. In order to effectively model geomagnetic activity phenomena we simulate the motion of a large number of particles with different initial conditions (energy, coordinates, etc). This variety in conditions may lead to signicant variations in the simulation/computation time of different particles, causing load imbalances and reduces performance under parallel execution scenarios.

To deal with the above problems we test static and various dynamic (self) scheduling schemes such as SS, TSS, FSS and GSS that provide different tradeoffs between the load balance achieved and the scheduling overhead. The nature of the computations in our numerical code involves large computational times (in the order of magnitude of 100

seconds per task/particle) rather uniformly distributed since the initial values are set randomly per particle. Nevertheless, dynamic scheduling is meaningful even in the case of homogeneous computing nodes and numerous tasks, providing a performance improvement of 10% in 64-process execution for 100 tasks per process. In this homogeneous case, if the number of particles assigned to each process is smaller then this performance improvement can reach up to 20% for 5 particles per process. The best performance in this case is attained by the simplest self-scheduling scheme (pure Self Scheduling – SS) since, as expected, the combination of execution times and scheduling times for this code allows for scheduling synchronization per task which achieves the best possible load balance.

The positive impact of dynamic scheduling is even more evident when additional imbalance or heterogeneity factors come into play. In a heterogeneous computing platform the benefit of dynamic scheduling over the weighted static case reaches up to 26%. Finally, in an non-dedicated environment loaded with stochastic loads, dynamic scheduling techniques provided up to a 50% reduction in the total execution time of the algorithm.

For future work, we plan to compare the efficiency of the dynamic scheduling methods in larger-scale clusters where the scheduling overhead is expected to play a more important role, thus possibly accentuating the use of an alternative scheduling method that is more aware of scheduling overhead penalties. The same will possibly hold after the optimization of the sequential code which will reduce the execution times per particle thus posing more frequent scheduling requests.

References

[1] D. N. Baker, T. I. Pulkkinen, V. Angelopoulos, W. Baumjohann, and R. L. McPherron. Neutral line model of substorms: Past results and present view. *Journal of Geophysical Research*, 101:12975–13010, 1996.

[2] I. Banicescu and Z. Liu. Adaptive factoring: A dynamic scheduling method tuned to the rate of weight changes. *Proc. of the High Performance Computing Symposium*, pages 122–129, 2000.

[3] A. T. Chronopoulos, M. Benche, D. Grosu, and R. Andonie. A class of loop self-scheduling for heterogeneous clusters. In *CLUSTER '01: Proceedings of the 3rd IEEE International Conference on Cluster Computing*, page 282, Washington, DC, USA, 2001. IEEE Computer Society.

[4] I. A. Daglis and Y. Kamide. The role of substorms in stormtime particle acceleration, in Disturbances in Geospace: The Storm-Substorm Relationship. *Geophysical Monograph Series, DOI 10.1029/142GM11, edited by A. S. Sharma, Y. Kamide, and G. S. Lakhina, American Geophysical Union, Washington, DC,*, 142:119–129, 2003.

[5] D. Delcourt. Particle acceleration by inductive electric fields in the inner magnetosphere. *Journal of Atmospheric and Solar-Terrestrial Physics*, 64:551–559, 2002.

[6] D. Delcourt, J.-A. Sauvaud, and A. Pedersen. Dynamics of single-particle orbits during substorm expansion phase. *Journal of Geophysical Research*, 95:20853–20865, 1990.

[7] D. L. Eager, E. D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. Softw. Eng.*, 12(5):662–675, 1986.

[8] M. Harchol-Balter and A. B. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Trans. Comput. Syst.*, 15(3):253–285, 1997.

[9] S. F. Hummel, J. Schmidt, R. N. Uma, and J. Wein. Load-sharing in heterogeneous systems via weighted factoring. In *SPAA '96: Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, pages 318–328, New York, NY, USA, 1996. ACM.

[10] V. V. I. Banicescu and J.Devaprasad. On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring. *Cluster Computing: The Journal of Networks, Software Tools and Applications*, 6(3):215–226, 2003.

[11] C. F. Kennel. Convection and Substorms - Paradigms of magnetospheric phenomenology. *Oxford University Press, New York,*, pages –, 1995.

[12] C. Kruskal and A. Weiss. Allocating independent subtasks on parallel processors. *IEEE Trans. on Software Eng*, 11(10):1001–1016, 1985.

[13] M. Liemohn and J. Kozyra. Assessing the importance of convective and inductive electric fields in forming the stormtime ring current. *R.L. Winglee (ed.), Sixth International Conference on Substorms, Univ. Washington, Seattle*, 103:456–462, 2002.

[14] C. Polychronopoulos and D. Kuck. Guided self-scheduling: A practical self-scheduling scheme for parallel supercomputers. *IEEE Trans. on Computer.*, C-36(12):1425–1439, 1987.

[15] E. S. S.F. Hummel and L. Flynn. Factoring: A method for scheduling parallel loops. *Comm. of the ACM.*, 35(8):90–101, 1992.

[16] D. Stern. The motion of a proton in the equatorial magnetosphere. *Journal of Geophysical Research*, 80:595–599, 1975.

[17] N. A. Tsyganenko. A magnetospheric magnetic field model with a warped tail current sheet. *Planetary and Space Science*, 37:5–20, 1989.

[18] T. Tzen and L. Ni. Trapezoid self-scheduling: A practical scheduling scheme for parallel compilers. *IEEE Trans. on Paral. and Dist. Sys.*, 4(1):87–98, 1993.

[19] H. Volland. semi-empirical model of large-scale magnetospheric electric field. *Journal of Geophysical Research*, 78:171–180, 1973.

[20] K. Yue and D. Lilja. Parallel loop scheduling for high-performance computers, 1993.