1 USING A HPC SYSTEM FOR THE SIMULATION OF THE TRAJECTORIES OF SOLAR WIND PARTICLES IN THE IONOSPHERE

Gladimir V. G. Baranoski and Jon G. Rokne

Department of Computer Science, University of Utah, Salt Lake City, Utah, USA
Department of Computer Science, The University of Calgary, Calgary, Alberta, Canada
{gbaranos, rokne}@cpsc.ucalgary.ca

Abstract: As the Sun rotates, charged particles are thrown out in spiraling streams. These high energetic particles form the solar wind, and may eventually interact with atoms and molecules in the Earth's ionosphere. Light emissions in various wavelengths may result from these interactions. These emissions constitute natural displays commonly known as the "Aurorae". They are a direct manifestation of plasma physics and, therefore, represent a natural laboratory for studies on this field. The computational bottleneck in the modeling of these phenomena is the simulation of the trajectories of the solar wind particles in the ionosphere. This paper describes the use of a high performance computing system in conjunction with the Message Passing Interface (MPI) standard to perform these simulations. Along with the description of this application we present experimental results to illustrate performance gains that can be obtained using this approach.

Keywords: MPI, distributed processing, solar wind, stochastic simulation.

1.1 INTRODUCTION

During solar flares and coronal mass ejections (Burtnyk, 2000), plasma particles, namely protons and electrons, are emitted from the Sun primarily from highly magnetized areas in the solar photosphere known as sunspots, which have a lower temperature than the solar corona. As the sun rotates these particles are thrown out in spiraling streams, and after few days this "solar wind" hits the Earth's magnetosphere. A shock front is then formed facing the Sun

(day side), while a plasma sheet is created in the opposite direction (night side) (Figure 1.1). In the core of this plasma sheet there is a magnetic field reversal where a small fraction of the solar wind particles are trapped and accelerated towards the regions around Earth's magnetic poles. These particles create a global instability known as a magnetospheric substorm.

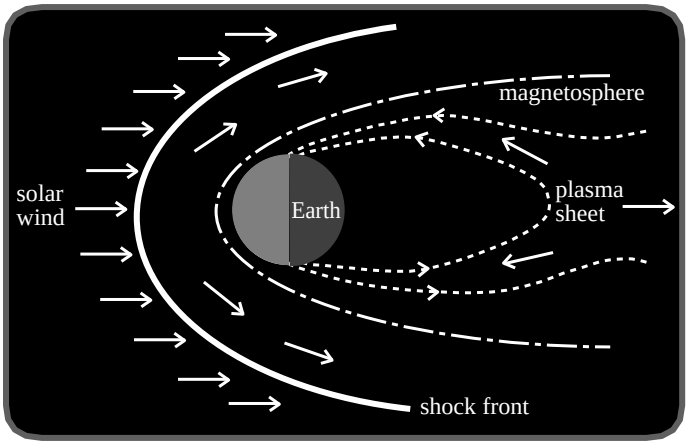

Figure 1.1 Sketch describing solar wind interaction with the Earth's magnetosphere.

These magnetic substorms are characterized by large variations in the magnetic field and can occur over a short period of time. They may affect systems of energy transmission and the operation of satellites used in communication and global positioning systems (Odenwald, 2000). The bright and colorful displays of the Aurora Borealis and its southern counterpart, the Aurora Australis, are manifestations of such phenomena. In other words, the upper atmosphere is like a "TV screen" that conveniently allow us to view and study phenomena in the distant tail of the plasma sheet (Figure 1.1). More "explosive" substorms, and consequently more impressive auroral displays, are more likely to occur when the Sun goes through periods of high sunspot activity at approximately 11-years intervals, the next peak of solar activity being expect to occur in the 2000-2001 period (Burtnyk, 2000; Brekke and Egeland, 1994).

Recently we have developed a research projected aiming at the visual modeling of auroral phenomena (Baranoski et al., 2000). The computational bottleneck of our experiments was the simulation of the trajectories of solar wind particles in the ionosphere. In order to accelerate the feedback cycle from our experiments, we needed to resort to a high performance computing system. In this paper we describe how we implemented these simulations on such a system using the Message Passing Interface (MPI) standard (Gropp et al., 1996; Gropp and Lusk, 1996). We first outline the underlying physical aspects involved in these trajectories and the simulation approach used in our experiments. We then describe the high performance computational platform used in our simu-

lations and provide an overview of the MPI code used in our implementation. Some results illustrating the performance gains follow this presentation. The paper closes with a summary and directions for future work.

1.2 TRAJECTORIES OF SOLAR WIND PARTICLES IN THE IONOSPHERE

1.2.1 Physical Aspects

A thin sheet of energetic high-speed alpha particles (protons and electrons) is precipitated in the ionosphere causing auroral light emissions. The auroral displays are caused by both electrons and protons. Electrons are the primary auroral particles, since their collisions with atmospheric atoms and molecules produce most of the light of the visual aurora. Because the trajectories of protons can be handled similarly to trajectories of electrons, we will concentrate on the electrons when discussion the simulation of the trajectories of auroral particles in this paper, implicitly including the protons by analogy.

The angle between the electron's directed velocity vector ($\vec{v}$) and the geomagnetic field vector $\vec{B}$ is called the pitch angle, α (Chamberlain, 1961). A "loss cone" of pitch angles is bounded at an altitude h by an angle α_D. This angle is given by an adiabatic invariant[1], which takes into account the ratio between the strength of $\vec{B}$ at h and at an altitude of $100km$ (the auroral lower border) (Haymes, 1971). Electrons with $\alpha \leq \alpha_D$ are in the loss cone and are precipitated ("lost") into the atmosphere. The boundary of this loss cone is somewhat diffuse ($\alpha_D \approx 2 - 3°$), since the value of α_D decreases with altitude (or the strength of $\vec{B}$).

As the electrons travel down along the Earth's magnetic field lines they suffer many random deflections as they hit atmospheric particles. These random deflections will spread the electrons out horizontally. The electron trajectories are governed by three processes (Borovsky et al., 1991): spiraling along the geomagnetic field lines, slowing down due to interaction with atmospheric atoms and molecules, and scattering off the gas atoms.

The directed velocity of an electron can be separated into two components, parallel and perpendicular to $\vec{B}$. The parallel component is unaffected by the geomagnetic field, and an electron, therefore, moves along $\vec{B}$ with a speed equal to its parallel velocity. An electron does not cross the magnetic field line. Instead, it travels around on a circle at a speed equal to the perpendicular component of its velocity. The combination of both components leads to a spiraling motion for the electrons.

Electrons with a kinetic energy of $1keV$ (velocity of $\approx 18000km/s$) penetrate to less than $150km$ altitude. At each collision the electron looses about $35eV$ of energy (Brekke and Egeland, 1994). The visible auroral emissions are caused

[1] $\alpha_D = arcsin\sqrt{\frac{B}{B_{100}}}$ (Chamberlain, 1961; Haymes, 1971).

by medium-energy electrons ($\approx 0.5 - 20keV$). On average an electron with $10keV$ (60000km/s) can collide 300 times before being brought to a halt at an altitude of about $100km$ above the ground. These electrons are not destroyed, but in the course of their passage through the atmosphere they eventually become indistinguishable from the ambient electron population. This loss of energy is owed to multiple inelastic scattering due to collisions with atoms, molecules and ions (Rees, 1989). This scattering is characterized by a change of direction as well as of energy. Typically, fluxes required for a moderate aurora are 10^8 electrons per square centimeter every second, with average speeds of $20000km/s$. The electrons can also be scattered elastically, *i.e.*, with angular deflection, but without energy loss.

Because of these deflections the electron trajectory keeps changing from an old spiral to a new spiral displaced from the old one. As it undergoes many scatterings the electron suffers random displacements across the magnetic field lines. This cause a group of electrons starting on the same field line to be spread out onto other field lines. Therefore, even if the electrons form an infinitely thin sheet high above the atmosphere (this is still a topic of ongoing research), they will broaden to a sheet with a non-zero thickness in the upper atmosphere (Borovsky et al., 1991).

1.2.2 Simulation

Our modeling approach consists of simulating the trajectories taken by beams of electrons instead of following the trajectories of individual electrons. These beams represent auroral rays, or curls (Haymes, 1971). Before simulating their trajectories along the geomagnetic field lines, the starting points of the beams are determined using sine waves (Baranoski et al., 2000). This modeling approach was based on studies of auroral physics (Chmyrev et al., 1992; Hallinan, 1976) which show that the stream of precipitating electrons can accurately be represented by sheets with boundaries given by sine waves. They have a phase shift which may be as high as 0.35π radians (Hallinan, 1976).

Each path is simulated incrementally, through a vertical displacement t such that:

$$t^{new} = t^{old} + (d_t\ \xi_1) \tag{1.1}$$

where ξ_1 is an uniformly distributed random number in the interval [0..1].

The use of this random displacement is consistent with the spatial inhomogeneity of auroral electrons (Rees, 1989). The threshold d_t is an input parameter which depends on the initial energy of the electrons. For instance, we could use $d_t = \frac{1}{300}$ since we assume that $t \in [0..1]$.

The random deflections along a beam trajectory are simulated using the following sequence of steps which is based on the diagram sketched in Figure 1.2b. An intermediate point $P^{new'}$ is computed using:

$$P^{new'} = P^{old} + (L\ d_t\ \xi_1)\,\frac{\vec{B}}{|\vec{B}|} \tag{1.2}$$

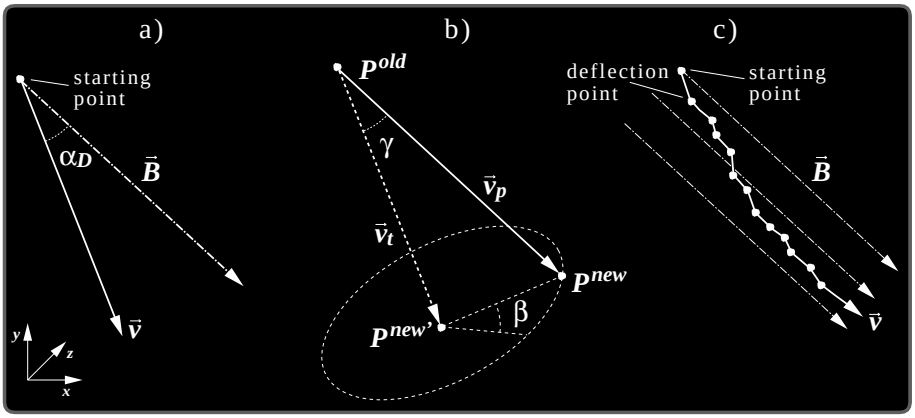

Figure 1.2 Simulating the trajectory of an electron beam: a) starting point, b) computation of a deflection point along the trajectory and c) electron beam crossing onto other magnetic field lines.

A temporary precipitation vector is then computed:

$$\vec{v_t} = P^{new'} - P^{old} \tag{1.3}$$

The perturbation of this vector using a polar angle γ and an azimuthal angle β provides the direction for the precipitating vector $\vec{v_p}$. The new point is computed using $\vec{v_p}$ as a directional displacement such that:

$$P^{new} = P^{old} + \vec{v_p} \tag{1.4}$$

The polar angle γ corresponds to the pitch angle adjusted to follow the reduction of α_D with altitude. This adjustment is performed using:

$$\gamma = \alpha_D - (t\,\Delta\gamma) \tag{1.5}$$

where $\Delta\gamma$ is an input angular parameter usually on the order of 0.015 radians (Haymes, 1971).

The azimuthal angle β is obtained randomly such that:

$$\beta = 2\pi\,\xi_2 \tag{1.6}$$

where ξ_2 is an uniformly distributed random number in the interval [0..1]. This choice for β also follows the stochastic characteristics of the phase space distribution of the auroral electrons (Rees, 1989). The net result of this displacements is the electron beam being spread out onto other field lines (Figure 1.2c). In our simulation a deflection points is considered as an emission point, and its y coordinate is used to access tables which provide data for spectral and

intensity variations of auroral emissions with respect to the altitude (Baranoski et al., 2000).

Because the electron trajectories are governed by statistical processes, the actual penetration depths are not identical even for two electrons with identical initial conditions (Borovsky et al., 1991; Kivelson and Russell, 1995). In our simulations we account for this aspect through two kinds of perturbations on the electron beams. The first consists of changing the interval for the parametric variable t, which becomes $[0..(1.0 - \xi_3 \Delta L)]$, where ξ_3 corresponds to an uniformly distributed random number in the interval $[0..1]$, and ΔL corresponds to an input parameter representing a variation on the beam path length, L (an input parameter). The second consists of perturbing the z coordinate of the starting point of a electron beam (Figure 1.2a). This perturbation is performed by applying a 3D correlated noise function (Perlin, 1985) and also causes a variation in the beam path length. However, in the second case the variation will be related to perturbations performed in the path length of the neighboring beams and it will be more noticeable in a plane perpendicular to an auroral display.

1.3 PARALLEL STRATEGY

Since the stochastic trajectories of the electron beams are independent, we can apply a divide and conquer strategy to reduce the total running time of our simulations. In other words, we can process N trajectories separately in several processors. In our current implementation one processor, the master, controls the distribution of the tasks among a number of other processors, the slaves (Figure 1.3). In this way, assuming that there are np processors available, each slave will responsible for the computation of $\frac{N}{np-1}$ trajectories.

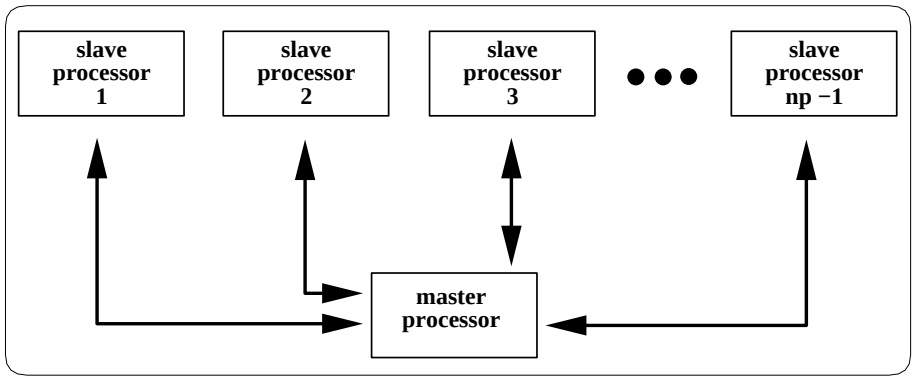

Figure 1.3 Sketch of the distributed processing environment used in our simulations showing the message passing connection between the master and slave processors.

The communication between the master processor and the slaves is performed using the Message Passing Interface standard, a library specification for message-passing designed for high performance on both massively parallel machines and on workstation clusters (Gropp et al., 1996; Gropp and Lusk, 1996). The main advantages of using MPI are portability and ease-of-use. Building and debugging a MPI application is a rather simple task since it is possible to use original host tools such as compilers, linkers, debuggers. Figure 1.4 presents a C++ code fragment illustrating the MPI commands used in our implementation to perform the communication between the master and slave processors.

The most important design step of a MPI application is the partition of the sequential algorithm into a parallel solution composed by different tasks. The application described in this paper was divided into three tasks:

- *Server task:* executed by a set of $np-1$ replicas of the routine **electron**, which performs the computation of the deflection points and and their corresponding spectral and intensity values. Each of these replicas should be allocated to a different slave processor. These values are updated directly in a temporary image raster file stored in the array **color**.

- *Control task:* corresponds to the initialization of the variables, loading of the parameter files and the control of the execution of the server tasks by the slave processors through remote calls to the routine **electron**. This task is performed by the master processor.

- *Join task:* also executed by the master processor, this task composes the final image raster file stored in the array **image** using the results of the calculations performed by the slave processors and stored in the array **color**.

Although the master processor remains idle during the execution of server task by the slaves in our current implementation, the MPI commands could be modified to allow the use of the master processor for the execution of this task as well.

The high performance hardware used in our simulations was a Compaq Alpha cluster of 30 machines, consisting of $500-Mhz$ Alpha PWS500 processors, connected through a Myricom Myrinet network (providing speeds of up to 1 gigabyte per second). In this distributed processing environment the firmware used to move data between the host memory and the network link is the Trapeze network driver (Chase et al., 1999).

1.4 RESULTS AND DISCUSSION

We have performed two sets of experiments involving different numbers of trajectories: 10^5 and 10^6. For each trajectory approximately 300 deflection points were computed. The graph presented in Figure 1.5 illustrates the speed-up gain of the parallel over the sequential execution. Through the analysis of this graph we noticed that the application of the parallel strategy described in

```
// MPI initialization
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
MPI_Status status;
.
.
.
b=N/(np-1); // number of trajectories per slave processor
data[0]=nx; // number of horizontal pixels
data[1]=ny; // number of vertical pixels
data[2]=0;
data[3]=b;
m=(nx+1)*(ny+1)*3; // size of color array
bd=4; // size of data vector
.
.
.
if(rank==0) // master
{
 for(i=1;i<nprocs;i++)
 {
   MPI_Send(color,m,MPI_FLOAT,i,0,MPI_COMM_WORLD);
   MPI_Send(data,bd,MPI_INT,i,0,MPI_COMM_WORLD);
   MPI_Send(&b,1,MPI_INT,i,0,MPI_COMM_WORLD);
   MPI_Send(&i,1,MPI_INT,i,0,MPI_COMM_WORLD);
 }
 while(flag<nprocs-1)
 {
   MPI_Status status;
   MPI_Recv(color,m,MPI_FLOAT,MPI_ANY_SOURCE,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
   sp=status.MPI_SOURCE;
   if (sp>0)
   {
      flag++;
      for (i=0; i < nx ; i++)
        for (j=0; j < ny; j++)
          for (k=0; k<3; k++)
            image[i][j][k]=image[i][j][k]+color[i][j][k];
   }
 }
.
.
.
}
else // slaves
{
   MPI_Status status;
   MPI_Recv(color,m,MPI_FLOAT,0,0,MPI_COMM_WORLD,&status);
   MPI_Recv(data,bd,MPI_INT,0,0,MPI_COMM_WORLD,&status);
   MPI_Recv(&b,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);
   MPI_Recv(&p,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);

   data[2]=(p-1)*b;
   data[3]=p*b-1;
   fraction=(p-1)/p;

   if (electron(fraction, data, color))
      MPI_Send(color,m,MPI_FLOAT,0,0,MPI_COMM_WORLD);
}
.
.
.
// MPI clean-up
MPI_Finalize();
```

Figure 1.4 Code fragment illustrating MPI commands used for communication between the master processor and slave processors.

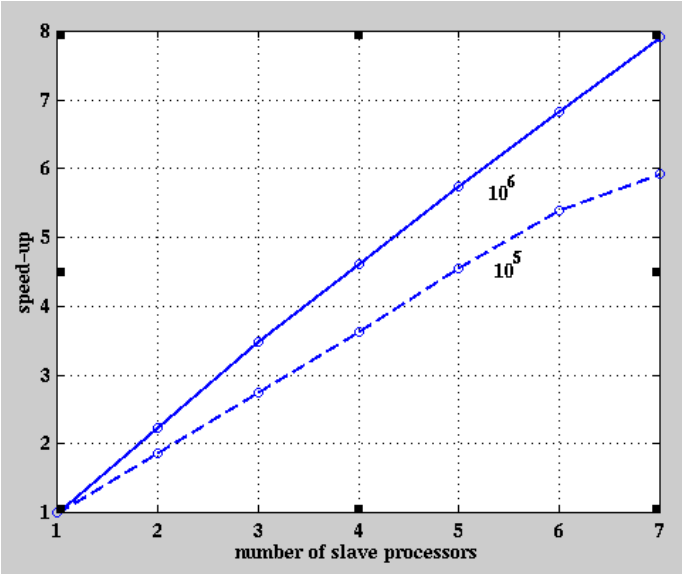

Figure 1.5 Comparison of speed-ups for the two sets of experiments involving the two different numbers of trajectories: 10^5 and 10^6.

the previous section results in best performances for the experiments involving a larger number of trajectories. Both sets of experiments are affected by overhead costs inherent to a parallel application. These costs are associated with remote process initialization, interprocess communication, synchronization, general application management and remote process termination. For experiments involving a large number of trajectories the overhead costs remain the same and the processing time increases. Therefore, the overhead costs percentage in the total running time becomes smaller as the number of trajectories increases. This aspect is also illustrated in Table 1.1, which presents the running times and speed-ups for the 7-processors case.

trajectories	running time (s)	speed-up
10^5	36	6.3
10^6	241	7.9

Table 1.1 Comparison of running times (given in elapsed CPU time) and speed-ups considering the use of seven slave processors.

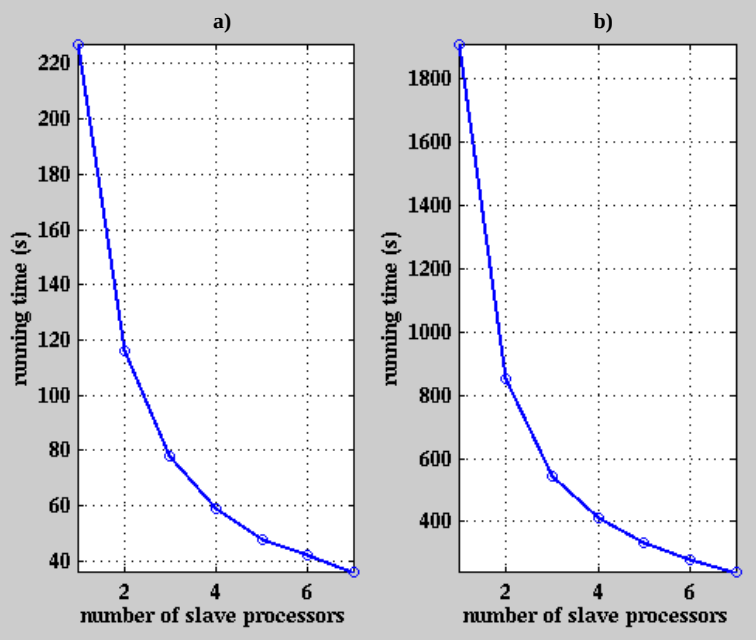

Figure 1.6 Comparison of running times (given in elapsed CPU time) for the two sets of experiments involving the two different numbers of trajectories: a) 10^5 and b) 10^6.

The results obtained in our experiments also demonstrate that the use of additional slave processors does not necessarily improve performance. In fact, we tend to obtain an asymptotic convergence of the running time as we increase the number of processors as shown in Figure 1.6. This happens mostly due to interprocess communication costs. This aspect may be attenuated through the use of a more efficient network driver. We expect to have the Trapeze driver replaced by the GM driver[2] shortly. This driver has a smaller latency than the Trapeze driver, and initial experiments have demonstrated the increase in communication efficiency by one order of magnitude.

1.5 CONCLUSION AND FUTURE WORK

The use of a high performance computing system was crucial in our research involving the visual simulation of auroral displays (Figure 1.7), since the study of the dynamic nature of auroral phenomena demands a large number of experiments, which in turn may require the simulation of a large number of trajec-

[2]http://www.myri.com:80/scs/

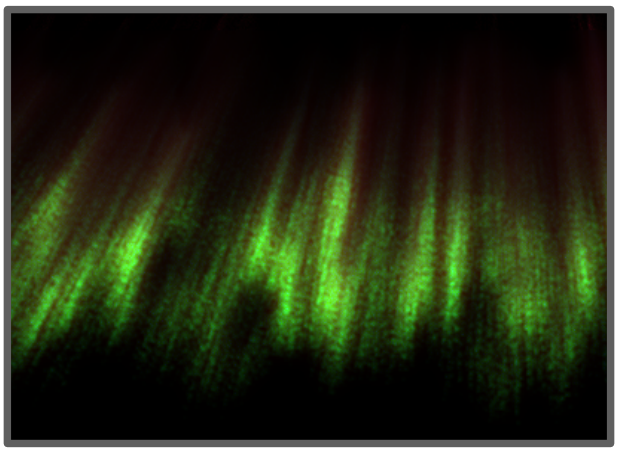

Figure 1.7 Computer generated image of an auroral corona (Baranoski et al., 2000).

tories. These simulations can be performed independently and, therefore, are suitable for the use of parallel processing techniques.

In this context the application of the parallel strategy described in this paper provided significant speed-up gains over the sequential execution of the algorithm for simulation of the trajectories of solar wind particles. The results of our experiments showed that the MPI standard is an effective tool for the distribution of for these tasks. Moreover, these experiments demonstrated that, although the number and type of processors are relevant in terms of running times, the speed-ups obtained are bound by the data transmission characteristics of the network.

Future efforts will involve the improvement of the current code and its execution on an expected upgraded configuration of the Compaq Alpha cluster, which will consist of 64 $667 - Mhz$ Alpha XPS1000 processors connected through a GM Myricom Myrinet network. We also intend to compare the speed ups obtained running our parallel application on this distributed processing configuration with the speed-ups obtained using other high performance computing platforms.

Acknowledgements

The authors would like to thank the Netera Alliance[3] for granting us access to the MACI cluster[4], specially Dr. Rob Simmonds (Research Fellow) and Mr. Robert Fridman (Computing Systems Administrator) for providing

[3]http://www.netera.ca/about.htm
[4]http://www.hpc.maci.ucalgary.ca/

software and hardware support during the implementation of the parallel processing application described in this paper. This research was supported by NSERC (Canada) grant number PDF-207026.

References

Baranoski, G., Rokne, J. G., Shirley, P., Trondsen, T., and Bastos, R. (2000). Simulating the aurora borealis. Technical Report 2000/655/07, Computer Science Department, The University of Calgary.

Borovsky, J., Suszcynsky, D., Buchwald, M., and DeHaven, H. (1991). Measuring the thickness of auroral curtains. *Arctic*, 44(3):231–238.

Brekke, A. and Egeland, A. (1994). *The Northern Lights, Their Heritage and Science*. Gröndahl og Dreyers Forlag, AS, Oslo.

Burtnyk, K. (2000). Anatomy of an aurora. *Sky & Telescope*, 99(3):35–40.

Chamberlain, J. (1961). *Physics of the Aurora and Airglow*. Academic Press, New York.

Chase, J., Anderson, D., Gallatin, A., Lebeck, A., and Yocum, K. (1999). Network i/o with trapeze. In *Hot Interconnects Symposium*.

Chmyrev, V., Marchenko, V., Pokhotelov, O., Shukla, P., Stenflo, L., and Streltsov, A. (1992). The development of discrete active auroral forms. *IEEE Transactions on Plasma Science*, 20(6):764–769.

Gropp, W., Lusk, E., Doss, N., and Skjellum, A. (1996). A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828.

Gropp, W. D. and Lusk, E. (1996). *User's Guide for* `mpich`*, a Portable Implementation of MPI*. Mathematics and Computer Science Division, Argonne National Laboratory. ANL-96/6.

Hallinan, T. (1976). Auroral spirals. *Journal of Geophysical Research*, 81(22):3959–3965.

Haymes, R. (1971). *Introduction to Space Science*. John Wiley & Sons, Inc., New York.

Kivelson, M. and Russell, C. (1995). *Introduction to Space Physics*. Cambridge University Press, Cambridge.

Odenwald, S. (2000). Solar storms: The silent menace. *Sky & Telescope*, 99(3):41–56.

Perlin, K. (1985). An image synthesizer. *Computer Graphics (SIGGRAPH Proceedings)*, 19(3):287–296.

Rees, M. (1989). *Physics and Chemistry of the Upper Atmosphere*. Cambridge University Press, Cambridge.