Dataparallel Semi-Lagrangian Numerical Weather Forecasting

Lex Wolters*†
High Performance Computing Division
Department of Computer Science, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands
llexx@cs.leidenuniv.nl

Gerard Cats†
Royal Netherlands Meteorological Institute
P.O. Box 201, 3730 AE De Bilt, The Netherlands
cats@knmi.nl

Nils Gustafsson†
Swedish Meteorological and Hydrological Institute
S-60176 Norrköping, Sweden
ngustafsson@smhi.se

Tomas Wilhelmsson
NADA
Royal Institute of Technology
Stockholm, Sweden
towil@nada.kth.se

Abstract

Different implementations on a massively parallel computer system of a semi-Lagrangian method within the numerical weather forecast model HIRLAM are presented. In principle semi-Lagrangian methods on massively parallel architectures result in irregular communications, i.e., communications between arbitrary processors. It is shown that the fastest implementation increases the total execution time per time step with an acceptable amount in relation to the advantage of applying a semi-Lagrangian method.

1 Introduction

Numerical weather prediction (NWP) has always taken advantage of high performance computer systems. Numerical weather forecast models were one of the first applications when computers were invented, and since that time have been implemented on the fastest systems available. This is mainly due to the economical and social importance of weather prediction.

A numerical/computational reason for using high performance systems is also easy to understand. On the one hand important factors determining the accuracy of the models are the horizontal and vertical resolutions applied within these models: the higher the resolution, the better the accuracy, but also the more calculations have to be carried out. On the other hand the forecasts must be available within a fraction of the time that they may considered to be valid. This shows a trade-off between the resolution and the total execution time of the model. Unfortunately present day computer power limits the resolutions to values that are unsatisfactory from a physical point of view. However, with the arrival of massively parallel systems on the market, and the expected power they can deliver, it is time to investigate if these systems can be applied efficiently for numerical weather forecast models. In [9] we have shown that this is possible for the state of the art HIRLAM[1] model, which is used for producing limited area numerical forecasts. The results of an implementation on massively parallel MasPar systems were demonstrated.

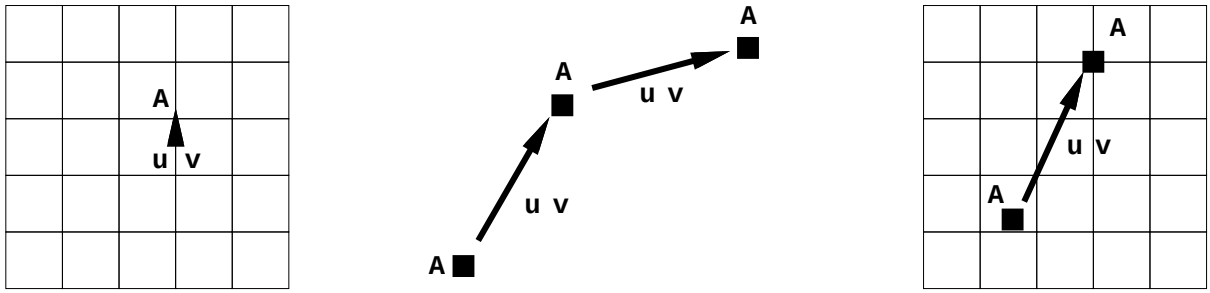
Besides using faster computer systems it is possible to achieve higher resolutions by decreasing the time-resolution, i.e., larger time steps. Better numerical techniques will allow larger time steps, and thereby save execution time. As a result the spatial resolutions can be increased. What determines the size of a time step?

A modern atmospheric model consists of two main parts. The first is called the 'dynamics'; its task is to solve a set of equations discretized to the model grid points. This set consists of several three-dimensional coupled non-linear hyperbolic partial differential equations (PDEs). They are known as the Primitive Equations, and can be derived from the Navier–Stokes equations (see e.g., [1]). The set contains two horizontal momentum equations, a hydrostatic equation, a mass continuity equation, a thermodynamic equation, and a continuity equation for water vapor. They describe the behavior of five prognostic variables: two horizontal wind components u and v, the tem-

*Support was provided by the Esprit Agency EC-DGIII under Grant No. APPARC 6634 BRA III.

†Support was provided through the Human Capital and Mobility Programme of the European Community.

[1] The HIRLAM system was developed by the HIRLAM-project group, a cooperative project of Denmark, Finland, Iceland, Ireland, The Netherlands, Norway, and Sweden.



(a) (b) (c)

Figure 1: Different formulations of physical problems: a. Eulerian formulation. b. Lagrangian formulation. c. Semi-Lagrangian formulation. **A** is the required variable, **u** and **v** are the horizontal velocities.

perature T, the specific humidity q, and the surface pressure p_s.

The second part is called the 'physics'; it describes the aggregate effect of the physical processes with scales smaller than the model resolution, on the larger, resolved, scales. Examples are vertical diffusion and convection. Some physical processes like radiation, not directly described by the basic model equations, are also parameterized.

The size of the time step is often determined by the numerical method applied in the dynamics to solve the set of partial differential equations. The HIRLAM model contains the following options: fully explicit versus semi-implicit integration schemes in combination with Eulerian versus semi-Lagrangian methods. Semi-implicit techniques can result in larger time steps (see e.g., [4]), but semi-Lagrangian descriptions allow an even bigger increase in time step [5]. An option within the HIRLAM forecast model, which does not deal directly with the time step, but has physical advantages, is to solve the Primitive Equations with spectral methods instead of grid point formulations.

In [9] we measured and compared the performance of each method on a massively parallel system. The method that currently is in use at several of the meteorological services participating in the HIRLAM project for their routine weather forecasting procedures is the so-called semi-implicit Eulerian grid point method, and it was shown that the algorithms are very well scalable both in the number of data point and the number of processors (up to 16K). However, in that investigation semi-Lagrangian methods were not considered despite the fact that they can increase the time step significantly, and therefore have become very attractive in the meteorological community. In this paper we discuss several implementations of the semi-Lagrangian method as applied in the HIRLAM model on a massively parallel system.

2 Semi-Lagrangian methods

In this section we will explain the computational aspects of semi-Lagrangian methods as they are applied in NWP models. For a more detailed physical and mathematical description the reader is referred to the review article [6].

Physical problems, like NWP, can be formulated in different ways. A common way is to describe the model in a Eulerian formulation: at each time step the variables are determined on fix points in space (see figure 1a). A different way to observe the variables is to travel with a set of particles as they evolve in time (see figure 1b). The problem of this method for NWP models is that after some time the chosen set of particles can be distributed irregularly [7]. As a result the required variables are known at the places where the particles are concentrated, and no information is available at other areas. This is of course unacceptable for weather forecasting. A method that combines the two formulations is known as semi-Lagrangian: each time step one travels with the set of particles, which during that time step ends exactly in one of the fixed grid points (see figure 1c). As mentioned before, semi-Lagrangian methods allow a much larger time step than Eulerian methods.

Semi-Lagrangian method are handled numerically in three steps. The first step determines the trajectory of each particle. For each grid point (arrival point) the displacements in three dimensions to the departure point of the particle at the beginning of the time step are determined. This is achieved by an inter- or extrapolation of the horizontal and vertical velocities. The second step consists of the evaluation of the value of the required variable at the departure

point. Because in general the departure point will not be a grid point, its value has to be determined by an interpolation of values of grid points surrounding the departure point. Finally, the third step updates the value of the variable at the arrival point.

This can be illustrated by a simple example (taken from [6]), namely the one-dimensional advection equation

$$\frac{dF}{dt} = \frac{\partial F}{\partial t} + \frac{dx}{dt}\frac{\partial F}{\partial x} = 0 \ , \qquad (1)$$

where

$$\frac{dx}{dt} = U(x,t) \ , \qquad (2)$$

and $U(x,t)$ is a given function. We assume that the values $F(x,t)$ are known at all grid points x_m at times $t_n - \Delta t$ and t_n, where Δt is the applied time step. In that case the semi-Lagrangian method determines the values on the grid points at time $t_n + \Delta t$ by

$$\frac{F(x_m, t_n + \Delta t) - F(x_m - 2\alpha_m, t_n - \Delta t)}{2\Delta t} = 0 \ , \qquad (3)$$

where the displacement α_m is the distance a particle travels in the x-direction in time t. If α_m is known, the value of F at arrival point x_m at time $t_n + \Delta t$ can be calculated with equation (3). The value α_m can be obtained by approximating equation (2) by (see [6])

$$\alpha_m = \Delta t U(x_m - \alpha_m, t_n) \ . \qquad (4)$$

This equation can be solved by an iteration process

$$\alpha_m^{(k+1)} = \Delta U(x_m - \alpha_m^{(k)}, t_n) \ , \qquad (5)$$

with an initial guess for $\alpha_m^{(0)}$, and where the values of U, possibly between grid points, are determined by an interpolation formula. An interpolation method should also be used to obtain the values $F(x_m - 2\alpha_m, t_n - \Delta t)$ in equation (3), since in general these positions will not correspond with grid points. Summarizing the three steps are: 1) compute the displacements α_m for all grid points x_m by the iteration process, given by equation (5), and an interpolation formula; 2) determine all values F at the departure points $x_m - 2\alpha_m$ at time $t_n - \Delta t$ by interpolation; 3) evaluate F at the arrival points x_m at time $t + \Delta t$ using equation (3).

This method can be extended to multi-dimensional problems. It is also possible to define a scheme using only two time levels instead of three time levels, which is in principle two times faster and therefore applied more often in NWP models. The details of these schemes can be found in [5, 6].

In the HIRLAM model [2] the first step of the two-level semi-Lagrangian method is performed by an iterative procedure where in each step a higher order interpolation procedure is used to obtain the displacements. After this step the trajectory is known, and the values at the departure points are evaluated by an interpolation routine, see also figure 2. It is possible to apply a linear, quadratic, cubic, or linear/cubic interpolation procedure. In figure 3 the 32 grid points involved in a linear/cubic interpolation are shown. The obtained values are then used to update the values at the arrival points.

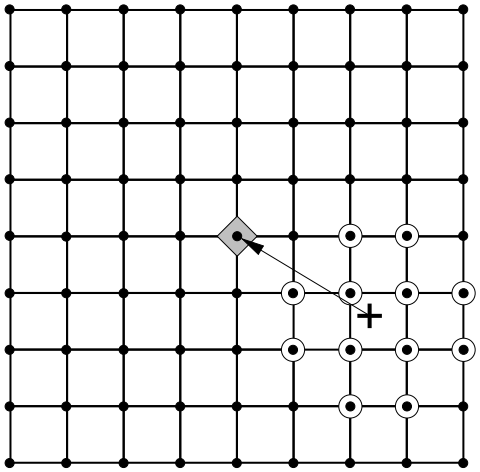


Figure 2: Calculation in a semi-Lagrangian formulation, projected on the horizontal plane. The arrow shows the trajectory from the departure point to the arrival point, which is a grid point. The grid points acting as interpolation points in a linear/cubic interpolation are encircled.

3 Parallelization

In this section we will discuss several implementations of the semi-Lagrangian method within the HIRLAM model on a MasPar computer system. A MasPar system [3] has a SIMD architecture with from 1,024 (1K) up to 16,384 (16K) processors. Each processor is called a Processor Element (PE). All together they form the PE-array, which is controlled by the Array Control Unit (ACU). A PE is an 80 ns load/store arithmetic processor with a 16 Kbytes or 64 Kbytes data memory.

The communication between the Processor Elements (PEs) can be divided into two classes: Xnet and Router communication. Xnet communication performs nearest-neighbor communications. The PE-array is arranged in a 2-dimensional mesh with toroidal wrap-around. With Xnet communication one can send data to or receive data from the eight neighboring PEs or even longer distances. The maximum communication bandwidth using Xnet is 23 Gbyte/s for a full 16K configuration. Router communi-

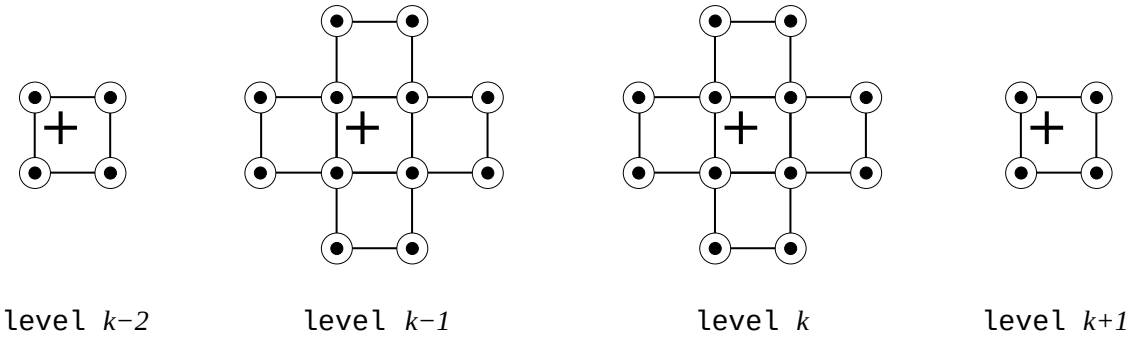


Figure 3: Linear/cubic interpolation. In this procedure 32 grid points are involved, that are distributed over four vertical levels. The plus sign indicates the location of the required interpolated value.

cation provides the possibility to send/receive data between two arbitrary PEs via a multi-stage crossbar network, so it takes care of the global communications. The communication time is independent of the distance between the PEs, but its maximum speed is considerably slower than for Xnet communication: 1.3 Gbyte/s. A limitation is that there is only one Router channel for 16 PEs.

The programming model for the MasPar systems is data-parallel programming with implicit communications or message passing.

The real challenge in parallelizing the semi-Lagrangian scheme on a dataparallel machine like a MasPar is the type of calculations performed during the interpolations. All other calculations can be easily implemented efficiently on a MasPar. Therefore we will concentrate on the interpolation routines. As mentioned before, the interpolations are carried out to determine the trajectory and to calculate the values at the departure points. We will restrict ourselves to the most common type of interpolation: linear/cubic.

An interpolation formula contains values between which the interpolation is carried out (the interpolation points) together with a weight for each value, that determines how much a value will contribute to the final interpolated value. The size of a weight is dependent on the distance between interpolation point and departure point. The closest grid point to the departure point around which the interpolation is performed, will be called the base point of the interpolation.

In [8] it was shown that in the dataparallel HIRLAM model on MasPar all horizontal grid points are distributed over the two-dimensional processor grid in a cyclic (so non-blocked) and cut-and-stack way. Based on this distribution one can make the following observations:

1. If the length of a trajectory in the horizontal direction is longer than half the horizontal grid size, the grid point that acts as the base point for the interpolation will be on a different processor than the processor with the arrival point. In general this means that all other grid points (read: processors) can act as the base point for the interpolation, which is required to determine the value at the arrival point on the current grid point (processor).

2. Based on a realistic maximum windspeed, the applied horizontal grid size, and the time step, it can be shown that the horizontal distance between the arrival point and the departure point of the trajectory will never be greater than four grid sizes. As a result the base point for the interpolation will be at most three grid points away from the current grid point.

3. All interpolation points will be on different processors than the one on which the base point resides (except for the points in 'pure' vertical direction). Taken into account the second observation and the linear/cubic interpolation one can conclude that the maximum horizontal distance between the arrival point (processor) and all interpolation points (processors) is four grid distances.

4. All weights in an interpolation are dependent on the exact position of the departure point of a trajectory. This position is available as a displacement with respect to the arrival point of each trajectory and therefore resides on the processor corresponding with that arrival point. As a result all weights for one interpolation are also determined and stored on that processor. The interpolation would be more local if it were performed on the base point processor instead of the arrival point processor. However, the resulting code will not be efficient, because each grid point may serve as base point of zero, one, two, or even more arrival points.

Summarizing: the contribution to the new value of the arrival point due to the semi-Lagrangian scheme is depen-

dent on 32 values within an area of $9 \times 9 \times n$ grid points, where n is the number of vertical levels. These 32 values need to be interpolated using 32 weights (one weight for each value), which are known at the arrival point. From a computational point of view this means that to update the value at the arrival point, the corresponding processor needs the values of some neighboring processors around that processor together with weights that already reside on that processor. It is known that these neighboring processors lay in a 9×9 square of processors with the 'current' processor as middle-point, which can also been seen from figure 2.

We now continue with a discussion about the different possible dataparallel versions of the interpolation routine. Of each version we will give a short description of the idea behind it, the main characteristics in particular concerning communications and memory accesses, and the actual performance in a test run of the spectral HIRLAM model with an integration area of $110 \times 100 \times 16$ points. The actual performance was obtained by timing the version on a 16K processor MasPar DPU Model MP-2216. All tests were performed with system release 3.2.0 of the MasPar software, which included the Mpfortran compiler (version 2.2.7). In all cases the –nodebug and –Omax compiler-options were specified, which prevents the inclusion of extra code for debug purposes, and performs the highest degree of optimization possible on a MasPar system. As discussed in [9] array and loop bounds were made known at compile time to prevent the generation of redundant communications.

3.1 Original version

This implementation is based on the original version of the HIRLAM reference model, except that the fields are stored as three dimensional arrays instead of two dimensional ones. For each grid/arrival point the necessary interpolation points are obtained by indirect addressing in the three dimensional arrays. With the applied data distribution this will result in this dataparallel version in indirect loads to obtain values on one processor and in Router communications to move the required values to the processor corresponding with the arrival point. This method shows a quite random memory access and communication pattern, and will probably result in collisions during the communications due to the fact that there is only one Router channel per 16 PEs. It turned out that the total extra time for this complete implementation of the semi-Lagrangian method is 3.50 s per time step. For comparison: one time step in the spectral semi-implicit Eulerian implementation takes 0.68 s.

3.2 Router versions

In these versions one tries to control the random behavior of the communications and indirect addressing. The interpolation routine in these versions consists of three phases. In the first phase, the 'collect' phase, each grid point is considered to be a possible base point of an interpolation. The required interpolation points are collected on the base point processor. Since it is known that the interpolations points are located on neighboring processors (zero, one or two grid distances from the base point processor, see figure 2) they can be collected by indirect loads to access the values in the vertical direction, and by Xnet communication to receive values from other processors. Since this holds for all grid points (processors) this results in regular and fast communications. In the second phase, the 'fetch' phase, each grid/arrival point fetches the collected interpolation values from his base point processor. Since the location of the base point processor is dependent on the displacement of each trajectory, this phase requires communication between arbitrary processors, so Router communication still has to be applied. The third phase consists of the actual interpolation phase, in which the interpolated value is calculated using the weights and fetched values.

There are several strategies possible to implement the idea outlined above, varying from collecting different numbers of possible interpolation points (in particular in the vertical direction) to applying different methods in collecting the values (e.g., using cshifts). However, the resulting performances of all these versions are quite disappointing: the extra time for the fastest method is 4.46 s per time step, which is larger than in the original version. An explanation for this outcome has not been found yet. It is possible that it is due to some characteristics of our test run.

3.3 Xnet versions

The general idea behind these versions is to remove the Router communications completely, and replace them by Xnet or nearest neighbor communications. How can this be achieved? As has been demonstrated earlier the area containing the possible interpolation points in the horizontal directions is limited to a 9×9 square of grid points with the 'current' grid point as middle point (see figure 2). If we just move all the values from these 9×9 neighboring processors to the memory of the 'current' processor without global communications, the actual interpolation can be performed on the processor without communications. The requirement to move all the values into the processor without global communications and the fact that it should be done for all processors simultaneously, can be solved by using cshifts. If one applies a series of cshifts, which covers the 9×9 area, and stores the required values between each

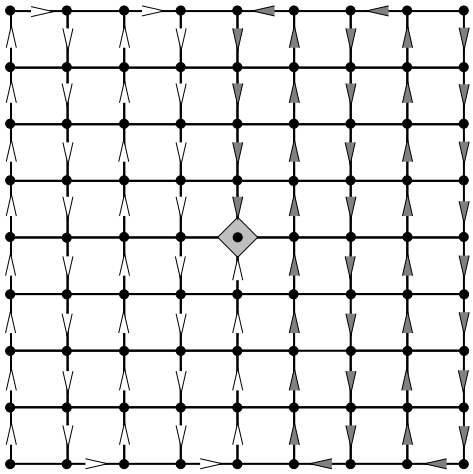


Figure 4: The order of values that are 'shifted-in' by the two series of cshifts, indicated by the two different kinds of pointers.

cshift, all necessary values are 'shifted in' in the processors. In figure 4 the applied order of cshifts is shown. A clear disadvantage of this method is that it increases the total usage of memory per processor significantly, since all possible interpolations points are shifted in and have to be stored in the local memory.

Instead of using global (Router) communications there are only nearest neighbor communications. In the interpolation-phase we still have indirect memory accesses, since we have to determine which of the 'in-shifted' values are the real interpolation points. The performance of this method was an extra time of 2.30 s per time step. It will be called the 2D Xnet version.

A significant improvement could be achieved by the observation that the interpolations for all grid points, in particular for all vertical levels, can be performed in one call of the interpolation routine. This in contrast with the original version where one call of the interpolation routine carries out the interpolations only on one vertical level. The reason for the improvement is that in the 2D Xnet version one has to 'shift in' all the values of all vertical levels, since for each grid point, the corresponding base points for the interpolation can be on different vertical levels. So the processor will contain all possible interpolation points for all vertical levels. The number of cshifts per shift-in is the same as in the previous version, but the total number of shift-in's is reduced to one instead of the number of vertical levels. The performance of this method was the best of all versions: 0.35 s per time step. We will call this the 3D Xnet version.

Table 1: Extra times for executing one time step using the semi-Lagrangian formulation with the different implementations of the interpolation routine in the spectral semi-implicit HIRLAM forecast model with a $110 \times 100 \times 16$ integration area on a MasPar MP-2 with 16K processors. Also the total execution time for one time step in the spectral semi-implicit Eulerian formulation is shown.

Implementation	Time (in sec)
Original	3.50
Router versions	4.46
2D Xnet version	2.30
3D Xnet version	0.35
Eulerian time step	0.68

4 Conclusions

In table 1 the extra execution times mentioned in the previous subsections for the different implementations of the semi-Lagrangian method per time step in the spectral HIRLAM forecast have been gathered. Also the time for executing one time step of a corresponding run in a spectral semi-implicit Eulerian formulation has been added, taken from [9].

As was mentioned in section 1 the semi-Lagrangian formulation allows a larger time step than the Eulerian formulation. In the HIRLAM model a three times larger time step can be chosen, which is related to the physics part of the model. Observing the additional costs for the semi-Lagrangian method as can be seen in table 1, it is clear that only the 3D Xnet version leads to an overall improvement in total execution time of the model: $0.68 + 0.35 = 1.03$ seconds per time step versus $3 \times 0.68 = 2.04$ seconds per time step for the Eulerian version. So a factor of nearly two can be saved in execution time when the semi-Lagrangian formulation is used. All other versions result in execution times that are too large in relation to the gain in time step size.

It can be concluded that the semi-Lagrangian formulation in NWP models can be implemented successfully on a massively parallel computer system. However, a disadvantage of the resulting implementation is the increased memory usage per processor. A second disadvantage is that the interpolation routine is not portable anymore. But, since this is limited to just one routine, it should not be considered a prohibitive objection for a production code as HIRLAM, running on several architectures.

References

[1] G.J. Haltiner and R.T. Williams, *Numerical Prediction and Dynamic Meteorology, second edition*, John Wiley & Sons, New York, 1980.

[2] P. Kållberg (editor), *Documentation Manual of the Hirlam Level 1 Analysis-Forecast System*, June 1990.

[3] MasPar, *MasPar MP-1 Hardware Manuals*, July 1992.

[4] A. Robert, J. Henderson, and C. Turnbull, *An Implicit Time Integration Scheme for Baroclinic Models of the Atmosphere*, Mon. Wea. Rev. 100 (1972) 329–335.

[5] A. Robert, *A Semi-Lagrangian and Semi-Implicit Numerical Integration Scheme for the Primitive Meteorological Equations*, Jpn. Meteor. Soc. 60 (1982) 319–325.

[6] A. Staniforth and J. Côté, *Semi-Lagrangian Integration Schemes for Atmospheric Models – A Review*, Mon. Wea. Rev. 119 (1991) 2206–2223.

[7] P. Welander, *Studies on the General Development of Motion in a Two-Dimensional, Ideal Fluid*, Tellus 7 (1955) 141–156.

[8] L. Wolters and G. Cats, *A Parallel Implementation of the HIRLAM Model*, in G.-R. Hoffmann and T. Kauranne (eds.), *Parallel Supercomputing in Atmospheric Science*, proceedings of the Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology, World Scientific Publ., 1993, pp 486–499.

[9] L. Wolters, G. Cats, and N. Gustafsson, *Limited Area Numerical Weather Forecasting on a Massively Parallel Computer*, in proceedings of the 8^{th} ACM International Conference on Supercomputing, July 11–15 1994, Manchester, England, ACM press, pp 289–296.