A DATA PARALLEL HIRLAM FORECAST MODEL

Lex Wolters*†, Robert van Engelen‡
High Performance Computing Division,
Dept. of Computer Science, Leiden University
P.O. Box 9512
2300 RA Leiden, The Netherlands
{llexx,robert}@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

This paper describes the current status of our research on data parallel implementations of the complete HIRLAM forecast model with both gridpoint and spectral dynamics, on massively parallel architectures. It is part of a research project, that investigates the possibilities to apply parallel computer systems for routine numerical weather forecasting.

1 Introduction

In our research to investigate the (dis)advantages and the (in)efficiency of data parallel implementations of the HIRLAM forecast model on massively parallel computer systems, we have achieved new results compared to the situation described in [9]. This paper summarizes these results, of which some are preliminary.

The HIRLAM system [3, 6] was developed by the HIRLAM-project group, a cooperative project of Denmark, Finland, Iceland, Ireland, The Netherlands, Norway, and Sweden. The model, which is used for producing limited area numerical forecasts, is in use at several of the meteorological services participating in the HIRLAM project for their routine weather forecasting.

*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.

‡Support was provided by the Foundation for Computer Science (SION) of the Netherlands Organization for Scientific Research (NWO) under Project No. 612–17–120.

On the basis of the achieved performances in full production runs several issues will be discussed. After a short introduction in section 2 about data parallelism, section 3 compares the parallel efficiency of numerical methods (semi-implicit versus fully-explicit, gridpoint versus spectral). Some execution profiles will be presented in section 4 and compared to one obtained on a traditional vector-architecture. Section 5 deals with non-numerical topics, like pre/post-processing and I/O. Preliminary results with the semi-Lagrangian formulation will be presented in section 6. Finally, section 7 shows first results concerning the relation between the achieved performance and the selected data-distribution on a MasPar and a CRAY T3D system.

It should be clear that the goal of this paper is to provide the reader with a global overview of our results only. It falls outside the scope of this paper to include all results and to describe all details. However, we will give for each topic in the corresponding section one or more references to publications in which a complete and detailed presentation of that specific topic can be found.

2 Data Parallelism

Data parallelism is one of the programming models to exploit parallel systems. It is based on the fact that data are distributed in some way over a possibly restricted set of processors. On current parallel platforms this is achieved by including compiler directives in the source code or by using a default distribution. Parallel code segments are indicated by array or vector syntax, e.g. Fortran 90 syntax [1]. If during the parallel execution a processor needs data that reside on an other processor, communication is required. However, within the data parallel programming model no calls to communication primitives have to be included in the code, since the compiler will take care of that. This is known as implicit communication. It is one of the advantages of data parallelism, in particular compared to programming models with explicit communication, where the programmer has to include send and receive statements or equivalent primitives. The requirement that one should use array or vector syntax is not a real drawback, because powerful tools are available to transform Fortran 77 code to Fortran 90 code. From a point of maintenance this an important observation.

Data parallelism has two possible disadvantages. The first one concerns the fact that the resulting efficiency of the program depends highly on the capacities of the compiler to generate efficient code. In programming models with more explicit communication (e.g., message passing), the programmer can take advantage of his knowledge about the program and include the most efficient communication primitives at the best locations in the code. However, it can be expected that this code with message passing will be more difficult to maintain, in particular if the same code will be adapted and modified by several programmers. The second disadvantage of data parallelism is more serious. Currently this programming model can only be applied in problems which lead to regular data distributions. Irregular data distributions,

e.g. in finite element codes, can not be handled efficiently in this model. Fortunately, many numerical weather prediction codes are based on finite difference or spectral methods, that lead in general to very regular data distributions.

In our investigation we have used mainly MasPar systems. These systems have a SIMD architecture with 1024–16384 processors. The processors are physically and logically arranged in a two dimensional grid with wrap-around. Basically, data parallelism is the only programming model available for MasPar systems. Two types of communications are implemented: Xnet communication for nearest neighbor communications and Router communication for communication between arbitrary processors. For more detailed information concerning MasPar platforms the reader is referred to [7].

After our – still ongoing – research activities on Maspar systems, we also started to investigate the MIMD CRAY T3D parallel system. A T3D system contains from 32 up to 2048 Dec Alpha processors. These processors are physically distributed in a torus, but can be mapped logically in any grid structure. As programming model the T3D supports data parallelism, work sharing, message passing based on PVM, and a so-called shared memory model. More details about the CRAY T3D systems can be found in e.g., [2].

3 Different Numerical Methods

A modern atmospheric model such as HIRLAM, consists of two main parts. The first is called the 'dynamics', which solves the basic model equations known as the primitive equations (see e.g., [4]). The second part is called the 'physics', which describes the aggregate effect of the physical processes with scales smaller than the model resolution, on the larger, resolved, scales. Some physical processes like radiation, not directly described by the basic model equations, are also parameterized.

To solve the primitive equations in the dynamics, HIRLAM provides different numerical methods: fully explicit versus semi-implicit techniques, and gridpoint versus spectral methods. The physics is independent of the applied numerical method in the dynamics. In this section we show a few results concerning the comparison of these methods with respect to their data parallel implementation on a MasPar MP-2. More results and backgrounds on this topic can be found in [10, 11, 12]. In [11] a detailed description of the programming techniques is given.

In table 1 the resulting execution times for one time step using the different methods in the dynamics are shown. From this table we observe that the execution time for the fully explicit gridpoint method is the smallest. However, this method requires a five times smaller time step than the other two methods, which are therefore favorable. Due to this significant differences in time steps, we should drop the fully explicit method, despite the fact that the semi-implicit gridpoint and spectral formulations depend on global communications while the fully explicit gridpoint formulation needs only nearest-neighbor communications.

Table 1: Elapsed time (in sec) for the dynamics per time step on a MasPar MP-2 with 4K processors.

Method	Dynamics
Fully explicit gridpoint	0.14
Semi-implicit gridpoint	0.29
Spectral	0.39

To compare the semi-implicit gridpoint method and the spectral version in more detail, we present in table 2 some execution times for one time step achieved by both methods. Before drawing some conclusions based on this table, we should make the following remarks. The spectral method in a limited area model required a so-called extension zone (see [5]) to obtain periodicity and to allow the use of efficient Fast Fourier Transforms (FFT). The gridsizes for the spectral model given in table 2 include this extension zone. The actual sizes of the spectral integration area are equal to $50 \times 50 \times 16$ and $110 \times 100 \times 16$. A second point is related to the fact that in the HIRLAM gridpoint model the two- and three-dimensional fields are stored as one- and two-dimensional arrays, respectively, by combining the two horizontal dimensions to one. The code for the spectral model dynamics is more recent than the gridpoint model code. The two- and three-dimensional fields are stored indeed as two- and three-dimensional arrays, respectively. To couple this code to the physics routines, which are identical to those in the gridpoint model, the arrays must be re-dimensioned by the Fortran 90 intrinsic function 'reshape' [1] before and after physics.

From table 2 one can conclude that the elapsed times for executing the dynamics, which is the essential difference between the two models, show that the spectral model takes 30–70% more time than the gridpoint model. Furthermore, the scalability for the gridpoint model is quite well both with respect to the number of processors and to the number of gridpoints. For the spectral model we observe a non-linear increase in computing time for the FFTs as function of the number of horizontal points. Finally,

Table 2: Elapsed execution times (in milliseconds) using various MasPar MP-2 configurations for one time step by the semi-implicit gridpoint and spectral models on different grid sizes. The additional time required for the 'reshaping' of the data in the spectral model (see text) is given between parenthesis.

# proc.	Grid size	Dynamics		Physics
		Gridpoint	Spectral	
1K	$64 \times 64 \times 16$	1052	1569	773 (+140)
4K	$64 \times 64 \times 16$	291	390	209 (+0)
16K	$128 \times 128 \times 16$	302	472	204 (+0)

the necessary 'reshaping' of the data structures in the spectral model require extra time, which is given between parenthesis in table 2, and becomes substantial when the number of datapoints is larger than the number of processors.

4 Execution Profiles

In this section several execution profiles of the semi-implicit gridpoint model on a MasPar MP-2 are discussed and compared to the profile obtained on a one processor CRAY C90. For more detailed information and other results the reader is referred to [12].

The calculation of the semi-implicit corrections requires the solution of a set of Helmholtz equations. In the HIRLAM reference code this Helmholtz solver is based on a direct method, which consists of a Fourier sine-transform in the east-west direction, followed by a Gaussian-elimination in north-south. Several constraints within the applied Fourier sine-transform algorithm result in an extension of the integration area by several rows and columns. It is clear that this is not advisable for a massively parallel system. Therefore we replaced the direct Helmholtz solver by an iterative solver based on the Conjugate Gradient (CG) method.

Table 3 shows execution profiles for the semi-implicit gridpoint dynamics. The percentages of the total elapsed time for one time step are presented for the different components: to calculate the explicit dynamical tendencies, to carry out the horizontal diffusion, the total costs for the semi-implicit corrections, the percentage for other routines (e.g., time filter, time stepping, boundary relaxation, extension of fields), and to perform the physics. The percentage for the CG algorithm is presented separately. Concerning the Cray profile it should be mentioned that the profile is almost independent of the grid size, and it is based on an implementation of the HIRLAM reference code, and therefore contains the original Helmholtz solver.

From table 3 one can observe the following facts: the calculations of the dynamical tendencies and the horizontal diffusion are relatively cheaper using configurations with more processors for the same grid; this does not hold for the semi-implicit costs, in particular for CG. Comparing the MasPar and Cray profiles shows that on the

Table 3: Percentages of execution times using various MasPar MP-2 configurations split into the different components of the semi-implicit gridpoint version.

system, # processors	Grid size	Dyn. tend.	Hor. diff.	Semi-implicit		Others	Phys.
				Total	CG		
MP-2, 1K	$64 \times 64 \times 16$	13	7	22	10	16	42
MP-2, 4K	$64 \times 64 \times 16$	9	4	30	17	15	42
MP-2, 4K	$128 \times 128 \times 16$	13	7	23	11	15	42
MP-2, 16K	$128 \times 128 \times 16$	9	4	32	19	14	42
C90, 1 proc.	any	17	5	11	na	7	60

MP-2 58% and 42% of the total time is spent in the dynamics and in the physics, respectively, while for the C90 these numbers are 40% and 60%; the costs for the semi-implicit corrections are significantly smaller on the C90 compared to the results on the MP-2; on the MasPar the 'other' routines contribute considerably more to the total elapsed time than on the Cray (15% versus 7%).

Finally, some remarks concerning the efficiency on MasPar systems and a CRAY C90. On a MasPar MP-1 one can achieve in the dynamics part $\approx 50\%$ of the theoretical peak-performance. For the different physics routines this number varies from 53% to 86%. On a MP-2 the dynamics routines have an efficiency between 19% and 28%, while for the physics one obtains 30–52%. A one processor CRAY C90 results in an overall efficiency of 45%. See for more details [11, 12].

5 Non-numerical topics

This section deals with two aspects of a full HIRLAM production run that have not been addressed yet: I/O and pre/post-processing. For all details the reader is referred to [11, 12]. Examples of input are the initial and lateral boundary data. The output consists for instance of the calculated fields. All these data are stored in the standard GRIB (gridded binary) format. Pre- and post-processing routines transform these GRIB files into internal computer words and vice versa.

To investigate the influence of these non-numerical issues on a massively parallel system we executed several full 6-hour production runs with the semi-implicit gridpoint HIRLAM model on different MasPar configurations. The resulting elapsed times are presented in table 4. This table demonstrates that the execution time of the pre- and post-processing routines together with the I/O routines dominates the total execution time. This is mainly due to the fact that these routines are not parallelized and therefore are executed sequentially on the front-end of the MasPar.

Table 5 shows some results if one makes the effort to parallelize the routines involved with I/O and pre/post-processing. A comparison with a 4-processor Convex C-3840 is also presented. From this table is can be concluded that the paralleliza-

Table 4: Total elapsed times (in sec) to complete a 6-hour production run with the semi-implicit gridpoint HIRLAM model on different MasPar configurations. This time is split into two parts: one part shows the time spent in the pre/post-processing and I/O routines, and the other part denotes the time for the actual forecast.

MP-2 # proc.	Grid size	Pre- and post-proc. & I/O	Forecast	Total time
1K	$64 \times 64 \times 16$	145	106	251
4K	$64 \times 64 \times 16$	91	33	124
4K	$128 \times 128 \times 16$	445	109	554
16K	$128 \times 128 \times 16$	292	34	326

Table 5: Profiling of pre/post-processing versus model integration.

Stage in forecast	MasPar MP-2, 16K	Convex C-3840, 4 proc.
Input, unpacking, and pre-processing	8.7%	3.7%
Time integration	83.1%	92.9%
Post-processing, packing, and output	8.2%	3.4%

tion of the I/O and pre/post-processing routines leads to acceptable results for full operational forecast runs on the MasPar MP-2. However, compared to the results on the Convex the MP-2 percentages are still high. It should be noticed that these percentages are strongly dependent on the amount of data, in which the modeller is interested. This can be specified by the user. In general one can conclude that the role of these non-numerical issues on the performance of full production runs on parallel systems should not be underestimated.

6 Semi-Lagrangian Formulation

The semi-Lagrangian formulation is one of the methods to increase the time step significantly compared to the more common Eulerian formulations. Therefore it has become very attractive for numerical weather forecasting. A detailed physical and mathematical description of the semi-Lagrangian methods can be found in the review article [8]. The HIRLAM forecast model contains a two-time-level semi-Lagrangian method with different interpolation routines. We have implemented this method including a linear/cubic interpolation on a MasPar MP-2.

A detailed description of the different implementations falls outside the scope of this paper. The reader can find all details in [13]. We restrict ourselves to providing a global idea on the problems in implementing this method. Also some preliminary results are presented in this section.

From a numerical point of view, semi-Lagrangian methods consist of three steps (see also figure 1). In the first step the trajectory of each particle is determined. 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.

The basic problem in implementing these methods on a massively parallel computer are the irregular communications, that result from the interpolation routine. Based on the fact that in the data parallel HIRLAM model on MasPar all horizontal

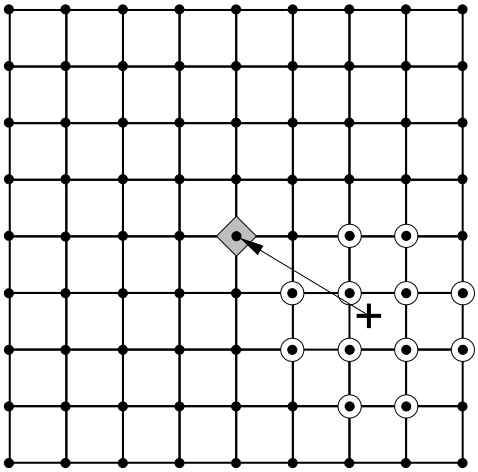

Figure 1: Calculation in a semi-Lagrangian formulation, projected on the horizontal plane. The arrow shows the trajectory from the departure point to the arrival point, which is a grid point. The grid points acting as interpolation points in a linear/cubic interpolation are encircled.

grid points are distributed over the two-dimensional processor grid in a cyclic (so non-blocked) and cut-and-stack way, one can describe the problem from a computational point of view as follows: 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 form a 9×9 square of processors with the 'current' processor as middle-point, which can also been seen from figure 1 (see for all details [13]).

With these observations in mind several implementations are possible. The most simplest one is to use the implementation from the reference HIRLAM code. This is called the original version. In the reference code this means that 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 within this data parallel 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. It turned out that the total extra time for this complete implementation of the semi-Lagrangian method is 3.50 s per time step.

In other versions we tried to control the random behavior of the communications and indirect addressing. The performance of these implementation was disappointing (see [13]), except for one. This version is based only on nearest-neighbor communications (called Xnet communications on MasPar, see section 2). Therefore we call it the Xnet version. The idea is to collect all possible $9 \times 9 \times k$ interpolation values on the processors corresponding with an arrival point of a trajectory, where k is the number of vertical levels. This can be realized by applying two series of cshift-routines [1]

Table 6: 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
Xnet version	0.35
Eulerian time step	0.68

(see [13]). It results in an extra execution time of 0.35 s per time step.

In table 6 the resulting extra execution times 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 [10]. As mentioned before the semi-Lagrangian formulation allows a larger time step than the Eulerian formulation. In the HIRLAM model we can take a three times larger time step. This is related to the physics part of the model. Observing the additional costs for the semi-Lagrangian method, as can be seen in table 6, it is clear that the 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.

It can be concluded that the semi-Lagrangian formulation in numerical weather prediction models can be implemented successfully on a massively parallel computer system. However, two disadvantages can be observed: 1) the increased memory usage per processor; 2) the interpolation routine is not portable anymore.

7 Data-distributions

In this section we present some first results concerning the question how the execution time depends on the distribution of the data over the processors. The different distributions in HIRLAM can be realized in two ways. Firstly, as mentioned earlier, in the reference HIRLAM gridpoint model the two- and three-dimensional fields are stored as one- and two-dimensional arrays, respectively, by combining the two horizontal dimensions to one. We call it the 1D model. To simplify the use of data parallelism we have created a modified code, in which the horizontal dimensions are not combined. This code is called the 2D model. These two models will in general result in different data-distributions.

The second method is based on two well-known ways to distribute data over processors: cyclic and blocked distributions. In the cyclic distribution the first element of an array is placed on processor 1, the second element on processor 2, and so on. If the number of elements is larger than the number of processors, one starts again with processor 1. This is known as the cut-and-stack technique. In a blocked distribution one divides the number of elements by the number of processors, and on each processor a block of elements is placed, where the number of elements in one block is equal to the outcome of the division. Hybrid methods are also known, both for one dimension and for more dimensions. However, they are not applied in our investigation.

The influence of the different distributions on the execution time has not been determined for the complete HIRLAM model yet. We restricted ourselves to one time-consuming routine with relatively many communications: DYN. This routine calculates the explicit dynamical tendencies of several variables by applying finite difference methods on the partial differential equations. This means that the communications in DYN are between nearest-neighbor processors.

The figures 2 and 3 show the resulting execution times on a MasPar MP-1 with 32×32 processors for cyclic and blocked distributions in the 2D and 1D model, respectively. In these figures the x-axis gives the gridsize, which is defined as the number of gridpoints in one horizontal direction. A gridsize of 64 means an integration area of 64×64 gridpoints. The number of vertical levels is equal to 16 in all tests.

Figure 2 shows a pattern, that could be expected for the 2D-version on a SIMD architecture like MasPar. Since we have 32×32 processors we observe an increase in execution time at gridsizes 33 and 65. At these gridsizes the data is stacked in 4 and 9 layers on 32×32 processors, respectively. However, the observed execution times are not increased by the same factors. Furthermore, there is no significant difference between the cyclic and blocked distribution.

The results for the 1D-version on the MasPar are presented in figure 3. We observe again two step functions, in which each step corresponds with a new data layer. For the block distribution the communications are more regular and less in number than in the cyclic distribution, which can also be seen in figure 3. Despite this fact the cyclic distribution is faster, which is due to the type of communication instructions generated by the compiler.

Comparing figures 2 and 3 leads to the conclusion that the 2D-version is not necessarily always faster than the 1D-version. The performance of the 2D-version is clearly more predictable.

We have implemented the same versions of DYN on a CRAY T3D with 8×8 processors. However, instead of data parallelism we had to use work sharing as programming model due to compiler bugs. Work sharing requires significantly more programming efforts, but is more flexible. With work sharing it is possible to simulate data parallelism. We have performed some tests, that show that the T3D compiler can generate more efficient code for a work sharing program than for program based

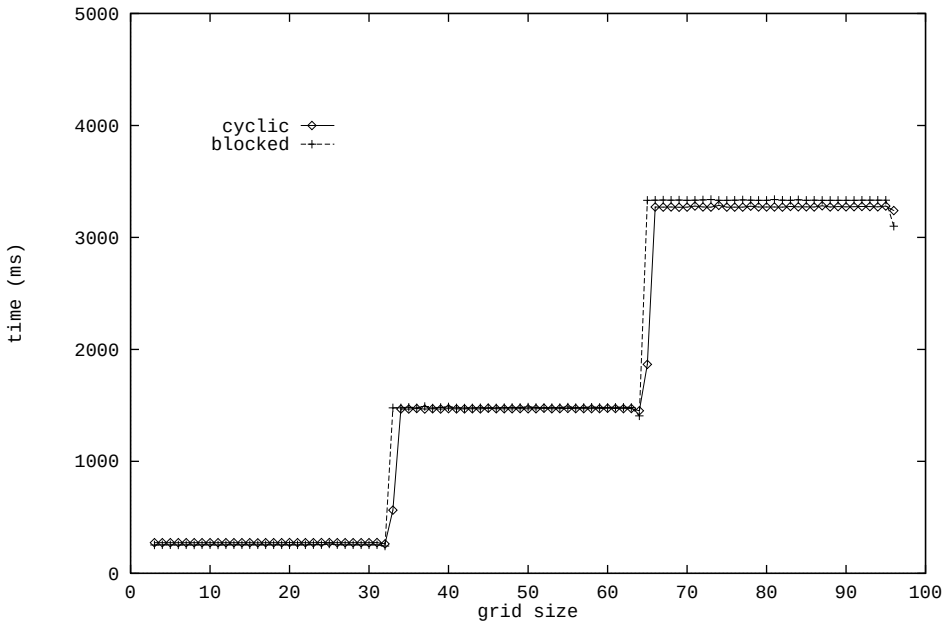

Figure 2: Execution times for the 2D version of DYN on a MasPar MP–1 with 32×32 processors using different data-distributions and varying the gridsize. See text for more details

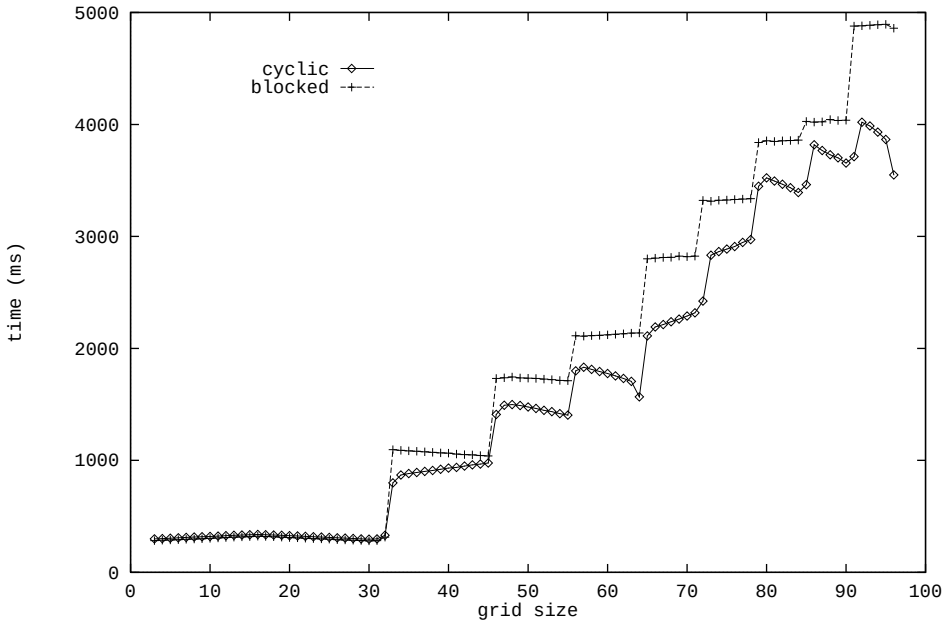

Figure 3: Execution times for the 1D version of DYN on a MasPar MP–1 with 32×32 processors using different data-distributions and varying the gridsize. See text for more details

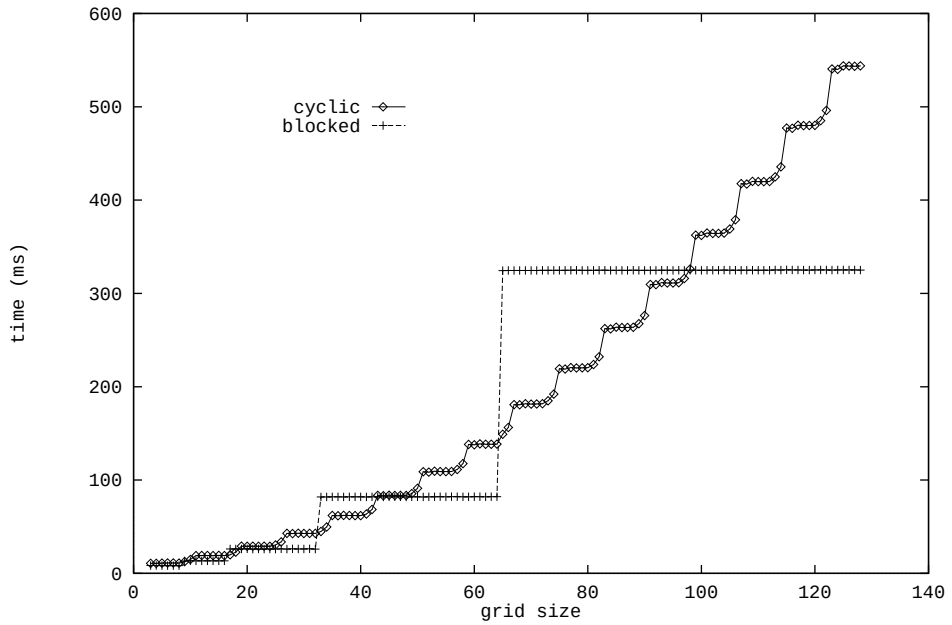

Figure 4: Execution times for the 2D version of DYN on a CRAY T3D with 8×8 processors using different data-distributions and varying the gridsize. See text for more details

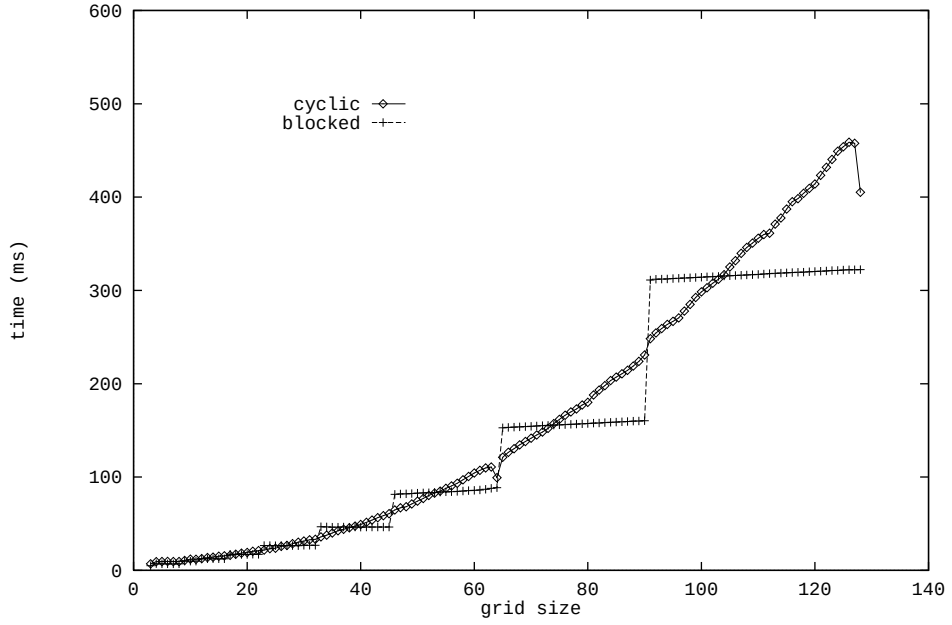

Figure 5: Execution times for the 1D version of DYN on a CRAY T3D with 8×8 processors using different data-distributions and varying the gridsize. See text for more details

on data parallelism ($\approx$ 20% more efficient). So, figures 4 and 5 show the obtained execution times for our tests based on the work sharing programming model. An important restriction of the T3D compiler is that the dimensions of distributed arrays should always be a power of two. Overall we see the same effects on the T3D as on the MasPar taken into account that the T3D contains only 64 processors compared to the 1024 processors on the MasPar. Therefore each processor on the T3D contains significantly more data points than on the MasPar. This investigation is not finished yet, so the figures show only preliminary results. An important factor that could influence these results considerably is the single node performance on the T3D. At the moment this performance is very disappointing.

8 Conclusion

The results in this paper show that the numerical weather forecast model HIRLAM can benefit from massively parallel computer systems. However, several topics have to be addressed in more detail, in particular I/O and data-distributions. Data parallelism can be successfully applied as programming model on MasPar systems. Based on preliminary results with the T3D it is clear that its compiler for this programming model is not efficient yet, in particular compared to other programming models available for this system.

Acknowledgments

This work was sponsored by the National Computing Facilities Foundation (NCF) for the use of supercomputer facilities, with financial support from the Netherlands Organization for Scientific Research (NWO).

References

[1] J.C. Adams, W.S. Brainerd, S. Walter, and J.T. Martin, *Fortran 90 Handbook*, Intertext, New York, 1992.

[2] Cray Research Inc., *CRAY T3D System Architecture Overview*, 1993.

[3] N. Gustafsson (editor), *The HIRLAM 2 Final Report*, HIRLAM Technical Report No. 9, 1993, (available from SMHI, S-60176 Norrköping, Sweden).

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

[5] J.E. Haugen and B. Machenhauer, *A Spectral Limited-Area Model Formulation with Time-dependent Boundary Conditions Applied to the Shallow-Water Equations*, Mon. Wea. Rev. 121 (1993) 2631–2636.

[6] B. Machenhauer (editor), *The HIRLAM Final Report*, HIRLAM Technical Report No. 5, DMI, Copenhagen, Denmark, December 1988.

[7] MasPar, *MasPar MP–1 Hardware Manuals*, July 1992.

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

[9] 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, 486–499.

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

[11] L. Wolters, G. Cats, and N. Gustafsson, *Data-Parallel Numerical Weather Forecasting*, to be published in a special issue of Scientific Programming on 'Application Performance Analysis'.

[12] L. Wolters, G. Cats, N. Gustaffson, and T. Wilhelmsson, *'Computing the Weather of Tomorrow in Parallel?'*, in Proceedings of the CWI-RUU Symposia on Massively Parallel Computing and Applications, CWI, Amsterdam, the Netherlands, to appear in a special issue of the IMACS Journal on Applied Numerical Mathematics.

[13] L. Wolters, G. Cats, N. Gustaffson, and T. Wilhelmsson, *Dataparallel Semi-Lagrangian Numerical Weather Forecasting*, to appear in the proceedings of Frontiers 95, the Fifth Symposium on the Frontiers of Massively Parallel Computation, February 6-9, 1995, McLean, Virginia, USA.