Data-Parallel 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`

Abstract

In this paper we describe the implementation of a numerical weather forecast model on a massively parallel computer system. This model is a production code used for routine weather forecasting at the meteorological institutes of several European countries. The modifications needed to achieve a data-parallel version of this model without explicit message passing are outlined. The achieved performance of different numerical solution methods within this model are presented and compared.

1 Introduction and overview of HIRLAM

Numerical models of the atmosphere have much contributed to our general understanding of atmospheric processes. The use of such models has resulted in improved weather forecasts, with important economical impact; also these models are now being used as components in climate simulation models.

The horizontal and vertical resolutions of atmospheric models are important factors determining the accuracy of the models. Present day computer power limits the number of gridpoints and thus the resolution to values that are unsatisfactory from a physics point of view. For example, the current resolution of local ('limited area') models is restricted to ≈ 50 km in horizontal direction, and to roughly 16 levels in the vertical, while the same models can be used for resolutions up to ≈ 5 km and 60 layers. The lower resolutions are also enforced since the calculations have to be completed within some reasonable elapsed time: for weather forecasting, the forecasts must be available within a fraction of the time that they may be considered valid; and for climate simulation, the simulated periods may cover several centuries, yet the calculations should be done within months. From these considerations it follows that continuous attempts are being made to run the models on the fastest available computer platforms. In this paper we investigate the role of massively parallel computer systems to achieve the required computer power for numerical weather prediction.

As early as 1922 L.F. Richardson described a 'forecast-factory' with 64,000 computers to calculate the weather for the whole globe [16]:

> Imagine a large hall like a theatre, except that the circles and galleries go right round through the space usually occupied by the stage. The walls of this chamber are painted to form a map of the globe. The ceiling represents the north polar regions, England is in the gallery,

[*]Support was provided by the Esprit EC Agency CEC-DGXIII under Grant No. APPARC 6634 BRA III.

the tropics in the upper circle, Australia on the dress circle and the antarctic in the pit. A myriad computers are at work upon the weather of the part of the map where each sits, but each computer attends only to one equation or part of an equation. The work of each region is coordinated by an official of higher rank. Numerous little "night signs" display the instantaneous values so that neighbouring computers can read them. Each number is displayed in three adjacent zones as to maintain communications to the North and South on the map. From the floor of the pit a tall pillar rises to half the height of the hall. It carries a large pulpit on its top. In this sits the man in charge of the whole theatre; he is surrounded by several assistants and messengers. One of his duties is to maintain a uniform speed of progress in all parts of the globe. In this respect he is like the conductor of an orchestra in which the instruments are slide-rules and calculating machines. But instead of waving a baton he turns a beam of rosy light upon any region that is running ahead of the the rest, and a beam of blue light upon those who are behindhand.

It is amazing to see from this quotation that despite the fact that Richardson's computers were humans, many important topics dealing with parallel computations are mentioned: data distribution, communications, loadbalancing, and synchronization. On the other hand we now know that Richardson's estimate of the required computer power is a gross underestimate: it should rather be in the Gflop/s range, see also section 4.

In this paper we will describe how these topics turn out for an existing numerical weather forecast model on a massively parallel computer system. The implementation and resulting performance of the HIRLAM forecast model on MasPar systems will be outlined. HIRLAM[1] stands for HIgh Resolution Limited Area Model, and is a state-of-the-art analysis and forecast system for numerical weather forecasts up to +48 hours. It is in use at several of the meteorological services participating in the HIRLAM project for their routine weather forecasting.

The HIRLAM forecast model [10] contains five prognostic variables: two horizontal wind components u and v, temperature T, specific humidity q, and surface pressure p_s. These variables are computed on a spherical horizontal coordinate system with longitude λ and latitude θ. The vertical coordinate is hybrid between pressure based and terrain following coordinates. The core and most computationally intensive parts of the model, like most modern atmospheric models, are the routines for the 'dynamics' and for the 'physics'. The dynamics solves a set of 3D coupled nonlinear hyperbolic partial differential equations (PDEs), the so-called primitive equations (see e.g. [5]). This set of equations models the behavior of the six prognostic variables in time and space. It contains two horizontal momentum equations, a hydrostatic equation, a mass continuity equation, a thermodynamic equation, and a continuity equation for water vapour. The physics describes the aggregate effect of the physical processes, with scales smaller than the model resolution, on the resolved scales. Examples are vertical diffusion and convection. Some physical processes like radiation and condensation, not directly described by the basic model equations, are also parameterized.

In concept, the forecast model solves identical equations of motion on a large number of gridpoints. Therefore, in theory, it should not be too difficult to code it for high efficiency on either scalar, vector or parallel computers. Current codes have almost invariably been optimized for vector architectures. With this research we intend to find out how much work is involved to convert vector code to data-parallel code; how cost-effective massive parallel machines can be for weather forecasting; and how meteorological models should be coded in future to achieve maximum portability between different hardware architectures (from SISD, SIMD to MIMD).

2 Description of the Algorithms

Within HIRLAM one can choose between several numerical methods in the dynamics part to solve the PDEs. In this section these methods will be compared mainly with respect to parallel issues.

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

Firstly, one can choose between the gridpoint HIRLAM model and the spectral HIRLAM model. These models are based on two numerical techniques, the finite difference or gridpoint technique and the spectral transform technique to solve PDEs. Both are commonly applied within the meteorological community. A third method, the finite element technique, has reached some popularity in recent years.

As an illustration of how the two most common techniques work, we will look at the simple example of the one-dimensional advection of temperature. In analytic form one has

$$\frac{\partial T}{\partial t} = u\,\frac{\partial T}{\partial x} \tag{1}$$

with temperature T and wind-component u in the x-direction at time t. In the most simple form (a non-staggered horizontal grid) the discrete gridpoint version with a leap-frog time-stepping results in

$$T(x,t+\Delta t) = T(x,t-\Delta t) + \frac{\Delta t}{\Delta x}\,u(x,t)\,\left(T(x+\Delta x,t) - T(x-\Delta x,t)\right)\,, \tag{2}$$

where $T(x,t)$ is the temperature in gridpoint x at time t, $u(x,t)$ is the wind-component in the x-direction in gridpoint x at time t, Δt is the time step, and Δx is the grid-distance in the x-direction.

For the spectral transform technique one will start the integrations from spectral coefficients $\hat{T}$ and $\hat{u}$, defined by e.g.

$$\begin{aligned} T(x,t) &= \frac{1}{\sqrt{2\pi}}\sum_k \hat{T}(k,t)\,\exp(ikx)\,, && (3)\\ \hat{T}(k,t) &= \frac{1}{\sqrt{2\pi}}\sum_x T(x,t)\,\exp(-ikx)\,, && (4) \end{aligned}$$

where k is the wave-number. The computation of non-linear terms is carried out in gridpoint space and the gridpoint values of the fields and their derivatives are obtained by inverse transforms, e.g.,

$$\frac{\partial T}{\partial x} = \frac{1}{\sqrt{2\pi}}\sum_k ik\,\hat{T}(k,t)\,\exp(ikx)\,. \tag{5}$$

Once the gridpoint values of u and $\partial T/\partial x$ have been obtained in gridpoint space, it is easy to carry out the multiplication and then to do a transform back to spectral space for $d\hat{T}/dt$.

It is important to realize that both the gridpoint and the spectral HIRLAM model use the same basic dynamical equations, the same vertical and temporal discretizations by finite differences and the same physical parameterization schemes. The differences between the methods concern the horizontal discretization and solution technique for solving the PDEs. In figure 1 an example of the horizontal integration area in the HIRLAM forecast model is shown. This example is based on a horizontal grid of 110×100 points.

The comparison of the gridpoint and spectral methods should be based on physical considerations. However, a discussion of these considerations (see e.g. [8]) falls outside the scope of this paper. Instead, we limit ourselves mainly to the computational differences.

Because the computation of non-linear terms and the physics part of the model require the fields to be available in gridpoint space, the spectral model requires transformations between spectral space and gridpoint space, and vice versa, each time step. The costs of these transformations are substantial. On multi-processor machines with physically distributed memory this is even more relevant, because the transformations require global communications, as can be seen from equations (3) and (4). In HIRLAM the transformations are Fourier transforms, and the cost-efficiency of the spectral model heavily depends on the availability of efficient library routines for fast Fourier transforms. In contrast, equation (2) shows that the gridpoint technique requires communications in the neighborhood of each gridpoint only.

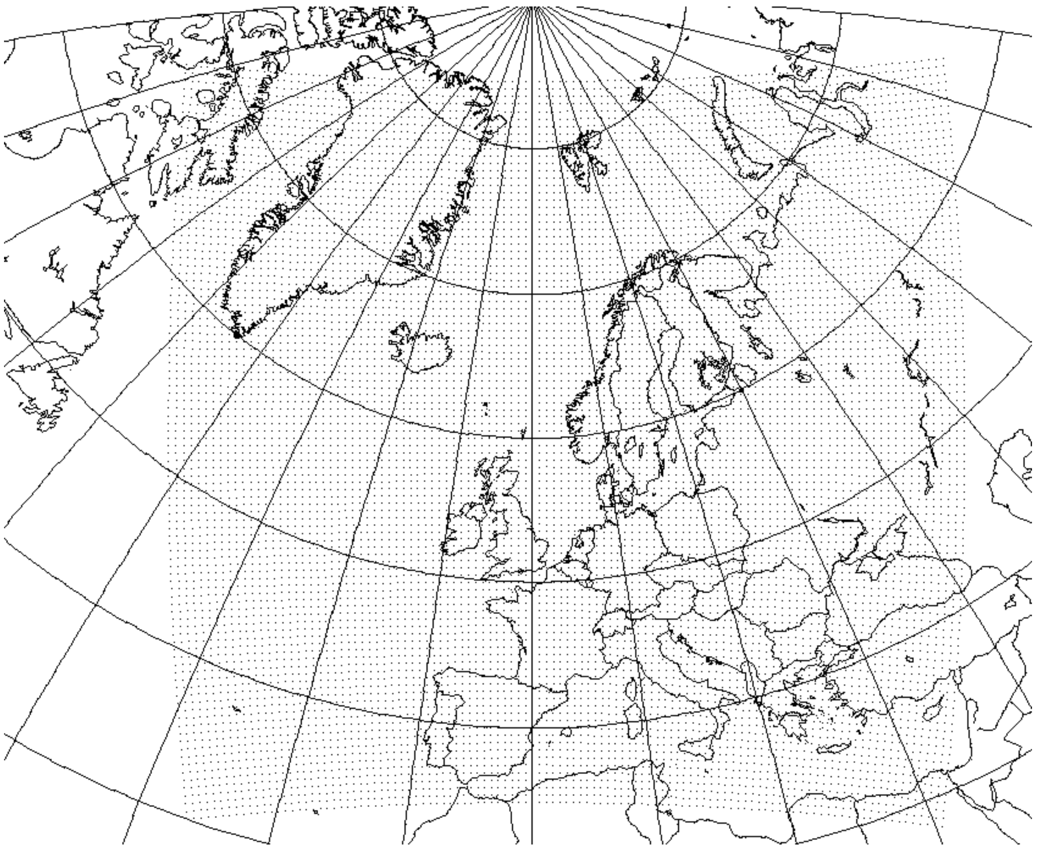

Figure 1: An example of a HIRLAM horizontal integration area with 110×100 gridpoints.

On the other hand the communications within the spectral model are limited to the spectral transforms, while for the gridpoint model they can in principle show up on all places where a finite difference is required. This means that a parallel gridpoint model contains communications throughout the complete dynamics. For the spectral model they are restricted to the transform routines.

The spectral transformations require periodic boundary conditions. This is straightforward for a global geometry, since this geometry allows for a natural periodic variation of all variables in both directions. Therefore the spectral transform technique has attained a great popularity for global numerical weather prediction [4, 14]. Initial steps to apply the spectral transform technique to a limited area were taken by Haugen and Machenhauer [8], who developed a spectral limited area shallow water model based on the idea of extending the limited area in the two horizontal dimensions in order to obtain periodicity in these two dimensions and to permit the use of efficient Fast Fourier Transforms. The same idea was implemented by Gustafsson [7] into the full multi-level HIRLAM framework.

There is no clear answer yet to the question whether it is preferable to apply the spectral or the gridpoint technique for a particular model configuration and for a particular computer architecture. Considering only computational accuracy, it is generally agreed that for a gridpoint model based on a second order horizontal difference scheme, the shortest waves that can be forecasted with a similar accuracy as in a spectral model are the four grid distance waves. The shortest wave in a spectral model generally corresponds to three grid distances in the transform grid. Thus, the number of horizontal gridpoints in a gridpoint model should be roughly twice ($\approx (4/3)^2$) the number of transform gridpoints in a spectral model to obtain a similar accuracy. There is not a similar difference with regard to the vertical coordinate, since spectral and gridpoint models normally use the same finite difference schemes in the vertical.

The time steps for both versions of the model are dictated by the stability of horizontal and vertical advection. In case of identical grid geometry, the spectral model is expected to require a shorter time step due to the inherent higher order accuracy of the computation of horizontal derivatives [11]. Taking physics, vertical advection and horizontal diffusion also into account, it has often been possible to apply spectral models with time steps similar to those of gridpoint models with the same grid distance [17]. In our performance comparison, we have taken the conservative approach of assuming the same 5 min time step for the spectral model with a 70 km transform grid as for the gridpoint-model with a 55 km grid.

With regard to computational efficiency of the spectral HIRLAM versus the gridpoint HIRLAM, the need for an extension zone in the spectral HIRLAM to obtain double periodicity should also be taken into account. This is not a problem on traditional vector computers with a shared memory or on MIMD computers programmed with explicit message communications, since all dynamics and physics calculations can be done in the non-extended 'real' computational area. In principle, this is true also for the data-parallel programming model on a parallel distributed memory architecture. As was the case in the present study, however, available software packages for Fast Fourier Transforms may put restrictions on the extended grid geometry, e.g., that the number of gridpoints should be a factor of two in each direction. Ideally, the number of gridpoints in the extended area should be about 25% larger than in the inner computational area (10% in each direction). This means that an equal percentage of processors may have to be applied to perform the extra calculations in this 'artificial' area.

Two more remarks could be made on the efficiency of spectral versus gridpoint models. First, in favor of the spectral model, all calculations in gridpoint space can be made strictly local since there is no horizontal staggering of the gridpoints. This also holds for the calculations in spectral space, since the coefficients of each wave-number can be treated separately. As a second point, when we move to larger number of horizontal points, say above 10^6, the gridpoint methods become relatively more efficient, since the computational time needed for Fourier-transforms increases faster than linearly in the number of points.

A second option the HIRLAM user can specify concerns the fully explicit or semi-implicit time-integration of the dynamics. Equation (2) is an example of an explicit scheme. As mentioned before this method results in nearest-neighbor communications in the horizontal directions. By application of semi-implicit corrections the integration scheme becomes more stable numerically, thus allowing longer time steps, and saving a factor of the order five in computational requirements. The reason for this is that the primitive equation model describes both fast (gravity) waves and slow meteorological (Rossby) waves. One is mainly interested in the last type, and with the semi-implicit method one can 'slow down' the gravity waves, which results in a longer time step. The calculation of the semi-implicit corrections requires the solution of a set of Helmholtz equations. On vector machines the additional costs for solving the Helmholtz equations are small. In gridpoint models, this is not necessarily true on distributed memory machines, since the Helmholtz solver requires global communications. In the spectral formulation, there are almost no costs associated with the solution of the Helmholtz equations, because the spectral components are the eigenfunctions of the Helmholtz operator.

A third option within HIRLAM is related to the Eulerian versus semi-Lagrangian description of the model. The semi-Lagrangian methods allows again longer time steps and gives a better description of variables with sharp gradients. On the other hand it requires a trajectory calculation and interpolation of several values. This results in substantial additional costs and local communications over a maximal distance of five grid points. Since the semi-Lagrangian method has still too many numerical problems, we will not investigate this further despite the fact that it gives rise to several interesting problems concerning the parallelization of these methods.

All options mentioned above affect the dynamics in the forecast model. Each of them results in different communication patterns. The other part of the forecast model, the physics, consists of routines for integration and filtering algorithms, etc. Almost all physical processes take place in one-dimensional vertical columns, without mutual communications. Therefore the model physics can be described as N disjunct processes, where N is the number of gridpoints in the horizontal.

Finally, with all these options it is interesting to note that the only method that is currently in use at several of the meteorological services participating in the HIRLAM project for their routine weather forecasting procedures is the semi-implicit Eulerian gridpoint method, run on either scalar or vector architectures.

3 Description of the Programming Techniques

In this section several implementation issues are discussed, that one encounters during the port of the original HIRLAM code to a MasPar system. The reference HIRLAM forecast model is coded in standard Fortran 77 and consists of about 28,000 lines. For an earlier and a more extended overview of implementation issues the reader is referred to [18].

First we need to mention some characteristics of the parallel architectures used in this investigation. These are the MasPar systems, which are also sold by Digital Equipment Corporation under the name of DECmpp systems.

Concerning the hardware, a MasPar system has a SIMD architecture with from 1,024 (1K) up to 16,384 (16K) processors, which are called processing elements (PEs). MasPar provides two models: the MP-1 and MP-2. The main difference between the MP-1 and the MP-2 configurations is the increase in peak performance by a factor five. The communication network is exactly the same for both model types. A full 16K MP-1 system has a peak performance of 26,000 Mips and 550 Mflop/s (64-bits) or 1,200 Mflop/s (32-bits). For a full MP-2 system these numbers are 68,000 Mips, and 2,400 Mflop/s or 6,300 Mflop/s, respectively. The PEs are controlled by the Array Control Unit (ACU), which is responsible for the instruction decode and broadcast of instructions and data. The PE-array and ACU form the Data Parallel Unit (DPU). A MasPar system needs a front-end, that serves as an interface to the DPU and is host for tools and compilers. In our case the front-end is a Dec 5000 workstation. For a detailed description of the systems see e.g. [12].

The programming model on the MasPar system is data-parallelism. For Fortran programs the MasPar Fortran (MPF) compiler is provided. This compiler is an implementation of Fortran 90. Operations on Fortran-arrays expressed by the Fortran 90 array-syntax will be executed in parallel, and the arrays involved will be distributed over the PEs. Operations with Fortran 77 syntax will be executed sequentially. It should be mentioned that this model does not contain any explicit message passing primitives. The compiler generates communication operations if necessary.

Let us now look at the data-parallel implementation of the HIRLAM forecast model. All model parameters are kept in core memory. The three-dimensional fields (temperature, wind, water vapour and liquid water) are stored as two-dimensional arrays; the first dimension runs over all horizontal gridpoints, the second over the layers in the vertical. The two-dimensional fields (surface pressure and several soil parameters like land-sea mask) are kept as one-dimensional arrays.

Basic idea. As mentioned before, the core of the model are the subroutines to carry out the dynamics and to carry out the physics. These routines are the most time consuming parts of the model on scalar and vector architectures. The other routines deal with I/O, pre- and post-processing, and do not count too much to the total execution time on scalar and vector systems. Therefore, we concentrate in our data-parallel implementation of the dynamics and physics routines. They should run on the DPU, while the pre- and post-processing and I/O routines are executed on the front-end. This means that at some point data has to be transferred from the front-end to the DPU and vice versa. With this basic idea in mind we will now discuss some implementation issues.

Data distribution. The first issue concerns the distribution of the data. On the one hand one has to deal with three-dimensional fields in the forecast model, and on the other hand the processors in the parallel system are organized in a two-dimensional mesh. As is stated in section 2 the dependencies, that could result in communications between the processors, in the physics-part of HIRLAM are almost exclusively in the vertical direction, in contrast to those in the dynamics, which are mainly in the horizontal directions. The number of dependencies in the physics-part is much larger than in the dynamics-part. Therefore, to minimize the number of communications we chose for a data distribution where the data are mapped on the two-dimensional processor-array by projection of the vertical dimension onto the horizontal plane. This distribution is quite common for this kind of application.

Compiler directives. A second issue concerns the inclusion of compiler directives in the original code. This is a result from the fact that the three- and two-dimensional fields are stored in two- and one-dimensional arrays, respectively, where the first dimension runs over all horizontal gridpoints. This means that for a distribution of the horizontal gridpoints over all processors, we have to use compiler directives, since the default mapping on a MasPar system maps the first dimension of a data array only in the x-direction of the processor-array, and the second dimension in the y-direction. With the MAP-directive the user can overrule this default mapping and specify the desired distribution.

To be more specific, the semi-implicit Eulerian gridpoint version of the HIRLAM model needs 944 directives to run on the MasPar system. This is nearly 4% of the total code, and therefore striking the question if the usage/inclusion of compiler directives can be considered as the introduction of a second program within the original code. On the other hand we should mention that the inclusion of these compiler directives has a historical reason, namely that the HIRLAM was optimized for vector architectures, and as a result contains two and one dimensional data structures instead of three and two, respectively. With the latter data structures most of the current directives would become redundant.

Interface blocks. This issue is strongly related to the previous one. An interface block is a Fortran 90 concept and is part of the specification part of a program unit. In an interface block the user can specify the names and properties of the dummy parameters for each subroutine and/or function called within that program unit. In most cases it contains the same declarations and specification statements as appear in the definition of the subroutine or function. For a more detailed description of interface blocks the reader is referred to e.g. [1]. In MasPar Fortran it is mandatory to use interface blocks for all subroutines and functions with dummy parameters mentioned in compiler directives, in particular the MAP-directive. Including these interface blocks in HIRLAM sometimes resulted in a fourfold increase of the number of source-lines of a subroutine.

Arrays as parameters. As a consequence of the distributed memory of a MasPar system one cannot assume a sequential address space. This means that the actual and formal parameter in a subroutine call should always have the same size and shape. So 'dirty' Fortran 77 tricks like passing only the first address of an array-portion with some arbitrary length or passing 1-dimensional arrays to routine, which require a two-dimensional parameter, and vice versa, are not allowed. Unfortunately, these tricks were heavily used in the original HIRLAM code.

Indirect addressing. Indirect addressing is another topic. Since memory addressing is part of an instruction, indirect addressing is often not possible on a SIMD architecture. However, on the MasPar one specific FORALL-statement allows the use of indirect addressing. As a result in some routines in HIRLAM such a FORALL-statement had to been included, since they contain statements with indirect addressing.

Fortran 90. In principle all Fortran 90 code was produced from the original Fortran 77 source code by using the MPVAST compiler. VAST transforms Fortran 77 code to Fortran 90 code. Since the HIRLAM code has been optimized for vector architectures, most of these transformations could indeed be obtained automatically by MPVAST. However, some Fortran 90 statements were added manually, partly because of the fact that VAST could not produce the most efficient code, partly because special constructs were necessary, like for indirect addressing.

Helmholtz solver. In the original HIRLAM code the Helmholtz solver for the semi-implicit time stepping 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 conditions in the application of the given Fourier sine-transform resulted in the fact that the integration area had to be extended by several rows and columns. Since it is clear that this is not advisable on a parallel system, the solver was rewritten to an iterative solver based on the Conjugate Gradient method [3].

Fourier transforms. The performance of the spectral HIRLAM is crucially depending on the availability of efficient FFT subroutines. The basic algorithms of the package for 'Super-Parallel' FFTs of Munthe-Kaas [13] were designed and developed for applications on SIMD computers. The term 'Super-Parallel' algorithms is used to denote "algorithms that in a SIMD fashion can solve multiple instances of similar problems, with a degree of parallelism that is in the order of the sum of the sizes of all the sub-problems", see [15]. The only restriction in the FFT package of Munthe-Kaas is that the sizes of the problems should be powers of two (in all dimensions in the case of multi-dimensional problems) and, in addition, that the data must satisfy certain alignment requirements with the address space in the computer.

Reshaping in spectral space. The code for the spectral model dynamics is more recent than the gridpoint model code. The two- and three-dimensional fields are stored 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.

Computations in spectral space. The organization of the computations in spectral space in the original spectral HIRLAM model was based on a re-sorting of all spectral coefficients to avoid unnecessary computations for spectral components that are to be truncated. This organization of the spectral computations would not have been efficient on the MasPar. Thus, the computations in spectral space were re-organized – all computations are done for all spectral components followed by an explicit truncation. In order to optimize the spectral model, FFT-routines based on scrambled spectral coefficients were utilized. This had the effect that a number of coefficients fields needed to be calculated in advance and scrambled to the same sorting order as the spectral coefficients.

Compiler problem. Finally, it turned out that the compiler generates redundant communications in several routines, especially in the physics-routines. A work-around for this problem was to make sure that array-dimensions and several loop-bounds were known at compile time. From a research point of view this is a serious restriction, but for a production code this would improve the performance on all kinds of platforms.

4 Realized Performance

In this section the performance of the different numerical methods for the forecast routines will be discussed and compared. At the end of the section issues like pre- and post-processing and I/O will be investigated.

First we provide some details concerning the hardware and software used in this investigation. The MasPar MP-1 system was a MasPar DPU Model MP-1104 (64 rows, 64 columns) with a DEC 5000/240 front-end, while the MasPar MP-2 system contained a MasPar DPU Model MP-2216 (128 rows, 128 columns). All tests were performed with system release 3.2.0 of the MasPar software, which included the Vast-II (version 3.06), and the MPF 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.

Gridpoint Model Results

We adopted as test runs the calculation of a 6-hour forecast on a $64\times64\times16$ grid and a $128\times128\times16$ grid, both at 55 km spacing. The semi-implicit gridpoint version with this resolution requires 72 time steps of 5 minutes. The small grid fits perfectly on a 4K (64×64) PE array, while for the large grid this holds for a 16K (128×128) PE array. On a PE array with fewer processors than horizontal gridpoints, the grid is split automatically in layers. If the number of gridpoints is not a multiple of the number of PEs, some PEs will be idle during part of the calculation on a SIMD architecture. Therefore the only sensible choice for the number of gridpoints in one horizontal level is a multiple of the number of processors. From physical arguments one would choose the highest feasible number of gridpoints for any chosen model domain, because that leads to the highest possible resolution. Alternatively, if modellers would choose to keep the resolution fixed, they would surely extend the model domain as much as possible, given the available computational power, so as to reduce the influence of the lateral boundary conditions.

In table 1 the elapsed times are presented to complete the 6-hour forecast on different MasPar configurations. It also contains a more detailed view by showing the elapsed time per time step, together with a break-down in the times needed for the 'dynamics' and the 'physics'. From this table several observations can be made. We want to mention the following points:

– Comparison of the corresponding timings for the 1K, 4K and 16K configurations per model show that the calculations scale well with respect both to the number of processors and to the number of gridpoints.

– The gridpoint version runs only a factor two faster on a MP-2 configuration than on the corresponding MP-1 model. This is clearly less than the theoretical factor five. A reason is the design decision to enhance the processor power in the MP-2 by a factor five, the memory bandwidth by a factor two, and no enhancements in the communication bandwidth with respect to the MP-1. These different factors make it difficult to predict the actual MP-2 performance compared to the MP-1. This can also be seen in table 1, where the ratio between the time for 'dynamics' and the time for 'physics' differs significantly on both systems. Remember, the dynamics contains many nearest-neighbor communications, while for the physics communication is much less important.

– The Mflop-rate for the most computationally intensive routines in the dynamics measures about 150 Mflop/s for the $64 \times 64 \times 16$ run on the MP-1 with 4K processors, which is 50% of the maximum rate. For the physics-routines in this run it varies from 160 up to 260 Mflop/s (53%-86%). For the $128 \times 128 \times 16$ grid size on a MP-2 with 16K processors, we found for

Table 1: Elapsed execution time (in sec) using various MasPar configurations for a 6-hour forecast with the gridpoint HIRLAM model on different grid sizes. Also the elapsed time (in millisec) for one time step with the break down into the time spent in the dynamics and in the physics.

Model and	Grid size	Forecast	1 Time step (in ms)		
# processors		(in s)	Total	Dynamics	Physics
MP-1 1K	$64 \times 64 \times 16$	286	3877	1972	1905
MP-1 4K	$64 \times 64 \times 16$	79	1047	552	495
MP-1 4K	$128 \times 128 \times 16$	291	3934	2020	1914
MP-2 1K	$64 \times 64 \times 16$	135	1825	1052	773
MP-2 4K	$64 \times 64 \times 16$	39	500	291	209
MP-2 4K	$128 \times 128 \times 16$	137	1841	1070	771
MP-2 16K	$128 \times 128 \times 16$	39	506	302	204

the most time consuming dynamics-routines a speed of 1200 up to 1750 Mflop/s (19%–28%), and for the physics-routines 1900 up to 3300 Mflop/s (30%–52%).

Spectral Model Results

Two operational data sets of the HIRLAM system were used for benchmarks on the MasPar MP-2 systems. For most of the tests, data from a horizontal area consisting of 110 × 100 gridpoints (128 × 128 in the extended area, see section 2) and with 16 vertical levels were utilized. The horizontal grid distance in this data set is approximately 70 km. In order to have a proper test of the smaller MasPar systems, a data set with 50 × 50 horizontal gridpoints (64 × 64 in the extended area) was used in addition. Also in this test a time step of 5 minutes was possible, so 72 time steps were carried out to obtain forecasts valid at +6 hour. In order to test the MasPar also on a larger data set, the 110 × 100 × 16 data set was interpolated horizontally to a data-set with 221 × 221 × 16 (256 × 256 × 16 in the extended area) transform gridpoints.

The total elapsed computing times for different HIRLAM spectral forecast model test runs on different MasPar sizes are contained in table 2. Again the elapsed time for each time step, with the break down into the time spent in the dynamics and the physics, for the different MasPar runs are also given. The following of more general interest could be noted about the results in table 2:

– In case the spectral model is run with four times as many processors for a particular transform grid size, the computing time decreases with a factor slightly greater than four. Thus, in this limited sense, also the spectral model scales well on MasPar. But that factor is much greater than four for the physics, since the performance of the reshape function, included in the physics timing (see section 2), becomes worse when the number of gridpoints exceeds the number of processors.

– Running on a particular processor configuration with a 4 times larger horizontal area (e.g., on 221 × 221 extended to 256 × 256 horizontal points as compared to 110 × 100 extended to 128 × 128 horizontal points) increases the computing time with a factor somewhat greater than four. This could be explained by the non-linear increase in computing time for the FFTs as a function of the number of horizontal points.

– The elapsed time for a 6-hour forecast with the spectral HIRLAM without I/O on the MP-2 with 16K processors was 49 seconds (grid size 110x100x16). The corresponding elapsed time on a single processor Cray C90 was 70 seconds and the measured computational speed was 450 MFlop/s. Redundant calculations in the extension zone as well as in spectral space were avoided in the Cray version of the code. It should be noticed that the computations on the Cray are performed in 64-bit and on the MasPar in 32-bit. Since we have not seen that 32 bits is insufficient for the HIRLAM model at the resolution studied here, the best comparison should be between a 32-bits MasPar and 32-bits Cray. However, this last machine is not available, so the Cray is too expensive for its purpose.

Table 2: Elapsed execution time (in sec) using various MasPar MP-2 configurations for a 6-hour forecast with the spectral HIRLAM model on different grid sizes. Also the elapsed time (in millisec) for one time step with the break down into the time spent in the dynamics and in the physics.

Model and	Grid size	Forecast	1 Time step (in ms)		
# processors		(in s)	Total	Dynamics	Physics
MP-2 1K	50 × 50 × 16	179	2482	1569	913
MP-2 4K	50 × 50 × 16	43	599	396	203
MP-2 4K	110 × 100 × 16	201	2786	1832	954
MP-2 16K	110 × 100 × 16	49	676	458	218
MP-2 16K	221 × 221 × 16	232	3218	2209	1010

Performance Comparison

In this subsection we compare the performance of the gridpoint versions, both semi-implicit and fully explicit, and the spectral versions of the HIRLAM model with respect to the pure forecast calculations. As explained in section 2 each method has its own characteristics resulting in different spatial and temporal resolutions. This makes a comparison not trivial. A general discussion about this topic can be found in [9]. Our strategy consists of comparing the execution times of the different methods for performing calculations on the same physical area during the same simulated time span with the same accuracy.

A first comparison between the gridpoint and spectral model can be based on the tables 1 and 2. In table 1 it was shown that a time step within the semi-implicit gridpoint version took 500 ms for a $64 \times 64 \times 16$ grid on a MasPar MP-2 with 4K processors. Of this time 291 ms was spent in the dynamics and the remaining 209 ms in the physics. For the semi-implicit spectral model with $50 \times 50 \times 16$ points, see table 2, the total time was 599 ms, divided in 396 ms for the dynamics and 203 ms for the physics. The reduction of the number of gridpoints from $64 \times 64 \times 16$ in the gridpoint formulation to $50 \times 50 \times 16$ is compensated by the higher intrinsic accuracy of the spectral method (see section 2). In this case the time spent in the physics is nearly equal for both methods. Note, however, that this is particular to the chosen grid configuration; comparing tables 1 and 2 shows that in case the number of gridpoints exceeds the number of processors the inefficiency of the reshape function (see section 3) makes the spectral model rather more expensive.

A better comparison would probably be based on the performance of a HIRLAM production run. In operational gridpoint and spectral implementations with 55 km and 70 km resolution, respectively, the dynamics can be calculated with time steps of 5 minutes because of numerical stability (see section 2). For the physics a larger time step can be chosen, namely 15 minutes. In addition to the dynamics and the physics, a third component in a production forecast can be distinguished, which is called the statistics. It consists of routines that calculate diagnostic information about changes in pressure, wind-speed, etc. This information is usually requested every dynamics time step, which results in a time step of 5 minutes for the statistics.

Taking these facts into account we can calculate the total averaged costs for the 6-hour forecast on a MasPar MP-2 system with 4K processors. The execution time to obtain the statistics has been measured to be 50 ms on the MP-2 with 4K processors.

To get an impression of the efficiency of a fully explicit method, which contains nearest-neighbor communications only, we measured the time for the dynamics without the semi-implicit correction. A time of 142 ms was found. However, due to the stability of this numerical method a time step of one minute had to be chosen for the dynamics, which results in 360 time steps for a 6-hour forecast. For the physics and statistics nothing changes in the fully explicit formulation compared to the semi-implicit method.

Table 3 shows the resulting execution times to produce a 6-hour forecast with the three numerical methods. From these execution times we can first conclude that despite the fact that although the semi-implicit gridpoint and spectral formulations depend on global communications and the fully explicit gridpoint formulation needs only nearest-neighbor communications, the first two methods are favorable. This is due to the fact that these methods allow a five times larger time step. Comparing the execution times of the semi-implicit gridpoint method and the spectral version shows that the semi-implicit is the fastest way to calculate the 6-hour forecast. However, the difference is not very significant contrary to the fact that the number of global communications

Table 3: Execution times (in sec) to calculate a 6-hour forecast on a MasPar MP-2 with 4K processors. See text for details.

Method	Dynamics	Physics	Statistics	Total
Semi-implicit	$72 \times 0.29 = 20.9$	$24 \times 0.21 = 5.0$	$72 \times 0.05 = 3.6$	29.5
Spectral	$72 \times 0.40 = 28.8$	$24 \times 0.20 = 4.8$	$72 \times 0.05 = 3.6$	37.2
Fully explicit	$360 \times 0.14 = 50.4$	$24 \times 0.21 = 5.0$	$72 \times 0.05 = 3.6$	59.0

Table 4: Total elapsed times (in sec) to complete a full production with the semi-implicit gridpoint HIRLAM model on different MasPar configurations. Also the differentiation into the pre- and post-processing time and into the actual forecast time are shown. The pre/post-processing time is split into time for the front-end to back-end communications and visa versa (denote as copy), and the time spent in front-end calculations and I/O (denoted as front-end).

Model and	Grid size	Pre- and post-processing			Forecast	Total
# processors		Copy	Front-end	Total		time
MP-1 1K	$64 \times 64 \times 16$	42	63	105	217	322
MP-1 4K	$64 \times 64 \times 16$	24	60	84	66	150
MP-1 4K	$128 \times 128 \times 16$	133	202	335	222	557
MP-2 1K	$64 \times 64 \times 16$	86	59	145	106	252
MP-2 4K	$64 \times 64 \times 16$	33	58	91	33	124
MP-2 4K	$128 \times 128 \times 16$	249	196	445	109	554
MP-2 16K	$128 \times 128 \times 16$	100	192	292	34	326

is considerably larger in the spectral formulation. One reason for this is the highly optimized FFT package that has been used for the spectral model. This also shows an advantage of the spectral method with respect to parallelization aspects. As explained, all inter-processor communication in the spectral model occur within the FFT package. So if one wants to reduce the communication overhead, one could concentrate on this package, while in a gridpoint model inter-processor communication is spread throughout large parts of the code (see also [2, 6]).

In the comparison above, the positive effect of the improved accuracy of the spectral model was assumed to be equal to the negative effect of the need for an extension zone.

Production Runs

Until now we have only considered the performance of the routines executing the computations for a forecast. However, a full production run does not consist of calculations only. In a HIRLAM production run we need first of all initialization fields for the various physical variables. These values are available as files on disk, and are read during the input phase at the start of the run. Secondly, since we deal with a limited area model, there will be an input phase for new boundary values every 6 hours. Furthermore, every hour the values of several fields are written to disk to obtain information about the changes of the variables. These issues can be considered as the I/O phase, which exists in every computer program.

However, all data on disk is stored in a standardized, machine-independent format. As a result this data needs to be transformed to/from this standard. This is also a part of the full production run. Besides, there are several other transformations from the raw calculated variables into other information, that is useful to produce a weather forecast.

To investigate the influence of these issues for 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. To get the timings for an actual production run, which simulates 36 or 48 hours, one should multiply the numbers in this table by 6 or 8, respectively, ignoring the few seconds required for initialization.

In our implementation all the 'extra' issues within a full production run mentioned in this section are executed on the front-end of the MasPar system. We consider them as the pre- and post-processing phases of the production run. As a result of our implementation the data has to be copied between the DPU (back-end) and the front-end. Also these copies are assumed to be part of the pre- and post-processing phases.

From table 4 one can draw the following conclusions:

– The pre- and post-processing time dominates the total execution time in several runs or counts for a considerable part to it.

- The ‘copy’-timings do not scale with the amount of data. This is mainly due to the fact that, in case one does not use the full number of processors on a MasPar system, the data is replicated to the subsets of unused processors. This is embedded in the MasPar runtime system to provide nearest-neighbor communications even if not all PEs are used in particular for circular shifts (communications with toroidal wrap-around). The overhead for these replications is related to the ratio between the number of physical PEs on the system and the number of PEs that is actual used. In our results the only timings without this overhead are those obtained with 4K and 16K processors on the MP-1 and MP-2, respectively.
- The ‘front-end’ timings for each grid size are nearly equal on the various configurations.

Based on the achieved results it seems that parallel computing is not useful for numerical weather forecasting. However, the observed ‘overhead’ is mainly due to the basic idea of our implementation, see section 3, namely to concentrate on the parallelization of the most time consuming routines on vector and scalar architectures. These are the routines where all calculations are performed. On a parallel machine all routines become important, which is a direct consequence of Amdahl’s law. There are several ways to improve the observed overhead:

1. Replace the front-end by a faster machine with better I/O capacities. This will affect the front-end calculations.
2. Execute the pre- and post-processing on the DPU. A large part of these computations can be executed in parallel. This is currently under investigation. First results show that the execution time for pre- and post-processing can be reduced to less than 10% of the total execution time. However, this requires obviously extra implementation efforts.
3. Dump the calculated values of the desired variables directly from the back-end on disk in a raw form. MasPar systems provide several hardware and software improvements to read or write data directly by the DPU from or to disk in a fast way. Subsequently, a second process could perform all the pre- and post-processing concurrently with the forecast calculations.

5 Conclusions

To conclude a summary of the main results:

- The semi-implicit and spectral versions of the HIRLAM model are preferable to a fully explicit gridpoint version, despite the global communications needed versus the nearest-neighbor communications.
- The semi-implicit gridpoint version results in a higher performance than the spectral version.
- Copying data into and out of a parallel system is time-consuming. This issue should not be underestimated on a parallel architecture.
- The algorithms for numerical weather forecasting used in this application can be executed efficiently. In the gridpoint version they give evidence of a good scalability both to the number of data points and to the number of processors. In the spectral version the algorithms are scalable with respect to the number of processors. The non-linear increase in computing time for the FFTs as function of the number of horizontal data points prevents scalability with respect to the number of data points.
- The effort to port an application like HIRLAM to this class of parallel architecture is quite considerable. However, this was also true for vector platforms when they entered the market. Furthermore, for the gridpoint version other classes of parallel architecture require even more effort due to the fact that communications can show up on many places. This does

not necessarily hold for the spectral model, since all communications are restricted to the transform routines.

It is clear that parallel computer systems could play an important role in numerical weather forecasting and climate simulations. Based on the results shown in this paper there is hope that the computer power offered by parallel systems can be utilized efficiently for these kinds of applications. For an example of the resulting cost/performance relation the reader is referred to [19].

Furthermore, this investigation shows that the applied programming model, data-parallelism without explicit message passing, can lead to acceptably efficient code for this application area. This is mainly due to the fact that we deal here with regular computations. For irregular cases, like sparse computations, present day compilers are frequently not smart enough to produce efficient code. The role of the underlying hardware with respect to data-parallelism without explicit message passing is not clear yet. The MasPar implementation is currently ported to the CM-5 of Thinking Machines Corporation. Also the T3D of Cray will be investigated. It is expected that the current version can be easily ported to these systems. How efficient the resulting implementations will be is still an open question.

Finally, it is interesting that the idea (see section 1) proposed by Richardson far before the invention of what we call computers now, can be realized! Fortunately we do not need thousands of humans to implement it, some sophisticated hardware with accompanying software will do the job.

Acknowledgements

We thank Ben Wichers Schreur (KNMI) for providing us the figure of the HIRLAM integration area. We also thank the anonymous reviewer and the guest editor for their comments, by which the clarity and the contents of this paper have been improved.

References

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

[2] S.R.M. Barros and T. Kauranne, *On the Parallelization of Global Spectral Weather Models*, submitted to Parallel Computing, 1993.

[3] J.W. Demmel, M.T. Heath, and H.A. van der Vorst, *Parallel Numerical Linear Algebra*, Acta Numerica 1993, pp. 111-197, Cambridge.

[4] E. Eliassen, B. Machenhauer, and E. Rasmussen, *On a Numerical Method for Integration of the Hydrodynamical Equations with a Spectral Representation of the Horizontal Fields*, Report No. 2, Institut for Teoretisk Meteorologi, University of Copenhagen, 1970.

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

[6] U. Gärtel, W. Joppich, and A. Schüller, *Parallelizing the ECMWF's Weather Forecast Program: The 2D Case*, Technical Report 740, GMD, Sankt Augustin, 1993.

[7] N. Gustafsson, *The HIRLAM Model*, in proceedings of Seminar on Numerical Methods in Atmospheric Models, ECMWF, Reading, UK, September 1991.

[8] 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.

[9] J.P. Singh, J.L. Hennessy, and A. Gupta, *Scaling Parallel Programs for Multiprocessors: Methodology and Examples*, IEEE Computer, Vol. 26, No. 7, July 1993, 42–50.

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

[11] B. Machenhauer, *The Spectral Method*, in Numerical Methods used in Atmospheric Models, Volume II. GARP Publication Series, No. 17, 1979, 124–277.

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

[13] H. Munthe-Kaas, *Super Parallel FFTs*, SIAM J. Sci. Stat. Comput. 14 (1993) 349–367.

[14] S.A. Orzag, *Transform method for calculation of vector-coupled sums. Application to the spectral form of the vorticity equation*, J. Atmos. Sci. 27 (1970) 890–895.

[15] D. Parkinson, *Super Parallel Algorithms*, in Supercomputing, NATO ASI series F, Vol. 62, Springer, 1989.

[16] L.F. Richardson, *Weather Prediction by Numerical Process*, Cambridge University Press, London, 1922.

[17] A.J. Simmons, *Some aspects of the design and performance of the global ECMWF spectral model*, in proceedings of the Workshop on Techniques for Horizontal Discretization in Numerical Weather Prediction Models, ECMWF, Reading, UK, November 1987, 249–304.

[18] 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.

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