Finite Difference and Spectral Models for Numerical Weather Forecasting on a Massively Parallel Computer.

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@lurch.smhi.se

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

Abstract

A computationally intensive part in a model for numerical weather forecasting solves a set of partial differential equations. There are several techniques to obtain this solution numerically. In this paper we will discuss the implementation of a numerical weather forecasting model on a massively parallel architecture using two techniques: a finite difference (gridpoint) method and a spectral method. A comparison between the two methods based on their actual performance will be presented. Besides the price/performance ratio of several compute platforms for this forecast model will be discussed.

1 Introduction

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 resolution of atmospheric models is an important factor 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 number of floating point operations is proportional to (some power of) the number of gridpoints; and 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.

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

A general distinction can be made between global and local models. Local models ('limited area models') have the advantage of a lower number of gridpoints at the same resolution as global models. The disadvantage, on the other hand, is that they require lateral boundary conditions. Assuming the boundary conditions are generated by a lower resolution global model, the time range over which the higher resolution results from the local model are valid is roughly limited to the time it takes the lower resolution boundary conditions to penetrate into the central part of the limited area.

A modern atmospheric model consists of two main parts. The first is called the 'dynamics'; its task is to solve the equations of motion discretised to the model gridpoints. 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. Some physical processes like radiation, not directly described by the basic model equations, are also parameterized.

If the model is to be used for weather forecasting, it is to start from initial conditions that represent a very recent state of the atmosphere. Therefore, weather forecasting systems always consist of an analysis scheme, which is a system to generate the initial conditions from observations, and of the actual forecast model. These two components are supported by sophisticated systems to collect observations and to distribute the results.

In concept, the actual 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 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 SIMD to MIMD).

As an initial step, we decided to start with attempts to implement a limited area model on a massively parallel SIMD machine. We chose a limited area model because there is a natural projection of the model domain onto a rectangular array of processors; the choice of a SIMD machine was motivated by the fact that most available codes are for vector, i.e. SIMD, machines. In this way we eliminated many questions, like which data model to choose, and which strategy to follow for converting an existing production code for parallellisation. Instead, we were able to start our investigations concentrating on the questions:

- What is the required effort to implement an existing application to achieve high efficiency?
- What is the cost-effectiveness of such implementation?
- What is the scalability of such implementation?

Based on the considerations above, the HIRLAM[1] forecast model was chosen for the application to be implemented, and the hardware platform we selected was a MasPar system.

2 The HIRLAM forecast model

The HIRLAM forecast model [4] is coded in standard Fortran 77, with the exception of some I/O routines, that are in C. The core of the model are the subroutines to do the dynamics and to do the physics. 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. The physics routines are coded as one-dimensional loops over all horizontal gridpoints. Because almost all physical processes are in one-dimensional vertical columns, without mutual communications, the model physics can be described as N disjunct processes, where N is the number of gridpoints in the horizontal.

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

The solution of the dynamics of the model, on the other hand, requires horizontal communications between the columns. The amount of communications depends on the solution method for the dynamics. 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 gridpoint method. Other existing integration schemes are the fully explicit methods, semi-Lagrangian methods, and spectral methods. The semi-implicit schemes are coded as relatively small corrections to the explicit methods. In the next paragraphs the several integration methods are compared for their amount of horizontal communications.

The explicit Eulerian gridpoint dynamics require nearest-neighbour communications in both horizontal directions. The integration is on a staggered grid [1], i.e., the wind variables are kept at points halfway in between the points where the other variables are kept. At some place the fields must be destaggered, (e.g., before entering the physics routines). Due to destaggering, there are some diagonal communications in the dynamics routines.

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 CP requirements. The calculation of the semi-implicit corrections requires the solution of a set of Helmholtz equations. On vector machines the solution of the Helmholtz equations costs a negligible extra of CP time, but this may be different on multi-processor machines, because it requires global communications.

Semi-Lagrangian methods allow another increase of the time steps, again with a factor in the order of five. They require more than just nearest-neighbour communications, but the communications are still local: they do not extend over more than five (or so) grid distances. Although the additional costs, over the Eulerian schemes, are substantial, semi-Lagrangian methods promise to be very cost-effective. At this moment, however, these methods are not yet in a state of development that they can be applied in routine forecasting.

The comparison between gridpoint and spectral methods is of a different kind: whereas semi-implicit and semi-Lagrangian methods were developed mainly for reasons of numerical stability, spectral methods offer rather more physical advantages. The spectral method itself is cheaper than the gridpoint formulation; as an example we mention that the solution of the Helmholtz equation for semi-implicit methods is almost free of costs in the spectral formulation. But because the computation of non-linear terms and the physics part of the model requires 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 this is even more relevant, because the transformations require global communications. In HIRLAM, the transformations are Fourier transforms, and cost-efficiency of the spectral model heavily depends on the availability of efficient library routines for fast Fourier transforms.

We limited our investigations to the fully explicit gridpoint Eulerian and the semi-implicit spectral Eulerian formulations. The former was chosen because of its minimum of global communications; the implementation model is the simplest possible: one or more gridpoints are mapped onto a processor. By opting for the explicit formulation, we accept a loss of a factor of five on efficiency with respect to the current production version of HIRLAM, which is semi-implicit. The spectral model, on the other hand, is more challenging because of all formulations it is the one with most global communications.

So both the gridpoint as 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 concern the horizontal discretization and solution technique, of which the advantages and disadvantages will be discussed in section 3.

Figure 1 shows an example of the horizontal integration area in the HIRLAM model. This example is based on a horizontal grid of 110×100 points, as it is used in a production model.

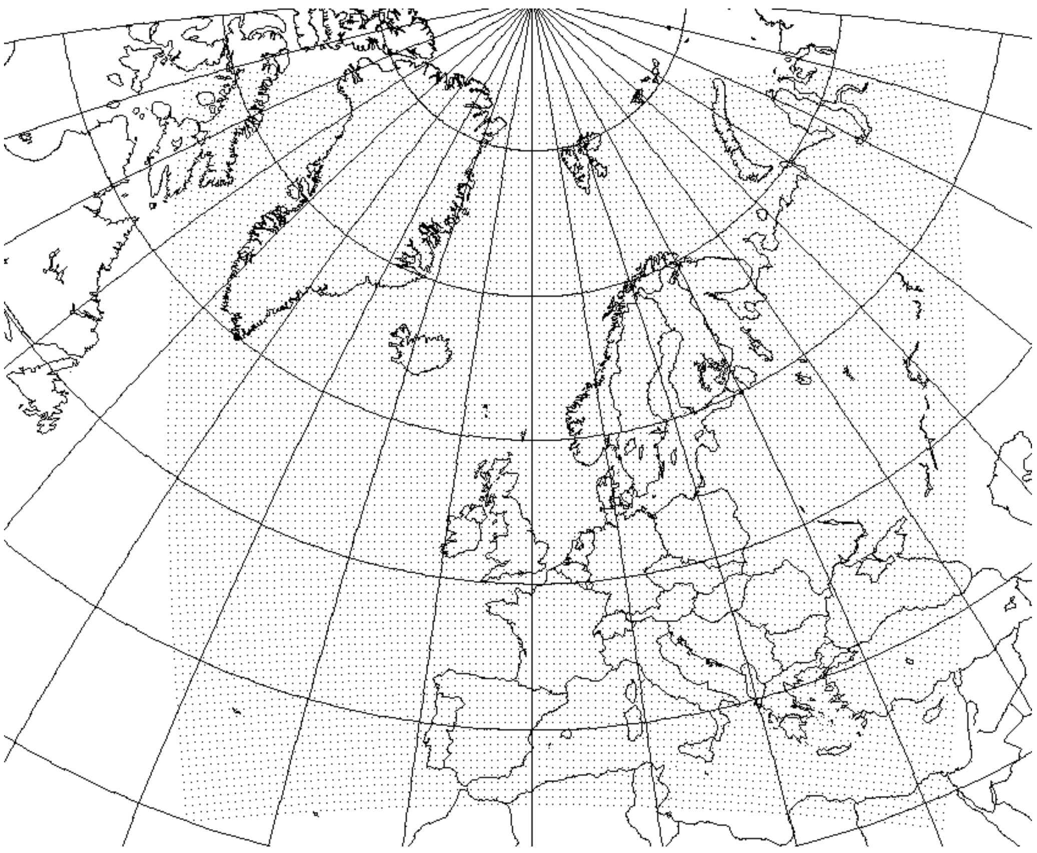

Figure 1: An example of a HIRLAM horizontal integration area with 110 × 100 gridpoints.

3 Gridpoint Model versus Spectral Model

Two numerical techniques, the finite difference or gridpoint technique and the spectral transform technique, are most commonly applied within the meteorological community. Also the finite element technique has reached some popularity in recent years.

Let us illustrate the two most popular techniques by the simple example of one-dimensional advection of temperature. In analytic form:

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

Introducing the most simple non-staggered horizontal grid, the discretisized gridpoint version with a leap-frog time-stepping will read:

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

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.:

$$T(x, t) = \frac{1}{\sqrt{2\pi}} \sum_k \hat{T}(k, t) \exp(ikx) \ , \tag{3}$$

$$\hat{T}(k, t) = \frac{1}{\sqrt{2\pi}} \sum_x T(x, t) \exp(-ikx) \ . \tag{4}$$

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 $d\hat{T}/dt$. Note that we have transformed the partial differential equation into an ordinary differential equation by using the orthogonal space functions $\exp(ikx)$.

From the simple example above, it is clear that the gridpoint technique requires communications in the neighbourhood of each gridpoint only, while the spectral transform technique requires communication over all the gridpoints through the spectral transforms.

For a global geometry the spectral transformations are straightforward, since this geometry allows for a natural periodic variation of all variables in both directions. Therefore the spectral transform technique [2, 8] has attained a great popularity for global numerical weather prediction. Advantages of the spectral transform technique to be mentioned are:

1. Complete control the spatial spectrum makes it possible to avoid a devastating influence of interaction between the physical processes and small scale noise generated by, e.g., the finite difference approximations in a gridpoint model.

2. A proper truncation of transformed non-linear terms will allow for an aliasing-free representation of quadratic terms.

3. High order accuracy of spatial derivatives is obtained by computation of these derivatives in spectral space followed by inverse transforms.

4. Possibility to use implicit time integration techniques, as well as horizontal diffusion schemes, since the resulting implicit equations are easily solved in spectral space.

Initial steps to apply the spectral transform technique to a limited area were taken by Machenhauer and Haugen [5], 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 [3] into the full multi-level HIRLAM framework.

There is yet no clear answer to the question whether it is preferable to apply the spectral or the gridpoint technique for a particular model configuration and for 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 4 grid distance waves. The shortest wave in a spectral model generally corresponds to 3 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.

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, since all dynamics and physics calculations can be done in the non-extended 'real' computational area. However, on a parallel distributed memory architecture it is necessary to map the extended area grid on the processor grid. Ideally, the number of gridpoints in the extended area should be about 25% larger than in the inner computational area (10% in each direction), which means that an equal percentage of processors has to be applied to perform the extra calculations in this 'artificial' area.

The performance of the spectral HIRLAM is crucially depending on the availability of fast FFT subroutines. The basic algorithms of the package for 'Super-Parallel' FFTs of Munthe-Kaas [7] were designed and developed for applications on SIMD computers. The idea of super parallel FFTs is a novel one, developed by Munthe-Kaas. 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 [9]. The only restriction in the FFT package of Munthe-Kaas is that the sizes of the problems should be powers of 2 (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.

Two more remarks could be made on the efficiency of spectral versus gridpoint models. First, in favour of the spectral model, all calculations in gridpoint space could be made strictly local since there is no horizontal staggering of the gridpoints. Also in spectral space, all calculations could be made strictly local, the coefficients of each wave-number are treated separately. As a second point, when we move to larger number of horizontal point, 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.

4 The Parallel Architecture

In this section we present some characteristics of the massively parallel MasPar systems used in this investigation. These systems are also sold by Digital under the name of DECmpp systems. For a detailed description of the systems see, e.g. [6].

A MasPar system 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. A PE is an 80 ns load/store arithmetic processor with a 16 Kbytes or 64 Kbytes data memory. On a MP-1 system the PE is a 4-bits processor, while the newer MP-2 systems contain 32-bits processors. The PE can operate on 1, 8, 16, 32, and 64 bit integers. The floating point precision is 32 or 64 bits. A full 16K MP-1 system has a peak performance of 26,000 MIPS and 550 Mflops (64-bits) or 1,200 Mflops (32-bits). For a full MP-2 system these numbers are 68,000 Mips, and 2,400 Mflops or 6,300 Mflops, respectively.

The PEs are controlled by the Array Control Unit (ACU). This is a register-based load/store processor with 128 Kbytes data memory and 1 Mbytes instruction memory. The ACU is responsible for the instruction decode and broadcast of instructions and data. It also includes a 12 MIPS scalar RISC processor for operations on scalar data. The PE-array and ACU form the Data Parallel Unit (DPU).

In parallel distributed memory computer systems communications between the processors form a critical component. They are often the bottleneck in achieving higher performance. For the MasPar system we will discuss two important types of communications.

The first type is the communication between the Processor Elements (PEs). It can be divided into two classes: Xnet and Router communication. Xnet communication performs nearest-neighbour communications. The Processor Element 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, so in horizontal, vertical and diagonal directions. This can be extended to communication between two PEs that lie on a straight line in each of the eight directions. The maximum communication bandwidth using Xnet is 23 Gbyte/s for a full 16K configuration. Router communication 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. Another limitation is that there is only one Router channel for 16 PEs.

The second important type of communication is formed by the communication channels between the FE and the DPU, and vice versa. Usually these channels are used to distribute input data over the DPU and return output data from the DPU with a theoretical peak transfer rate of 2 Mbyte/s. With additional hardware this can be improved to 10 Mbyte/s. These are only theoretical figures: typical achievable rates are around 1 Mbyte/s and 6 Mbyte/s, respectively.

This shows that these communications are extremely expensive and should be limited as much as possible. The programmer is responsible for distributing the data over the FE and the DPU, and much of a conversion effort should be aimed at keeping the data as much as possible on the 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. It determines the operating system and as a result one can use all software which is available for the front-end and operating system (ULTRIX). In addition we have the dedicated software to utilize the massively parallel system: the MasPar Fortran (MPF) compiler. This compiler is an implementation of Fortran 90. This indicates the programming model for the MasPar system: data-parallel programming. 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. To translate Fortran 77 programs to Fortran 90 MasPar provides the VAST-II compiler. Another very powerful software tool is the MasPar Programming Environment (MPPE). MPPE has been shown to be very useful for executing, debugging, profiling, and visualizing programs or program parts.

Some details concerning the hard- and software used in our investigation. The MasPar MP-1 system was a MasPar DPU Model MP-1104 (64 rows, 64 columns), using a DEC 5000/240 front-end running ULTRIX V4.2A (MP-3.1), and with the Mpfortran compiler version 2.1.46. The MasPar MP-2 system contained a MasPar DPU Model MP-2216 (128 rows, 128 columns).

5 Gridpoint Model Results

Before discussing the performance results for the gridpoint version, we should mention some implementation issues one encounters during porting the 28,000 lines of Fortran 77 HIRLAM code to a MasPar system. A detailed overview of all implementation issues can be found in [10].

The first issue concerns the distribution of the data. 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.

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, resp., 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.

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 some routines in HIRLAM had to be rewritten, since they depend on indirect addressing.

Due to several problems the 'output'-routine PUTDAT could not be ported successfully to the MasPar system yet. As a result no output-field could be generated, since that is the task of PUTDAT.

Finally, it turned out that the compiler generates redundant Xnet-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 will improve the performance on all kind of platforms.

We will now present several timings achieved by porting the HIRLAM code to different configurations of the MasPar architecture. We adopted as a test run the calculation of a 1-hour forecast on a $64 \times 64 \times 16$ grid at 55 km spacing. The gridpoint version with this resolution requires 60 time steps of 1 minute. The grid fits perfectly on a 4K (64×64) PE array. On a 1K (32×32) PE array the grid is splitted automatically in four layers of $32 \times 32 \times 16$ gridpoints. If the number of gridpoints is not a multiple of 32×32 some PEs will be idle during part of the calculation. Therefore the only sensible choice for the number of gridpoints in either horizontal direction is a multiple of 32. 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 the modeller would choose to keep the resolution fixed, he 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. So in practice, any implementation will be on a 32×32, or multiples thereof, grid, and for our investigations it is no limitation that we only study the $64 \times 64 \times 16$ grid, which exactly matches the 1K or 4K PE arrays.

In table 1 the total elapsed times are presented, that are needed to complete the test run on different MasPar configurations. Furthermore, the separate times spent in the input phase and in the calculation of the 1-hour forecast are shown. The timings for the configurations denoted with 'opt.' are obtained by specifying the -Omax compiler option. It prevents the inclusion of extra code for debug purposes, and performs the highest degree of optimization possible on the MasPar system.

Table 2 gives 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 these two tables several observations can be made. We want to mention the following points:

- From table 1 is it clear that I/O is an important factor. The input phase exists of two calls to the routine GETDAT. The first time this routine reads the initial values and the second time the boundary values. Since in a production run we will also have an output phase, and every six forecast hours an input phase of new boundary values, the time to perform the I/O operations will contribute significantly to the total elapsed time. This will be discussed in full detail in section 7.

- Comparison of the corresponding timings for the 1K and 4K configurations show that the calculations are nearly perfectly scalable with respect to the number of processors, if one excludes the I/O. For 'physics' the speedup factor is slightly higher than the theoretical factor four, while for the 'dynamics' a factor somewhat smaller than four is found.

- The codes produced by the optimizing compiler (the 'opt.'-versions) result in a dramatic decrease of 40% in the CP time spent in the input phase, see table 1. The gain in elapsed time for the actual forecast is only 10–20%.

- The explicit gridpoint version runs less than a factor two faster on the MasPar MP-2 than on the MasPar MP-1 (excluding again the input phase). A reason is the design decision

Table 1: Total elapsed times (in sec) to complete the test run with the gridpoint HIRLAM model on different MasPar configurations. Also the separate times for the input phase and the calculation of a 1-hour forecast are shown. See text for the meaning of the opt.-versions.

Model and # processors	Total time	Input phase	Forecast
MasPar MP-1 1K	258	25	233
MasPar MP-1 4K	84	25	59
MasPar MP-1 4K (opt.)	67	15	52
MasPar MP-2 1K	158	25	133
MasPar MP-2 4K	59	25	34
MasPar MP-2 4K (opt.)	44	15	29

Table 2: Elapsed times (in millisec) for one time step with the break down into the time spent in the 'dynamics' and in the 'physics'. See text for the meaning of the opt.-versions.

Model and # processors	Total	Dynamics	Physics
MasPar MP-1 1K	3790	1530	2260
MasPar MP-1 4K	937	408	529
MasPar MP-1 4K (opt.)	809	338	471
MasPar MP-2 1K	2142	1040	1106
MasPar MP-2 4K	550	287	263
MasPar MP-2 4K (opt.)	456	217	239

to enhance the processor power in the MP-2 only, and not to improve the communication bandwidth with respect to the MP-1. This can also be seen in table 2, where the ratio between the time for 'dynamics' and the time for 'physics' differs significantly on both systems. Remember, the dynamics contains many nearest-neighbour communications, while for the physics communication is much less important.

6 Spectral Model Results

The following strategy was adopted for implementation of the HIRLAM spectral model on the MasPar:

1. Available software packages for two-dimensional Fast Fourier Transforms on the MasPar are based on an organization of the input and output data according to a two-dimensional cut-and-stack mapping of the two-dimensional data arrays on the two-dimensional processor grid. Therefore, this two-dimensional organization and mapping of the data was used in the dynamical part of the model.
2. 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 very 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.
3. For the physics, exactly the same code as in the gridpoint model was used.

With this strategy for implementation of the spectral HIRLAM model on the MasPar, all interprocessor communication is carried out within the FFT routines, while the dynamics, physics and spectral space calculations are strictly local.

It was possible to introduce the computer code changes corresponding to this implementation strategy and also to convert the dynamical part of the code by simple editing commands. To summarize the experiences from the implementation of the HIRLAM spectral model on the MasPar system, it should first be mentioned that the implementation did not cause any major problems. The model had already been optimized for vector processors and the MasPar processor grid may be looked upon as a huge vector processor. As mentioned above, the major change was related to some of the data structures that needed to be changed to obtain an optimal mapping of the data on the processor grid. In other words, the data parallel programming style had to be introduced throughout the code. As in the gridpoint version to run efficiently on the MasPar architecture, actual array dimensions and loop bounds had to be introduced in some critical subroutines. Some coefficient matrices used for vertical transforms had to be forced to the PE memories by mapping directives in order to minimize the sloshes of scalar values between the FE and the DPU. All compilations were done with the highest degree of optimization (-Omax), which corresponds with

Table 3: Elapsed computing time (in sec) on different MasPar MP-2 configurations for a 6-hour forecast with the spectral HIRLAM model (16 vertical levels), and with a different number of horizontal points. See text for the meaning of the opt.-versions.

Model and # processors	# horizontal points	Forecast
Maspar MP-2 1K (opt.)	50×50	216
Maspar MP-2 4K (opt.)	50×50	50
Maspar MP-2 4K (opt.)	110×100	241
Maspar MP-2 16K (opt.)	110×100	55
Maspar MP-2 16K (opt.)	221×221	287

the 'opt.'-versions in the gridpoint model, and this will be also denoted as such in the spectral model.

Two operational data sets from the application of the HIRLAM system at the Swedish Meteorological and Hydrological Institute were used for benchmarks on the MasPar systems. For most of the tests, data from a horizontal area consisting of 110×110 gridpoints (128×128 in the extended area, see section 3) and with 16 vertical levels were utilized. The horizontal grid distance in this data set is approximately 55 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.

For all the runs with a transform grid resolution of 55 km, it was possible to use a time step of 5 minutes. So 72 time steps were carried to obtain forecasts valid at +6 hour. In order to test the MasPar also on a larger data set, the $110 \times 100 \times 16$ data set was interpolated horizontally to a data-set with $221 \times 221 \times 16$ ($256 \times 256 \times 16$ 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 3. Note that only the elapsed computing time for the pure forecast model integration is presented for each run, since the timing for the input phase is the same as for the gridpoint version taken into account the different horizontal areas of course.

The elapsed computing 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 given in table 4.

The following of more general interest could be noted about the results in tables 3 and 4:

- The speedup factor to run the same forecast on four times as many processors seems to be slightly greater than four. This means that the spectral formulation also leads to scalable algorithms with respect to 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.

Table 4: Elapsed times (in millisec) for one time step with the break down into the time spent in the 'dynamics' and in the 'physics'. See text for the meaning of the opt.-versions.

Model and	Number of	Forecast time		
# processors	horizontal points	Total	Dynamics	Physics
Maspar MP-2 1K (opt.)	50×50	3006	1754	1252
Maspar MP-2 4K (opt.)	50×50	699	457	242
Maspar MP-2 4K (opt.)	110×100	3347	2051	1296
Maspar MP-2 16K (opt.)	110×100	765	512	253
Maspar MP-2 16K (opt.)	221×221	3986	2611	1375

– Finally, from other benchmarks with HIRLAM spectral and gridpoint models on the Convex C3840 and Cray Y-MP, we can calculate that for the spectral HIRLAM model without I/O phases the processing speed on the MP-2 with 16K processors is about 1000 MFlops. For the HIRLAM model this number has never been achieved on any other compute platform. Yet it is only 1/6 of the theoretical maximum of the MP-2.

7 Performance Comparison

In this section we compare the performance of the gridpoint and spectral versions of the HIRLAM model.

Let us first concentrate on the timings presented in the previous sections to perform one time step in the integration of the forecast model. In table 2 it is shown that such a time step with the 'optimized' gridpoint version takes 456 ms for a $64 \times 64 \times 16$ grid on a MasPar MP-2 with 4K processors. Of this time 217 ms is spent in the 'dynamics' and the remaining 239 ms in the 'physics'. For the spectral model with $50 \times 50 \times 16$ points, see table 4, the total time is 699 ms, divided in 457 ms for the 'dynamics' and 242 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 3). Since the 'physics'-part of both versions is the same, we find nearly equal times for this part, while for the 'dynamics' a difference of a factor two is obtained. This is a result of the different types of communications. However, as explained in section 2 the spectral method allows a larger time step than in the gridpoint model. How this turns out, will be discussed in the next paragraphs.

For a real HIRLAM production forecast one has to consider the following facts. In an operational implementation for a 55 km resolution, the 'dynamics' will be calculated with time steps of 1 minute in the fully explicit version because of numerical stability. For the 'physics' a larger time step can be chosen, namely 15 minutes. Furthermore, there will be an input phase for new boundary values every 6 hours and an output phase every hour. Finally, to get some information about changes in pressure, wind-speed, etc., and to save on the total amount of output-information, every 5 minutes some statistics will be calculated.

Taken into account these facts we can calculate the total averaged costs for an operational 1-hour forecast on a MasPar MP-2 system with 4K processors. The time for one time step in the 'dynamics' and in the 'physics' can be found in table 2. Since the output phase (the routine PUTDAT) is the inverse of the input phase (the routine GETDAT) it can be assumed that they each will take the same amount of time. In table 1 one sees that an input phase, existing of two calls to GETDAT, takes 15 s (optimized version). Therefore the averaged time spent in the I/O phases in our 1-hour forecast will be: $(1/6 + 1) \times 15/2 = 8.7$ s. The time to compute the statistics has been measured to be 0.06 ms on the MP-2 with 4K processors.

As a result the total time to produce a 1-hour forecast with the fully explicit method is 23.6 s, as is shown in table 5. Of this time 56% is spent in the 'dynamics'. With a semi-implicit scheme this contribution would be reduced by a factor five theoretically, since the time step can be increased by this factor, see section 2. On the other hand it implies the solution of a set of Helmholtz equations, which introduces an overhead. A reasonable estimate for this overhead is 100% of the dynamics time, so we should double the dynamics time for each time step. Since the other components will be the same as in the explicit version, the total time for 1-hour forecast will become 15.7 s for

Table 5: Times to calculate an operational 1-hour forecast on a MasPar MP-2 with 4K processors. See text for details.

Method	Dynamics	Physics	Statistics	I/O	Total
Fully explicit	$60 \times 0.22 = 13.2$	$4 \times 0.24 = 1.0$	$12 \times 0.06 = 0.7$	8.7	23.6
Semi-implicit	$12 \times 0.44 = 5.3$	$4 \times 0.24 = 1.0$	$12 \times 0.06 = 0.7$	8.7	15.7
Spectral	$12 \times 0.46 = 5.5$	$4 \times 0.24 = 1.0$	$12 \times 0.06 = 0.7$	8.7	15.9

the semi-implicit version, see table 5. We are currently implementing the semi-implicit scheme to investigate the actual performance.

The effort to implement the semi-implicit method would in the end lead to an overall reduction in CP time of a factor 1.5. This rather moderate result stems of course from the considerable I/O overhead. We intend to investigate if this overhead can be reduced by the use of disk arrays and other I/O improvements provided by MasPar. This of course will also effect the total time for the explicit version.

Using table 4 one can make the same calculations for the spectral version. It takes 0.46 s for the dynamics and 0.24 s for the physics on a MASPAR MP-2 with 4K processors to complete one time step. Again the differences in the number of horizontal gridpoints is compensated by the higher intrinsic accuracy of the spectral method. Since the I/O and the statistics will take the same amount of time as in the gridpoint version, the resulting time for the spectral version on the MasPar MP-2 is 15.9 s, see table 5.

If we subtract the I/O overhead in the comparison we obtain the following times to complete a 1-hour forecast: for the fully explicit gridpoint formulation 14.9 s, as an estimate for the semi-implicit gridpoint formulation 7.0 s, and for the spectral formulation 7.2 s. From these numbers we can first conclude that despite the fact that although the spectral formulation heavily depends on global communications and the explicit gridpoint formulation needs only nearest-neighbour communications, the spectral version is a factor two faster to produce the 1-hour forecast. This is due to the fact that the spectral method allows a five times larger time step. However, if one includes the semi-implicit scheme to the explicit method, also resulting in global communications, it is estimated that the semi-implicit gridpoint version and the spectral version have on the balance the same efficiency like on sequential or vector platforms. One reason for this is the highly optimized FFT package that has been used for spectral model. As explained most of the inter-processor communication in the spectral model occur within this package. So if one wants to reduce the communication overhead, one should concentrate on this package, while in the gridpoint model inter-processor communication is spread throughout large parts of the code.

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.

8 Cost/Performance for the HIRLAM Model

From our findings we now can derive rough estimates of the cost-efficiency of the MasPar system compared to other hardware platforms. Before doing so, however, we stress the fact that to achieve the timings above, much human investment was needed. The human costs are not reflected in the estimates below.

In table 6 we present the total CP time to complete a 1-hour forecast on different computing platforms. The MasPar MP-1 timings are determined in the same way as in the previous section for the MP-2 numbers. For the spectral version we assume that the MP-2 system is twice as efficient as a MP-1, when it executes parallel code. This is based on the experiences with the gridpoint version and on performance measurements with the FFT package.

In this table all calculations are performed in 32 bits precision, unless stated otherwise. Concerning the comparison of 32 bit and 64 bit calculations, we have the following remarks: 1) Up to now we have not seen that 32 bits is insufficient for the HIRLAM model, perhaps with the exception of initialization. But this may be different at high resolutions. 2) If 32 bits is sufficient then the 32 bits MasPar should be compared to a 32 bits Convex or a 32 bits Cray. The last is not available, so the Cray is too expensive for its purpose. 3) On a Convex 210 64 bits arithmetic is as expensive as 32 bits.

As a result the numbers in table 6 can be compared. The Cray and Convex are simply too accurate for the investigated versions of HIRLAM, which is paid by a relatively bad cost/benefit ratio. This is a hardware feature, just like the MasPar has to pay for its hardware concept by extra communications.

Concerning this cost/performance comparison it is clear that a parallel system like the MasPar

Table 6: Price/performance comparison for different systems. Performance is measured as the total elapsed time to complete a 1-hour forecast.

System	Method	List price ($)	1-hour forecast (sec)
MasPar MP-1 4K	explicit	$\approx$ 195K	32
	semi-implicit		20[a]
	spectral		23[b]
MasPar MP-2 4K	explicit	$\approx$ 540K	24
	semi-implicit		16[a]
	spectral		16
Dec Alpha	semi-implicit	$\approx$ 60K	42
Convex C210 (64-bits)	semi-implicit	$\approx$ 100K[c]	96
Cray Y-MP (64-bits)[d]	semi-implicit	$\approx$ 2.5M[e]	6

[a]estimated
[b]assumption: parallel performance of a MP-2 is twice the performance of a MP-1
[c]not listed anymore
[d]1 processor
[e]per processor

is competitive with vector-architectures for this application. However, the system is defeated by a state-of-the-art workstation like a DEC-alpha. Only if an improvement of the I/O overhead (e.g., by parallel disk-arrays) leads to a substantial reduction, the parallel MasPar MP-1 system can become price-competitive to a DEC-alpha workstation, and then only for semi-implicit and spectral versions.

Finally, it should be mentioned that for operational weather forecasting the performance is more important than the costs: producing a forecast for a few hours ago is not useful, even if it is relatively cheap.

9 Conclusions

To conclude a summary of the main results of this investigation:

- A spectral HIRLAM model is preferable to an explicit gridpoint version, despite the global communications needed versus the nearest-neighbour communications.
- It has been estimated (an actual implementation is currently undertaken) that a semi-implicit gridpoint version will result in an equal performance as the spectral model.
- I/O is an important issue in parallel systems. For the MasPar architecture it contributes for 25-55% to the total elapsed time for producing a HIRLAM forecast (MasPar provides several options to improve I/O, which is also currently under investigation).
- Concerning cost/performance a massively parallel system, like MasPar, can compete with vector-architectures. However, improvements in sequential processing should not be lost track of.
- The algorithms for numerical weather forecasting used in this application give evidence of a good scalability.
- The effort to port an application like HIRLAM to a parallel architecture is quite considerable. However, this was also true for vector platforms when they entered the market.

Acknowledgments

Finally, we would thank Hans Munthe-Kaas (Bergen University, Norway) for the use of his library with Super-Parallel FFTs, the Para//ab at the Institute of Informatics (Bergen University) for giving access to their DECmpp 12000/Sx Model 200 (MasPar MP-2216), and Nigel Jagger (MasPar Computer Corporation, Reading, UK) for his support and the access to their MasPar MP-1104.

References

[1] A. Arakawa and V.R. Lamb, 1976: *Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model*, Report, Dept. of Meteorology, University of California, Los Angeles, 1976.

[2] 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.

[3] N. Gustafsson, *The HIRLAM model*, in Proceedings of *Seminar on Numerical Methods in Atmospheric Models*, ECMWF, Reading, UK, 9–13 September 1991.

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

[5] B. Machenhauer and J.E. Haugen: *Test of a spectral limited area shallow water model with time-dependent lateral boundary conditions and combined normal mode/semi-Lagrangian time integration schemes*, in Proceedings of *Workshop on Techniques for Horizontal Discretization in Numerical Weather Prediction Models*, ECMWF, 2–4 November 1987, pp 361-377.

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

[7] H. Munthe-Kaas: *Super Parallel FFTs*, to appear in SIAM J. on Scientific and Stat. Comput.

[8] 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.

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

[10] L. Wolters and G. Cats, *A Parallel Implementation Of the HIRLAM Model*, to appear in Proceedings of *The Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology, ECMWF, 23–27 November, 1992*, ECMWF, Reading.