Limited Area Numerical Weather Forecasting on a Massively Parallel Computer

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

A data-parallel implementation on a SIMD platform of an operational numerical weather forecast model is presented. The performances of two popular numerical techniques within these models are discussed, namely finite difference (gridpoint) methods and spectral methods. In this paper in particular the performance achieved for a full production run is investigated. Also the price/performance ratios of several compute platforms for this forecast model are shown.

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.

Published in proceedings of the 8th ACM International Conference on Supercomputing, July 1994, Manchester, England, pp. 289–296.

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 discretized 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, the HIRLAM[1] forecast model was chosen for the application to be implemented, and the hardware

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

platform we selected was a MasPar system.

2 The HIRLAM Forecast Model

The HIRLAM forecast model [7] is coded in standard Fortran 77. 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.

The solution of the dynamics of the model, on the other hand, requires horizontal communications between the different 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. The semi-Lagrangian method has still too many numerical problems, and therefore we will not investigate this further. In the next paragraphs the other integration methods are compared for their amount of horizontal communications.

The explicit Eulerian gridpoint dynamics require nearest-neighbour communications in the 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.

The comparison between gridpoint and spectral methods is of a different kind: whereas semi-implicit 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 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 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.

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.

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. Finite element techniques have reached some popularity in recent years too.

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 [3, 10] has attained a great popularity for global numerical weather prediction.

Initial steps to apply the spectral transform technique to a limited area were taken by Haugen and Machenhauer [6], 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 [5] 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 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. However, using the data-parallel programming model 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 [9] 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 [11]. 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 wavenumber 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. [8].

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).

The MasPar systems have 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 to 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.

It should be mentioned that the main difference between the MP-1 and the MP-2 configurations is the increase in peak performance by a factor five. However, the communication network is exactly the same for both model types.

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. 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. It 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.

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) 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 the Vast-II (version 3.06), and the Mpfortran compiler (version 2.2.7). In all cases the –nodebug and –Omax compiler-options were specified, which prevents the inclusion of extra code for debug purposes, and performs the highest degree of optimization possible on a MasPar system.

5 Forecast Timings

In this section the performance of the different numerical methods for the forecast routines will be discussed and compared. Issues like pre/post-processing and I/O will be investigated in section 6.

5.1 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 [12].

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.

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 test runs the calculation of a 6-hour forecast on a $64 \times 64 \times 16$ grid and a $128 \times 128 \times 16$ 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 PE array with less processors than the number of horizontal gridpoints, the grid is splitted 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. 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 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.

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 are scalable with respect both to the number of processors and to the number of gridpoints.

- The gridpoint version runs roughly a factor two faster on the MasPar MP-2 than on the MasPar MP-1, which is clearly less than the theoretical factor five, see section 4. A reason is the design decision 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 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-neighbour 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 Mflops 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 Mflops

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 # processors	Grid size	Forecast (in s)	1 Time step (in ms)		
			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

(53%-86%). For the $128 \times 128 \times 16$ grid size on a MP-2 with 16K processors, we found for the most time consuming 'dynamics' routines a speed of 1200 up to 1750 Mflops (19%–28%), and for the 'physics' routines 1900 up to 3300 Mflops (30%–52%).

5.2 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. 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.

3. For the physics the same code as in the gridpoint model was used.

With this strategy for implementation of the spectral HIRLAM model on the MasPar, all inter-processor 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. 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.

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 MP-2 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 2. Again the elapsed execution 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:

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

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

– The spectral HIRLAM model without I/O ran on the MP-2 with 16K processors with a speed of 600 Mflops. This is comparable to the performance of that model on a multiprocessor CRAY C90 (450 Mflops).

5.3 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.

In table 1 it is shown that a time step within the semi-implicit version takes 500 ms for a $64 \times 64 \times 16$ grid on a MasPar MP-2 with 4K processors. Of this time 291 ms is spent in the 'dynamics' and the remaining 209 ms in the 'physics'. For the spectral model with $50 \times 50 \times 16$ points, see table 2, the total time is 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 3).

For a real HIRLAM production forecast one has to consider the following facts. In an operational implementation for a 55 km resolution, the 'dynamics' can be calculated with time steps of 5 minutes because of numerical stability. For the 'physics' a larger time step can be chosen, namely 15 minutes. Furthermore, to obtain some information about changes in pressure, wind-speed, etc., every 5 minutes some statistics will be calculated.

Taken into account these facts we can calculate the total averaged costs for the 6-hour forecast on a MasPar MP-2 system with 4K processors. The time to compute 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-neighbour 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'. So 360 time steps for a 6-hour forecast. For the physics and statistics nothing changes 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-neighbour communications, the first two methods are favourable. This is mainly due to the fact that these methods allow a five times larger time step. Comparing execution time of the semi-implicit method with the one of the spectral version shows that the semi-implicit is the fastest way to calculate the 6-hour forecast. However, the difference is not very large contrary to the fact that the number of global communications is considerably larger in the spectral formulation. One reason for this is the highly optimized FFT package that has been used for 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, 4]).

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.

6 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.

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

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.

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/post-processing phases of the production run. As a result of our implementation the data has to be copied to and from the DPU (back-end) to the front-end. Also these copies are assumed to be part of the pre/post-processing phases.

From table 4 one can draw the following conclusions:

- The pre/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 the data is copied automatically several times if one does not use all available processors of the MasPar system. This is embedded in the MasPar runtime system to provide real nearest-neighbour communications even if not all PEs are used.
- 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, it is possible to improve the observed 'overhead':

1. Replace the front-end by a faster machine with better I/O capacities. This will affect the fore-end calculations.
2. Execute the pre/post-processing on the DPU. A large part of these computations can be executed in parallel.
3. Dump the calculated values of the desired variables directly from the back-end on disk in a raw form. MasPar systems provide several hard- and software improvements to read or write data directly by the DPU from or to disk in a very fast way. Subsequently, a second process could perform all the pre- and post-processing concurrently with the forecast calculations.

None of these options have been implemented yet. It is clear from the achieved timings that the pre/post-processing, in particular getting data in or out of the parallel system, is very important.

7 Cost/Performance for the HIRLAM Forecast 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 5 we present the total elapsed time to complete a 6-hour forecast on a grid of $128 \times 128 \times 16$ point using different computing platforms. 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. 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 5 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 is competitive with vector-architectures for this application. However, the system is defeated by a state-of-the-art workstation like a DEC-alpha. But if an improvement of the pre/post-processing phases leads to a substantial reduction of elapsed time, the parallel MasPar systems can also become price-competitive in comparison to a DEC-alpha workstation.

Table 5: Price/performance comparison for different systems. Performance is measured as the total elapsed time (in sec) to complete a 6-hour production run for a 128 × 128 × 16 grid with the semi-implicit version of the HIRLAM forecast model.

System	List price ($)	6-hour production run
MasPar MP-1 4K	≈ 200K	557[a]
MasPar MP-2 4K	≈ 550K	554[b]
Dec Alpha	≈ 60K	1000
Convex C210 (64-bits)	≈ 100K[c]	2300
Cray Y-MP (64-bits)[d]	≈ 2.5M[e]	140

[a] forecast only: 222 s
[b] forecast only: 109 s
[c] not listed anymore
[d] 1 processor
[e] per processor

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

8 Conclusions

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

- The semi-implicit and spectral version of the HIRLAM model are preferable to a fully explicit gridpoint version, despite the global communications needed versus the nearest-neighbour communications.
- The semi-implicit gridpoint version results in a higher performance than the spectral version.
- Copying data in and out of a parallel system is very time-consuming. However, there are several hard- and software options to improve this issue.
- 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 can be executed very efficient and give evidence of a good scalability both to the number of data-points as to the number of processors.
- 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, *Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model*, Report, Dept. of Meteorology, University of California, Los Angeles, 1976.

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

[3] 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.

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

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

[6] 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.

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

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

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

[10] 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.

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

[12] 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.