PARALLEL IMPLEMENTATION OF THE NUMERICAL WEATHER PREDICTION ETA MODEL ON DISTRIBUTED MEMORY PROCESSORS

N.M. Missirlis  
Department of Informatics,  
University of Athens,  
Panepistimiopolis 157 10,  
Athens, Greece.  
Tel: +(301) 7217941  
Fax: +(301) 7231569  
Email: nmis@di.uoa.gr

L.A. Boukas  
Department of Informatics,  
University of Athens,  
Panepistimiopolis 157 10,  
Athens, Greece.  
Tel: +(301) 7217941  
Fax: +(301) 7231569  
Email: lbouk@di.uoa.gr

N.E. Argiropoulos  
Athens High Performance Computing Laboratory,  
University of Athens,  
Panepistimiopolis 157 10,  
Athens, Greece.  
Tel: +(301) 7217941  
Email: nargi@apollonios.di.uoa.gr

N.Th. Mimikou  
Brainware A.E.,  
105 Michalakopoulou,  
115 27 Athens,  
Greece.  
Tel: +(301) 7486590  
Email: nmimikou@di.uoa.gr

J.G. Papageorgiou  
Hellenic National Meteorological Service  
P.O. BOX 73502, GR–16603 Hellinikon,  
Athens, Greece.  
Tel: +(301) 9620704

Author for correspondence: N.M. Missirlis
PARALLEL IMPLEMENTATION OF THE NUMERICAL WEATHER PREDICTION ETA MODEL ON DISTRIBUTED MEMORY PROCESSORS

N.E. Argiropoulos\textsuperscript{a}, L.A. Boukas\textsuperscript{b}, N.Th. Mimikou\textsuperscript{c}, N.M. Missirlis\textsuperscript{b,}\textsuperscript{*} and J.G. Papageorgiou\textsuperscript{d}

\textsuperscript{a} Athens High Performance Computing Laboratory, University of Athens, Panepistimiopolis 157 10, Athens, Greece.

\textsuperscript{b} Department of Informatics, University of Athens, Panepistimiopolis 157 10, Athens, Greece. Email: nmis@di.uoa.gr

\textsuperscript{c} Brainware A.E., 105 Michalakopoulou, 115 27 Athens, Greece.

\textsuperscript{d} Hellenic National Meteorological Service P.O. BOX 73502, GR–16603 Hellinikon, Athens, Greece.

Abstract

In this paper we present a parallel implementation of the mesoscale numerical weather prediction Eta model on distributed memory MIMD processors. The Parallel Virtual Machine (PVM) message passing library was selected for implementing the required interprocessor communication. Domain decomposition techniques are used for the horizontal discretization, whereas in the vertical direction each column per grid point is computed. It is found that for a mesh network topology the computation domain of each processor must be a square of side $\sqrt{N/p}$ for a $\sqrt{N} \times \sqrt{N}$ mesh of grid points, where $p$ is the number of processors. Moreover, we show that the number of grid points per processor plays an important role in the communication complexity and when $t_s < \sqrt{N/p}$, where $t_s$ is the start-up time, is satisfied, then the communication to computation ratio is improved by an order of magnitude. All data communication for finite differencing and I/O is encapsulated within a parallel library. Finally, we present performance results on the Parsytec CC-8 machine.

\textsuperscript{*}Research is supported by the Greek General Secretary of Research and Technology (ELEET II No. 322)
1 Introduction

The Eta model has been developed at the US National Meteorological Center (NMC) based on a prior “minimum physics” version of the code written and tested at the Federal Hydrometeorological Institute and Belgrade University and at the Geophysical Fluid Dynamics Laboratory of Princeton University. It is the current-generation mesoscale Numerical Weather Prediction model running in production at NMC for forecasts longer than 1/2 hours (Henderson et. al. 1995). The numerical and physical parameterization schemes of the Eta model have been described in Janjic (1984, 1990) and Mesinger et. al. (1988). In short the Eta model uses

- the Eta vertical coordinate (Mesinger 1984) which permits step-like representation of mountains and quasi-horizontal coordinate surfaces.
- an Arakawa E grid and
- the Janjic (1984) horizontal advection scheme which imposes a strict control on false energy cascade (e.g. Janjic and Mesinger 1984).

Eta was originally developed and optimized for vector-based architectures such as the Cray C90. Recently, the code was transformed into the two dimensional version in order to be suitable for parallelization on MIMD machines. In this paper we present the parallel implementation of the Eta model for distributed memory processors when their network is a mesh.

The computation and communication analysis for a domain decomposition reveals that the computation domain of each processor must be a square in order to achieve linear speedup and constant efficiency. The Parallel Virtual Machine (PVM) message passing library was used for the required communication. Finally, we present our results on a Parsytec CC-8 machine.

2 The Eta model

2.1 The coordinate system

The Eta coordinate system \((x, y, \eta)\) is a generalization of the sigma \((x, y, \sigma)\) coordinate system (Phillips 1957). It has been proposed by Mesinger (1984) and has been used in the so called Eta regional model. The iso-\(\eta\) surfaces are quasi-horizontal while mountains have a step-like representation. The vertical axis is transformed using the formula,

\[
\eta = \eta_s \frac{p - p_T}{p_s - p_T}
\]  (1)

where \(p, p_s\) and \(p_T\) is pressure and the subscripts \(s\) and \(T\) stand for the ground and the top values of the model atmosphere, respectively. The parameter \(\eta_s\) is given as

\[
\eta_s = \frac{p_s f(z_s) - p_T}{p_s f(0) - p_T}
\]  (2)
where \( z \) is the value of the vertical Cartesian coordinate and the subscript \( rf \) stands for reference surfaces taken at the \( z_s \) height and at the mean sea surface, where \( p_{rf}(0) = 1013.25 \) mb. In (2) formula \( p_{rf}(z_s) = p_{rf}(0) \cdot \exp\left(-\frac{\Gamma z_s}{RT}\right) \) where \( \Gamma = 6.5 \), \( R \) is the specific gas constant and \( T \) is the temperature at \( z_s \). It can easily be derived that, for \( p_{rf}(z_s) = p_{rf}(0) \), the system gives the \( \sigma \)-system. A schematic representation of mountains is depicted in Fig.1.

![Figure 1: Representation of mountains.](image)

According to this representation, the velocity component which is normal to the vertical sides of the mountain, is set equal to zero. The discretization of the model variables is made on a semi-staggered \( E \)-grid mesh (Arakawa and Lamb, 1977) which conserves momentum and imposes a more stringent constraint on the false cascade of energy toward smaller scales. A special technique has been developed (Mesinger 1973; Janjic 1979; Vasiljevic 1982; Gullen 1983; Janjic and Mesinger 1984) to prevent the grid separation problem with the external and the lower internal modes of short waves. The horizontal coordinate system is rotated for 45° counterclockwise, resulting a new system \((x', y', \eta)\) which is used for horizontal advection calculations.

### 2.2 The governing equations.

The basic prognostic variables are: pressure difference between the top and the ground level of the model atmosphere \((PD = p_S - p_T)\), temperature \((T)\), specific humidity \((Q)\), horizontal components of wind \((u \text{ and } v)\) and turbulent kinetic energy \((Q2)\). The parameterization of surface uses the accumulated precipitation \((\text{PREC})\), the soil moisture \((\text{WET})\) and the surface potential temperature \((\text{QSL})\). The model is divided into two basic processes: the dynamical and the physical process. The first includes continuity equation modifications, horizontal advection, vertical advection, horizontal diffusion and divergence dumping, while the second includes turbulence, surface, moisture and radiation parameterization.

3
The basic set of equations is:

\[
\frac{\partial}{\partial t} \left( \frac{\partial p}{\partial \eta} V \right) + \nabla_\eta \left( \frac{\partial p}{\partial \eta} V \cdot V \right) + \frac{\partial}{\partial \eta} \left( \frac{\partial p}{\partial \eta} \dot{V} \right) + \frac{\partial p}{\partial \eta} \left( f k \times V + \nabla_\eta \Phi + \frac{R_d T}{p} \nabla_\eta p + F \right) = 0, \tag{3}
\]

\[
\frac{dT}{dt} - \frac{KT}{p} + T' + \frac{g}{c_p} \frac{\partial R}{\partial \eta} \frac{\partial p}{\partial \eta} = 0, \tag{4}
\]

\[
\frac{\partial \Phi}{\partial \eta} = -\frac{R_d T}{p} \frac{\partial p}{\partial \eta}, \tag{5}
\]

\[
\frac{1}{\eta_s} \frac{\partial p_s}{\partial t} + \nabla_\eta \cdot \left( \frac{\partial p}{\partial \eta} V \right) + \frac{\partial}{\partial \eta} \left( \frac{\partial p}{\partial \eta} \dot{V} \right) = 0, \tag{6}
\]

\[
\frac{\partial p_s}{\partial t} = -\int_{0}^{\eta} \nabla_\eta \cdot \left( \frac{\partial p}{\partial \eta} V \right) d\eta, \tag{7}
\]

\[
\frac{\dot{\eta}}{\partial \eta} = -\frac{\eta}{\eta_s} \frac{\partial p_s}{\partial t} - \int_{0}^{\eta} \nabla_\eta \cdot \left( \frac{\partial p}{\partial \eta} V \right) d\eta \quad \text{and} \quad \dot{\eta} \frac{\partial p}{\partial \eta} = \frac{1}{\eta_s} \frac{\partial p_s}{\partial t} - \int_{0}^{\eta} \nabla_\eta \cdot \left( \frac{\partial p}{\partial \eta} V \right) d\eta \tag{8}
\]

\[
\frac{dQ}{dt} + Q' = S, \tag{9}
\]

where

- \( V(u,v) \): horizontal velocity
- \( \dot{\eta} \): vertical velocity
- \( p \): pressure
- \( \Phi \): geopotential
- \( T \): temperature
- \( f \): Coriolis parameter
- \( R_d \): gas constant for dry air
- \( F \): frictional force
- \( K \): specific heat at constant pressure
- \( \omega \): earth's gravity acceleration
- \( R \): net vertical radiative flux
- \( Q \): specific humidity
- \( S \): sources and sinks of water vapour
- \( T', Q' \): turbulent effects on temperature and specific humidity respectively.

The symbols \( \frac{d}{dt} \) and \( \nabla_\eta \) stand for total time derivative and horizontal gradient, respectively.

### 2.3 Numerical aspects

The set of model equations is solved using explicit time differencing (Mesinger 1974, 1977; Gadd 1978) separating the adjustment processes from advection. The computations of adjustment processes are executed using a forward–backward scheme in time differencing (Mesinger 1973, 1974, 1977, Mesinger and Arakawa 1976, Janjic 1979). Horizontal and vertical advection use 2\( \Delta t \) time step while physical parameterization is applied every 4\( \Delta t \).

Horizontal advection process is treated by using a semi–Langrangian scheme near the boundaries and Janjic's (1984) scheme together with modified Euler–backward scheme in the main terrain. The position of the prognostic variables on the semi–staggered \( E \)-grid mesh is presented in Fig.2. Vertical advection uses the Euler–backward time scheme. Centered differences are used to calculate horizontal nonlinear diffusion.

The time step \( \Delta t \) must be chosen so that the Courant–Friedrichs–Levy (CFL) criterion
Figure 2: A part of the grid mesh used in Eta model. The horizontal components of wind are calculated on $V$–points while the other variables are calculated on $h$–points. On dot–points and star–points differences on $Ox'$ and $Oy'$ axis respectively are calculated.

to be satisfied. Horizontal grid separation can be chosen as 1, 0.5, 0.25 degrees. The model atmosphere can be divided into 16, 24 or 32 levels.

2.4 Physical parameterization

The Planetary Boundary Layer (PBL) is represented by the Mellor–Yamada Level 2.5 closure scheme (Mellor and Yamada 1974, 1982). This scheme uses a prognostic differential equation for turbulent kinetic energy (TKE) to parameterize turbulence in the PBL above surface layer. The set of equations which govern the level 2.5 scheme are:

\[
\frac{d}{dt} \left( \frac{q^2}{2} \right) - \frac{\partial}{\partial z} \left[ \ell q S, \frac{\partial}{\partial z} \left( \frac{q^2}{2} \right) \right] = P_s + P_b - \varepsilon \quad (10)
\]

\[
P_s = K_M \frac{\partial u}{\partial z} + K_M \frac{\partial v}{\partial z} \quad (11)
\]

\[
P_b = \beta g K_H \frac{\partial \Theta}{\partial z} \quad (12)
\]

\[
\varepsilon = \frac{q^3}{B_1 \ell} \quad (13)
\]

\[
K_M = \ell q S_M \quad (14)
\]

\[
K_H = \ell q S_H \quad (15)
\]
\[ \ell = \frac{\ell_0 k z}{k z + \ell_0} \quad \text{(Blackadar’s formula)} \quad (16) \]

\[ \ell_0 = a \int_{P_T}^{P_S} |z| dq dp / \int_{P_T}^{P_S} dq dp \quad (17) \]

where \( q \), \( B_1 \), \( a \), \( S_M \) and \( S_H \) are constants which are determined from experimental data (Mellor and Yamada 1982) and \( k \) von Karman’s constant. The meaning of the symbols used is 
\[ \frac{q^2}{2} : \text{turbulent kinetic energy}, \quad \ell : \text{mixing length}, \quad P_s : \text{production of TKE due to shear}, \quad P_b : \text{production of TKE due to buoyancy}, \quad \varepsilon : \text{dissipation term}, \quad K_M, K_H : \text{eddy exchange coefficients for momentum and head respectively and } \Theta_V : \text{virtual potential temperature}. \]

Numerical solution of the above set of equations (10)-(17) is obtained in a forward–backward manner. The obtained values of TKE are used to calculate new values of eddy exchange coefficients. The surface layer is parameterized using the Mellor–Yamada level 2 scheme which consists only of diagnostic equations (Mellor and Yamada 1974, 1982). The viscous sublayer over the sea is treated using Janjic (1994) approach. The deep and shallow convection is parameterized according to Betts and Miller’s scheme (Betts 1986, Betts and Miller 1986 modified by Janjic (1994)). Large scale condensation is treated in a simple and conventional manner.

3 Parallelization of the Eta model

In this section, we describe how to implement the Eta model on a distributed memory parallel computer. The most common techniques to introduce parallelism in atmospheric models is domain decomposition. The basic idea is to decompose the original domain into subdomains and assign each subdomain to a different processor. To keep the computations consistent with the sequential code inter–processor communication is usually needed.

The Eta model is organized into two major packages: dynamics and physics. The first solves the basic model equations while the second describes the effect of the physical processes (parameterizations for convection, turbulence and radiation). For the numerical computation of the dynamics, explicit schemes are used as they are ideally suited for parallel computing since they do not require the numerical solution of linear systems. The main advantage of the explicit schemes is that they require only local communication in contrast to the implicit schemes where global communication is needed. The disadvantage is that they must satisfy time–step constraints like the Courant condition. Regarding the physics, the computation for each grid point requires information from an entire column (radiative process) and to avoid communication it is important to keep such data local. Moreover, shortwave radiation can create load imbalances when a diurnal cycle is simulated as it depends on the location of the sun and on the cloud distribution. Thus, all grid columns can be considered to be independent of each other, allowing for a simple data decomposition. The model also contains a semi–Langrangian advection scheme [21]. However, the technique requires access to data from nearby grid columns, since it is necessary to compute the trajectory of a parcel of air. Therefore, a degree of local communication is introduced.
3.1 Domain Decomposition

A spatial domain can be decomposed in a number of different ways. However, for simplicity of algorithm implementation, the domain is usually divided into subdomains with simple and regular geometries. Next, by considering a mesh network topology for our processors, each subdomain is allocated to a processor. This mapping is crucial because it affects the communication, the degree of parallelism and the load balance among the processors. The ideal situation is the minimization of the ratio communication to computation time and a uniform work load across the processors throughout the execution of the algorithm. Taking into consideration the fact that the computations are implicit in the vertical direction, only the horizontal domain is decomposed. Let us consider the decomposition shown in figure 3 (a) for a $\sqrt{N} \times \sqrt{N}$ domain of grid points.

![Figure 3: Domain decomposition and mapping to processors $T_i$, $1 \leq i \leq p$.](image)

As is shown the mesh is partitioned into $p/q$ horizontal strips and each strip is again partitioned into $q$ rectangles of equal size. Each of these rectangles (see figure 1 (b) has $N/p$ points. In every grid point the stencil shown in figure 4 for the horizontal discretization is applied.

Because of the structure of the computational stencil, processors have to communicate their boundary points to the four neighbouring processors after each scan of the domain (figure 5). Therefore, each processor contains all the grid points corresponding to its domain, as well as those grid points which form its artificial boundary (in our case two rows/columns in each side (figure 5).

The communication pattern is shown in figure 5 and is carried out concurrently in all processors. To avoid communication between diagonal processors which contain only a common corner, the communication is carried out in two phases. In the first phase the column boundaries are
exchanged with the left and right neighbour and in the second phase the row boundaries are exchanged with the upper and lower neighbour. Note, that the aforementioned phases may be performed in any order. The computational complexity, for each scan of the domain is analogous to the number of grid points in the rectangle (area), namely

\[ t_{\text{comp}}(T_i) = O \left( \frac{N}{p} \right), \]  

(18)

which is a decreasing function of the number of processors \( p \). On the other hand, the communication time is the time required for each processor to communicate with its four neighbours. This time involves the exchange of data of two rows and two columns with the neighbour processors (figure 5). The communication complexity, therefore, is analogous to the number of grid points which lie on the two rows/columns perimeter of the domain allocated to each processor (figure 3 (b)), hence

\[ t_{\text{comm}}(T_i) = 2 \left( 2t_s + \frac{\sqrt{N}}{q} + \frac{q\sqrt{N}}{p} \right), \]  

(19)

where \( t_s \) is the communication start-up time. From (18) and (19) it follows that we can determine the optimum value of \( q \) such that the ration communication to computation is minimized. We have

\[ r = \min_{1 \leq q \leq p} \frac{t_{\text{comm}}(T_i)}{t_{\text{comp}}(T_i)} = \min_{1 \leq q \leq p} \frac{2 \left( 2t_s + \frac{\sqrt{N}}{q} + \frac{q\sqrt{N}}{p} \right)}{O \left( \frac{N}{p} \right)} = \min_{1 \leq q \leq p} \frac{2[2t_s pq + (p + q^2)\sqrt{N}]}{O(qN)} \]  

(20)

Letting \( \frac{\partial r}{\partial q} = 0 \) we find \( \hat{q} = \sqrt{p} \) which is the optimum value for \( q \). Note that the determination of the optimum value for \( q \) does not depend on the number of rows/columns which are interchanged. Therefore, for the decomposition of figure 1 (a), it is best to assign to each processor a square
block of grid side equal to $\sqrt{\frac{N}{p}}$. For the optimum value of $q$ the corresponding minimum value of the ratio $r$ is given by

$$r = c_1 \frac{t_s p}{N} + c_2 \sqrt{\frac{p}{N}}$$

where $c_1$ and $c_2$ are machine-dependent constants. From (21) we remark the ratio $r$ is composed of two terms and is governed by the first term as long as the start-up time $t_s$ is larger than or equal to the side of the square assigned to each processor, namely

$$t_s \geq \sqrt{\frac{N}{p}}.$$  

(22)

In fact, this term depends linearly on $p$ which means that for large number of processors, if the side of points per processor is less than the start-up time $t_s$ ((22) is satisfied), then $r$ will behave as a linear function of $p$. On the other hand, if (22) does not hold, then the second term in (21) dominates in which case $r$ is a very slowly increasing function of $p$. In this case the ratio $r$ is improved by an order of magnitude. In conclusion it is best to choose $N$ and $p$ such that

$$t_s < \sqrt{\frac{N}{p}}.$$  

(23)

From the above analysis it is easily derived that the parallel run time needed for the computation

Figure 5: Communication pattern. (‘O’ Artificial Boundary (halo points). ‘X’ Computation points.)
of all the points in the domain is

\[ T_p = t_c \frac{N}{p} + 4t_c + 4t_w \sqrt{\frac{N}{p}} \]  

(24)

where \( t_c, t_w \) are machine-dependent constants and are the constant amount of time—computation per grid point and the per—word transfer time, respectively. Under the assumption (23), the expressions for speedup and efficiency are as follows:

\[ S_p = \frac{T_1}{T_p} = \frac{t_c N}{t_c \frac{N}{p} + 4(1 + t_w)\sqrt{\frac{N}{p}}} = \frac{t_c p}{t_c + 4(1 + t_w)\sqrt{\frac{p}{N}}} \]

and

\[ E_p = \frac{S_p}{p} = \frac{t_c}{t_c + 4(1 + t_w)\sqrt{\frac{p}{N}}} \]  

(25)

From (25) we see that, for maintaining constant efficiency, \( n \) must be proportional to \( p \). However, the constant of proportionality depends on the hardware constants \( t_c \) and \( t_w \).

4 Parallel Implementation of the Eta code

In this section we present the programs and subroutines that were used to parallelise and test the meteorological code Eta.

4.1 Data partitioning

The first step of a parallel program is to allocate the data to the available processors. For the parallel meteorological code Eta this demand is satisfied by using as a preprocessing part, a program, called split1, that handles the data mapping to the processors. In particular, program split1 reads the input files of the sequential model, and distributes their contents to smaller processor-dependent files that in a latter step will be used as input to the parallel code. Consequently, a partitioning of data via program split1 occurs each time is needed to change the set of input data or the number of the participating processors.

4.2 Processor Identification

During the execution of the parallel code each processor is assigned a set of tasks that differ from the ones of the other processors. Thus, it is needed to distinguish one processor from another. This is achieved by numbering the processors from 0 to \( np - 1 \), and relating them to the coordinates \( (\text{my}_x, \text{my}_y) \) of the position that they hold on the two-dimensional mesh topology.
4.3 Topology

Subroutine \texttt{make	extunderscore topo} is called once at the beginning of the program and creates the two-dimensional mesh topology through which the message passing procedures will take place. Each processor will communicate with his top, down, left and right adjacent processor, named ‘upper’, ‘down’, ‘left’ and ‘right’ respectively.

4.4 Message Passing

Subroutine \texttt{Exchange} handles all steps involved in the procedure of message passing. Each message is formed by internal elements of the sending processor, then it is packed in a vector array, and is finally transmitted to the receiving processor, which will unpack the message and use it to update its halo points. According to the processor that the message will be sent to, we have the following four variants of the \texttt{Send} subroutine:

\begin{align*}
\text{Send	extunderscore up}, & \quad \text{Send	extunderscore down}, \quad \text{Send	extunderscore left}, \quad \text{Send	extunderscore right} \\
\end{align*}

Similarly, according to the processor from which the message has been sent, we have the following variants of the \texttt{Receive} subroutine:

\begin{align*}
\text{Recv	extunderscore down}, & \quad \text{Recv	extunderscore up}, \quad \text{Recv	extunderscore right}, \quad \text{Recv	extunderscore left} \\
\end{align*}

Considering the fact that the proposed communication scheme consists of four \texttt{Send} calls, that can be performed in pairs concurrently, and four blocking \texttt{Receive} calls, the body of subroutine \texttt{Exchange} is either:

\begin{align*}
\text{call Send	extunderscore up} & \quad (\text{message1}) \quad \text{call Send	extunderscore left} & \quad (\text{message1}) \\
\text{call Send	extunderscore down} & \quad (\text{message2}) \quad \text{call Send	extunderscore right} & \quad (\text{message2}) \\
\text{call Recv	extunderscore down} & \quad (\text{message1}) \quad \text{call Recv	extunderscore right} & \quad (\text{message1}) \\
\text{call Recv	extunderscore up} & \quad (\text{message2}) \quad \text{call Recv	extunderscore left} & \quad (\text{message2}) \\
\text{call Send	extunderscore left} & \quad (\text{message3}) \quad \text{call Send	extunderscore up} & \quad (\text{message3}) \\
\text{call Send	extunderscore right} & \quad (\text{message4}) \quad \text{call Send	extunderscore down} & \quad (\text{message4}) \\
\text{call Recv	extunderscore right} & \quad (\text{message3}) \quad \text{call Recv	extunderscore down} & \quad (\text{message3}) \\
\text{call Recv	extunderscore left} & \quad (\text{message4}) \quad \text{call Recv	extunderscore up} & \quad (\text{message4}) \\
\end{align*}

4.5 Parallel Output

During the execution of the sequential program, a subroutine named \texttt{Outvar} creates in predefined time periods, the output files of the meteorological model. Executing the same subroutine in parallel, results in acquiring a number of output files analogous to the number of processors that participated. These files contain subsets of what would normally be the output of the model. Program \texttt{merge1}, acting as a postprocessing part, joins the output data of the parallel program in files, such as to produce the appropriate output.
5 Numerical results

The aforementioned data partitioning and message passing techniques have been successfully tested and validated in both shared and distributed memory architectures. The programming model that was selected for the implementation is the Oak Ridge National Laboratory (ORNL) Parallel Virtual Machine (PVM), because it combines machine independance, portability and combatibility. The runs were made on the distributed memory Parsytec CC platform and they are highly encouraging, although they reflect only an early approach to the parallelization of the meteorological code Eta.

5.1 Target Machine

The parallel version of the Eta meteorological model was tested and validated on the parallel platform Parsytec CC–8.

Parallel machine CC–8 of Parsytec is an MIMD machine with 8 nodes. Two of the nodes are I/O nodes, whilst the remaining six are strictly computing nodes. Each node is a Power PC–133MHz with 128 MB RAM on each I/O and 64 MB on each computing node. The internal network is a high network with about 40 MB/sec node to node bandwidth.

5.2 Performance Analysis

The following Figures show the numerical results of the model as they were estimated for a 48h run on a \(121 \times 161 \times 32\) grid, with 4, 6 and 8 processors respectively. Times are measured in seconds in all cases.

![Figure 6: Total time.](image)
Figure 7: Speedup-Efficiency.

Figure 8: Computation-Communication.
From our results we can see that the communication time tends to remain constant while the number of processors increases as we already expected according to the theoretical analysis presented on the previous sections. In addition, the computational time decreases with a rate analogous to the number of processors. The combination of a constant communication time and a decreasing computational time results to a significant increase of the speedup and to an efficiency about 80%.

References


