Cluster of Supercomputers – an Alternative to Massively Parallel Computing –

Hermann Mierendorff, Anton Schüller and Ulrich Trottenberg
Institute for Algorithms and Scientific Computing (SCAI)
GMD – German National Research Center for Information Technology
Schloß Birlinghoven, D-53754 Sankt Augustin, Germany
Email: mierendorff@gmd.de, anton.schueller@gmd.de, trottenberg@gmd.de

Abstract. We discuss the question, whether the parallel IFS weather prediction model of the ECMWF can efficiently run on a *cluster of supercomputers.* In a joint project with Cray Research, we studied C90 systems which were connected by Hippi channels. The analysis shows that indeed a very high efficiency can be achieved provided the right parallelisation, cluster partitioning, and mapping approach is applied.

Keywords. cluster computing, cluster of supercomputers, parallel computing, performance prediction, Cray C90, Hippi channel.

1 Introduction

Cluster computing – the use of a strongly connected cluster of (shared memory) high performance computers instead of a single supercomputer – is a topic of central interest. However very little practical experience has been made so far with *large scale production codes* on clusters of supercomputers.

In a joint project with CRAY Research and in close co-operation with the ECMWF, we investigated the potential of cluster computing for weather (and climate) predictions. Clearly, weather prediction is one of the classical and one of the most important application fields where high performance computers are needed. We therefore started with this challenging application.

For our investigation, we consider the IFS (Integrated Forecasting System) code of the ECMWF. This code is of central significance for European medium range weather prediction and for the atmospheric calculations in climate research. SCAI has a very detailed knowledge of this code [4], since they provided the first parallel version of the code based on the portable interface PARMACS [2].

Meanwhile several parallel versions of the code are available: 1D and 2D *partitionings* and a large set of *mappings* [3]. An additional aspect is the resolution of the basic 3D-grid. Current fine resolution leads to a *calculation/communication ratio* which is favourable to efficient parallelisation. Good results have been achieved on a variety of parallel machines [4, 5]. Our study shows that it is necessary to thoroughly tailor the parallelisation strategy to the supercomputer cluster configuration in order to obtain optimal cluster computing efficiencies.

As a model in our cluster computing configuration, CRAY proposed to base the study on a cluster of C90 computers connected by one or several Hippi channels. *For the daily forecast model which contains about 6.3 million grid points we obtained a very good efficiency. Model evaluation delivered a speed-up of 1.7 on two C90 systems with 1 Hippi channel and an (even better) speed-up of 3.5 on four C90 systems with 6 Hippi channels if compared to a single C90 system.*

We introduce the *cluster efficiency* as a special metric to compare cluster of systems and single systems in section 2. The two subsequent sections contain a definition of the considered machine configuration and the structure of the parallel application. We briefly describe our main result in section 5. Section 6 gives an idea in which way our results had been obtained. Finally, we discuss several results of our investigation in section 7 in some greater detail.

2 Cluster Efficiency

The C90 computer itself is already a parallel computer (with shared memory). In order to measure the effects of clustering several C90 systems rather than measuring the overall parallel efficiency (counting all computing nodes involved), we define the following expression as *cluster speed-up*

$$S_{cl}(P) = \frac{T_{cl}(1)}{T_{cl}(P)}$$

and as *cluster efficiency*

$$E_{cl}(P) = \frac{S_{cl}(P)}{P} \; .$$

Here P denotes the number of supercomputers (C90 systems) used in the cluster and $T_{cl}(P)$ the computing time of the parallel code on a cluster consisting of P systems.

For the parallel IFS code, we consider the 2D partitioning consisting of $r \times s$ processes. We assume that the parallel code is *partitioned* to the cluster situation in an appropriate way. $T_{cl}(P)$ represents always the *shortest* runtime among all partitionings and mappings implemented with the IFS version RAPS 2.0.

3 Cluster computing assumptions

We consider an architecture consisting of Cray C90 systems that are connected by one or more Hippi channels. A C90 system contains up to 16 computing nodes with a shared memory. The Hippi channel provides a high raw data rate of 100 Mbyte/s at maximum. For realistic measurements, we considered the TCP/IP protocol and an appropriate C-environment. Using a single Hippi channel, 50 Mbyte/s for sustained data rate of very long messages and a start-up time of 0.46 ms have been measured on existing C90/Hippi configurations for inter-system transfer in the case of saturation (bidirectional ping-pong between 8 pairs of communicating processors [5, 7]). Since such systems were not implemented to show maximum performance on the Hippi, there might be some margin for improvement. Nevertheless, our consideration is based on the values measured. However, a single pair of communicating processes is not able to exhaust the Hippi transfer capacity. Only 3 or 4 pairs of communicating processes can achieve the above performance.

The intra-system communication within a C90 system is very fast. In this case, the PARMACS version 5 was used. We achieved a performance which was faster than that of the inter-system communication via the Hippi by more than one order of magnitude.

To simplify our case study, we considered fully connected systems only, i.e. each pair of C90 systems is connected by a separate Hippi channel. Two systems, for example, are connected by a single Hippi, 4 systems are connected by 6 Hippi channels.

Though the Hippi channel is at present the strongest possibility of connecting supercomputers, it is clearly the weakest part of the interconnection system for the C90 computing nodes. Therefore, it seems to be useless to investigate incomplete C90 systems. As usual, we use C916 to denote such systems consisting of 16 computing nodes. To minimise the worklaod of the Hippi, the structure of a cluster has to be considered when an application problem is partitioned and mapped onto the computing nodes. Subsets of application processes should be mapped onto the same system if they communicate very frequently .

4 The Parallel IFS – Partitioning Concepts

Each time step of the IFS consists of computations in three different spaces (grid point space, Fourier space and spectral space) and has the following order:

1. computations in grid point space,
2. transition to Fourier space by Fast Fourier Transforms,
3. computations in Fourier space,
4. transition to spectral space by Legendre transforms,

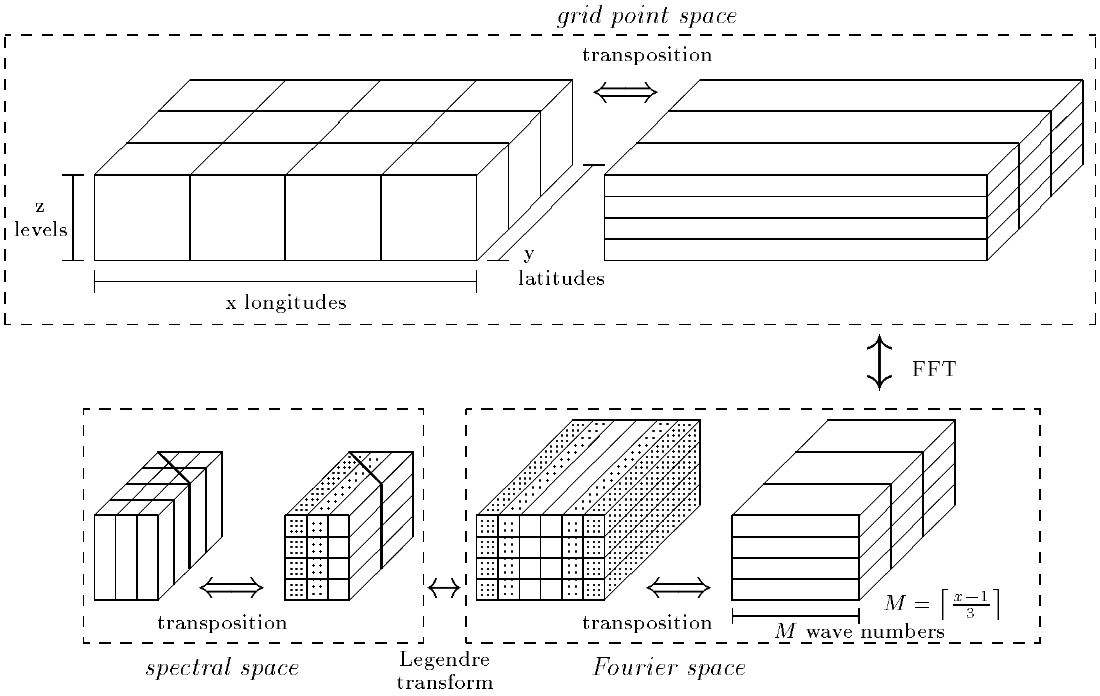
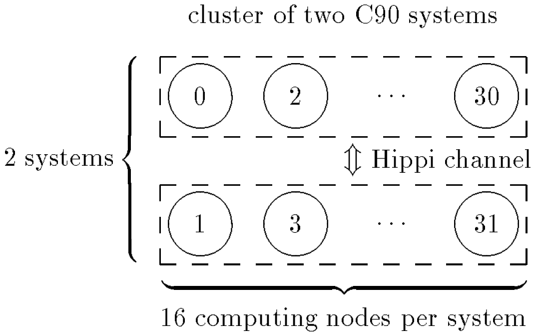


Fig. 1 Rough model for the 2D partitioning of the IFS data structure in grid point, Fourier, and spectral space into 3×4 parts.

5. computations in spectral space (phase I),
6. computations in spectral space (phase II),
7. transition to Fourier space by inverse Legendre transforms,
8. computations in Fourier space and
9. transition to grid point space by inverse Fourier transforms.

In each of these computational steps complex data dependencies exist in exactly one dimension while the other two dimensions are freely available for parallelisation and vectorisation. Unfortunately, the independent directions vary among the different computational steps. In grid point space, for example, these data dependencies are in the vertical direction without any influence of longitudes and latitudes, while in Fourier space, the data dependencies refer to the longitudinal dimension.

So, in each computational step a partitioning of the data and the computations to processes is possible with respect to two dimensions, providing a high



4×8 partitioning of one level of grid point space into 32 processes

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

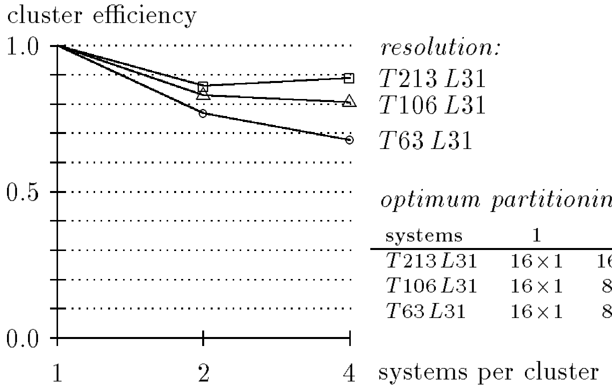
Fig. 2 Mapping of 32 processes onto processors for 4×8 partitioning of grid point space.

degree of parallelism. The data transposition strategy [1, 4] allows to utilise this approach efficiently on parallel systems. Its idea is to re-distribute the complete data to the processes (data transposition) at some stages of the algorithm such that the arithmetic computations between two consecutive transpositions can be performed without any interprocess communication: during the corresponding computational step the direction with data dependencies is not partitioned among processes.

Figure 1 shows the structure of the parallel algorithm: here, the data are partitioned among 3×4 processes. Transpositions are performed in each data space in order to allow a fully parallelised computational phase without any interprocess communication. The dashed cells in Fourier and spectral space illustrate the data partitioning before and after the Legendre transforms; in Fourier space each process is responsible for two of the indicated process columns (which are in general not adjacent) in order to provide load balanced Legendre transforms.

The usual expression for the size of the IFS data structure is TM Lz, where M is the *wave number* and z is the number of *levels*. We investigate TM L31 with $M = 63, 106, 213$. The standard relation between M and the number of grid points in direction of longitudes or latitudes is $M = \lceil \frac{x-1}{3} \rceil$ and $y = \frac{x}{2}$ (cf. Figure 1). For the present paper we consider the RAPS version 2.0 with full grid and Eulerian treatment of advection. The finest grid size corresponds to a 60 km network on the equator.

For the mapping of the process structure onto the computing nodes we used a simple mapping function called *column-mapping*. Here the columns of the pro-



optimum partitioning:

systems	1	2	4
$T213\,L31$	16×1	16×2	16×4
$T106\,L31$	16×1	8×4	16×4
$T63\,L31$	16×1	8×4	8×8

Fig. 3 Cluster efficiency.

cess structure are mapped onto the columns of the cluster structure one after the other. Depending on the size of these columns, more than one process column might be mapped onto one cluster column or one process column might be split to several cluster columns. Let $(i,k)_{0\le i\le r-1,0\le k\le s-1}$ and $(\iota,\kappa)_{0\le\iota\le\varrho-1,0\le\kappa\le\sigma-1}$ denote the elements of the $r\times s$-process structure or the $\varrho\times\sigma$-cluster structure respectively. The $\varrho\times\sigma$-cluster structure represents a cluster of ϱ C90 systems each of which consists of σ computing nodes. Then the column-mapping is defined by $kr+i=\kappa\varrho+\iota$. We refer to Figure 2 as an example of this mapping.

5 Main result

The estimated cluster efficiency is shown in Figure 3. Three resolutions were used for model evaluation: $T213\,L31$, $T106\,L31$, and $T63\,L31$. The largest size corresponds to the production code currently used in the ECMWF. The smallest case is just the lower limit where parallel runs on a C90 cluster might be useful. For all these cases we obtained a good speed-up. The two large examples show an efficiency between 80 and 90%. Therefore, a C90 cluster of P systems is able to execute the IFS for large problem sizes nearly P times faster than a single system.

Hence our main result is that the parallel IFS runs efficiently on C90/Hippi clusters for sufficiently large examples.

6 Performance Modelling

We model only the inner loop of the IFS code which takes the major part of the runtime for large examples. We consider a 2D partitioning of the 3D spaces where a structure of $r\times s$ processes of nearly identical size is produced. The current partitioning of the RAPS-2.0 code is roughly described in section 4 (cf. [3]).

We consider only those cases where the number of computing nodes of the cluster equals the number of processes ($r \cdot s = \varrho \cdot \sigma$). The inner loop consists of six computational steps separated by transpositions of the space. The transpositions are required to provide just that dimension of the space unsplit which is entirely requested for the current computational step. These transpositions mainly consist of data exchange where all processes of a row or column of our 2D process structure send a message to all other processes of the same row or column respectively. Since some of the exchange phases are very short when containing intra-system transfer only and because of data dependences, we can observe a series-parallel behaviour. As a consequence, we can model the total runtime of the inner loop approximately by a sum of runtimes of subsequent calculation or communication phases:

$$t = t_{calc} + t_{trans}$$

where t_{calc} represents the computing time and t_{trans} the transposition time. These functions are approximately determined by

$$t_{calc} = \max_{i,k}(t_{CPU}(i,k))$$

$$t_{trans} = \sum_{l=1}^{6} \max(t_{Hippi}(l), \max_{i,k}(t_{lat}(i,k,l)))$$

where $t_{CPU}(i,k)$ is the calculation time of process (i,k), $t_{Hippi}(l)$ is the maximum time cost needed by one of the Hippi channels for all related messages, and $t_{lat}(i,k,l)$ is the time cost of a processor for all messages of process (i,k) that are transported via the Hippi channel during transposition l ($l = 1, \cdots, 6$). The time cost for buffering data before or after the transfer and the intra-system communication has for simplicity been suppressed in our description. These time costs had been considered in the actual modelling additionally. But they turned out to be so small that they could not be shown in the graphical representation of results. For the model evaluation, we used a special tool [6].

To calculate t_{Hippi}, we used the Hippi transfer parameters (0.02 ms/kbyte and 0.46 ms/start) and the actual message numbers and sizes from corresponding IFS runs.

t_{lat} is calculated considering the messages of single processes using the parameters 0.07 ms/kbyte and 1.46 ms/start.

To estimate t_{calc}, we assume that processes of the parallel IFS show nearly the same computational work if they are identical in size and shape. The sequential IFS and the parallel message passing version of IFS were measured on a single C90 system using resolutions up to $T106\,L41$. We used the standard timing values produced by the considered code. These are mainly *total time,* *Legendre transform, inverse Legendre transform, Fourier space calculations together with FFT and inverse FFT, and spectral space calculations.*

We developed approximative complexity formulas for four different computational parts: *grid point space calculations, FFT, Legendre transform, and spectral space calculations.* The grid point space calculations represent the most important part of the computational work. For the majority of calculations it is in principle possible to integrate all grid points of a single level in one vector operation [3]. This way of vectorisation is considered a best case model. It is used to obtain our main result. For more details we refer to [7].

Based on 45 sets of measurements on 1 to 8 processors, we determined the following approximative expression for the computational work t_{CPU}:

$$
\begin{aligned}
t_{CPU} &= t_g + t_f + t_l + t_s \\
t_g &= 476 + (46.3 + 5.45z) \left\lceil \frac{\left\lceil \frac{x}{s} \right\rceil \left\lceil \frac{y}{r} \right\rceil}{128} \right\rceil + \left(0.0139z + 0.00202z^2\right) \left\lceil \frac{x}{s} \right\rceil \left\lceil \frac{y}{r} \right\rceil \\
t_f &= \left(0.0565 + 0.0631 \left(\left\lceil \frac{x}{128} \right\rceil + \left\lceil \frac{x'}{128} \right\rceil \right)\right) \left\lceil \frac{y}{r} \right\rceil \left\lceil \frac{z}{s} \right\rceil \\
&\qquad + \left(0.00262 + 0.000748 \left\lceil \frac{z}{s} \right\rceil\right) \left\lceil \frac{y}{r} \right\rceil (x \log x' + x' \log x) \\
t_l &= 6.57 \left\lceil \frac{x''}{r} \right\rceil + \left(0.00416 + 0.0628 \left(\left\lceil \frac{y}{128} \right\rceil + \left\lceil \frac{y'}{128} \right\rceil \right) + 0.000357yy'\right) \left\lceil \frac{x''}{r} \right\rceil \left\lceil \frac{z}{s} \right\rceil \\
t_s &= 0.161 + 0.082z + (0.0564 + 0.122z) \left\lceil \frac{x''}{r} \right\rceil + \left(0.00374z + 0.000185z^2\right) \left\lceil \frac{x''}{r} \right\rceil \left\lceil \frac{y'}{s} \right\rceil
\end{aligned}
$$

where t_g, t_f, t_l, and t_s represent calculations in the grid point space, during FFT, during Legendre transform, or in the spectral space respectively. In relation to the problem size TM Lz, the variables are defined by $M = \left\lceil \frac{x-1}{3} \right\rceil$, $y = \frac{x}{2}$, $x' = M + 1$, $x'' = \frac{x'}{2}$, and $y' = x' + 1$. All parameters are in *ms*. The start-up time of the vector unit is modelled by terms showing 128 as denominator. The deviation of the estimated values from measurements was in general less than 10%.

As an alternative worst case model, we also used vectorisation in direction of longitudes or latitudes only from one boundary to the opposite boundary. Since the C90 hardware does not allow to start vector operations for more than 128 elements at once, there is no big difference between our models in the case of large examples. For small examples, however, or in the case of very fine partitioning to many parallel processes, the worst case model delivered runtime values which were higher than those of the best case model by 10 to 20%. Since this effect is observed for 16 processors of a single system as well as for the 32 or 64 processors of a cluster, there was no big difference in terms of cluster efficiency. Therefore, we decided to discuss the best case model only throughout this paper.

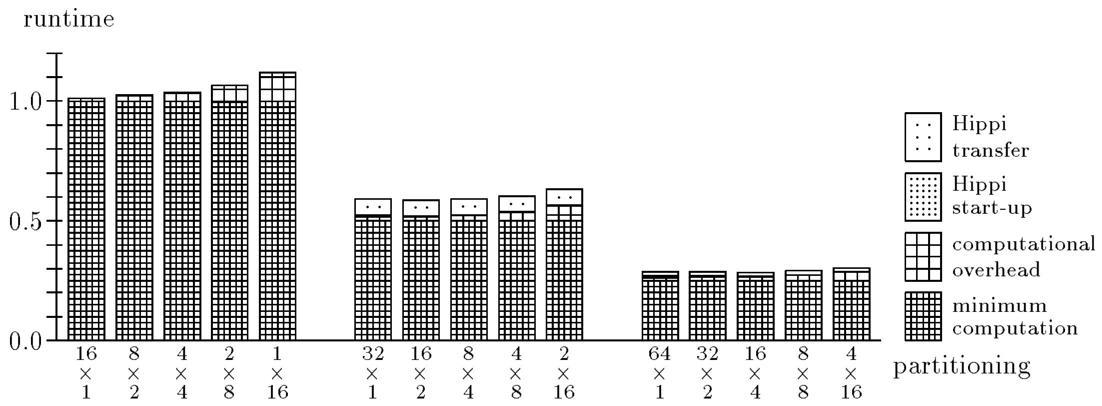


Fig. 4 Runtime for $T213\,L31$ on different system sizes and with different partitionings.

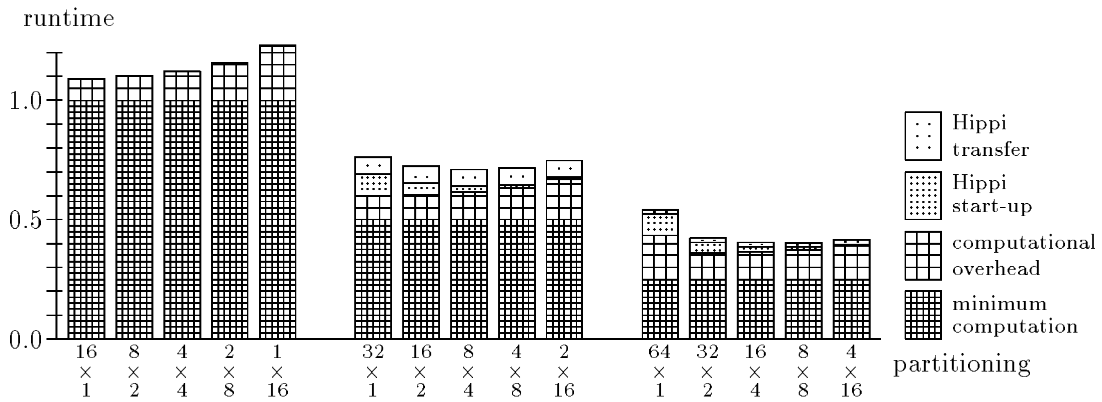


Fig. 5 Runtime for $T63\,L31$ on different system sizes and with different partitionings.

7 Discussion of results

The estimated runtime is represented in Figures 4 and 5. The 2D partitioning allows us to study a variety of $r \times s$ process structures. The part of the computational work is gridded and the Hippis part is dashed. Since we assume a series-parallel model, there is no overlap between these parts. The intra-system communication is of course existing and is considered. But this part takes less than 1% and cannot be seen in the figures.

The computational work is always split into a small gridded and a largely gridded part. Here the ideal *minimum runtime* on a single C90 system (i.e. *sequential_runtime/16*) is standardised to be 1. This part is represented by the small gridded bar. The influence of different partitionings as provided by the IFS is only marginal in the case of this large problem. In the case of 2 and 4 systems per cluster, we see nearly 1/2 or even 1/4 for the estimated runtime. Here the ideal runtime is of course *sequential_runtime/32* or *sequential_runtime/64* respectively. In all cases the overall runtime is dominated by the part called ideal runtime in our large example.

The remaining amount of computational work (largely gridded part) is caused mainly by more inefficient vectorisation and load imbalancing in the parallel case. We considered only those examples where load imbalancing is of minor importance. Load imbalancing would occur for an $r \times s$ process structure, if s is not a *good* divisor of the number of levels z. In this case, load imbalancing would occur at least within the Fourier space (cf. Figure 1 and [5]).

The Hippi time cost are split into a densely dashed part for start-up time and a sparsely dashed part for transmission time. The start-up time does not play any role for large problems. Therefore, the best partitioning is here $r \times 1$. For the $T63\,L31$-case, however, start-up time takes a considerable amount of time in particular on a 4-system cluster and with high values of r. Since the vectorisation is more efficient with high r-values, the best partitioning is here a squared partitioning.

Figure 3 shows better efficiency on a 4-system cluster than on a 2-system cluster for our largest example. To explain this effect, we remind that a 4-system cluster has 6 Hippi channels and a 2-system cluster has only one. This is of importance in cases showing high transmission time via Hippi.

We considered also other mappings (cf. [5]) but the column mapping used here showed the best results as long as $r \geq \varrho$.

8 Summary

Clusters of C90 systems that are connected via Hippi channels seem to be an interesting architecture for the parallel IFS. For clusters showing more than two systems, a network of Hippi channels is required. In the ideal case, each pair of systems should be connected by a separate channel. Nevertheless, an appropriate partitioning, mapping, and some other actions to minimise the data transfer via Hippi are required. The parallel program should be carefully vectorised. The best results are obtained by a partitioning that prefers to subdivide the latitudes. An efficiency in the range of 80 to 90% can be expected for clusters showing up to 64 processors and for large test cases.

Just the effort required for a good parallelisation shows that essentially slower interconnection systems like Ethernet do not promise efficient runs of the IFS on clustered supercomputers. Even if the capacity of the Ethernet is fully available for the cluster computing, we should expect an efficiency less than 30%, i.e. to run the IFS on ethernetted clusters is useless.

Acknowledgment. The RAPS version 2.0 of the IFS we used for our investigation was developed at the ECMWF. Essential parts of the work reported here and the development of the basic parallel algorithms have been carried out in close co-operation with the ECMWF (Prof. Geerd Hoffmann). The authors would like to thank Saulo de Barros, David Dent, and Lars Isaksen for their advice. The performance expected for C90 clusters has been investigated in co-operation with Cray Research. We would like to thank Michael O'Neill and Markus Kienemund who provided us with measurements on the IFS and on communication for the C90.

References

[1] S.R.M. Barros, T. Kauranne, *Spectral and multigrid spherical Helmholtz equation solvers on distributed memory parallel computers.* Proceedings of Fourth Workshop on Use of Parallel Computers in Meteorology, 26–30 Nov, 1990, ECMWF, Reading, UK, 1–27.

[2] R. Calkin, R. Hempel, H.-C. Hoppe, and P. Wypior, *Portable programming with the PARMACS message passing library.* Parallel Computing **20** (1994) 615–632.

[3] D. Dent, G. Robinson, S. Barros, and L. Isaksen, *The IFS model – overview and parallel strategies*, Proceedings of the 6th Workshop on the Use of Parallel Computers in Meteorology, 21–25 Nov, 1994, Reading ,UK (to appear).

[4] U. Gärtel, W. Joppich, A. Schüller, *Parallelizing the ECMWF's weather forecast program: the 2D case* Parallel Computing **19** (1993).

[5] U. Gärtel, W. Joppich, H. Mierendorff, and A. Schüller, *Performance of the IFS on Parallel Machines – Measurements and Predictions*, Proceedings of the 6th Workshop on the Use of Parallel Computers in Meteorology, 21–25 Nov, 1994, Reading ,UK (to appear).

[6] H. Mierendorff and R. Schwarzwald, *LAPAS: A Performance Evaluation Tool for Large Parallel Systems*, in P. Müller-Stoy (ed.), Architektur von Rechensystemen, VDE-Verlag, Berlin (1990) 245–253.

[7] H. Mierendorff, *The Weather Prediction Code of the ECMWF on clusters of C90 Systems - Performance Analysis and Efficiency Estimation*, Arbeitspapiere der GMD, Sankt Augustin (1995), to appear.