Performance Prediction for Parallel Local Weather Forecast Programs

W. Joppich and H. Mierendorff

GMD – German National Research Center for Information Technology
Institute for Algorithms and Scientific Computing (SCAI)
Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. Performance modeling for scientific production codes is of interest both for program tuning and for the selection of new machines. An empirical method is used for developing a model to predict the runtime of large parallel weather prediction programs. The different steps of the method are outlined giving examples from two different programs of the DWD (German Weather Service). The first one is the new local model (LM) of DWD. The second one is the old Deutschland Model (DM) which is characterized by more complicated algorithmic steps and parallelization strategies.

1 Introduction

Weather forecast belongs to a class of large applications for parallel computing since more than ten years. The life time of the codes is considerably longer than that of the fast evolving computer systems to be used for weather forecast.

Having in mind a large existing system or an hypothetical one, the key questions which have been posed by the DWD are: Will a one day LM forecast with about $800 \times 800 \times 50$ grid points, using a time step size of 10 seconds, be finished within half an hour. And for the DM: Will a one day forecast with about $811 \times 740 \times 40$ grid points, using a time step size of 7 seconds, be finished within half an hour, too. Additional questions concern the number of processors: for economical and practical reasons the number of processors should not exceed 1024. How should the components of the desired machine look like? Is it necessary to change the code dramatically in order to reach the required speed?

2 Basic Decisions

At first, proper test cases had to be defined. To avoid technical problems with parallel input and output, an artificial topography has been chosen. The initial weather situation is artificial, too. Local models require an update of boundary values from time to time. The new values are generated within the program itself, again in order to avoid either input and output from files or communication with a global model. To have a rather realistic view of the algorithm the decision was to model a 24 hour forecast. By this, especially the time-consuming radiation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 492–501, 2001.

part is included into the modeling pretty well. For each program test cases are selected in order to derive the model. The final model is applied to predict the run time for the test cases. Where possible, some test cases were not used for modeling but have been left for verification of the performance prediction model. Because the test cases have to run on existing machines and on machines being considerably smaller than the target machine the resolution is far from the finally desired one. The test cases which have been considered both for LM and DM are listed in Table 1.

Table 1. Collection of test cases for LM and DM

	resolution			time-step	processors
	x-dir.	y-dir.	z-dir.	ΔT [s]	
test cases LM	51	51	10, 15, 20	60	1x1 to 4x8
	153	153	10, 15, 20, 25	30	4x4 and 4x8
	325	325	35	30	4x8
target resolution LM	800	800	50	10	
test cases DM	41	37	20,30	15, 45, 90	4x4
	81	73	20	90	2x8, 4x4, 8x2
					16x4, 4x16, 8x8
	81	73	30, 40	90	4x4
	121	109	30	90	4x4, 8x8
	271	244	40	90	8x8
target resolution DM	811	741	40	7	

3 Analysis of the Parallel Programs

Partial differential equations are the mathematical background of weather prediction models. The equations are discretized in space and in time. The programs are written in FORTRAN 90 and the message passing interface MPI is used for communication on parallel machines. The basic concept of parallelization is grid partitioning. This concept is realized in a straight forward manner, including the usual overlap region for each partition both in the LM and in the DM. Nevertheless, the mapping of the different data fields to logical processes is slightly different within the two programs. In Figure 1 the 2D-partitioning strategy and the mapping to processes is shown both for the LM and for the DM. The number of inner points per partition and the logical process numbers are given. For the DM (right picture) the process identification number p_{id} is determined by the given pair of indices (p_{idx}, p_{idy}): $p_{id} = nproc_x(p_{idy} - 1) + p_{idx}$ where the number of participating processes $nproc$ is the product of $nproc_x$ and $nproc_y$. The interior overlap areas are not shown. But this overlap area in practice enlarges for instance a DM partition by two lines and columns in each direction. This type of partitioning is used everywhere within the LM and in most parts of the DM.

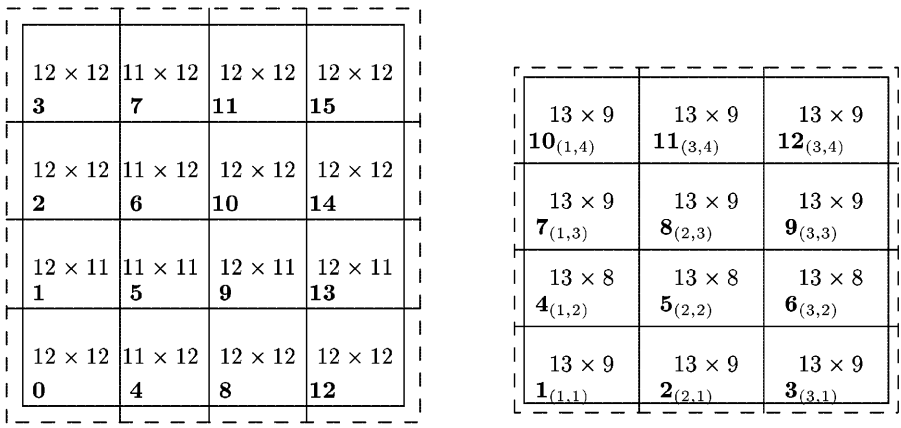

Fig. 1. LM grid (51×51, left) partitioned for 4×4 processes, DM grid (41×37, right) partitioned for $12 = 3 \times 4$ processes

The general approach to solve the discrete equations derived from the continuous model consists of an iterative sequence of several phases like dynamics computations, physics computations, FFT, Gauss-elimination, local communication, global communication, input/output of results, and transposition of data within the system of parallel processes. The LM equations are discretized explicitly in time and require no special solver for large systems of algebraic equations. The DM instead uses a semi-implicit method which leeds to a discrete equation of Helmholtz type. The corresponding discrete equation is solved by a well-established algorithm using Fast Fourier Transformation (FFT) in combination with Gauss elimination (GE). The data-partitioning for these cases is described in Figure 2. It is necessary to switch between the different partitioning strategies within the algorithm (transposition). Such a transposition is a very challenging task for the communication network of massively parallel computers.

The programs have been instrumented such that they provide with detailed timing information about every essential part of the program. From this information the computational complexity of the two main parts of each program (dynamics and physics) have been modeled. The model depends on critical parameters of the parallel application like size of the partition, relative position of a partition within the partitioning structure (boundary process, interior process), and time steps. The analysis of the parallel programs has led to a complete set of information about the communication pattern, the communication complexity, and the communication frequency of the programs. As an example, the result of the analysis is given for the subroutine TRANSPOSE of the program DM. This subroutine, which is called four times per time step, exchanges the data between the different processes when switching from 2D-partitioning to FFT row partitioning, to GE column partitioning, and back to 2D-partitioning. FFT and GE belong to the dynamics part of the DM. In Table 2 myi, myj, and myk

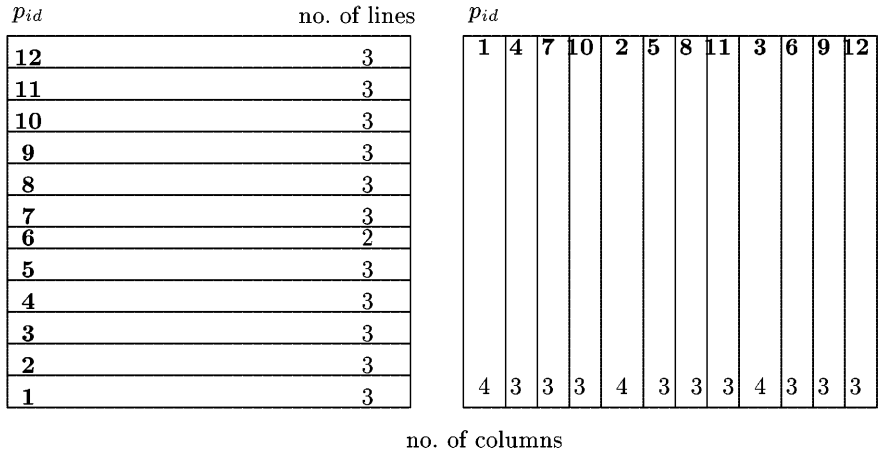

Fig. 2. DM grid (41×37) distributed to 3×4 processes for the steps FFT (left) and GE (right)

denote the local size of the 2D partition in x, y, and z direction, respectively. $rfft$ and cge are the rows and columns of a process for FFT and GE (the index pp denotes the same quantity for the partner process).

Table 2. Messages which are sent by TRANSPOSE in DM

no. of calls per day: $\frac{24 \cdot 3600}{\Delta T}$ action	total no. of processes	messages per process	message length
2D $\rightarrow$ 1D-FFT	$nproc_x \times nproc_y$	$nproc_x - 1$	$myk \cdot rfft_{pp} \cdot myi$
1D-FFT $\rightarrow$ 1D-GE	$nproc_x \times nproc_y$	$nproc - 1$	$myk \cdot rfft \cdot cge_{pp}$
1D-GE $\rightarrow$ 1D-FFT	$nproc_x \times nproc_y$	$nproc - 1$	$myk \cdot rfft_{pp} \cdot cge$
1D-FFT $\rightarrow$ 2D	$nproc_x \times nproc_y$	$nproc_x - 1$	$myk \cdot rfft \cdot myi_{pp}$

The different phases are synchronized because of data dependences and by local and global communication phases. The LM also contains MPI barrier calls for synchronization and the DM uses wait-all at the end of subroutines which realize the exchange of data with neighboring processes. Therefore, an additive model can be used to estimate the overall runtime by adding the runtime of all the single phases: dynamics, physics, FFT (only DM), GE (only DM), and communication which includes the time for buffering of data into a linearly ordered send buffer (and similar for the receive).

4 Modeling Computational Time

For estimating the runtime of the dynamics part of the LM it turned out that the numerical effort is mainly depending on the number of interior grid points. The upper left picture of Figure 3 shows this for the first hour of a low resolution experiment. There is some effort related to exterior boundary points which can be identified by analyzing several examples numerically. But this has almost no influence. The dynamics computing time depends linearly on the number of levels in vertical direction and linearly on the time step size as the lower pictures of Figure 3 show for the DM.

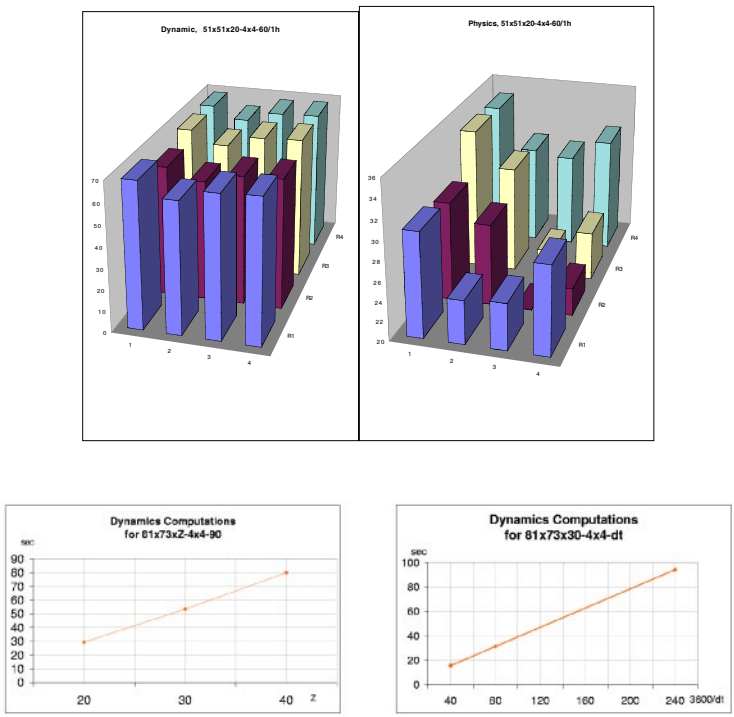

Fig. 3. Distribution of computational time for the first hour. LM dynamics (upper left) and LM physics (upper right) on 4×4 processors, low resolution. Linear dependence of the dynamics computation time for DM (levels, lower part left and time-step size, lower part right).

The computational time is approximated by a multi-linear function in the number of interior grid points (and derived quantities) and the number of time steps. The coefficients of the multi-linear function have been determined by a least squares method in order to match all considered measurements. Because it

is intended to extrapolate the performance of very large application cases running on very large computing systems on the basis of measurements of relatively small examples running on small parallel systems the leading term of the function has to be determined as exact as possible.

The runtime of physics computations depends on the number of grid points, the hour, and on the local weather situation. Especially the last mentioned effect causes non-balanced load, see the upper right picture in Figure 3. To avoid the dependence on the hour it is possible to accumulate all values of a day. Since the different numerical processes of the parallel codes are synchronized by data dependences and calls of some synchronizing MPI-constructs, the additive model has been justified and therefore the maximum runtime among all processes is of main interest. However, this maximum should not be caused by the number of grid points but by the local weather. The physics measurements are related according to the effective number of grid points before using them for estimating the runtime of physics computations. In principle, it is the ratio of physics and dynamics computation time which is approximated in order to get the computational time for the physics part of the programs [4].

5 The Machine Models

The machine models are based on measurements on existing machines. The benchmarking itself took much time and new questions arose while evaluating the results. The large amount of data required a careful analysis and not all of the data was reliable at all. The programs behaved differently on different architectures. One example is the influence of cache effects. This was observed both for LM and DM. Therefore, if necessary, the model includes for each part of the algorithm a cache factor which represents the effect of slow down by L2-cache misses. This factor depends on the local partition size, on the size of the cache and on the algorithm. It is desirable after all that the local partition fits into the cache. Figure 4 shows these slow-down factors both for the dynamics computation and the Gauss elimination in DM. This heuristic approach is strongly depending on the program under consideration and is reliable only in the range of the measured points. In addition to this cache effect the memory access is also included into the models. The time for copying data from 3D-data structures into linearly organized send buffers and back from receive buffers into 3D-arrays is not necessarily neglectable. The model for buffering is based on measurements with a program which uses a similar data structure for data access and storing as in the applications themselves. The analysis of the communication pattern combined with measurements was used to set up a formula to compute the time for local communication. Models for the barrier- and reduce-function, respectively, have been developed from measurements, too. They also reflect the underlying implementation of these MPI-constructs. The knowledge about the frequency and the required action of these events finally allows to estimate the time for these parts of the programs (see Tables 6 and 7).

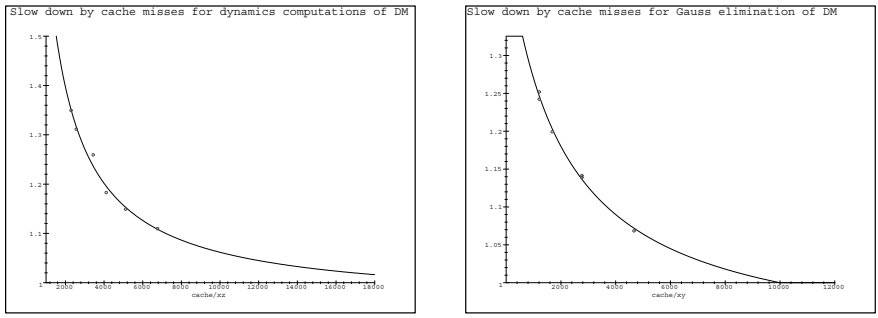

Fig. 4. Model for slow down of the DM by cache effects; dynamics left, GE right, Origin2000

6 Verification of the Models

The initial model for the LM has been developed on a SP-2 with 32 processors using 9 out of 13 parallel runs. Due to the needs of vector machines the models have been ported to an SX-4 using 3 calibration runs to adapt the model parameter. The vector start-up and the computational effort per grid point had to be included. This could be done because the main direction of vectorization was known. Further, the partitioning has to be adapted to the direction of vectorization in order to get long vectors. But none of the application programs is particularly tuned for vector processors, yet.

For proving the reliability of the models, the deviation of model evaluation and runtime values for a collection of test cases is shown. Some of the test cases are used for model development, others are exclusively used for testing the performance prediction. The values are given in Table 3 for the LM on an SP-2. Similar results are given for the LM running on an SX-4 in Table 4. All times concern 24 hour simulations. The different cases are characterized by the resolution in x, y, and z direction, the number of processors, $nproc_x \times nproc_y$, and the time step size in seconds ($x \times y \times z - r \times s - \Delta T$).

Table 3. Comparison of measured and estimated timing values for the LM on SP-2

case	runtime [h]		deviation [%]
	measured	estimated	
$51 \times 51 \times 20 - 1 \times 1 - 60$	8.75	8.85	1.17
$51 \times 51 \times 20 - 2 \times 2 - 60$	2.42	2.48	2.41
$153 \times 153 \times 20 - 4 \times 4 - 30$	11.20	11.30	0.85
$153 \times 153 \times 25 - 4 \times 8 - 30$	7.42	7.47	0.78
$325 \times 325 \times 35 - 4 \times 8 - 30$	43.50	45.87	5.44

Table 4. Comparison of measured and estimated timing values for the LM on SX-4

case	runtime [h]		deviation [%]
	measured	estimated	
$153 \times 153 \times 20 - 1 \times 1 - 30$	1.23	1.32	7.40
$153 \times 153 \times 20 - 1 \times 4 - 30$	0.35	0.34	-0.99
$255 \times 255 \times 25 - 1 \times 1 - 30$	3.56	3.77	5.86
$255 \times 255 \times 25 - 1 \times 4 - 30$	1.02	0.96	-6.07
$255 \times 255 \times 25 - 1 \times 12 - 30$	0.37	0.33	-11.24

Models of the DM have originally been developed for the VPP700 as well as for an Origin2000. The verification runs for the DM on a VPP700 show a maximum deviation from measurement in the range of less than ten percent. For a twelve hour forecast (the above tables show results for 24 hour simulations) on an Origin2000 Table 5 shows an acceptable agreement between estimated and measured time.

Table 5. Comparison of measured and estimated timing values for the DM on Origin2000

case	runtime [h]		deviation [%]
	estimated	measured [s]	
	(12 hrs)	(12 hrs)	
$41 \times 37 \times 20 - 1 \times 1 - 90$	216,54	222,62	-2,73
$41 \times 37 \times 30 - 4 \times 4 - 90$	33,73	32,93	2,44
$81 \times 73 \times 20 - 8 \times 8 - 90$	31,19	32,25	-3,29
$81 \times 73 \times 20 - 16 \times 4 - 90$	34,72	37,41	-7,18
$121 \times 109 \times 30 - 8 \times 8 - 90$	77,84	80,84	-3,72
$271 \times 244 \times 40 - 8 \times 8 - 90$	468,78	447,41	4,78

7 Application of the Models

As already mentioned, the desired resolution for the LM is 800×800×50 with $\Delta T = 10$ seconds. The extrapolation to this final resolution both in space and time and considering the defined set of parameters for the test case neglects the fact that not each of the systems having in mind allows an extremely large configuration. Therefore the 1024-processor system is assumed to be a cluster architecture, if necessary. The parameters for the cluster network communication can be specified.

Applying the model to a T3E-600 the prediction for the LM delivers an estimated runtime of more than 5 hours. The expected distribution of work is specified in Table 6 both for 1024 and 2048 processors.

Table 6. Runtime estimation for the final resolution on T3E-600

case	runtime	distribution of work [%]						eff.
	[h]	dynam.	physic	comm.	buffer.	barrier	reduce	
800×800×50−32×32−10	5.62	54.36	41.21	0.75	3.61	0.06	0.01	0.82
800×800×50−32×64−10	3.08	53.29	41.06	1.11	4.40	0.11	0.03	0.75

This shows that the processor speed compared to the T3E-600 has to be increased by a factor of about 11 in order to reach the requirements of the DWD for the LM. The results also show that the T3E interconnect network is powerful enough to work with processors having the required speed. We have applied the model to a currently available Origin2000 (195 MHz processors, Table 7). Because of the shared memory architecture we had to assume this machine to consist of several (4×4 and 8×8) Origin2000 systems with 64 shared memory processors each.

Table 7. Runtime estimation for the final resolution on an Origin2000 cluster architecture

case	runtime	distribution of work [%]						eff.
	[h]	dynam.	physic	comm.	buffer.	barrier	reduce	
800x800x50-32x32-10	5.53	76.34	17.73	4.19	1.48	0.25	0.01	0.83
800x800x50-64x64-10	1.78	71.03	16.76	8.67	2.55	0.94	0.05	0.64

8 Conclusion

Two performance prediction models (for LM and DM) have been developed both for vector machines and for clusters of shared memory architectures. Although the old DM is no longer used for operational purposes it may serve as benchmark because of its sophisticated algorithm and due to the changing partitioning strategy within different algorithmic steps. The original version of the DM contains the data transposition as described in [1] - [3]. This type of transposition turns out to be the bottle-neck on large systems because the number of messages then increases with the square of the processors used (a transposition strategy which linearly depends on the number of processors is assumed for the model). Further improvement should be made by parallel FFT and Gauss-elimination. Otherwise the degree of parallelism would be limited by the 1D-partitioning of these computational steps.

The initial version of the LM performance prediction model (1998) predicted that no 1024-processor system of at that time existing processors would reach the goal of finishing a one day forecast using the required resolution within half

an hour computing time. But it was observed that a Cray T3E-like network is powerful enough to work with processors which are up to ten times faster. Such a T3E-like system with nodes of approximately seven to ten GigaFlop has been estimated to be able to satisfy the required condition.

After adaptation of our model to the needs of vector machines (1999) and choosing an appropriate partitioning (long vectors) large systems of about 512 vector processors are expected to be close to the requirements. It is still an open question which performance parameters the architecture at the DWD will show at the end of 2001.

The development of the performance prediction model has led to improvements of the codes themselves. The application of the models allows to estimate the influence of hardware parameters of future computer architectures to operational weather forecast.

Acknowledgments

This work has been initiated by G.-R. Hoffmann from DWD. E. Krenzien and U. Schättler advised us how to use the codes. We had helpful discussions with E. Tschirschnitz (NEC) and R. Vogelsang (SGI). K. Cassirer (GMD), R. Hess (GMD, now DWD), and H. Schwamborn (GMD) substantially contributed to this work.

References

1. D. Dent, G. Robinson, S. Barros, L. Isaksen: *The IFS model – overview and parallel strategies.* Proceedings of the Sixth Workshop on Use of Parallel Processors in Meteorology, ECMWF, 21–25 November 1994.
2. Foster, I., Gropp, W., Stevens, R.; *The parallel scalability of the spectral transform method.* Monthly Weather Review 120 (1992) 835 – 850.
3. U. Gärtel, W. Joppich, A. Schüller, *Portable parallelization of the ECMWF's weather forecast program*, Arbeitspapiere der GMD 820, GMD, St. Augustin, 1994.
4. H. Mierendorff, W. Joppich, *Empirical performance modeling for parallel weather prediction codes*, Parallel Computing, 25 (1999), pp. 2135-2148.
5. W. Gropp and E. Lusk, *Reproducible Measurements of MPI Performance Characteristics*, Argonne NL, `http://www.mcs.anl.gov/mpi/mpich/perftest` or by ftp from `ftp://ftp.mcs.anl.gov/pub/mpi/misc/perftest.tar.gz`.
6. P. J. Mucci, K. London, J. Thurman, *The MPBench Report*, November 1998, `http://www.cs.utk.edu/~mucci` or by ftp from `ftp://cs.utk.edu/thurman/pub/llcbench.tar.gz`.
7. *Pallas MPI Benchmarks - PMB, Part MPI-1*, Revision 2.1, 1998, `http://www.pallas.de/pages/pmbd.htm`.
8. R. Reussner, *User Manual of SKaMPI (Special Karlsruher MPI-Benchmark)*, University of Karlsruhe, Department of Informatics, 1999.