P3.13 A LARGE-EDDY SIMULATION MODEL PERFORMING ON MASSIVELY PARALLEL COMPUTERS

Siegfried Raasch* and Michael Schröter
Institute of Meteorology und Climatology, University of Hannover, Germany

1 INTRODUCTION

Although many of the present-day LES studies can be carried out on single processor machines, there are still a lot of open problems in boundary layer research requiring extreme computational power (memory as well as CPU power), for instance the study of interactions between turbulent structures of different scales. Typical examples are cold-air outbreak (CAO) flows, where large-scale organized convection simultaneously occurs together with the usual small-scale convective plumes, or flows around obstacles, where the large eddies downstream of the obstacle have their origin in small-scale turbulence generated at the sharp edges of the obstacle. These studies require very high grid resolutions and/or large computational domains. Very high resolution LES could also be used to test subgrid-scale models or to check and validate nested LES.

Today, massively parallel computers allow to tackle some of these problems for the first time. Of course, on such machines also normal sized LES studies are running much faster than on single processor computers. This permits to carry out large sets of runs with parameter variations in a comparably short time.

Nevertheless, existing LES models would have to be extensively modified, i.e. parallelized, before they could be efficiently used on parallel computer architectures. The optimum parallelization strategy may depend on the type of parallel architecture. Available are machines with distributed (SGI/Cray-T3E) or with shared memory (SGI-ORIGIN). On shared memory systems, each processor has access to all of a single, shared address space (although this access is often nonuniform) while on distributed architectures each PE has his own dedicated memory. In the latter case special communication calls are necessary to exchange data between them on a special network. Today, more and more clusters of small-scale shared-memory machines are available (for example IBM pSeries 690). Data exchange between these shared-memory nodes is also done by a special network, so these machines combine the characteristics of distributed and shared memory systems. All these multi processor systems are available with more than 1000 processor elements (PEs). Regardless of the system architecture, an optimally parallelized LES code should meet at least three requirements: (1) The additional communication overhead between PEs caused by the parallelization method in combination with the special architecture of the machine should be small compared with the total CPU time of the code. (2) The program should show good scalability, i.e. it should efficiently run on an arbitrary large number of PEs. (3) There should be a good load-balancing between the PEs in order to avoid idling of PEs.

*Corresponding author address: Siegfried Raasch, Institut für Meteorologie und Klimatologie, Universität Hannover, Herrenhäuser Str. 2, 30419 Hannover, Germany; raasch@muk.uni-hannover.de

2 THE LES-MODEL PALM

Because the parallelization affects the basic structure and many parts of the LES code, we decided to develop and write the code of our new **PA**rallelized **L**ES **M**odel **PALM** from scratch. Nevertheless, the new model was based on the well established methods and the same systems of equations that we used before in our old sequential LES model, which had been successfully applied earlier to problems of atmospheric and oceanic convection (Schröter et al., 2000, e.g.). It solves the non-hydrostatic Boussinesq-approximated Navier-Stokes equations and uses a one-and-a-half-order subgrid closure scheme. It contains a water cycle with cloud formation and precipitation processes and it takes into account infrared radiative cooling at the cloud tops. Time integration is performed using the leap-frog scheme. Lateral boundary conditions of the model are cyclic and Monin-Obukhov similarity is assumed between the surface and the first computational grid level above.

PALM is originally designed for use on parallel computers with distributed memory, but it also runs efficiently on machines with shared memory architecture (currently, we are also optimiz-

ing the code for use on clustered systems). As it is totally recoded, we took this as a chance to switch from FORTRAN77 to FORTRAN90 in order to make use of helpful features like array assignments available in the new standard.

2.1 PARALLELIZATION TECHNIQUE

In case of an LES model for boundary layer flows, we found that the requirements for an optimally parallelized code mentioned in Section 1 are achieved best by the method of two-dimensional domain decomposition. The total computational domain is divided into equally sized subdomains, which are assigned to the PEs - one subdomain per each PE -, and each PE solves the whole set of model equations on its subdomain. In case of PALM, the decomposition is done in the two horizontal directions x and y (s. Figure 1) because decomposing the domain in this way guarantees

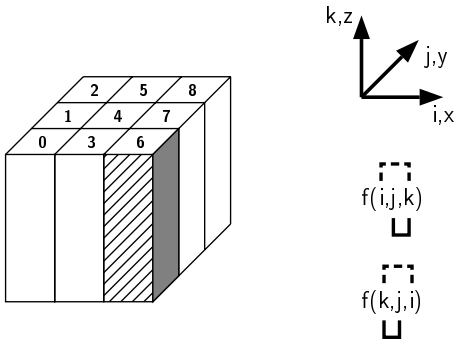

Figure 1: Principal view of the two-dimensional domain decomposition. Subdomains of each processor element (PE) are labelled with the PE number. Grid points on the side walls of the subdomains (shaded rectangles) are stored almost consecutively in memory, when array indices are arranged as f(k,j,i).

that all subdomains have identical properties, especially with respect to the bottom and top boundary conditions. Additionally, the decomposition provides scalability up to several thousand PEs. In such a case, a one-dimensional decomposition would require the model to have several thousand grid points in at least one direction.

PALM uses finite differencing, since the numerous calculations of horizontal Fourier transforms necessary when using pseudospectral differencing would significantly decrease the model performance in case of a 2D-decomposition. Problems with local data dependencies, which arise from the need of computing central finite differences at the side boundaries of the subdomains,

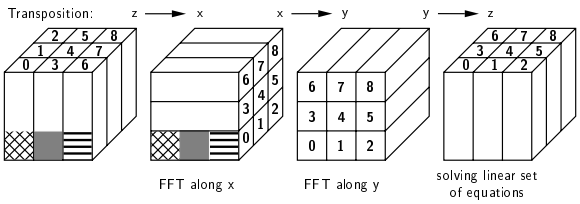

Figure 2: Assignment of subdomains to the PEs (0-8) at the beginning (left) and after each transposition step needed for solving the Poisson equation (only half of the cycle is shown). Those parts of the Poisson solver to be carried out after the respective transpositions are listed at the bottom.

are solved by introducing additional ghost points, which means that arrays are expanded in each horizontal direction to hold necessary data from neighbouring processor elements. In order to achieve an optimal speed for the exchange, grid points at the side boundaries should be arranged in consecutive order in memory. This is ensured best (in FORTRAN) when the order of array indices, which has been f(i,j,k) in the old model, is rearranged to f(k,j,i), where indices i, j, and k are identified with coordinates x, y, and z, respectively (s. Figure 1).

The solution of the Poisson equation is also complicated by the domain decomposition, because non-local data dependencies appear in all three directions. In the direct method that we are actually using, FFTs have to be performed along x and y and a tridiagonal system of linear equations has to be solved in z-direction. This method is parallelized as follows: first, the three dimensional pressure field is transposed in such a way that all related data values along the x-direction (of the total domain) reside on the same PE (s. Figure 2). After that, the x-FFT can be done in conventional manner (for example by using FFT routines from standard scientific libraries), followed by a transposition from x to y, an FFT along y and another transposition from y to z. Now the transformed perturbation pressure can be computed by solving the resulting tridiagonal linear system (also non-parallelized). Finally the procedure has to be passed through in backward order to transform the pressure from phase space back to Cartesian space.

Although the transpositions lead to extensive data traffic between all PEs, the overhead produced by this procedure on machines with fast PE interconnections (SGI/Cray-T3E, SGI-Origin) is small compared with the total computing time of the model (see Section 2.2). Recently, we imple-

mented a multigrid method to solve the Poisson equation on cluster architectures with slow node interconnections as an alternative to the FFT method (see Section 4).

Communication between the PEs is realized by the message passing interface MPI, which has become a quasi-standard on massively parallel systems. The cyclic horizontal boundary conditions of the model are realized implicitly by creating a virtual two-dimensional topology of processors where the processors at the corresponding 'ends' of single rows or columns are connected together, respectively. In this virtual topology, the leftmost processor in a row, for example, knows, that it has to exchange ghost points with the rightmost processor of the same row and vice versa. Due to the periodic boundary conditions, identical lines of code can be executed on each PE, which allows the parallel structure of the model to be very simple.

2.2 PERFORMANCE ANALYSIS

PALM has been extensively tested on Cray-T3E machines. Figure 3 shows the model speed-up $s(P)$ (defined by the ratio between the computational time needed on one PE and the time needed on P PEs) for runs with a 'small' domain of $160 \times 160 \times 64$ grid points. In case of an ideal scalability the speed-up should be doubled by doubling the number of PEs used. The model meets this condition very well. For PE numbers larger than 100 the speed-up is a little bit smaller than in the ideal case, because the subdomains become too small. In these cases the ratio R between the total number of grid points of a subdomain and the number of ghost points is getting larger so that the communication overhead for exchanging ghost points becomes noticeable. Figure 4 illustrates the communication

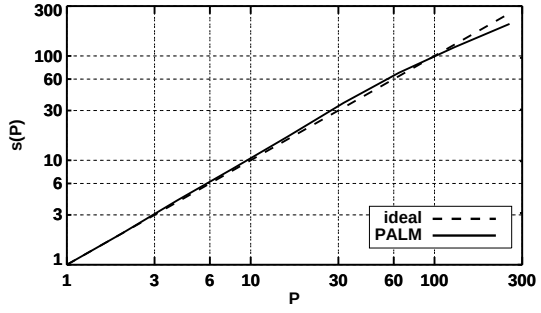

Figure 3: Speed-up $s(P)$ of PALM on a Cray-T3E as a function of the number of PEs P used.

overhead. Shown are execution times relative to the total execution time of PALM for the exchange of ghost points (MPI function *sendrecv*) and for the data transpositions (MPI function *alltoallv*) as a function of the grid ratio R. In this test all runs were performed on a constant number of 512 PEs whereas the domain sizes vary from $32 \times 32 \times 160$ ($R = 0.3$) up to $1216 \times 1216 \times 160$ grid points ($R = 12.6$). Corresponding subdomains vary from $2 \times 1 \times 169$ to $76 \times 38 \times 160$ grid points, excluding ghost points. The additional ghost points increase the total number of grid points by 400 % in case of the smallest total domain and by 8 % in case of the largest domain, which demonstrates that the memory overhead due to parallelization remains small provided that the subdomains are sufficiently large. Figure 4 shows that the total communication time of the model remains smaller than 10 % of the total execution time of the model, as long as $R > 2$. For small subdomains the exchange of ghost points dominates the communication time, while the transposition time is nearly independent from R. Using the multigrid method instead of the FFT

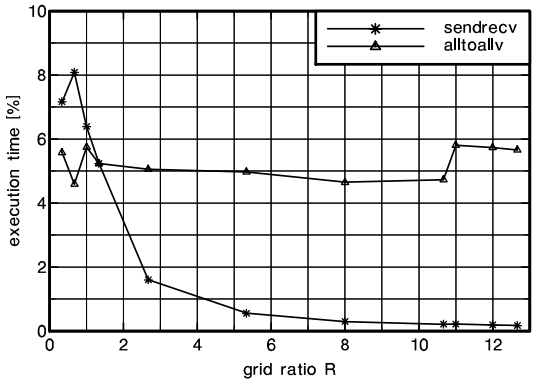

Figure 4: Relative execution time needed for communication between PEs as a function of the grid ratio R. See text for further explanations.

method for solving the Poisson equation eliminates the need for the exchange of large amounts of data due to the transposition. Presumably, the multigrid method will be of advantage for cluster systems. On the Cray-T3E, this method needs as much CPU time as the FFT method, as long as the total number of grid points is large enough. Figure 5 shows the total CPU time used for 205 timesteps with 384^3 grid points for different numbers of PEs. Additionally, the figure demonstrates that the model also scales on shared memory systems (SGI-Origin 2800). Overall, PALM meets all the requirements for an optimally parallelized

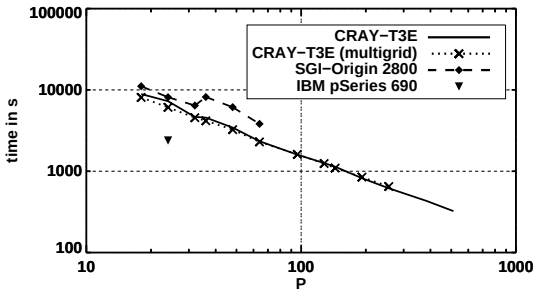

Figure 5: Comparison of PALM execution time on different massively parallel computers.

code mentioned in Section 1. More details of the performance analysis can be found in Raasch and Schröter (2001).

2.3 ADDITIONAL MODEL FEATURES

Even modern graphic hard- and software on front-end machines is unable to process the huge three-dimensional data at a sufficient speed. We are solving this difficulty by running parts of the visualization process in parallel as a part of the numerical model. Instead of raw data, only graphics primitives (descriptions of polygons, lines or points in 3D space) are sent to the graphic workstation from which the final pictures can be created with considerably reduced expense of computational power. Visualization examples can be found under http://www.rvs.uni-hannover.de/projekte/tele-immersion/demo/cbl.html. PALM includes a particle model which can be used for flow visualization as well as for dispersion modelling. A detailed documentation is available via the PALM-group web-pages of our institute (http://www.muk.uni-hannover.de/∼raasch/PALM_group).

3 APPLICATIONS

PALM has recently been applied successfully to study boundary layer turbulence above inhomogeneous terrain (Raasch and Harbusch, 2001, see also papers in this preprint volume: P6.18, 15.8), the turbulent flow in the vicinity of Arctic leads (Weinbrecht and Raasch, 2001), and organized convection during cold air outbreaks (see paper 3.5) as well as to improve turbulence parameterization (see paper P5.10). The CAO study used a very large domain with $704 \times 704 \times 80$ grid points. The model is also used for comparison with near-surface turbulence measurements (see papers 9.4 and 9.9). For these studies, simulations with grid spacings of 0.5 m and $700 \times 700 \times 350$ grid points have been performed.

4 OUTLOOK

First runs of PALM on 24 PEs of an IBM pSeries 690 are quite promising (see Figure 5). Nevertheless, considerable effort will be neccessary to account for the very slow node interconnection of the cluster, which will become apparent if more than 32 PEs are used. Actually, we are working on the implementation of non-cyclic horizontal boundary conditions in order to study the effect of skewed inversion layers on the cloud street structure within CAOs. We also plan to implement an irregular lower boundary, which is needed to simulate the turbulent flow around buildings, and to realize grid-nesting by coupling two or more simulations running simultaneously on different sets of PEs.

ACKNOWLEDGEMENTS

This project was funded by the Deutsche Forschungsgemeinschaft under contract numbers RA 617/3-3 and 446 KOR 113/124. Most of the runs were performed on massively parallel machines at the Konrad Zuse Zentrum für Informationstechnik in Berlin and at the John von Neumann Institut für Computing in Jülich, Germany.

References

Raasch, S., G. Harbusch, 2001: An analysis of secondary circulations and their effects caused by small-scale surface inhomogeneities using large-eddy simulation. – Boundary-Layer Meteorol. **101**, 31–59.

Raasch, S., M. Schröter, 2001: **PALM** - A large-eddy simulation model performing on massively parallel computers. – Meteorol. Z. **10**, 363–372.

Schröter, M., J. Bange, S. Raasch, 2000: Simulated airborne flux measurements in a LES generated convective boundary layer. – Boundary-Layer Meteorol. **95**, 437–456.

Weinbrecht, S., S. Raasch, 2001: High resolution simulations of the turbulent flow in the vicinity of an Arctic lead. – J. Geophys. Res. **106**, C11, 27035–27046.