The 1. Euro–Conference on Parallel and
Distributed Computing for Computational Mechanics.
April 26 – May 1st, 1997, Lochinver, Scotland, UK.

A COMPARISON OF PARALLEL ALGORITHMS FOR THE NUMERICAL SIMULATION OF MULTIPHASE FLOWS

E. Wassen, Th. Frank, Q. Yu
Technical University of Chemnitz-Zwickau
Faculty of Mechanical Engineering and Process Technology
Research Group of Multiphase Flow
Chemnitz, Germany

Abstract

Two parallel algorithms for the Lagrangian simulation of disperse multiphase flows are presented and compared. The first algorithm is based on the domain decomposition method which is widely used for the parallel computation of fluid flows. The numerical grid is devided in partitions. All particle trajectories that cross a certain partition are calculated by the same processor. In the second parallelization method there is no fixed relation between a processor and a grid partition. Each processor can calculate trajectories in any part of the flow geometry. Therefore a complex memory management is established. The results of the test case calculations show that for the second method the load balancing is much better and the total computing time much lower than for the first.

1 Introduction

Recently the Lagrangian simulation has become an efficient and widely used method for the calculation of various kinds of 2- and 3-dimensional disperse multiphase flows (e.g. gas-particle flows, gas-droplet flows). Considering the field of computational fluid dynamics, the Lagrangian simulation of coupled multiphase flows with strong phase interaction ranks among the applications with the highest demand on computational power and system recources. Massively parallel computers provide the capability for cost-effective calculations of multiphase flows. In order to use the architecture of parallel computers efficiently, new solution algorithms have to be developed. Difficulties arise from the complex data dependence between the fluid flow calculation and the prediction of particle motion, and from the generally inhomogeneous distribution of particle trajectories in the flow field.

2 Physical and Mathematical Fundamentals

2.1 Basic Equations of Fluid Motion

The fluid phase considered here is assumed to be Newtonian and to have constant physical properties. The fluid flow is 2-dimensional, steady, incompressible, turbulent and isothermal. Fluid turbulence is modelled using the standard $k-\varepsilon$ model and neglecting the influence of particle motion on fluid turbulence. Under these assumptions the time-averaged equations describing the motion of the fluid phase are given by the following form of the general transport equation:

$$\frac{\partial}{\partial x}(\rho_F\, u_F\, \Phi) + \frac{\partial}{\partial y}(\rho_F\, v_F \Phi) = \frac{\partial}{\partial x}\left(\Gamma\, \frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma\, \frac{\partial \Phi}{\partial y}\right) + S_\Phi + S_\Phi^P \qquad (1)$$

Here Φ is a general variable, Γ a diffusion coefficient, S_Φ a general source term and S_Φ^P symbolizes the source term due to momentum exchange between the fluid and the particle phase. The variables u_F and v_F represent the fluid velocity components, k is the turbulent kinetic energy and ε is the rate of dissipation of k.

A detailed description of all terms and their correlations is shown in Table 1. In this table ρ_F is the fluid density and μ is the laminar viscosity.

2.2 Equations of Motion of the Disperse Phase

The disperse phase is treated by the application of the Lagrangian approach, i.e. discrete particle trajectories are calculated. Each calculated particle represents a large number of physical particles of the same physical properties. The prediction of the particle trajectories is carried out by solving the ordinary differential equations for the particle location and velocities. Assuming that the ratio of fluid density to particle density is small ($\rho_F/\rho_P \ll 1$) these equations read:

$$\frac{d\,x_P}{dt} = u_P \quad , \quad \frac{d\,y_P}{dt} = v_P \qquad (2)$$

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix} u_P \\ v_P \end{bmatrix} &= \frac{3}{4}\frac{\nu\,\rho_F}{\rho_P\, d_P^2} Re_P \left(C_D(Re_P) \begin{bmatrix} u_F - u_P \\ v_F - v_P \end{bmatrix} \right. \\ &+ \left. C_M(\sigma) \begin{bmatrix} v_F - v_P \\ u_P - u_F \end{bmatrix} \right) \\ &+ \frac{\rho_P - \rho_F}{\rho_P} \begin{bmatrix} g_x \\ g_y \end{bmatrix} \end{aligned} \qquad (3)$$

Φ	S_Φ	S_Φ^P	Γ
1	0	0	0
u_F	$\frac{\partial}{\partial x}\left(\Gamma\,\frac{\partial u_F}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\,\frac{\partial v_F}{\partial x}\right)-\frac{\partial p}{\partial x}$	$S_{u_F}^P$	μ_{eff}
v_F	$\frac{\partial}{\partial x}\left(\Gamma\,\frac{\partial u_F}{\partial y}\right)+\frac{\partial}{\partial y}\left(\Gamma\,\frac{\partial v_F}{\partial y}\right)-\frac{\partial p}{\partial y}$	$S_{v_F}^P$	μ_{eff}
k	$P_k-\rho_F\,\varepsilon$	0	$\frac{\mu_t}{\sigma_k}$
ε	$\frac{\varepsilon}{k}(c_{\varepsilon_1}\,P_k-c_{\varepsilon_2}\,\rho_F\,\varepsilon)$	0	$\frac{\mu_t}{\sigma_\varepsilon}$
$P_k=\mu_t\left\{2\cdot\left[\left(\frac{\partial u_F}{\partial x}\right)^2+\left(\frac{\partial v_F}{\partial y}\right)^2\right]+\left(\frac{\partial u_F}{\partial y}+\frac{\partial v_F}{\partial x}\right)^2\right\}$			
$\mu_{eff}=\mu+\mu_t \quad , \quad \mu_t=\rho_F c_\mu \frac{k^2}{\varepsilon}$ $c_\mu=0.09 \quad , \quad c_{\varepsilon_1}=1.44 \quad , \quad c_{\varepsilon_2}=1.92$ $\sigma_k=1.0 \quad , \quad \sigma_\varepsilon=1.3$			
$S_{u_i}^P$: see Equation 4			

Table 1: Source terms and diffusion coefficients for different variables Φ

with :

$$Re_P=\frac{d_P\,v_{rel}}{\nu} \quad , \quad \sigma=\frac{1}{2}\frac{d_P\omega_{rel}}{v_{rel}}$$
$$v_{rel}=\sqrt{(u_F-u_P)^2+(v_F-v_P)^2}$$

In these equations the subscript P indicates *Particle* and the subscript F indicates *Fluid*. ν is the fluid kinematic viscosity, d_P the paticle diameter and ω_{rel} the absolute value of the relative rotational velocity between fluid and particle. The first term on the right hand side of Equation (3) represents the drag force exerted on the particle by the fluid. The second term gives the lift force due to particle rotation (Magnus force) and the third term the gravitational force. The values for the constant coefficients C_D and C_M can be found in [3]. The effect of fluid turbulence on the motion of the disperse phase is modelled by the Lagrangian Stochastic-Deterministic (LSD) turbulence model proposed by Schönung [8] and Milojević [5]. The particle's influence on the fluid phase is modelled by the PSI-cell (Particle-Source-In-cell) method [1, 2]. Therefore an additional source term is introduced in the fluid momentum equation as shown in Table 1. This source term is calculated as follows:

$$\begin{aligned} S_{u_i}^P = & - \frac{1}{V_{ij}}\sum m_P\dot{N}_P\Big[u_{Pi,out}-u_{Pi,in} \\ & - g_i(1-\frac{\rho_F}{\rho_P})(t_{out}-t_{in})\Big] \end{aligned} \tag{4}$$

Here V_{ij} is the volume of a cell of the numerical grid. $\dot{N}_P$ is the number of physical particles per unit time represented by the currently calculated trajectory, and m_p is the mass of a single particle. The subscripts *in* and *out* indicate the locations where the trajectory enters and leaves the grid cell, respectively.

2.3 Solution Algorithm

For the numerical solution of the equations described in the above sections the physical space must be discretized. Therefore a boundary–fitted, non–orthogonal numerical grid is used. The grid is block–structured and consists of quadrangular cells. The equations of fluid motion (1) are numerically solved on the basis of a finite volume discretization. A pressure correction technique of SIMPLE kind [6] is applied. The program package used to predict the motion of the fluid phase is based on developments by Perić and Lilek [7]. When a converged solution for the fluid flow field has been calculated, the prediction of the particle motion is carried out. Therefore Equation (3) is solved by using a standard Runge-Kutta scheme of 4th order accuracy. The source terms according to Equation (4) are predicted simultaneously during trajectory calculation. After all particle trajectories are calculated the source terms are included in the fluid momentum equations and a new converged solution for the fluid flow field is computed.
The iterative algorithm for the numerical simulation of the coupled two–phase flow is summarized as follows:

1. calculation of a converged solution for the fluid flow field without taking the source terms of the disperse phase into account
2. tracing a large number of particles through the flow field and computing the source terms simultaneously
3. recalculation of the fluid flow field considering the source terms of the disperse phase
4. repeating Steps 2 and 3 until the solution of the coupled equations has converged.

3 Parallelization

3.1 The Parallel Algorithm for Fluid Flow Calculation

The use of parallel computers generally has two advantages. The first is a reduction of the total amount of time needed for solving a special problem, e.g. computing a flow field. The second is the ability to solve "larger" problems. Considering the case of a flow field calculation this means that finer numerical grids, i.e. containing more and smaller grid cells, can be used and thus the accuracy of the solution can be improved. Both advantages are reached by dividing the original problem in a number of smaller problems that are solved separately and simultaneously by the processors of a parallel computer. Hence the solution algorithm has to be adapted to the architecture of the computer in order to use its computational power efficiently.

In the case of the flow field calculation the division of the original problem in smaller problems is carried out by partitioning the numerical grid. The so called domain

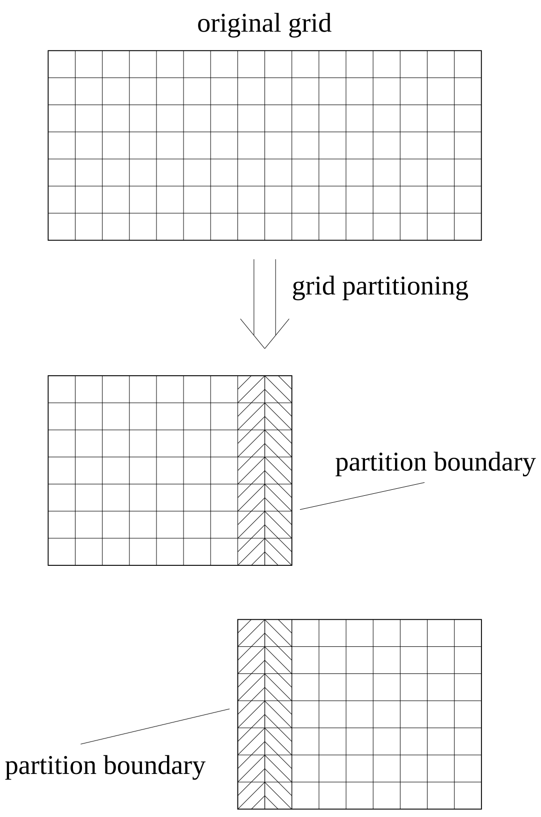

Figure 1: Principle of grid partitioning

decomposition method is illustrated in Figure 1. By applying this method the flow domain, which is represented by the numerical grid, is divided in smaller parts. The flow on each part of the flow domain, i.e. the numerical grid, is calculated by a single processor of the parallel computer. The solution algorithm requires that the resulting partitions overlap as shown in Figure 1. As a consequence the total number of grid cells and hence the total computational effort is increased. This fact reduces the efficiency of the parallel algorithm. The flow field data in the interior part of a partition and at the physical boundaries can be calculated independently. At a parallel boundary, i.e. a boundary between adjacent parts of the grid, an indepedent calculation is not possible. There is the need to exchange data between the neighbouring partitions. Therefore a kind of communication between the processors of the parallel computer must be established. In general the time needed for communication is significant and increases the total time needed for solving a special problem. Thus inter-processor communication is another factor that reduces the efficiency of the parallel solution algorithm. When a numerical grid is partitioned, the resulting parts of the grid should be sized almost equally. This implicates that the computational load is balanced well among the processors. Otherwise there may arise waiting times because computation and communication are not synchronized.

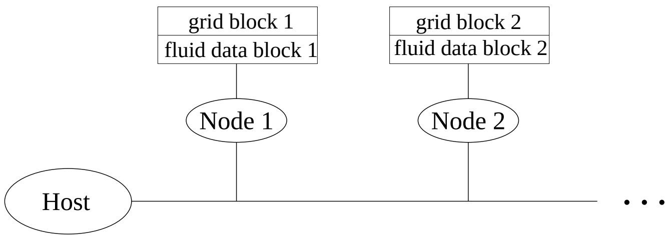

Figure 2: Scheme of processes for the Domain Decomposition Method

3.2 Parallel Algorithms for Particle Trajectory Calculation

The prediction of the motion of the disperse phase is carried out by the application of the Lagrangian approach as described in Section 2.2. Considering the parallelization of this algorithm there are two important features. The first is that in general particle trajectories are not uniformly distributed in the flow domain even if there is a uniform distribution at the inflow cross-section. This is caused by the high density ratio ρ_P/ρ_F by what particle motion is determined rather by inertia than by fluid forces. As a second characteristic the location and distribution of particle trajectories is not known at the beginning of the computation. Considering these features the following parallelization methods have been developed [4]:

Method 1: Domain Decomposition (DD) - Method

As indicated by its name this method is based on the domain decomposition used for calculating the flow field. An explicit host-node scheme is established as illustrated in Figure 2. The trajectory calculation is done by the node processes whereas the host process carries out only management tasks. The node processes are identical to those that do the flow field calculation. The grid and fluid data of a single grid partition are stored in the local memory of each node process.

The principle of this method is that in a node process only those trajectory parts are calculated that cross the grid part assigned to this process. In other words, all trajectory parts crossing a certain partition are calculated by the process related to this partition. The particle state (location, velocity, diameter, ...) at the entry point to the current partition is sent by the host to the node process. The entry point can either be an inflow cross section or a boundary to a neighbouring partition. After the trajectory part computation is finished, the particle state at the exit point (outlet cross section or partition boundary) is sent back to the host. If the exit point is located at the interface of two grid parts, the host sends the particle state to the process related to the neighbouring part for continuing trajectory computation.

An advantage of this kind of communication scheme is that direct inter-node communication is avoided. If the particle data were exchanged directly between the nodes, large waiting times would arise because the times needed for predicting different trajectory parts in general differ significantly. By involving a host process as a distributor or collector, respectively, of particle states the nodes hardly ever have to wait for the arrivel of data, especially when host-node communication is performed asynchronously.

As pointed out above in general two-phase flows the trajectories of the disperse phase are inhomogeneously distributed in the flow field. This can cause a pour load balancing when applying the DD method. In regions where many trajectories are located the computational effort is higher than in other regions. Hence the computational load on some processes is high and on other processes it is low due to the fixed relation between grid partitions and processes. Futhermore, even if the particle trajectories are distributed uniformly in physical space the computational load can be distributed non-uniformly during computing time. This case occurs e.g. if a simple pipe or channel flow is considered where the grid partitions are arranged in a row along the pipe or channel axis. At the beginning of computation the computational load near the inflow cross section is high and near the outflow cross section it is low. Towards the end of computation the situation is vice versa.

Method 2: Distributed Shared Memory (DSM) - Method

This method has been developed to overcome the disadvantages of the DD method concerning the balancing of the computational load. In the DSM method there exist three classes of processes (Figure 3). Just as in the DD method the host process distributes the particle initial conditions among the calculating nodes and collects the particle's state when the trajectory segment calculation has been finished. The calculating nodes receive the particle initial condition from the host and predict the trajectory segment on a certain grid partition. In contrast to the DD method there is no fixed relation between a node and a grid partition. Hence it is possible to compute different trajectories on one grid partition at the same time by different node processes. When a particle initial state is received by a node there are two situations that may occur. The first is that the particle is located on the part of the numerical grid that is already present at the node. In the second situation this grid part is currently not present. If the latter one occurs the partition must be loaded from a memory manager process. Each memory manager stores permanently the grid geometry and fluid data for a single grid part. Thus a flexible assignment of grid partitions to calculating nodes is made possible according to the actual requirements of trajectory calculation.

The tasks of the different process classes are summarized as follows:

Host:

1. send particle initial conditions to nodes
2. if a node has finished a trajectory segment calculation, receive back the particle state
3. treat particle state as new initial condition and send it to a node for further calculation

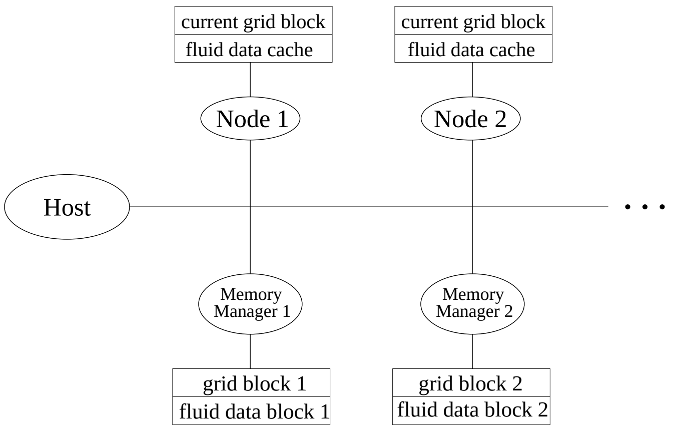

Figure 3: Scheme of processes for the Distributed Shared Memory Method

4. repeat Steps 1-3 until all trajectories have reached the outlet cross section

Node:

1. receive particle initial condition from host
2. check if particle is located in the grid part currently loaded; if not, load new grid part from the memory manager where it is stored
3. calculate trajectory part
4. send particle state at the exit point (e.g. partition boundary or outlet cross section) back to host
5. repeat Steps 1-4 as long as the host sends initial conditions

Memory Manager:

1. store grid and fluid data for a certain grid block permanently
2. if a node requires data of the stored partition, send data to node

There are some further details of the DSM method that should be pointed out. For predicting a trajectory segment a node needs the complete geometry data of the actual grid part but only the fluid data for the grid cell in which the particle is currently located. Hence there are two thinkable algorithms concerning the treatment of the fluid data. The first is loading the complete fluid data for the whole partition from the memory manager at the beginning of computation. This algorithm implies a single communication while transferring a large amount of data. The second possibility is to load the fluid data only for the single grid cell the particle is currently located in and to reload the fluid data every time the particle enters a new cell. The second algorithm implies a large number of communications transferring a low amount of data. Which algorithm is the more efficient one depends on the available memory space and the velocity of communication und thus is highly machine dependent. Furthermore,

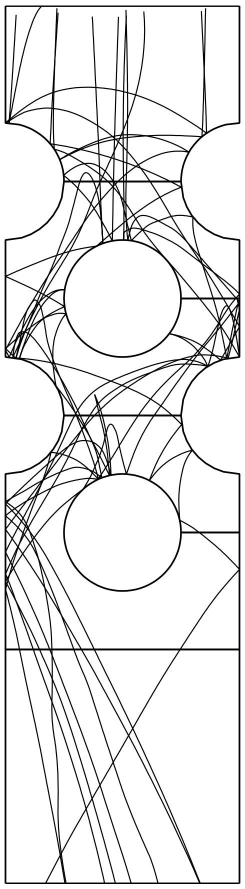

Figure 4: Particle trajectories in a heat exchanger geometry

other algorithms are thinkable in between, e.g. transferring data of a small environment around the current cell and/or caching of data that have already been transferred before.

As described above the DSM method involves three classes of processes: the host, the calculating nodes and the memory manager tasks. The node processes are characterized by needing little memory space but doing a large number of computations. On the other hand the memory manager processes need much space but do not carry out any computation. Hence, if the parallel computer allows running more than one process on a single CPU, a node and a memory manager process can be executed on the same processor without restricting each other significantly.

4 Results and Discussion

4.1 Description of the Test Case

For the test case calculations a vertical two-phase flow in a heat exchanger geometry has been simulated. The flow is 2-dimensional, steady, incompressible and turbulent. The fluid phase is chosen as air with standard aerodynamical properties. The disperse phase consists of solid silicon particles with a density of $\rho_P \approx 2500 kg/m^3$ and a medium particle diameter of $d_P = 260\mu m$. In Figure 4 some par-

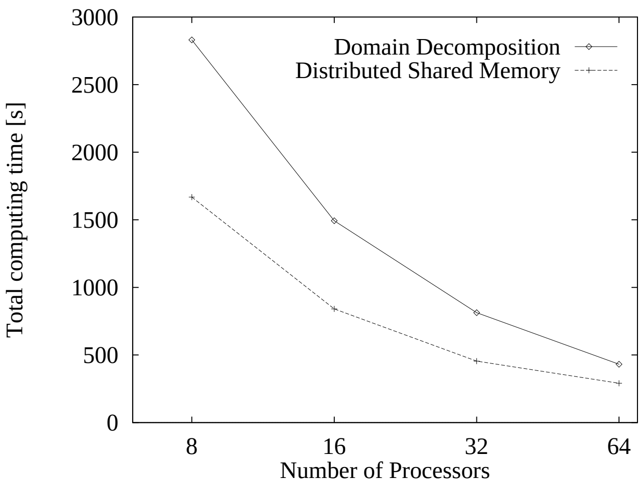

Figure 5: Total computing time (calculation time + communication time) needed for calculating 5000 trajectories

ticle trajectories are shown. The inlet cross section for fluid and particles is located at the upper border of the figure, the outlet cross section at the lower border. The right and left boundaries of the geometry are symmetry boundaries.

It can be easily seen that due to the particle-wall interaction the particles stay much longer in the middle of the geometry than near the inlet or outlet cross sections. Obviously this behaviour causes an inhomogeneous distribution of the computational effort.

All test case calculations have been carried out on a Cray T3D using the message passing interface MPI.

4.2 Comparison of the Parallelization Methods

For each of the algorithms described in Section 3.2 test case calculations have been done using different numbers of processors. In Figure 5 the total computing times, i.e. calculation time plus communication time, are given. For both algorithms the computing time decreases substantially with an increasing number of processors. The DSM method is found to achieve computing times that are nearly 50 % lower than those for the DD method.

The Figures 6 - 9 show the results obtained from calculations involving 16 node processors. The results for other numbers of processors are qualitatively similar.

Figure 6 shows the time that each of the 16 processors spends on trajectory calculation. This is the time only needed for computing purposes, i.e. not including any communication or waiting times. The processors with low processor numbers are related to grid partitions near the inlet cross section. The part of the grid near the outlet cross section is assigned to the processors having high processor numbers. Comparing Figure 6 with Figure 4 it is obvious, that the computational load on a certain processor directly depends on the number and lenght of trajectories that cross the grid part related to the processor. Since the assignment of the grid part to the processor is fixed throughout the calculation, the computational load is not balanced well.

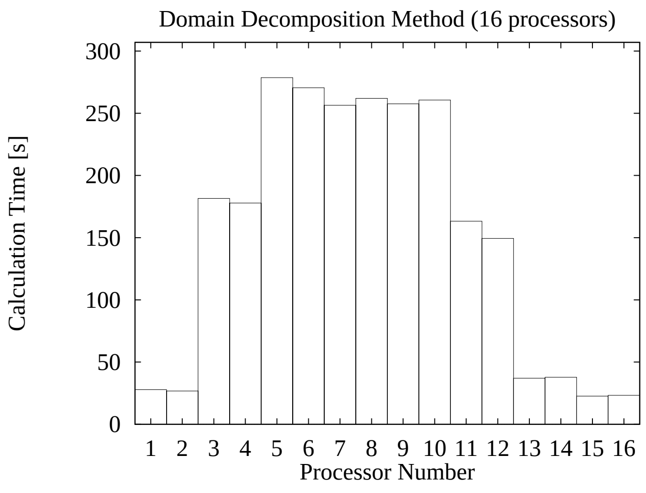

Figure 6: Calculation time (without communication time) needed for calculating 1000 trajectories on 16 node processors using the Domain Decomposition Method

In Figure 7 the communication time for the same case is shown. The communication time includes the time for data exchange and the waiting time. Since in the DD method only few data have to be exchanged, the communication time mainly consists of waiting time, i.e. time during which a node processor waits for receiving a particle initial condition from the host. The largest waiting times are observed for the processors with the lowest computational load and vice versa.

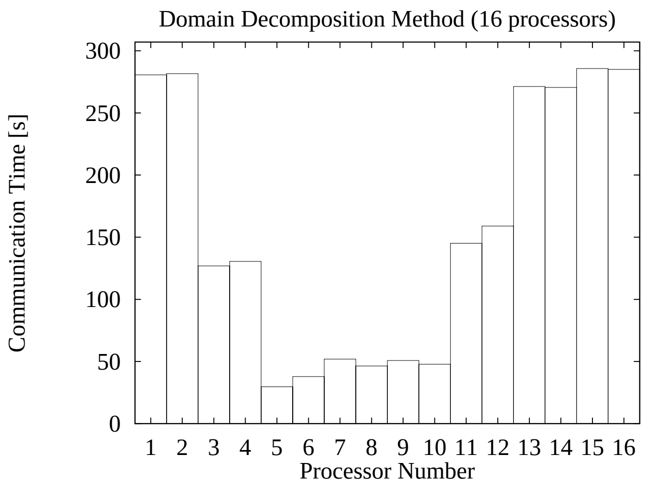

Figure 7: Communication time (including waiting time) needed for calculating 1000 trajectories on 16 node processors using the Domain Decomposition Method

The Figures 8 and 9 show the results obtained by applying the DSM method. These results differ widely from those obtained for the DD method. In the DSM method there is no fixed relation between a node processor and a certain grid partition. A node can calculate a trajectory segment on every partition depending on the particle initial condition received from the host. The calculation time (not including communication and waiting time) for each node processor is given in Figure 8. It can be seen that the computational load is balanced very well among the processors. Furthermore, the maximum calculation time is much lower than the maximum time observed for the DD method.

The communication time (including waiting time) measured for the DSM method is presented in Figure 9. Just as the calculation time the communication time is nearly the same for all processors. In comparison to the DD method the communication time is much lower, although the amount of data to be exchanged is higher. This is due to the fact that in the DSM method there arises hardly any waiting time. When a processor has finished a trajectory segment calculation, immediately a new initial condition is send by the host. Thus the communication time measured for the DSM method mainly consists of time for data exchange, whereas in the DD method it mainly consists of waiting time.

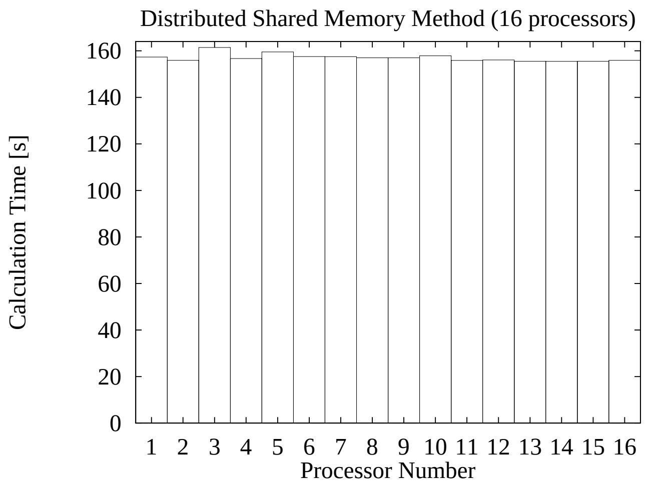

Figure 8: Calculation time (without communication time) needed for calculating 1000 trajectories on 16 node processors using the Distributed Shared Memory Method

5 Conclusion

Two parallelization methods for the Lagrangian simulation of disperse multiphase flows are presented. The Domain Decomposition (DD) Method is based on the method having the same name widely used for the parallel simulation of fluid flows. There is a fixed assignment of grid partitions to processors. In the Distributed Shared Memory (DSM) method a complex memory management is established by which every processor is able to calculate particle trajectories in any part of the numerical grid.

Both parallel algorithms provide the capability to reduce computing time substantially. The test case calculations presented here show that for the DSM method the load balancing is much better and the total computing time is much lower than for the DD method.

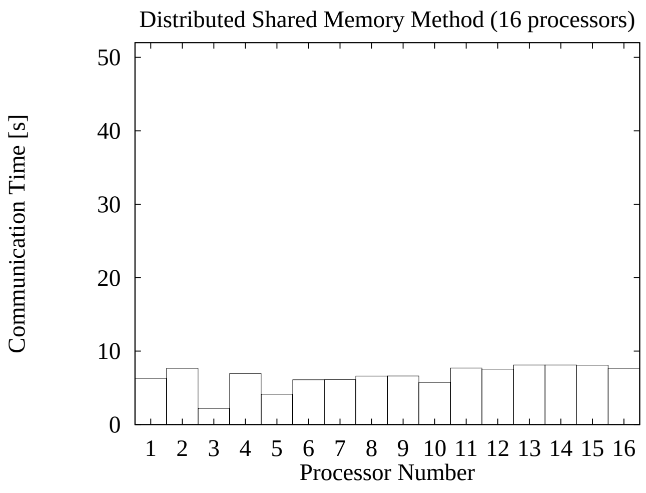

Figure 9: Communication time (including waiting time) needed for calculating 1000 trajectories on 16 node processors using the Distributed Shared Memory Method

References

[1] Crowe, C.T., Sharma, M.P., Stock, D.E., "The Particle–Source–In Cell (PSI–Cell) Model for Gas–Droplet Flows", in "Trans. of ASME, J. Fluids Eng.", Vol. 99, pp. 325–332., 1977

[2] Crowe, C.T., "REVIEW — Numerical Models for dilute Gas–Particle Flows", in "Trans. of ASME, J. Fluids Eng.", Vol. 104, pp. 297–303., 1982

[3] Frank, Th., "Numerische Simulation der feststoffbeladenen Gasströmung im horizontalen Kanal unter Berücksichtigung von Wandrauhigkeiten", PhD Thesis, Techn. University Bergakademie Freiberg, Germany, 1992

[4] Frank, Th., Wassen, E.,"Parallel solution algorithms for Lagrangian simulation of disperse multiphase flows", in "Proceedings of the 2nd Int. Symposium on Numerical Methods for Multiphase Flows", ASME Fluids Engineering Division Summer Meeting, San Diego, CA, U.S.A., July 7-11, 1996

[5] Milojević, D., "Lagrangian Stochastic–Deterministic (LSD) Predictions of Particle Dispersion in Turbulence", in "Part. Part. Syst. Charact.", Vol. 7, pp. 181–190., 1990

[6] Patankar, S.V., "Numerical Heat Transfer and Fluid Flow", McGraw–Hill, New York., 1980

[7] Perić, M., Lilek, Ž., "Users Manual for the FAN–2D Software for the Calculation of Incompressible Flows", Institut für Schiffbau der Universität Hamburg, Germany., 1993

[8] Schönung, B., "Comparison of Different Dispersion Models for Particles in Lagrangian and Eulerian Prediction Codes", in "Proceedings of the International Conference on Fluid Mechanics", Peking, Peking University Press, China, July 1.-4., 1987