**2nd Int. Symposium on Numerical Methods for Multiphase Flows,
ASME Fluids Engineering Division Summer Meeting,
July 7–11, 1996, San Diego, CA, U.S.A.
Proceedings, FED–Vol. 236, Vol. 1, pp. 11–20**

PARALLEL SOLUTION ALGORITHMS FOR LAGRANGIAN SIMULATION OF DISPERSE MULTIPHASE FLOWS

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

1. ABSTRACT

The paper deals with different methods for the parallelization of numerical algorithms which are widely used for the prediction of disperse multiphase flows (e.g. gas–particle or gas–droplet flows). The underlying numerical algorithm is based on the Lagrangian (PSI–cell) approach, where trajectories of a large number of particles/droplets are calculated from the equations of motion of the disperse phase along with the continuity and momentum equations of the fluid phase.

All parallelization methods are developed for MIMD computer architectures and are based on a former serial implementation of the Lagrangian approach [3, 4, 5]. Parallelization of the solution algorithm for the set of continuity, Navier–Stokes and turbulence model equations is carried out by application of a domain decomposition method to the block structure of the numerical grid as proposed by Perić et al. in [10, 9]. For the Lagrangian solution algorithm 3 different parallelization methods are investigated and compared with each other. Results of performance evaluations are given for two typical test cases, which are calculated on a wide range of numbers of processors of a massively parallel MIMD machine.

2. INTRODUCTION

Recently 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 or gas–droplet flows). On the other hand Lagrangian simulation of coupled multiphase flow systems with strong phase interactions are among the applications with the highest demands on computational effort and system resources in the field of computational fluid dynamics. Massively parallel computers (MIMD) provide new capabilities for efficient and cost–effective multiphase flow calculations. However, special parallel solution algorithms have to be developed in order to use the computational power of MIMD computers. The main problem in the parallelization of Lagrangian solvers is the complex dependence between the fluid flow data and the data requirements of the solution algorithm for the particles/droplets equation of motion. The problem arises from the distributed storage of the fluid flow data over the processor nodes of the parallel computer system in accordance with the domain decomposition method. This data dependence has to be solved by an efficient parallel solution algorithm while introducing a minimum of inter-processor communication.

3. THE EULERIAN/LAGRANGIAN MODEL FOR DISPERSE MULTIPHASE FLOWS

3.1. Fundamental equations of fluid motion

For the construction of the physical model we assume, that the turbulent two–phase flow under consideration is dilute, but the particle loading is appreciable. So particle–particle interaction can be neglected, but the effects of the particles on the fluid flow has to be taken into account. The two–phase flow is steady, incompressible and isothermal. The fluid phase has constant physical properties and is Newtonian. Under these assumptions the time–averaged form of the governing fluid phase equations can be cast into the following form of the general transport equation :

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

where Φ stands for the different variables u_F, v_F, k and ε. The terms S_Φ and Γ represent the source term and the effective diffusion coefficient, respectively, and S_Φ^P represents the source term due to the momentum exchange between phases. This last term is calculated by solving the Lagrangian equation of particle motion using the PSI–cell–method [1, 2]. The source term expressions are summarized in Table 1 for different variables S_Φ.

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	$\frac{\mu_t}{\sigma_\varepsilon}$
$P_k = \mu_t \left\{2 \cdot \left[\left(\frac{\partial u_F}{\partial x}\right)^2 + \left(\frac{\partial v_F}{\partial y}\right)^2\right] + \left(\frac{\partial u_F}{\partial y} + \frac{\partial v_F}{\partial x}\right)^2\right\}$			
$S_{u_i}^P = -\frac{1}{V_{ij}}\sum m_P\,\dot{N}_P\,\big[u_{Pi,out} - u_{Pi,in} - g_i(1 - \frac{\rho_F}{\rho_P})(t_{out} - t_{in})\big]$			

Table 1: Expressions for source terms and effective diffusion coefficients

For modelling of fluid turbulence the standard k–ε turbulence model together with isotropic eddy viscosity and standard model constants are used. The influence of particle motion on fluid turbulence is neglected ($S_k^P = S_\varepsilon^P = 0$).

3.2. Equations of motion of the disperse phase

The disperse phase is treated by the Lagrangian approach where a large number of particles are followed in time along their trajectories through the flow domain. The particle trajectories are determined by solving the ordinary differential equations for the particle location, the translational and rotational velocities. For the formulation of the particles equations of motion only the drag force, the lift force due to particle rotation (Magnus force) and the gravitational force are taken into account. It is assumed that other forces like the Basset history force can be neglected due to a small density ratio ρ_F/ρ_P.

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

with :

$$Re_P = \frac{d_P\, v_{rel}}{\nu} \quad , \quad \sigma = \frac{1}{2}\frac{d_P \omega}{v_{rel}}$$

$$v_{rel} = \sqrt{(u_F - u_P)^2 + (v_F - v_P)^2}$$

The drag coefficient C_D, the lift coefficient of the Magnus force C_M and other model constants, e.g. restitution coefficient k and coefficient of kinetic friction f in the particle–wall interaction model, are taken from literature [3]. The effect of turbulence of the fluid flow on the motion of the disperse phase is modelled by the so–called Lagrangian stochastic–deterministic (LSD) turbulence model proposed by Schönung [12] and Milojević [6].

3.3. Solution algorithm

The above equations of fluid motion are solved by the FAN–2D program package developed by Perić and Lilek [11]. The code is designed for the prediction of two-dimensional (plane or axisymmetric), laminar or turbulent, incompressible flows of Newtonian fluid in domains of arbitrary geometry. The numerical solution method implemented is based on the finite volume discretization of the governing equations. Characteristics of this method are : non–orthogonal, boundary fitted arbitrary numerical grids; block structured numerical grids for optimum geometrical approximation of complex flow fields and for parallelization purposes; colocated arrangement of variables on numerical grids;

Cartesian vector and tensor components; segregated solution approach of SIMPLE kind [7]; acceleration of convergence by use of several levels of grid refinement [8].

The original program code is extended by introduction of the particle momentum source terms in the momentum equations of fluid motion. Efficiency of the solution method is ensured by implementing an optimized underrelaxation practice concerning not only the fluid variables but also the additional source terms.

The equations of motion of the dispersed phase are solved by using a standard Runge–Kutta solution scheme of 4th order accuracy as already used in previous work [3, 5]. A converged solution for both the fluid and disperse phase flow field is then obtained by an iterative solution procedure :

1. First a converged solution of the gas flow field is calculated without the source terms of the disperse phase.
2. A large number of particles is traced through the flow field, and the values of the source terms are calculated for all control volumes of the numerical grid.
3. The fluid flow field is recalculated by considering the source terms of the disperse phase, where appropriate underrelaxation factors have to be applied.
4. Steps 2 and 3 are repeated until convergence is reached.

4. THE PARALLEL SOLUTION ALGORITHM

4.1. General remarks on parallelization requirements

Normally calculations on parallel computers need more numerical operations to obtain a solution with a certain accuracy than calculations on a single processor or workstation. Further processor nodes in a parallel computer spend time on node communication and wait for delivery of data from neighbouring processors. So, efficiency of a parallel algorithm depends on :

- **Numerical efficiency :** describes the increase of the number of numerical operations due to changes in the algorithm necessary for parallelization;
- **Parallel efficiency :** describes the relative increase in calculation time due to communication between processor nodes;
- **Efficiency of load balancing :** describes the effect, that processors have to wait for each other due to unbalanced numerical and/or communicational workload distribution among the processors of the parallel computer. In the grid partitioning method unbalanced workload distribution is mainly caused by the different numbers of control volumes per grid block/processor node. In the case of parallel computation of disperse multiphase flows there can be other reasons for poor load balancing. In dependence on the used parallelization method these reasons can be flow separation, great changes in the mean particle concentration in the flow domain and the interaction of disperse particles with fluid turbulence.

Therefore the optimization of a parallel algorithm depends on the optimization of all efficiency factors and not only of the parallel efficiency.

4.2. Parallelization of the Navier–Stokes solver

The parallelization of the solution algorithm for the set of continuity, Navier–Stokes and turbulence model equations is carried out by parallelization in space, that means by application of the domain decomposition or grid partitioning method. This parallelization method was proposed e.g. by Perić [9] and Schreck [10] and ranks among the established and thorough investigated methods in the field of high performance computing. Grid partitioning methods were investigated in the past by many authors and so this parallelization algorithm was applied without significant changes.

The method is based on the further partitioning of the flow domain analogous to the block structuring of the numerical grid which is used for initial geometrical approximation of the flow domain (see Fig. 1). The resulting subdomains are assigned to the single processor nodes of the parallel computer. Because we consider MIMD computers with distributed storage of data, each processor node has to store not only the fluid flow data inside the grid block assigned to this node but also the values of fluid flow characteristics of the neighbouring grid blocks along the common boundaries of the grid subdomains. Each time the data along the grid block boundaries are altered by one processor they have to

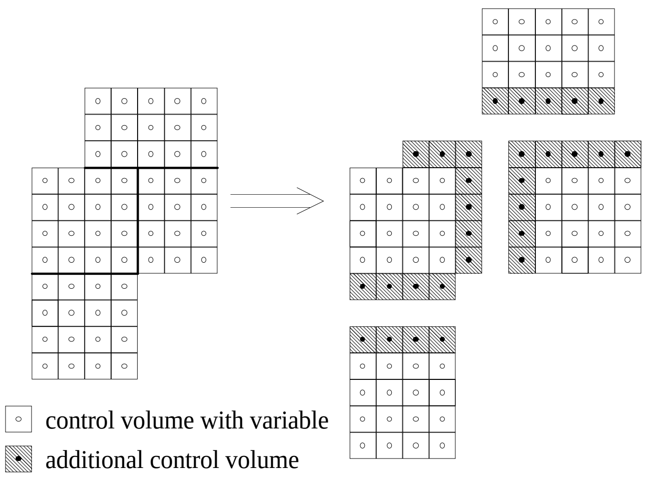


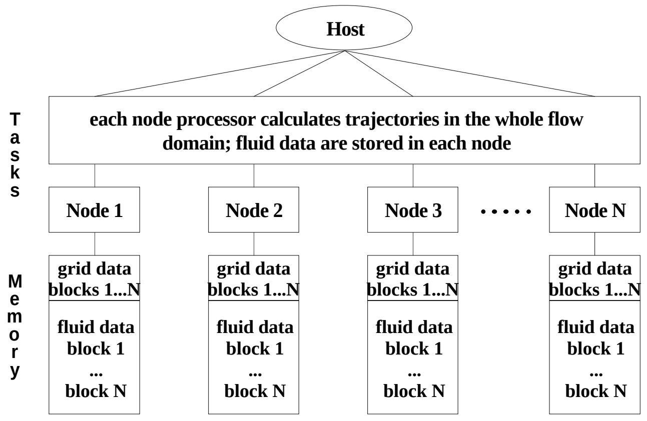
Figure 1: Domain decomposition for the numerical grid.

be exchanged between the adjacent processor nodes by inter–processor communication.

Using this parallelization method for the solution of the equations of motion of the fluid phase the processor nodes of the parallel computer can calculate the fluid flow field on their grid subdomains almost independently. Only after each inner iteration cycle of the iterative solution procedure for the linear set of equations the fluid flow data at the subdomain boundaries have to be exchanged. Further increase in parallel efficiency can be achieved by exchanging data at block boundaries only after each outer iteration cycle of the solution procedure. Although the use of “old” values at the block boundaries during one outer iteration cycle decreases numerical efficiency, a greater overall performance of the algorithm can be observed [9] for certain classes of fluid flow phenomena.

4.3. Parallelization methods for the Lagrangian approach

The main problem in parallelization of Lagrangian solvers is the complex dependence between the fluid flow data and the data requirements of the solution algorithm for the particles/droplets equation of motion. Because the location of a particle trajectory in the flow domain is prior unknown, a forecast about the fluid flow data requirements for particle trajectory calculation can not be made. Considering a MIMD computer with local node memory and with distributed storage of the fluid flow data in accordance with the domain decomposition method a parallel solution algorithm for Lagrangian simulation has either to provide all the fluid flow data in the local node memory of all processor nodes or the data which are necessary for particle trajectory calculations have to be delivered from other processor nodes at the moment when they are required. This results in a number of different parallelization methods.



Host
Tasks
each node processor calculates trajectories in the whole flow domain; fluid data are stored in each node
Node 1
Node 2
Node 3
Node N
Memory
grid data blocks 1...N
fluid data block 1 ... block N

Figure 2: Parallelization method 1 for the Lagrangian solver.

Method 1 :

In this method we introduce a host–node or divide–and–conquer parallelization scheme where the host generates the starting locations of the dispersed particles within the flow domain and distributes them to the nodes for trajectory and source term calculation. The nodes check the initial location of the particles on the numerical grid and calculate trajectories, the corresponding contributions to the source terms and to the mean values of particle phase characteristics (e.g. volume concentration, mean particle velocity and mean particle diameter). After particle trajectory calculations the host sums up the contributions to the source terms and mean values over all nodes and over all control volumes of the numerical grid. Then the values of the source terms are submited to the Navier–Stokes solver for recalculation of the modified fluid flow. In order to provide the necessary fluid flow data for the particle trajectory calculations the whole fluid flow field is stored in each processors node memory.

Load balancing for this method is automatically established due to the large number of calculated particle trajectories in comparison to the number of processor nodes. Although this method introduces a very small amount of node communication due to the distribution of initial values and collection of source terms and mean values of the disperse phase it has a major disadvantage which makes it applicable only to networked workstation clusters with large amount of memory or for multiphase flow calculations on rather small or coarse numerical grids. The redundant storage of fluid flow fields on each processor node leads to high demands on local node memory, which can not be satisfied on massively parallel MIMD machines.

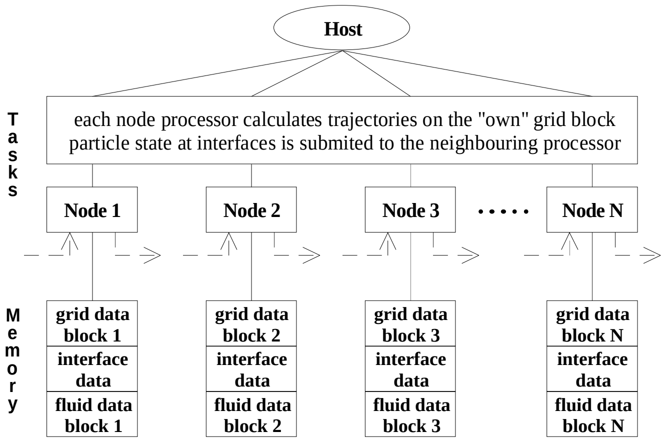


Figure 3: Parallelization method 2 for the Lagrangian solver.

Method 2 :

For the implementation of the Lagrangian solver on a massively parallel MIMD machine like the Parsytec Power–GC it is necessary to let the Lagrangian solver operate on a distributed set of fluid flow data because the node memory on such machines is rather limited and does not allow the storage of the whole fields of fluid flow data in each processor node. In the second method we use the same assignment of processor nodes to the blocks of the numerical grid as used by the grid partitioning method for the Navier–Stokes solver. In each node the fluid flow data of the corresponding grid block are stored. Now the node processors calculate particle trajectories from their entry point to the current grid block (from an inlet cross section or from a boundary to a neighbouring grid block) to their exit point (block boundary or outlet cross section). The amount of communication between nodes is again very small because it is reduced to the delivery of the particle state to the neighbouring processor in the case if a particle trajectory leaves the current block through a boundary which is a block interface with a neighbouring grid block. The calculation of global sums over all processor nodes is no longer necessary because the contributions to the source term fields are calculated and stored at the right location during the calculation process.

Load balancing can be a serious disadvantage of this method. That can be illustrated by a simple example of a pipe or channel flow where grid blocks are arranged one behind the other along the pipe or channel axis. In this case some start–up time is required until the calculation process propagates throughout the processors of the parallel computer. The same situation can be observed at the end of the calculation process where all processors have to wait until the last processor at the end of the pipe or channel has finished its calculation. Similiar situations of poor load balancing can occur for flows around nozzles, recirculating and highly separated flows where most of the numerical effort has to be performed by a small subset of all processors used.

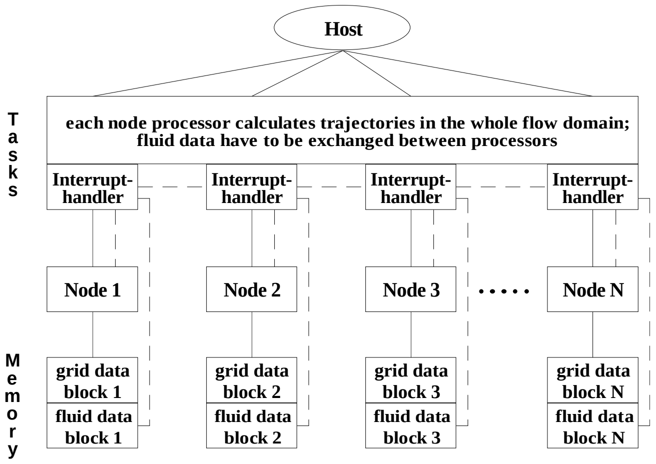


Figure 4: Parallelization method 3 for the Lagrangian solver.

Method 3 :

Since method 1 is not a fully parallelized method due to non–distributed storage of fluid flow data and since load balancing problems for method 2 for the named class of multiphase flow phenomena is expected, a third parallelization method is investigated. This parallelization method again uses the host–node programming model and the same distribution of fluid flow data among the nodes of the parallel machine as in the grid partitioning method of the Navier–Stokes solver. But in contrast to the second method a processor node calculates a particle trajectory from its entry point to the flow domain to its final exit location at an outlet cross section. While the particle is moving in the processors "own" grid block, fluid flow data needed for the particle trajectory calculation can be taken from the processors local node memory. If the trajectory leaves this grid block, fluid flow data have to be made available by node communication. This access method for the fluid flow data can be implemented in two different ways :

a) If the EXPRESS message passing library is used as the basis for parallelization a neat solution of the problem can be implemented by using a special feature of EXPRESS. EXPRESS message passing library offers a functionality of so–called message induced procedure calls (remote execution) or interrupt messages. A special procedure is registered by the EXHANDLE–function on each processor node. Then this procedure can be executed by any other processor node of the parallel computer simply by sending a message with a special message tag. Using this functionality of remote execution of the EXPRESS message passing standard it is possible to implement an efficient handling of distributed fluid flow data. Sending a message with a special message tag together with the control volume coordinates of the required fluid flow data to the processor with the appropriate grid block number starts a message induced procedure at that processor with the same adress space like the main routine on that processor. While this special procedure is executed the particle trajectory calculation process on the processor is interrupted. The special fluid flow data handling procedure now can read the required data from the local node memory and sends them back to the calling processor.

b) Unfortunatly other popular message passing libraries like PVM or MPI don't yet support comparable features of remote execution. In implementations of this parallelization method based on one of these message passing standards the functionality of remote execution has to be substituted. This can be realized by starting an additional memory handler task on each processor node. The memory handler provides fluid flow data of the grid block corresponding to the processor node number and waits in a permanent message loop for requests from other processor nodes. In this way requests for fluid flow data can be answered using normal inter–processor communication and without interruption of the particle trajectory calculation of the node program.

The algorithm described above can be enhanced by various caching and look–forward algorithms for the transfered fluid flow data. A similiar algorithm can be used for the calculation and distributed storage of the global source terms due to phase interaction and for the mean characteristics of the disperse phase.

The method requires a larger amount of node communication than the first two methods, but it was found to work with satisfactory efficiency. It operates with distributed fluid flow data and therefore needs the same amount of node memory as the grid partitioning algorithm of the Navier–Stokes solver (implementations based on PVM or MPI need twice the amount of node memory used by the Navier–Stokes solver). Further, the method has automatically a good load balancing due to the large number of calculated particle trajectories in comparison with the number of processors.

5. TEST PLATFORM AND FORMULATION OF TEST CASES

In order to compare the different parallelization methods for the Lagrangian approach calculations on a massively parallel computer were carried out for two typical test cases.

5.1. Test case 1

As a first test case we use a 2–dimensional, downward directed channel flow in a rectangular duct. The ratio of the channel height to the channel length is $H/L = 1/6$. For the fluid phase we use air with normal aerodynamical properties and for the disperse phase we assume solid particles with a density ratio of $\rho_F/\rho_P = 1/2000$ and a particle diameter of $d_P = 400\,\mu m$. At the inlet cross section the fluid velocity profile and the profile of particle volume concentration are homogeneous.

The flow domain is divided into 128×32 control volumes on the first grid level and 256×64 control volumes on the second grid level. For parallel computation the grid is divided into $1 \ldots 128$ grid blocks while the number of control volumes remains unchanged.

5.2. Test case 2

The aim of the second test case is to investigate the dependence of the efficiency of the various parallelization methods on the multiphase flow regime. As an example of a highly separated two–phase flow we consider an axisymmetric, upward directed pipe flow around a full–cone nozzle. The ratio of the pipe radius to the pipe length is again $R/L = 1/6$. The nozzle is located at the pipe (symmetry) axis at $x = \frac{1}{3}L$ and the jet from the full–cone nozzle with a cone angle of $90°$ is directed downward. For the fluid phase we use air with a homogeneous velocity profile at the lower inlet cross section with $u_F = 4\,m/s$. The disperse phase is represented by water droplets with an initial velocity of $8\,m/s$ and with a given droplet diameter distribution in the range from $30\,\mu m \ldots 1400\,\mu m$. The numbers of control volumes, grid levels and grid blocks of the numerical grid are the same as in test case 1.

5.3. Test platform

Most of the implementation effort and first performance evaluation was carried out on a networked workstation cluster of 3 HP 9000/735 linked by a 100 Mbit/s FDDI network. Each workstation in the cluster has a minimum node memory of about 80 Megabyte. The workstations ran under HP–UX 9.05 with EXPRESS Vers. 3.2.5 and PVM Vers. 3.2.6 message passing standards.

For the final performance evaluations for the two test cases we used the massively parallel MIMD computer Parsytec Power–GC–128 at the Technical University Chemnitz. The basic characteristics of this parallel machine are :

- Node processors : Motorola PowerPC 601–80 ; Maximum number of processors : 128 ; Node memory : 32 MB
- Communication : 35 MB/s sustained; 80 MB/s peak ; Message setup time : $5\,\mu s$; Minimum network latency : $40\,\mu s$

The machine runs under the operating system Parix 1.2–PPC (a UNIX derivative for this kind of parallel computers) with Power–PVM Vers. 1.1 (a subset of PVM Vers. 3.2 which excludes support for heterogeneous computer platforms) and Power–MPI Vers. 1.0. Unfortunatly Power–EXPRESS was not yet available for the Parsytec Power–GC–128 due to a bug in the message passing library.

6. RESULTS AND DISCUSSION

Fig. 5 — Fig. 10 show the results of the performance evaluation tests on the Parsytec Power–GC–128 for the two test cases and for all 3 investigated parallelization methods. For the performance evaluation tests execution time of one iteration cycle of the Lagrangian particle trajectory solver was measured. The start–up time of the PVM–system, the execution time of the Navier–Stokes solver and the time spent on I/O–operations for postprocessing purposes were not included in the time measurements. From the measured execution time (cpu–time in seconds) the speed–up S_N and the efficiency E_N :

$$S_N = \frac{\text{Execution time on 1 proc.}}{\text{Execution time on N proc.}} \quad ; \quad E_N = \frac{S_N}{N}$$

for the different parallelization methods were calculated. It has to be pointed out that for the given test cases the numerical effort (number of numerical operations) of the Lagrangian solver doesn't remain constant with increase of the number of grid blocks on the numerical grid. The implemented method for particle localization on the numerical grid leads to an increase in program efficiency proportional to the number of grid blocks independent of the number of processors used for program execution. This additional increase in program efficiency is the explanation for the high speed–up and efficiency values that occur especially for the second test case (Fig. 9 — Fig. 10). Nevertheless the measured performance results can serve for a direct comparison of the 3 investigated parallelization methods.

Fig. 5 — Fig. 7 show the results of the performance evaluation tests for test case 1 and for parallelization methods 1–3. In order to show the influence of the numerical workload on the efficiency of the various parallelization schemes calculations were carried out for two different numbers of particle trajectories (5000 and 20000). But the comparison of these 2 series of calculations showed only minor differences for all 3 parallelization methods. Therefore only the results for calculations of 20000 particle trajectories are shown in this paper.

Best results were obtained by the parallelization method 1 with a speed–up of 47.1 on 64 processor nodes. But as already mentioned above the method 1 is basically the serial Lagrangian approach adopted to the parallel machine and does not support the distributed storage concept for the fluid flow data. Thus method 1 is not applicable to numerical grids with a finer grid resolution or to multiphase flows in geometrically complex flow domains due to limited node memory on MIMD computers. Method 2 and 3 show similar performance results with a maximum speed–up of about 18 and an efficiency of about 0.3 for calculations on 64 processor nodes.

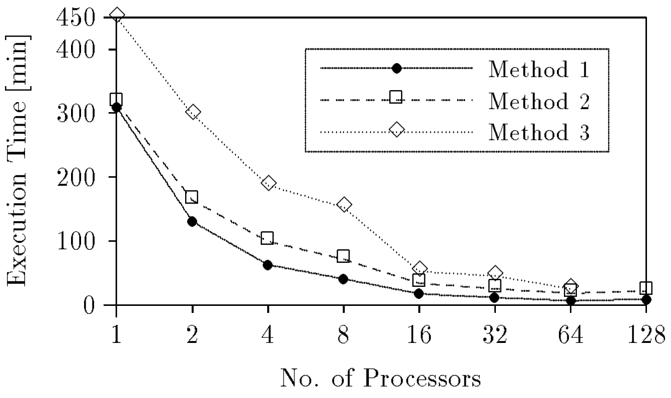


Figure 5: Execution time for parallelization method 1–3 for the first test case.

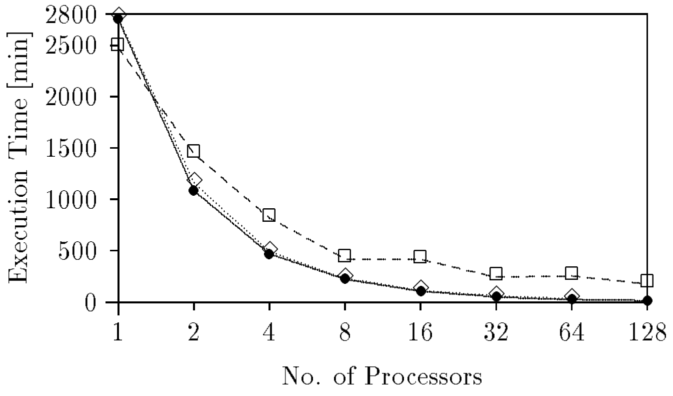


Figure 8: Execution time for parallelization method 1–3 for the second test case.

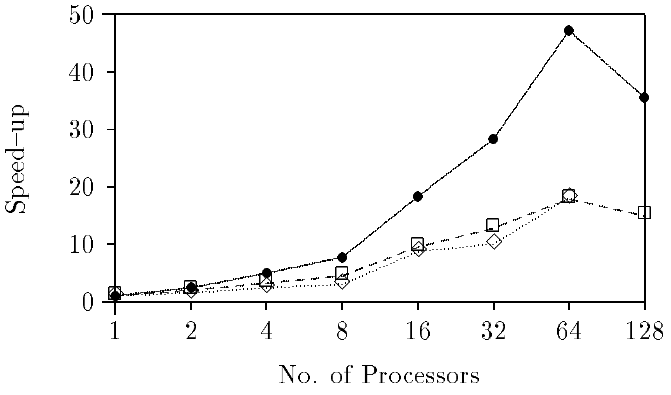


Figure 6: Speed–up for parallelization method 1–3 for the first test case.

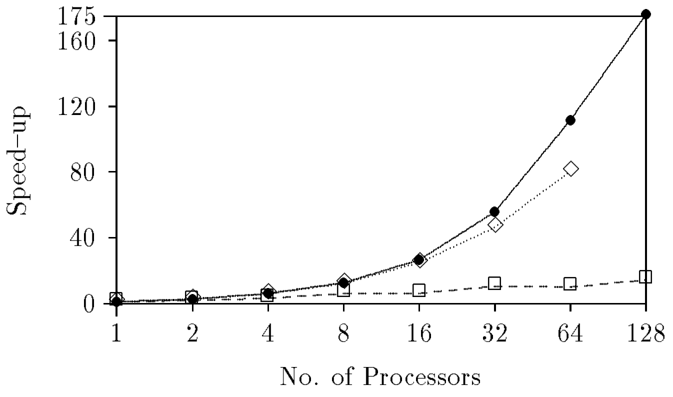


Figure 9: Speed–up for parallelization method 1–3 for the second test case.

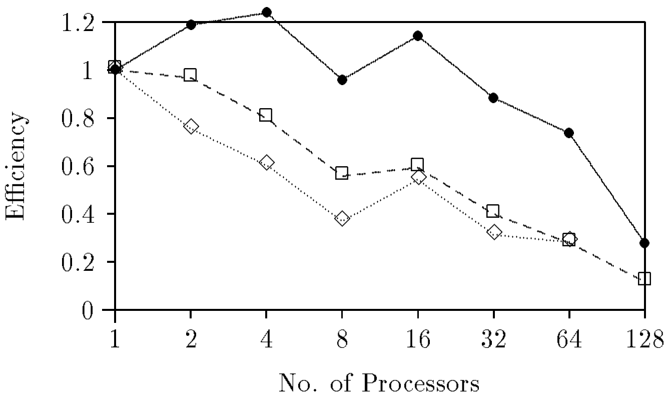


Figure 7: Efficiency for parallelization method 1–3 for the first test case.

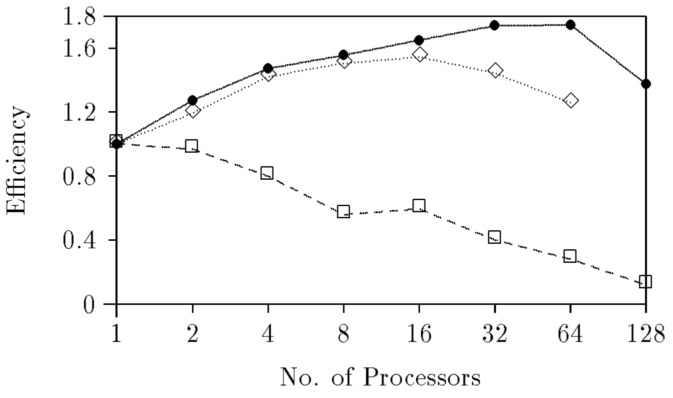


Figure 10: Efficiency for parallelization method 1–3 for the second test case.

The strong decrease in performance of all the methods 1–3 on 128 processor nodes does not result from construction or implementation of the parallelization methods but is mostly due to a known bottleneck for node communication between the upper and the lower 64 processor partition of the Parsytec Power–GC–128. Less performance for calculations on 8 processor nodes in comparison with calculations on 16 processor nodes was found to be due to the arrangement of the grid blocks of the numerical grid. For calculations on 16 processor nodes the grid blocks are arranged in 2 rows per 8 grid blocks in the x–direction. This leads to a distribution of the inlet cross section (x=0) to 2 different processor nodes and therefore to an increase in program performance.

Fig. 8 — Fig. 10 show the results of the performance evaluation tests for the second test case. Again the best results could be achieved for method 1 (the serial Lagrangian approach adopted to the parallel machine). As already mentioned above the large speed–up values and the efficiency values greater 1 result from a reduction of the numerical effort due to higher efficiency of the particle localization algorithm on block structured grids with a larger number of grid blocks.

Further the results for the method 2 show a substantial deterioration in comparison to the other methods and to the first test case due to worser load balancing for separated multiphase flows.In this method the computational power of a processor is assigned to a fixed grid block of the numerical grid. So the execution time of the program is mostly determined by the computational time spent by the processor which is assigned to the grid block with the highest particle concentrations inside the flow domain.

The results for method 3 show similar performance of the parallelization method in comparison with method 1. Only for a larger number of processors (32 and more) results show a decrease in efficiency (Fig. 10) due to the larger amount of inter–processor communication necessary in the third parallelization method. Further the results of the second test case for method 3 show, that the method is applicable independent of the flow regime of the investigated disperse multiphase flow even in the case of flow separation or greater changes in particle concentration in the flow domain.

7. CONCLUSIONS

The test cases for parallelization methods 1–3 show performance results which are typical for CFD applications on massively parallel MIMD computers. Although the parallel efficiency decreases for calculations on a larger number of processor nodes, the results show the applicability of the presented methods for Lagrangian multiphase flow calculations on parallel computers with distributed memory and a moderate number of high–performance processors. In this case remarkable speed–up can be achieved which makes calculation of complex multiphase flows possible in reasonable time. Especially the results for the third parallelization method show the universal applicability of the method for parallel computations of disperse multiphase flows. Because parallelization of the Lagrangian approach is carried out by parallelization in space using the domain decomposition method the described algorithm is applicable to steady and unsteady flow calculations as well.

8. Acknoledgement

The authors are indebted to Prof. M. Perić for allowing the use of his CFD code FAN–2D in this research. Further this work was supported by the Deutsche Forschungsgemeinschaft (DFG) under Contract No. Fr 1069/3–1.

References

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

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

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

[4] Frank Th., Schulze I., 1994, “Ein numerisches Verfahren zur Berechnung disperser Mehrphasenströmungen auf parallelen Hochleistungsrechnern”, *Proc. Arbeitssitzung des GVC–Fachausschusses Mehrphasenströmungen*, February 17–18, 1994, Würzburg, Germany.

[5] Frank Th., Schulze I., 1994, “Numerical simulation of gas–droplet flow around a nozzle in a cylindrical

chamber using Lagrangian model based on a multigrid Navier–Stokes solver", International Symposium on Numerical Methods for Multiphase Flows, June 19–23, 1994, Lake Tahoe (NV), USA.

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

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

[8] Perić M., 1989, "A Finite Volume Multigrid Method for Calculating Turbulent Flows," *Proc. 7th Symposium on Turbulent Shear Flows,* Vol. 1, pp. 7.3.1.–7.3.6., Stanford University, USA.

[9] Perić M., 1992, "Ein zum Parallelrechnen geeignetes Finite–Volumen–Mehrgitterverfahren zur Berechnung komplexer Strömungen auf blockstrukturierten Gittern mit lokaler Verfeinerung", Abschlußbericht zum DFG–Vorhaben Pe 350/3–1 im DFG–Habilitandenstipendiumprogramm, Stanford University, USA.

[10] Schreck E., Perić M., 1992, "Parallelization of implicit solution methods", *ASME Fluids Engineering Conference*, June 22–23, 1992, Los Angeles (CA), USA.

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

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