JSME Centennial Grand Congress 1997
ISAC '97 Conference on Advanced Computing on Multiphase Flow
July 18–19, 1997, Tokyo, Japan

Parallel Efficiency of PVM and MPI Implementations of Two Algorithms for the Lagrangian Prediction 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
09107 Chemnitz, Germany
Phone: +49 371 531 4643
Fax: +49 371 531 4644
E–Mail: frank@imech.tu–chemnitz.de

ABSTRACT

This paper deals with two different methods for the parallelization of Lagrangian (PSI–cell) approach which is widely used for the prediction of disperse multiphase flows (e.g. gas–particle or gas–droplet flows). In both presented methods the parallelization of the solution algorithm for the fluids equations of motion is carried out by application of a domain decomposition method to the block structured grid. For the Lagrangian solution algorithm for the equations of motion of the disperse phase two different parallelization methods are investigated and compared with each other. Results of performance evaluation are provided for a typical test case for PVM and MPI implementations of the algorithms and for the two different MIMD computer architectures Parsytec GC–128 and Cray T3D as well.

NOMENCLATURE

C_D, C_M	drag and lift coefficients
Re	Reynolds number
S_Φ	source term
S_Φ^P	source term due to particle–fluid interaction
V	volume
g	gravitational acceleration
k	turbulence kinetic energy
m	particle mass
t	time
u, v	velocity in x– and y–direction
v_{rel}	absolute value of particle–fluid relative velocity
Γ	general diffusion coefficient
Φ	general variable in transport equation
ε	dissipation of turbulent kinetic energy
ν	kinematic viscosity
ρ	density
ω	particle rotational velocity
Subscripts	
F	fluid phase
P	particle phase

1. THE EULERIAN/LAGRANGIAN MODEL FOR DISPERSE MULTIPHASE FLOWS

All parallelization methods described in this paper are based on the Eulerian/Lagrangian approach (PSI–cell) developed by C.T. Crowe [1, 2]. The model has been described in more detail in earlier publications of the authors [3, 4, 5]. For the flow calculation of the continuous phase a modified finite volume Navier–Stokes solver developed by M. Perić and Ž. Lilek [9] is used. For the case of a steady, incompressible and isothermal two–phase flow 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].

$$S_{u_i}^P = -\frac{1}{V_{ij}} \sum m_P \dot{N}_P \times$$
$$\times \left[u_{Pi,out} - u_{Pi,in} - g_i(1 - \frac{\rho_F}{\rho_P})(t_{out} - t_{in}) \right] \quad (2)$$

The Navier–Stokes solver is operating on structured, non–orthogonal, curvilinear grids with a bisectional refinement strategy for the construction of the various grid levels.

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.

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

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 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 and Milojević [6]. The standard iteration procedure is applied to the coupled system of equations of fluid and particle motion as described in [5]. The iterative solution procedure is continued until convergence for the fluid and particle flow field is achieved.

2. THE PARALLELIZATION METHODS

2.1. 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. Using the block structure of the numerical grid the flow domain is partitioned in a number of subdomains according to the number of computing nodes of the parallel machine (Fig. 1). The resulting subdomains with their geometrical and fluid flow data are assigned to the individual processor nodes for calculation. Fluid flow characteristics along the grid block boundaries which are common to two different nodes have to be exchanged during the solution process by inter–processor communication. This parallelization method was proposed e.g. by Perić [7] and Schreck [8] 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.

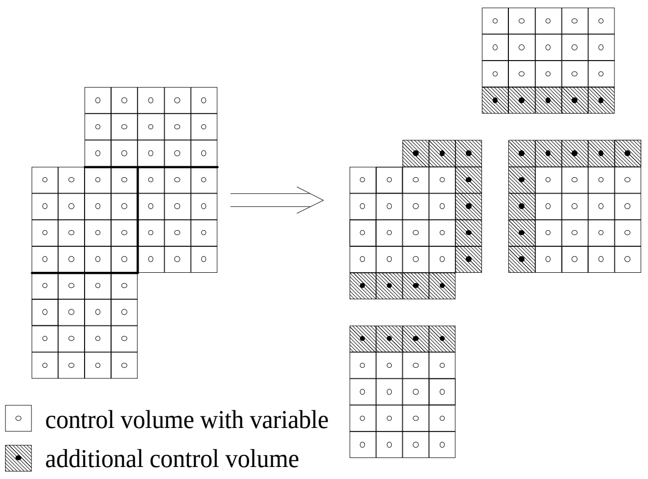

Fig. 1: Domain decomposition for the numerical grid.

Domain Decomposition Approach For The Lagrangian Solver

Parallel solution algorithms for the particle equations of motion have to deal with the global data dependence between the distributed storage of fluid flow data and the local data requirements for particle trajectory calculation. A parallel Lagrangian solution algorithm has either to provide all fluid flow data necessary for the calculation of a certain particle trajectory segment in the local memory of the processor node or the fluid flow data have to be delivered from other processor nodes at the moment when they are required.

One approach in parallelization of Lagrangian particle trajectory calculations is the application of the same parallelization method as for the fluid flow calculation to the Lagrangian solver as well, that means domain decomposition. In this approach geometry and fluid flow data are distributed over the processor nodes of the parallel machine in accordance to the block structure of the numerical grid. The assignment between processor nodes and grid blocks is the same as used for the grid partitioning method for the Navier–Stokes solver.

Now we introduce a host–node parallelization scheme, where the host processor generates the starting locations and initial conditions of the dispersed particles within the flow domain. In a first stage of the calculation this particle initial conditions (p.i.c.) are passed to the processor nodes $1, \ldots, N$ for determination of the grid block number where particle trajectory calculation has to be initiated. After that the host processor distributes the p.i.c. to the node processor with the corresponding grid block number for processing. Nodes $1, \ldots, N$ calculate the particle trajectory segments from the entry point to their "own" grid block (from an inlet cross section or from a boundary to a neighbouring grid block) to their exit location (block boundary or outlet cross section). After a particle trajectory has reached such an exit location on a certain node/grid block the particle state at the block boundary is returned to the host process. There it is treated as a new p.i.c. for the neighbouring block/processor node until all particle trajectories have satisfied certain break condition (e.g. an outlet cross section is reached). During the particle trajectory calculation process the source terms for momentum exchange between the two phases are calculated locally on the processor nodes $1, \ldots, N$ from where they can be passed to the Navier–Stokes solver without further processing (Fig. 2).

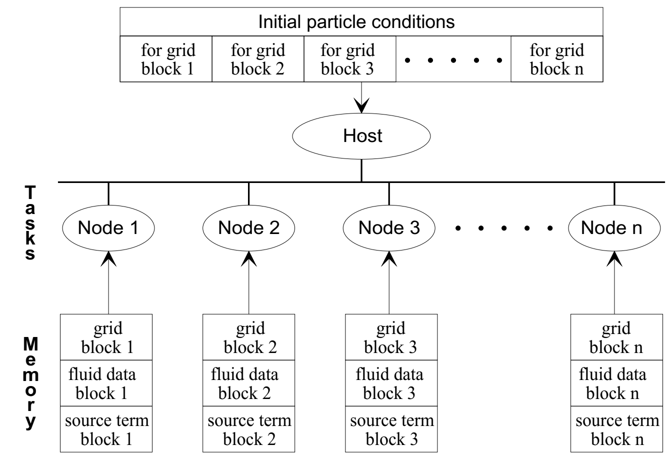

Fig. 2: Domain decomposition approach for the Lagrangian solver.

An advantage of the domain decomposition approach is that it is easy to implement and uses the same data distribution over the processor nodes as the Navier–Stokes solver. But load balancing can be a serious disadvantage of this method as shown later for the presented test case. Poor load balancing can be caused by different circumstances, as there are :

1. Unequal processing power of the calculating nodes, e.g. in a heterogenous workstation cluster.
2. Unequal size of the grid blocks of the numerical grid. This results in different numbers of control volumes/grid cells per processor node and in unequal work load for the processors.

   These reasons for poor load balancing are common to all domain decomposition approaches and apply to the parallelization method for the Navier–Stokes solver as well.
3. Differences in particle concentration distribution throughout the flow domain. Situations of poor load balancing can occur e.g. for flows around free jets/nozzles, in recirculating or highly separated flows where most of the numerical effort has to be performed by a small subset of all processor nodes used.
4. Multiple particle–wall collisions. Highly frequent particle–wall collisions occur especially on curved walls where the particles are brought in contact with the wall by the fluid flow multiple times. This results in a higher work load for the corresponding processor node due to the reduction of the integration time step and the extra effort for detection/calculation of the particle–wall collision itself.
5. Flow regions of high fluid velocity gradients/small fluid turbulence time scale. This leads to a reduction of the integration time step for the Lagrangian approach in order to preserve accuracy of the calculation and therefore to a higher work load for the corresponding processor node.

Most of these factors leading to poor load balancing in the domain decomposition approach cannot be foreseen without prior knowledge about the flow regime inside the flow domain (e.g. from experimental investigations). Therefore an adjustment of the numerical grid to meet the load balancing requirements by redistribution of grid cells is almost impossible. The second parallelization method shows how to overcome these limitations.

Distributed Shared Memory Approach For The Lagrangian Solver

This parallelization method also uses the host–node programming model and the same distribution of fluid flow data among the processor nodes of the parallel machine as in the previous method. Also the task of the host processor node is basically unchanged. But in contrast to the domain decomposition approach the assignment of a processor node to a certain grid block is not static over the period of the calculation process.

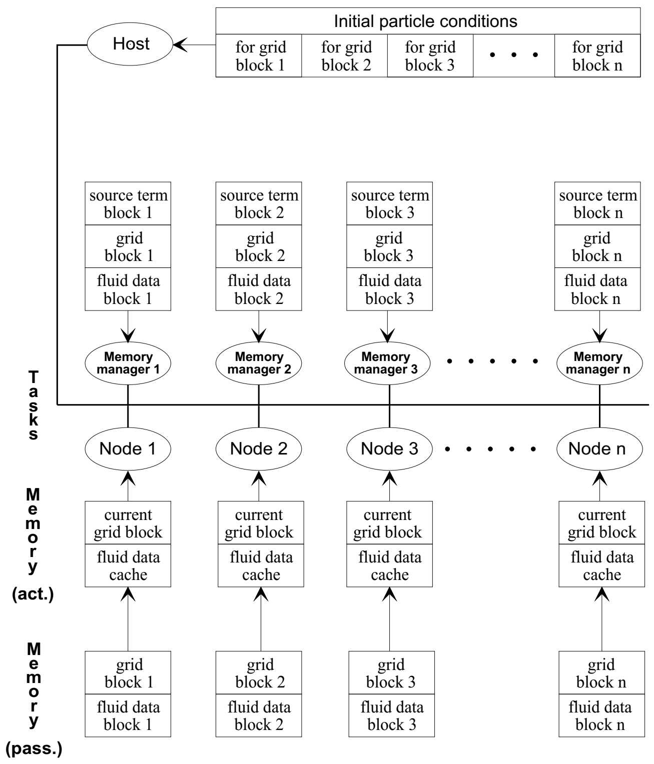

Fig. 3: Distributed shared memory approach for the Lagrangian solver.

If a request of a node processor for a new p.i.c. for the currently processed grid block cannot be satisfied by the host processor, then an p.i.c. for another grid block is submitted to the node. The calculating node is now able to obtain necessary geometry and fluid flow data for the calculation of the trajectory segment on the grid block corresponding to the assigned p.i.c. In the current implementation this is achieved by introduction of N additional tasks $(N+1), \ldots, 2N$, called 'memory manager' nodes. The memory manager tasks provide memory for the fluid flow data, source terms and mean particle flow characteristics of the grid block corresponding to their processor node number minus N and remains in a permanent message loop waiting for requests from other calculating nodes (Fig. 3).

It has to be mentioned that the memory manager tasks do not have to be implemented as independent node processes running on a separate physical node, since their work load is quite neglectable. In EXPRESS they can be implemented as message induced procedure calls (remote execution by active messages) on the calculating nodes using their memory. As soon as threads become available in MPI–2 they can also be implemented as separate threads running on the same physical processor node as the corresponding calculating task. In current PVM–3 and MPI–1 implementations the memory manager tasks are implemented as separate node processes. Although on parallel machines allowing multiple processes per physical processor node they can be executed parallel to the corresponding calculating process on the same physical processor node.

The delivery of fluid flow data to the calculating nodes can be established in two different ways :

1. If a particle trajectory crosses a certain control volume/grid cell, the fluid flow data corresponding to this grid cell and to the nearest neighbouring grid cells in each coordinate direction are required for further calculation. Now the fluid flow data for these 5 control volumes can be delivered at a time (further referred to as DSM–Point method). Even this method introduces more frequent communications with lower volume of transfered data it can be advantageous in dependence on the computer hardware and the size of the grid blocks of the numerical mesh because only a 1–dimensional subset of the fluid flow data has to be transfered for the calculation of the particle trajectory.

2. Otherwise the full information about the fluid flow data corresponding to the grid block of the assigned p.i.c. can be transfered on the first request from the calculating node (further referred to as DSM–Block method). This method is more advantageous for parallel computer architectures with fast communication network and high bandwith of communication.

In addition to the previous method the host processor is used for optimization of the assignment of p.i.c. to the calculating processor nodes. In the case of a request for a new p.i.c. the following order of precedence for p.i.c. selection is used :

1. The number of unprocessed p.i.c. on the grid block corresponding to the node number of the requesting processor node (its "own" grid block) exceeds a threshold value. Then a p.i.c. for this grid block is submitted.

2. There are unprocessed p.i.c. for the grid block the requesting node is currently working on (may be not the "own" grid block).

3. There are unprocessed p.i.c. for the nodes "own" grid block.

4. There are unprocessed p.i.c. for other grid blocks. Then p.i.c. for the grid block with the maximum work load is submited to the requesting node processor.

As shown below one of the greatest advantages of the distributed shared memory approach for the Lagrangian solver is the automatically established load balancing which is independent of all contributing factors discussed in 3.2. Disadvantageous are the higher memory requirements and a slightly higher inter–processor communication.

3. PVM/MPI IMPLEMENTATIONS AND THE MIMD COMPUTER ARCHITECTURES

The parallelization methods are implemented using the PVM and MPI implementations on the Parsytec GC–128 and the Cray T3D. The PVM implementations are a subset of PVM 3.2 while the MPI implementations are in compliance with MPI 1.1 standard. Both the PVM and the MPI implementation on this MIMD machines do not support parallel computing on heterogenous computer platforms. But this is not a limitation for the resulting implementation for the Eulerian/Lagrangian solver which can be applied e.g. to heterogenous workstation clusters as well.

The code was developed and tested on two different massively parallel MIMD architectures. The Parsytec GC–128 of the Technical University of Chemnitz is based on 80 MHz Motorola PowerPC 601–80 with a maximum of 128 processors, 32 MB memory each. Processors are arranged in a 2–dimensional communication network (grid) delivering a communication bandwith of 35 MB/s sustained and 80 MB/s peak, a setup time of 5 μs and a minimum network latency of 40 μs.

The Cray T3D at the Edinburgh Parallel Computing Centre (EPCC) consists of 512 150 MHz 21064 Alpha processor nodes, each with 64 MB of local memory. Nodes are arranged in a 3–dimensional torus, with each of the six links from each node simultaneously supporting hardware transfer rates of up to 300 MB/s.

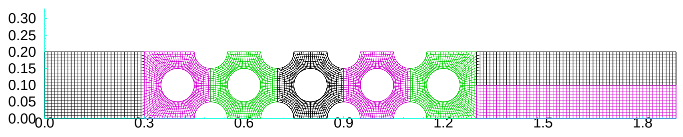

Fig. 4: Numerical mesh for staggered tube rows with 8 different grid blocks.

4. TEST CASE, RESULTS AND DISCUSSION

4.1. Formulation Of The Test Case

For the test case a gas–particle flow through an arrangement of staggered tube rows has been choosen as it is used e.g. for heat exchangers. Fig. 4 shows the numerical mesh for the geometry of the flow domain for a subdivision into 8 grid blocks. The flow enters the flow geometry from the left with $u_F = 10\ m/s$ and has the outlet cross section on the right. Symmetry boundary conditions are applied to the upper and lower boundary of the flow geometry. The gas phase (air under normal conditions) carries a disperse phase of particles with $u_P = 9 \ldots 11\ m/s$, $d_P = 20 \ldots 500\ \mu m$ and $\rho_P = 2500\ kg/m^3$ which are uniformly distributed over the inlet cross section.

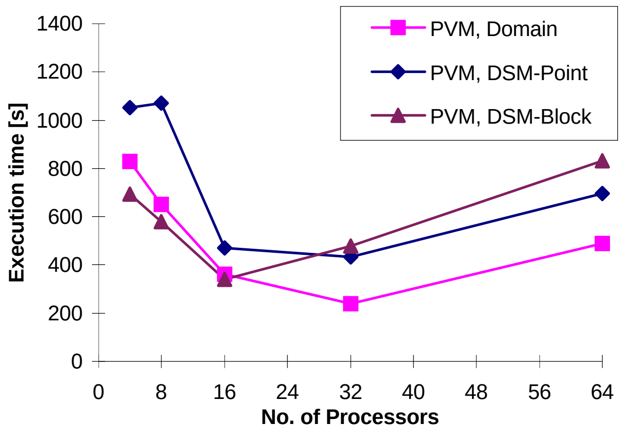

Fig. 5: Execution time for PVM implementation on Parsytec GC–128.

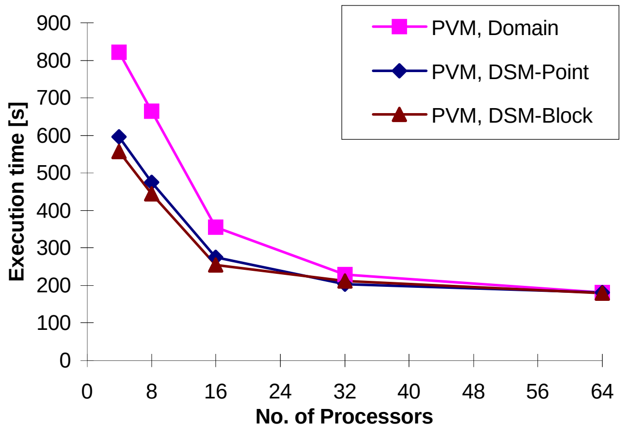

Fig. 6: Execution time for PVM implementation on Cray T3D.

For the test case only one iteration cycle of the solution procedure has been executed. For the Navier–Stokes solver the number of iterations on each grid level was restricted to 3000 and the number of particle trajectories calculated by the Lagrangian solver was 20000. Due to CPU time restrictions on the Cray T3D the number of particle trajectories for these test case runs has to be decreased to 1000. For comparison the measured execution times for the Parsytec GC–128 are divided by the appropriate factor. Test case calculations have been performed for different subdivisions of the flow geometry into 4, 8, 16, 32 and 64 grid blocks (each with 2 grid levels of refinement) on the corresponding number of processor nodes. But it has to be mentioned that the numerical mesh with 4 grid blocks contains only half the number of control volumes of the other numerical meshes. This was caused by restrictions in mesh generation for the given geometry. But the results show, that

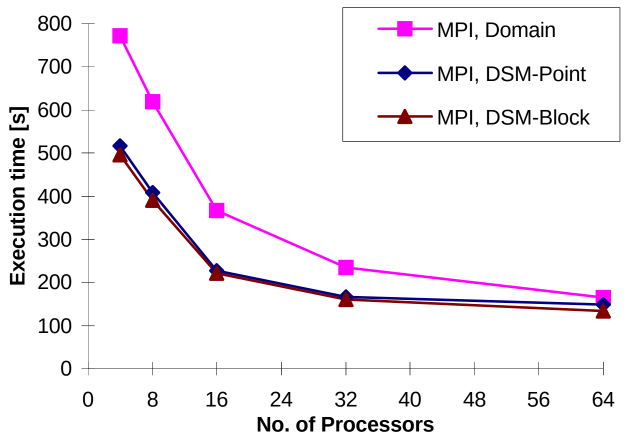

Fig. 7: Execution time for MPI implementation on Cray T3D.

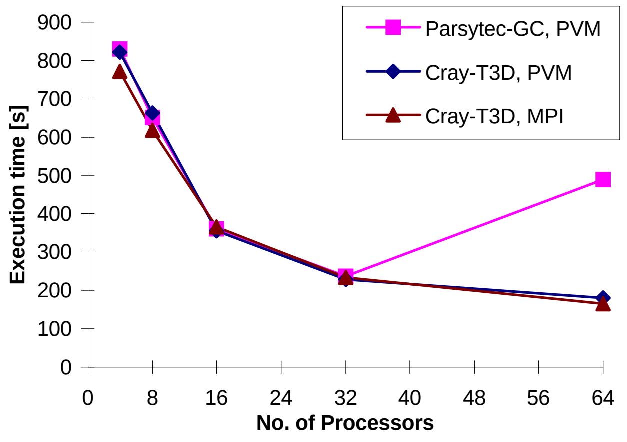

Fig. 8: Comparison of domain decomposition approach for Parsytec GC–128 and Cray T3D.

the calculations on this slightly changed configuration for 4 grid blocks are unfortunately not fully comparable with the other test case results.

4.2. Results Of Performance Evaluations

For the test case calculations the total execution time, calculation time, communication time and I/O time have been measured for the execution of one iteration cycle of the Lagrangian solver. From these measurements the difference time (= total exec. time - calc. time - comm. time - I/O time) has been calculated. This difference time contains mainly the waiting time for the processor nodes in synchronous receive operations and global barriers (what can also be established from Fig. 11 and Fig. 12).

Now Fig. 5 – Fig. 7 show the results for the total execution time for the 3 different parallelization methods and for both the PVM and MPI implementations on the Parsytec GC–128 and Cray T3D. While execution time for all 3 different methods increases on 64 processor nodes on the Parsytec GC–128 in comparison with the results on 32 nodes, the execution times on the Cray T3D could be further decreased even for 64 processor nodes. This behavior is mainly caused by the slower communication network and a hardware bottleneck for calculations on more than 64 processor nodes on the Parsytec machine.

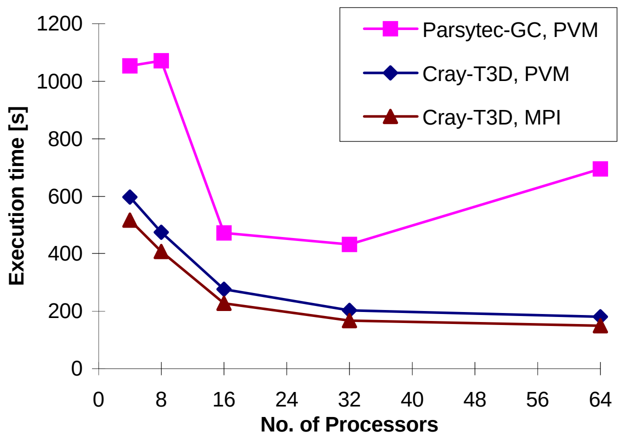

Fig. 9: Comparison of DSM–Point method for Parsytec GC–128 and Cray T3D.

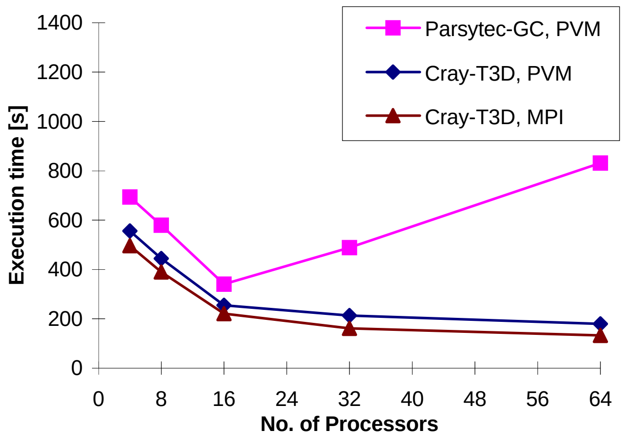

Fig. 10: Comparison of DSM–Block method for Parsytec GC–128 and Cray T3D.

Fig. 6 for the PVM implementation shows the advantage of the distributed shared memory approach for the Lagrangian solver with a slight gain in performance for the DSM–Block method. But they also show, that execution times for the calculations on 64 nodes were almost determined by communication and I/O time (which is about 2/3 of the total execution time). Because calculation times show further decrease by a factor of 0.63, better scaling results can be expected for larger tasks (larger grid blocks, greater number of control volumes and/or more particle trajectories to calculate).

Fig. 7 shows the improvements introduced in the MPI implementation of the 3 parallelization methods. Total execution time of all 3 methods could be further decreased which is most remarkable for the distributed shared memory approach (see also Fig. 9 and Fig. 10). Since the better performance was achieved by reduction in communication time this results also in a better scalability over the investigated range of processor node numbers used in the calculations.

4.3. Load Balancing

The results shown in Fig. 8 are quite unexpected because nearly equal performance of the Parsytec GC–128 and the Cray T3D is observed for the domain decomposition approach over the range from 4 to 32 processor nodes. An explanation of this behavior can be found in Fig. 11 and Fig. 12. These figures show the different contributions to the total execution time for test case runs on 8 processor nodes on the Cray T3D (MPI implementation). It can be found from the figures, that 1) difference time consists mainly of waiting time for the I/O performed by other processor nodes and 2) I/O time takes about 9–14% of the total execution time (this part of the total execution time remains constant for all numbers of nodes). But the most important thing to point out from these figures are the great differences in load balancing for the calculating processor nodes for the domain decomposition and the distributed shared memory approach. The total execution time of the domain decomposition approach (Fig. 11) is mainly determined by the calculation time spent on

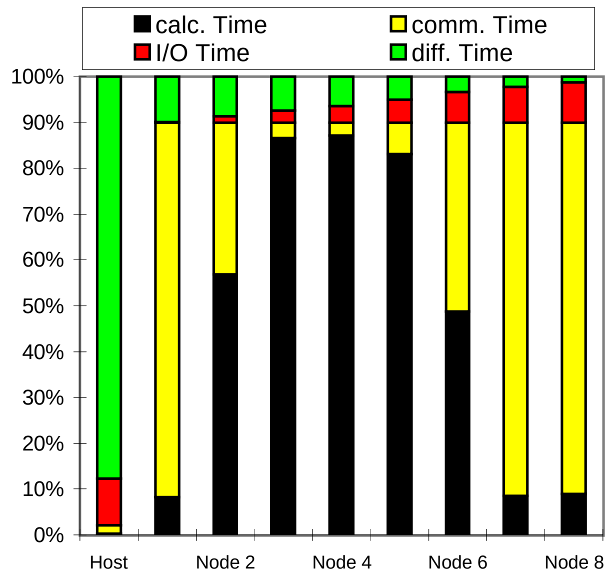

Fig. 11: Work load distribution for domain decomposition approach.

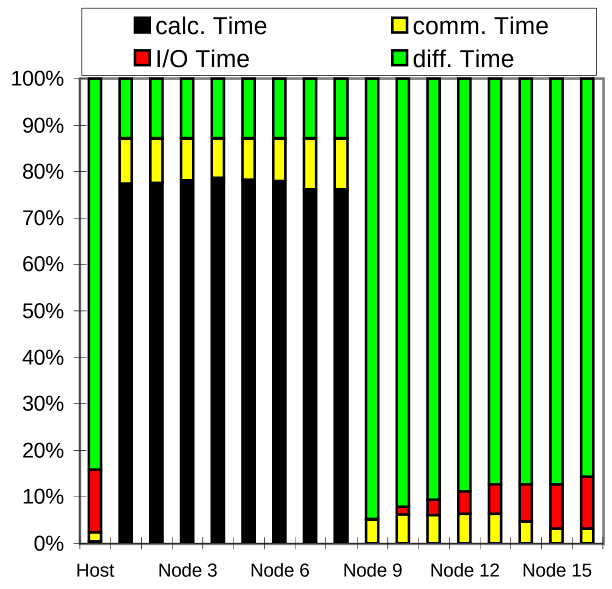

Fig. 12: Work load distribution for distributed shared memory approach (DSM–Block).

the inner grid blocks surrounding the tubes, which is effected by multiple particle–wall collisions and strong fluid velocity gradients (integration time step reduction). These influences on calculation time cannot be observed on the rectangular grid blocks near the inlet and outlet cross sections. This causes greater differences in the work load of the processor nodes and a poor load balancing. Consequently this poor load balancing is the limiting factor for the domain decomposition approach and leads to the unexpected results in Fig. 8. On the contrary, Fig. 12 shows a homogenous work load distribution over the calculating processor nodes for the distributed shared memory approach. Even if the amount of communication is higher in this method, better performance results can be obtained on MIMD systems with a high bandwith of the communication hardware as e.g. for the Cray T3D.

5. CONCLUSIONS

The paper presents two parallelization methods for Eulerian/Lagrangian calculations of disperse multiphase flows together with their PVM and MPI implementations. Performance results are given for a typical test case and for two different massively parallel MIMD architectures. The obtained results show besides the general applicability of the parallelization methods for parallel computers with distributed memory and message passing paradigm the importance of homogenous work load distribution which has to be treated differently in comparison with domain decomposition methods for single phase flow calculations. With the presented distributed shared memory approach remarkable speed–up can be achieved for parallel calculations on a moderate number of processor nodes which makes calculations of complex multiphase flows possible in reasonable time.

6. 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. SFB 393/D2 and No. Fr 1069/3–1. Work on the Cray T3D at the Edinburgh Parallel Computing Centre (EPCC) was carried out as a part of the TRACS/TMR program founded by the European Community.

7. REFERENCES

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

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

[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., Schulze I. "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*, ASME Fluids Engineering Division Summer Meeting, Lake Tahoe (NV), USA, June 19–23, (1994), FED–Vol. 185, pp.93–107.

[5] Frank Th., Wassen E. "Parallel Solution Algorithms for Lagrangian Simulation of Disperse Multiphase Flows", *Proc. 2nd Int. Symposium on Numerical Methods for Multiphase Flows*, ASME Fluids Engineering Division Summer Meeting, San Diego, CA, USA, July 7–11, (1996), Vol. 1 (FED–Vol. 236), pp. 11–20.

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

[7] Perić M. "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, 1992.

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

[9] 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.