International Parallel CFD'2001 Conference
Egmond aan Zee, The Netherlands, May 21-23, 2001

Efficient Parallel Simulation of Disperse Gas–Particle Flows on Cluster Computers

Th. Frank*, K. Bernert*, K. Pachler*, H. Schneider**
Chemnitz University of Technology
Faculty of Mechanical Engineering and Process Technology
* Research Group on Multiphase Flow / ** SIVUS gGmbH
Chemnitz, Germany
Email : frank@imech.tu–chemnitz.de

Keywords : Gas–Particle flows, Eulerian–Lagrangian approach, domain decomposition, load balancing, cluster computing

Abstract

This paper deals with two different parallelization methods for the Lagrangian (PSI–Cell) approach, a frequently used method for the numerical prediction of disperse multiphase flows. Both presented algorithms are based on the Domain Decomposition method. The first algorithm applies a static assignment of grid partitions to the processors of the parallel machine (PM).
In the second parallelization method — the so–called Dynamic Domain Decomposition (DDD) — a dynamic assignment of grid and fluid flow information to PM processor nodes is used, leading to considerably increased parallel efficiency and a higher degree of flexibility in the application of the computational method to different flow conditions.
Results of performance evaluations are provided for two different typical test cases and for MIMD computer architectures of a large 528–processor Linux workstation cluster (CLIC – Chemnitz Linux Cluster) and a 64–processor Cray T3E as well.

1 The Parallel Algorithm for Fluid Flow Calculation

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 a block-structured grid the flow domain is partitioned in a number of subdomains. Usually the number of grid blocks exceeds the number of processors, so that each processor of the parallel machine (PM) has to handle a few blocks. The grid-block-to-processor assignment is given by a heuristicaly determined block–processor allocation table and remains static and unchanged over the time of fluid flow calculation process.
The gas flow calculation is then performed by individual processor nodes on the grid partitions stored in their local memory. 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. Details of the parallel solution method for the gas flow can be found in [1].

2 Parallel Algorithms for the Lagrangian Approach

The prediction of the motion of the disperse phase is carried out by the application of the Lagrangian approach as described in references 5-9 in [1]. Considering the parallelization of this algorithm there are two important issues. 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. Therefore the distribution of the numerical work load in space is not known at the beginning of the computation. As a second characteristic 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. Considering these issues the following parallelization methods have been developed:

2.1 Static Domain Decomposition (SDD) Method

In the first approach geometry and fluid flow data are staticly distributed over the processor nodes of the PM in accordance with the block–processor allocation table as already used in the fluid flow field calculation of the Navier–Stokes solver.
Further an explicit host–node process scheme is established. The trajectory calculation is done by the node processes whereas a host process carries out only management tasks. The node processes are identical to those that do the flow field calculation. Now the basic principle of the SDD method is that in a node process only those trajectory segments are calculated that cross the grid partition(s) assigned to this process.
An advantage of the SDD method is that it is easy to implement and uses the same data distribution over the processor nodes as the flow solver. But poor load balancing can be a serious disadvantage of this method, e.g. due to large differences in the particle concentration distribution in the flow.

2.2 Dynamic Domain Decomposition (DDD) Method

This method has been developed to overcome the disadvantages of the SDD method concerning the balancing of the computational work load. In the DDD method there exist three classes of processes: the host, the servicing nodes and the calculating nodes (Figure 1).

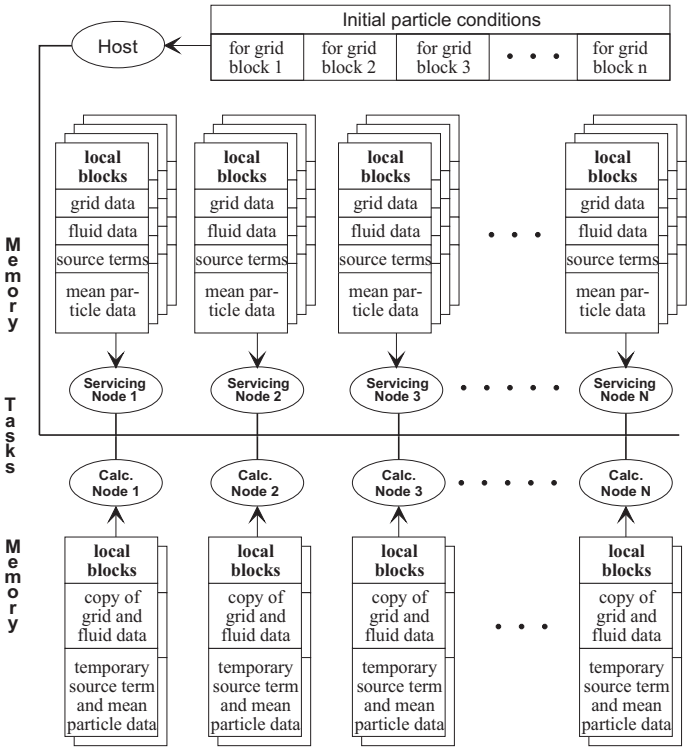

Figure 1: DDD method for Lagrangian solver.

Just as in the SDD 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 new class of servicing nodes uses the already known block-processor assignment table from the Navier–Stokes solver for storage of grid and fluid flow data. But in contrast to the SDD method they do not perform trajectory calculations but delegate that task to the class of calculating nodes. So the work of the servicing nodes is restricted to the management of the geometry, fluid flow and particle flow data in the data structure prescribed by the block-processor assignment table. On request a servicing node is able to dynamically retrieve or store data from/to the grid partition data structure stored in its local memory.
The calculating nodes are performing the real work on particle trajectory calculation. These nodes receive the particle initial conditions from the host and predict particle motion on an arbitrary grid partition. In contrast to the SDD method there is no fixed block-processor assignment table for the calculating nodes. Starting with an empty memory structure the calculating nodes are able to obtain dynamically geometry and fluid flow data for an arbitrary grid partition from the corresponding servicing node managing this

part of the numerical grid. The correlation between the required data and the corresponding servicing node can be looked up from the block-processor assignment table. Once geometry and fluid flow data for a certain grid partition has been retrieved by the calculating node, this information is locally stored in a pipeline with a history of a certain depth. So the concept of the DDD method makes it possible to perform calculation of a certain trajectory segment on an arbitrary calculating node process and to compute different trajectories on one grid partition at the same time by different calculating node processes, thus establishing a nearly perfect load balancing between processors of the PM.

3 Results and Discussion

Calculations with both parallelization methods have been performed on two typical test cases. The simulations on a varying number of processor nodes has been carried out on two cluster computer systems : a) a 12–processor AMD–Athlon (600 MHz) based cluster of workstations and b) the Chemnitz Linux Cluster (CLIC) with up to 528 Intel/Pentium III nodes (800 MHz) with 512 Mb memory per node and FastEthernet interonnect. For comparison a number of calculations has been made on a 64–processor Cray T3E. 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 (calculation of 5000 particle trajectories, one-way-coupling). From these measurements the difference time ($T_{diff} = T_{total} - T_{calc} - T_{comm} - T_{I/O}$) has been calculated. This difference time contains mainly the waiting time for the processor nodes in receive operations and global barriers.

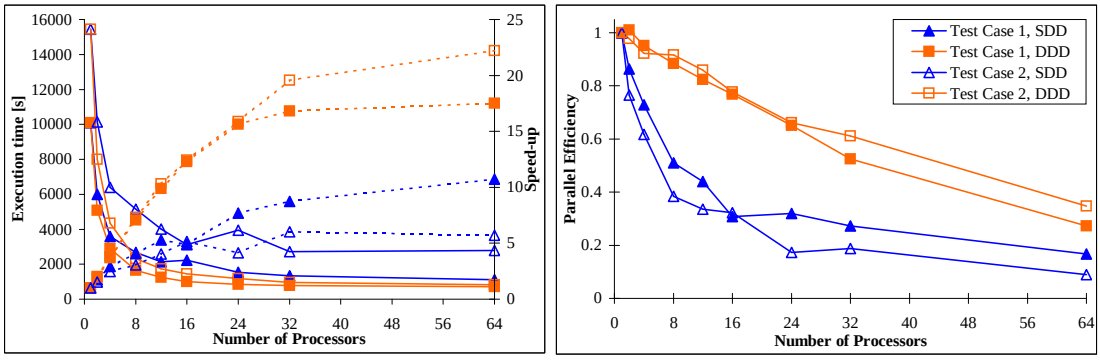

Figure 2: Total execution times, speed–up and parallel efficiency vs. number of processor nodes (CLIC); comparison of parallelization methods for both test cases.

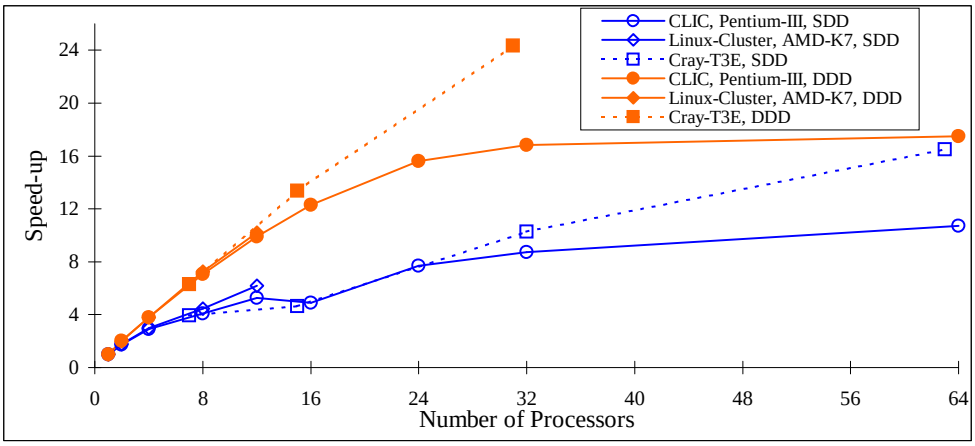

Figure 3: Comparison of parallel performace on Chemnitz Linux Cluster(s) vs. Cray T3E

Figure 2 show the total execution times, the speed–up and the parallel efficiency for calculations on both test cases with SDD and DDD methods vs. the number of processor nodes. All test case calculations in this experiments had been carried out on the second finest grid level with 396.800 CV's. Figure 2 show the remarkable reduction in computation time with both parallelization methods. It can be seen from the figures that in all cases the Dynamic Domain Decomposition (DDD) method has a clear advantage over the SDD method. But this advantage for the DDD method in the first test case is not as remarkable

as for the second test case. This is due to the fact, that the gas–particle flow in the first test case is quiet homogeneous in respect to particle concentration distribution which leads to a more balanced work load distribution in the SDD method. So the possible gain in performance with the DDD method is not as large as for the second test case, where the gas–particle flow is strongly separated and where we can observe particle roping and sliding of particles along the solid walls of the channel leading to a much higher amount of numerical work in certain regions of the flow. Consequently the SDD method shows a very poor parallel efficiency for the second test case due to poor load balancing between the processors of the PM (figure 2).
Figure 3 shows the comparison of test case calculations between the CLIC, an AMD–Athlon based workstation cluster and the Cray T3E for the test case 1. Again we can observe the clear advantage of the DDD method with the implemented dynamic load balancing scheme leading to an acceleration of the Lagrangian particle trajectory calculation by a factor of up to 3 on the Cray T3E in comparison with the SDD method. The impact of the Cray high-bandwith-low-latency interconnection network can clearly be seen from the figure. So the speed–up for the test case calculations on the Cray increases almost linearly with increasing number of processors up to 32 nodes. On the CLIC we observe minor speed–up values and reach saturation (effect of higher latency, bandwith limits of the FastEthernet communication network) for calculations on more than 32 processor nodes where a further substantial decrease of the total execution time for the Lagrangian solver could not be achieved. But similiar scaling behavior of the presented Eulerian–Lagrangian approach as it was observed for the Cray T3E should be within reach on workstation clusters with highspeed interconnects like e.g. Myrinet, GigaNet, SCI or other new emerging network technologies.

4 Conclusions

The paper presents two parallelization methods for the Eulerian–Lagrangian approach for disperse multiphase flow calculations together with their MPI implementations. Performance results are given for two typical test cases. The obtained results show the importance of homogeneous work load distribution, which has to be treated differently in comparison with Static Domain Decomposition (SDD) methods for common single phase flow computations. With the presented Dynamic Domain Decomposition (DDD) method remarkable speed–up can be achieved for the Eulerian–Lagrangian computation of disperse multiphase flows on MIMD computers and clusters of workstations. The main advantages of the DDD method are :

- dynamic work load distribution among the processors of the parallel machine; work load balancing is effective even on heterogeneous computer systems like eg. workstation clusters with different computational power of processor nodes;
- independence of performance from the subdivision of the numerical grid into grid blocks,
- independence of the algorithm performance from the flow regime (e.g. phase separation, non–homogeneous concentration distribution of the dispersed phase, local occurence of strong particle–wall interaction).

So the developed parallelization method with dynamic work load balancing offers new perspectives for the computation of strongly coupled multiphase flows with complex phase interactions and higher particle concentrations.

Acknoledgements

This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) in the framework of the Collaborative Research Centre SFB–393 under Contract No. SFB 393/D2.

References

[1] **Bernert K., Frank Th., Schneider H., Pachler K. :** "Multi–Grid Acceleration of a SIMPLE-Based CFD-Code and Aspects of Parallelization", IEEE Int. Conf. on Cluster Computing – CLUSTER 2000, Nov 28.–Dec 2., 2000, Chemnitz, Germany.

[2] Web Site of the Research Group on Multiphase Flow
http://www.imech.tu-chemnitz.de