**IEEE International Conference on Cluster Computing,
CLUSTER 2000, November 28.–December 2., 2000,
Chemnitz University of Technology, Saxony, Germany.**

Efficient Parallelization of Eulerian–Lagrangian Approach for Disperse Multiphase Flow Calculations on MIMD Computer Architectures

Thomas Frank Klaus Bernert Klaus Pachler Hellfried Schneider

Chemnitz University of Technology
Reichenhainer Straße 70, 09107 Chemnitz, Germany
Email: frank@imech.tu-chemnitz.de

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).

Just as in the SDD method the host process distributes the particle initial conditions among the calculating nodes

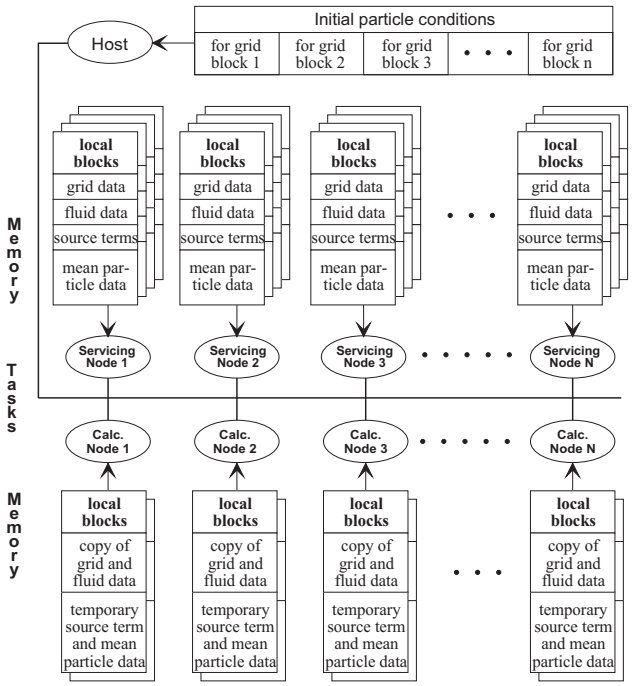

Figure 1. DDD method for Lagrangian solver.

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.

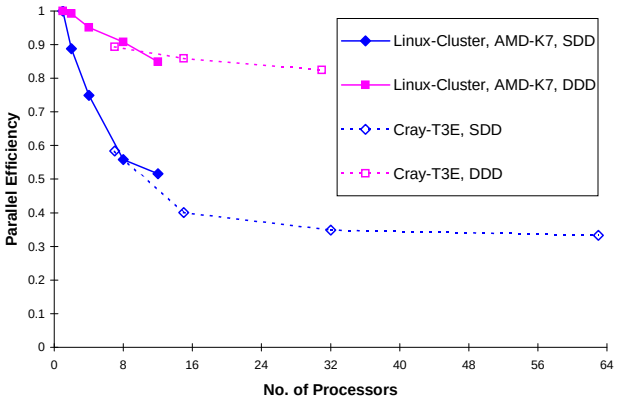

Figure 2. Comparison for test case 1.

3 Results and Discussion

Two different test cases has been investigated. Figure 2 shows a comparison of performance results for the calculation of a moderately separated gas–particle flow in a multiple bended channel with corner vanes. Calculations were performed on a Linux cluster of 12 AMD-Athlon-PC's under LAM–MPI (TU Chemnitz) and on a Cray–T3E with 64 PE's (TU Dresden). Results for the large 528 processor Linux cluster of the TU Chemnitz will be available on the CLUSTER 2000 conference and on the Web [2].

The obtained results show besides the general applicability of the SDD and DDD parallelization methods to parallel MIMD computers the importance of homogenous work load distribution, which has to be treated differently in comparison with SDD methods for common computations in the field of computational fluid dynamics (CFD). With the presented DDD method remarkable speed–up can be achieved for the Eulerian–Lagrangian prediction of disperse multiphase flows on MIMD computers and clusters of workstations. 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.

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