Fine-Grained Parallel Algorithms in TIM-3D Code

Andrey Alexandrovich Voropinov(✉) and Ivan Gennadievich Novikov

FSUE Russian Federal Nuclear Center – All-Russian Research Institute of Experimental Physics, Sarov, Russia
{AAVoropinov,IGNovikov}@vniief.ru

Abstract. TIM-3D is a continuum-mechanics simulation code that uses arbitrary-shape unstructured polyhedral Lagrangian meshes. Parallelism in TIM-3D is provided at three levels in the mixed-memory model. The first two levels use space decomposition in the MPI-based distributed-memory model. At the first level, calculations are parallelized in task fragments (domains). At the second level, calculations within one domain are parallelized in para-domains. At the third level, iterations of calculation loops are parallelized in the OpenMP-based shared-memory model. The paper considers the fine-grained paralleling algorithms (second level). These algorithms are complementary to the OpenMP shared-memory parallelism implemented earlier. The fine-grained paralleling can be done both with overlapping in one row of para-domain interface cells and without overlapping. These approaches are compared in their parallel efficiency using one of test simulations.

Keywords: TIM-3D code · Distributed-memory parallelism · MPI
Unstructured meshes

1 Introduction

TIM-3D [1] is an unsteady continuum-mechanics simulation code that employs unstructured arbitrary-shape polyhedral Lagrangian meshes. Cells can have an arbitrary number of faces, and the faces can have an arbitrary number of nodes connecting an arbitrary number of cells and edges. Figure 1 shows some simple examples of meshes used in the code.

Parallelism in TIM-3D is provided at three levels. This approach is an extension of the three-level parallelism in TIM-2D [2] to the three-dimensional case. The first two levels use space decomposition in the MPI-based distributed-memory model. At the first level, calculations are parallelized in task fragments (domains). At the second level, calculations within one domain are parallelized in para-domains. At the third level, iterations of calculation loops are parallelized in the OpenMP-based shared-memory model. These approaches can be used both together in different combinations, and separately in one calculation.

Earlier, TIM-3D used shared-memory model parallelism [3]. Shared-memory parallelism is not sufficient, because the number of memory-sharing processor

L. Sokolinsky and M. Zymbler (Eds.): PCT 2018, CCIS 910, pp. 147–161, 2018.
https://doi.org/10.1007/978-3-319-99673-8_11

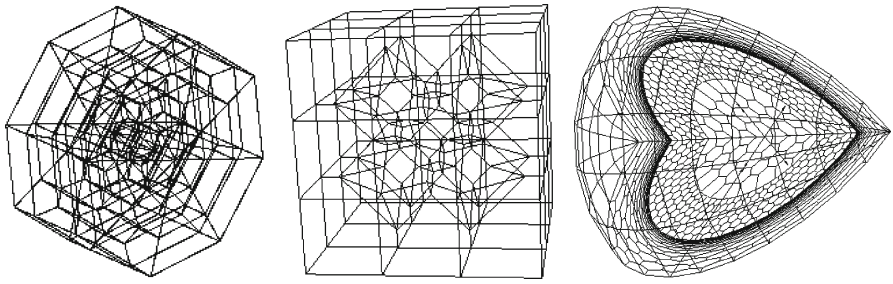

Fig. 1. Examples of polyhedral meshes used in TIM-3D

cores in up-to-date cluster computers is not very large. TIM-3D uses decomposition into domains. This involves solving a contact interaction problem between domains. Domains are calculated independently. This circumstance is used for the first level of parallelism. Its constraint is a small number of domains (usually up to 10). Thus, in order to remove the constraints on the number of computational resources engaged, an intermediate level of parallelism is required: at the sub-domain level, where the domain is divided into smaller geometric "grains". These fine-grained paralleling algorithms are described in the present paper.

The development of the parallel algorithms is based on the following principles:

- Identity of calculation outputs in any mode of calculations.
- Scalability, or possibility of running calculations on any number of cores with easy switch-over between the modes from one start to another for a single task.
- Minimum memory consumption—to run tasks that are too large for the memory available to a single core.
- Optimum utilization of computational resources. Prevention of imbalance, or in-process balancing when this occurs.

To facilitate the development of computational programs:

- Universal data representation in any mode of calculations.
- Minimum revision of codes to make them parallel; the burden of managing parallel computations lies with a set of supporting programs.

2 Data Decomposition

Efficient use of computational programs on parallel computers requires decomposition to ensure uniform distribution of work load among computer cores with as little communication as possible. Decomposition in the distributed-memory model includes distribution of data among processes (data decomposition) in such a way that the number of data transfers and the volume of communicated

data between them is minimum. For distributed-memory parallelism, TIM-3D uses space decomposition. Decomposition principles for fine-grained parallelism are as follows:

- The decomposition is performed by cells (cells are the basic computing mesh elements in TIM-3D).
- All domain cells are distributed among compacts so that each cell belongs to only one compact.
- Each domain is split into compacts irrespective of other domains.

The problem of decomposition for fine-grained parallelism comes down to solving a problem of graph partitioning into subgraphs. This is accomplished by the following algorithm:

- A graph representing the mesh structure is built based on the unstructured mesh. Graph nodes correspond to mesh cells, and graph edges, to neighborhoods between cells.
- Graph nodes are assigned the weight reflecting the computational load associated with the corresponding cell. Weights of graph edges are used to introduce additional decomposition properties. For example, extending compacts along boundaries reduces the number of data transfers in contact interaction calculations.
- The problem of graph partitioning into subgraphs is solved using algorithms from the ParMeTiS or SCOTCH libraries [4–6] and our own hybrid (topological and geometric) decomposition algorithm.

Examples of resulting decompositions done by SCOTCH algorithms and by our own algorithm are shown in Fig. 2.

Fig. 2. Examples of decompositions: SCOTCH (left), hybrid (right)

3 Fine-Grained Parallelism

TIM-3D uses the staggered centering stencil. Kinematic quantities (velocities, accelerations, coordinates) are assigned to cell nodes, while thermodynamic quantities (energy, pressure, density, etc.) are assigned to cells. As a result, the key issue associated with the fine-grained parallelism is the way of calculating the mesh nodes (cell vertices) surrounding the cells belonging to different compacts (para-boundary nodes). For simplicity, let us illustrate this with the two-dimensional case shown in Fig. 3. In the figure, the white cells belong to compact 1, and the yellow ones, to compact 2.

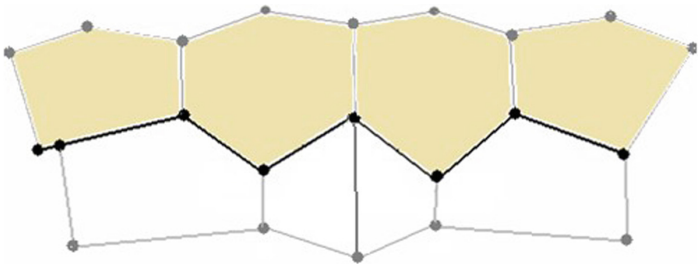

Fig. 3. A mesh fragment partitioned into compacts (Color figure online)

In accordance with the difference scheme of TIM-3D [1], a closed integration contour of cell centers and edge centers is constructed to calculate a node. For the two-dimensional case of interest, an example of an integration contour for the node V in Fig. 4 is marked by a green line. The integration contour is defined by the centers of the surrounding cells C_1, C_2, C_3, C_4, and the "centers" of the edges VV_1, VV_2, VV_3, VV_4. In conformity with the integration contour, the node V is calculated using the quantities in both the cells C_1, C_2, C_3, C_4, and the nodes V, V_1, V_2, V_3, V_4.

If the integration contour is preserved, we obtain the first type of fine-grained parallelism (with one layer of overlapping cells). In order to preserve the node integration contour on the side of the first para-domain, the cells C_2, C_3 do not need to be generated completely, i.e. no information on the nodes V_5, V_6 is required. However, if we do not attach these nodes to the first para-domain, the mesh will be incomplete, and some operations on the cells C_2, C_3, for example, definition of mesh nodes, volume calculations, or determination of the center, will be unavailable. In this case, the attached cells need to be described in the data structure in a special way and accounted for in different computing algorithms. As the highest possible transparency of parallelism for computing algorithms is one of the basic paralleling principles, it was decided to include such nodes into para-domains as attached nodes, i.e. the nodes V_3, V_5, V_6 are attached with respect to para-domain 1.

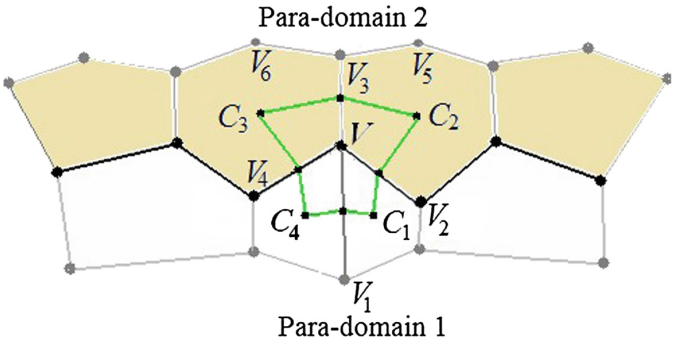

Fig. 4. Node integration contour (Color figure online)

On the other hand, the integration contour can be represented as a set of closed contours on the side of each cell (this representation is also used in the difference scheme to determine the node mass [1]). Similarly, the integration contour can also be represented as a set of closed contours on the side of each para-domain. Such a partitioning for the case under consideration is shown in Fig. 5, where the partitioning line of the integration contour is marked with red. When the integration contour is partitioned, the nodes along the para-domain interface are divided into pairs (or proportional to the number of para-domains connected at the node), for example, the node V is partitioned into V' and V''. For each node, a separate integration contour is used to determine the mass and accelerations, which are then combined to calculate the common velocity. Such an integration contour partitioning in the simulation makes it possible not to use the overlapping between para-domains. A similar approach is used for "no-slip" boundaries [7] and for fine-grained parallelism in TIM-2D [8].

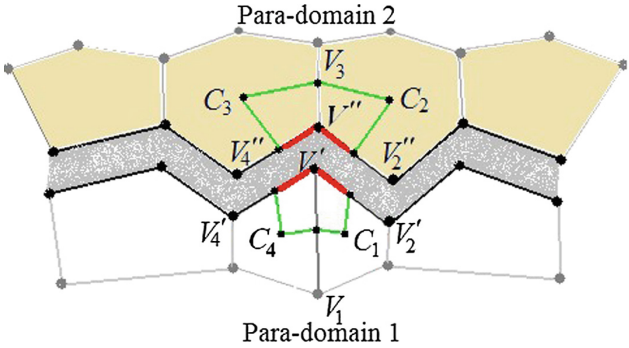

Fig. 5. Node integration contour partitioning during fine-grained paralleling without overlapping (Color figure online)

A similar volume element in the three-dimensional case is constructed as follows (see Fig. 6). Consider node i. Let the point j be the center of a cell adjacent to the node i. We draw three planes through the center j of the polyhedron (Fig. 6):

- the first passing through the node of interest i and one of the edges connected at this node and belonging to the cell having its center at j, for example, ii_1;
- the second passing through the node i and the center c_1 of one of two faces of polyhedron j to which the edge ii_1 also belongs;
- the third passing through the middle i_2 of the same edge ii_1 and the center c_1 of the face.

The triangular pyramid jii_2c_1 is part of the volume of the mass belonging to the node i. Using the same procedure, we construct another triangular pyramid with another edge belonging both to the face c_1, the node i and the cell of interest centered at j. Now we proceed to other faces belonging to the cell j and the node i at the same time. The number of such faces is equal to the number of faces of the polyhedral angle corresponding to the node i and the cell j (in most cases, they are three).

This resulting set of triangular pyramids generates the polyhedron belonging to the node on the face of the cell of interest j.

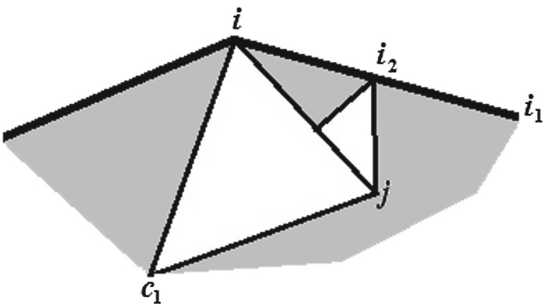

Fig. 6. Nodal cell element in TIM-3D

Interactions between para-domains are always pairwise. One can therefore speak about introducing para-boundaries. Para-boundaries include para-boundary nodes and faces separating the cells calculated in different para-domains. The para-boundaries also contain cell elements in the overlapping layer (if it is used): attached and near-boundary cells, faces, nodes.

Domain partitioning into para-domains does not change the difference scheme of TIM-3D in both fine-grained parallel modes, ensuring the identity of their results with that of the serial mode.

Finite-difference codes generally employ fine-grained parallelism with overlapping (see, e.g., [9–11]). A similar approach is to fill in missing data that are

needed to calculate equations (see, e.g., [12]). The approach with node integration contour partitioning has been proposed for TIM-2D [8]. The present work considers its extension to the three-dimensional case.

If one compares the approaches, then each of them will have both advantages and disadvantages. The non-overlapping method demands less communication, because only nodal quantities are exchanged, whereas both nodal and cell-centered quantities are exchanged in the mode with overlapping. The volume of communicated data in the non-overlapping mode is also much smaller, because only data on para-boundary nodes are transferred, whereas in the mode with overlapping, this volume also includes data on the nodes of the overlapping layer. This results in higher efficiency of the non-overlapping mode. This approach is more convenient for algorithm programming, because all cell elements are computable, and the para-domain is in fact nearly identical to the mathematical domain. A constraint of the non-overlapping approach is that the difference scheme of the code should allow for integration contour partitioning. The drawback of the non-overlapping approach is that new limitations of the algorithms involving cell analysis around nodes at the para-domain interface occur. Examples of such algorithms include mesh maintenance algorithms (for example, it becomes impossible to combine directly cells from different para-domains). The computational load also increases a little because of the recovery of the common node integration contour.

The mode with overlapping is free of these limitations and drawbacks, which makes it more general. But its efficiency turns out to be a little lower because of the growing volume of exchanged data and number of data transfers.

4 Specific Features of Cell Neighborhood in the Three-Dimensional Case

In the three-dimensional case, para-domain generation has a number of specific features that distort the mesh structure in the para-domain. Such features are impossible or exceptional in the two-dimensional case, while in the three-dimensional case they are present in quite consistent decompositions.

The first class of features occurs at para-domain interfaces. They are cell neighborhoods along an edge or across a node (see Fig. 7). Such features make it difficult to describe the mesh structure [13], for example, to get a list of surrounding cells for nodes.

Features of the second class occur at outer domain boundaries. They include outbreaks of para-domains to the outer boundary with one edge or node. The simplest example of such a feature is the decomposition of a regular spherical mesh into columns or rows (see Fig. 8). The features of this class cause problems in calculations of contact interactions since they involve calculations of surface interactions, even though edges and nodes do not generate any surfaces.

These features constitute a certain challenge for the non-overlapping fine-grained parallelism because they directly influence calculations of the respective mesh nodes. These problems are not so evident in the overlapping mode since

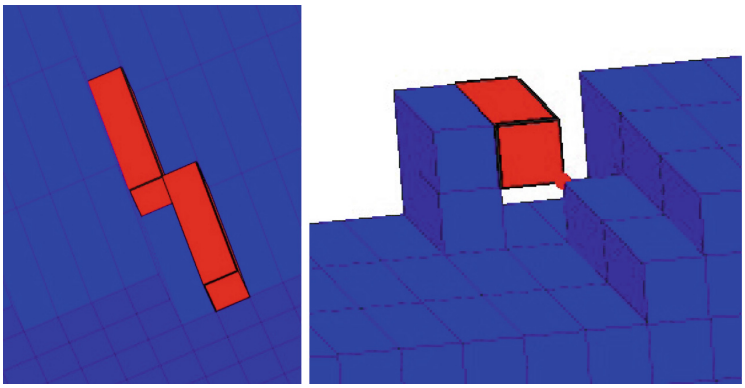

Fig. 7. Features in para-domain cell neighborhood: neighborhood along an edge (left), neighborhood across a node (right)

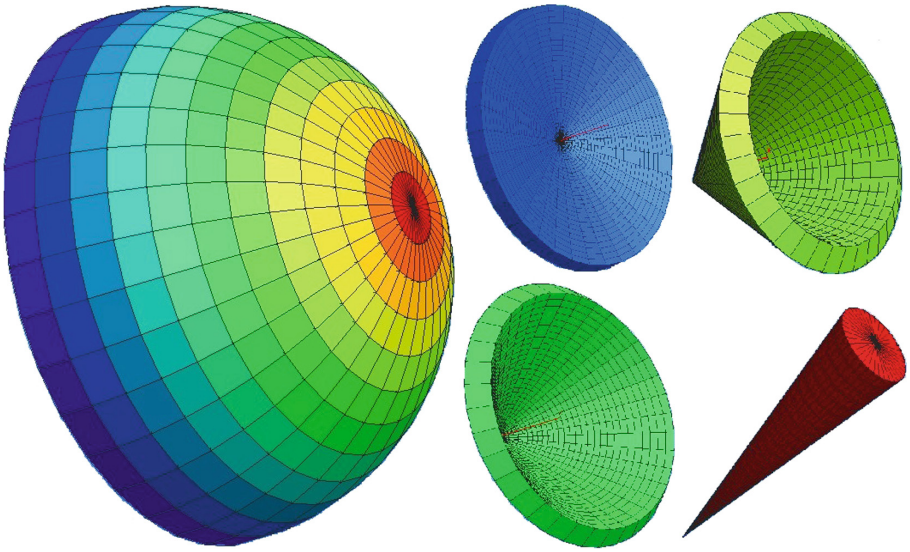

Fig. 8. Example of single-point outbreaks of para-domains onto the outer surface

the data of the feature are mostly transferred to an attached layer which is not processed. Nevertheless, these features should be kept in mind in this mode too.

To solve the above-mentioned problems, an additional object, a boundary node, has been introduced into the algorithms. This object is introduced for nodes at both outer and parallel boundaries. The following information is stored for boundary nodes:

- A full set of boundary conditions (both outer and parallel).
- A number of countable quantities essential for contact interaction calculations, such as vectors of outward normals, work, etc.

– Specifically for fine-grained parallelism, numbers of all faces connected at the boundary node are stored.

The storage of additional boundary node information makes it possible to incorporate arising features into the algorithms.

5 Fine-Grained Paralleling Algorithms

Fine-grained parallelism in para-domains includes processing of interior and near-boundary mesh elements. Cell calculations are therefore strictly associated with para-domains. However, this poses a question regarding nodes: how to calculate para-boundary nodes? As stated in Sect. 3 above, in the non-overlapping mode, para-boundary nodes are firstly calculated independently in each para-domain and then matched. This requirement is removed in the overlapping fine-grained paralleling mode, and para-boundary nodes can be calculated by any process calculating surrounding para-domains. As node calculations are rather inexpensive in TIM-3D, calculations of para-boundary nodes are backed up in this case. In TIM-3D, gas dynamic quantities are calculated in two major steps:

1. Calculations of nodal quantities, such as velocities, coordinates (calculation of the equation of motion). Node calculations are performed using data from neighbor nodes and surrounding cells from the previous time step.
2. Calculations of cell quantities, such as density, pressure, energy (calculation of the energy equation). These are done with updated node locations of the cells under consideration, based on which changes in the cell volume at an iteration time step are calculated.

The overlapping mode encounters the issue of data update in attached mesh elements (cells and nodes). This update is performed by asynchronous communication in para-boundary, near-boundary and attached elements. Calls of communication procedures are placed in such a way that the required information is updated before its use.

A flow diagram of a time step involving fine-grained paralleling with overlapping is shown in Fig. 9. The flow diagram for the non-overlapping mode remains the same, except for missing transfer of cell quantities. Likewise, once information is received, an additional integration contour recovery operation is performed.

The green boxes in Fig. 9 represent the calls of asynchronous communication procedures for nodal quantities, and the yellow boxes, for cell quantities; dashed lines are actual data streams between processes. The flow diagram shows that data transfers are combined with calculations of interior cells and nodes, which enables their parallel running. Communication involves packaging and unpackaging of quantities in the buffer array.

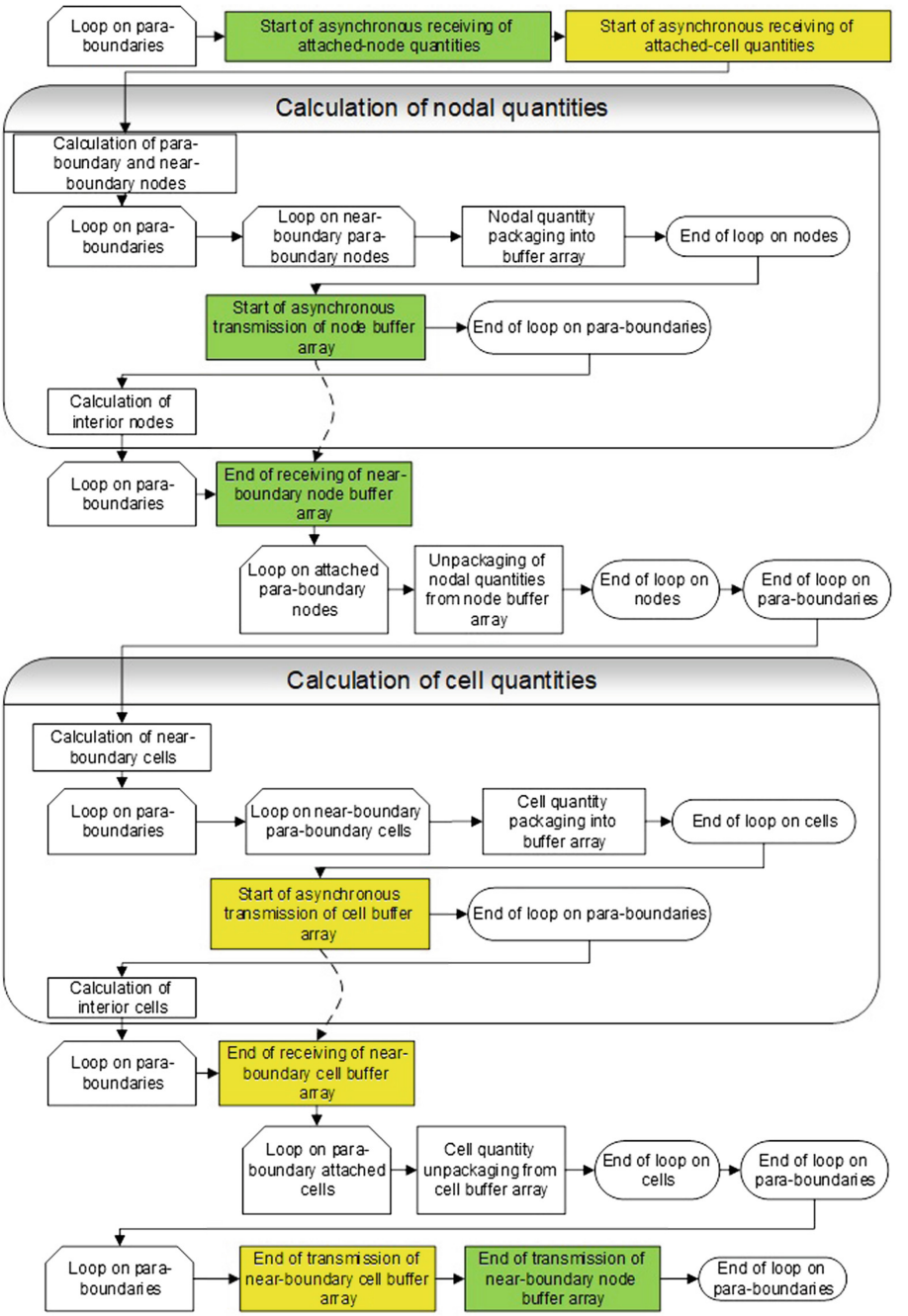

Fig. 9. Flow diagram of nodal and cell quantities in the overlapping fine-grained paralleling mode (Color figure online)

6 Distinctive Features of Non-overlapping Fine-Grained Parallelism

The approaches employed in TIM-3D paralleling algorithms are basically similar to those employed in TIM-2D [2]. In addition, most computational modules in both codes use the same programs for the two-dimensional and three-dimensional cases. The major difference in the parallelism of TIM-3D lies in the fine-grained paralleling algorithms with non-overlapping cell layers.

In TIM-2D, the integration contour partitioning scheme is used to calculate the no-slip boundary motion, when no partitioning is allowed at the domain boundary not only for the normal, but also for the tangential velocity component. In TIM-2D, the non-overlapping fine-grained parallelism is implemented by a modified algorithm for no-slip boundary calculations [7].

In the three-dimensional case, however, the approach employing the contact interaction algorithms cannot be used for a number of reasons. The main one is that the boundary interface in the two-dimensional case is a combination of two broken lines. Each line is drawn using a strictly defined series of boundary points in each domains. The three-dimensional case involves surfaces, and it is impossible in this case to set up a strict series of points and hence ensure the node-to-node point matching especially with active execution of mesh maintenance algorithms (to maintain the required shape of Lagrangian cells). In addition, contact interaction algorithms in the three-dimensional case become much more complicated themselves, primarily as a result of the transition to surface interaction. This makes the contact interaction algorithms significantly more expensive. Note that even in the two-dimensional case, calculations of boundary points are several times more expensive than calculations of interior points. It is therefore preferable to prevent the buildup of boundary points in the three-dimensional case.

For these reasons, in the three-dimensional case, for the non-overlapping fine-grained parallelism, it was decided to implement a program matching nodal quantities upon exchange of accelerations and masses for para-boundary nodes, rather than to use the contact interaction algorithms.

The node integration contour recovery algorithm is as follows:

- Prior to starting a calculation, masses of para-boundary nodes are calculated over the integration contour belonging to the para-domain of interest. That is, one para-boundary node has its own mass in different para-domains, and its total mass is the sum of its masses in the para-domains ($m_i = \sum_{j=1}^{k} m_j$ is the mass of the para-boundary node i with respect to k para-domains ($k \geq 2$), m_j is the mass of the para-boundary node in the corresponding para-domain).
- The para-boundary nodes are calculated in each para-domain completely independently. Here, no forces are applied from the side of the parallel boundary (the parallel boundary serves as a free surface with pressure $P_G = 0$).
- Once all the para-boundary nodes are calculated, their resulting accelerations and masses are exchanged between the para-domains. Generally speaking,

node masses do not vary in the course of gas dynamic equation calculations, but their variations are possible as a result of the execution of mesh maintenance algorithms. In addition, more complicated algorithms may require some additional quantities, so whole sets of nodal quantities are exchanged.

– Once the exchange is over, total accelerations, velocities and positions of para-boundary nodes are calculated in each para-domain as follows:

$$\mathbf{a}_i^{n+1} = \frac{\sum\limits_{j=1}^{k} m_i \mathbf{a}_j^{n+1}}{\sum\limits_{j=1}^{k} m_j} - \text{total acceleration of node } i;$$

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \tau \mathbf{a}_i^{n+1} - \text{total velocity of node } i;$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \tau \mathbf{v}_i^{n+1} - \text{updated node position;}$$

where τ is the time step.

Note that the additional update can lead to truncation errors due to specific features of machine arithmetics, i.e., the resulting velocity calculated by different processes can differ in its 16th digit. This error can build up with time and result in mismatch between para-boundary nodes (a gap or overlapping between para-domains). Although the mismatch is tiny, some algorithms can be sensitive even to such discrepancies. To overcome this problem, (1) the resulting velocity is rounded by assuming that near-zero accelerations and velocities are zero, and (2) the velocities and node positions are averaged at time instant n. Velocities and coordinates are calculated as follows:

$$\mathbf{v}_i^{n+1} = \frac{\sum\limits_{j=1}^{k} \mathbf{v}_j^n}{k} + \tau \mathbf{a}_i^{n+1} - \text{total velocity of node } i;$$

$$\mathbf{r}_i^{n+1} = \frac{\sum\limits_{j=1}^{k} \mathbf{r}_j^n}{k} + \tau \mathbf{v}_i^{n+1} - \text{updated node position.}$$

This prevents any mismatch between matched para-boundary nodes belonging to different para-domains.

7 Measurements of Parallel Efficiency

To assess the parallel efficiency, we used the functions $S_p = \frac{t_1}{t_p}$ (speedup of calculations), and $E_p = \frac{t_1}{pt_p} \cdot 100\%$ (parallel efficiency), where t_1 is the calculation time on one processor of the parallel computer (serial calculations), t_p is the calculation time on p processors.

For the test, we chose a planar-wave problem [14]. The calculation was run on an unstructured hexahedral mesh of 1 million cells. The results of time,

acceleration and efficiency measurements are summarized in Table 1. We used as a basic unit the time for running the calculation of one compute node with OpenMP parallelism only. This allowed us to evaluate the efficiency of the fine-grained paralleling block in the mixed mode.

The results indicate that the efficiencies of the fine-grained parallel modes with and without overlapping are close up to 10 compute nodes, whereas, for a greater number of nodes, the efficiency of the non-overlapping parallel mode becomes higher (by 7 to 10 %). This is explained by a smaller number of data transfers and a smaller volume of communicated data.

Table 1. Measured speedup and parallel efficiency

Mode→		With overlapping			Without overlapping		
Core count	Node count	Time, s	Speedup	Efficiency	Time, s	Speedup	Efficiency
16	1	4012.10	1	100%	4012.10	1	100%
32	2	2224.96	1.80	90.16%	2268.58	1.76	88.42%
64	4	1116.10	3.59	89.86%	1144.32	3.50	87.65%
128	8	629.42	6.37	79.67%	594.82	6.74	84.31%
160	10	515.88	7.77	77.77%	495.80	8.09	80.92%
256	16	359.40	11.16	69.77%	321.14	12.49	78.08%
320	20	303.34	13.22	66.13%	266.54	15.05	75.26%
384	24	264.42	15.17	63.22%	230.37	17.41	72.56%
512	32	222.41	18.03	56.37%	186.29	21.53	67.30%
640	40	193.15	20.77	51.92%	164.17	24.43	61.09%
800	50	169.12	23.72	47.44%	143.87	27.88	55.77%
1600	100	146.71	27.34	27.34%	110.78	36.21	36.21%

8 Conclusions

The paper describes two fine-grained parallel methods used in TIM-3D code. In the first method, the whole node integration contour is preserved, and para-domains overlap in one layer of cells. The overlapping layer serves for node and cell data communication. In the second method, the node integration contour is partitioned, and para-domain interactions are calculated. These methods are close in their parallel efficiency on a small number of compute nodes (up to 10), while, on a large number of nodes, the non-overlapping method turns out to be 7 to 10 % more efficient. The higher efficiency is achieved owing to a smaller number of data transfers and a smaller volume of communicated data. Most calculations in TIM-3D are performed in the non-overlapping fine-grained parallel mode.

The algorithms developed are complementary to the OpenMP parallelism implemented earlier in TIM-3D.

References

1. Sokolov, S.S., Panov A.I., Voropinov, A.A., et al.: The code TIM for three-dimensional continuum mechanics simulations on unstructured polyhedral Lagrangian meshes. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 3, pp. 37–52 (2005). (in Russian)
2. Voropinov, A.A., Sokolov, S.S.: The method of three-level paralleling in the code TIM-2D. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 4, pp. 70–77 (2013). (in Russian)
3. Voropinov, A.A., Novikov, I.G., Sobolev, I.V., Sokolov, S.S.: Paralleling of the code TIM in the shared-memory model using the OpenMP interface. Vychislitel'nye metody i programmirovanie. **8**(1), 134–141 (2007). (in Russian)
4. Polovnikova, T.N., Voropinov, A.A.: Experience of using the SCOTCH and MeTiS libraries in unstructured mesh decomposition in the code TIM. In: Shagaliev, R.M. (ed.) Proceedings of the XII International Workshop on Supercomputing and Mathematical Modeling, pp. 282–288. FSUE RFNC-VNIIEF, Sarov (2011). (in Russian)
5. ParMETIS: Parallel graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview
6. Pellegrini, F.: SCOTCH: Static mapping, graph, mesh and hypergraph partitioning, and parallel and sequential sparse matrix ordering package. http://www.labri.fr/perso/pelegrin/scotch/
7. Voropinov, A.A., Novikov, I.G., Sokolov, S.S.: Calculations of contact interaction between domains in the code TIM-2D. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 2, pp. 5–20 (2008). (in Russian)
8. Voropinov, A.A., Novikov, I.G., Sokolov, S.S.: Fine-grained paralleling methods in the code TIM-2D. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 3, pp. 24–33 (2012). (in Russian)
9. Pronin, V.A.: Paralleling methods for two-dimensional gas dynamics simulations on unstructured meshes with variable topology in the code MEDUZA. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 1, pp. 54–67 (2011). (in Russian)
10. Gasilov, V.A., Diachenko, S.V., Boldarev, A.S., et al.: Application package MARPLE3D for pulsed magnetically driven plasma simulations on high-performance computers, p. 20. Keldysh Institute of Applied Mathematics, Moscow (2011, preprint). (in Russian)
11. Lyapin, V.V., Korolev, R.A., Vetchinnikov, A.V.: A paralleling method with two-dimensional mesh decomposition for numerical solution of the two-dimensional heat transfer equation using the code KORONA-2D. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 2, pp. 69–77 (2014). (in Russian)
12. Andrianov, A.N., Efimkin, K.N.: An approach to implementation of numerical methods on unstructured meshes. Vychislitel'nye metody i programmirovanie **8**, 6–17 (2007). [in Russian]
13. Voropinov, A.A., Sokolov, S.S., Panov, A.I., Novikov, I.G.: Polyhedral arbitrary-structure mesh description format in the code TIM. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 3–4, pp. 55–63 (2007). (in Russian)

14. Bondarenko, Yu.A., Voronin, B.L., Delov, V.I., et al.: Description of a test suite for two-dimensional gas dynamic codes and programs. Part 1. Test requirements. Tests 1–7. In: Voprosy atomnoi nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov, no. 2, pp. 3–9 (1991). (in Russian)