Parallel algorithm for the solution of the Boltzmann Transport Equation on unstructured meshes

G. Colomer*, R. Borrell*, O. Lehmkuhl*,†, C.D. Pérez-Segarra†
†Centre Tecnològic de Transferència de Calor (CTTC)
Universitat Politècnica de Catalunya (UPC)
ETSEIAT, Colom 11, 08222 Terrassa, Barcelona, Spain
Email: cttc@cttc.upc.edu

∗ Termo Fluids, S.L., Magí Colet 8, 08204 Sabadell, Spain.

Abstract: The Boltzmann transport equation is solved by means of a parallel explicit algorithm known as parallel sweep, and a source iteration procedure to solve the coupling between different angular directions. Using an explicit solver implies that the nodes have to be visited in a specific order. Factors that influence the scalability of the algorithm have been analysed, namely the different strategies to group the tasks to be done at the calculation stage, and how deep should the task graph be worked prior to the communication stage. Two and three dimensional unstructured meshes have been considered. With relatively small meshes, good weak and strong speedup results are obtained, using up to $\simeq$ 900 processors.

Keywords: Parallel BTE solver, radiation, unstructured meshes, parallel sweep

1 Introduction

The discrete ordinates form of the Boltzmann transport equation, generally used for the modelisation of the radiation effects, reads:

$$\hat{s}^i \cdot \nabla\phi^i + \beta_1\phi^i = \beta_0 + \beta_2 \sum_j \omega^j \psi^{ji} \phi^j \tag{1}$$

where ϕ is the radiative flux, the superindex i indicates that the magnitude is evaluated at the ordinate $\hat{s}^i$, normally $\beta_1\phi^i$ accounts for the loss factors produced by the absorption or scattering, β_0 is a source term, and the term $\beta_2 \sum_j \omega^j \psi^{ji} \phi^j$, which couples the different ordinate directions, accounts for the particles that change their propagation direction. This magnitude is evaluated at specific directions, weighted accordingly with weight ω^j. The phase function ψ^{ji} is a measure of the probability that a particle changes its propagation direction from $\hat{s}^j$ to $\hat{s}^i$.

The computational cost associated to the numerical solution of the BTE is high in terms of memory requirements and computational time, because the equations are discretised over the spatial and also the angular domain. One option is to solve all the ordinate directions at the same time by means of a single system [1]. However this is expensive in terms of memory requirements. Moreover, uncoupling the different ordinate directions, by means of source iteration algorithm for instance, we can take advantage of the good properties of the resulting subsistems and solve them in a very efficient way. Of the many numerical methods used to solve the BTE, results of this work are applicable on these that employ a finite volume discretisation for the spatial part, and that use any quadrature to perform the angular integration. Among the most popular quadratures, there are the Discrete Ordinate Method (DOM) [2] and the Finite Volume Method (FVM) [3].

In cartesian grids, the solution of the spatial coupling of the BTE can be achieved by means of an explicit calculation of the unknown field from previously calculated values in the cells that lie upstream in the solving direction. This approach, as explained by Coelho [4], is equivalent to solving a triangular system by means of a back substitution, and yields to the solution in a extremely efficient manner. More complex geometries can benefit also of this technique, using cartesian grids with a blocking off method, as shown by Chai et al. [5]. Additionally, an effort was made by Chui and Raithby [6] to extend this procedure for body fitted meshes, that has been used by Asllanaj et al. [7, 8] in their work.

This problem, known as Sweep Scheduling, has also been studied from a mathematical perspective, where the main focus lies on the proper scheduling of the tasks on a given set of processors. For instance, Pautz [9] provides a concise explanation of a parallel algorithm to solve the BTE explicitly, using unstructured meshes. The main concern in his work is to properly schedule the visiting order on a given processor, once the partitioning of the domain has been decided. On the other hand, Plimpton et al. [10] describe a parallel implementation of the explicit solution procedure mentioned earlier, along the lines of the work by Pautz. The algorithm is further refined by priorizing tasks in order to minimise the waiting cycles, which are the main factor that degrades the scalability.

Therefore, motivated by the good parallel performance of the algorithm presented by Plimpton et al. [10], we developed a different implementation of the parallel explicit solver for the BTE, using their work as a starting point. We have focussed on different aspects that may affect the performance of our algorithm, such as the way in which the tasks are ordered, or the communication strategy.

2 Parallel solution of the BTE

To solve the couplings between the different ordinates in the Boltzmann transport equation we use the source iteration procedure [2]. Thus, when solving the ordinate $\hat{s}^i$, the dependences of ϕ^i on the other ordinates ϕ^j are treated as constants (taking the values from the previous iteration), and deferred to the source term. Next, once all the ordinates have been solved, the source term is updated with the new ϕ. This procedure is repeated until the changes in ϕ are below the desired tolerance.

The directional nature of the phenomenon modelised by the BTE can be discretised in a lower triangle matrix for each direction. Thus, this systems can be solved explicitly sweeping the nodes in a particular order, performing what is known as a back substitution. Given a direction, for each pair of adjacent nodes only one of them can contribute to the flux of the other. This relation can be described by means of a directed graph, where the vertices represent the mesh nodes, and the edges (or arrows) represent the faces between two adjacent nodes, and point to the *downstream* one. This is illustrated in figure 1 for a two-dimensional unstructured mesh. Moreover, for any node, we define its *outgoing* and *ingoing* sets, which correspond to the *downstream* and *upstream* neighbours, respectively. For example, in figure 1 the outgoing set of node 4 consists on the nodes {5,9}, whereas its ingoing set is formed by nodes {1,3}.

If the dependency graph does not contain cycles, it's a directed acyclic graph (DAG). In this case, all the nodes can be evaluated explicitly if they are swept in an order in which each node comes before the nodes in its outgoing set. Such an order is referred as a topological sort of the associated DAG.

Therefore, an option to solve all the directions (once dependencies between them are deferred to the source terms) is to find a topological sort for each ordinate, and then solve one system after the other by means of back substitutions. However, in order to find a topological sort, it is necessary to have information from the whole mesh. As this not convenient with the spatial domain decomposition strategy used in this paper, we use a more indirect algorithm, without such a limitation, detailed next:

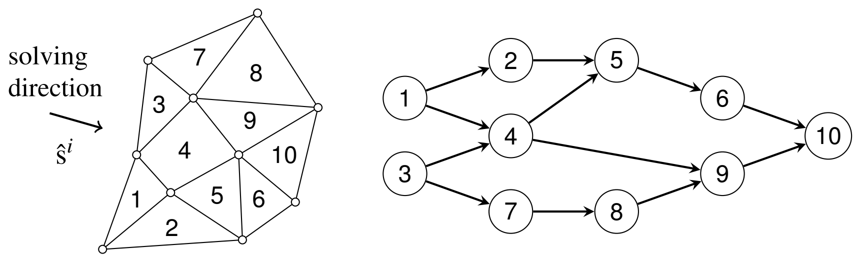

Figure 1: Dependence relations for a particular angular direction $\hat{s}^i$ on a two dimensional unstructured mesh, represented by means of a directed graph

Algorithm 1: parall sweep algorithm by directions and buffering (PSD-b) for processor p

```
1.  for each node $k$ at ordinate $i$:
2.      evaluate $count_{k,i}$ = # of ingoing nodes for $k$ at ordinate $i$
3.      if ($count_{k,i} == 0$): insert node $k$ into *doable* list $l_p^i$
4.  evaluate $work_p = m \times$# of nodes of subdomain $p$
5.  while ($work_p > 0$):
6.      for each ordinate $i$:
7.          while (tasks in $l_p^i$):
8.              remove node $k$ from $l_p^i$ and solve it
9.              decrement $work_p$
10.             for each outgoing node $k'$ of $k$:
11.                 if ($k'$ is an owned node):
12.                     decrement $count_{k',i}$
13.                     if ($count_{k',i} == 0$): add to $l_p^i$
14.                 else
15.                     $q$ = owner of node $k'$
16.                     save info for node $k'$ in BUFFER$_q$
17.     for each processor $q \neq p$:
18.         SEND BUFFER$_q$
19.     while (messages to be read):
20.         READ next pair $(k,i)$ from message
21.         for each owned outgoing node $k'$ of $k$ in ordinate $i$:
22.             decrement $count_{k',i}$
23.             if ($count_{k',i} == 0$): add to $l_p^i$
```

The above algorithm is based on the use of a list, l_p^i, of the *solvable* nodes for direction i at processor p. These are the nodes for which all the *ingoing* neighbours have already been evaluated for the ordinate i. Initially l_p^i, contains the upstream boundary nodes at direction i. For the rest of nodes there is a counter which accounts for the number of their unevaluated ingoing nodes. As the tasks of each list are being solved, the counters of their outgoing elements is decremented and some of them may become *solvable* too. This process is continued until all the nodes and directions are solved. Note that the communications are performed by means of buffers in order to reduce latency effects. This algorithm has been preferred after studding different strategies of grouping doable tasks and buffering information to be communicated.

3 Numerical experiments

Since only the resolution of the spatial discretisation is studied in this work, non scattering media ($\beta_2 = 0$) are considered. Under these conditions there are no couplings between different angular directions. The algorithm explained in the previous section is tested in 2D and 3D geometries, both discretised using unstructured meshes. We have used relatively small meshes because at most 1024 CPU were available for the tests, and we wanted to study the speedup limitations. With bigger meshes, scalability extends to higher number of CPU and the degradation of the speedup would start out of the range of processors available. The angular domain is divided in 80 ordinates. For the two dimensional geometry, the ordinates lie on the XY plane and are uniformly distributed on the $[0,2\pi)$ interval. In the 3D case the ordinates are arranged as the S_8 quadrature prescribes.

All the numerical tests presented have been carried out on the MareNostrum supercomputer at the Barcelona Supercomputing Center (BSC). At the time of writting this work, this is an IBM BladeCenter JS21 Cluster with 10240 PowerPC 970MP processors at 2.3 GHZz. Quad-core nodes with 8GB were coupled by means of a high-performance Myrinet network.

The variation of the speedup with the number of ordinate directions is depicted, in figure 2, for two and three dimensional meshes of about 150,000 elements each one. In both cases, increase the size of the angular discretization benefits the speedup. Since the matrix of each direction has a different associated DAG, increasing the number of directions, there are more chances of more processors working at the same time.

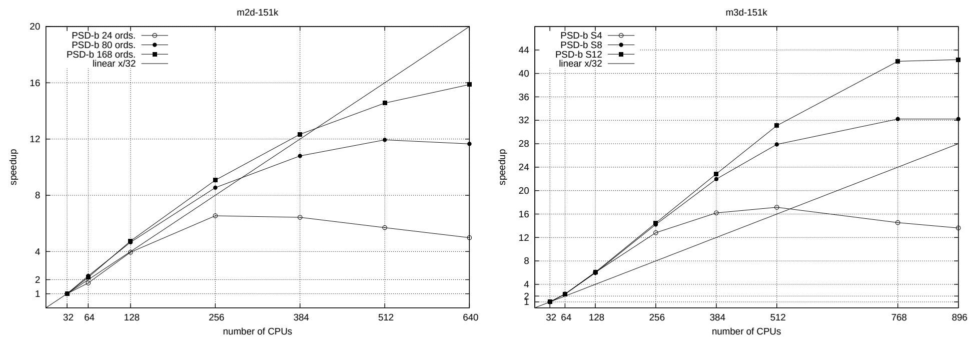

Figure 2: Variation of the speedup with the number of ordinate directions. Left: 2D mesh. Right: 3D mesh.

In figure 3, the weak scalcability is displayed for the two and three dimensional cases. In both cases we tried to keep the load per CPU constant around 1170 nodes. When using unstructured meshes it is difficult to obtain a specific number of nodes; nevertheless, the maximal deviation obtained is around 1% .

We can see that, although the size of the mesh and the number of CPUs are increased 64 times, the solution time grows a factor of 5 and 2.5 for the 2D an 3D discretizations, respectively. Similarly than for the strong speedup, the weak speedup is worse in the 2D case because the resolution is faster and the relative weight of the degradation factors becomes more important. Initially, using 16 CPU, the time for 3D case is a 61% higher (0.18 vs 0.29 sec.) but, with 1024 CPU, the times become much more similar (0.89 vs 0.73 sec.) as the 3D scalability is better.

4 Concluding remarks

A algorithm for the direct parallel solution of the spatial couplings given by the BTE is presented. The works of Pautz [9] and Plimpton et al. [10], are taken as a starting point but different strategies for grouping tasks on the *doble* list and buffering information prior to communication episodes, are adopted. Our, algorithm has been tested up to 1024 CPU with rather small meshes and promising results.

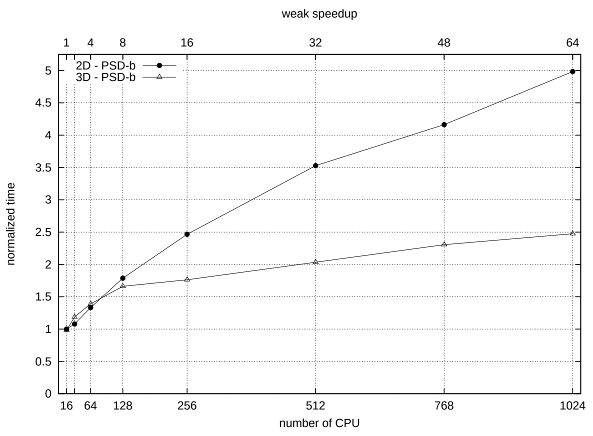

Figure 3: Weak speedup of the PSD-b algorithm, for the 2D and 3D discretizations. The number of nodes per CPU is kept constant at around 1170 elements.

5 Acknowledgements

This work has been financially supported by *Termo Fluids S.L.*, and by the Spanish *Ministerio de Educación y Ciencia* (Project references ENE2006-14247 and ENE2009-07689).

References

[1] M. Ben Salah, F. Askri, A. Jemni, and S. Ben Nasrallah. Numerical analyses of radiative heat transfer in any arbitrarily-shaped axisymmetric enclosures. *J. Quant. Spectrosc. Radiat. Transfer*, 97(3):395–414, February 2006.

[2] W. A. Fiveland. Discrete-ordinates solutions of the radiative transport equation for rectangular enclosures. *Journal of Heat Transfer*, 106:699–706, November 1984.

[3] G. D. Raithby and E. H. Chui. A finite volume method for predicting radiant heat transfer in enclosures with participating media. *Journal of Heat Transfer*, 112:415–423, 1990.

[4] P. J. Coelho. Numerical simulation of radiative heat transfer from non-gray gases in three-dimensional enclosures. *J. Quant. Spectrosc. Radiat. Transfer*, 74(3):307–328, 2002.

[5] John C. Chai, HaeOk S. Lee, and Suhas V. Patankar. Treatment of irregular geometries using a cartesian coordinates finite-volume radiation heat transfer procedure. *Numerical Heat Transfer, Part B*, 26(2):225–235, November 1994.

[6] E. H. Chui and G. D. Raithby. Computation of Radiant Heat Transfer on a Non- orthogonal Mesh Using the Finite Volume Method. *Numerical Heat Transfer, Part B*, 23(3):269–288, February 1993.

[7] F. Asllanaj, V. Feldheim, and P. Lybaert. Solution of Radiative Heat Transfer in 2-D Geometries by a Modified Finite-Volume Method Based on a Cell Vertex Scheme Using Unstructured Triangular Meshes . *Numerical Heat Transfer, Part B*, 51(2):97–119, February 2007.

[8] F. Asllanaj, G. Parent, and G. Jeandel. Transient radiation and conduction heat transfer in a gray absorbing-emitting medium applied on two-dimensional complex-shaped domains. *Numerical Heat Transfer, Part B*, 52(2):179–200, August 2007.

[9] Shawn D. Pautz. An Algorithm for Parallel S_n Sweeps on Unstructured Meshes. *Nuclear Science and Engineering*, 140:111–136, 2002.

[10] Steven J. Plimpton, Bruce Hendrickson, Shawn P. Burns, William McLendon III, and Lawrence Rauchwerger. Parallel S_n Sweeps on Unstructured Grids: Algorithms for Priorization, Grid Partitioning and Cycle Detection. *Nuclear Science and Engineering*, 150:267–283, 2005.