Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 277 (2014) 131–153
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Massively parallel grid generation on HPC systems

A. Lintermann*, S. Schlimpert, J.H. Grimmen, C. Günther, M. Meinke, W. Schröder

Institute of Aerodynamics, RWTH Aachen University, Wüllnerstr. 5a, 52062 Aachen, Germany

Received 13 November 2013; received in revised form 17 March 2014; accepted 16 April 2014
Available online 5 May 2014

Abstract

The automatic grid generation on high performance computers is a challenging task under the restriction of computational power and memory availability. The increasing demand for high grid resolutions to simulate complex flow configurations necessitates parallel grid generation on multicore machines with distributed memory. In this study, a new robust algorithm to automatically generate hierarchical Cartesian meshes on distributed multicore HPC systems with multiple levels of refinement is presented. The number of cells is only restricted by the number of available cores and memory. The algorithm efficiently realizes a computational domain decompositioning for an arbitrary number of cores based on a Hilbert curve. The grids are efficiently stored and accessed via a high capacity parallel I/O method. The efficiency of the approach is demonstrated by considering human nasal cavity and internal combustion engine flow problems.

Keywords: Parallel grid generation; Cartesian meshes; Distributed memory; Lattice-Boltzmann method; Finite-volume method; High performance computing

1. Introduction

Computational Fluid Dynamics (CFD) simulations have become an attractive method for analyzing complex flows in fundamental and applied engineering problems. Investigations are often complemented by experiments to validate the numerical results which are based on a physical model to describe the flow. This model contains assumptions on the numerical method and on the flow structure such as inviscid or viscous, laminar or turbulent flow. In other words, the mesh resolution, i.e., the links between the continuous and the discrete problem, and the prospective turbulence model play an important role. It is well known that to fully understand complex turbulent flow phenomena, Reynolds-averaged Navier–Stokes (RANS) computations are often not accurate enough and higher fidelity approaches such as Large-Eddy Simulations (LES) or Direct Numerical Simulations (DNS) are required [1]. Such high fidelity approaches may require mesh resolutions containing billions of grid cells. The generation of such grids is a challenging task, especially if structured body-fitted meshes are used, which expect intensive manual input and are in general not fully parallelizable [2,3]. Unstructured grids can be generated completely automatically but have the disadvantage that

* Corresponding author. Tel.: +49 241 80 90419.
E-mail address: A.Lintermann@aia.rwth-aachen.de (A. Lintermann).

http://dx.doi.org/10.1016/j.cma.2014.04.009
0045-7825/

cells cannot be indexed by a structured scheme. However, they are more flexible and the automatic generation process makes them attractive even for complex geometries.

In this study, we pursue the concept of unstructured Cartesian meshes, which are widely used in CFD because of the straightforward formulation of higher-order schemes on such meshes. Their generation is of low algorithmic complexity, i.e., mesh refinement can be handled by uniform cell subdivision and the structure can be efficiently stored in an octree. However, the surfaces of the cells are in general not aligned with the surface of the geometry. This problem is handled, e.g., by applying cut-cell methods [4–8] in finite-volume methods (FVMs). In lattice-Boltzmann methods (LBMs) a simple interpolated bounce-back can be used [9].

The construction of such high resolution grids consumes a large amount of memory and computational time, due to the massive amount of cells that need to be generated. A grid generation on only a single processor is not efficient and parallel algorithms need to be developed. To overcome the problems of serial grid generation, Nakahashi et al. proposed the so-called Building-Cube Method (BCM) in [10], which is based on block-structured Cartesian meshes and applies the Message Passing Interface (MPI) for parallelization on distributed memory systems [11,12]. It is used in [3,12] for the simulation of the flow over wing profiles and for data compression in large-scale aeroacoustic simulations. In this method, each process holds an initial cube, which is refined according to the distance to the geometry or predefined regions, i.e., the number of cubes is fixed by the number of processes. Each of the cubes is then further refined such that all cubes have the same number of cells. Since the mesh inside a cube is structured, cells outside the computational domain are kept. Such cells do not participate in the solution process and cause an imbalance of the computation. Furthermore, the different number of cells required to exchange information between cubes of different sizes demands an increased communication effort, e.g., in the worst case in which a cube has six neighbors on a higher refinement level a 1–24 ratio of communication cells is obtained. Burstedde et al. [13] present a method for parallel adaptive mesh refinement and coarsening, partitioning and balancing on a "forest of octrees", i.e., a collection of octrees, hierarchically constituting the total computational mesh. This method uses MPI for parallelization and is applied to generate unstructured grids of up to 5.13×10^{11} cells. The algorithm sorts the leaf nodes of the trees by a Z-curve [14] and uses a different coordinate system per process. The use of the "forest of octrees" requires to determine inter-octree neighbor relations between cells, which is performed for adjacent cell corners, cell edges, and cell faces. It operates on a macro- and on a micro-tier, i.e., on a base mesh, which constitutes the basic geometry by a predefined collection of octrees and the refined mesh within the octrees. The inter-octree connectivity on the macro-tier requires to be defined by hand or makes use of the output of hexagonal mesh generators such that it does not include a fully automatic grid generation or it depends on other software tools. The parallel meshing tool Octor [15] is, e.g., used in the simulation for earthquake modeling [16] and scales up to 2^{16} cores [17]. It generates an octree of cells representing the basic geometric shape of the geometry and decomposes the tree based on a pre-order traversal. Each process continuously refines the local cells and allocates memory for application-specific data, e.g., for the density and velocity. Neighbor links across domains are collected and exchanged level-wise and balancing is obtained by exchanging cells between processes before the final mesh is extracted.

The meshing procedure presented in this study is based on hierarchical unstructured Cartesian meshes consisting of multiple levels of refinement and uses MPI for parallelization. In contrast to the BCM, the number of processes used for the simulation is independent from the number of processes used in the grid generation. Likewise, the number of CPUs in the meshing process is independent from the number of initial cubes or cells, i.e., an arbitrary number of CPUs can be used which is only limited by the amount of available memory. Similar to the BCM in [11], a space-filling Hilbert curve [18] is used to optimally order the initial cells. On the one hand, the solution method can operate on structured meshes in the BCM and on the other hand, the overhead of unused cells outside the computational domain causes an imbalance of the computation. In the presented algorithm, removing outside cells from the octree rebalances the distribution, i.e., the imbalance is avoided. Compared to the work presented by Burstedde et al. [13], a single octree is created fully automatically by defining the fluid domain via a geometry and by inside/outside determination of the created cells. That is, there is no need to define the initial macro-tier inter-octree connectivity by hand or to use external meshing tools. This also invokes, that inter-octree neighborhood determinations are not required in our algorithm, i.e., all cells are integrated into only one octree starting at a predefined level and only direct cell neighborhood and no cell corner, cell edge, and cell face adjacency is determined. Furthermore, the cells in our grid generator reside in the same coordinate system, while in [13] the octrees can have different coordinate systems such that the orientation of neighboring octants need to be determined. Unlike the partition of the octrees based on a Z-curve in [13,15] the Hilbert curve is used to decompose the mesh. Therefore, cells are continuously refined to an initial grid, on which the decomposition