V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

PARALLEL PERFORMANCE OF ADAPTIVE ALGORITHMS WITH DYNAMIC LOAD BALANCING

S. Gepner*, J. Rokicki*, J. Majewski*

*Warsaw University of Technology, The Institute of Aeronautics and Applied Mechanics,
Nowowiejska 24, 00-665 Warsaw, Poland
sgepner@meil.pw.edu.pl, jack@meil.pw.edu.pl, jmajewski@meil.pw.edu.pl

Key words: adaptation, dynamic load balancing, mesh refinement, parallel efficiency, parallel adaptation

Abstract. *Parallelization of adaptive algorithms leads to problems with parallel efficiency. Adaptation is a method which introduces dynamic perturbations to computational environment. This in turn causes problems with proper load balance. To ensure proper efficiency of a parallel simulation it is necessary to perform load balancing whenever certain threshold of load balance is breached. In this paper authors present their approach to parallel anisotropic adaptation on unstructured meshes. High parallel efficiency of the code is maintained through the use of dynamic load balancing algorithm. Measurements of parallel efficiency of an adaptive and dynamically load balanced flow application are presented.*

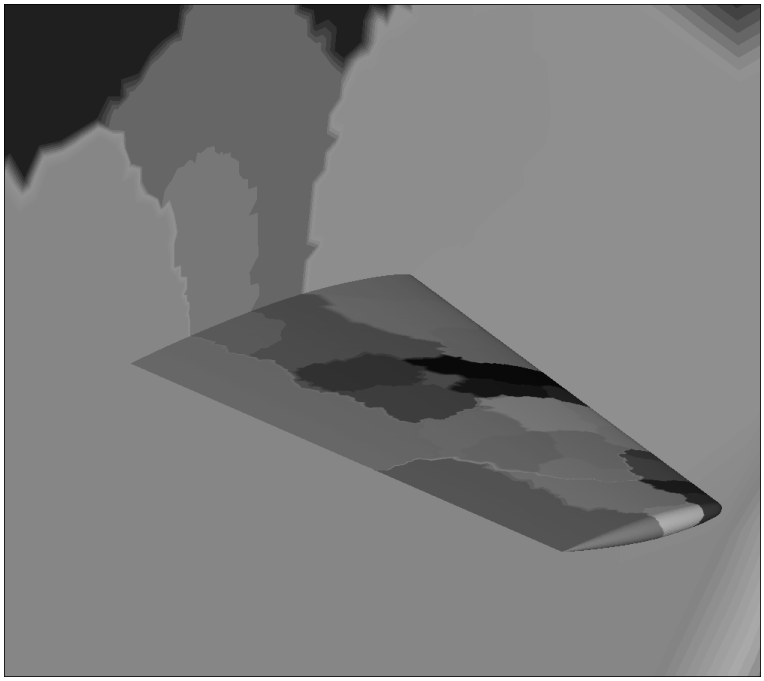

Figure 1: Initial partitioning for a 3D geometry. Onera m6 wing

1 Introduction

High efficiency of parallelization is required to converge large scale engineering problems in reasonable time. Domain decomposition is a common approach to parallelization. It requires the computational domain to be partitioned into sub-domains, prior to the actual simulation. In order to maximize simulation parallel performance, partitioning should minimize both, processor idle time and the volume of interprocessor communication. Above conditions should be satisfied throughout all of the simulation run time. There exist many partitioning methods and tools developed to fulfill the quality partitioning requirements (see reference [8, 5, 9] for a survey). Figure 1 shows a surface mesh of an Onera m6 wing test case after initial partitioning.

Adaptive techniques allow for computation of localized phenomena (boundary layers, shock waves, etc) with high resolution, while limiting the number of elements in less interesting regions. This increases the effectiveness of computations, as the necessary amount of resources is kept limited. Two alternative approaches to adaptivity are commonly used. One, based on global re meshing of the entire computational domain. Making use of an existing meshing techniques [13, 11, 12, 6, 3, 1]. Or by applying local mesh modifications to the computational mesh only in regions of interest [18, 2, 14, 17]. An example of a solution adapted mesh is shown in Fig. 2. A weak shock wave has been captured for a DLR-F6 geometry engine nacelle [12].

Adaptive techniques used in a parallel simulation inevitably lead to the perturbation of

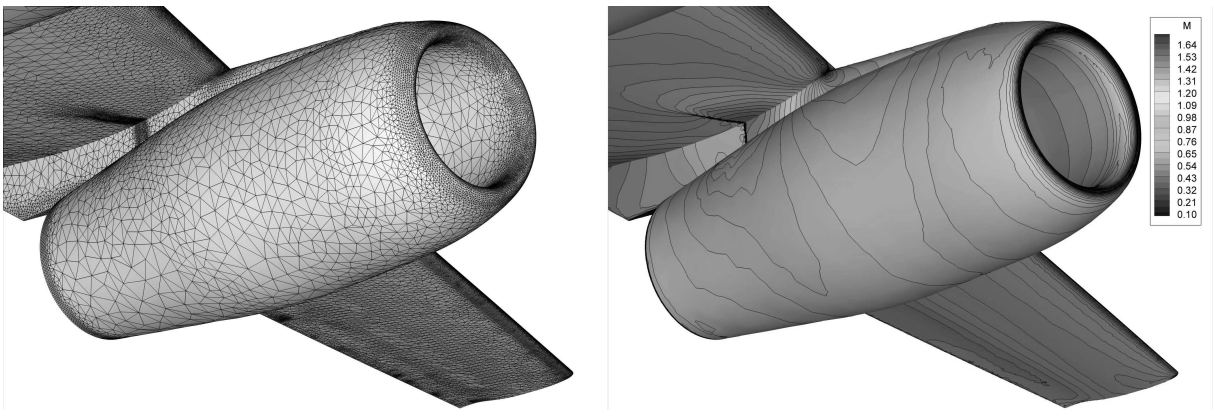

Figure 2: Surface mesh for DLR-F6 geometry (left) after a series of adaptations steps, and the resulting Mach field (right) [12].

numerical load balance. This has a negative influence on the partitioning quality. And in consequence on the overall parallel effectiveness. A load balancing algorithm must be used if adaptivity is to be employed in a parallel simulation. Use of dynamic load balancing algorithms for an adaptive application have been presented in [2, 14, 16].

In this document authors present their approach to parallel adaptivity of an unstructured mesh through local mesh refinement. An implementation of dynamical load balancing algorithm is also described. Measurements of parallel effectiveness for an adaptive and parallel application are presented. Parallel speedup and efficiency for cases with and without load balancing are demonstrated. The tests are run using the in-house RED code [13] (parallelized, based on Residual Distribution method [15, 4], explicit flow solver).

2 Parallel performance of the RED flow solver

To describe parallel effectiveness of an application it is common to use parallel speedup S_p and parallel efficiency E_p. Letting p be the number of CPU's used and T_p the time necessary to perform a parallelized task. Parallel speedup S_p is defined as:

$$S_p = \frac{T_1}{T_p} \tag{1}$$

and parallel efficiency E_p as:

$$E_p = \frac{S_p}{p} \cdot 100\% \tag{2}$$

Linear (or ideal) speedup is obtained when task performed using p processors requires time T_1/p to be completed (parallel efficiency is than equal to unity). Since speeds of memory access on a computer vary, it is possible to achieve the so called super-linear speedup by limiting the size of parallel subtasks [7]. In such a case parallel efficiency is greater than unity.

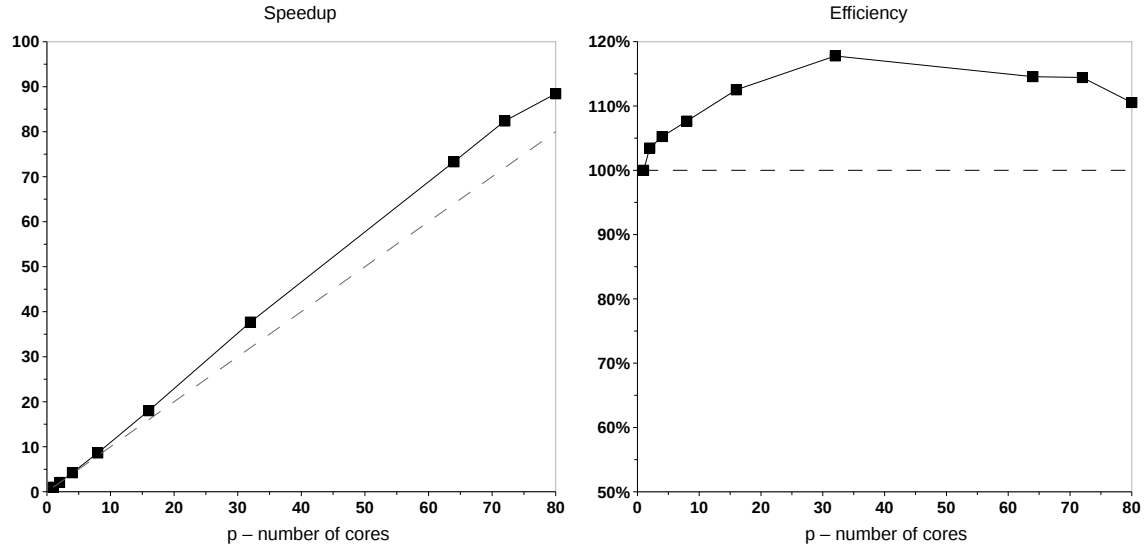

Figure 3: Parallel speedup (left) and efficiency (right) as a function of the number of computational cores used. Dashed line shows theoretical values. Squares mark computations performed on the initial not adapted mesh (super linear scaling can be observed).

To measure parallel performance of the RED flow solver mesh of 1600054 computational cells was used. The test geometry is shown in Fig. 4. Since adaptivity was not used for this case the load balancing was secured by initial partitioning.

Measurements were performed using 1, 2, 4, 8, 16, 32, 64, 72 and 80 computational cores of a computational cluster containing 20 Quad-Core AMD Opteron processors (4 cores per processor) with 2 GB of memory per core, connected by Fast Ethernet connection. Figure 3 illustrates plots of the results. Parallel speedup and efficiency are plotted against the number of computational cores used. On both graphs dashed lines represent theoretical values of parallel performance. Lines marked by squares show values obtained through the test. Reported values of parallel efficiency indicate super-linear scaling effect of the RED solver.

3 Parallel adaptive algorithm

The adaptive algorithm implemented into the RED flow solver is based on a local mesh refinement technique. Gradient based error estimator is used to mark edges of elements that should be split in order to fulfill the adaptivity criteria. Once edges are selected for refinement an appropriate cell spitting method is used for each cell. Figure 5 shows possible configuration to be used for splitting of a 2D simplex elements. For a 3D case there is a significant increase in complexity to be considered.

Since adaptation is to be performed in a parallel environment some attention has to be taken in respect to regions where domains overlap. Processes must perform communi-

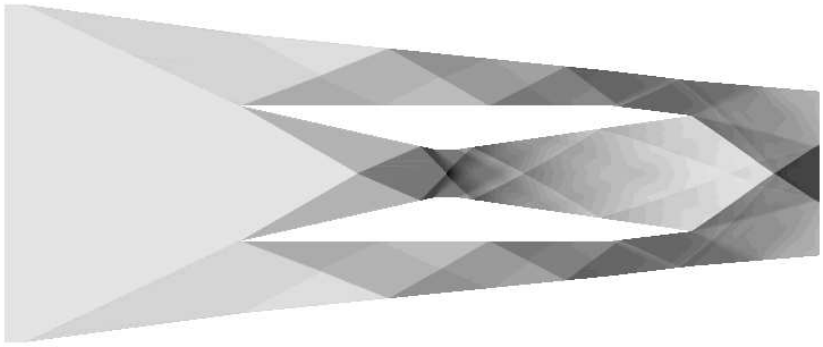

Figure 4: Resulting Mach number field for a SCRAMJET geometry (Ma = 3.0 at inlet).

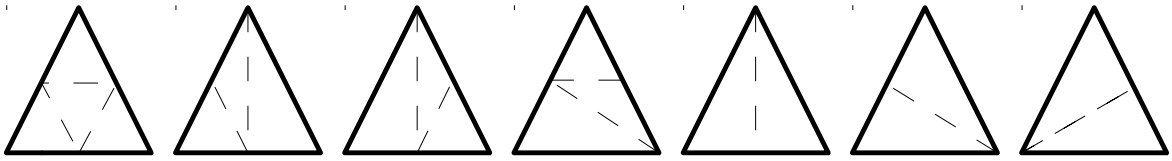

Figure 5: Possible splitting for 2D simplex elements.

cation sessions to assure coherence those zones. Communication structure of the parallel solver needs to undergo reconfiguration.

4 Influence of adaptivity on parallel performance

The influence of an adaptive method mentioned in Sec. 3 on parallel performance has been tested. Measurements were performed for the same initial geometry as used in Sec. 2 (see Fig. 4). Again measurements were performed using 1, 2, 4, 8, 16, 32, 64, 72 and 80 computational cores. A single adaptive step was taken.

Figure 6 holds the results of the measurements. Parallel speedup and efficiency are plotted as the functions of the number of computational cores being used. Again dashed lines represent theoretical values of parallel performance. Solid lines with triangles mark values recorded for the case with adaptivity used. It should be mentioned that time T_1 used as a base of comparison is different from one used in Sec. 2 (Fig. 3). In the present case numerical load is larger due to the adaptivity. Therefore results in Fig. 6 present only the influence of load imbalance on parallel performance.

One can observe that the drop in parallel efficiency, due to adaptation is substantial. Further adaptive steps would decrease the efficiency even more.

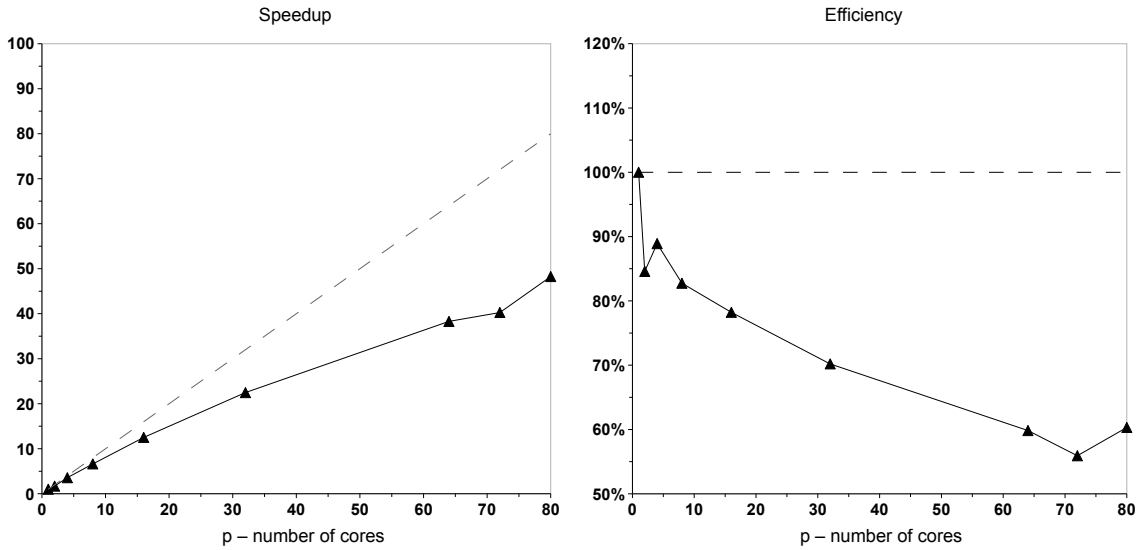

Figure 6: Parallel speedup (left) and efficiency (right) as a function of the number of computational cores used. Dashed line shows theoretical values. Triangles mark the case where adaptation was applied. Significant drop in parallel performance can be observed in comparison to results from Fig. 3

5 Load balancing algorithm

To counter the negative influence of adaptivity on parallel performance (see Sec. 4) some dynamically load balancing algorithm must be introduced. Commonly load balancing should be triggered whenever a threshold of acceptable load unbalance is breached (after the adaptation step is taken). Then it is necessary to calculate and apply a new better balanced partitioning. Subsequently the mesh entities are redistributed between processors according to the obtained coloring. In this work ParMETIS [10] graph partitioner was used to find subsequent distribution of nodes.

To perform dynamic load balancing, the following steps had to be undertaken:

1. Elements of the mesh are given unique numbering throughout the whole domain. This makes mesh elements easily distinguishable throughout the parallel environment.

2. Grid has to be described as a connectivity graph. When working with ParMETIS the so called distributed CSR format is used.

3. With graph connectivity ready it is possible to calculate new mesh partitioning (graph coloring).

4. According to the obtained partitioning mesh entities are moved between processors.

Applying the new partitioning introduces massive changes to the simulation data structure. Some mesh entities have to be send to other computing nodes while some must be

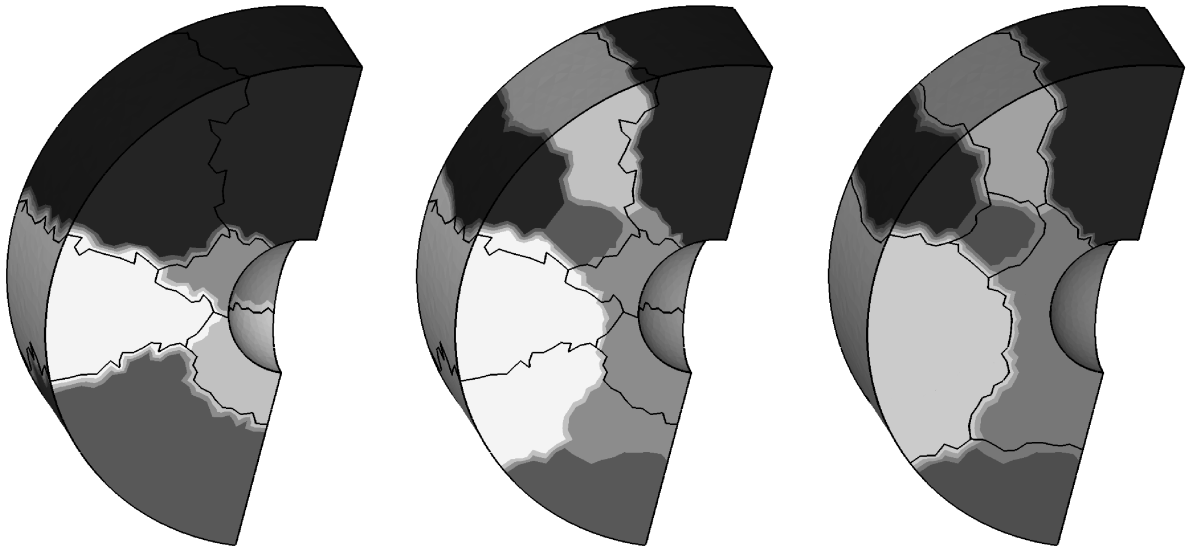

Figure 7: Partitions are being modified according to the previously calculated coloring. On the left is the original partitioning and coloring, the middle picture shows original partitioning with new coloring. On the right is the partitioning after elements have been sent / deleted.

Table 1: Repartition time as a fraction of total simulation time, for a case of 5078927 nodes (32622210 cells).

No of cores	Repart. time / Sim. time
32	0.213%
72	0.105%
80	0.099%

removed. The received data has to be added to the data structure accessible to a given CPU. This process is strongly dependent on the data structure used within the solver. Once the information is exchanged the solver is ready to restart calculations.

Repartition process is illustrated in Fig. 7 where a three dimensional domain is repartitioned. On the left the initial partitioning is shown. The middle figure presents the new coloring obtained during the graph partitioning phase plotted onto original partitioning. The rightmost picture shows the mesh after appropriate data migration.

To estimate the cost of a single re balancing step a three dimensional simulation of the flow around Onera M6 wing was selected. Mesh of 5078927 nodes (32622210 cells) was used. Results were obtained by running the RED solver on the cluster containing 20 Quad-Core AMD Opteron processors (4 cores per processor) with 2 GB of memory per core, connected by Fast Ethernet connection.

As can be observed in Table 1 numerical cost of re balancing is relatively small in comparison to the total simulation time. Table 1 shows comparison of the CPU time required to perform the repartition as a fraction of total simulation time required to

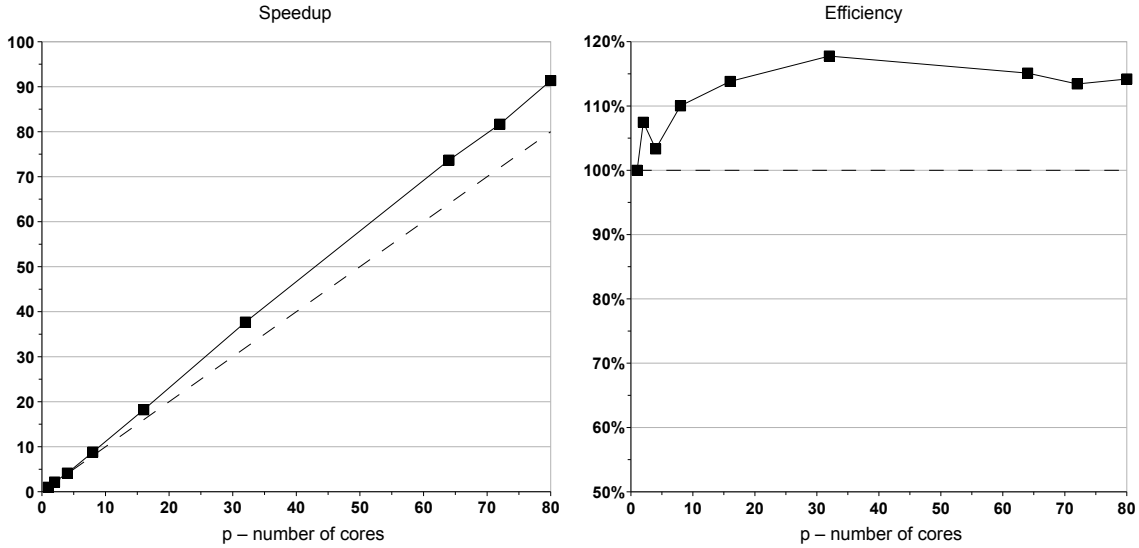

Figure 8: Parallel speedup (left) and efficiency (right) as a function of the number of computational cores used. Dashed line shows theoretical values. Solid line represents the values noted for the adapted case, where dynamical balancing was applied. Super linear scaling is again present.

converge an explicit Euler solver on a given number of processors. It is worth noticing that the time share required for re balancing decreases as the number of CPU's grows.

6 Influence of Dynamic load Balancing on parallel efficiency

As stated in Section 4 the use of adaptation has a negative impact on the parallel performance of the code. To judge the merits of applying the DLB to an adaptive flow computation such an algorithm was used and parallel performance was tested. Figure 8 shows plots of parallel performance measured for the case in which adaptation was used together with the DLB algorithm. Solid line marked by squares shows the measured values, while a dashed line shows values of the linear speedup.

It should be noted, that by taking advantage of dynamic re balancing, parallel efficiency of the code is maintained. Super linear scaling effect is again present.

7 Conclusions

Parallel performance of a Residual Distribution flow code has been presented. It was shown that application of adaptivity to a parallel simulation has a consequence in a significant drop of application's parallel performance. Further a relatively fast method of repartitioning and dynamical load balancing has been discussed. It was shown that by taking advantage of dynamical load balancing it is possible to regain high parallel efficiency of an adaptive application.

8 Acknowledgments

This work was supported by the FP6 EU ADIGMA project (Adaptive Higher-Order Variational Methods for Aerodynamic Applications in Industry), and by MNiSW, grant No: N N516 4280 33, Dynamic load balancing for adaptive engineering applications in parallel environments (Dynamiczne rownowazenie obciarzenia adaptacyjnych aplikacji inzynierskich w srodowiskach rownolegych i rozproszonych), 2007-2009.

REFERENCES

[1] Frédéric Alauzet, Paul Louis George, Bijan Mohammadi, Pascal Jean Frey, and Houman Borouchaki. Transient fixed point-based unstructured mesh adaptation. *International Journal for Numerical Methods in Fluids*, 43:729–745, 2003.

[2] Frédéric Alauzet, Xiangrong Li, E. Seegyoung Seol, and Mark S. Shephard. Parallel anisotropic 3D mesh adaptation by mesh modification. *Eng. Comput. (Lond.)*, 21(3):247–258, 2006.

[3] M. J. Castro-Daz, Frédéric Hecht, Bijan Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulation. *International Journal for Numerical Methods in Fluids*, 25:475–491, 1997.

[4] Herman Deconinck, Kurt Sermeus, and Remi Abgrall. Status of multidimensional upwind residual distribution schemes and applications in aeronautics. *AIAA Conference Proceedings*, pages 2000–2328, 2000.

[5] Karen D. Devine, Erik G. Boman, Robert T. Heaphy, Bruce A. Hendrickson, James D. Teresco, Jamal Faik, Joseph E. Flaherty, and Luis G. Gervasio. New challenges in dynamic load balancing. *Applied Numerical Mathematics*, 52(2-3):133 – 152, 2005. ADAPT '03: Conference on Adaptive Methods for Partial Differential Equations and Large-Scale Computation.

[6] Pascal Jean Frey and Frédéric Alauzet. Anisotropic mesh adaptation for transient flows simulations. In *IMR*, pages 335–348, 2003.

[7] John L. Gustafson. Fixed time, tiered memory, and superlinear speedup. 1990.

[8] Bruce Hendrickson and Karen Devine. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 184(2-4):485 – 500, 2000.

[9] Bruce Hendrickson and Tamara G. Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, 26(12):1519 – 1534, 2000. Graph Partitioning and Parallel Computing.

[10] George Karypis. Parmetis - parallel graph partitioning and fill-reducing matrix ordering.

[11] Jerzy Majewski. An anisotropic adaptation for simulation of compressible flows. *Mathematical Modelling and Analysis*, 7:127–134, 2002.

[12] Jerzy Majewski. Anisotropic adaptation for flow simulations in complex geometries. In *36th Lecture Series on Computational Fluid Dynamics / ADIGMA course on hp-adaptive and hp-multigrid methods*. von Karman Institute for Fluid Dynamics, 2009.

[13] Jerzy Majewski and Aristotelis Athanasiadis. Anisotropic solution-adaptive technique applied to simulations of steady and unsteady compressible flows. In Herman Deconinck and Erik Dick, editors, *Computational Fluid Dynamics 2006 - Proceedings of the 4th International Conference on Computational Fluid Dynamics, ICCFD4, Ghent, Belgium, July 10-14*, pages 353–359. Springer, 2006.

[14] Young Min Park and Oh Joon Kwon. A parallel unstructured dynamic mesh adaptation algorithm for 3-d unsteady flows. *International Journal for Numerical Methods in Fluids*, 48:671–690, 2005.

[15] Kurt Sermeus and Herman Deconinck. Solution of steady euler and navier-stokes equations using residual distribution schemes. In *33rd Lecture Series on Computational Fluid Dynamics - Novel Methods for Solving Convection Dominated Systems (LS2003-05)*. von Karman Institute for Fluid Dynamics, 2003.

[16] Christoph Troyer, Daniele Baraldi, Dieter Kranzlmuller, Heinz Wilkening, and Jens Volkert.

[17] Jacob Waltz. Parallel adaptive refinement for unsteady flow calculations on 3d unstructured grids. *International Journal for Numerical Methods in Fluids*, 46:37–57, 2004. 10.1002/fld.674.

[18] Z. Zhu, P. Wang, and S. Tuo. An adaptive solution of the 3-d euler equations on an unstructured grid. *Acta Mechanica*, 155:215–231, 2002. 10.1007/BF01176244.