49

Efficient Mesh Partitioning For Adaptive HP Finite Element Meshes

A. K. Patra [1] & D. W. Kim[2]

Introduction

The use of domain decomposition solvers presumes the existence of a partitioning of the domain that distributes the computational effort equitably. Adaptive *hp* finite elements which are capable of delivering solution accuracies far superior to classical $h-$ or $p-$version finite element methods, for a given discretization size [BS94] create special difficulties in generating such load balanced partitions. Two major difficulties that arise in partitioning such meshes are a) the choice of a good *a priori* measure of computational effort, which can be equidistributed among the processors and b) minimizing the data migration among processors as the mesh changes and is repartitioned. In uniform meshes using simple solvers, computational effort is directly related to the degrees of freedom in each sub-domain. In schemes using *hp* meshes, and domain decomposition solvers (e.g. preconditioned iterative substructuring [JTOF97]), the computational effort is not simply function of the degree of freedom. The distribution of polynomial orders, cost of constructing the preconditioner, connectivity have important effects that are difficult to quantify. In *hp* methods, especially when using adaptive strategies like the "Texas 3 step" [JTOF97] the mesh changes in size very rapidly (often an order of magnitude more degrees

1 Corresponding Author, Department of Mechanical Engineering, University at Buffalo, Buffalo NY 14260, email: abani@eng.buffalo.edu
2 Address same as above. Authors were supported by National Science Foundation Award # ASC-9702947, Computations were performed on NPACI computers

Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund ©1999 DDM.org

of freedom in two solution cycles). Repartitioning to balance the load again usually results in migration of large data sets.

We introduce here a new measure of computational effort and load balancing strategy designed to mitigate these problems. The load measure introduced is error on a coarser mesh. We will study its use for several complex adaptive *hp* meshes and compare its effectivity to the degrees of freedom(DOF) normally used. Our use of the error on a coarse mesh as a measure of computational effort has motivated a simple strategy whereby we repartition before we refine/enrich the mesh. Thus new elements are created on the processors on which they will eventually be used greatly alleviating the data migration problem. Again, we will test this strategy on several complex meshes and compare it to the traditional repartitioning after refinement. Such a strategy will greatly reduce the need for complex dynamic rebalancing procedures.

Mesh Partitioning And Partitioners

The mesh partitioning problem may be viewed as a weighted graph partitioning problem with either the nodes/elements representing vertices of a graph and the edges being either the edges of the finite element mesh or the element connectivity respectively. Vertex and edge weights can be assigned based on the computational effort associated with different nodes and the the need for communication between them. The abstract problem may then be stated as:

Given a graph G with n weighted vertices and m weighted edges, partition the vertices into p sets in such a way that the sum of vertex weights is equidistributed and that of edges crossing between sets is minimized.

This is a problem that is provably "NP hard" and hence solved with heuristics [KK97]. Several tools are available to implement the partitioning once the weighted graph is defined. The primary tools we use here are the Metis (Karypis and Kumar [KK97]) and Chaco packages (Hendrickson et al.[HL93]. These tools are based on multilevel partitioning ideas. At the coarsest level of the graph the Fiedler vector is used to define an ordering of the nodes of the graph and this is then recursively bisected. While broadly similar in their underlying ideas, Chaco and Metis differ significantly in their implementation of these ideas. Another partitioner we make use of in these studies is based on using space filling curves to order the nodes/elements of the finite element mesh (Patra and Oden [OP95]). These curves are mappings to n dimensional unit hypercubes ($[0,1] \times [0,1] ... \times [0,1]$) from the unit interval $[0,1]$. The inverse of these mappings defines an ordering of the elements which can then be used in a k-way split or a recursive bisection to obtain the partitioning.

For adaptive meshes in which the mesh changes as the computation procedes the partitioning must also change to reflect the changing grid. The changing of partitioning causes data to be migrated to reflect the new partitioning. This migration can be a serious bottleneck in obtaining good parallel efficiencies. We can now define the three goals of partitioning such meshes as obtaining a good balance of computational effort, minimizing communications among processors and data migration as the mesh and partitioning change. While the basic mesh partitioning problem has been explored by many researchers, the problem of minimizing data migration when dealing with changing meshes(dynamic graphs) has not received much attention. Together, with the problem of estimating computational effort in *hp* meshes we thus have a challenging task in formulating appropriate algorithms and strategies for partitioning adaptive *hp* meshes.

A priori Measures Of Computational Effort

Load Measures

Any problem decomposition strategy for efficient parallel computing requires an estimate of computational effort on the different tasks that are a part of the computation. In mesh partitioning for data parallel computations on finite element meshes, this translates to a measure of computational effort linked to the elements/nodes that comprise the mesh. This implies that a usable measure of computational effort must also posses the local additivity property. The total load measure of should be the sum of the load measure associated with each element.

Difficulties With DOF based measures

In simpler uniform or h-adaptive meshes and solution strategies using direct solvers, the number of simultaneous equations to be solved also known as degrees of freedom (DOF) in the model is an efficient measure of computational effort in the problem. This follows from well known results relating the operation count in the equation solver to O(N^α) where α is determined by the choice of solver and the sparsity of the matrix, and N is the number of unknowns. This has motivated the traditional choice of DOF as a computational load measure. Communication costs are usually measured by counting the number of unknowns on the interfaces generated by a partitioning of the mesh.

However, when using higher order elements and iterative solvers (preconditioned conjugate gradient type in our applications [JTOF97]), such a simple and straightforward count of the number of unknowns is not adequate. The spatial distribution of computational effort for these meshes is highly non-uniform. The element stiffness matrix corresponding to a bilinear quadrilateral element for solving the equations of elasticity in two dimensions comprises of only 8 DOF whereas the same matrix for a seventh order element has 64 DOF! In adaptive *hp* meshes where the mix of element orders is highly non-homogeneous and determined at run time the non-uniformity of the distribution of computational effort can thus be very severe.

We also note that degrees of freedom associated with higher orders do not involve the same amount of computation as those associated with lower orders especially for the degrees of freedom internal to an element as these are usually eliminated at the element level using a substructuring technique. The cost associated with other degrees of freedom like those on the side of the element is also significantly lower as they are shared by only two elements. The connectivity (number of other DOF that any DOF interacts with) thus has great influence on the computation cost. The matrix sparsity is also dictated by this and thus this has great implications for storage and memory requirements. The conditioning of the system (usually defined as the ratio of the maximum and minimum eigen values of the system) will also influence the computational effort greatly since it controls the number of iterations required for an iterative solver to converge. While both connectivity and conditioning appear to be good measures of computational effort they fail the local additivity test. Local connectivities do not add up. Conditioning cannot even be defined at the local level. Thus neither of these may be used as a suitable primary computational measure.

Numerical Error Based Measure

In the solution adaptive meshing process distribution of h and p is determined by estimates of the error distribution in a previous coarser mesh [OP95]. It is thus reasonable to assume that the local distribution of computational effort on a refined/enriched mesh would also be determined by the error distribution in a previous mesh. A variety of such error estimates are widely available now in the research literature. For our study here we use an error estimator of the type introduced by Bank and Smith [BS93] and further developed by Patra and Oden[PO97]. In this type of error estimator the coarse mesh problem is used to define a local boundary value problem on each element using the underlying operator and a hierarchical discretization. For example if the mesh uses quadratic elements then the local problem for the error in each element uses the additional shape functions corresponding to a cubic element. We now propose that the **square of the norm of the error in a coarser mesh** be used as a measure of computational effort for an adapted mesh and compare this with the more classical degrees of freedom in the mesh (DOF) measure. Communication costs will be factored into either measure by minimizing the number of DOF on the interfaces.

Use of Load Measures and Weighted Graph Partitioning

The weighted graph partitioners (Metis and Chaco) can account for different computational measures through the use of edge and vertex weights for the graph. The partitioners then attempt to equidistribute the weights among the different partitioners while keeping the weights associated with the edges cut by the partitioning minimum. The conventional weighting schemes are all designed for h adaptive meshes. We introduce here a modified weighting scheme for hp adaptive meshes. Since each element in the mesh constitutes a node in the graph either the square of the norm of the error or the square of the polynomial order associated with the element is used as the vertex weight. The purpose of the edge weight is to minimize the number of unknowns on the inter-partition boundary thus minimizing communications. For the edge weight we thus use the higher polynomial order of the two elements that the edge connects. This provides a good edge weight since the number of unknowns on the inter-partition boundary is governed by the number of elements on the boundary and their polynomial orders.

Partitioning and Repartitioning Strategies

Basic Parallel Adaptive Algorithm

This type of error based problem decomposition can be easily embedded in an overall adaptive algorithm using for example the "Texas-three step" *hp* adaptive strategy developed by Oden and Patra [OP95]. In this strategy the adaptive process is started using a uniform mesh. Errors, convergence rates and other constants are then estimated on this simple mesh. *a priori* estimates of the form $||e||_1 \leq C\frac{h^\mu}{p^\nu}||u||_1$ where $||e||_1$ is the norm of the error, C is a constant, μ and ν are convergence rates, and $||u||_1$ is the H^1 norm of the solution, are then used to predict the level of mesh modification required for a target error, first using only h refinements (the second step of three), and then using p enrichment (the third step of three). When performed in a parallel environment each step of mesh modification is followed by a repartitioning of the mesh to balance the loads again. We call this strategy **Partition After Refinement (PAR)**. This type of repartitioning from scratch often results in high data migration that can cripple the computations as repartitioning takes place. The use of the coarse mesh error as a partitioning measure motivates an alternate strategy that promises to greatly reduce data migration.

Partition Before Refinement (PBR)

In this strategy we take advantage of our *a priori* knowledge of the computational effort in the next mesh via the error estimate to compute a repartitioning before modifying the mesh. The new/modified elements are then generated on the processor that will use them – thus greatly reducing mesh migration. It is obvious that the cost of moving a few parent elements will be significantly less than migrating a host of child elements. We name this strategy **Partition Before Refinement (PBR)**.

The above parallel adaptive process can be modified to incorporate this strategy. After the first step the mesh can be repartitioned and redistributed among the processors based on the error in the solution. Then, the mesh modification is carried out. Again following the solution of this step the mesh is repartitioned and redistributed among the processors based on the error in the solution. We have implemented both of the load measures and partitioning strategies outlined here in an MPI (message passing interface) based parallel *hp* adaptive code. Results and conclusions will be discussed in the next section.

Results and Discussion

Measures of partition quality

We define two quantities from [PO95] to measure the quality of partitioning;

Imbalance Fraction(IF): This is designed to measure load imbalance and is defined as:. $IF_i = \frac{N_i - N_{av}}{N_{max}}$ where N_i is the DOF associated with the subdomain i, N_{av} and N_{max} are the average and maximum DOF associated with the different subdomains.

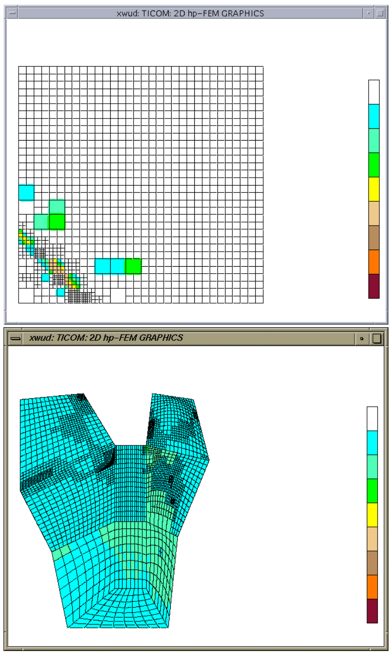

xwud: TICOM: 2D hp-FEM GRAPHICS
xwud: TICOM: 2D hp-FEM GRAPHICS

Figure 1 Test Mesh1 and 4

Shape Fraction(SF): This is defined as the percentage of interface DOF in a particular partition with respect to the total DOF associated that partition. For the whole domain all DOF on interfaces are compared to the total DOF in the problem. This is a measure of interprocessor communication. $SF = \frac{N_{int}}{N_i}$ where N_{int} is the number of DOF on all the interfaces associated with subdomain i.

Test Problem

We used the model problem of heat conduction with a variety of geometries and boundaries to test our problem. This choice of test problem helps keep the solution to the problem used in this study simple and allows us to focus on the partitioning. Four adaptive *hp* meshes are used this study (See Fig. 1 for samples) They have different sizes ranging from 2364 to 11242 degrees of freedom. We use all the partitioners and partitioning strategies described in the previous sections.

Error Based Partitioning

We have used error in a coarser mesh to partition the grid. New elements inherit error from parent elements. Figure 2 show the corresponding mesh partitions for mesh 4. We recorded the 2 measures IF and SF defined previously for all meshes. We average their measures using different partitioners and present the results in Figure 3.

We observe here some general trends when partitioning the above test meshes using the strategies described above. We note that the shape fractions of the partitions

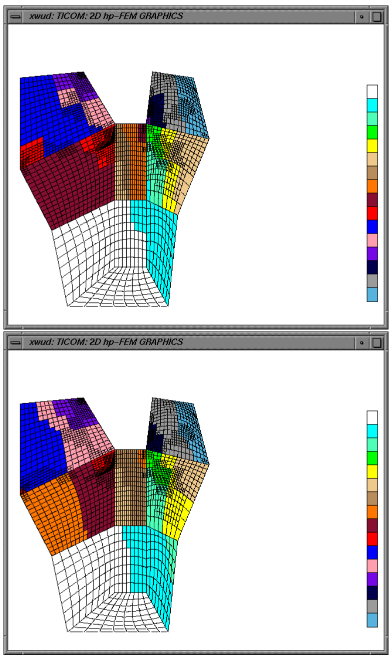
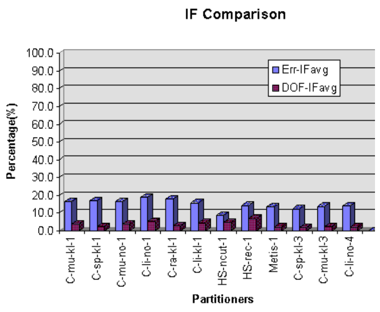

xwud: TICOM: 2D hp-FEM GRAPHICS
xwud: TICOM: 2D hp-FEM GRAPHICS

Figure 2 Error Based Partition & DOF Based Partition Using Linear Method and No local Refinement on Test Mesh4

when using error as the measure are comparable (usually within 10-15%) with those obtained from using the traditional DOF. The imbalance fractions when using error are somewhat higher but still usually within 20%. This can be partially explained by the limitations of the adaptive algorithm which can only refine 2 levels i.e. one element may have no more than 16 children. However, on problems with singularties, the error in some element may be larger by 2 orders of magnitude. In this case, the error would not be a very good predictor of the mesh refinement and hence a bad measure of computational load too. Further, we note that both IF and SF are DOF based measures and hence are limited measures of partitioning effort. Again, as we show below – error based partitioning will enable us to greatly reduce data migration during repartitioning.

Repartitioning and Data Migration

In our studies here, each run includes 3 steps of partitioning corresponding to the 3 steps of the adaptive algorithm. We test the PBR and PAR strategies on four test meshes and measure number of elements that have to migrate between processors. Table 1 shows the number of elements that migrated using different partitioners for meshes 1 and 4. We observe drastic reductions in the data migration when using the PBR based strategy as opposed to the classical PAR.In phase 1 (repartitioning after the *h*-refinement stage), we observe that PBR strategy requires 93% less data migration than PAR for our test meshes. In phase 2(repartitioning after the *hp*



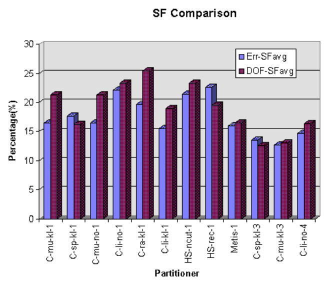


Figure 3 Average imbalance and shape fractions for different partitioning algorithms. Data averaged over 4 test meshes. Different partitioning algorithsm on abscissa.(e.g.C-mu-kl-1:Chaco, Multi-level, Kernigan-Lin on Test Mesh 1, HS-ncut-1: Hilbert space filling curve, k-way partition on Mesh 1)

–	–	PBR	PAR	PBR	PAR
Partitioner	Mesh	P1-P2	P1-P2	P2-P3	P2-P3
Ch-mu-no	1	253	345	400	1221
Ch-sp-kl	1	246	427	422	1261
Ch-ra-kl	1	210	335	284	1074
Ch-li-kl	1	218	349	291	733
Ch-mu-kl	1	253	345	400	1221
Ch-sp-no	1	240	412	432	1264
AVG	1	236.7	368.8	371.5	1129.0
Ch-mu-kl	4	570	1278	1278	2621
Ch-sp-kl	4	572	879	1263	2901
Ch-sp-no	4	576	1250	1242	2901
Ch-mu-no	4	566	1228	1278	2745
AVG	4	571.0	1156.8	1267.3	2792.0

Table 1 Comparison between PBR and PAR in Mesh 4. Rows show results for different choices of partitioning algorithms.(e.g. Ch-sp-kl: Chaco, spectral, with Kernighan-Lin)

refinement), this advantage goes up to 171% . Repartitioning data before refinement can thus greatly reduce the data migration. In conclusion, we can thus state that a) we have introduced a new measure of computational effort namely, the error in a coarser mesh, and, b) our use of the error in a corse mesh as a measure of computational effort has motivated a novel strategy whereby we repartition before we refine/enrich the mesh.

REFERENCES

[BS93] Bank R. E. and Smith R. K. (July 1993) *A Posteriori* error estimates based on hierarchical bases. Technical report, Institute for Mathematics and its Applications.

[BS94] Babuska I. and Suri M. (December 1994) The p and $h-p$ versions of the finite element method, basic principles and properties. *SIAM Review* 36(4).

[HL93] Hendrickson B. and Leland R. (1993) A multilevel algorithm for partitioning for graphs. Technical Report SAND93-1301, Sandia National Laboratories.

[JTOF97] J. T. Oden A. P. and Feng Y. S. (December 1997) Parallel domain decomposition solver for adaptive *hp* finite element methods. *SIAM Journal on Numerical Analysis* pages 2090–2118.

[KK97] Karypis G. and Kumar V. (1997) Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing* .

[OP95] Oden J. T. and Patra A. (1995) A parallel adaptive strategy for *hp* finite element computations. *Comput. Methods. Appl. Mech. and Engg.* 121.

[PO95] Patra A. and Oden J. T. (1995) Problem decomposition for adaptive *hp* finite element methods. *Computing Systems in Engineering* 6(2): 97–109.

[PO97] Patra A. and Oden J. (1997) Computational techniques for adaptive *hp* finite element methods. *Finite Elements in Analysis and Design* 25: 27–39.