Parallel implementation of recursive spectral bisection on the Connection Machine CM-5 system

Parallel implementation of recursive spectral bisection on the Connection Machine CM-5 system

Zdeněk Johan
Kapil K. Mathur
S. Lennart Johnsson
Thomas J.R. Hughes

TR-07-94

April 1994

Parallel Computing Research Group

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts

To appear in *Proceedings of Parallel CFD '93*.

Parallel implementation of recursive spectral bisection on the Connection Machine CM-5 system

Zdeněk Johan[a], Kapil K. Mathur[a], S. Lennart Johnsson[a] and Thomas J.R. Hughes[b]

[a]Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142–1264, USA

[b]Division of Applied Mechanics, Stanford University, Stanford, CA 94305–4040, USA

Submitted to the proceedings of the Parallel CFD'93 conference

1. INTRODUCTION

The recursive spectral bisection (RSB) algorithm was proposed by Pothen *et al.* [1] as the basis for computing small vertex separators for sparse matrices. Simon [2] applied this algorithm to mesh decomposition and showed that spectral bisection compared favorably with other decomposition techniques. Since then, the RSB algorithm has been widely accepted in the scientific community because of its robustness and its consistency in the high-quality partitionings it generates. The major drawback of the RSB algorithm is its high computing cost, as noted in [2], caused by the need for solving a series of eigenvalue problems. It is often stated that an unstructured mesh can be decomposed after it is generated, and the decomposition reused for the different calculations performed on that mesh. However, a new partitioning is to be obtained if adaptive mesh refinement is required. The mesh also has to be re-decomposed if the number of processing nodes available to the user changes between two calculations. In order to avoid the mesh decomposition from becoming a significant computational bottleneck, an efficient data-parallel implementation of the RSB algorithm using the CM Fortran language [3] is developed. In this paper, we present only an abbreviated description of the parallel implementation of the RSB algorithm, followed by two decomposition examples. Details of the implementation can be found in [4].

2. PARALLEL RECURSIVE SPECTRAL BISECTION

The first step is to define a graph representation of the mesh topology. This is done through the ***dual mesh connectivity*** array `idual` of dimension $n_{\text{faces}} \times n_{\text{el}}$ which contains the list of elements sharing a face with a given element. n_{faces} is the number of element faces (e.g., $n_{\text{faces}} = 4$ for a tetrahedron and $n_{\text{faces}} = 6$ for a brick); and n_{el} is the number of elements. An element having a face on the mesh boundary has its corresponding entry in `idual` set to zero. In this representation, the elements become the graph vertices and the internal faces correspond to the graph edges. The purpose of the RSB algorithm is to generate a reordering of the elements based on `idual` such that nicely shaped

partitions of adjacent elements are obtained. These partitions are then mapped to the vector units of the CM-5 system, with the constraint of having at most one partition per vector unit. The partitioning procedure follows exactly the array ***block distribution*** format used by the CM-5 run-time system. In this format, all partitions contain the same number of elements except the last one which has whatever elements remain. It should be noted that our parallel implementation of the RSB algorithm is tightly linked to the data mapping format just described. Major changes to the implementation would be required if another mapping format was used. Since current CM-5 configurations have power-of-two numbers of vector units, the RSB algorithm is based on an iterative partitioning process which decomposes the whole mesh into 2 partitions, each of which in turn is decomposed into 2 partitions, and so on. The number of iterations in the recursive process is therefore $\log_2(n_{\mathrm{vu}})$, n_{vu} being the number of vector units in the CM-5 configuration considered.

The implementation of the algorithm is done such that all elements of the mesh are treated in parallel. It implies a two-level parallelization; one level on the partitions generated at a given stage of the recursive process and the other on the elements in each partition. One should note that there is no performance loss during the recursive process since the CM-5 system always processes the same number of data, namely the number of elements in the whole mesh.

The array `idual` is used to evaluate the Laplacian matrix $\boldsymbol{L}$, defined as

$$L_{ij} = \begin{cases} -1, & \text{if elements } i \text{ and } j \text{ share a face;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$L_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{n_{\mathrm{el}}} L_{ij} \tag{2}$$

$\boldsymbol{L}$ is a positive semi-definite matrix. It can be easily shown that the eigenvector associated with the zero eigenvalue is $\boldsymbol{e} = \{1, 1, \ldots, 1\}^T$. The zero eigenvalue has a multiplicity equal to the number of connected element blocks in the mesh (or in the considered partition obtained at a given stage of the recursive process). By definition, a partition is said to be connected if the graph defined by `idual` for that partition is connected. The properties of the smallest non-zero eigenvalue and its associated eigenvector $\boldsymbol{f}$ have been studied by Fiedler in the framework of graph theory [5, 6, 7]. He has shown that reordering the components of $\boldsymbol{f}$ provides a reordering of the elements in the mesh (or in the corresponding partition). The reordered list of elements is then split as desired. The vector $\boldsymbol{f}$ will be referred to as the ***Fiedler vector***.

In the data-parallel implementation of the RSB algorithm, the smallest non-zero eigenvalue and the Fiedler vector are evaluated using a modified version of the Lanczos algorithm: The unnecessary computation of the zero eigenvalue is avoided by orthogonalizing all Lanczos vectors against $\boldsymbol{e}$, the eigenvector corresponding to the zero eigenvalue. Moreover, the smallest non-zero eigenvalues of the tridiagonal matrices generated by the Lanczos algorithm are computed using a modified method of bisection. The complete parallel RSB algorithm can be summarized as follows:

For $n = 1, \ldots, \log_2(n_{\mathrm{vu}})$
    Identify the connected element blocks (see Section 2.1)

Calculate the Fiedler vector $\boldsymbol{f}$ (see Section 2.2)
Reorder `idual` based on the ranking of the components of $\boldsymbol{f}$
Set `idual` entry to 0 for elements having a neighbor in a different partition

The following sections describe in greater details the important issues arising in the implementation of the parallel RSB algorithm on the CM-5 system.

2.1. Identification of connected element blocks

Extending Fiedler's work to mesh partitioning shows that a connected partition is guaranteed to be decomposed into two connected subdomains only if the reordered components of the Fiedler vector are split according to their sign, i.e., negative components are associated with the first subdomain and positive components with the second subdomain. Unfortunately, the block distribution format imposes a split which may not yield connected partitions. If such a case occurs, the orthogonalization of the Lanczos vectors against $\boldsymbol{e}$ has to be done for each connected block of elements independently of the others. We therefore have to design a coloring algorithm which identifies the connected element blocks. In this algorithm, one element in each partition sends its color (initially set to 1) to its neighbors, which in turn send the color to their neighbors, and so on. The color is incremented and the algorithm restarted if there are no non-colored neighbors left. The algorithm terminates when all elements are assigned a color. The low latency of the data network makes the CM-5 system suitable for such algorithms, especially in the initial phase of the iterative process where only a small number of data are sent through the network.

2.2. Lanczos algorithm

The Lanczos algorithm used to compute the Fiedler vector contains several computationally intensive operations executed at each Lanczos iteration:

1. Three dot-product operations for each partition;
2. One matrix-vector product of the form $\boldsymbol{u} = \boldsymbol{L}\,\boldsymbol{v}$; and
3. Computation of the smallest non-zero eigenvalues of tridiagonal matrices using a modified method of bisection [4]. This part has been implemented in a macro-assembly language.

A description of the implementation of dot-products and matrix-vector products on the CM-5 system follows.

2.2.1. Concurrent dot-products

In the case all partitions are connected, dot-products for each partition are performed in the following steps:

1. Compute the inner product on each vector unit using BLAS kernels available in the Connection Machine Scientific Software Library (CMSSL) [8].
2. Use active messages [9] to perform the reductions between all vector units within each partition and to spread the inner product results back to the vector units.

If some partitions are not connected, the orthogonalization of the Lanczos vectors against $\boldsymbol{e}$ is performed using segmented scan operations [10].

2.2.2. Matrix-vector products

The most expensive operations in the evaluation of the Fiedler vector are the matrix-vector products of the form $\boldsymbol{u} = \boldsymbol{L}\,\boldsymbol{v}$. They have to be handled with special care to achieve good performance. The matrix-vector product $\boldsymbol{u} = \boldsymbol{L}\,\boldsymbol{v}$ can be decomposed into two parts:

$$u_k = L_{kk} v_k + \sum_{\substack{l=1 \\ l \neq k}}^{n_{\mathrm{el}}} L_{kl} v_l \qquad k = 1, \ldots, n_{\mathrm{el}} \tag{3}$$

The first term is simply a component-wise product between the vectors diag($\boldsymbol{L}$) and $\boldsymbol{v}$ and does not require any communication between processing nodes. Since $L_{kl} = 0$ or -1 for all $k \neq l$, the second term is actually a scatter operation. It can be rewritten

$$\sum_{\substack{l=1 \\ l \neq k}}^{n_{\mathrm{el}}} L_{kl} v_l = - \sum_{\substack{l=1 \\ \mathtt{idual}(l,k) \neq 0}}^{n_{\mathrm{faces}}} v_{\mathtt{idual}(l,k)} \qquad k = 1, \ldots, n_{\mathrm{el}} \tag{4}$$

The scatter operation is achieved through the communication primitive `sparse_util_scatter` available in the CMSSL. A mask is passed to this routine to allow the scatter only for the non-zero entries of `idual`.

3. MESH DECOMPOSITION EXAMPLES

We present two numerical examples which demonstrate the performance of the parallel RSB algorithm and the quality of the partitions it generates. The mesh decomposition program was compiled with CMF 2.1 and was run on timeshared CM-5 systems. The CM-5 systems were running the Connection Machine operating system CMOST 7.2. The Lanczos tolerance was set to 10^{-3}. All computations were performed in 64-bit arithmetic. All reported timings correspond to *elapsed times*.

3.1. Mesh around an ONERA M6 wing

This first partitioning example uses a tetrahedral mesh composed of 48,011 nodes and 266,556 elements. The graph representation of the mesh has 527,966 edges. Figure 1 presents a view of the surface mesh on the outer boundaries of the domain. One can see the high concentration of boundary elements on the plane of symmetry near the root of the wing. This mesh was partitioned on a 64-processing node CM-5 system equipped with 256 vector units. A decomposition into 16 subdomains is depicted in Figure 2.

Figure 3 shows the cost of the parallel RSB algorithm as the bisection procedure progresses. The sub-$O(\log_2(\text{no. of partitions}))$ cost is due to the combined effects of the two-level parallelization of the algorithm (see Section 2) and of the decrease in the number of Lanczos iterations as the bisection procedure progresses.

The total cost of partitioning the mesh into 256 subdomains is 76 seconds. At this level of partitioning, there are 57,063 cuts in the graph, which represents 10.8% of the total number of graph edges. Table 1 shows the computing costs of the different parts of the RSB algorithm. The computation of the Fiedler vector using the Lanczos algorithm dominates with almost 80% of the total time. A more detailed cost analysis of the Lanczos algorithm is presented in Table 2. One can deduct from these two tables that about 80% of the total time is spent in communication between processing nodes (the communication-dominated portions of the code are the identification of connected blocks, matrix-vector products, and data ranking and reordering). Nonetheless, the parallel RSB algorithm exhibits good performance on the CM-5 system.

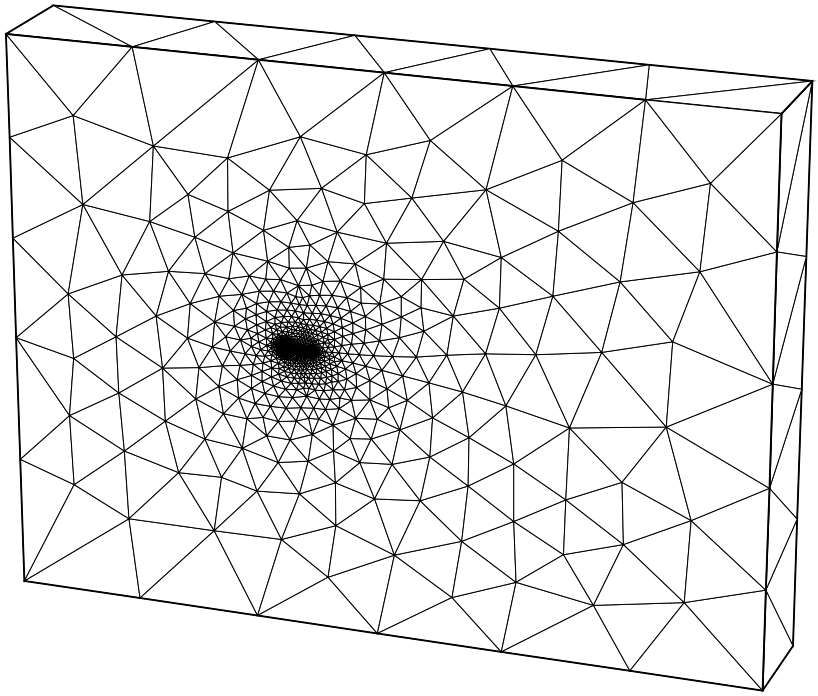

Figure 1. M6 wing. View of surface mesh on outer boundaries.

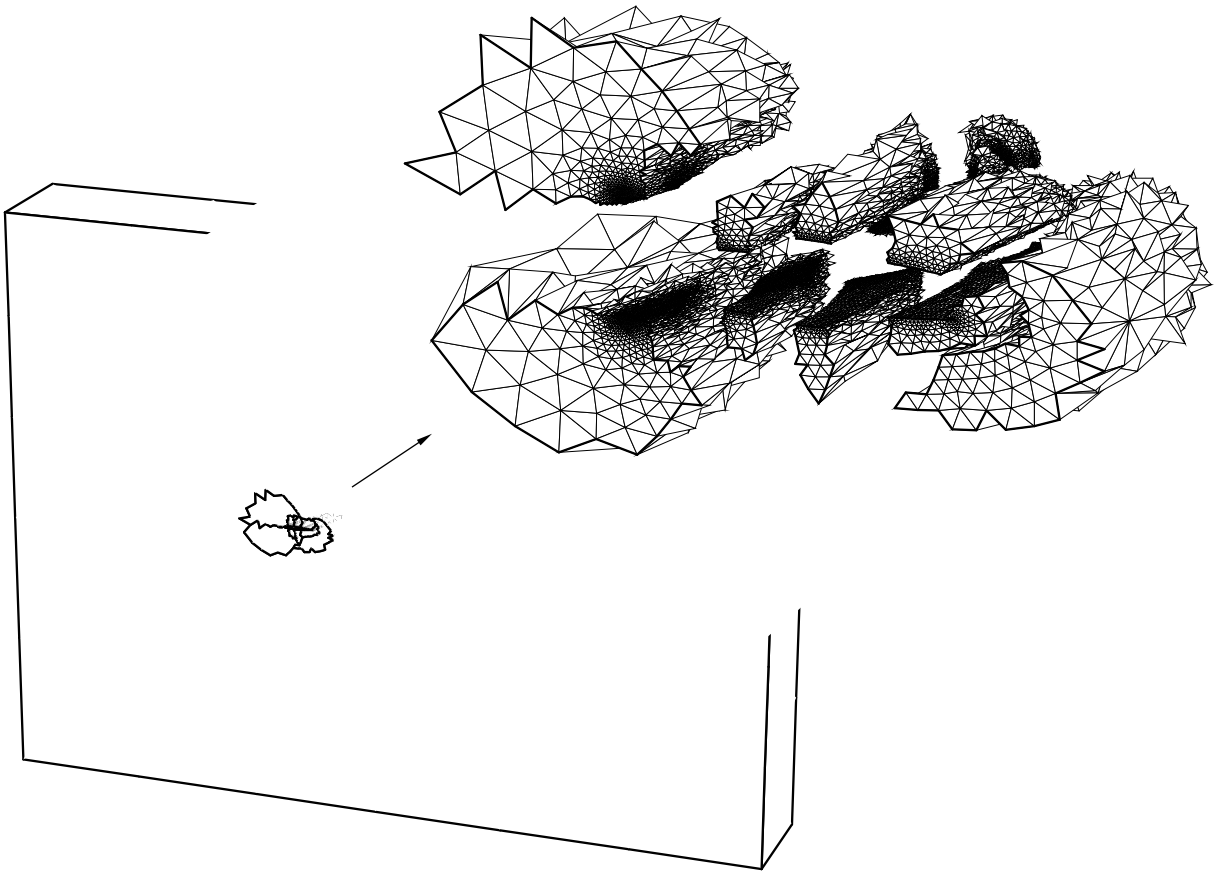

Figure 2. M6 wing. Decomposition into 16 subdomains.

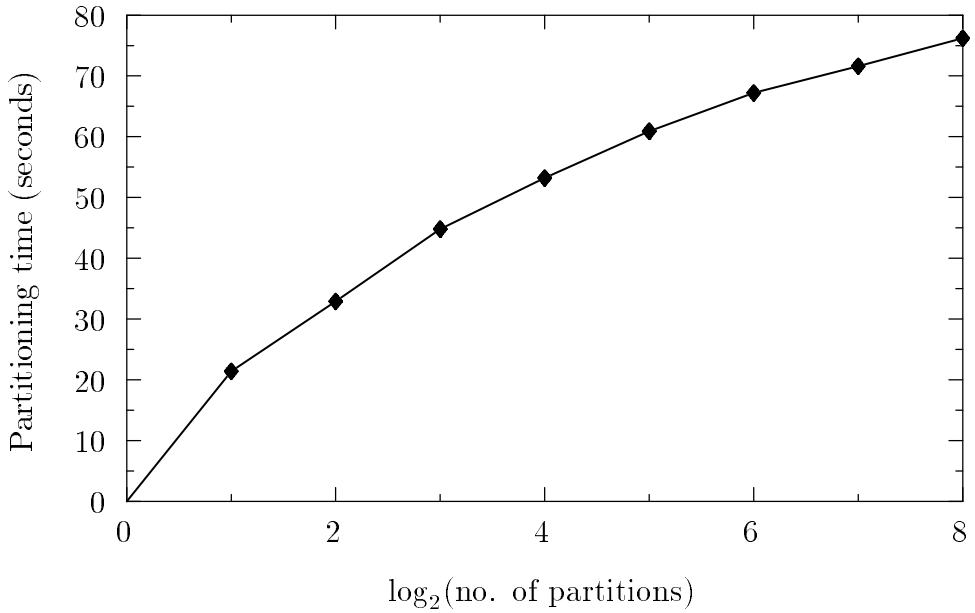

Figure 3. M6 wing. Partitioning cost as a function of recursive bisection on a 64-node CM-5 system.

Table 1. M6 wing. Elapsed times for different parts of the RSB algorithm for a partitioning into 256 subdomains on a 64-node CM-5 system.

	Timings	Percentage
ident. of connected blocks	10.2 s	13.4%
comp. of Fiedler vector	59.8 s	78.5%
data ranking/reordering	3.5 s	4.6%
miscellaneous	2.7 s	3.5%
Total	76.2 s	100.0%

Table 2. M6 wing. Cost analysis for the computation of the Fiedler vector.

	Timings	Percentage
matrix-vector products	43.9 s	73.4%
dot-products	5.2 s	8.7%
eigenvalue analyses	3.6 s	6.0%
SAXPYs and miscellaneous	7.1 s	11.9%
Total	59.8 s	100.0%

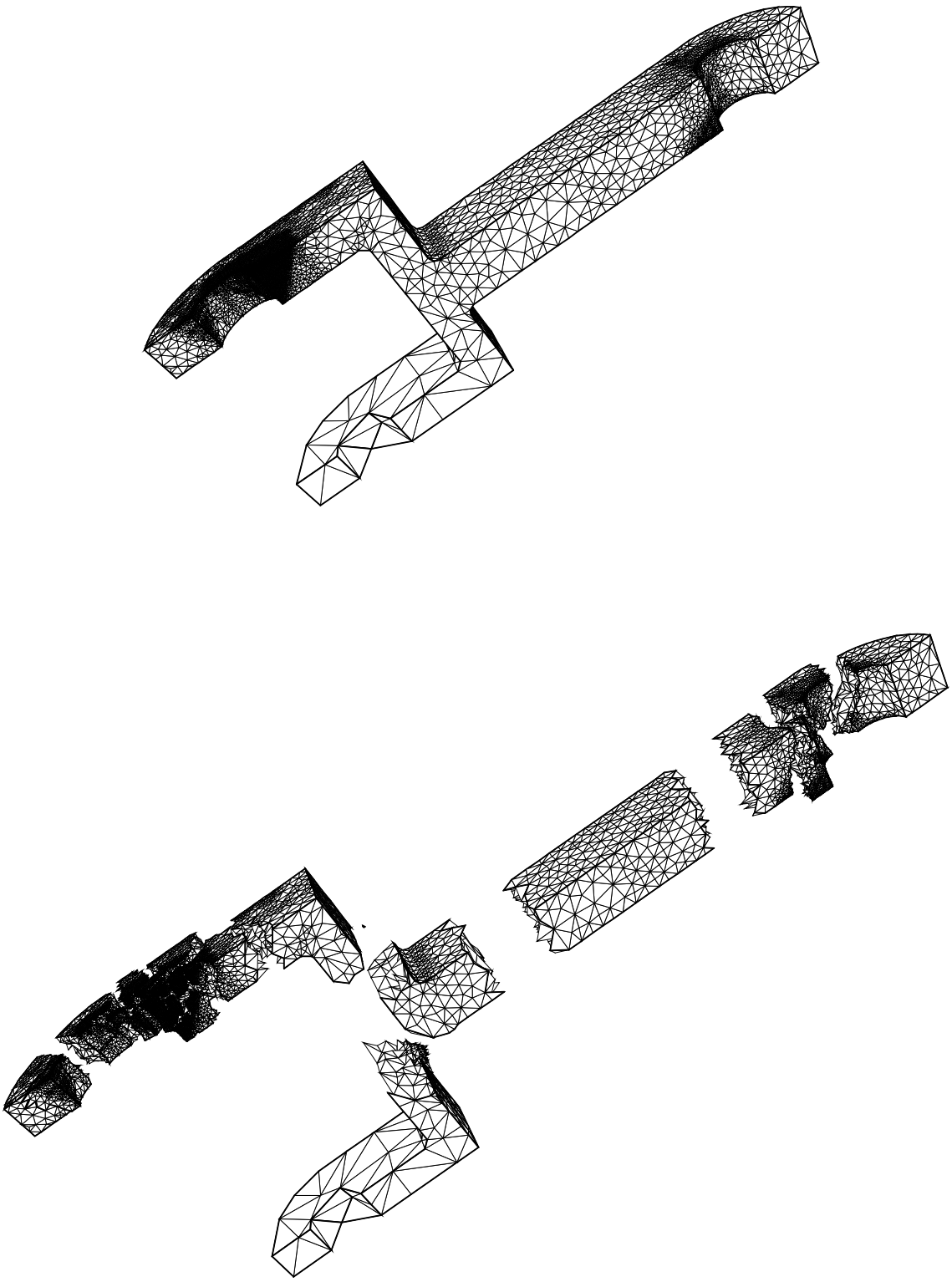

Figure 4. Half-bracket mesh decomposed into 32 subdomains.

3.2. Mesh of a half-bracket

An adaptively refined mesh of a half-bracket having 21,497 nodes and 98,052 tetrahedra was decomposed on a 32-processing node CM-5 equipped with 128 vector units. The graph representation has 187,958 edges. Views of the surface mesh and a decomposition into 32 subdomains are shown in Figure 4. The partitioning cost as a function of the recursive bisection process is presented in Figure 5. The same cost-related behavior as in the previous example can be observed. Partitioning the mesh into 128 subdomains took 58 seconds. 14,627 cuts in the graph were generated, which represents 7.8% of the total number of edges.

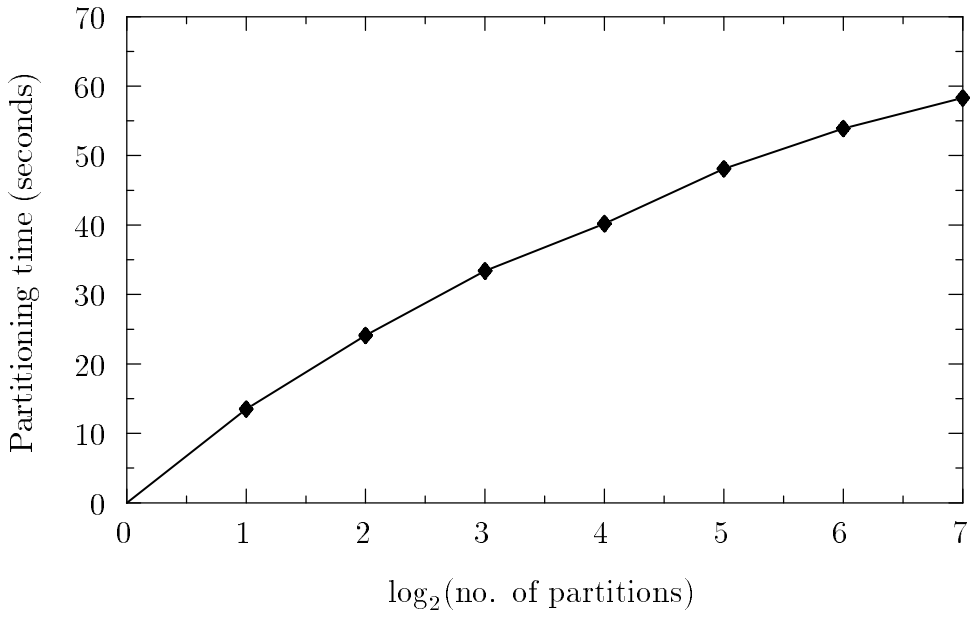

Figure 5. Half-bracket. Partitioning cost as a function of recursive bisection on a 32-node CM-5 system.

4. CONCLUSIONS

We have presented a data-parallel implementation of the recursive spectral bisection algorithm *without any sequential bottleneck*. The computing power of the CM-5 system has made possible the decomposition of large three-dimensional unstructured meshes in a reasonable time. The partitioning code was incorporated into the latest release of the CMSSL and made available to the CM-5 user community. The enhancements to the CM-5 architecture announced in November 1993 should make the RSB algorithm even more competitive with other mesh decomposition techniques [11].

ACKNOWLEDGMENTS

We would like to thank Arthur Raefsky (CENTRIC Engineering Systems), Horst Simon (NASA Ames), John Kennedy, Jacek Myczkowski and Richard Shapiro (Thinking Machines) for their helpful comments. We would also like to express our gratitude

to Jean Cabello and Rainald Löhner (George Washington University) for giving us the ONERA M6 wing mesh and to Mark Shephard (Rensselear Polytechnic Institute) for giving us the half-bracket mesh. Computing resources for this project have been provided by the Los Alamos Advanced Computing Laboratory, the National Center for Supercomputing Applications at the University of Illinois Urbana-Champaign and the Naval Research Laboratory.

REFERENCES

1. A. Pothen, H.D. Simon and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal of Matrix Analysis and Applications*, **11** (1990) 430–452.

2. H.D. Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering*, **2** (1991) 135–148.

3. *CM Fortran Language Reference Manual*, Version 2.1, Thinking Machines Corporation, Cambridge, MA, 1994.

4. Z. Johan, K.K. Mathur, S.L. Johnsson and T.J.R. Hughes, "An efficient communication strategy for finite element methods on the Connection Machine CM-5 system," *Computer Methods in Applied Mechanics and Engineering*, in press.

5. M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, **23** (1973) 298–305.

6. M. Fiedler, "Eigenvectors of acyclic matrices," *Czechoslovak Mathematical Journal*, **25** (1975) 607–618.

7. M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematical Journal*, **25** (1975) 619–633.

8. *CMSSL for CM Fortran: CM-5 Edition*, Version 3.1, Thinking Machines Corporation, Cambridge, MA, 1993.

9. *CMMD Reference Manual*, Version 3.0, Thinking Machines Corporation, Cambridge, MA, 1993.

10. *CM Fortran Libraries Reference Manual*, Version 2.1, Thinking Machines Corporation, Cambridge, MA, 1994.

11. C. Farhat and H.D. Simon, "TOP/DOMDEC: A software tool for mesh partitioning and parallel processing," Center for Space Structures and Controls, University of Colorado at Boulder, Report CU-CSSC-93-11, 1993.