Load Balancing Issues for a Multiple Front Method

Christophe Denis[1], Jean-Paul Boufflet[1], Piotr Breitkopf[2], Michel Vayssade[2], and Barbara Glut[3*]

[1] Department of Computing Engineering, UMR 6599 Heudiasyc,
Compiègne University of Technology, BP 20529
F-60205 Compiègne cedex, France
[2] Department of Mechanical Engineering, UMR 6066 Roberval
Compiègne University of Technology, BP 20529
F-60205 Compiègne cedex, France
{Christophe.Denis,Jean-Paul.Boufflet,
Piotr.Breitkopf,Michel.Vayssade}@utc.fr
[3] Institute of Computer Science
AGH University of Science and Technology
Cracow, Poland

Abstract. We investigate a load balancing strategy that uses a model of the computational behavior of a parallel solver to correct an initial partition of data.

1 Introduction

We deal with linear systems $K \cdot u = f$ issued from finite elements. The frontal approach interleaves assembly and elimination avoiding to directly manage the entire matrix K. A variable is eliminated when its corresponding equation is fully summed (I. Duff et al [1,2]). Rather than parallelize an existing code (P.R. Amestoy et al [3]), one can perform tasks in independent modules like in J. Scott [4] *MP42* solver based on the frontal code by I. Duff and J. Reid [2]. We use an implementation of a multiple front parallel method in the context of our academic software *SIC* [5,6].

The domain is partitioned using *METIS* [7] and *CHACO* [8]. This initial partition tends to minimize the communications and to balance the subdomain amount of data assuming that the computation cost is proportional to the number of vertices of the subgraph and that the order of assembly does not matter.

[9] seems to confirm the analysis presented by B. Hendrickson [10,11]: equipartitioning of data volumes does not result systematically in well balanced computational times.

We design a load balancing process transferring finite elements between subdomains to improve the initial partition. Test data are from the PARASOL project (http://www.parallab.uib.no/parasol).

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 163–170, 2004.

2 Problem Formulation

We use a non-overlapping domain decomposition:

1. the graph G_{elem} associated with the finite element mesh is partitioned into N_s subdomains $SD^{(j)}$;
2. each $SD^{(j)}$ is partially condensed in parallel;
3. an interface problem is built and then treated.

This process for an equivalent assembled matrix K can be block ordered:

$$\begin{pmatrix} K_{ii}^{(1)} & & & K_{ib}^{(1)} \\ & \ddots & & \vdots \\ & & K_{ii}^{(N_s)} & K_{ib}^{(N_s)} \\ K_{bi}^{(1)} & \dots & K_{bi}^{(N_s)} & K_{bb} = \sum_{j=1}^{N_s} K_{bb}^{(j)} \end{pmatrix}$$

Subscript i indicates "internal" and b "boundary". $K_{ii}^{(j)}$ are the terms of K associated with the internal variables of $SD^{(j)}$. The terms of $K_{bi}^{(j)}$ (resp. $K_{ib}^{(j)}$) correspond to the interactions between internal variables of $SD^{(j)}$ and the boundary ones. For each $SD^{(j)}$, we build the following matrix and a partial LU condensation

$$\begin{pmatrix} K_{ii}^{(j)} & K_{ib}^{(j)} \\ K_{bi}^{(j)} & K_{bb}^{(j)} \end{pmatrix} = \begin{pmatrix} L_{ii}^{(j)} & 0 \\ L_{bi}^{(j)} & I \end{pmatrix} \cdot \begin{pmatrix} U_{ii}^{(j)} & U_{ib}^{(j)} \\ 0 & S_{bb}^{(j)} \end{pmatrix} \tag{1}$$

The block $S_{bb}^{(j)}$ denotes the local Schur complement of $SD^{(j)}$.

$$S = \sum_{j=1}^{N_s} S_{bb}^{(j)} = \sum_{j=1}^{N_s} \left(K_{bb}^{(j)} - (L_{bi}^{(j)} \cdot U_{ib}^{(j)}) \right) \tag{2}$$

Using (1) we get the global Schur complement matrix :

$$S = \sum_{j=1}^{N_s} \left(K_{bb}^{(j)} - K_{bi}^{(j)} \cdot (K_{ii}^{(j)})^{-1} \cdot K_{ib}^{(j)} \right) \tag{3}$$

We use a frontal method to partially condense the matrices associated with each $SD^{(j)}$ and to treat the interface problem. The nested multiple front approach is based on the treatment of groups of $(K_{bb}^{(j)} - K_{bi}^{(j)} \cdot (K_{ii}^{(j)})^{-1} \cdot K_{ib}^{(j)})$. Matrices $S_{bb}^{(j)}$ can be viewed as a super-elements that can be assembled in a frontal matrix and partially condensed. The computational scheme we consider is a tree of tasks (1).

Definition 1. *A computation tree $\mathcal{A}^{N_s,y}$ has N_s leaves and y levels.*

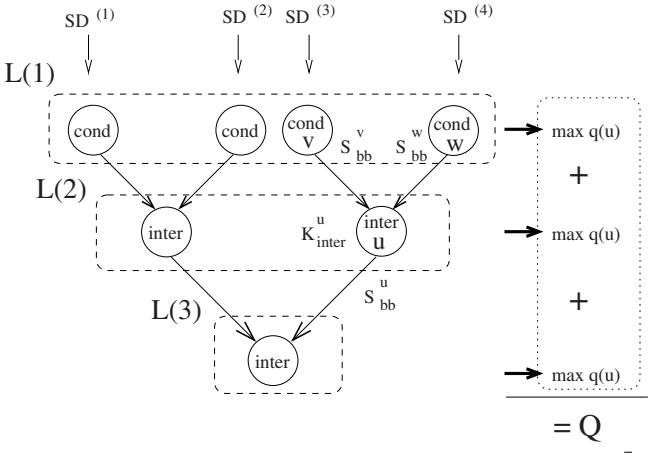

Fig. 1. The computation tree $\mathcal{A}^{4,3}$ and the principle of the estimation of the computation

A task u is associated with a vertex u of the computation tree and $q(u)$ is an estimation of the number of operations. Let $L(i)$ be the set of vertices of $\mathcal{A}^{N_s,y}$ at level i. The leaves in $L(1)$ correspond to the partial condensations of the $SD^{(j)}$. For a task u associated with an internal vertex of $\mathcal{A}^{N_s,y}$ we define:

- S^u_{bb} obtained by partial condensation on $SD^{(j)}$ or on the assembly K^u_{inter} of two matrices $S^{(i)}_{bb}$;
- K^u_{inter} the interface matrix obtained by assembling two matrices $S^{(i)}_{bb}$.

On the computation tree $\mathcal{A}^{4,3}$ of Fig. (1), subdomains $SD^{(3)}$ and $SD^{(4)}$ are partially condensed by tasks v and w. We obtain two matrices S^v_{bb} and S^w_{bb}. They are assembled in the interface matrix K^u_{inter}. The boundary variables between subdomains $SD^{(3)}$ and $SD^{(4)}$ correspond to fully summed rows and columns of K^u_{inter}. We obtain S^u_{bb} by partially condensing K^u_{inter}. The interface problem of level $L(3)$ is then solved and individual variables are obtained by successive back substitutions.

We use a coarse grain parallel approach where tasks are the partial condensation and interface problems. The communication times and the back restitution times are negligible.

The goal is to correct an initial partition of the graph G_{elem}. An estimator of the number of operations of the frontal method is applied on $SD^{(j)}$:

$$Q_1(V_e^{(j)}, SD^{(j)}) = \sum_{\gamma \in assembling} |F_\gamma| + \sum_{\gamma \in elimination} |F_\gamma|^2 \qquad (4)$$

where $V_e^{(j)}$ is the reordering vector of the finite elements of $SD^{(j)}$. Q_1 [12] counts operations and gives 10% error between the estimated time for $SD^{(j)}$ and the actual time $T_{SD^{(j)}}$.

The second estimator counts the number of operations for the partial condensation of K^u_{inter}.

We evaluate then the maximum number of operations $\max q(u)$ for each level $L(i)$ as shown in Fig. (1). The sum $Q = \sum \max q(u)$ provides an estimation of the cost. In an ideal case of equal tasks at each level, Q is a tight estimation, otherwise it gives an upper bound.

We consider balanced trees obtained with multi-level tools [7]. First a unique task of $L(1)$ is assigned per processor. Then, $S^{(i)}_{bb}$ are sent to processors computing the associated tasks according to the computation tree.

Table 1. The PARASOL data and the finite element meshes used for our experiments

name	No. elts	order	name	No. elts	order
MT1	5 328	97 578	SUSPEN_D1	18 171	14 517
SHIPSEC8	35 280	114 919	C1	42 689	34 707
X104	6 019	108 384	MISSILE4	27 804	166 824

3 Principle of the Heuristics

The initial partition $\mathcal{P}$ is first computed using [7]. Then we apply the following heuristics:

1. for each subdomain $SD^{(j)}$ compute first a $V_e^{(j)}$, then $Q_1(V_e^{(j)}, SD^{(j)})$;
2. select $SD^{(max)}$ with maximum estimated number of operations;
3. determine the set $\mathcal{N}_{max}$ of indices of subdomains that are neighbors to $SD^{(max)}$;
4. virtually aggregate the subdomains of $\mathcal{N}_{max}$;
5. compute $Q^{\mathcal{N}_{max}}_{moy}$ the average number of operation of these subdomains;
6. compute Q_{trans} the number of operations to be transferred from $SD^{(max)}$;
7. compute m_t the number of elements to be transferred;
8. transfer a subset of m_t finite elements from $SD^{(max)}$ to the virtual subdomain.

The volume Q_{trans} is half the difference between the maximum estimated number of operations and $Q^{\mathcal{N}_{max}}_{moy}$ and m_t is ratio Q_{trans} over the number of operations per element. By applying this process k times we improve the initial partition. For our experiments we set $k = 100$ and select the best result.

A transfer primitive chooses finite elements near the common boundary in order to limit the growth of the interface. Consider examples from Fig. (2) to Fig. (4).

In Fig. (2) $SD^{(1)}$ has the maximum estimated number of operations. Grey elements are near the boundary between $SD^{(1)}$ and the virtual subdomain $[SD^{(2)}, SD^{(3)}]$ ($\mathcal{N}_1 = \{2, 3\}$).

Table 2. Q the estimated amount of computation, T_{glob} the real computing time (in s), the obtained gain (in %), and Δ^{mesu} the load balancing criterion

name	tree	method	Q	T_{glob} (s)	gain (%)	Δ^{mesu} (%)
MT1	A	Pmetis	$1,17.10^{11}$	94,4		96
	A	PmetisC	$1,00.10^{11}$	89,6	5,1	98
	B	Pmetis	$1,09.10^{11}$	66,7		71
	B	PmetisC	$8,05.10^{10}$	59.1	11,4	80
	C	Pmetis	$4,26.10^{10}$	43,0		54
	C	PmetisC	$3,15.10^{10}$	36,9	14,2	72
SHIPSEC8	A	Pmetis	$4,02.10^{11}$	266,6		81
	A	PmetisC	$3,85.10^{11}$	239,1	10,3	89
	B	Pmetis	$2,75.10^{11}$	160,5		71
	B	PmetisC	$1,75.10^{11}$	112,7	29,8	89
	C	Pmetis	$1,40.10^{11}$	108,3		71
	C	PmetisC	$1,20.10^{11}$	101,1	6,6	79
X104	A	Pmetis	$2,01.10^{11}$	162,9		98
	A	PmetisC	$2,01.10^{11}$	162,9	0	98
	B	Pmetis	$7,84.10^{10}$	70,1		76
	B	PmetisC	$6,39.10^{10}$	57,4	18.1	91
	C	Pmetis	$4,53.10^{10}$	48,5		46
	C	PmetisC	$4,48.10^{10}$	46,8	3.5	48
SUSPEN_D1	A	Pmetis	$2,67.10^{9}$	$21,1$		88
	A	PmetisC	$1,19.10^{9}$	$9,9$	$53,1$	73
	B	Pmetis	$1,20.10^{9}$	$9,9$		50
	B	PmetisC	$9,28.10^{8}$	$4,1$	$58,6$	80
	C	Pmetis	$7,75.10^{8}$	$6,7$		44
	C	PmetisC	$4,05.10^{8}$	$3,8$	$43,8$	73
C1	A	Pmetis	$1,78.10^{10}$	$129,9$		80
	A	PmetisC	$8,87.10^{9}$	$84,2$	$35,2$	88
	B	Pmetis	$9,23.10^{9}$	$99,4$		54
	B	PmetisC	$3,38.10^{9}$	$30,1$	$69,7$	90
	C	Pmetis	$3,11.10^{9}$	$24,7$		54
	C	PmetisC	$1,74.10^{9}$	$15,4$	$37,7$	90
MISSILE4	A	Pmetis	$1,44.10^{11}$	1220		71
	A	PmetisC	$3,91.10^{10}$	$395,3$	$67,6$	99
	B	Pmetis	$1,04.10^{11}$	$891,6$		65
	B	PmetisC	$4,81.10^{10}$	$406,6$	$54,4$	67
	C	Pmetis	$6,13.10^{10}$	$448,3$		59
	C	PmetisC	$2,26.10^{10}$	$204,5$	$54,4$	93

We compute a level structure through $SD^{(1)}$ from the boundary elements of [$SD^{(2)}$,$SD^{(3)}$]. We apply the BFS algorithm on the element graph of $SD^{(1)}$ initializing its queue with boundary elements corresponding to level 0. We obtain a spanning tree where level l contains elements at a distance of l edges to level 0. It may be seen as using a virtual vertex r connecting $SD^{(1)}$ and its associated virtual subdomain (Fig. (3)). We assume m_t=4. We then transfer selected elements to the neighbor subdomains (Fig. (4)).

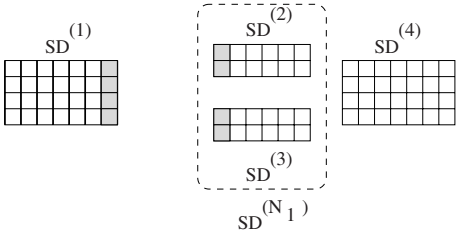

Fig. 2. The initial partition of the domain into 4 subdomains

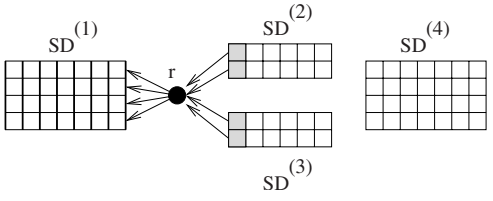

Fig. 3. Initialisation of the root r of the level structure in order to select the finite elements to be transfered from $SD^{(1)}$ to the virtually aggregated subdomain

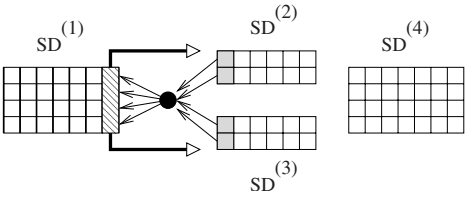

Fig. 4. m_t finite element are transfered

4 Results

The experiments were performed on a 10 Athlon 2200+, 1Gb bi-processors cluster, running LINUX Red Hat 7.1, with a 1 Gbit/s Ethernet network. Table (1) gives the sizes of the PARASOL data, and of some arbitrary meshes. The *order* column gives the size of the assembled matrix.

Three types of computation tree were used, and we define the labels: A for $\mathcal{A}^{2,2}$, 2 subdomains and 2 levels; B for $\mathcal{A}^{4,3}$, 4 subdomains and 3 levels and C for $\mathcal{A}^{8,4}$, 8 subdomains and 4 levels.

Table (2) presents the results: estimates Q, and measures T_{glob}. $Pmetis$ is the original $METIS$ decomposition and $PmetisC$ is the corrected one. $T_{SD^{(j)}}$ is measured for each $SD^{(j)}$ along with the load balancing criterion:

$$\Delta^{mesu} = \frac{1}{N_s} \frac{\sum_{j=1}^{N_s} T_{SD^{(j)}}}{\max_j T_{SD^{(j)}}}$$

In the ideal case the $T_{SD^{(j)}}$ are equal and $\Delta^{mesu} = 1$.

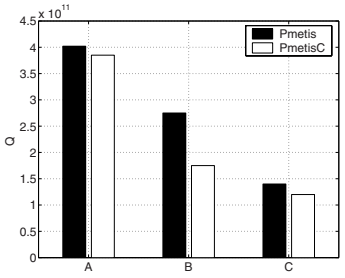

Fig. 5. Q : the estimated amount of computation before and after applying the heuristics for the SHIPSEC8 data

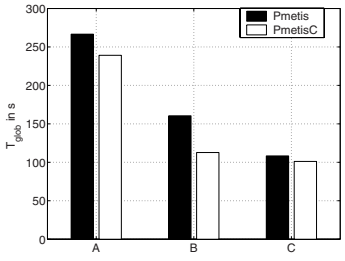

Fig. 6. T_{glob} : the real computing time (in s) before and after applying the heuristics for the SHIPSEC8 data

Table (2) shows that Δ^{mesu} is improved. The transfer primitive was modified in order to limit the number of interface nodes.

Figs. (5) and (6) show a good correlation between Q and T_{glob}. However, we do not obtain a perfect balance, because the estimations do not reflect exactly the real computations. Moreover, moving elements influences the ordering and consequently the computation time. It is therefore difficult to attain $\Delta^{mesu} = 1$. As a rule, fewer than 10 iterations of the heuristics provide the maximum gain reported in Table (2).

5 Conclusion

We propose a heuristics to correct an initial domain decomposition based on equal volumes of data, in order to balance the estimated number of operations of a multiple-front method. With this coarse-grained parallel approach, the preliminary results obtained on the benchmark improve computing time. The modification of the boundary due to the transfer of finite elements can increase the number of interface nodes and the size of the interface problem.

References

1. Duff, I., Erisman, A., Reid, J.: Direct Methods for Sparse Matrices. Monographs on Numerical Analysis. Clarendon Press - Oxford (1986)
2. Duff, I.S., Scott, J.A.: MA42 – A new frontal code for solving sparse unsymmetric systems, technical report ral 93-064. Technical report, Chilton, Oxon, England (1993)
3. P.R. Amestoy, I.S. Duff, J.Y.L., Koster, J.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM J. Matrix Anal. Appl. **23** (2001) 15–41
4. Scott, J.: The design of a parallel frontal solver,technical report ral-tr99-075. Technical report, Rutherford Appleton Laboratory (1999)

5. Escaig, Y., Vayssade, M., Touzot, G.: Une méthode de décomposition de domaines multifrontale multiniveaux. Revue Européenne des Eléments Finis **3** (1994) 311–337
6. Breitkopf, P., Escaig, Y.: Object oriented approach and distributed finite element simulations. Revue Européenne des Eléments Finis **7** (1998) 609–626
7. Karypis, G., Kumar, V.: Metis : A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Technical report, University of Minnesota, Department of Computer Science (1998)
8. Hendrickson, B., Leland, R.: The chaco user's guide, version 2.0. Technical report, Sandia National Laboratories (1995)
9. Boufflet, J., Breitkopf, P., Denis, C., Rassineux, A., Vayssade, M.: Optimal element numbering schemes for direct solution of mechanical problems using domain decomposition method. In: 4th ECCOMAS Solid Mechanics Conference. (2000) Espagne.
10. Hendrickson, B.: Graph partitioning and parallel solvers: Has the emperor no clothes? In: Irregular'98, Lecture Notes in Computer Science. Volume 1457. (1998) 218–225
11. Hendrickson, B.: Load balancing fictions, falsehoods and fallacies. Applied Mathematical Modelling **25** (2000) 99–108
12. Boufflet, J., Breitkopf, P., Denis, C., Rassineux, A., Vayssade, M.: Equilibrage en volume de calcul pour un solveur parallèle multi-niveau. In: 6ème Colloque National en Calcul des Structures. (2001) 349–356 Giens, France.