Parallel Mesh Refinement for 3-D Finite Element Electromagnetics With Tetrahedra: Strategies for Optimizing System Communication

Da Qi Ren and Dennis D. Giannacopoulos

Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 2A7, Canada

Communication strategies in parallel finite element methods can greatly affect system performance. The communication cost for a proposed parallel 3-D mesh refinement method with tetrahedra is analyzed. A Petri Nets-based model is developed for a target mesh refinement algorithm and parallel computing system architecture, which simulates the inter-processor communication. Subsequently, estimates for performance measures are derived from discrete event simulations. The potential benefits of this approach for developing high performance parallel mesh refinement algorithms are demonstrated by optimizing the system communication costs for varying problem size and numbers of processors.

Index Terms—**Adaptive systems, electromagnetic analysis, finite element methods, parallel processing, software methodology.**

I. Introduction

THE accuracy and efficiency of approximate solutions obtained with the finite element method (FEM) for practical electromagnetic problems can be highly dependent on 3-D mesh refinement algorithms. Advances in parallel computing have made higher fidelity at finer solution resolution possible. However, inter-processor communication costs in parallel FEM can greatly degrade the system performance and diminish the potential benefits of utilizing increased numbers of processors. The communication cost in parallel mesh refinement is dependent on the underlying computational algorithm as well as the system architecture. The objective of analyzing parallel communication paradigms for specific architectures in advance is to optimize use of system resources and improve performance.

In this paper, we develop a new approach for the modeling, analysis, and design of communication schemes in parallel finite element mesh refinement that utilizes Petri Nets. Petri Nets-based models allow for a relatively detailed description of a system due to their formal syntax and functional semantics, and can reveal key characteristics of system performance stochastically. While Petri Nets have been used for discrete event-based simulation of various applications, to our knowledge, they have not been considered previously for communication costs in parallel 3-D FEM mesh refinement [1]–[3]. In addition, we use the proposed approach for the optimization of the communication strategy for a 3-D parallel mesh refinement model suitable for FEM electromagnetics with tetrahedra [4], [5].

II. Parallel Mesh Refinement Approach

Tetrahedra are employed frequently in 3-D electromagnetic analysis and design with the FEM to achieve the geometric discretization of the problem domain. Several tetrahedral mesh refinement schemes are possible to improve the solution accuracy required for engineering tolerances [4]. To solidify concepts, consider the subdivision of a tetrahedron indicated by Fig. 1. This refinement rule initially subdivides a tetrahedron into four scaled duplicate tetrahedra and one octahedron as shown in Fig. 1(a). Next, the octahedron is further subdivided into six octahedra and eight tetrahedra, as illustrated in Fig. 1(b). Finally each octahedron from Fig. 1(b) will be subdivided into four tetrahedra as shown in Fig. 1(c), [3], [5]–[7].

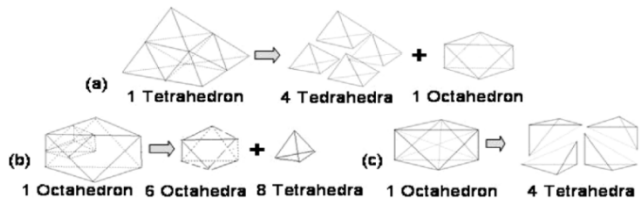

Fig. 1. Mesh refinement model: (a) tetrahedron subdivision; (b) primary octahedron subdivision; (c) secondary octahedron subdivision.

A master–slaves parallel computing model is assumed for implementing the mesh refinement method considered in this work [3], [6]. The master processing element (PE) initiates the program by checking the input data, gathering load information from slave PEs, and partitioning the initial set of geometric entities into sub-domains. The master PE then broadcasts the complete domain decomposition data and sub-domain assignments to corresponding slave PEs, which proceed with the mesh refinement of their assigned sub-domains, as shown in Fig. 2. The time for each slave PE to finish receiving a workload assignment from the master PE may not be the same because of differences in the workloads and communication delays. At this stage, the master PE will wait until each slave PE has acknowledged complete receipt of its workload assignment. Next the master PE broadcasts an instruction to all slave PEs to (approximately) synchronously start parallel computing [4]. The slave PEs executing the tetrahedral-octahedral subdivision algorithm (Fig. 1) work in parallel independently in each domain. When a slave PE completes its local tasks its data are written back to the master PE, where data from each sub-domain is merged to form the global result for the overall problem domain.

Digital Object Identifier 10.1109/TMAG.2006.872469

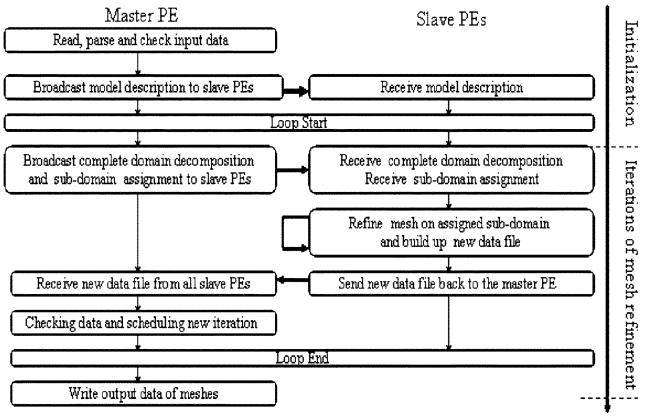

Fig. 2. Parallel mesh refinement approach.

III. Communication Model

In this section, a communication model is developed for the parallel mesh refinement strategy considered. Let $T_i^{(k)}$ and $O_i^{(k)}$ represent the quantity of tetrahedra and octahedra produced, respectively, in iteration i by PE P_k. In Fig. 1(a) and (b) subdivisions, for iteration i each tetrahedron of iteration $i-1$ can be subdivided into four smaller tetrahedra and one octahedron, and each octahedron of iteration $i-1$ can be subdivided into eight tetrahedra and six smaller octahedra. Thus, we have

$$T_i^{(k)} = 4T_{i-1}^{(k)} + 8O_{i-1}^{(k)} \quad (1)$$

$$O_i^{(k)} = T_{i-1}^{(k)} + 6O_{i-1}^{(k)}. \quad (2)$$

In any iteration of the mesh refinement, if matrix assembly is required, each octahedron in the element list will be subdivided into four tetrahedra as in Fig. 1(c), and in this case

$$T_i^{(k)} = T_{i-1}^{(k)} + 4O_{i-1}^{(k)} \quad (3)$$

$$O_i^{(k)} = 0. \quad (4)$$

Let t_{part} and o_{part} be the time required for a tetrahedron or octahedron subdivision, as shown in Figs. 1(a) and (b), respectively. For iteration i, the computation time t_i^{comp} and communication time t_i^{comm} for P_k are given by (5) and (6), respectively.[4]

$$t_i^{\text{comp}}(p_k) = 5T_{i-1}^{(k)} \cdot t_{\text{part}} + 14O_{i-1}^{(k)} \cdot o_{\text{part}} \quad (5)$$

$$t_i^{\text{comm}}(p_k) = t_{\text{startup}} + \left(T_i^{(k)} + O_i^{(k)}\right) \cdot t_{\text{data}}. \quad (6)$$

Here t_{startup} represents the message startup time and t_{data} is the transmission time to send the data for one element.

For n PEs in a master–slave model, there will be $n-1$ slave PEs in charge of $n-1$ sub-domains. Let $t_i^{\text{comm}}(p_0)$ be the time for the master PE p_0 to broadcast sub-domain workload assignments to all slave PEs. In total, n processors participate in the broadcast operation and the broadcast procedure involves $\log_2(n)$ point-to-point simple message transfers [4], each at a time cost of $t_{\text{startup}} + t_{\text{data}} \cdot (O_{i-1} + T_{i-1})$. Therefore, the total time taken by the procedure is

$$t_i^{\text{comm}}(p_0) = (t_{\text{startup}} + (T_{i-1} + O_{i-1}) \cdot t_{\text{data}}) \log_2(n). \quad (7)$$

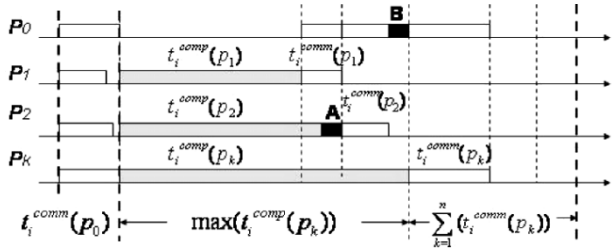

Fig. 3. Timing for parallel mesh refinement in typical master–slave model.

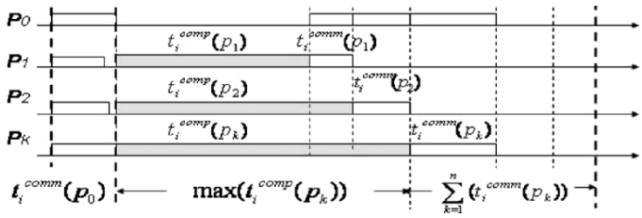

Fig. 4. Timing for pipelined communication design.

The time t_i to complete the mesh refinement for iteration i satisfies (8)–(9). The proof is given in the timing chart of Fig. 3

$$t_i \le t_i^{\text{comm}}(p_0) + \max\left(t_i^{\text{comp}}(p_k)\right) + \sum_{k=1}^{n-1}\left(t_i^{\text{comm}}(p_k)\right) \quad (8)$$

$$t_i \ge t_i^{\text{comm}}(p_0) + \max\left(t_i^{\text{comp}}(p_k) + t_i^{\text{comm}}(p_k)\right). \quad (9)$$

After each iteration of its computational loop (Fig. 2), a slave PE performs point-to-point communication to send data back to the master PE. As shown in the timing chart of Fig. 3, a PE's communication will be potentially blocked until another PE has finished sending/receiving data (point A and B). It would be preferable if we could overlap the transmission of these blocks with the computation for the mesh refinement, as many recent distributed-memory parallel computers have dedicated communication controllers that can perform the transmission of messages without interrupting the PE's CPU.

IV. Pipelined Communication Design

A pipelined communication strategy is designed for our mesh refinement scheme, which overlaps communication and computation in order to avoid inter-processor communication blocks (as described above). Briefly, the idea is to adjust the workload assigned to each PE so as to create load imbalances in the sub-domain partitioning stage. The load imbalances will result in differences in computation times for each PE. This time difference between PEs is used for overlapping (pipelining) one PE's computation time with the data transmission time of another PE. This is illustrated in Fig. 4.

In iteration i, the workloads for PE P_k and P_j are $O_{i-1}^{(k)} + T_{i-1}^{(k)}$ and $O_{i-1}^{(j)} + T_{i-1}^{(j)}$, respectively. The difference in computation time between P_k and P_j is $t_i^{\text{comp}}(p_j) - t_i^{\text{comp}}(p_k)$ (assuming that P_k finishes its computation first). Thus, the overlap in the communication time for P_k, in this case, is given by (10). When P_k finishes its computation it starts transferring results to the master PE P_0, while P_j keeps computing results for its domain. After $t_i^{\text{comp}}(p_j) - t_i^{\text{comp}}(p_k)$, P_k completes its data transfer and P_j finishes computing, and then P_j starts sending data to P_0. The

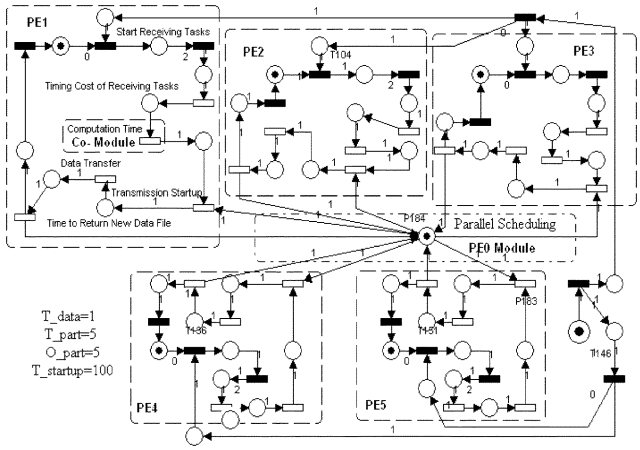

Fig. 5. Petri-Nets parallel communication model for 6 PEs.

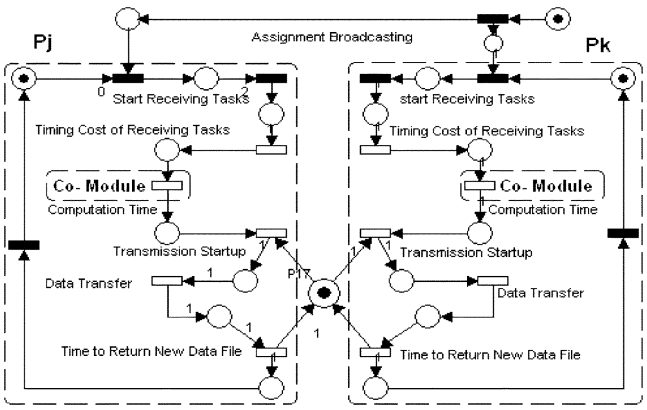

Fig. 6. Petri Nets module for pipelined communication of P_j and P_k.

time slot $t_i^{\text{comp}}(p_j) - t_i^{\text{comp}}(p_k)$ allows pipelining the communication of P_k and computation of P_j. To achieve this communication pipeline that satisfies (10), the appropriate difference in workloads between P_k and P_j must be determined, and is given by (11); where ΔT_{i-1} and ΔO_{i-1} are the required differences in quantities of the input tetrahedra and octahedra, respectively, between the two PEs P_k and P_j

$$t_i^{\text{comp}}(p_j) - t_i^{\text{comp}}(p_k) = t_i^{\text{comm}}(p_k) \tag{10}$$

$$\begin{aligned} &t_{\text{startup}} + \left(5T_{(i-1)}^{(k)} + 14O_{(i-1)}^{(k)}\right) \cdot t_{\text{data}} \\ &= 5 \cdot t_{\text{part}} \cdot \left(T_{i-1}^{(j)} - T_{i-1}^{(k)}\right) + 14 \cdot o_{\text{part}} \cdot \left(O_{(i-1)}^{(j)} - O_{(i-1)}^{(k)}\right) \\ &= 5 \cdot t_{\text{part}} \cdot \Delta T_{i-1} + 14 \cdot o_{\text{part}} \cdot \Delta O_{i-1}. \end{aligned} \tag{11}$$

It should be noted that, the sub-domain workload adjustment is justified based on the properties of the refinement rule in Fig. 1(a)–(c). That is, each element is covered exactly by its parent element, and all meshes in the hierarchy are conforming (no hanging nodes exist). Thus, the rule can be applied to neighboring elements in adjacent sub-domains without mesh consistency problems [3], [5], [7], [8].

V. Petri Nets Model and Simulation

The Petri Nets simulation developed for our parallel mesh refinement algorithm involves modeling the occurrence of events as they evolve in time and their effects as represented by transitions of states during the parallel mesh refinement process. We map the algorithm with the supporting formulae (1)–(11) into the Petri Nets model, which involves six modules: one master and five slave PEs. The Petri Nets Model and initial sub-domain distribution table are shown in Fig. 5 and Table I, respectively. A representative workload adjustment between two slave PEs is shown in Table II, and the corresponding Petri Nets module is shown in Fig. 6. The communication costs are defined by transitions that connect PEs in the system together (Figs. 5 and 6). The system parameters $t_{\text{part}}, o_{\text{part}}, t_{\text{startup}}$, and t_{data} are defined in the transition delays in each stage of the computation and communication model. Note that the computation time of the mesh refinement processes is comprised of tetrahedron and octahedron subdivision and data preparation. An individual module named "Co-Module" was developed for modeling this computation time. In both Figs. 5 and 6 the "Co- Module" is abbreviated as a transition, namely "computation time."

TABLE I
Sub-Domain Distribution

3 PEs 2nd iteration	4 PEs 2nd iteration	5 PEs 3rd iteration	6 PEs 2nd teration
P0: Master	*P0:Master*	*P0:Master*	*P0:Master*
P1:T=3	*P1:T=2:*	*P1:T=8,O=2*	*P1:T=1*
P2:T=1,O=11	*P2:T=2*	*P2:T=8,O=2*	*P2:T=1*
	P3:O=1	*P3:T=4,O=3*	*P3:T=1*
		P4:T=4,O=3	*P4:T=1*
			P5:O=1

T and *O* represent the number of tetrahedra and octahedra, respectively, assigned to each slave PE at the 2nd or 3rd iteration refinement.

TABLE II
Sub-Domain Adjustment

Iteration	*M*(*i*)	*T*(*i*)	*O*(*i*)	*F*(*i*)
4	*52*	*176*	*84*	*512*
5	*410*	*1376*	*680*	*4096*
6	*3277*	*10944*	*5456*	*32768*
7	*26215*	*87424*	*43680*	*262144*
8	*209716*	*699136*	*349504*	*2097152*

M(*i*) is the total number of tetrahedra and octahedra to be adjusted between two slave PEs at iteration *i*+1. *F*(*i*) is the final number of tetrahedra for iteration *i* after performing the refinement rule in Fig. 1(c).

VI. Results

The performance results of the Petri-Nets simulations for 3 to 6 PEs are shown in Figs. 7 (nonpipelined) and 8 (pipelined). It may be noted from Fig. 7(a), that when the message size is greater than a specific value ($\sim$450 000 bytes) the system communication costs for 5 or 6 PEs are less than for 3 or 4 PEs. This is due to the "natural" pipelining effect caused by the increased number of PEs, so that communication and computation overlap to decrease the number of block points. Fig. 7(b) shows the load imbalance ratio for different numbers of PEs over the range of communication costs considered. The load imbalance ratio is the difference in work load between PEs divided by the total work load in a given iteration i. Load imbalances cause PEs to complete their individual tasks asynchronously, and can slow down the parallel computing speed. However, differences in computing ending times can allow for effective overlap of computation with communication, which can reduce the overall communication cost. It may be noted from Fig. 7(b), that for

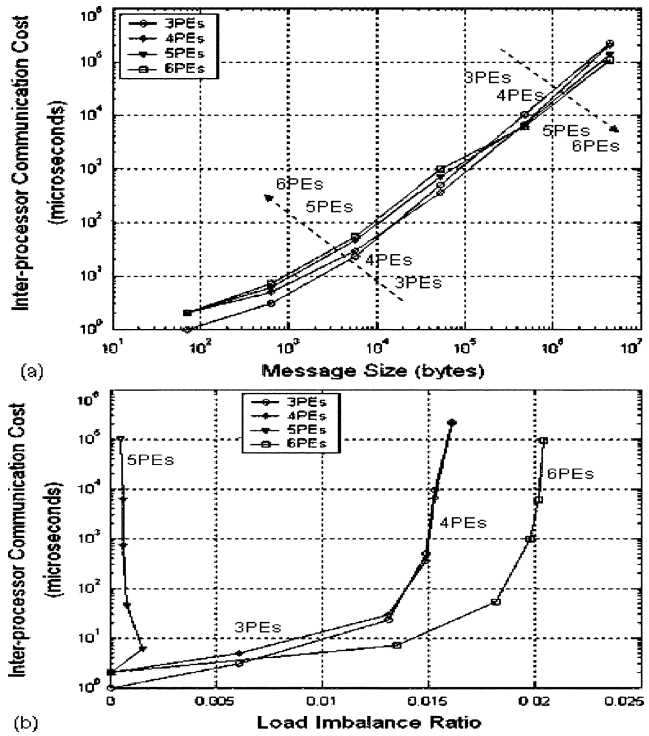

Fig. 7. Performance results (without designed pipeline): (a) communication cost; (b) load imbalance.

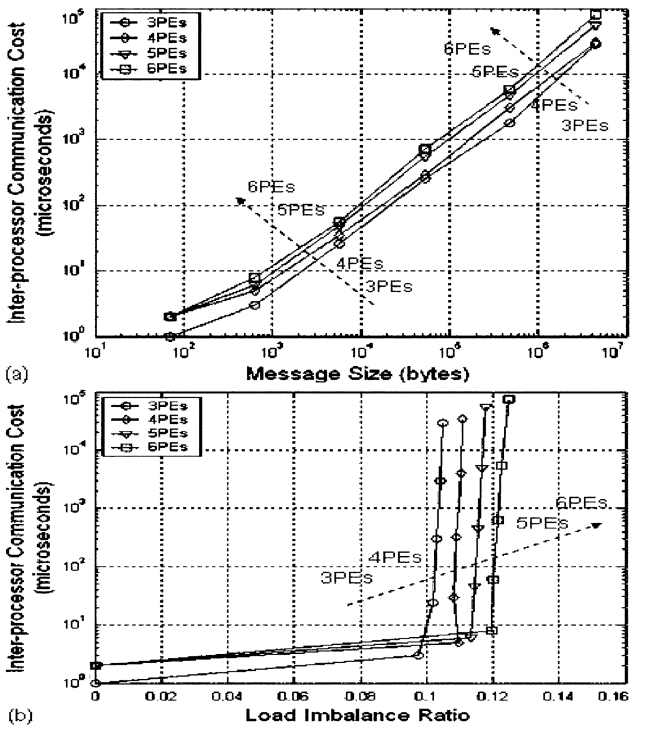

Fig. 8. Performance results (with designed pipeline): (a) communication cost; (b) load imbalance.

a given system communication cost, different load imbalance ratios result for different numbers of PEs. This information is useful for optimizing the design of the communication component of our mesh refinement strategy.

The corresponding results for the new pipelined design are shown in Fig. 8. The curves in Fig. 8(a) are ordered consistently, because the designed pipelined communication is scheduled intentionally to optimize system resources. For example, the communication cost for 6 PEs is greater than those for 5, 4, or 3 PEs at each iteration, because the parallel scheduling is controlled and increases in complexity with the number of PEs. The system communication costs are comparable with Fig. 7(a) until the fourth iteration, because the parallel scheduling cost is relatively large for smaller message sizes. However, there is a significant reduction in cost for the new pipelined design beginning with the fourth iteration. Fig. 8(b) shows a consistent increase in the load imbalance ratio for increasing PEs, as the system overlaps more computation and communication time. This is in agreement with the observation above: having more PEs incurs more scheduling cost. However, this is a beneficial tradeoff required to avoid the block points in the system.

VII. Conclusion

A new approach has been proposed and evaluated that utilizes Petri Nets as a modeling formalism for the analysis and design of communications strategies for parallel finite element mesh refinement systems. Modules have been developed for modeling each stage of the parallel algorithm, as well as the structure of the parallel system. The benefits of the new approach for overall system performance evaluation and design optimization are illustrated by the results for the new pipelined communication algorithm considered. Future work should include further performance optimization of the computation and communication cost in parallel FEM mesh refinement algorithm as well as other aspects of the FEM.

Acknowledgment

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds Québecois de la Recherche sur la Nature et le Technologies (FQRNT).

References

[1] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Berlin, Germany: Springer-Verlag, 2002.

[2] C. Lindemann, *Performance Modeling With Deterministic and Stochastic Petri Nets*. New York: Wiley, 1998.

[3] D. Q. Ren and D. D. Giannacopoulos, "A preliminary approach to simulate parallel mesh refinement with petri nets for 3D finite element electromagnetics," in *Proc. ANTEM 2004*, pp. 127–130.

[4] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, 2nd ed. New York: Addison-Wesley, 2003.

[5] G. Greiner and R. Grosso, "Hierarchical tetrahedral-octahedral subdivision for volume visualization," *Vis. Comput.*, vol. 16, pp. 357–369, Oct. 2000.

[6] P. J. Frey and P. L. George, *Mesh Generation: Application to Finite Elements*. Oxford, U.K.: Hermes Science, 2000.

[7] B. H. V. Topping, J. Muylle, P. Iványi, R. Putanowicz, and B. Cheng, *Finite Element Mesh Generation*. Kippen: Saxe-Coburg, 2004.

[8] W. P. Petersen and P. Arbenz, *Introduction to Parallel Computing: A Practical Guide With Examples in C*. Oxford, U.K.: Oxford Univ. Press, 2004.

Manuscript received June 20, 2005 (e-mail: dennis.giannacopoulos@mcgill.ca).