A New Strategy for Reducing Communication Latency in Parallel 3-D Finite Element Tetrahedral Mesh Refinement

Da Qi Ren, Steve McFee, and Dennis D. Giannacopoulos

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

A new, efficient pipelined communication strategy that significantly reduces latency for parallel 3-D finite element mesh refinement with tetrahedra is proposed. A Petri Nets-based model is developed in order to simulate the interprocessor communication costs for both the target mesh refinement algorithm and parallel architecture. Benchmark results show that the new pipelined design yields improved communication performance and overall parallel speedup for a range of refinement problem sizes, using different numbers of processors.

***Index Terms*—Finite element methods, mesh generation, parallel processing, petri nets.**

I. Introduction

MINIMIZING communication latency is an essential aspect of designing cost-effective parallel finite element methods (FEM) [1], [2]. The primary objective of this contribution is to introduce a new and efficient pipelined parallel processing communication strategy for tetrahedral FEM mesh refinement. The method is designed to minimize interprocessor communications latency, in order to limit the potential performance degradation that can occur in parallel tetrahedral mesh refinement implementations. The new strategy is fundamentally different from the communications strategies the authors reported earlier, and in particular, the previous calculation and assignment of specific net workload imbalances for the processors is not required [1].

Accurately designing, modeling and simulating communication schemes in advance can help to yield improved parallel speed-up by optimizing the use of available system resources [1]–[4]. In this work, a petri nets (PN)-based model is developed and used to investigate the performance of the new pipelined communication design with detailed simulations of practical mesh refinement applications, for a range of mesh sizes, and different numbers of processors [5], [6]. The parallel processing performance characteristics of the new design are evaluated and compared to nonpipelined methods, to assess the reduction in communications latency that can be achieved and the potential impact on overall parallel speed-up.

II. New Parallel Communication Strategy

A master-slaves parallel computing model is assumed for implementing a tetrahedral mesh refinement method [1], [7]. (Details of the model are provided in the following sections.) The master processing element (PE) initiates the program and controls the subdomain partitioning for corresponding slave PEs, which proceed with mesh refinement of their assigned subdomains. A slave PE performs point-to-point communication with the master PE after each mesh refinement iteration. However, a PE's communication will be potentially blocked until another one has finished transferring data with the master PE, as shown at point 'A' in Fig. 1. We have designed a new pipelined scheme capable of avoiding such interprocessor communication blocks

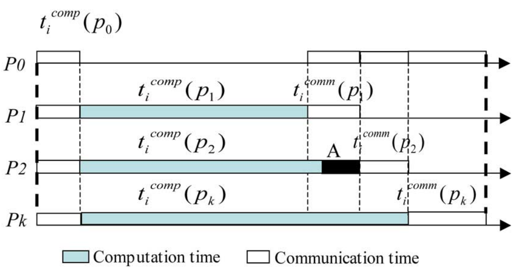
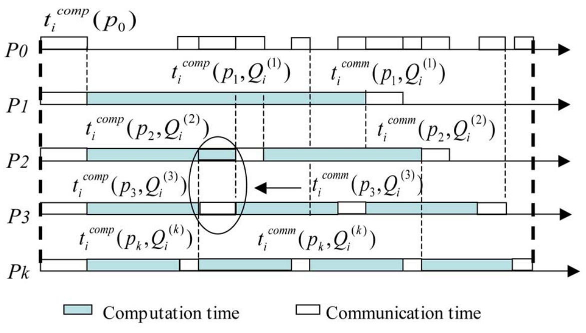


Fig. 1. Timing for parallel mesh refinement in a typical master-slaves design. (Note: $t_i^{\mathrm{comp}}(p_k)$ and $t_i^{\mathrm{comm}}(p_k)$ represent computation and communication times, respectively, for PE p_k during iteration i.)


$t_i^{comp}(p_0)$
P0
$t_i^{comp}(p_1,Q_i^{(1)})$
$t_i^{comm}(p_1,Q_i^{(1)})$
P1
$t_i^{comp}(p_2,Q_i^{(2)})$
$t_i^{comm}(p_2,Q_i^{(2)})$
P2
$t_i^{comp}(p_3,Q_i^{(3)})$
$t_i^{comm}(p_3,Q_i^{(3)})$
P3
$t_i^{comp}(p_k,Q_i^{(k)})$
$t_i^{comm}(p_k,Q_i^{(k)})$
Pk
Computation time
Communication time

$Q_i^{(1)}=1,\ Q_i^{(2)}=2,\ Q_i^{(3)}=3$ and $Q_i^{(k)}=k$, e.g.,

$$t_i^{comm}(p_3,Q_i^{(3)})=t_i^{comp}(p_2,Q_i^{(2)})-t_i^{comp}(p_3,Q_i^{(3)})$$

Fig. 2. Timing for parallel mesh refinement of new pipelined communication design. (Note: $Q_i^{(k)}$ is the number of workload segments for PE p_k during iteration i.)

as shown in Fig. 2. The key to the design is that the computational workload for each slave PE is divided into different numbers of segments; thus the PEs perform data transmission asynchronously as they complete *each* workload segment.

The new pipelined communication strategy can effectively overlap communication and computation operations to reduce interprocessor communication blocks without compromising the load balancing. This approach is fundamentally different from the method proposed by the authors in [1]. Essentially, this earlier contribution is based on pre-calculating and imposing systematic workload imbalances across the processors to achieve the overlap. With the present scheme, the master PE is free to assign the workload for each slave PE according to whatever net load balancing allocation is most appropriate [3],

Digital Object Identifier 10.1109/TMAG.2007.916038

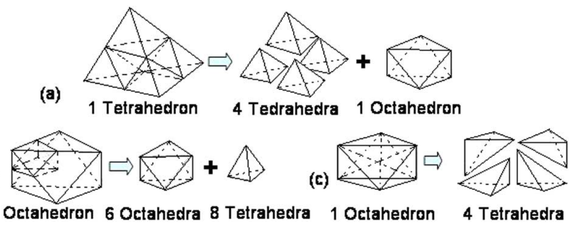


Fig. 3. Mesh refinement model: (a) tetrahedron subdivision; (b) primary octahedron subdivision; (c) secondary octahedron subdivision.

provided that the master PE also specifies a unique partitioning of each slave PE's total workload into a specific number of equal length subtask segments. Based on this model it should be possible to ensure that each slave PE receives a time period of unobstructed communication with the master PE, immediately upon completion of each subtask. An example, theoretical, limit case overlap for communication and computation possible with this approach is illustrated in Fig. 2. An efficient algorithm to partition the slave PEs' workloads is the key towards achieving this potential ideal performance. Additional details and analysis of the new algorithm follow a brief description of the parallel mesh refinement scheme and basic communication model.

A. Parallel Mesh Refinement Scheme

Several tetrahedron refinement schemes are possible for FEM applications, ranging from basic bisections to nested multicut refinement schemes designed to preserve different aspects of the geometric quality of the resulting mesh [6]. To fix concepts, consider the subdivision of a tetrahedron illustrated by Fig. 3. This refinement rule involves three steps: first, the tetrahedron is broken down into four scaled duplicate tetrahedra (one for each corner) and one octahedron (remainder) as shown in Fig. 3(a). Second, the resultant octahedron is then subdivided into six octahedra and eight tetrahedra, as illustrated by Fig. 3(b). Finally, each of the octahedra from Fig. 3(b) is subdivided into four tetrahedra as given in Fig. 3(c), [1], [4], [6]. The recursive application of these tetrahedral and octahedral refinement rules generates elements that belong to two congruence classes: one consisting of all generated tetrahedra, and one consisting of all generated octahedra. This refinement property is intentional, and it is useful for subsequent computations [1], [3].

The master PE initiates the mesh refinement program by gathering load information from the slave PEs and then partitioning the initial set of geometric entities into subdomains. The master PE then broadcasts the complete domain decomposition data and subdomain assignments to the relevant slave PEs, which proceed with the refinement of their assigned subdomains, as shown in Fig. 4. Once each of the slave PEs has acknowledged the receipt of its full workload assignment, the master PE broadcasts an instruction to all of the slave PEs to (approximately) synchronously start parallel computing [1], [7]. Each of the slave PEs executing the tetrahedral-octahedral subdivision algorithm (Fig. 3) work in parallel independently in each subdomain. Once a slave PE completes its assigned task its result data must be sent back to the master PE, where the data associated with each refinement subdomain is merged to form the global result for the overall problem domain.

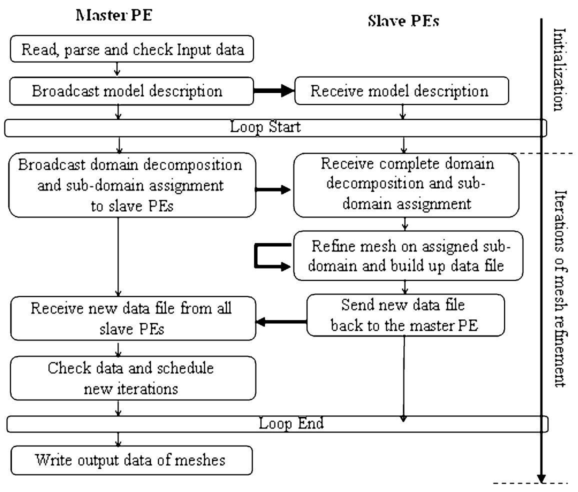

Master PE
Slave PEs
Read, parse and check Input data
Broadcast model description
Receive model description
Initialization
Loop Start
Broadcast domain decomposition and sub-domain assignment to slave PEs
Receive complete domain decomposition and sub-domain assignment
Refine mesh on assigned sub-domain and build up data file
Receive new data file from all slave PEs
Send new data file back to the master PE
Check data and schedule new iterations
Iterations of mesh refinement
Loop End
Write output data of meshes

Fig. 4. Parallel mesh refinement approach.

B. Basic Parallel Communication Model

A straightforward PN-based communication model suitable for representing parallel tetrahedral mesh refinement strategies based on the elemental subdivision scheme described above is developed below. All the relevant modeling parameters and relations between them need to be established at this time. Let $T_i^{(k)}$ and $O_i^{(k)}$ represent the numbers of tetrahedra and octahedra produced, respectively, by PE P_k, in iteration i. For the Fig. 3(a) and (b) subdivisions, in iteration i each tetrahedron of iteration $i-1$ can be subdivided into four smaller similar tetrahedra and one octahedron, and each octahedron of iteration $i-1$ can be subdivided into eight tetrahedra and six smaller octahedra. Therefore: $T_i^{(k)} = 4T_{i-1}^{(k)} + 8O_{i-1}^{(k)}$ and $O_i^{(k)} = T_{i-1}^{(k)} + 6O_{i-1}^{(k)}$. The Fig. 3(c) subdivision is not covered in this accounting because it only occurs in the case that matrix assembly is required. Let t_{part} and o_{part} be the times required for one tetrahedron and one octahedron subdivision, respectively, as defined by Fig. 3(a) and (b). Then, for iteration i, the computation time t_i^{comp} and communication time t_i^{comm} for PE P_k can be determined as

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

$$t_i^{\text{comm}}(p_k) = t_{\text{startup}} + (T_i^{(k)} + O_i^{(k)}) \cdot t_{\text{data}}. \tag{2}$$

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

For n PEs used in a master-slaves model, there will be $n-1$ slave PEs available to refine $n-1$ subdomains. Let $t_i^{\text{comm}}(p_0)$ be the time required for the master PE, p_0, to broadcast the subdomain workload assignments to all slave PEs. In total, n processors will participate in this broadcast operation, and the broadcast procedure will involve log (n) point-to-point simple message transfers, with each transfer counting for a time cost of $t_{\text{startup}} + t_{\text{data}} \cdot (O_{i-1} + T_{i-1})$ [7]. Therefore, the total time required for one complete broadcast procedure is

$$t_i^{\text{comm}}(p_0) = (t_{\text{startup}} + (T_{i-1} + O_{i-1}) \cdot t_{\text{data}}) \log(n). \tag{3}$$

Finally, the overall time t_i required to complete all the mesh refinements for iteration i will satisfy both (4) and (5), [1], [4]

$$t_i \leqslant t_i^{\mathrm{comm}}(p_0) + \max(t_i^{\mathrm{comp}}(p_k)) + \sum_{k=1}^{n-1}(t_i^{\mathrm{comm}}(p_k)) \quad (4)$$

$$t_i \geqslant \max(t_i^{\mathrm{comm}}(p_k) + t_i^{\mathrm{comp}}(p_k)). \quad (5)$$

Following an iteration of its computational loop (Fig. 4), a slave PE initiates a point-to-point communication to send data back to the master PE. However, as noted above and shown in the timing chart of Fig. 1, a PE's communication can be potentially blocked until another PE has finished transferring data to it (point A). If practical, it would be preferable to overlap the transmission of these blocks with the computation for the mesh refinement, as in the new pipelined communication design introduced in this paper.

C. New Pipelined Communication Design Details

As noted above, an efficient algorithm to partition the slave PEs' workloads is the key towards achieving the ideal potential performance of the new communication strategy. To meet this goal, recall that the workloads for PE P_k and PE P_j are $O_{i-1}^{(k)} + T_{i-1}^{(k)}$ and $O_{i-1}^{(j)} + T_{i-1}^{(j)}$, respectively, for iteration i. Let $Q_i^{(k)}$ and $Q_i^{(j)}$ represent the number of the workload segments assigned to PE P_k and PE P_j, respectively. For simplicity, all the workload segments for each individual slave PE can be set to equal size. However, the segment size used for one slave PE can not be set the same as that of any other. Then, for iteration i, the computation times per segment for PE P_k and PE P_j are

$$t_i^{\mathrm{comp}}\left(p_k, Q_i^{(k)}\right) = \left(5T_{i-1}^{(k)} \cdot t_{\mathrm{part}} + 14O_{i-1}^{(k)} \cdot o_{\mathrm{part}}\right)/Q_i^{(k)} \quad (6)$$

$$t_i^{\mathrm{comp}}\left(p_j, Q_i^{(j)}\right) = \left(5T_{i-1}^{(j)} \cdot t_{\mathrm{part}} + 14O_{i-1}^{(j)} \cdot o_{\mathrm{part}}\right)/Q_i^{(j)}. \quad (7)$$

Therefore, the difference between the times defined by (7) and (6) defines the time interval that can be used for pipelining the communications of PE P_k with the computations of PE P_j, i.e.,

$$t_i^{\mathrm{comp}}\left(p_j, Q_i^{(j)}\right) - t_i^{\mathrm{comp}}\left(p_k, Q_i^{(k)}\right) = t_i^{\mathrm{comm}}\left(p_k, Q_i^{(k)}\right). \quad (8)$$

To achieve the communication pipeline that satisfies (8), the required difference in workload segment size between PE P_k and PE P_j is determined by (9)

$$t_{\mathrm{startup}} + \frac{\left(5T_{i-1}^{(k)} + 14O_{i-1}^{(k)}\right)}{Q_i^{(k)}} \cdot t_{\mathrm{data}} = \frac{Q_i^{(j)} \cdot T_{i-1}^{(k)} - Q_i^{(k)} \cdot T_{i-1}^{(j)}}{Q_i^{(k)} \cdot Q_i^{(j)}} \cdot 5t_{\mathrm{part}} + \frac{Q_i^{(j)} \cdot O_{i-1}^{(k)} - Q_i^{(k)} \cdot O_{i-1}^{(j)}}{Q_i^{(k)} \cdot Q_i^{(j)}} \cdot 14o_{\mathrm{part}}. \quad (9)$$

The PN simulation developed for the new pipelined communication 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 [5]. The mesh refinement algorithm and associated interprocessor communications costs are mapped into the PN model, which has eight modules: one master and seven slave PEs, as shown in Fig. 5. The communication costs are defined by transitions that connect PEs in this system. The system parameters $t_{\mathrm{part}}, o_{\mathrm{part}}, t_{\mathrm{data}}$, and t_{startup} are defined in the transition delays in each stage of the computation and communication model [4]. Note that the computation time of the mesh refinement processes includes both tetrahedron and octahedron subdivision and data preparation; an individual module was defined for modeling this computation time and is abbreviated as the transition labeled "co-module" in Fig. 5.

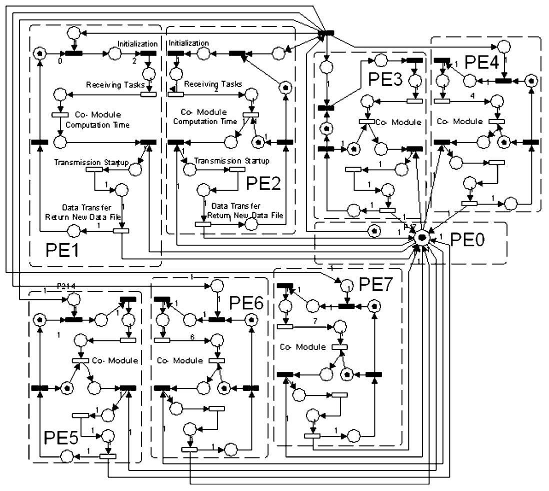


Fig. 5. PN parallel communication model for 8 PEs.

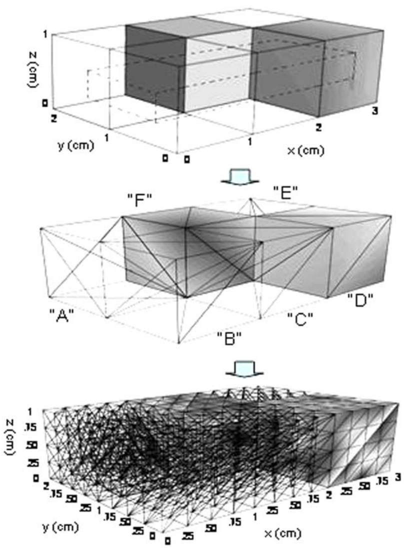


Fig. 6. Sub-domain decomposition of rectangular resonant cavity.

III. Results

The efficacy of the new pipelined communication design is investigated using the 3-D rectangular resonant cavity model illustrated in Fig. 6. The cavity was initially discretized into six smaller rectangular blocks (A–F); each of these blocks was subdivided into 6 tetrahedra. The resulting 36 tetrahedra are subdomains assigned to the slave PEs in the parallel system.

The performance results of the PN simulations for parallel refinement of the resonant cavity using 3 to 8 PEs are shown

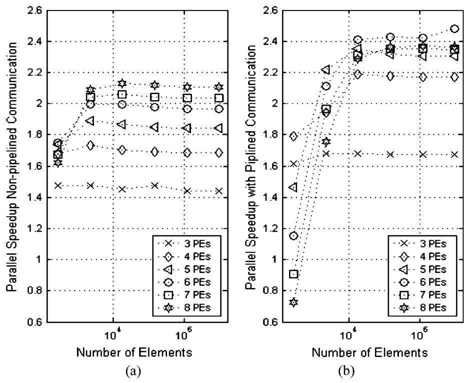


Fig. 7. Parallel communication speedup: (a) nonpipelined; (b) pipelined.

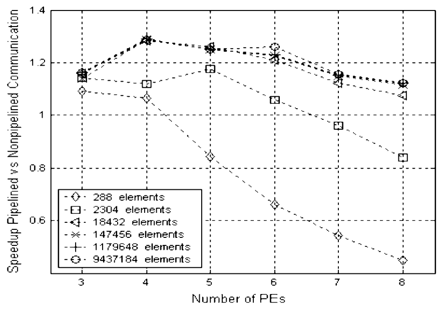


Fig. 8. Pipeline Speedup: pipelined versus nonpipelined communication.

in Figs. 7 and 8. Fig. 7(a) describes the regular parallelization speedup without pipelined communication: using 3–8 PEs can yield a speedup of 1.45∼2.15 times faster than one PE. For the initial iterations, 5–6 PEs yields a better speedup than 7–8 PEs, because the message size is too small relative to the parallel overhead [2]. Fig. 7(b) shows the parallel speedup achieved with the new pipelined communication design: for the initial iterations, the speedups of 7–8 PEs are less than 1, because the workloads in the initial iterations are relatively small and the pipelining cost counteracts the design benefit; for the second iteration 3–8 PEs yields speedups of 1.65 ∼ 2.5 times more than one PE. In each iteration of Fig. 7(b), 6PEs performed better than 5, 7, or 8 PEs. The reason is increasing the number of breakpoints can reduce the probability of a collision between any of two slave PEs in the data transmission, but when the number of work segments is increased, the pipelining cost and communication overhead are also increased accordingly [2]. This is a beneficial tradeoff required to avoid the block points in the system. The peak performance result for a specific number of PEs and break points, based on the different scenarios considered in this study, was achieved for the case of 6 PEs with 1–6 breakpoints.

To verify the computational advantage provided by the new pipelined communication design, the performance of pipelined and nonpipelined approaches are compared in Fig. 8. The results represent 3–8 PEs, operating over six mesh refinement iterations, ranging from 288 to 9, 437, 184 elements. Starting from the third iteration, there is an average speedup of $>10\%$ compared to nonpipelined communication. It should be noted, this speed-up is in addition to the parallel speed-ups observed as the number of PEs are increased for a given mesh size. For more than 4 or 6 PEs in the first and second iterations, respectively, no speed-up occurs because the message size is too small relative to the communication startup frequency required, which decreases the pipeline efficiency. These results corroborate previous findings that the most effective number of PEs for a given problem size is not necessarily intuitively evident, i.e., more is not always better [2].

IV. CONCLUSION

A new, efficient pipelined communication strategy to reduce latency for parallel tetrahedral finite element mesh refinement applications has been introduced, and a fully detailed PN-based model has been developed to evaluate its performance. It should be noted that this new pipelined communication design is intrinsically different from that previously reported in [1] for the following important reason: all slave PEs can be allocated, ideally, the same (balanced) *overall* computational workload. Compared with the previous approach, which required the master PE to compute optimal workload *imbalances* for each PE to pipeline the communications, the new pipeline method avoids both this costly calculation and also the idle periods that can occur when significant load imbalances are distributed over the slave PEs. Performance results from the simulations show that the new pipelined design yields significantly improved communication speedup for a range of refinement prloblem sizes, using different numbers of processors.

ACKNOWLEDGMENT

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and Le Fonds Québécois de la Recherche sur la Nature et les Technologies.

REFERENCES

[1] D. Q. Ren and D. D. Giannacopoulos, “Parallel mesh refinement for 3-D finite element electromagnetics with tetrahedra: Strategies for optimizing system communication,” *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1251–1254, Apr. 2006.

[2] S. McFee, Q. Wu, M. Dorica, and D. Giannacopoulos, “Parallel and distributed processing for h-p adaptive finite element analysis: A comparison of simulated and empirical studies,” *IEEE Trans. Magn.*, vol. 40, no. 2, pp. 928–933, Mar. 2004.

[3] D. D. Giannacopoulos and D. Q. Ren, “Analysis and design of parallel 3-D mesh refinement dynamic load balancing algorithms for finite element electromagnetics with tetrahedra,” *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1235–1238, Apr. 2006.

[4] 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.

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

[6] G. Greiner and R. Grosso, “Hierarchical tetrahedral-octahedral subdivision for volume visualization,” *The Visual Computer*, vol. 16, no. 6, pp. 357–369, Oct. 2000.

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

Manuscript received June 24, 2007. Corresponding author: D. D. Giannacopoulos (e-mail: dennis.giannacopoulos@mcgill.ca).