# A Methodology for Performance Modeling and Simulation Validation of Parallel 3-D Finite Element Mesh Refinement With Tetrahedra

Da Qi Ren, Tim Park, Baruyr Mirican, Steve McFee, and Dennis D. Giannacopoulos

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

**The design and implementation of parallel finite element methods (FEMs) is a complex and error-prone task that can benefit significantly by simulating models of them first. However, such simulations are useful only if they accurately predict the performance of the parallel system being modeled. The purpose of this contribution is to present a new, practical methodology for validation of a promising modeling and simulation approach for parallel 3-D FEMs. To meet this goal, a parallel 3-D unstructured mesh refinement model is developed and implemented based on a detailed software prototype and parallel system architecture parameters in order to simulate the functionality and runtime behavior of the algorithm. Estimates for key performance measures are derived from these simulations and are validated with benchmark problem computations obtained using the actual parallel system. The results illustrate the potential benefits of the new methodology for designing high performance parallel FEM algorithms.**

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

## I. Introduction

DETERMINING accurate finite element method (FEM) solutions for very-large-scale electromagnetics problems can be highly challenging and computationally expensive. A number of the component procedures involved in the FEM solution process can be accelerated with parallel processing; one important case is mesh refinement. Programming parallel FEMs can be a very demanding and complex task, and designing parallel FEM systems can benefit significantly by simulating them first. For example, modeling and simulation methods that can accurately predict the efficacy of proposed parallel algorithms before they are implemented could provide system designers with essential performance characteristics required for optimizing efficiency. However, before such methods can be used with confidence, it is essential to be certain of the limits imposed by the modeling approximations and to validate the accuracy of the simulations.

The main objective of this paper is to present a new, practical methodology for validation of a promising Petri Nets (PN)-based modeling and simulation approach suitable for parallel 3-D unstructured mesh refinement in FEM electromagnetics [1]–[3]. To accomplish this goal, detailed estimates for key performance measures, for the target mesh refinement algorithm and parallel system configuration, are derived from PN model simulations and are compared with and evaluated in terms of benchmark computation results obtained from actual parallel system implementations using our specific validation methodology.

## II. Previous Work

A hierarchical tetrahedral-octahedral (HTO) mesh refinement model is considered in this work because of its potential to produce high quality tetrahedral elements [4]. A promising modeling and simulation approach for HTO subdivisions suitable for parallel 3-D mesh refinement in FEM electromagnetics was recently developed based on PN [1]–[3]. The value of PN-based approaches is that they are capable of representing systems characterized by concurrent, distributed, parallel, nondeterministic and stochastic operation [5]. As a mathematical tool, PN make it possible to set up and use state equations, algebraic equations, and related mathematical models which can be used to represent the behavior of parallel computations executed on very-large-scale architectures [6].

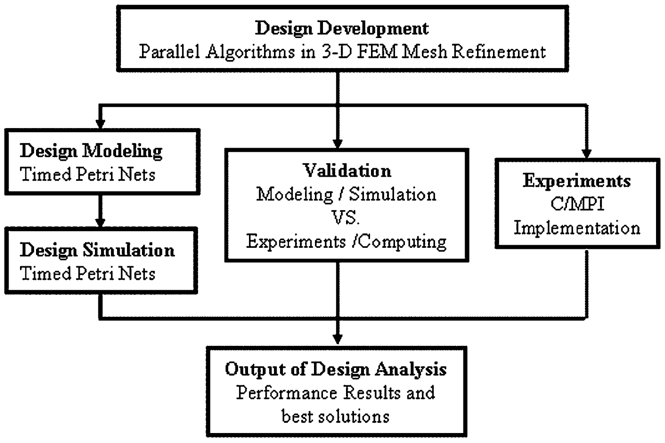

Fig. 1. Research framework for parallel 3-D FEM mesh refinement design.

Fig. 1 shows a framework for the effective and efficient design of parallel 3-D FEM mesh refinement algorithms. Some key components of this research framework have been addressed previously in [1]–[3]. Specifically, the performance of parallel HTO mesh refinement is modeled and simulated by PN, and the parallel speedup results with increasing number of elements are provided in [1]. In particular, we sample and translate to latency characteristics in the simulation model from the given application parameters such as the grid hierarchy and the number of processors; and system parameters such as CPU speed and communication bandwidth in [1]. Moreover, in [2], we model and simulate the performance of the Random Polling Dynamic Load Balancing (RPDLB) algorithm in parallel HTO mesh refinement, and examine the results by comparing them to the performance of parallel HTO mesh refinement without RPDLB. In addition, we develop efficient communication strategies for improving the parallel interprocessor communication performance

Digital Object Identifier 10.1109/TMAG.2007.916572

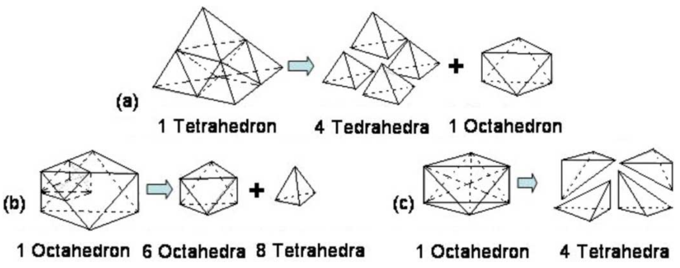

Fig. 2. Mesh refinement model: (a) tetrahedron subdivision; (b) primary octahedron subdivision; (c) secondary octahedron subdivision.

in [3]. As a fundamental component of this research framework not previously addressed, a goal of this paper is to conduct an experimental evaluation using a new, practical validation methodology for PN-based modeling, in order to determine its effectiveness for accurately capturing the dynamic behavior of parallel FEM applications. Specifically, the use of the PN-based approach for developing and simulating high performance parallel mesh refinement algorithms is examined using the new validation strategy specified in this work.

## III. Parallel Mesh Refinement Modeling With PN

The HTO mesh refinement model considered in this work is illustrated in Fig. 2. This method consists of cutting every edge into two and every face into four triangles, resulting in four tetrahedra, each a half-scale duplicate of the original, and an octahedron, as shown in Fig. 2(a). Next, the octahedron is cut into six half-size octahedra and eight tetrahedra, as illustrated by Fig. 2(b). The subdivisions of these two steps are repeated until all new tetrahedra satisfy specified mesh criteria [4]. Finally, each of the octachedra from Fig. 2(b) is subdivided into four tetrahedra as shown in Fig. 2(c). This mesh refinement model is considered because of its potential to produce high-quality tetrahedral elements [4].

A master-slaves parallel computing model was applied for implementing the mesh refinement method considered in this work [1], [7]. The master processing element (PE) initiates the program by partitioning the initial set of geometric entities into subdomains, and broadcasts the subdomain assignments to the corresponding slave PEs, which proceed with the mesh refinement of their assigned subdomains. When a slave PE completes its local tasks, its data will be sent back to the master PE, where data from each subdomain is merged to form the global result for the overall problem domain [1].

The key aspects of a PN-based model designed to represent the parallel mesh refinements in Fig. 2 are summarized below [1]. First, an individual module, named "Co-Module," was developed for modeling the mesh computation process associated with refining the tetrahedral and octahedral entities, and is shown schematically in Fig. 3. The operation of the Co-Module procedure starts with a scan of the tetrahedral/octahedral entities; then, the refinement rule is applied to each individual tetrahedron/octahedron. Once an individual element is processed, a signal is generated by "Scan Trigger" for loading the next geometrical entity.

The overall PN model development is shown schematically in Fig. 4 [1]. It involves six modules, representing one master and five slave PEs, belonging to a symmetric multiprocessor.

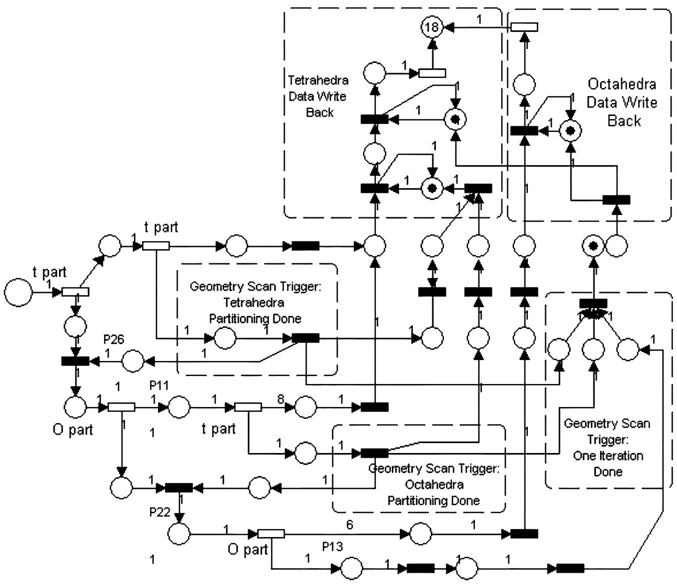
Tetrahedra Data Write Back
Octahedra Data Write Back
18
t part
t part
P26
Geometry Scan Trigger: Tetrahedra Partitioning Done
P11
t part
O part
Geometry Scan Trigger: Octahedra Partitioning Done
Geometry Scan Trigger: One Iteration Done
P22
O part
P13

Fig. 3. PN model of the Co-module (for one slave PE): tetrahedron and octahedron subdivision computations.

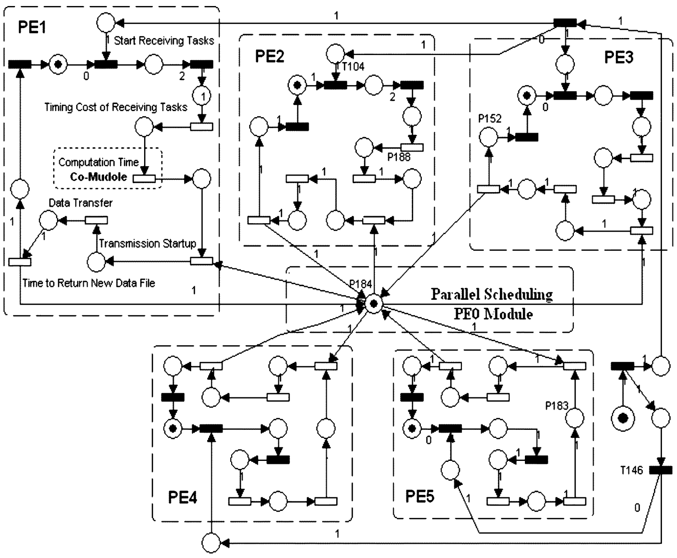
PE1
Start Receiving Tasks
Timing Cost of Receiving Tasks
Computation Time Co-Mudole
Data Transfer
Transmission Startup
Time to Return New Data File
PE2
T104
P188
PE3
P152
P184
Parallel Scheduling PE0 Module
PE4
PE5
P183
T146

Fig. 4. PN model for the parallel mesh refinement with six PEs.

The system parameters for program initiating, tetrahedron/octahedron partitioning and data transfer are each defined in the transition delays in each stage of the mesh refinement model. In Fig. 4, the "Co-Module" is abbreviated as a transition labeled "computation time."

## IV. Performance Validation Methodology

PN models are created as simplified representations of the parallel system at particular key points in time, which are critical to the specific meshing operation and functionality of the parallel system. The objective of a simulation is to facilitate the manipulation of the PN model in a manner appropriate for the way the parallel system would operate. A PN model is considered valid for a set of experimental conditions if its accuracy is within an acceptable range required for the intended application [8]. Therefore, validating a PN model, typically, requires comparing the input–output operations predicted by the model to the corresponding input–output operations of the system being modeled. To accomplish this for the target mesh refinement algorithm and parallel system configuration, it is necessary to compare and evaluate key performance measures from the PN simulations with corresponding benchmark computation results obtained from actual hardware implementations.

The benchmark program used to validate the PN simulation results was implemented using the hardware facilities at McGill University's CLUMEQ Supercomputer Centre. The parallel computing platform consists of 6 AMD Athlon 1900+ running at 1.6 GHz with 1.5 GB RAM, using a Myrinet-2000 Switch. The computing architecture used in this work is modeled based on the Message Passing Interface (MPI) for multiprocessors. The multiprocessors operate independently, however they share the same memory resource; MPI is the platform-independent communications library used to manage all aspects of internode communications and data transfers.

The model assumptions are based on the actual hardware environment used to perform the MPI benchmark computations. In the PN model, all application characteristics, such as real-time throughput and CPU timing statistics, were obtained from the CLUMEQ menu. Other required system parameters, including data transmission speed and communication startup time, were obtained experimentally with a separate, simple test code. These values are used to configure the parameters in the PN model, and all of the simulation results are derived from this PN model.

Table I provides the pseudocode for the MPI benchmark algorithm. All tests were conducted using the HTO mesh refinement model starting from one single tetrahedron, so they may more closely reflect what an individual task might be able to achieve in normal use. To eliminate the usual variation in run times that is common on some parallel systems, each test was run a minimum of three times and the average was used. Finally, we observed the variation in recorded wall-clock time for each test measurement was decreased after, approximately, the second round of HTO meshing. This may be due, in part, to the increasing data size and the statistical effects of caches or memory allocation for internal buffers being reduced [7].

## V. Results

The validation of the PN model is approached in three main ways, by examining three procedures of the parallel processes, as described by Figs. 5–7. The performance characteristics of the PN model simulations are compared with the results from the benchmark program. In particular, the time consumptions of each PE, for each aspect of the geometry computation and communication, predicted by the PN model are compared with actual time measurements of the MPI benchmark program.

The validation results for the mesh refinement computation process over five iterations, ranging from 512 to 2 097 152 elements, are provided in Fig. 5(a)–(e) for each PE in the parallel system. The master PE(1) does not have a mesh computation workload; its duty is to initiate slave PE processes at the start of each iteration, and therefore the time costs of the master PE are approximately the same for each iteration. Slave PEs (2)–(5) show similar computation times for each iteration, since their workloads are perfectly balanced. However, slave PE(6) is initially assigned one octahedron to refine, which results in a larger workload than the other slave PEs during subsequent iterations, and therefore its time costs are higher.

TABLE I
Pseudocode for MPI Benchmark Program

```
{Define time variables, initial MPI}
    double starttime, time0, time1,time2, time3, endtime;
      MPI_ Init();
{Start timing the parallel program}
      starttime=MPI_Wtime();
{Initial tetrahedron partition for sub-domain distribution}
      if(mype==0)
      Input Vertexes; Depth number i; Do  initial partition;
      MPI_Bcast();
{All PEs are synchronized to start computation}
      MPI_Barrier(MPI_COMM_WORLD);
{Measure computation time for each slave PE}
  time0 = MPI_Wtime()
          do Depth = i;
          if (mype == 1) then
             processTetra(t(pe1), Depth);
             processOcta (O(pe1), Depth);
                   .... ....
          if (mype ==6) then
             processTetra(t(pe6), Depth);
            processOcta (o(pe6), Depth);
          endif
          enddo
  time1 = MPI_Wtime()
{All PEs are synchronized for next step data gathering}
      MPI_Barrier(MPI_COMM_WORLD);
{Measure data gathering time between mater PE and each slave PE}
  time2 =MPI_Wtime()
         if(mype==0) then
         MPI_Gatherv (vertexes);
         MPI_Gatherv (faces)
         endif
  time3=MPI_Wtime()
         Format the vertexes by offsets
        Do print new data file
      endtime = MPI_Wtime()
{Report benchmark results}
      printf("Refinement Time:", mype, time1-time0);
     printf("Gather Time:", mype, time3-time2);
     printf("Formatting time:", endtime-time3);
     printf("Total Time:", mype, endtime-starttime)
```

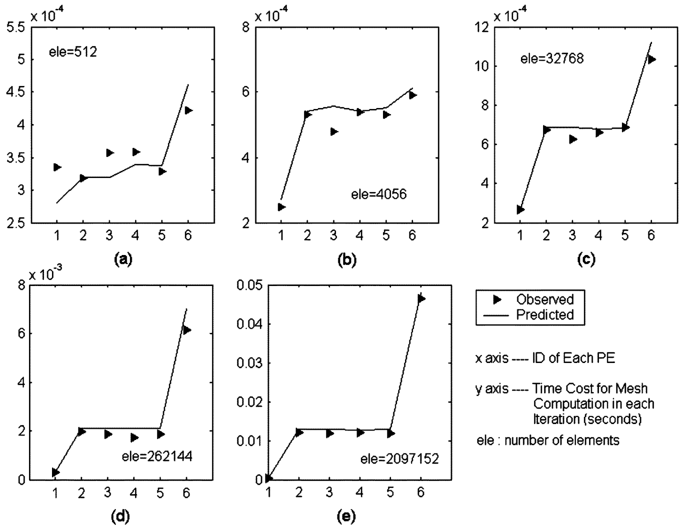

Fig. 5. Validation results for five iterations of the mesh computation processes ranging from (a) 512 elements to (e) 2 097 152 elements.

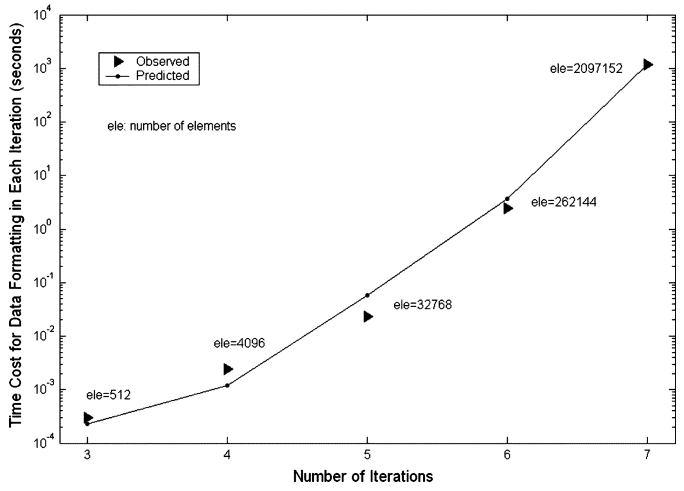

Fig. 6. Validation results for data formatting processes over five iterations.

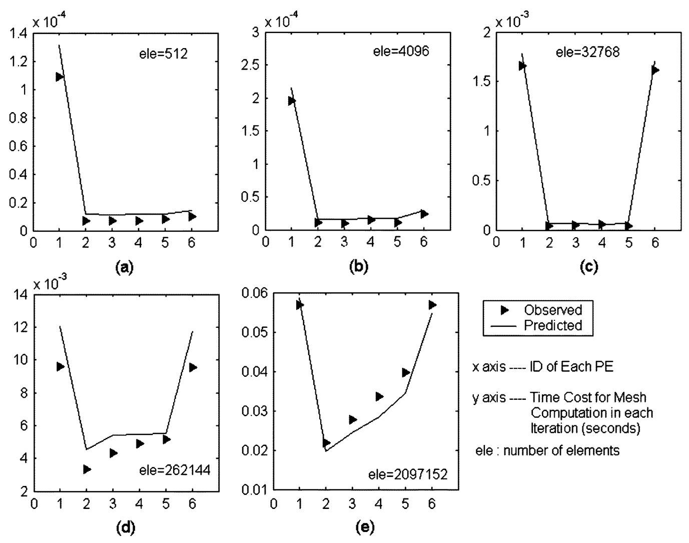

Fig. 7. Validation results for five iterations of data gathering processes ranging from (a) 512 elements to (e) 2 097 152 elements.

The validation results for the data formatting processes are shown in Fig. 6, in which the master and slave PEs work to write the result data of the mesh refinement computation into Object File Format (OFF) files, to provide a graphical data file for each slave PE. During this phase of operations, the master PE will make every slave PE wait until it completes fixing the coordinator offset for each slave PE; then the master releases all the slave PEs simultaneously. Therefore, during this phase the time cost of every PE in the system is the same since they operate synchronously in this manner.

The validation results for the data gathering processes over five iterations are shown in Fig. 7(a)–(e). In this stage, the master PE gathers data from each slave PE and builds the final data file. Each slave PE sends its data to the master PE asynchronously, and the master will wait until the last communication is completed; usually the last one is from PE(6) because it has the largest workload. In Fig. 7(a) and (b), the master PE(1) spends much more time than PE(6) because in iterations 1 and 2 the data size is relatively small and the time spent for initiating slave PE processes are significantly longer than the actual data communication times.

Figs. 5–7 all show close agreement between the timing results for the PN model simulations and the corresponding MPI benchmarks for a series of five mesh refinement iterations ranging from 512 to 2 097 152 elements. The discrepancies between the simulation and benchmark results may be due, in part, to the fact that the potential effects of the processor cache, MPI overheads, and the cluster pool is not accounted for in the PN models.

## VI. Conclusion

The specific methodology for validation of PN-based modeling and simulation introduced in this work was demonstrated for parallel 3-D FEM mesh refinement design and development. Overall, the series of comparative investigations examined confirm strong agreement between the simulated performance results and the actual system performance results, and together serve to validate the correctness of the PN modeling scheme. These successful findings suggest further development and experimental studies are merited. For example, the focus of future work should include incorporating potential processor cache, MPI overheads, and cluster pool effects in the PN models in order to further improve the accuracy provided by this methodology for performance modeling and simulation validation for parallel 3-D FEM mesh refinement.

## 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, "A preliminary approach to simulate parallel mesh refinement with petri nets for 3D finite element electromagnetics," in *Proc. ANTEM*, 2004, pp. 127–130.
[2] 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.
[3] 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.
[4] G. Greiner and R. Grosso, "Hierarchical tetrahedral-octahedral subdivision for volume visualization," *Vis. Comput.*, vol. 16, no. 6, pp. 357–369, Oct. 2000.
[5] S. Wang, J. Qiu, Q. Li, J. G. Zhu, and S. Wang, "Application of petri net in development of finite element analysis package for electromagnetic fields," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1255–1258, Apr. 2006.
[6] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Berlin, Germany: Springer-Verlag, 2002.
[7] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, 2nd ed. New York: Addison-Wesley, 2003.
[8] R. G. Sargent, "Verification and validation of simulation models," in *Proc. Winter Simulation Conf.*, 1998, pp. 121–130.

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