An Optimized Dynamic Load Balancing Method for Parallel 3-D Mesh Refinement for Finite Element Electromagnetics with Tetrahedra

Da Qi Ren, Dennis D. Giannnacopoulos, Reiji Suda

THE UNIVERSITY OF TOKYO

Introduction

An optimized Dynamic Load Balancing (DLB) method for parallel, adaptive, 3-D mesh refinement is developed based on study of characteristics of FEM on electromagnetics with tetrahedra;

Task pool locations and the initial data assignments are optimized in multiprocessor parallel architecture;

By comparing the benchmark results derived from other two existing DLB algorithms, the benefits of the optimized method for achieving high performance parallel mesh refinement are demonstrated.

DLB in Mesh Refinement

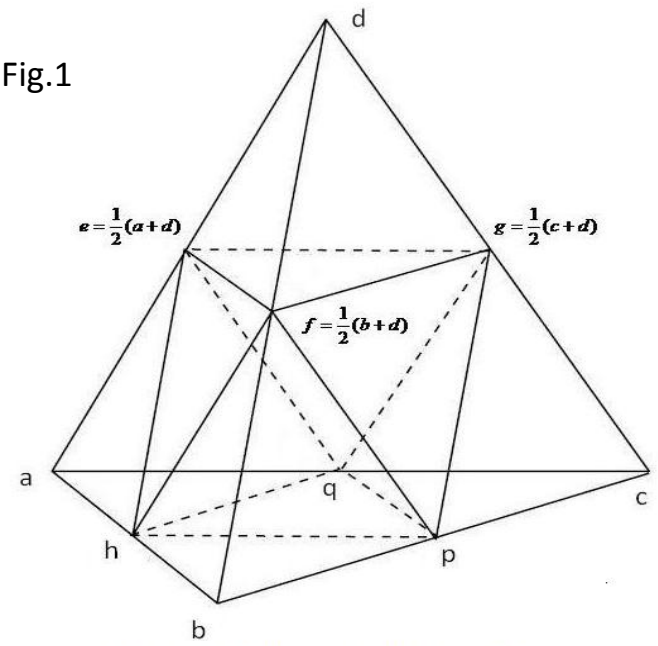
A DLB approach may not be the optimal without studying the characteristics of tasks;

Efficient and accurate solutions for some 3-D FEM applications require extremely large numbers of elements. Parallel processing is beneficial for each stage of the FEM including mesh refinement;

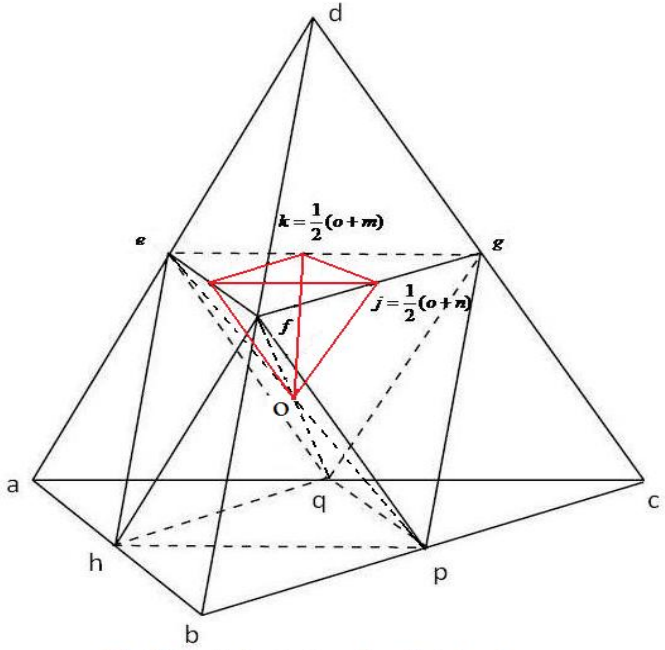
Due to the computational complexity of 3-D, parallel mesh generation and refinement, the computation performance is highly dependent on several factors including dynamic load balancing.

HTO Mesh Refinement

Fig.1

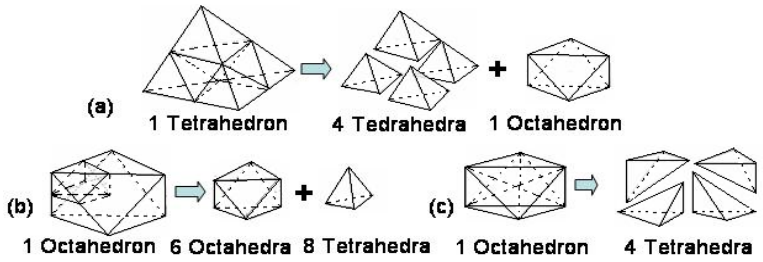


(a) Edge Calculation for Tedrahedron



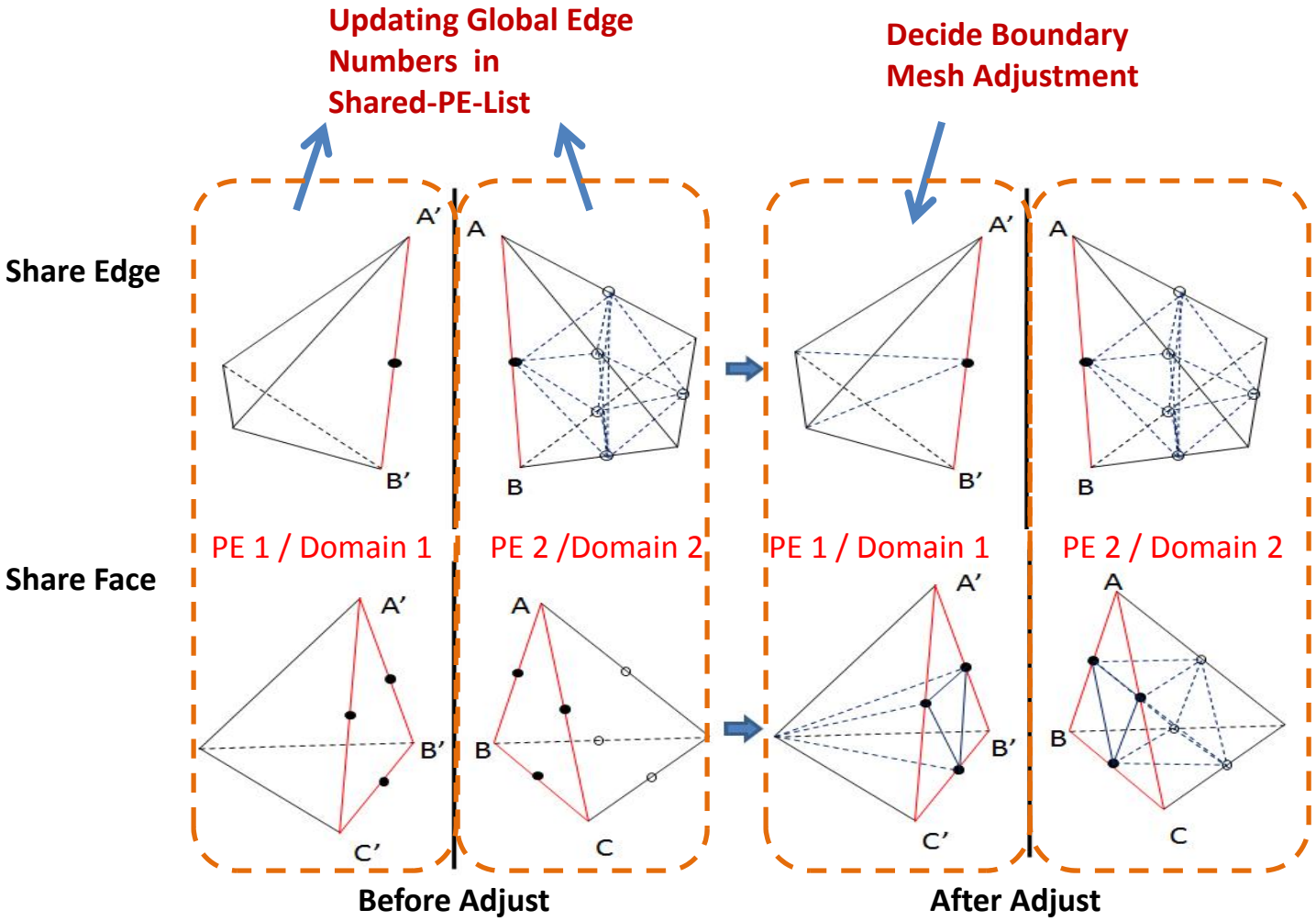
(b) Edge Calculation for Octahedron

Fig.2



THE UNIVERSITY OF TOKYO

Boundary Between two sub-domain



Inter-processor Communications

Elements have to be continuously upgraded to the allowed subdivision patterns.

Some propagation of edges happen that could mark local copies of shared edges inconsistently.

Communication required after each iteration of the propagation process.

Load Balancing Methods

The redistribution of load among the processors during execution time is performed in order to make each processor have the same or nearly the same amount of work load.

Two typical methods :
Centralized pool method, one control processor has all the incomplete works saved in a central task pool, and an idle processor asks this fixed processor for more work.

Distributed pool method, all the work is initially distributed among different processors, and an idle processor selects a peer processor as the work donor by using a DLB method.

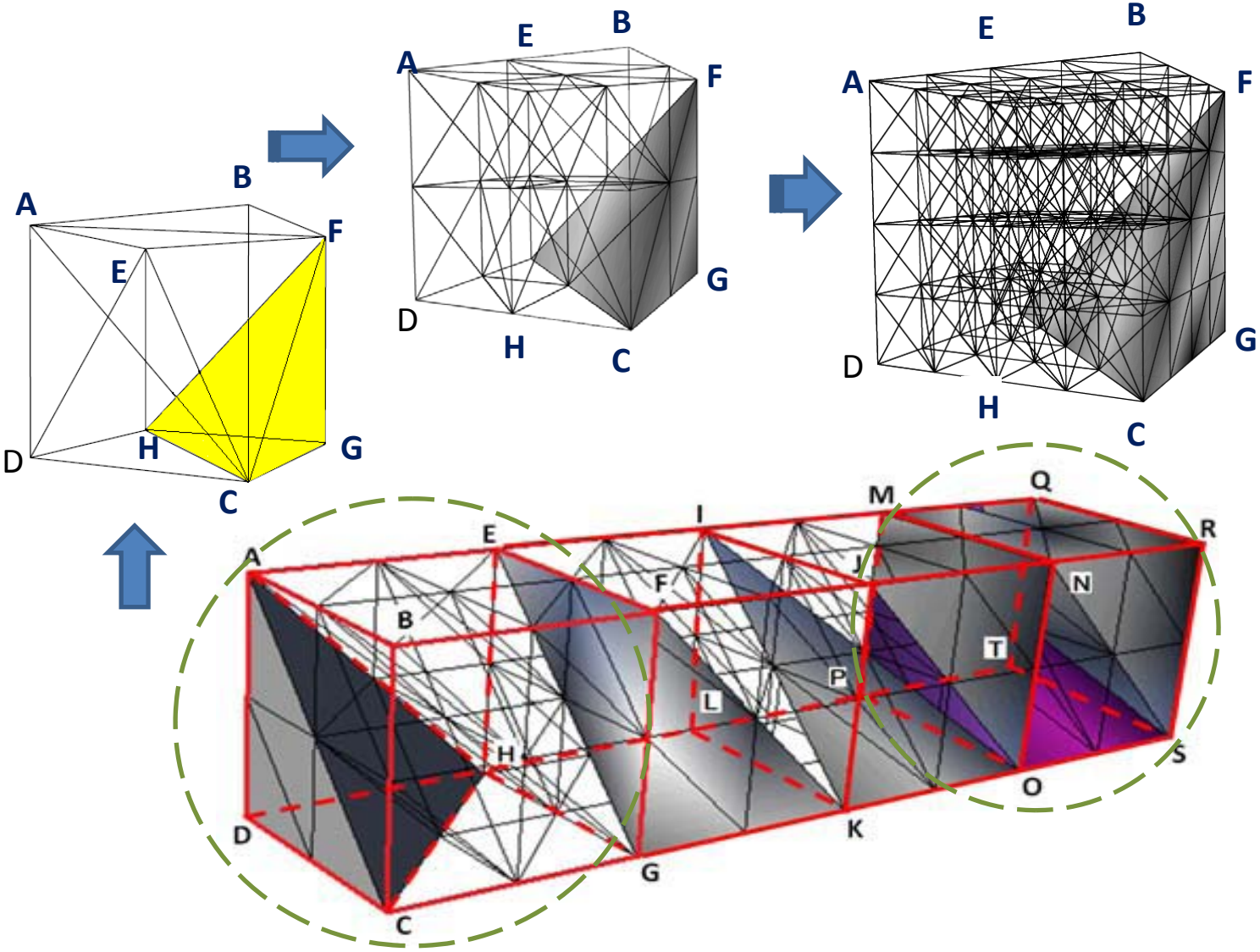
Atomic Task Elements

An optimization in HTO mesh DLB is multilevel control on each sub-domain. i.e., by further partition each sub-domain into reasonable sized smaller domains (sub-sub-domains);

These sub-sub-domains will work as atomic task elements (cannot be further split) during the parallel DLB process, it can be transferred between the parallel processors for load balancing purpose.

In such hierarchical domain management a PE is in charge of one sub-domain including a set of sub-sub-domains.

Hierarchical Working Domain



Determine the size of a Sub-sub-domain

Large number and smaller sized task elements can make load adjustment finer;

But increasing the number of atomic task elements could seriously deduct the performance benefit procured from the load balancing algorithm;

A mathematical estimation for producing reasonable sized sub-sub-domains that compromises the tradeoff between dynamic load balancing and the FEM computation and communication expense.

$$t_i^{comp}(p_k) > t_i^{split}(p_k) + t_i^{comm}(p_k) + t_i^{ini}(p_k) = \tau$$

Dynamic Load Balancing in HTO FEM

Task Pools Distribution for 4 PEs						
PE#	PE1	PE2	PE3	PE4		
Tasks	ACDH ABCH ABEH BCGH BEGH BEGF	EGHL EFGL EFIL FGKL FIKL FIKJ	IKLP IFKP IFMP JKOP JMOP JMON	MOPT MFOT MFQT NKST NQST NQSR		
Task Pools Distribution for 6 PEs						
PE#	PE1	PE2	PE3	PE4	PE5	PE6
Tasks	ACDH ABCH ABEH BCGH	EGHL EFGL EFIL FGKL	IKLP IFKP IFMP JKOP	MOPT MFOT MFQT NKST	BEGH BEGF FIKL FIKJ	JMOP JMON NQST NQSR
Task Pools Distribution for 8 PEs						
PE #	PE1	PE2	PE3	PE4	PE5	PE6
Tasks	ACDH ABCH ABEH	EGHL EFGL EFIL	IKLP IFKP IFMP	MOPT MFOT MFQT	BEGH BEGF FIKL	JMOP JMON NQST
PE#	PE7	PE8				
Tasks	BCGH FGKL JKOP	NKST FIKJ NQSR				

Task Pools Distribution for 12 PEs						
PE#	PE1	PE2	PE3	PE4	PE5	PE6
Tasks	ACDH ABCH	ABEH EGHL	EFGL EFIL	IKLP IFKP	IFMP MOPT	MFOT MFQT
PE#	PE7	PE8	PE9	PE10	PE11	PE12
Tasks	BEGH BEGF	FIKL JMOP	NQST JMON	BCGH FGKL	JKOP NKST	FIKJ NQSR
Task Pools Distribution for 24 PEs						
PE#	PE1	PE2	PE3	PE4	PE5	PE6
Tasks	ACDH	ABEH	EFGL	IKLP	ABCH	EGHL
PE#	PE7	PE8	PE9	PE10	PE11	PE12
Tasks	EFIL	IFKP	IFMP	MFOT	BEGH	FIKL
PE#	PE13	PE14	PE15	PE16	PE17	PE18
Tasks	MOPT	MFQT	BEGF	JMOP	JMON	BCGH
PE#	PE19	PE20	PE21	PE22	PE23	PE24
Tasks	JKOP	FIKJ	NQST	FGKL	NKST	NQSR

DLB Algorithms

Centralized

1. A centralized task pool is defined to store all sub- sub-domain data;
2. Each slave PE fetches one task element from the central task pool;
3. Once a PE finishes its local computation, it will visit the central task pool to take next task element.

Random Polling

1. Task pools are defined one for each slave PE and stored tasks assigned to that PE ;
2. Each slave PE fetches task element from its local task pool;
3. If a PE's local task pool is empty, it will repeat sending a request R to other randomly determined PE;
4. if R is not rejected, half of the remaining tasks from the "polled" PE will be copied to the "polling" PE.

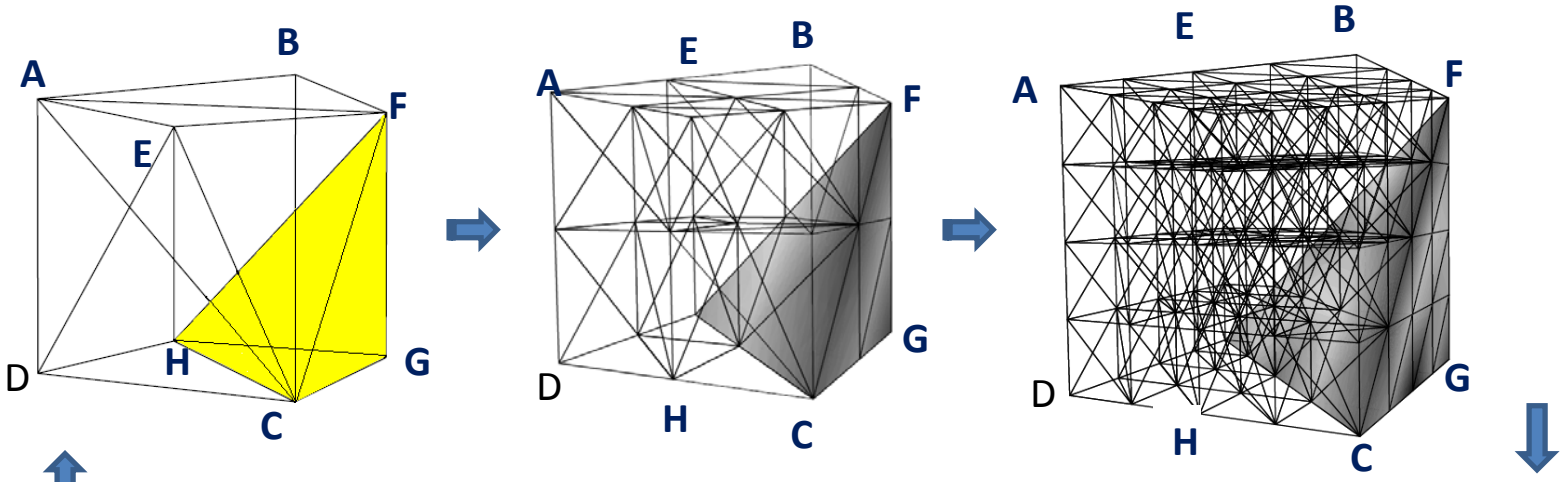
Enhanced Dynamic Load Balancing Algorithm

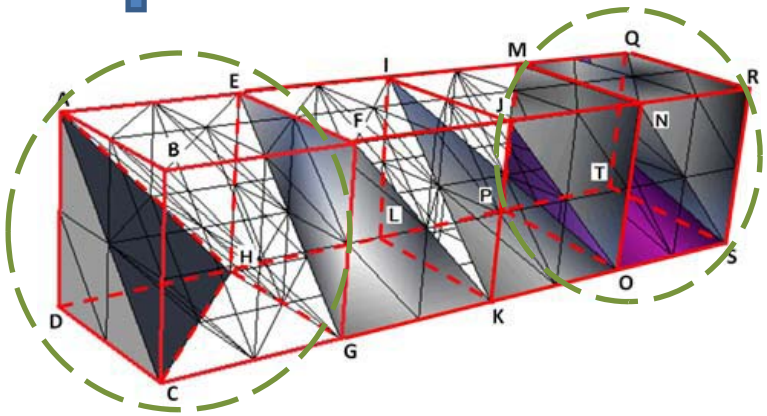
1. Defines task pools one for each slave PE, located in master PE0;

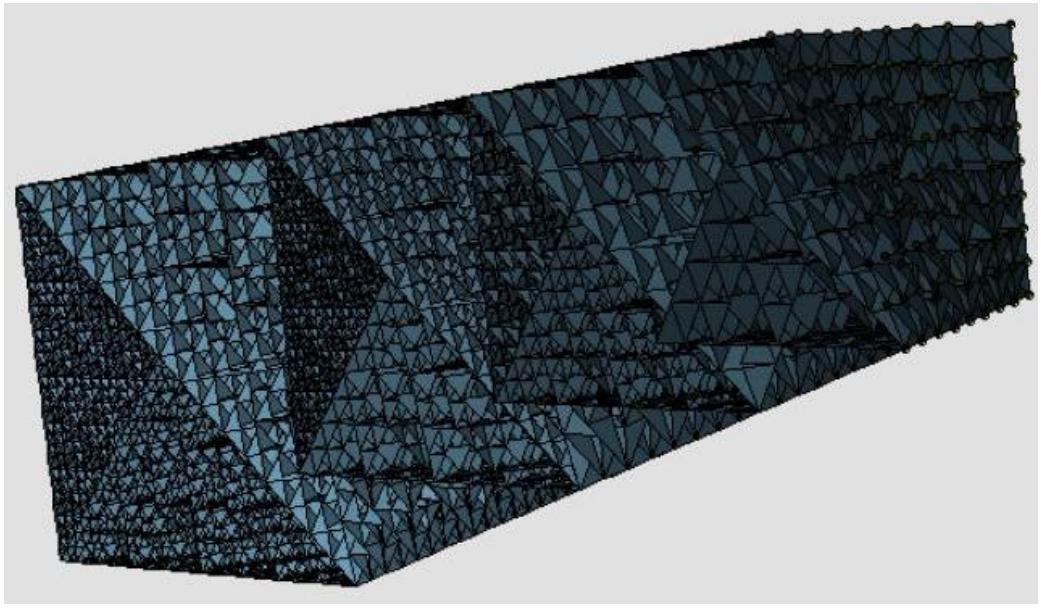
2. Each slave PE fetches task elements from its own task pool located in PE0, then run its local computation;

3. If a PE's own task pool is empty, it will randomly select another PE's task pool and take out one of the tasks if the target task pool is not empty.

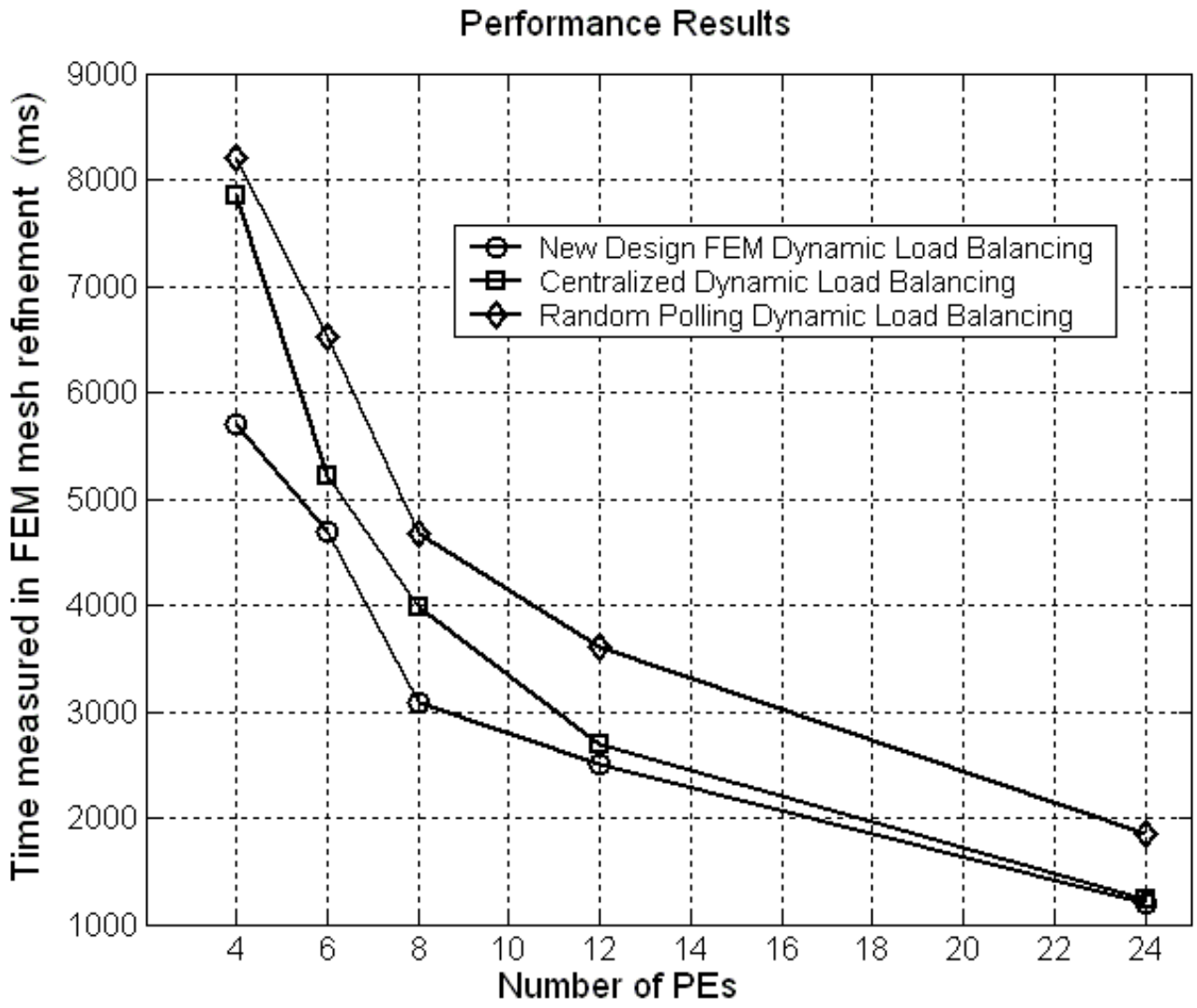
Hierarchical Working Domain







Dynamic Load Balancing in HTO FEM



Results

In HTO mesh refinement (Refer to the example) , the number of task elements is limited ;

Once a PE grabs a sub-sub-domain, it will never share this work with other PEs. Thus in the worst case all other PEs are idle and waiting for one PE computing a big element task;

The speed-up depends on the initial sub-sub-domain grouping and the task element assignments crossing each different slave PE;

It is still not necessarily to have the same performance result because the DLB operations are random in each experiment iteration.

Conclusion

In this paper the DLB design and performance analysis are, typically, based on benchmarking programs on known environments.

One promising route is to create modeling and simulating models that allow for a relatively detailed description of a system by formal syntax and functional semantics, and can reveal key characteristics of system performance stochastically.

Thank you!