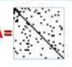
Task-Parallel Domain-wise Multifrontal Solver for High-Performance Finite Element Analysis on Multi-core Systems

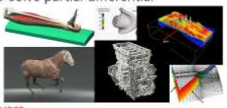
Jeong Ho Kim, Kyu Beom Lee, Jin Yeon Cho
Inha University, Korea

Introduction

- **Sparse Solvers**
  - Solve sparse linear systems for numerical analysis

    Solve $Ax=b$ or $AX=B$ where A is a sparse matrix

    $A=$ 
  - Sparse direct solvers
    - Direct factorization considering sparsity (fill-in inevitable)
    - **Numerical robustness**, efficient manipulation of multiple RHS
  - Sparse iterative solvers
    - Iterative computations of matrix-vector operations
    - Converging speed affected by matrix condition (may fail to converge)
- **Finite Element Method (FEM)**
  - Most widely used numerical analysis method to solve partial differential equations
  - Major application fields
    - Computational Structural Mechanics
    - Computational Fluid Dynamics
    - Computer Graphics (Physics-based sim.)
  - Most of computing time is spent on sparse solvers

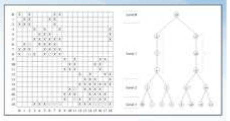


⇨ **Efficient sparse direct solver is required to obtain reliable solutions for problems from various fields**

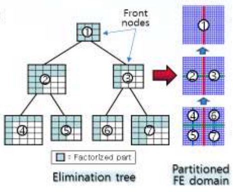
INHA UNIVERSITY

Domain-wise Multifrontal Solver

- **Multifrontal solver**
  - Proposed by Duff & Reid (1983)[1] as a generalization and extension of frontal method
  - Direct solver for general sparse matrices
  - A generalized element consists of non-zero terms in single row or column
  - Computation performed on partial dense matrices for nodes in elimination tree



- **Domain-wise Multifrontal Solver[2] (DMFS)**
  - Introducing the original frontal concept (based on finite elements) to multifrontal method
  - Use finite elements instead of generalized elements of multifrontal method
  - Elimination tree from nested dissection is mapped to partitioned finite element domain
  - Global matrix never constructed, and element matrices are computed by callback functions



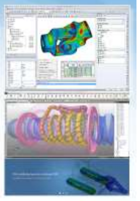
⇨ Advantage of DMFS in parallelizing the whole finite element analysis procedure

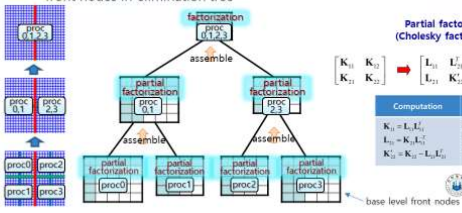
1) Duff, S., and Reid, J. K., "The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Transactions on Mathematical Software*, Vol. 9, 1973, pp. 302-325
2) Kim, J. H., Lee, C. S. and Kim, S. J., "High-performance Domain-wise Parallel Direct Solver for Large-scale Structural Analysis," *AIAA Journal*, Vol. 43, No. 3, pp.662~670, 2005

INHA UNIVERSITY

Domain-wise Multifrontal Solver

- **Application of Domain-wise Multifrontal Solver**
  - Commercial FEM codes by MIDAS IT (http://www.midasit.com)
  - RecurDyn (http://www.functionbay.com)
  - DAFUL (http://www.virtualmotion.co.kr)
  - Seismic migration codes in TOTAL
  - Research FEM codes such as IPSAP (http://ipsap.snu.ac.kr/)
- **Parallel implementation of original DMFS**
  - Both shared (POSIX threads or Windows threads) & distributed (MPI) memory version available
  - Static scheduling by distributing processors evenly to child front nodes in elimination tree





Partial factorization (Cholesky factorization)

$$\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix} \Rightarrow \begin{bmatrix}\mathbf{L}_{11} & \mathbf{L}_{21}^T\\ \mathbf{L}_{21} & \mathbf{K}'_{22}\end{bmatrix} \qquad \begin{aligned}\mathbf{K}_{11}&=\mathbf{L}_{11}\mathbf{L}_{11}^T\\ \mathbf{L}_{21}&=\mathbf{K}_{21}\mathbf{L}_{11}^{-T}\\ \mathbf{K}'_{22}&=\mathbf{K}_{22}-\mathbf{L}_{21}\mathbf{L}_{21}^T\end{aligned}$$

Computation	Subroutine (MKL)
$\mathbf{K}_{11}=\mathbf{L}_{11}\mathbf{L}_{11}^T$	DPOTRF (LAPACK)
$\mathbf{L}_{21}=\mathbf{K}_{21}\mathbf{L}_{11}^{-T}$	DTRSM (BLAS)
$\mathbf{K}'_{22}=\mathbf{K}_{22}-\mathbf{L}_{21}\mathbf{L}_{21}^T$	DSYRK (BLAS)

INHA UNIVERSITY

Parallelization of DMFS

- **Limitations in original parallel DMFS**
  - Work loads of sibling nodes in elimination tree are not equal in general
  - Load imbalance is inevitable for static scheduling

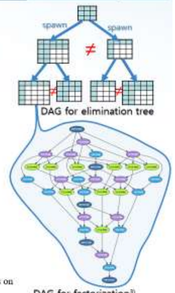
  ⇨ Dynamic scheduling is required for scalability
- **Task parallelism in DMFS**
  - Dynamic scheduling of DMFS can be enabled by task parallelism
  - Implemented by using Thread Building Blocks (TBB)
  - Task parallelism in elimination tree
    - Tasks for parent nodes are dependent on tasks for their child nodes

    ⇨ Parent nodes spawn tasks for their child nodes
  - Task parallelism in partial factorization
    - DAG(Directed Acyclic Graph) based task parallelism[3] for blocked matrices
    - All tasks for blocked matrices are subtasks of node tasks and have multiple dependencies

    ⇨ Reference counts for tasks are used to spawn tasks for blocked matrices

Thread Building Blocks (TBB)

What is it?	Widely used C++ template library for task parallelism
Primary features	• Parallel algorithms and data structures • Scalable memory allocation and task scheduling



DAG for factorization[3]

3) Kurzak, J., Ltaief, H., Dongarra, J. and Badia, R. M., "Scheduling Linear Algebra Operations on Multicore Processors", *LAPACK Working Note 213*, UT-CS-09-636, February 5, 2009

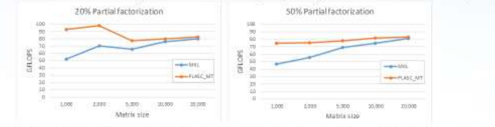
Parallelization of DMFS

- **Performance Issues for task-parallel DMFS**
  - Performance of partial factorization
  - Number of front node tasks spawned
  - Scheduling of spawned tasks : by TBB

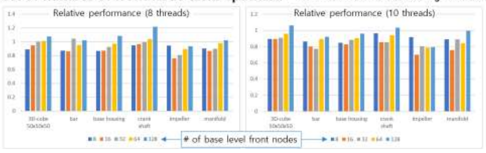
Task scheduling algorithm of TBB

1. The task returned by `t.execute()`
2. The successor of `t` if t was its last completed predecessor.
3. A task popped from the end of the thread's own deque.
4. A task with affinity for the thread.
5. A task popped from approximately the beginning of the shared queue.
6. …

- **Performance improvement for partial factorization**
  - Partial factorization by this work (PLASC_MT) vs. by Intel Math Kernel Lib. (MKL)

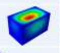


- **Effect of number of front node tasks spawned** Load imbalance ⇔ Scheduling overhead

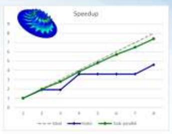


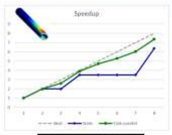
Performance of Task-Parallel DMFS

- **Sparse factorization performance**
  - static-scheduled DMFS vs. task-parallel DMFS on TBB (factorization time in seconds)

FE model	Scheduling	# of threads				
		1	4	6	8	10
	Static	1111	320	319	176	175
	Task-parallel		312	221	168	153
	Ratio		1.02	1.44	1.05	1.14
	Static	128.4	40.9	41.2	23.3	23.3
	Task-parallel		35.7	26.6	19.9	17.6
	Ratio		1.15	1.55	1.17	1.33
	Static	30.3	9.65	9.58	5.78	5.83
	Task-parallel		9.58	7.30	5.13	4.84
	Ratio		1.01	1.31	1.13	1.20
	Static	27.0	9.14	9.12	5.17	5.12
	Task-parallel		7.45	6.78	4.92	4.16
	Ratio		1.23	1.34	1.05	1.23
	Static	39.9	11.10	11.16	9.11	9.10
	Task-parallel		11.00	8.27	6.90	6.40
	Ratio		1.01	1.35	1.32	1.42
	Static	19.1	5.84	5.85	4.07	4.10
	Task-parallel		5.54	4.45	3.40	3.11
	Ratio		1.06	1.31	1.20	1.32

System Specification	
CPU	Intel i7-6950X (3.00GHz, 10 cores)
RAM	128GB





System Specification	
CPU	Intel Zeon E5-2698 (2.30GHz, 8 cores)
RAM	64GB

INHA UNIVERSITY

Performance of Task-Parallel DMFS

- **Performance of whole finite element analysis procedure**
  - HEMOS
    - Finite element analysis code developed by Inha University & KISTI
    - Both Pardiso (commercial sparse direct solver in MKL) and DMFS are available as sparse direct solvers
  - ABAQUS
    - Commercial software suite for finite element analysis by Dassault Systems®

Performance comparison of ABAQUS and HEMOS/Pardiso vs. HEMOS/DMFS

FE Model	# threads	HEMOS (DMFS)	ABAQUS(6.14)		HEMOS (PARDISO)	
		time(sec)	time(sec)	ratio	time(sec)	ratio
	1	136	288	2.12	192	1.41
	10	28	84	3.00	53	1.89
	1	41	137	3.34	67	1.63
	10	17	66	3.88	34	2.00
	1	49	162	3.31	69	1.41
	10	26	102	3.92	43	1.65
	1	38	151	3.97	58	1.53
	10	22	97	4.41	39	1.77
	1	60	188	3.13	104	1.73
	10	25	101	4.04	48	1.92

System Specification	
CPU	Intel i7-6950X (3.00GHz, 10 cores)
RAM	128GB

INHA UNIVERSITY

Result Analysis and Conclusions

- **Domain-wise multifrontal solver**
  - High-performance sparse direct solver based on multifrontal algorithm
  - Many successful applications to commercial and research FEM codes
  - Limited scalability due to static scheduling
- **Task-parallel domain-wise multifrontal solver**
  - Low overhead dynamic scheduling of spawned tasks is enabled by TBB
  - Task parallelism in DMFS can be realized just by spawning tasks with consideration for the size and dependency of each task
  - DAG based partial factorization by blocked factorization of leading diagonal blocks leads to performance improvements to multiple calls to MKL functions

  ⇨ Overall parallel performance has been improved significantly
- **Finite element analysis using DMFS**
  - outperforms ABAQUS up to 4 times
  - outperforms finite element analysis using Pardiso up to 2 times
  - shows better scalability

⇨ **The performance of the task-parallel DMFS is much superior to state-of-the-art finite element analysis codes and sparse solvers**

- **Further Work**
  - Performance evaluation and optimization for manycore processors
  - Affinity control for better performance on ccNUMA systems

INHA UNIVERSITY