ADVENTURE の次世代スパコン向けチューニングの現状と課題

河合浩志
東京大学大学院工学系研究科

ADVENTURE の次世代スパコン向けチューニングの現状と課題

河合浩志、荻野正雄、塩谷隆二、吉村忍

東京大学大学院工学系研究科システム創成学専攻
〒113-8656 東京都文京区本郷 7-3-1
kawai@save.sys.t.u-tokyo.ac.jp

１．ＡＤＶＥＮＴＵＲＥと領域分割法の概要

本研究で用いるＡＤＶＥＮＴＵＲＥシステムでは、その並列処理手法として主に領域分割法(Domain Decomposition Method：DDM)が用いられている。ＡＤＶＥＮＴＵＲＥで用いられているＤＤＭでは、解析領域を互いに重なり合わない複数の部分領域に分割し、部分領域間のインターフェイス問題をＣＧ法などを用いて反復的に解く。これは反復型サブストラクチャリング法とも呼ばれ、非オーバーラップ型のＤＤＭに分類される。ここで行なわれる部分領域ごとのＦＥＭ計算は独立に実行可能なため、DDM は並列計算、特に分散メモリ型超並列計算機やＰＣクラスタとの相性がよい。

２．部分領域ごとＦＥＭ計算のマルチコアＣＰＵ向け高速化

ＤＤＭの性能チューニングにおけるホットスポットは部分領域ごとに行なわれる有限要素解析である。このＦＥＭ計算を近年普及しつつあるマルチコアＣＰＵ向けに最適化することを考える。特に、キャッシュの利用効率を向上させ、同時にＳＩＭＤ拡張命令の利用を考慮する。部分領域ＦＥＭ計算では、これを直接法または反復法ソルバーで解くこと、およびメモリ保存型か省メモリ型かという観点から、主に次の４種類、Direct solver-based matrix Storage (DS)、Direct solver-based matrix Storage-Free (DSF)、Iterative solver-based matrix Storage (IS)、Iterative solver-based matrix Storage-Free (ISF)、に分類される。ＤＳは従来型の、あらかじめ剛性行列の行列分解を行なっておいてこれをメモリ上に保存し、求解時には前進後退代入だけですます方法である。これに対しメモリをあまり用いない、あるいはメッシュ以外メモリに保存しないＤＳＦ，ＩＳ，ＩＳＦは、より大規模な解析を可能とする。さらに、メニーコアのようなメモリバンド幅が相対的に弱い計算機環境では、ＤＳよりこれらが高速になる可能性もある。

３．ＢＤＤ前処理と並列直接法ソルバー導入によるＢＤＤ前処理の高速化

DDMを三次元複雑構造物の解析に応用するには、強力な前処理が不可欠である。ADVENTURE システムではBDD (Balancing Domain Decomposition)前処理を導入してこれに対処している。BDD 前処理はノイマン・ノイマン前処理とコースグリッド修正より構成される。今回はこのうち、並列化においてボトルネックとなりやすい、後者のコースグリッド修正に対する高速化を行なった。コースグリッド修正では中規模な連立一次方程式を解く必要があるが、これに対して並列直接法ソルバーを用いて分散メモリ計算機上でより大規模なコース問題を高速に解くことをねらう。現在の段階で25万領域、150万自由度のコース問題を扱うことが可能となっている。これに前記の領域FEM計算高速化を組み合わせ、BDDソルバーの大規模薄板問題ベンチマークを行なった。東大T2Kスパコンで4096コアを使用して、9億自由度問題を4分程度で求解することに成功した。

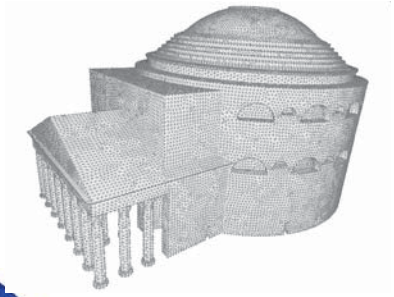
４．次世代スパコンに向けたロードマップと今後の課題

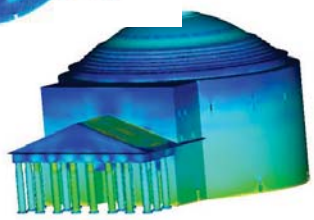
現在の段階で、大規模PCクラスタを用いて1億自由度モデルを数十秒、10億自由度モデルを数分で解くことに成功している。次世代スパコン「京」では100億自由度クラスの超大規模問題を狙う。これを同様に数分程度で求解することができれば、原子力発電プラントの三次元ソリッドフルモデル（建屋と圧力容器を含む）に対する耐震解析を約一日程度で行なうことができるようになる。このためには、DDM においてさらなる領域サイズの増大と領域総数の増加に対処し、また次世代スパコンのハードウェア環境に向けた性能チューニングが必要となる。

ADVENTUREの次世代スパコン向け チューニングの現状と課題

○ 河合浩志（東大）
荻野正雄（九大）
塩谷隆二（東洋大）
吉村忍（東大）

ADVENTURE：オープンソース CAEシステム



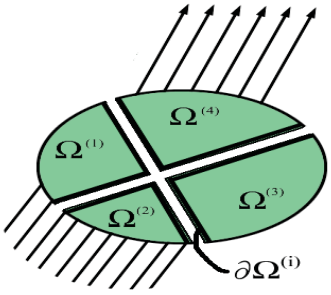


モジュールベース アーキテクチャ
CAD接続，メッシュ生成
解析コード（有限要素法），
可視化，設計最適化，
連成カプラ etc.

http://adventure.sys.t.u-tokyo.ac.jp

It’s open-source !

並列有限要素解析手法の分類



- 領域分割法(反復型サブストラクチャ法)

  ex.) DDM

  前処理: BDD, FETI, ...

  特徴: メッシュを元に領域分割する

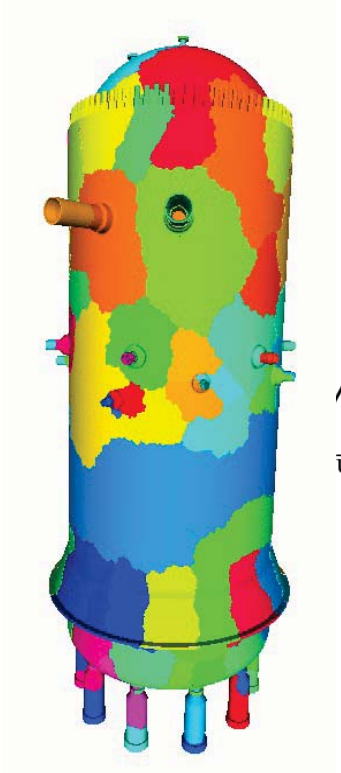
- (反復法で)行列ベクトル積などを並列化

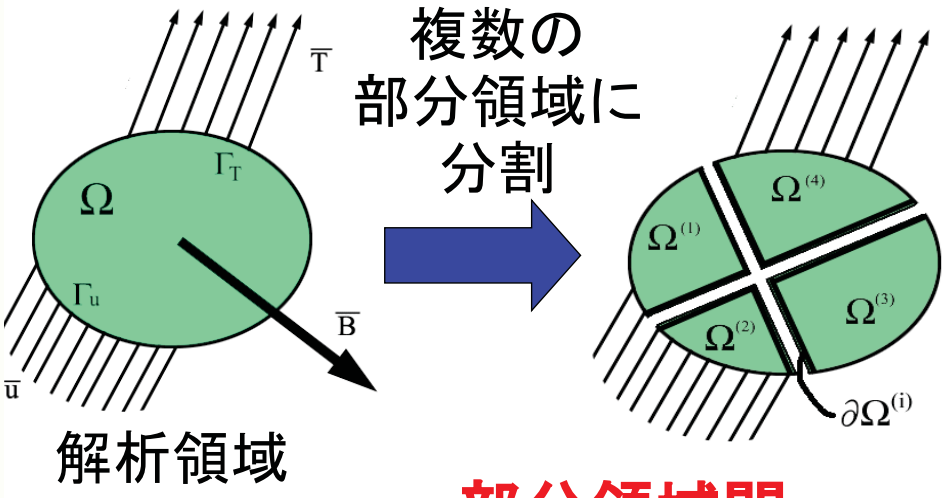
  ex.) 並列CGソルバー

  前処理: ブロックIC, 近似逆行列、AMG, ...

  特徴: 主に行列の情報だけを用いる

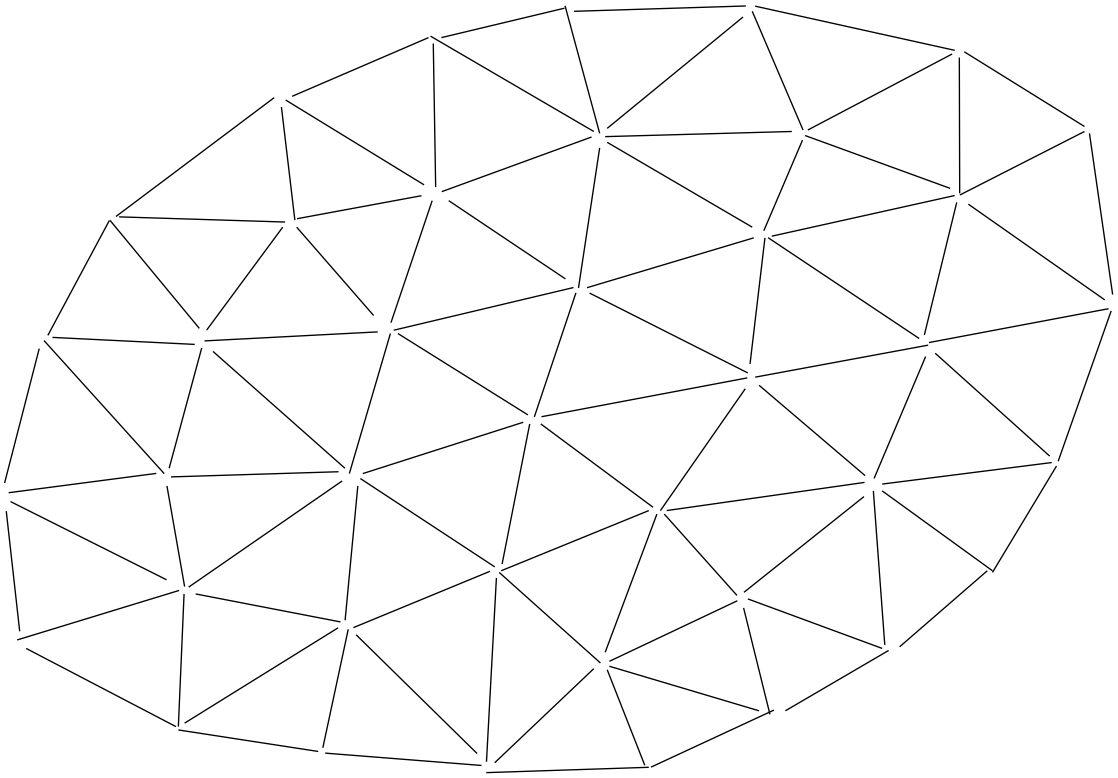
領域分割法
Domain Decomposition Method (DDM)



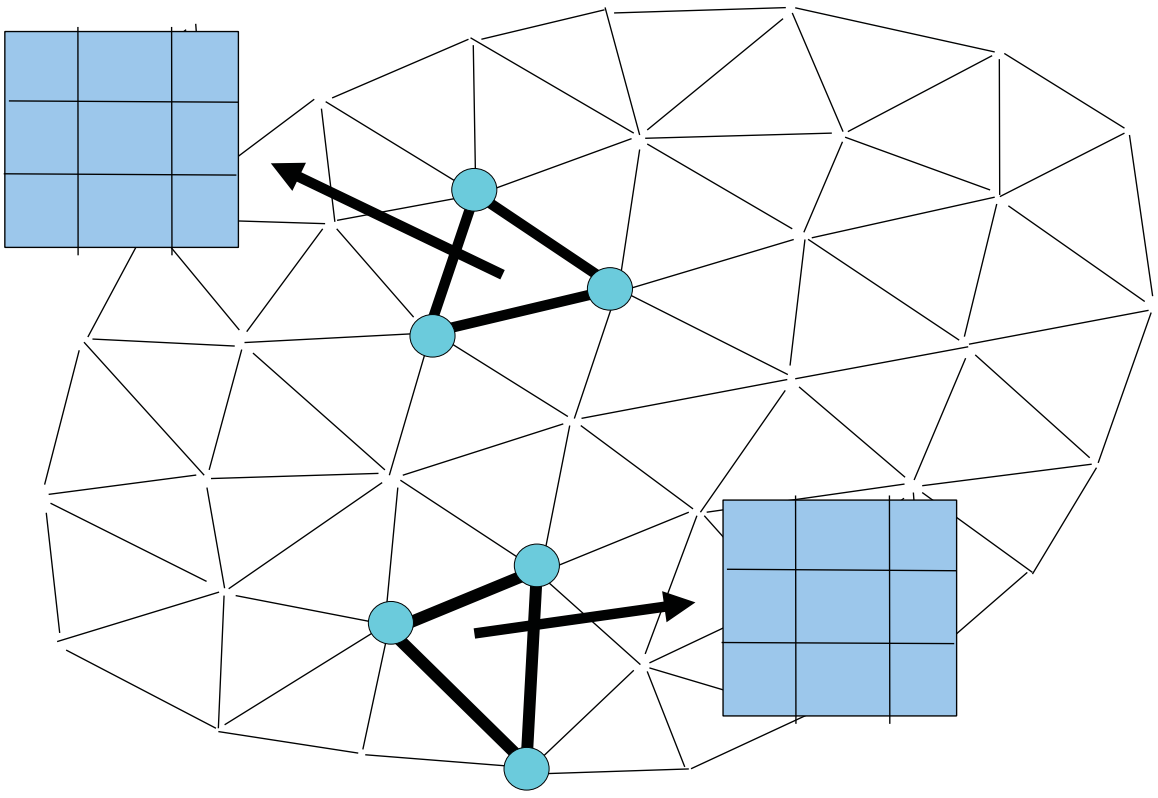


解析領域

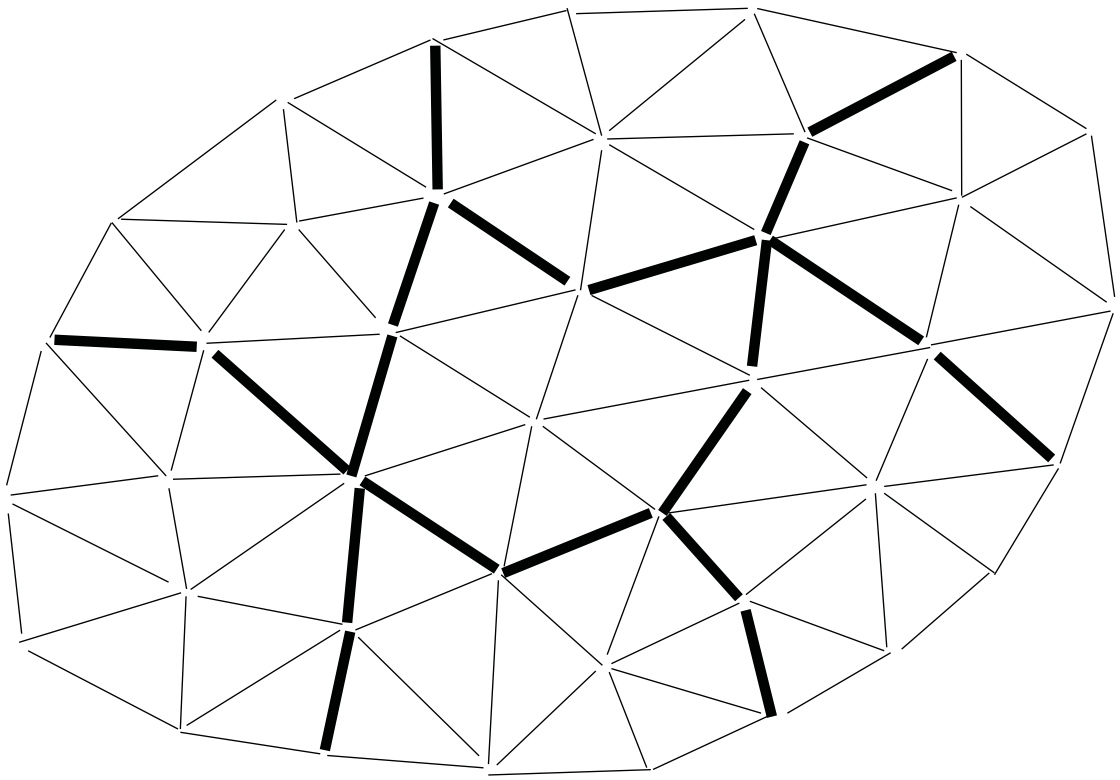
部分領域間
インターフェイス
問題に帰着させる



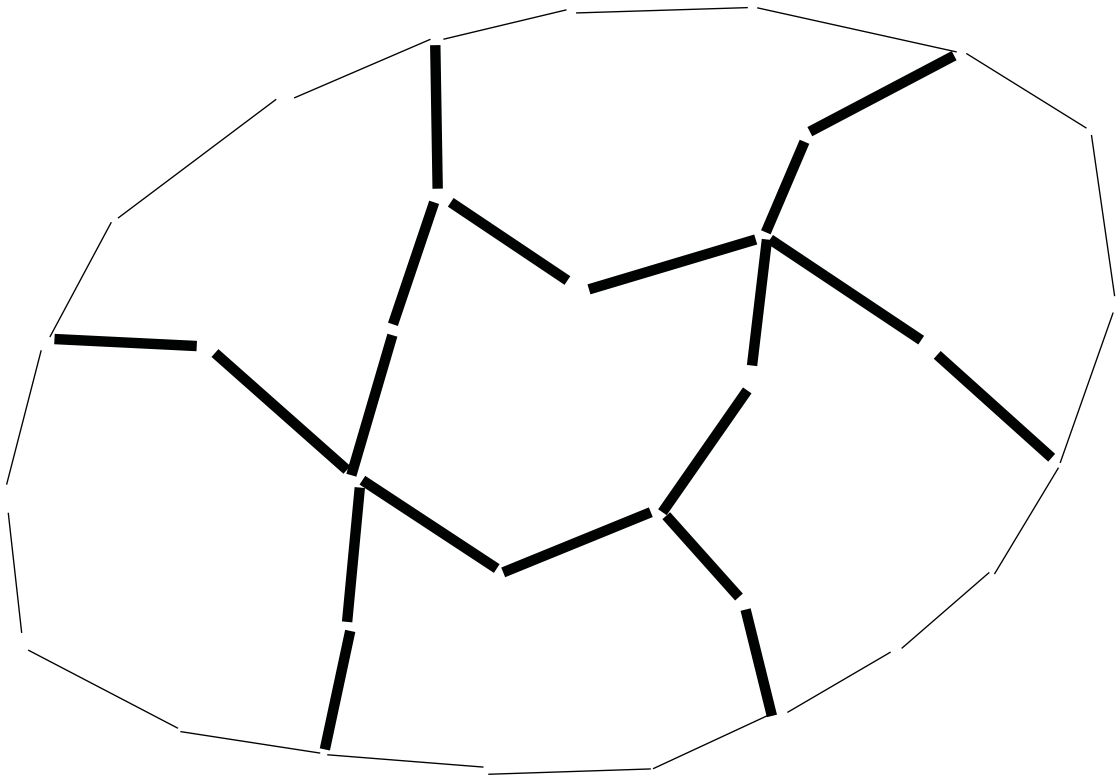
有限要素メッシュ（二次元、三角形要素）



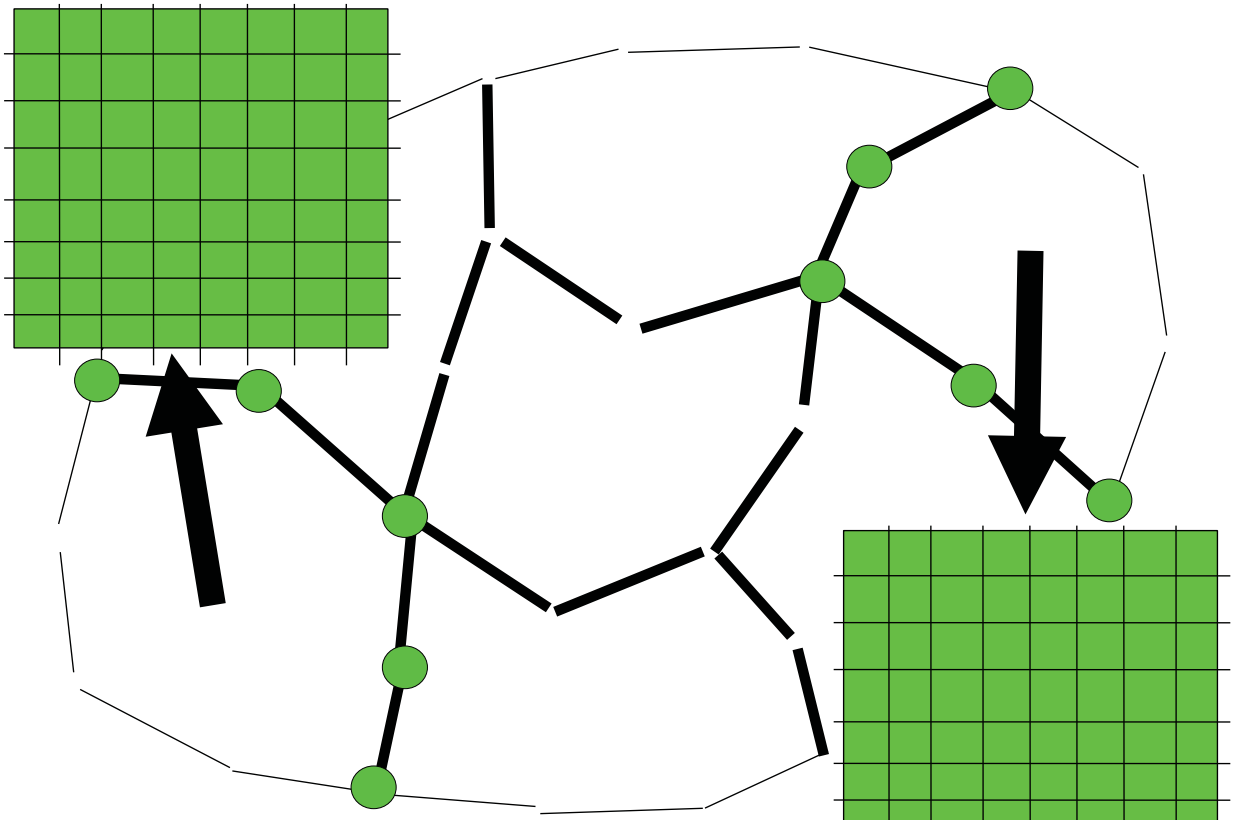
要素ごとの剛性行列（二次元、三角形要素）



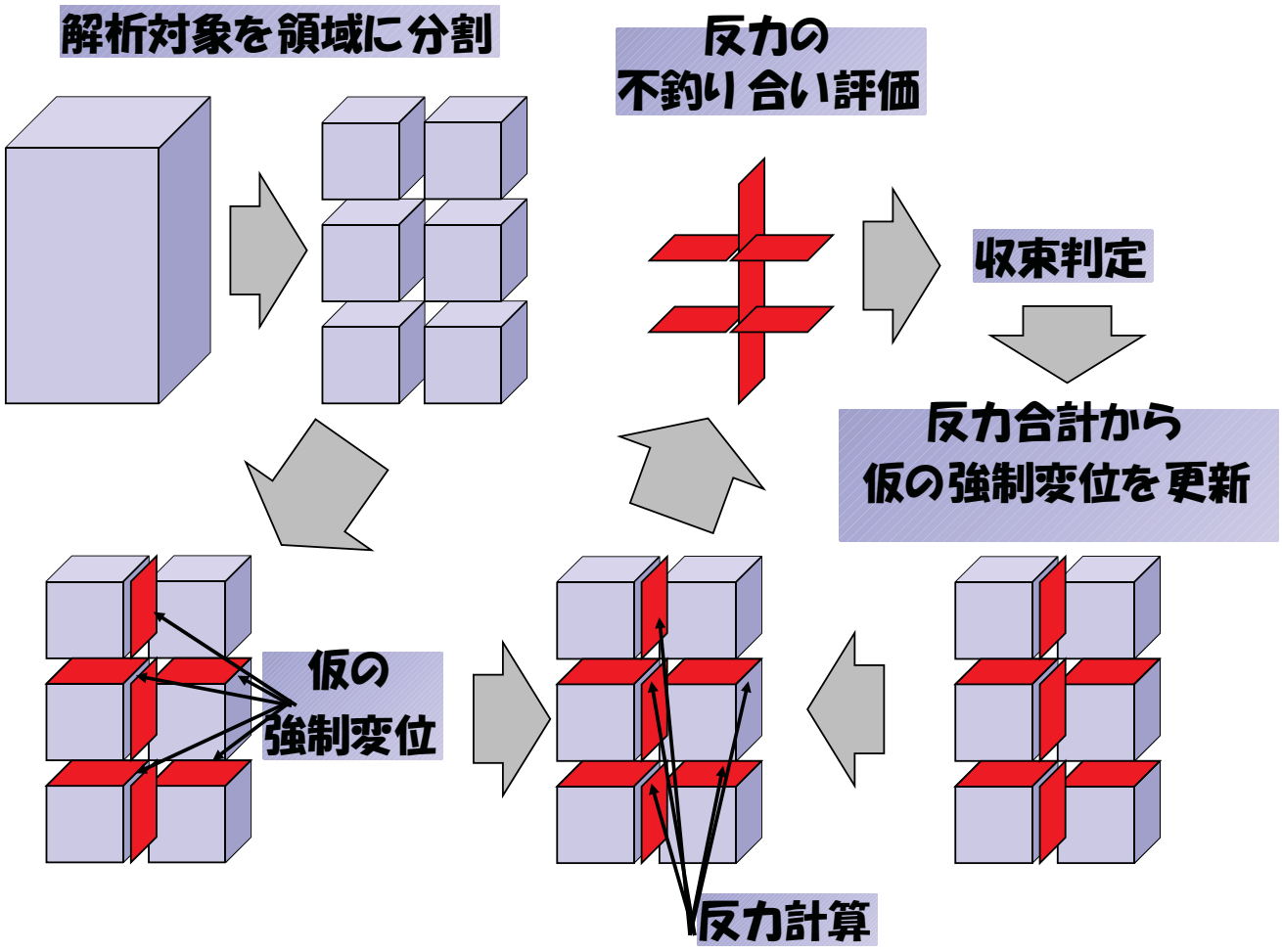
領域分割されたメッシュ
（部分領域/サブストラクチャに分割）



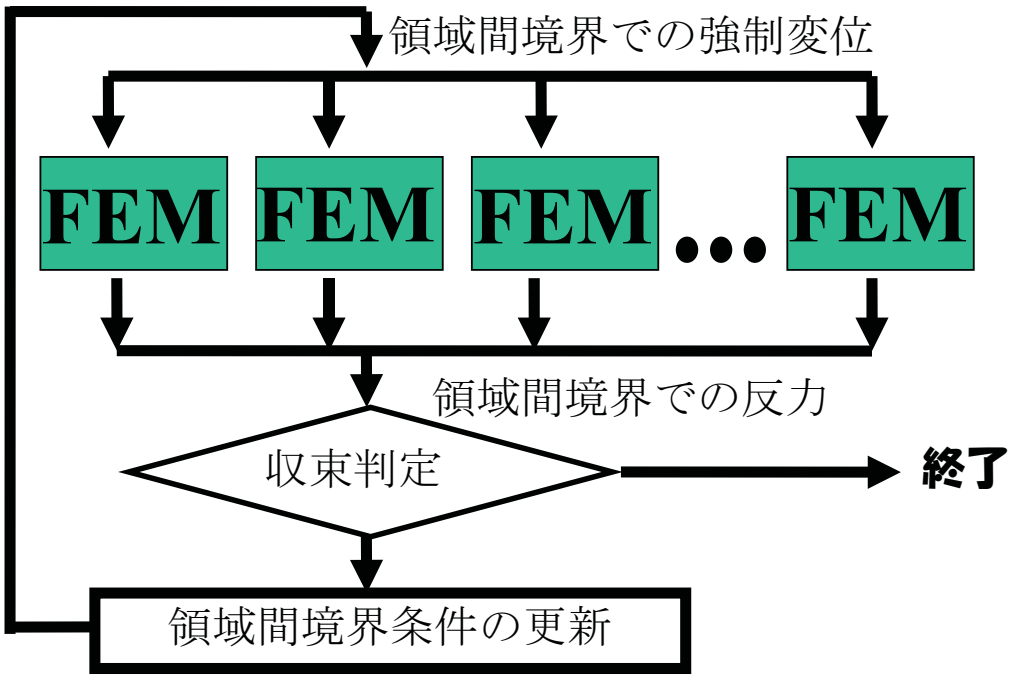
部分領域/サブストラクチャごとに
内部自由度を消去（静的縮約）



部分領域/サブストラクチャごとの
剛性行列（ローカルShur補元）



領域分割法のフロー



部分領域ごとのFEM計算

領域ごとFE解析における行列方程式　(i:内部、b:境界)

$$\begin{bmatrix} [K_{ii}] & [K_{ib}] \\ [K_{ib}]^T & [K_{bb}] \end{bmatrix} \begin{Bmatrix} \{u_i\} \\ \{u_b\} \end{Bmatrix} = \begin{Bmatrix} \{f_i\} \\ \{f_b\} \end{Bmatrix}$$

領域間境界での
強制変位境界条件

$$[K_{ii}]\{u_i\} = \{f_i\} - [K_{ib}]\{u_b\}$$

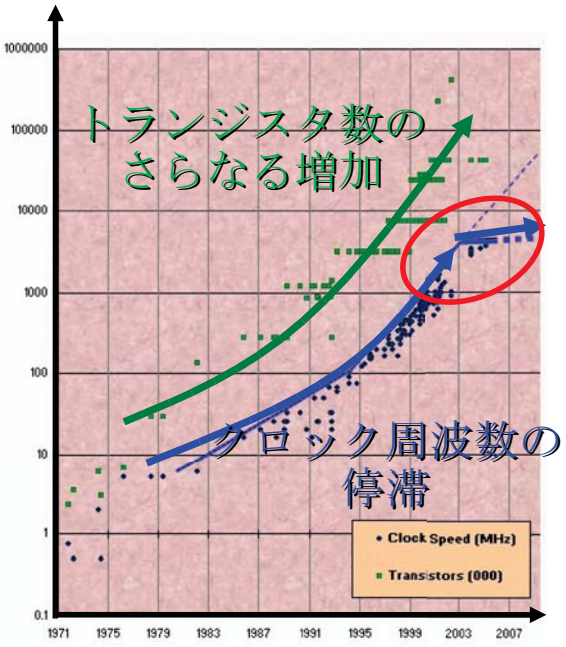
変位を求める

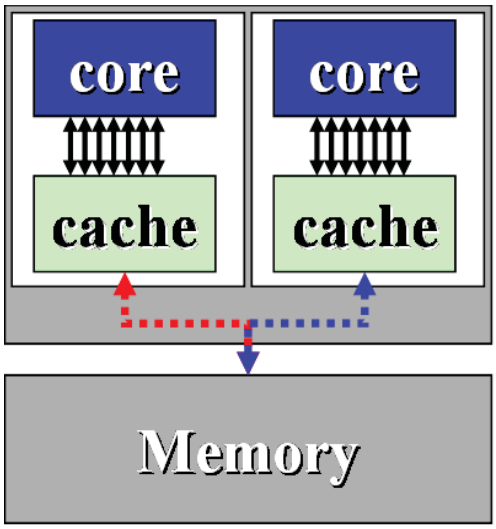
$$\{f_b\} = [K_{ib}]^T \{u_i\} + [K_{bb}]\{u_b\}$$

領域間境界での
反力を評価する

領域FEM計算の分類

- 線形代数ソルバーの種類
  - 直接法(Direct Solver)
    - LDL分解と前進消去後退代入、スカイライン記憶形式
  - 反復法(Iterative Solver)
    - CG法、前処理、非ゼロ成分記憶形式
    -
- 省メモリ性
  - メッシュ以外のデータを保存する(Storage)
    - 係数行列(スカイライン／非ゼロ記憶)
    - LDL分解結果
    - 前処理行列データ
  - メッシュ以外にメモリを使わない(Storage-Free)



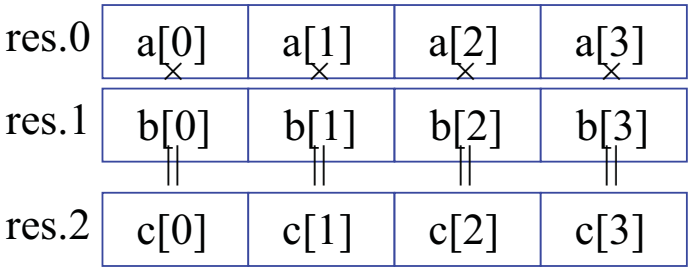


キャッシュ
有効利用

$c[i] = a[i] \times b[i]$

マルチコアから
メニーコアへ

SIMD
拡張命令

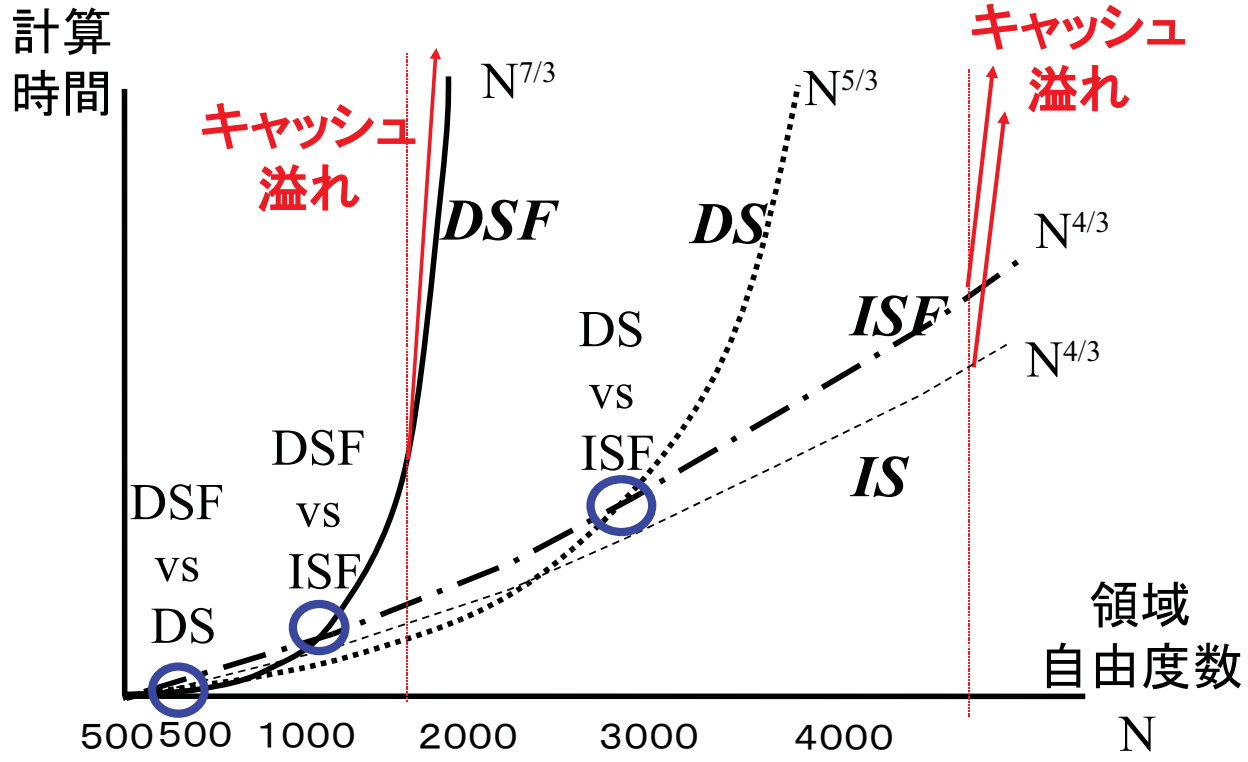


領域FEM計算の分類（続）

- 直接法ベース
  - 保存型：***DS (Direct solver-based matrix Storage)***
    - 係数行列とLDL分解結果をメモリ上に保存
    - 毎DDM反復ごとに前進後退計算だけで解を求める
  - 省メモリ型：***DSF (Direct solver-based matrix Storage-Free)***
    - 毎DDM反復ごとにLDL分解も行う必要あり

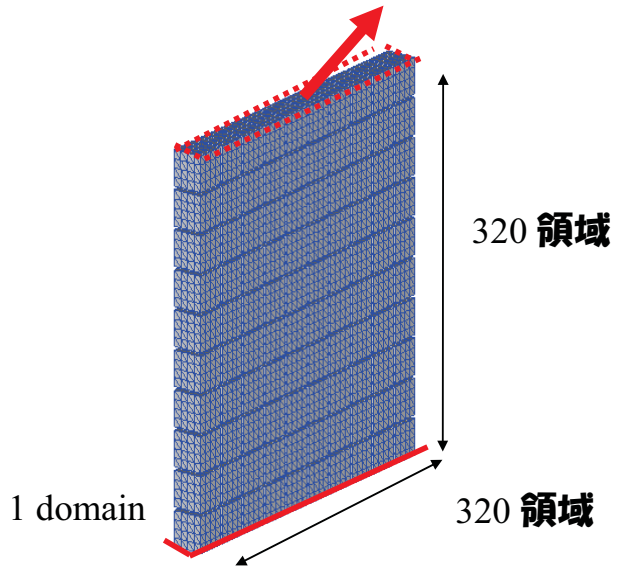
- 反復法ベース
  - 保存型：***IS (Iterative solver-based matrix Storage)***
    - 係数行列（または／かつ）前処理行列をメモリ上に保存
    - 毎DDM反復ごとにCG法で反復解を求める
  - 省メモリ型：***ISF (Iterative solver-based matrix Storage-Free)***
    - 毎DDM反復ごとに係数行列＋前処理行列の作成を行う

領域FEM計算
-領域サイズとの相関-

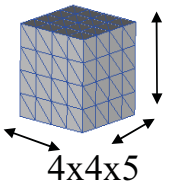


大規模板曲げ問題ベンチマーク

例: 220M(2億2千万) DOF モデル



フルモデル、100k (10万) 領域



領域メッシュ (2700 DOF)

境界条件
- 下面: 固定
- 上面:
  - 強制変位
    (u,v,w) = (1, 1, 1)

領域FEM計算ベンチマーク(ピーク性能比)

(単位: %)

	領域サイズ					
	小 (<1000自由度)			大 (3000自由度<)		
	Core i7	Opteron	Xeon-MP	Core i7	Opteron	Xeon-MP
DS_Sky_NZ	15	7	2	10	3	1
DS_Sky	39	22	12	17	8	2
DSF	51	34	41	54	31	36
IS-SSOR	30	19	20	23	19	19
IS-Jacobi	31	24		30	24	
ISF-EBE	54	40		53	40	

計測環境

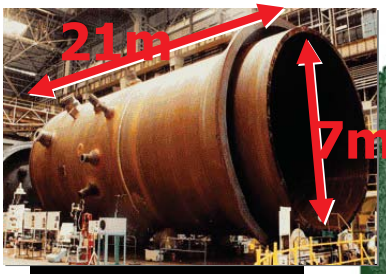
Core i7 : Intel Core i7 940 (Nehalem), 2.93 GHz, 4 core
Opteron : 東大T2K
CPUはAMD Opteron 8356 (Barcelona), 2.3 GHz, 4 core, 4 socket
Xeon-MP : Intel Xeon X7460 (Dunnington), 2.66 GHz, 6 core, 4 socket

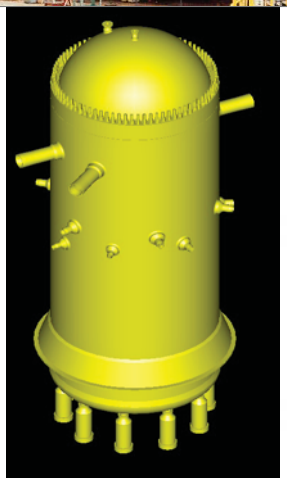
計算速度の比較

（単位:ミリ秒）

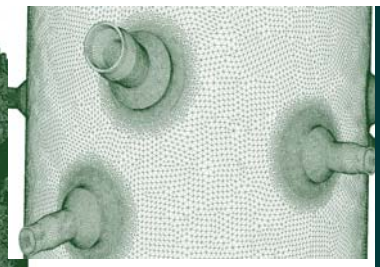
Intel Core i7 2.93 GHz	DSF	DS			ISF (SSOR)	IS (SSOR)
		NZ only	Sky_only	Sky_NZ		
400	0.12	N/A	0.046	0.053	0.32	0.16
1000	1.2	1.1	0.22	0.22	1.6	1.0
2200	9.0	8.8	0.90	0.91	6.3	4.9
4000	48	N/A	3.12	3.2	16	13

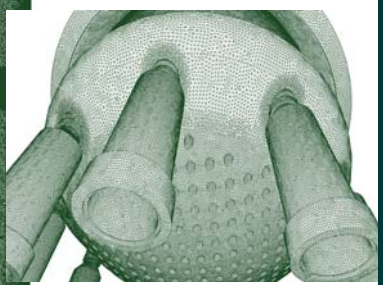
AMD Opteron 2.3 GHz	DSF	DS			ISF (SSOR)	IS (SSOR)
		NZ only	Sky_only	Sky_NZ		
400	0.22	0.20	0.11	0.14	N/A	N/A
1000	2.3	2.2	0.57	0.73	2.6	1.8
2200	20	20	3.4	3.9	9.3	7.5
4000	99	N/A	15	16	31	26

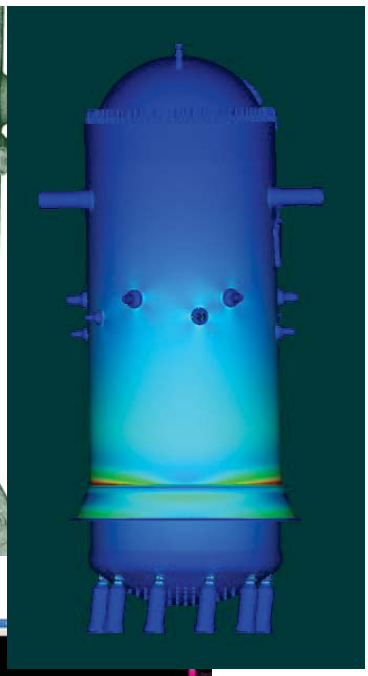


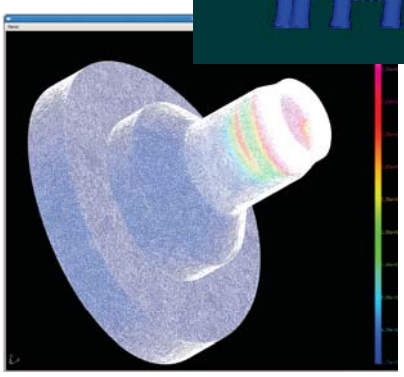


三次元
複雑形状
（薄肉構造）



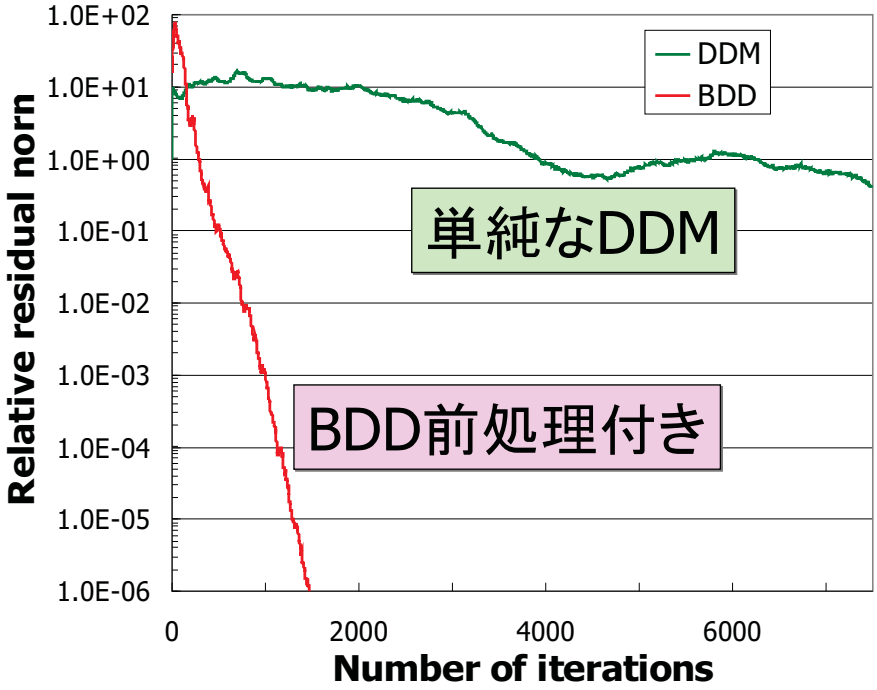






原子炉圧力容器
(二億自由度)

DDM向け前処理の必要性



•マルチグリッド型ソルバーの特性を利用

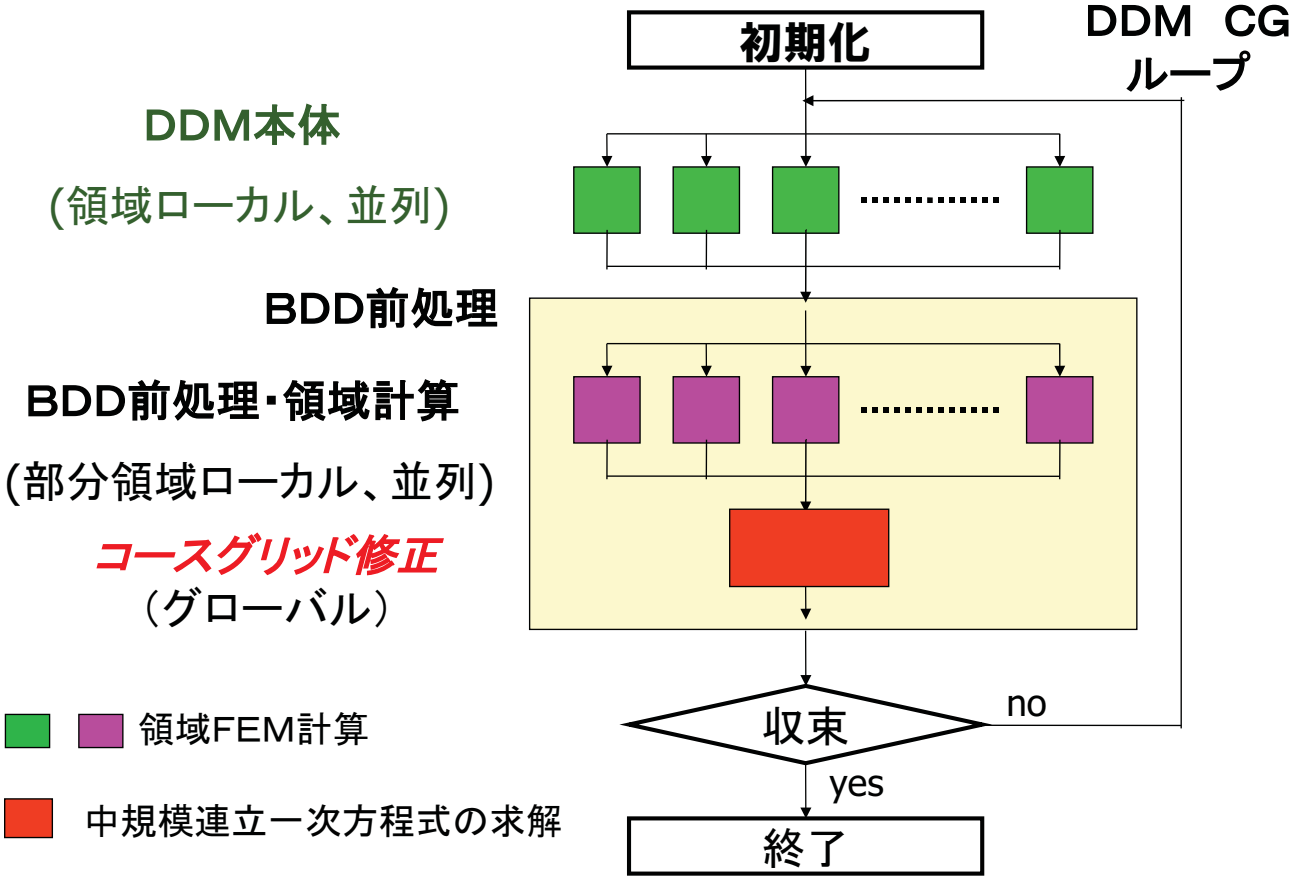
•梁や薄板状の（複雑形状アセンブリ）構造物に最適

DDM向け前処理： バランシング領域分割法 *Balancing Domain Decomposition (BDD)**

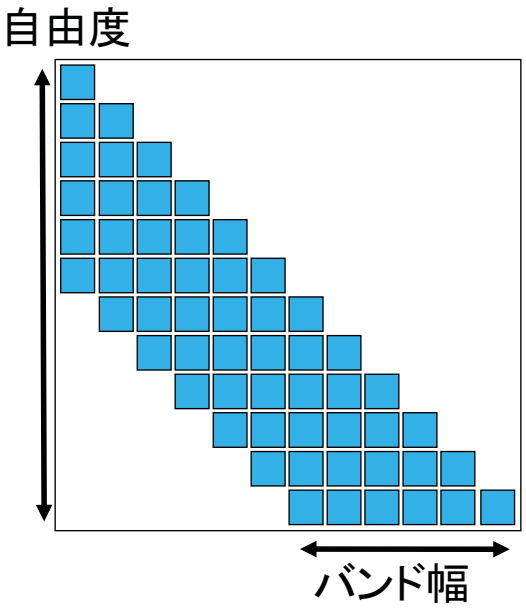
- ***DDM,***
  領域間インターフェイス問題をPCGで解く
  領域ごとにディリクレ問題を解く
- ***Neumann-Neumann (N-N) 前処理***
  領域単位でのローカルな前処理を行う
  領域ごとにノイマン問題を解く
- ***コースグリッド修正***
  領域分割から粗い格子（コースグリッド）を生成する
  このコース問題を解くことでグローバルな前処理を実現する

* **J.Mandel**:Balancing Domain Decomposition,Communications on Numerical Methods in Engineering,Vol.9,1993,pp233-241

BDD前処理付きDDMの流れ

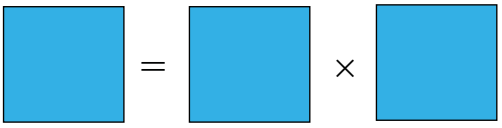


コースグリッド修正の高速化
並列バンド／スカイラインソルバー



ブロック化

1) キャッシュの有効利用
   --- スカラー最適化、OpenMP並列
2) MPIの通信コスト削減



N^3 浮動小数点演算
N^2 データ移動

広いバンド幅：← (より多くの領域数)
より多くのブロックが利用可能、より良い負荷バランス
ブロックサイズが大きくできる、通信コストの相対的低下
並列化しやすくなる

コースグリッド修正のベンチマーク

(単位：秒)

領域数	LDL分解			前進後退代入		
	6万	10万	16万	6万	10万	16万
研究室PCクラスタ	87	261	821	0.60	1.27	2.18
東大T2K	88	199	413	0.43	0.69	1.25
富士通FX1	77	199	405	0.55	0.96	1.54

計測環境

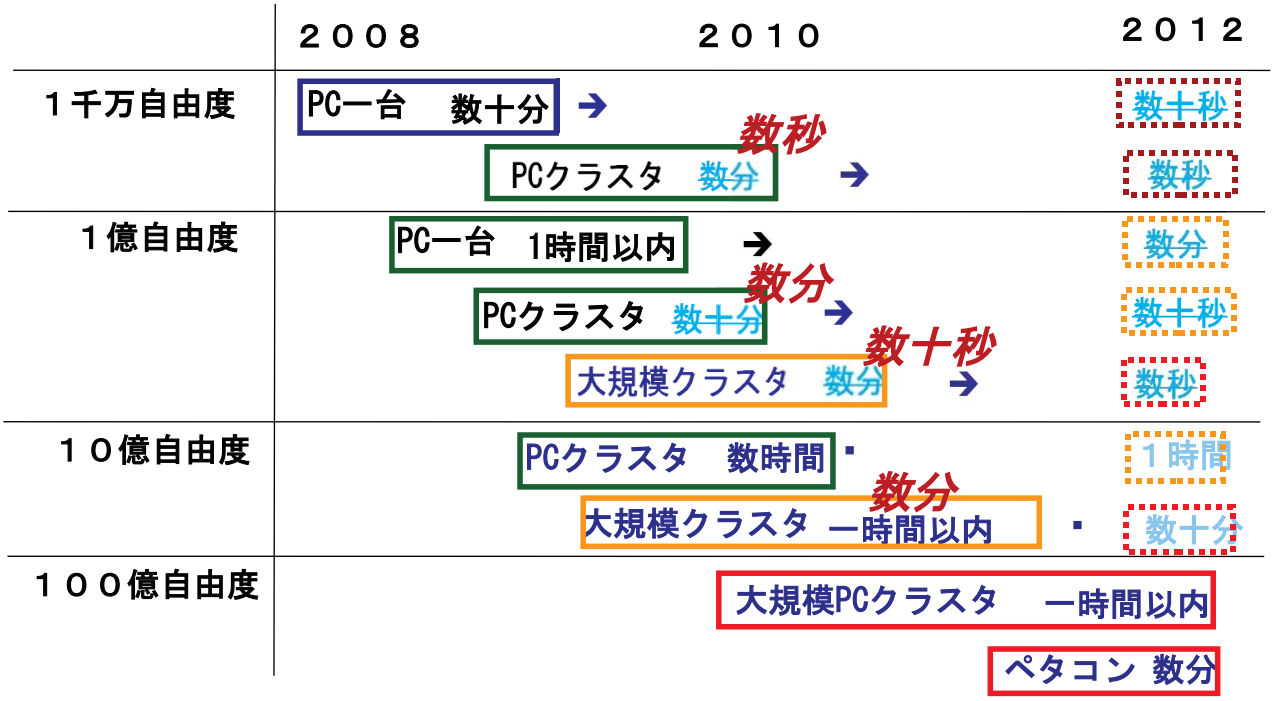
研究室PCクラスタ: Intel Core i7, GbE
東大T2K: AMD Opteron, Myrinet-10G
富士通FX1: SPARC64 VII, Infiniband

大規模PCクラスタ（東大T2K）

計算機環境:　東大T2K (日立HA8000クラスタ)
CPU: AMD Quad-core Opteron 8356 (2.3GHz) x 4 CPU
ネットワーク:　Myrinet-10G

総DOF	領域DOF	領域数	計算ノード数	コア数	計算時間 (単位：秒) コース行列の準備	求解
0．6億	1300	6万	16	256	104	38
1億	1300	10万	16	256	251	62
1億	2200	6万	64	1024	106	38
1．8億	2200	10万	64	1024	250	70
3．4億	4000	10万	256	4096	230	97
4．9億	4000	15万	256	4096	407	148
8．7億	4000	25万	256	4096	968	230

次世代スパコン向けチューニング
ロードマップと実績



課題：　さらなる大規模化に向けて

領域自由度	領域数	（コース行列サイズ）	総自由度数
300	3万	18万	1000万
	10万	60万	3000万
直接法	30万	180万	1億
1000	3万	18万	3000万
	10万	60万	1億
	30万	180万	3億
3000	3万	18万	1億
反復法	10万	60万	3億
	30万	180万	10億
1万	3万	18万	3億
	10万	60万	10億
反復法	30万	180万	30億
（EBE）	100万	600万	100億
3万	3万	18万	10億
	10万	60万	30億
	30万	180万	100億
	100万	600万	300億

並列直截法
ソルバー？