先駆的科学計算に関するフォーラム2008　～線形計算を中心に～

1億自由度規模解析の実用化に向けた並列バランシング領域分割法

荻野正雄(九州大学), 河合浩志(東京大学),
塩谷隆二(東洋大学),吉村忍(東京大学)

2008年9月3日

目次

- 研究背景と目的
- ADVENTUREシステム
- 領域分割法(DDM)
- BDD法，IBDD-DIAG法
- 1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

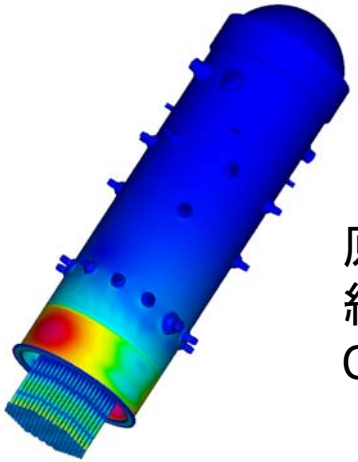
目次

- ✓研究背景と目的
- ADVENTUREシステム
- 領域分割法(DDM)
- BDD法，IBDD-DIAG法
- 1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

研究背景

- 材料・幾何学非線形問題
- 流体-構造連成問題
- モデル形状の複雑化

⇒ 計算量が膨大



原子炉圧力容器の耐震解析,
約2億自由度問題×1,000回,
Ogino et al., 2007

Roadrunner, LANL

T2K HA8000, ITC, Univ. of Tokyo

- マルチコアCPUや高性能計算機の低価格化による小規模並列計算機の普及
- 大規模並列計算機システム(1,000コア以上)の整備
- 進むメニーコア化

⇒ 並列計算方法の複雑化

tatara, RI2T, Kyushu Univ.

研究目的

- 複雑な機器や建築物の丸ごと構造解析
  ⇒ 数千万自由度規模以上の非線形非定常解析

- 1億自由度規模の線形問題を実用時間内に解く
  スパコンで1分　PCクラスタで30分

– 高速かつ安定した収束性を持つ反復法
– 共有-分散並列環境における高い並列効率
– マルチコアCPUにおける高い対ピーク性能比

大規模問題向け線形ソルバ

- 数千万から数億自由度規模では反復解法となる
  ⇒高い収束性を持つ前処理法, 高い並列効率が必要
- 代表的な前処理法
  - ICCG
  - Multigrid (Brandt '77)
  - AMG (Ruge and Stuben '87)
  - FETI (Farhat '92), BDD (Mandel '93)
  - FETI-DP (Farhat et al. '00), BDDC (Dohrmann '03)
  - CGCG (Suzuki et al. '02)
  - IBDD-DIAG (Ogino et al. '08)

並列化法HDDM
(塩谷・矢川, '01)

目次

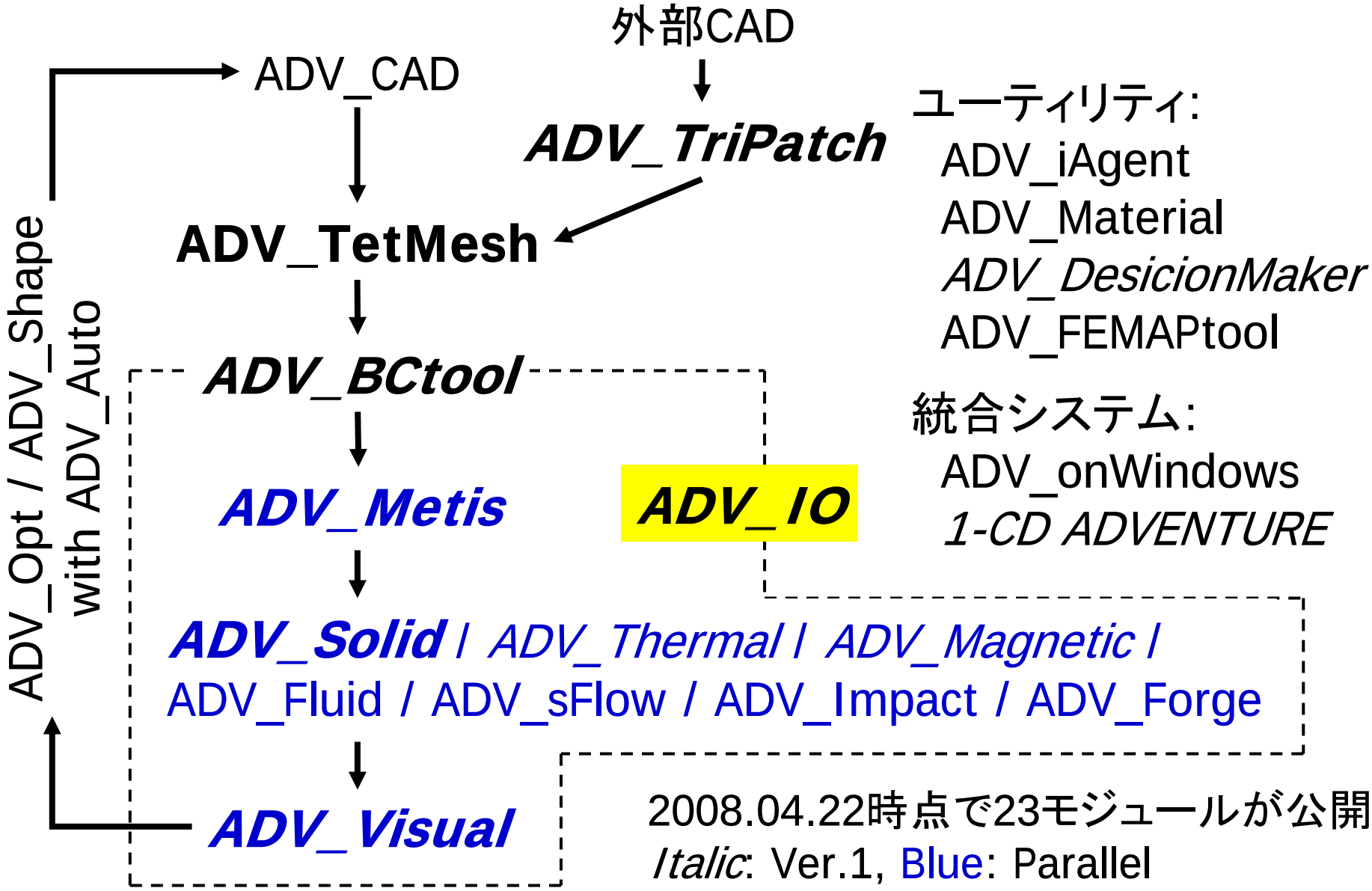
- ✓研究背景と目的
- ✓ADVENTUREシステム
- 領域分割法(DDM)
- BDD法，IBDD-DIAG法
- 1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

ADVENTUREシステム

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

- 日本学術振興会未来開拓事業(1997.8-2002.3)の1つとして，東京大学，九州大学，慶應義塾大学など大学主導の研究グループによって開発．現在もオープンソースCAEシステム開発プロジェクトとして研究開発を継続中．
- プレ，メイン，ポスト，設計最適化などを兼ね備えたシステム．
- 数百から数億自由度規模メッシュによる大規模解析が可能．
- 1,000プロセッサを越える超並列計算機環境でも90%を越える高い並列効率．
- 単一PC，PCクラスタ，超並列計算機，地球シミュレータ，Gridコンピュータなど多様な計算機環境で稼動．
- ライセンスフリー，オープンソース
- モジュール構造と標準化IO(ADV_IO)

ADVENTUREシステムのモジュール構成



構造解析用ADVENTUREモジュール

- **ADVENTURE_Metis**
  ミネソタ大学開発のグラフ分割ライブラリMETIS及びその並列版ParMETISを用いた階層型領域分割モジュール.

- **ADVENTURE_Solid**
  有限要素法による3次元静的弾塑性解析モジュール. 階層型領域分割法による高い並列効率, IBDD-DIAG法による高速安定解析.

目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- BDD法, IBDD-DIAG法
- 1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

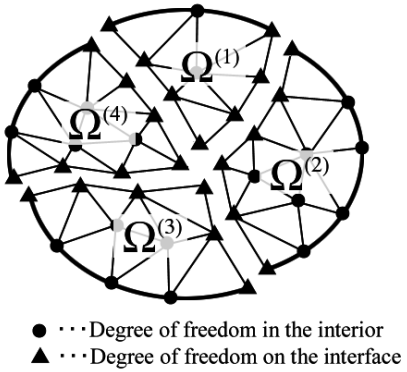
DDMアルゴリズム (1/2)

- 解くべき式

$$Ku = f$$

K : stiffness mateix, SPD

u : unknown vector, f : given vector



- N個に領域分割 $\Omega \rightarrow \Omega_1, \ldots, \Omega_N$

$$R_i : \Omega \rightarrow \Omega_i, \; u_i = R_i u, \; K = \sum_{i=1}^{N} R_i^T K_i R_i, \quad f = \sum_{i=1}^{N} R_i^T f_i$$

- 部分領域の自由度を領域内部(I)と領域間境界(B)に分割

$$u_i = \begin{bmatrix} u_{Ii} \\ u_{Bi} \end{bmatrix}, \; K_i = \begin{bmatrix} K_{IIi} & K_{IBi} \\ K_{IBi}^T & K_{BB} \end{bmatrix}, \; f_i = \begin{bmatrix} f_{Ii} \\ f_{Bi} \end{bmatrix}, \; R_i = \begin{bmatrix} R_{Ii} & 0 \\ 0 & R_{Bi} \end{bmatrix}$$

DDMアルゴリズム (2/2)

$$\begin{bmatrix} K_{II1} & & 0 & K_{IB1}R_{B1} \\ & \ddots & & \vdots \\ 0 & & K_{IIN} & K_{IBN}R_{BN} \\ R_{B1}^T K_{IB1}^T & \cdots & R_{BN}^T K_{IBN}^T & \sum R_{Bi}^T K_{BBi} R_{Bi} \end{bmatrix} \begin{bmatrix} u_{I1} \\ \vdots \\ u_{IN} \\ u_B \end{bmatrix} = \begin{bmatrix} f_{Ii} \\ \vdots \\ f_{IN} \\ f_B \end{bmatrix}$$

部分領域内部と領域間境界に式を分割

- 領域間境界上自由度の式　→前処理付き共役勾配法を適用

$$Su_B = g$$

$$S = \sum_{i=1}^{N} R_{Bi}^T S_i R_{Bi}, S_i = K_{BBi} - K_{IBi}^T K_{IIi}^{-1} K_{IBi}$$

$$g = \sum_{i=1}^{N} R_{Bi}^T \left(f_{Bi} - K_{IBi}^T K_{IIi}^{-1} f_{Ii} \right)$$

- 部分領域内部自由度の式　→直接法を適用

$$u_{Ii} = K_{IIi}^{-1} \left(f_{Ii} - K_{IBi} R_{Bi} u_B \right), \quad i = 1, \ldots, N$$

DDMの実装

- 係数行列-ベクトル積 $y = Sp$

  等価な領域単位の処理に置き換える

$$\begin{bmatrix} x_i \\ \cdot \end{bmatrix} = \begin{bmatrix} K_{II\,i} & K_{IB\,i} \\ K_{IB\,i}^{T} & K_{BB\,i} \end{bmatrix} \begin{bmatrix} 0 \\ -R_{B\,i}\,p \end{bmatrix}, i = 1,\ldots,N$$

(1) 基本境界条件の処理

$$\begin{bmatrix} w_i \\ \cdot \end{bmatrix} = \begin{bmatrix} K_{II\,i} & 0 \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} x_i \\ \cdot \end{bmatrix}, i = 1,\ldots,N$$

(2) 連立方程式の求解

$$\begin{bmatrix} \cdot \\ y_i \end{bmatrix} = \begin{bmatrix} K_{II\,i} & K_{IB\,i} \\ K_{IB\,i}^{T} & K_{BB\,i} \end{bmatrix} \begin{bmatrix} w_i \\ R_{B\,i}\,p \end{bmatrix}, i = 1,\ldots,N$$

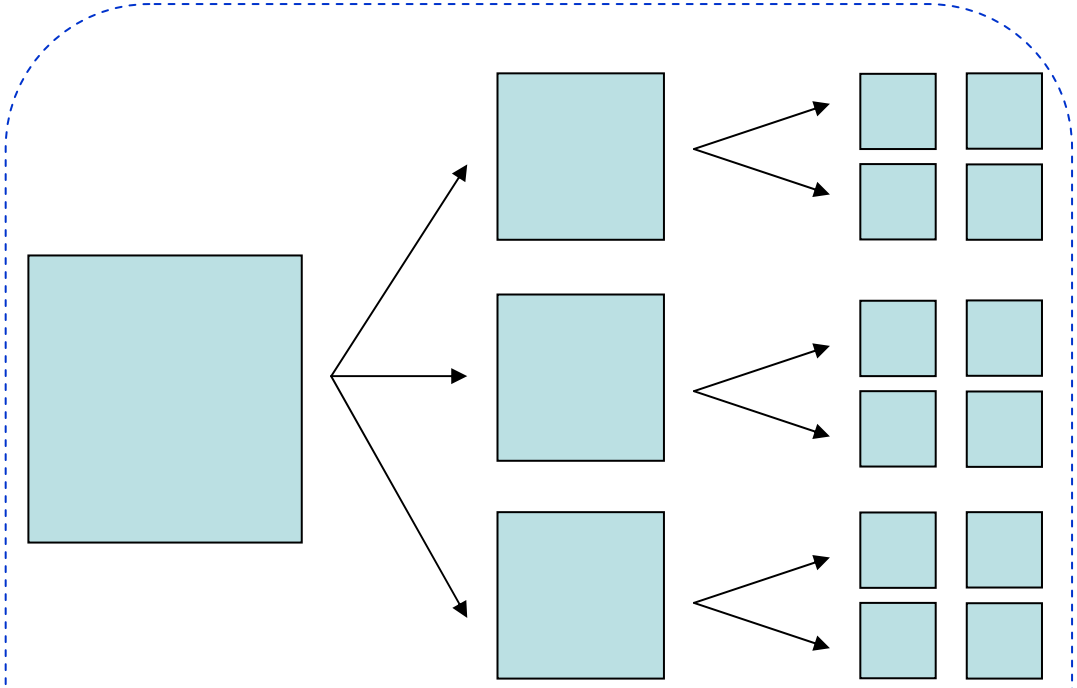
(3) 反力の計算

$$y = \sum_{i=1}^{N} R_{B\,i}^{T}\, y_i$$

(4) 領域間境界上で重ね合わせ

階層型領域分割法(HDDM)による並列実装

- **HDDM** (矢川・塩谷 '94) とは多様な並列計算機環境において高い並列効率が得られる領域分割法の並列実装方法の1つ.



HDDMの特徴
- 2階層の領域分割
- 親階層は入力データの負荷分散
- 子階層は全解析時間が最小となる分割を選択
- 並列データ処理は部分単位または部分領域単位で可能

目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- ✓BDD法，IBDD-DIAG法
- 1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

バランシング領域分割法

- BDD法 (Mandel '93)
  - コースグリッド修正を備えたDDM向けマルチグリッド的前処理法
  - ローカル前処理としてNeumann-Neumann前処理
  - コースグリッドはローカル係数行列の零空間から代数的に構築
- IBDD-DIAG法 (Ogino et al. '08)
  - 不完全コースグリッド修正を導入した大規模解析向けのBDD的手法
  - ローカル前処理は対角スケーリングで簡易化

Neumann-Neumann (N-N) 前処理*

Neumann-Neumann前処理行列

$$T_{\mathrm{N-N}} = \sum_{i=1}^{N} R_i^T D_i^T S_i^{\dagger} D_i R_i ,$$

where

$S_i^{\dagger}$: generalized inverse of a local Schur complement,

領域分割の重み行列

$$D_i : V_i \to V_i , \qquad \sum_{i=1}^{N} R_i^T D_i R_i = I\big|_V$$

V : The space of the dofs on interface, and

V_i : The space of the dofs on interface of Ω_i.

DeRoeck and LeTallec '91

コーススペースの構築

コーススペースWの定義

$$W = \left\{ v \in V \middle| v = \sum_{i=1}^{N} R_i^T D_i Z_i \lambda_i \ , \lambda_i : \text{arbitrary} \right\},$$

where $Ker\, S_i \subset Range\, Z_i \subset V_i$.

3次元構造問題においては,

at point $P(x_1, x_2, x_3)$ on $\partial\Omega_i$,

$$Z_i^P = \begin{bmatrix} 1 & 0 & 0 & 0 & x_3 & -x_2 \\ 0 & 1 & 0 & -x_3 & 0 & x_1 \\ 0 & 0 & 1 & x_2 & -x_1 & 0 \end{bmatrix}$$

$$Z_i = \sum_P B_i^P Z_i^P ,$$

$$B_i^P : P \to \partial\Omega_i .$$

BDD前処理

前処理付きCG法における前処理ベクトルの計算;

$$M_{\mathrm{BDD}}{}^{-1} r = PS^{-1} r + (I - P) T_{\mathrm{N-N}} (I - P)^T r ,$$

Coarse space components

N-N precon.

Coarse grid correction

r : 残差ベクトル

$S_0 = R_0 S R_0{}^T$:コース行列

$R_0{}^T = \left[R_{B1}{}^T D_1{}^T Z_1 \quad ,\ldots, \quad R_{BN}{}^T D_N{}^T Z_N \right]$:コーススペースへの射影行列

D_i :領域分割の重み行列

Z_i :ローカルコーススペースを定義する行列

大規模問題へのBDD法の適用について

- N-N前処理

  係数行列が非正則行列S_iとなるNeumann問題の計算コスト及び精度が問題となりやすい.

  ⇒ 対角スケーリング前処理 $\boldsymbol{T}_{\mathrm{DIAG}} = \sum_{i=1}^{N} \boldsymbol{R}_i^T \mathrm{diag}\left(\boldsymbol{K}_{\boldsymbol{BB}\,i}\right)^{-1} \boldsymbol{R}_i$

- コースグリッド修正(バランシング)

  コース行列の次元数は6Nとなり, 大規模問題ではコース問題を解く事が困難となりやすい.

  ⇒ 不完全バランシング $\tilde{\boldsymbol{S}}_0^{-1}$: Inexact balancing operator

IBDD-DIAG method

IBDD-DIAG前処理

$$M_{\text{IBDD-DIAG}}^{-1} r = \tilde{P} S^{-1} r + \left(I - \tilde{P}\right) T_{\text{DIAG}} \left(I - \tilde{P}\right)^T r ,$$

Incomplete coarse space components

DIAG precon.

Incomplete coarse grid correction

where

$\tilde{P} = R_0^T \tilde{S}_0^{-1} R_0 S$ ：不完全なコーススペース射影行列

$\tilde{S}_0^{-1}$ ：不完全バランシング行列

IBDD-DIAG法の実装 (1/2)

$$z = M_{\mathrm{IBDD-DIAG}}^{-1} r$$

1. 不完全コースグリッド修正

$$\overline{\lambda}_0 = \widetilde{S}_0^{-1} R_0 r$$ inexact coarse space component

$$\widetilde{r} = r - S R_0^T \overline{\lambda}_0$$ inexact balancing

2. 対角スケーリング前処理

$$\widetilde{u} = \sum_{i=1}^{N} R_i^T \mathrm{diag}\left(K_{BB\,i}\right)^{-1} R_i \widetilde{r}$$

3. 不完全コースグリッド修正

$$\overline{\mu}_0 = \widetilde{S}_0^{-1} R_0 \left(r - S\widetilde{u}\right)$$ inexact coarse space component

$$z = \widetilde{u} + R_0^T \overline{\mu}_0$$

IBDD-DIAG法の実装 (2/2)

- コース問題　$\boldsymbol{S}_0\boldsymbol{d}_0 = \boldsymbol{R}_0\boldsymbol{r}$

係数行列(コース行列)において,

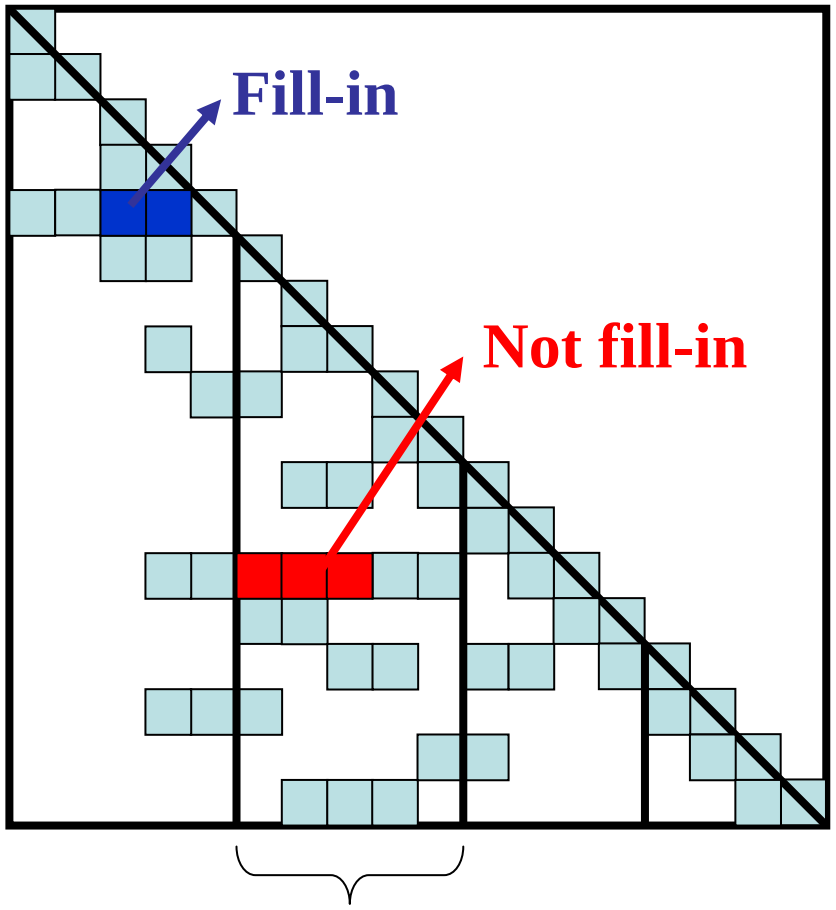
$$\boldsymbol{S}_0 = \boldsymbol{R}_0\boldsymbol{S}\boldsymbol{R}_0^{\boldsymbol{T}} \qquad \boldsymbol{R}_0^{\boldsymbol{T}} = \left[\boldsymbol{R}_{\boldsymbol{B}1}^{\boldsymbol{T}}\boldsymbol{D}_1^{\boldsymbol{T}}\boldsymbol{Z}_1 \quad ,\ldots, \quad \boldsymbol{R}_{\boldsymbol{B}N}^{\boldsymbol{T}}\boldsymbol{D}_N^{\boldsymbol{T}}\boldsymbol{Z}_N\right]$$

3次元構造解析ではコーススペースが剛体運動から構築されるため,次元数は***6N***となる.

IBDD-DIAG法では,

大規模問題においてはコース問題も大規模となることから, 不完全コレスキー分解した $\tilde{\boldsymbol{S}}_0^{-1}$ を用いて近似する.

並列不完全コレスキー分解における フィルイン棄却ルールの一例



Distributed matrix in part-wise

- HDDMのPart分割に基づいて，列ブロック分割
- 分散行列はスカイライン記憶
- 並列効率を阻害するようなフィルインを棄却

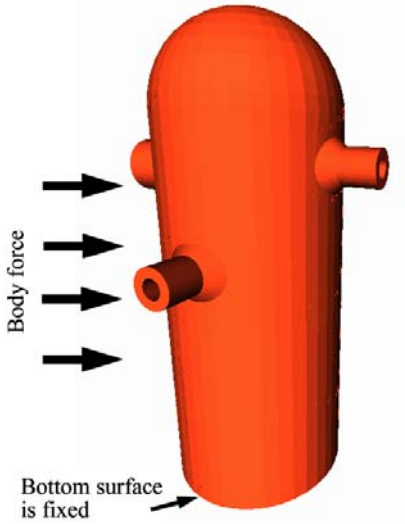
- コーディングしやすい
- 近似精度を制御しにくい

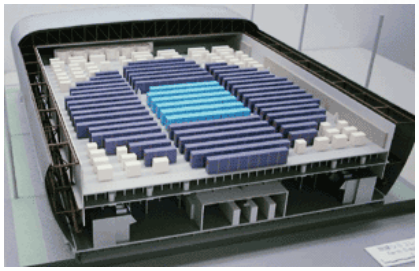
目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- ✓BDD法, IBDD-DIAG法
- ✓1億自由度規模問題の数値実験例
- Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

1億自由度規模の数値実験例

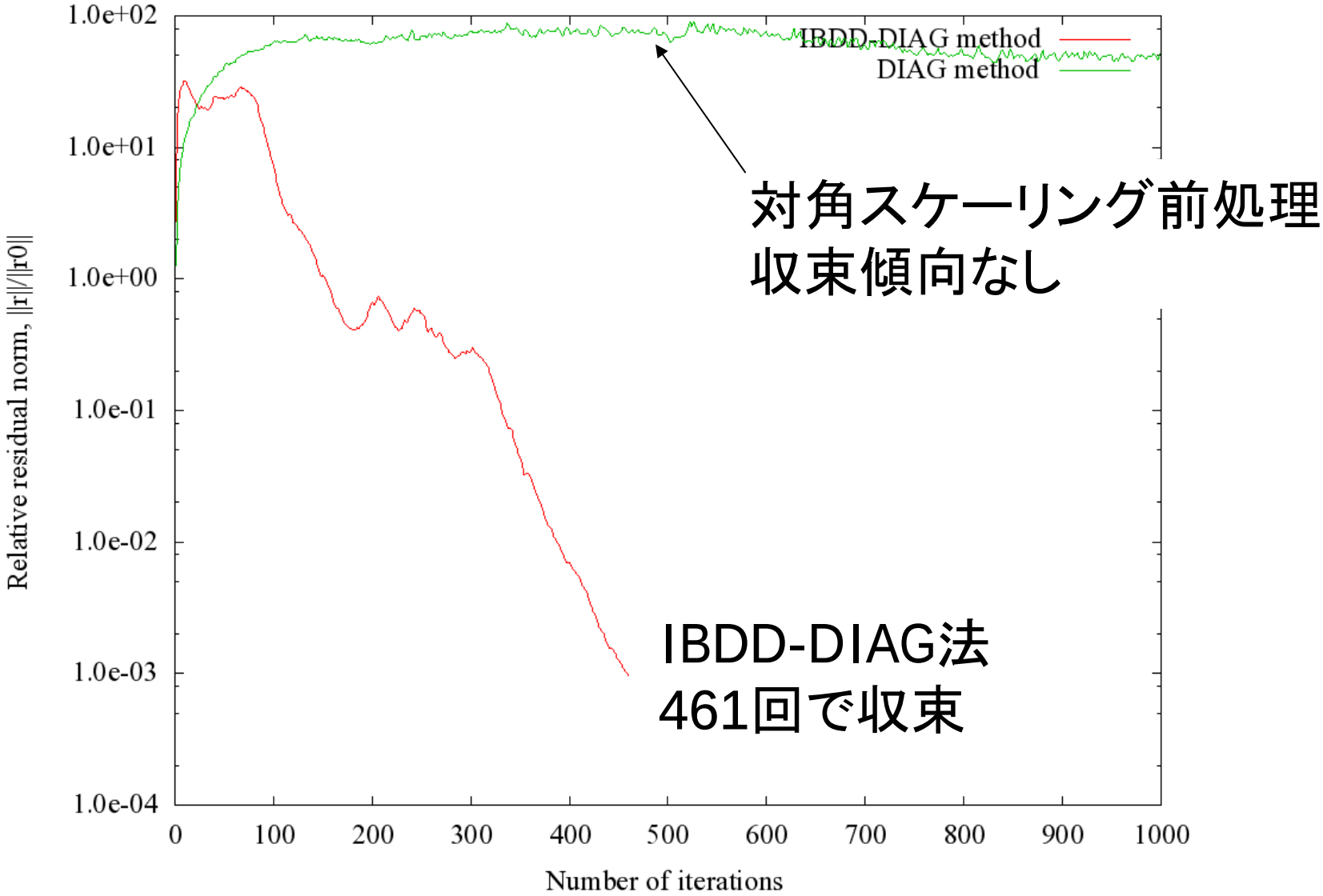
- 簡易原子炉圧力容器モデル
  要素数:　　25,083,456
  節点数:　　34,772,634
  自由度数: **104,195,502**
- ADV_Solid
  線形応力問題
  収束判定値:　相対残差ノルム<1.0e-3
  C, MPI
- 使用計算機
  Fujitsu PRIMEQUEST 580
  NEC Earth Simulator





RIIT Research Institute for Information Technology
Kyushu University

IBDD-DIAG法 vs. 対角スケーリング前処理



1億自由度規模の解析性能

Computation performances on **PRIMEQUEST 580**

Cores	Subdomains	Iter.	Time (sec.)	Speed-up	Mem. (GB)	Mem./core (MB)
192	173,760	461	**5,108**	1.00	138	**736**
256	173,824	528	**4,041**	1.26	135	**540**
512	174,080	613	**2,471**	2.07	132	**264**

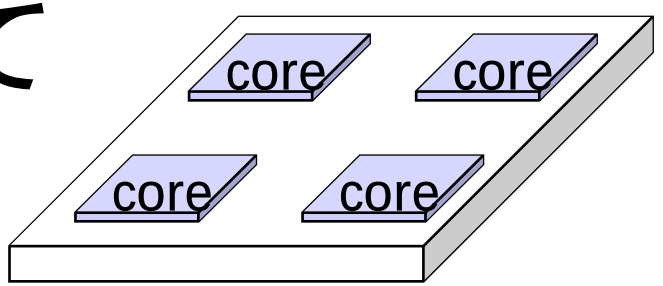
Computation performances on **Earth Simulator**

256	69,632	495	**1,128**	1.00	539	**2,156**
512	69,632	612	**588**	1.92	674	**1,347**
1,024	69,632	672	**335**	3.37	946	**946**

目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- ✓BDD法, IBDD-DIAG法
- ✓1億自由度規模問題の数値実験例
- ✓Direct Storage-Free (DSF)法
- メニーコアに向けた数値実験例
- まとめ

マルチコアCPUについて



- 従来設計の限界
  消費電力や熱密度の問題により,
  動作周波数の向上が困難

  ⇒ 今後は演算コアをCPUチップ上に多数集積することで
  演算性能の向上が図られる

- 演算性能の大幅向上が期待, 一方でメモリ性能は?
  - 現時点でメモリアクセス1回につき演算100回が可能
    今後はこの比が1:1,000に拡大 (Memory Wall問題)

- メモリアクセスに比べて演算量の多いアルゴリズムが優位
  - 演算コア数の増加に応じた高速化が可能
  - キャッシュ上にデータを置き, メモリアクセスを極力減らす

メニーコア向けDDMの実装方法

- 河合浩志氏の分類
  - 保存型 (Matrix Storage type):
    最初に演算コストの高い係数行列の作成, およびLDLt分解を最初に行う
    DDM反復ではメモリ上に記憶した結果を利用する
    各反復における演算量は少ないがメモリアクセスは多い

    Direct Storage (DS)

  - 非保存型 (Matrix Storage-Free type):
    DDM反復毎に係数行列の作成, LDLt分解を毎回行う
    メモリからの読み込みはメッシュデータ程度
    各反復における演算量は多いがメモリアクセスは少ない

    Direct Storage-Free (DSF)

DDMのDSF実装

- 係数行列-ベクトル積 $y = Sp$

  等価な領域単位の処理に置き換える

$$\begin{bmatrix} x_i \\ \cdot \end{bmatrix} = \begin{bmatrix} K_{II\,i} & K_{IB\,i} \\ K_{IB\,i}^{T} & K_{BB\,i} \end{bmatrix} \begin{bmatrix} 0 \\ -R_{B\,i}\,p \end{bmatrix}, i = 1,\ldots,N$$

剛性行列作成
行列-ベクトル積

$$\begin{bmatrix} w_i \\ \cdot \end{bmatrix} = \begin{bmatrix} K_{II\,i} & 0 \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} x_i \\ \cdot \end{bmatrix}, i = 1,\ldots,N$$

LDLt分解
直接法で求解

$$\begin{bmatrix} \cdot \\ y_i \end{bmatrix} = \begin{bmatrix} K_{II\,i} & K_{IB\,i} \\ K_{IB\,i}^{T} & K_{BB\,i} \end{bmatrix} \begin{bmatrix} w_i \\ R_{B\,i}\,p \end{bmatrix}, i = 1,\ldots,N$$

EBEで行列-ベクトル積

$$y = \sum_{i=1}^{N} R_{B\,i}^{T}\, y_i$$

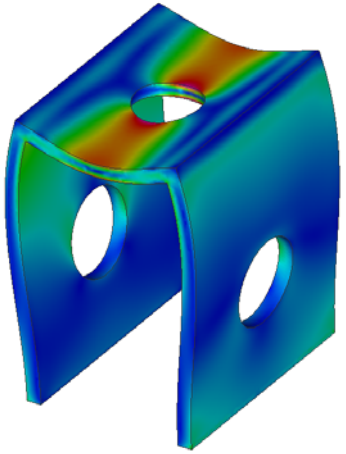
目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- ✓BDD法, IBDD-DIAG法
- ✓1億自由度規模問題の数値実験例
- ✓Direct-Storage Free (DSF)法
- ✓メニーコアに向けた数値実験例
- まとめ

実験環境

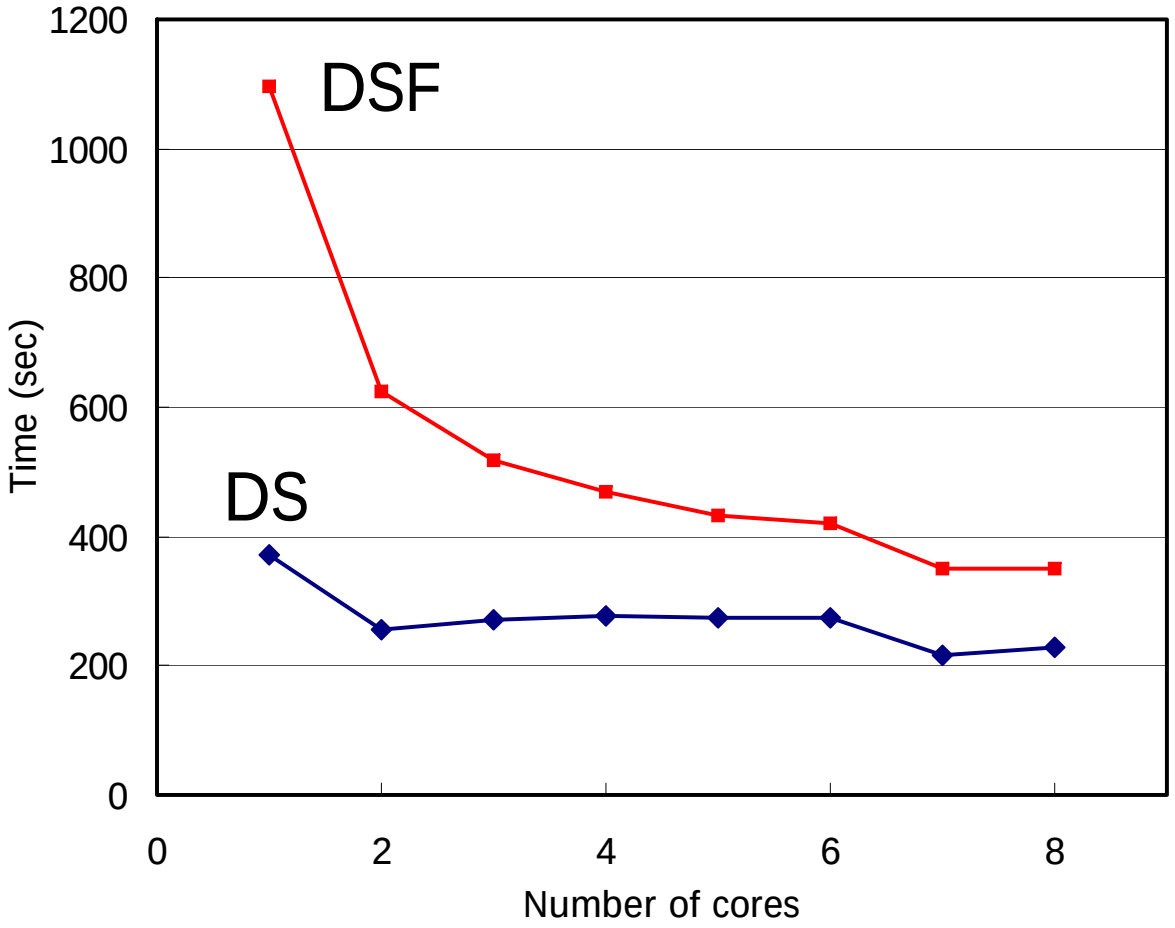
- 環境1(Xeon)
  - CPU: Intel Xeon 2.33GHz (OC) x 1 (= 8 core)
  - Memory: 32GB
  - OS: Redhat Linux Enterprise
  - Compiler: Intel C Compiler 10.1, MPICH2-1.0.7
- 環境2(PC cluster)
  - CPU: Intel Core2Duo 2.40GHz (DC) x 24 (= 48 core)
  - Memory: 4GB x 24
  - OS: openSUSE 10.3 x86_64
  - Compiler: Intel C Compiler 10.1, MPICH2-1.0.7

今回はthread並列は用いずにFlat MPI並列

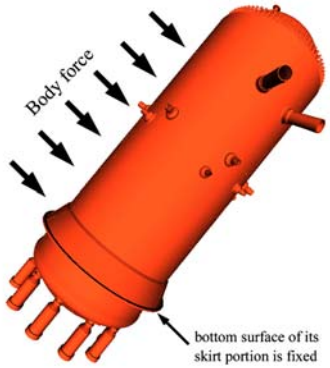


DSとDSFの比較: 300万自由度

Machine: Xeon
Solver: BDD-DIAG　収束判定: 相対残差6桁



DSFによる大規模解析例



ABWR型原子炉圧力容器モデル
要素数: 7,486,792
節点数: 11,794,506
自由度数: **35,383,518**
領域数: 58,944

収束判定: 相対残差6桁

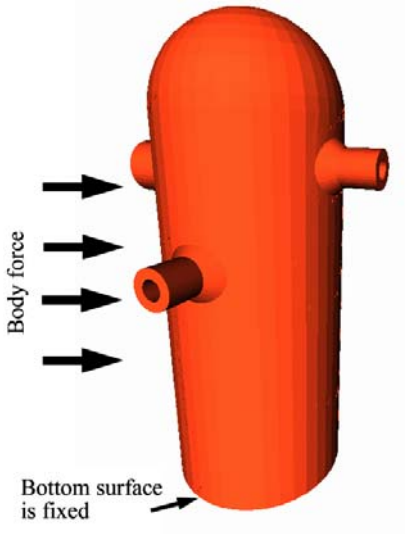
Machine	Cores	Type	Iter.	Time (min.)	Mem. (GB)	Mem./core (MB)
Cluster	24	**DS**	236	**21.3**	41.3	1,764
Cluster	48	**DS**	207	**13.2**	40.2	858
Cluster	24	**DSF**	236	**77.5**	11.0	471
Cluster	48	**DSF**	207	**40.9**	9.9	212
Xeon	8	**DSF**	351	**210.3**	13.4	1,494

DSFによる1億自由度解析

収束判定: 相対残差6桁

Cores	Subdomains	Iter.	Time (hr.)	Mem. (GB)	Mem./core (GB)
48	173,856	485	**6.46**	70.0	1.4

- Intel Core2Duo 24台構成のPCクラスタ上で1億自由度規模モデルを6.5時間で解析に成功
- うち2時間は約100万次元数のコース行列不完全コレスキー分解に要した



目次

- ✓研究背景と目的
- ✓ADVENTUREシステム
- ✓領域分割法(DDM)
- ✓BDD法, IBDD-DIAG法
- ✓1億自由度規模問題の数値実験例
- ✓Direct Storage-Free (DSF)法
- ✓メニーコアに向けた数値実験例
- ✓まとめ

まとめ

- IBDD-DIAG法をES, PRIMEQUEST 580, PCクラスタなどに移植し, 1億自由度規模の構造解析に成功した. 大規模解析向けの線形ソルバとしては, 反復回数が少なく, 並列効率が高いことが重要となる.
- メニーコアCPU向けのDDM実装としてDirect Storage-Free (DSF)法を提唱し, 実装を行った. メニーコアCPUではメモリアクセスが少ないことが重要となる.
- DSFによって, PCクラスタ上で1億自由度解析に6.5時間で成功した.

今後の課題

- Hybrid並列化
- コース問題の近似方法