並列計算機能

主任研究員　桑原 匠史

Made in Japan のLSIデバイス設計CADシステム
Advance/TCADのご紹介
2015年6月30日（火）
アドバンスソフト株式会社

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

Index

- デバイスシミュレータ、プロセスシミュレータについて
- 並列計算について
- プロセスシミュレータ（拡散計算）の並列化
- デバイスシミュレータの並列化

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

デバイスシミュレータ、プロセスシミュレータについて

- デバイスシミュレータ
  - 有限体積法
  - 3次元
  - 構造格子
  - FORTRAN90
- プロセスシミュレータ（拡散計算）
  - 有限体積法
  - 3次元
  - 非構造格子
  - FORTRAN90

イオン注入計算との大きな違いは、逆行列を解くため、コア間の通信量が多く並列化効率がイオン注入計算程よくない

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

並列計算について

- 並列化手法
  - 分散メモリ型並列
  - 共有メモリ型並列
  - ハイブリッド型並列
- 使用ライブラリMPI（Message Passing Interface）:MPICH
- 格子生成
  - デバイスシミュレータ：独自メッシャー
  - プロセスシミュレータ：adventure mesh

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- 行列解法ライブラリ

  偏微分方程式の数値計算に現れる線型方程式及び固有値問題を解くための並列反復解法ソフトウェアライブラリ
  - Lis（Library of Iterative Solvers for linear systems）
  - PETSc（Portrable, Extensible Toolkit for Scientific Computation）

	Lis	PETSc
反復解法	22	16
前処理	10	16+(外部＋7)
演算精度	実数のみ	実数、複素数
環境	逐次、MPI、OpenMP、MPI+OMP	逐次、MPI、OpenMP、MPI+OMP
使用可能言語	C FORTRAN	C C++ FORTRAN

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

プロセスシミュレータ（拡散計算）の並列化

- 基礎方程式

$$\frac{dC_{P_1}}{dt} = \nabla D_{P_1} \nabla C_{P_1} - GR_{P_1}$$
$$\frac{dC_{P_2}}{dt} = \nabla D_{P_2} \nabla C_{P_2} - GR_{P_2}$$
$$\vdots$$
$$\frac{dC_{P_m}}{dt} = \nabla D_{P_m} \nabla C_{P_m} - GR_{P_m}$$

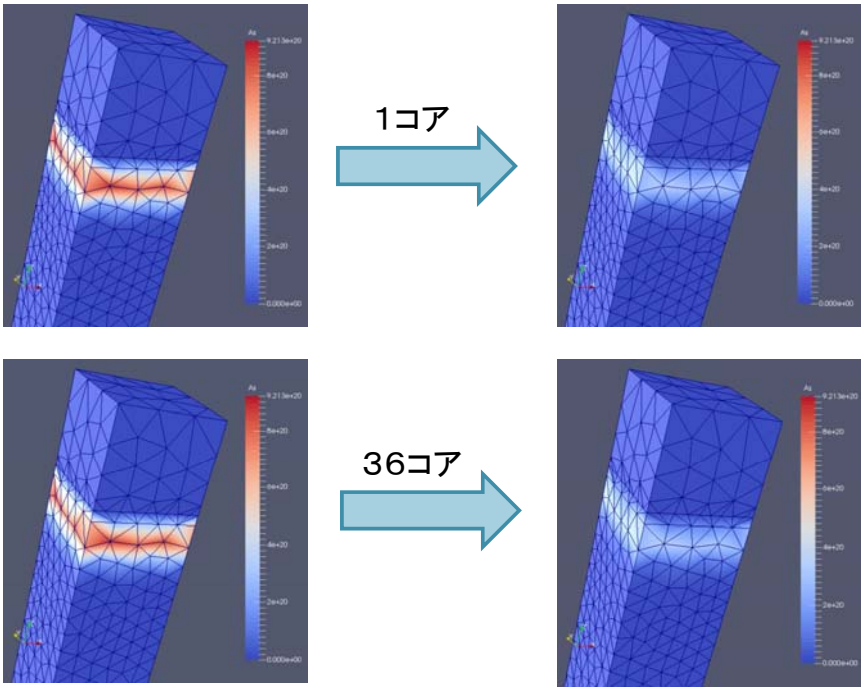
拡散種の数だけ
反応拡散方程式が必要

$$\nabla \varepsilon \nabla \varphi = -q\left(p - n + Q_{P_1} + Q_{P_2} + \cdots + Q_{P_m}\right)$$

全ての方程式をまとめて一つの行列として解く
デバイスシミュレーションに比べて処理は簡単

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

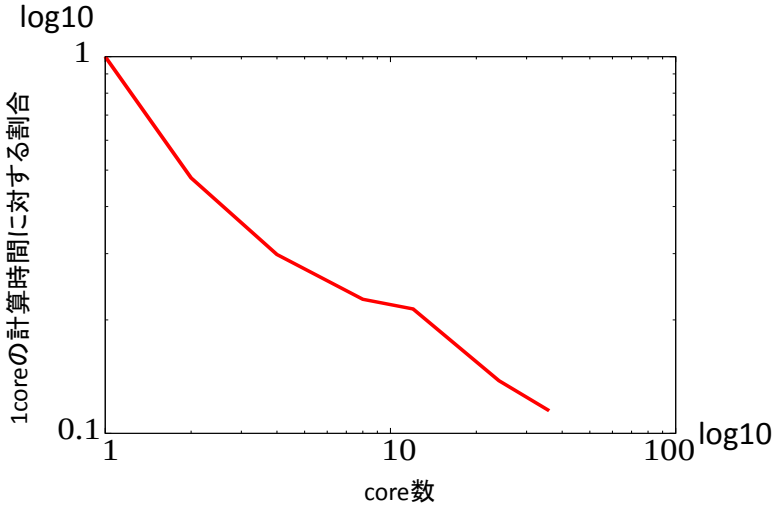
- Asの拡散の計算比較



並列計算においても同じ結果が得られる事を確認

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- 計算効率比較



Core数	割合
1	1
2	0.476618
4	0.298452
8	0.226937
12	0.213908
24	0.137997
36	0.114838

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

デバイスシミュレータの並列化

- 基礎方程式

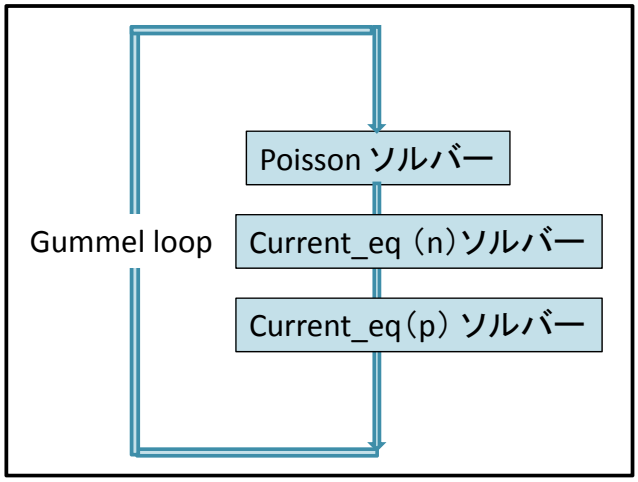
$$-\nabla(\varepsilon\nabla\psi)=q(-n+p+N_D-N_A)$$

$$\frac{\partial n}{\partial t}=\frac{1}{q}\nabla J_n-R$$

$$\frac{\partial p}{\partial t}=-\frac{1}{q}\nabla J_p-R$$

Gummel法を用いてこれらの方程式の解を求める

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.



計算手順の模式図

Poissonソルバーの行列の大きさと
Current_eqソルバーの行列の大きさが異なる

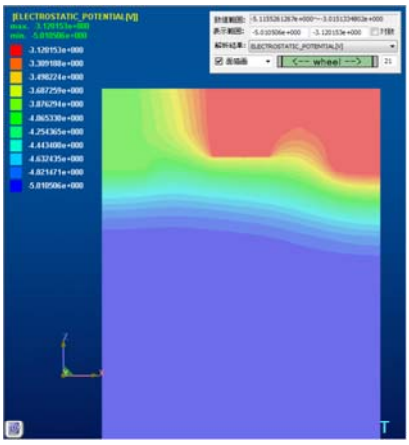
格子は行列を組み立てる前に分割する
ため、どちらも処理が均等になるように
分割することは難しい

PoissonソルバーとCurrent_eqソルバー
において解が判っている点（境界）に
ついても全部解く

計算が遅くならないか？

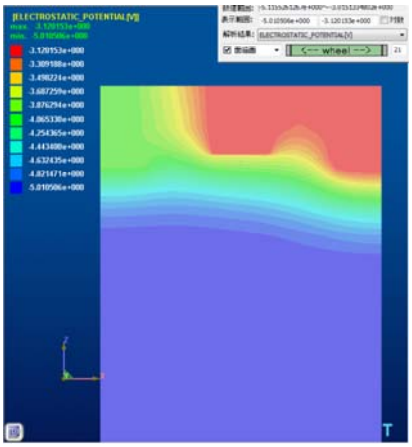
Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

元々の計算方法



CPU time..... 76.47 [sec]

境界まで含めて解く方法



CPU time..... 65.67 [sec]

計算時間にそれ程差はない

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- ポアソン方程式、電流連続方程式単体計算における性能

Poisson equation

コア数	計算時間[s]	収束回数
1	2.26	42
2	2.04	76
4	1.06	74
8	0.64	73
12	0.74	94

Current equation

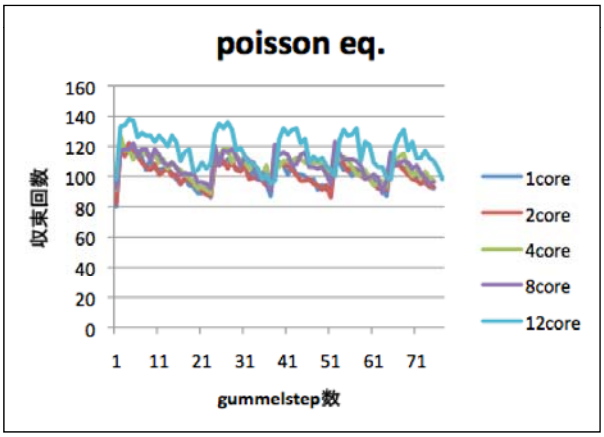
コア数	計算時間[s]	収束回数
1	44.68	896
2	24.04	997
4	12.43	986
8	17.48	2122
12	34.28	4942

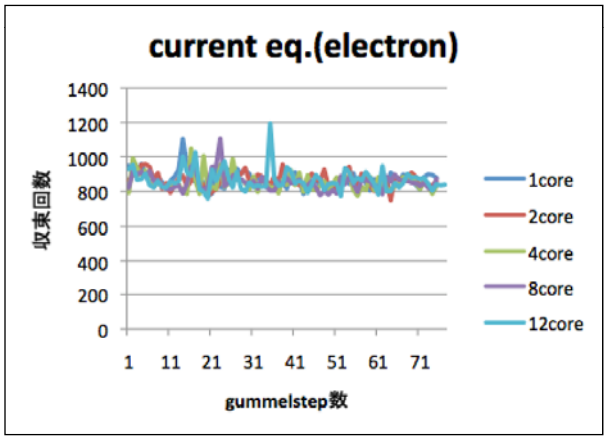
コア数を増やしていくと収束回数が劇的に増加してしまい効率が出ない。
特に電流連続方程式で顕著

収束回数を増やさないようにチューニング ⇨ 前処理とオプションが豊富なPETScに軍配

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- ポアソン方程式、電流連続方程式の性能（チューニング後のトータルな収束回数）

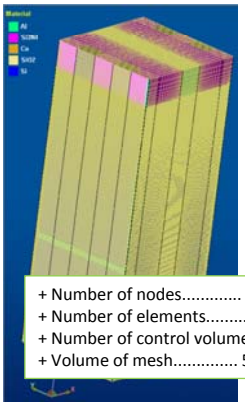




コア数を増やしていっても収束回数の増加が抑えられている
特に電流連続方程式で効果が顕著

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- 大規模モデルの計算
  - 基本モデルとそこから作成した2つのモデル（データ転送量による効率の比較）



69x34x495

+ Number of nodes..... 1215200
+ Number of elements..... 1161270
+ Number of control volumes... 1222366
+ Volume of mesh..... 5.171875E-21 [m3]

X-Y方向の分割のみ増やす：最も転送速度が出ないケース

+ Number of nodes..... 2604000
+ Number of elements..... 2522520
+ Number of control volumes... 2617844
+ Volume of mesh..... 5.171875E-21 [m3]

104x51x495

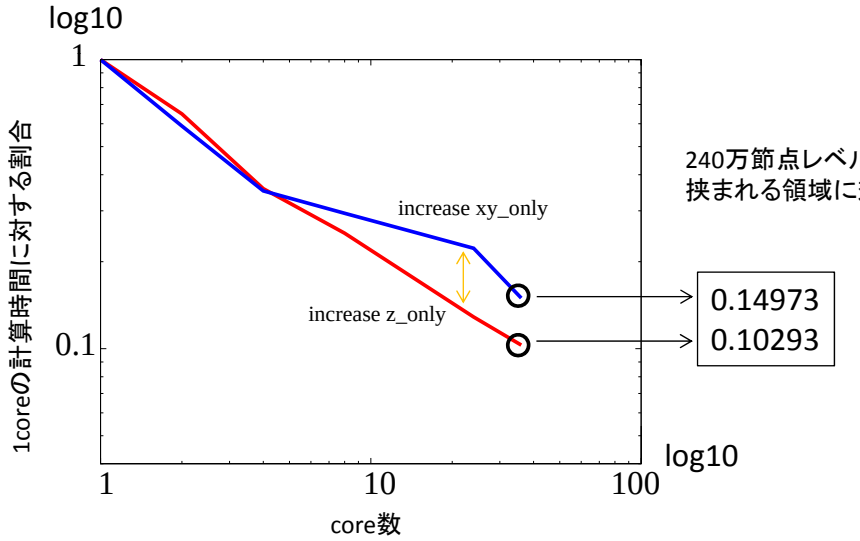
Z方向の分割のみ増やす：最も転送速度が出るケース

+ Number of nodes..... 2391200
+ Number of elements..... 2287350
+ Number of control volumes... 2398366
+ Volume of mesh..... 5.171875E-21 [m3]

69x34x975

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

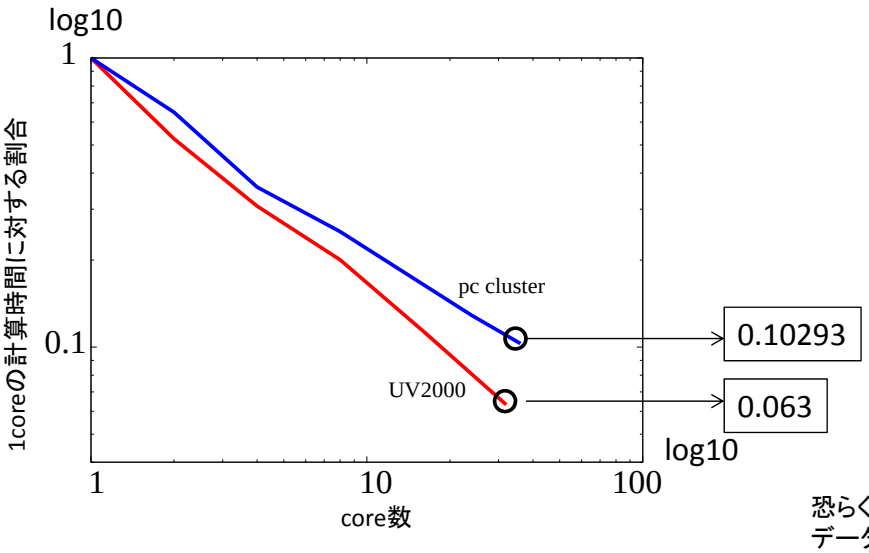
- 計算結果



240万節点レベルにおいて青線と赤線に
挟まれる領域に効率が収まる事が期待される

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

- 効率の環境依存性
  - UV2000における並列化効率の取得



恐らく、ハードウェアの違いによる
データ転送速度の高速化の影響では
ないかと考えられる

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.

まとめ

- デバイスシミュレータ、プロセスシミュレータ（拡散計算）の並列化をMPIライブライを用いた分散メモリ型並列を用いてった。
- プロセスシミュレータについては、弊社PCクラスタにおいて16コアで5倍以上の高速化を達成した。
- デバイスシミュレータについては、弊社PCクラスタにおいて36コアで約10倍、UV2000では16コアで8倍、32コアで16倍の高速化を達成した。

Copyright ⓒ2015 AdvanceSoft Corporation. All rights reserved.