

# やさしいcppmd (超並列高速MD計算コア)ソフトの紹介と 利用方法の詳細

理化学研究所  
生命システム研究センター(QBiC)  
生命モデリングコア 計算分子設計研究グループ  
上級研究員

計算科学研究機構(AICS)  
研究部門 プロセッサ研究チーム  
兼務

大野 洋介

# 目次

- 1. cppmd紹介(開発過程)
  - 1.0 cppmd概要
  - 1.1 計算手法としてのMD
  - 1.2 「京」を前提とした設計・開発
- 2. cppmd利用方法詳細
  - 2.1 構築方法
  - 2.2 使用方法
  - 2.3 計算性能評価

# 1. cppmd紹介(開発過程)

- 1.0 cppmd概要
- 1.1 古典分子動力学(MD)
- 1.2 数値計算としてのMD
- 1.3 「京」を前提とした設計・開発

# 1.0 cppmd 概要

- 次世代生命体統合シミュレーションソフトウェアの研究開発(ISLiM) 理化学研究所生命体基盤ソフトウェア・高度化チームで開発した大規模並列用MDコアプログラム
- 開発目的
  - 「京」運用開始時点で全システム規模の分子動力学 (Molecular Dynamics, MD)計算を可能にする。
    - 8万ノード(64万コア)の並列性能を有する。
    - 高いコア効率を有する。
  - 「京」のCPU・コンパイラの性能を引き出すノウハウを蓄積し、他のISLiMアプリケーション開発に反映する。
    - CPU・コンパイラの特徴を把握し最適化方法を確立する。
    - コードを転用して再利用する

# 基本仕様

- ファイル入出力
  - AMBER、独自バイナリ、HDF5
- 計算手法
  - 二体相互作用
    - カットオフ(GROMACS互換のShift-Function)、FMM、Zero-Dipole法(検証中)、PME
  - 結合力
    - 調和振動子近似(AMBER互換)、SHAKE/SETTLE/RATTLE(水素原子、水分子)
  - 積分
    - Velocity-verlet
    - Multiple time-step
  - アンサンブル
    - NVE、NVT(Nosé-Hoover)、NPT(Andersen-Hoover)

# cppmdの特徴

- 大規模並列性能
  - 8万CPU(64万コア)並列の weak scaling 90%  
(カットオフ法最適条件下)
    - MPI(CPU並列)とOpenMP(コア並列)のハイブリッド並列対応
  - 並列性の高いアルゴリズム
    - 高速多重極法(FMM)
    - Zero-Dipole法(検証中)
- カーネル(二体相互作用計算)性能
  - 二体相互作用計算部分の効率60%(最適条件下)
  - SIMD化率(最適条件下)

# 長所

- 高並列・高効率(ハイブリッド並列対応)
- 「京」で最適化されている
  - 「京」で最適化された二体相互作用計算ソース
- 高並列に適したFMMに対応している
- AMBER互換入出力

# 短所

- 「京」以外での最適化が不十分
- PME法は参照用で最適化が不十分
  - 小規模並列での性能は高くない
- 対応アルゴリズムが少ない
- ユーザーインターフェースが不十分
- AMBER以外の入出力に対応していない
- 8(2x2x2)MPI並列以上が必須
  - 非並列時の周期境界処理を実装していない

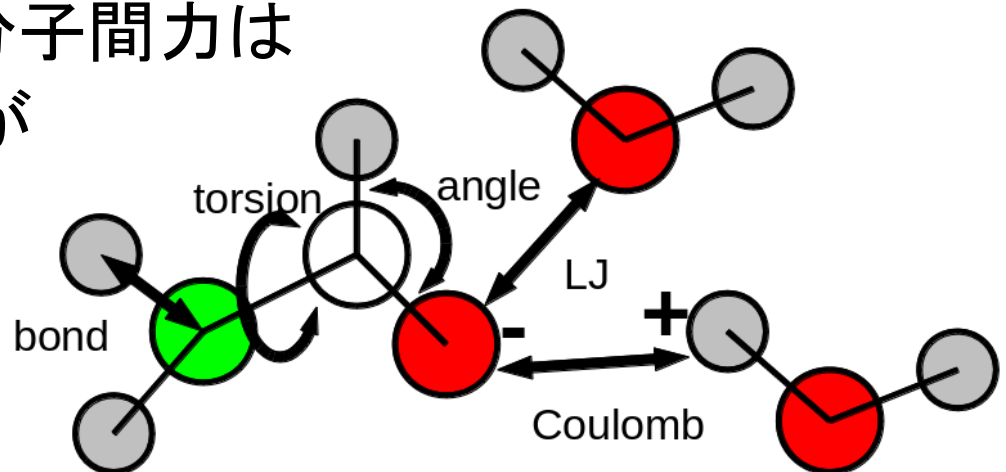


# 1.1 数値計算としてのMD

- 数値計算上のMDの特徴
- 実装上の注意点
  - 主な計算手法
  - 手法の比較

# 古典分子動力学(MD)

- 古典力学で分子・原子の運動を計算する
  - 運動方程式
  - クーロン力、分子間力、結合力
    - 量子力学・化学反応は扱わない。
    - 結合は調和振動子近似や距離・角度の拘束条件等で扱う
    - 生体分子の場合、分子間力はLennard-Jonesモデルがよく使われる



# MDの特徴

- クーロン力が遠距離相互作用

- 全原子が相互作用する。

- 長距離通信が必要

$$U = \sum_i \sum_{j>i} \frac{q_i q_j}{r_{ij}}$$

- N個の原子のクーロン力の演算量は $N^2$ に比例

- 一ペアの計算量も多い(～40演算)

→ 演算量が多い

- データあたりの演算量が多い

- メモリ速度はあまり問題にならない。

- クーロン力の計算量・通信量が計算時間を支配する

# クーロン力の演算量の削減

- 現在の生体分子計算での主流はPME(Particle Mesh Ewald)法
  - Ewald法
    - 周期境界条件化のクーロン力の無限和(Ewald和)
    - ガウス分布をかけてフーリエ変換を施し、波数空間で計算する。
      - 有限波数で打ちきる → 高周波成分をカットする
    - 近距離は点電荷とガウス分布の差を直接計算する
  - Particle Mesh
    - 電荷分布を格子(Mesh)で表現する
    - FFTを使う
  - 長距離は無限和も含めてFFTで計算し、短距離のペアは直接計算する。
- 演算量はFFTが $O(N \log N)$ 、短距離(カットオフ)が $O(N)$ に削減できる。

# PMEの問題点

- 周期境界条件を前提としている。
  - 非周期系に使うのは厳密な意味では近似
    - タンパク質の場合は周期サイズを大きくすることで孤立系との違いが少なくなると期待する。
    - 水の計算等では人工的な周期性が問題とされることもある。
    - 対称軸の異方性
- FFTの並列性
  - MDで使うサイズのFFTは数万並列では性能が低い
    - 一部のノードのみFFTを行なう、データの並べ替えを工夫しても、数千並列

# 多重極展開法(FMM)

- クーロン相互作用が遠方ほど弱くなる性質を利用
- 階層的に空間的な粗視化
  - 遠距離ほど大きい階層で相互作用を計算する。
    - 最下層では粒子ペアの計算
  - 八分木(Octree)が基本
    - 再帰的2x2x2の直方体分割
- 各階層の直方体内の電荷分布は多重極展開で表現する
- $O(N)$
- 長距離の通信量・回数が少ない
- 多重極の計算は負荷が大きい
  - 小規模では不利

# Multiple time-step

- 遠距離の原子からの力は、絶対値も時間変化も小さい
- 短距離より長いタイムステップにする。(時間的粗視化)

# カットオフ法

- カットオフ距離内のペアだけ計算する
  - $O(N)$
- 単純なカットオフ
  - 長距離ペアは無視
  - LJポテンシャルは1.2nm程度でも十分小さいが、クーロン力は2nmでも誤差が大きい
- 遠距離からの影響をペア計算に追加する。
  - Wolf法：電気的中性を仮定
  - Zero-Dipole法：dipole=0を仮定
    - Wolf法、ZD法はEwald法の近距離計算と類似した式になる
  - IPS：等方的一様性を仮定



# 短距離ペア計算の効率化

- カットオフ法、Ewald/PMEの近距離計算ではカットオフ距離内の粒子ペアのみ計算するが、カットオフするかどうかの判定に $O(N^2)$ の計算コストを費やしたのでは削減にならない。

## ➔ カットオフペア探索の効率化

- Cell Index法
- ペアリスト、近接粒子リスト

# Cell Index

- 原子を空間分割しておく。
  - 分割単位 : cell
- cell単位でペア判定することで全原子の判定をしなくてもよくなる。
  - 空間分割を変更しない限りcellペアは不変
- 空間分割はノード(プロセス)並列と兼ねることができる。
- cellの空間形状(通常は直方体)とカットオフ形状(球)の差の部分のペアが無駄となる。
  - カットオフ距離とcellサイズにもよるが、30-50%の無駄

# ペアリスト・近接粒子リスト

- ペアリスト：カットオフ距離内に入る可能性が高い原子ペアの一覧
- 近接粒子リスト：ある原子からカットオフ距離内に存在する可能性が高い原子の一覧
  - “可能性が高い”：運動も考慮してカットオフ距離より少し遠い原子も候補とする。
- 無駄なペアが少なくなる
- リスト作成コスト
  - Cell Index と併用
  - リスト更新間隔を長くとる。

# 結合力

- 結合の数  $O(N)$ 
  - 総演算量はクーロン力に比べて少ない
- タイムスケールが短い
  - クーロン力の計算に必要な時間刻みより、振動の周期が4-10倍短い
  - 結合力の振動に時間分解能をあわせるとクーロン力計算が不必要に増加する。



- 結合力は時間刻みを短くする(マルチプルタイムステップ)
- 距離等を一定値にする拘束条件を課す。
  - 微小振動を古典力場で近似するより一定値とするほうがよい

# 1.2 「京」を前提とした設計・開発

- 考慮する「京」の特徴
- 「京」に適した設計の検討
- 「京」のCPU向けの最適化

# 「京」の特有の条件

- 大規模並列
  - 8万CPU、64万コア
  - TOFU : 6次元トーラスネットワーク
    - 隣接通信が特に速い
- SPARC64 VIIIfx
  - スレッド並列
    - コア間同期等、スレッド並列用機能
    - ネットワークポートがコア数より少ない
  - SIMD拡張
    - 同一演算を複数データ(2セット)に施す

# 開発方針

- 「京」の利用
  - 大規模並列(8万CPU)
    - カットオフ法とFMMが本命
    - PMEは参照用
  - SPARC64 VIIIfx
    - 8コア、スレッド並列
    - SIMD型
- 他のライフアプリへのフィードバック
  - コンパイラでCPU性能を引き出す
  - 再利用性と性能の両立
    - 再利用性を期待してC++を選択
    - カーネル部分はCに近い記述(当初はFortran)

# 大規模並列

- 6次元トータラス(アプリケーションでは3次元扱い)
  - 隣接通信が速い。
  - 遠距離通信は遅延が大きくなる。
  - アルゴリズムから見直す
- FFTを使うPMEをやめる
  - 8万ノードの並列性は期待できない
    - FFTは遠距離通信が多い
    - MDの規模では通信単位が小さく、遅延の影響が大きい



# 大規模並列に適したMD計算手法

- 遠距離通信が少ない手法がよい
  - 遠距離相互作用の計算を削減する手法
- カットオフ法
  - 遠距離相互作用は無視するか、一様性の仮定等で近距離ペアの計算の補正で表現する。
    - 遠距離通信は不要
- FMM
  - 八分木(Octree)で遠距離相互作用を粗視化する。
    - 計算量が $O(N)$ になる。
  - 各回層は多重極展開で電荷分布を近似する。
    - 多重極計算が高負荷
  - 遠距離通信は8のべき乗で減少する
    - 通信は量・回数共に少ない。
  - 並進対称性が落ちる

# 遠距離計算手法の比較

	PME	単純カットオフ	Zero-Dipole	FMM
単体演算量	少	少	少	多
小規模性能	高	高	中	低
通信負荷	高 大域通信が多い	低 隣接通信のみ	低 隣接通信のみ	中 長距離通信が少ない
大規模性能	低 FFT性能に依存	高	高	中
精度	高	低	高	高
境界条件	周期境界	任意	任意	任意 (cppmdでは周期境界は近似処理)
普及率	高 低並列での主流	低	低	低 大規模での採用が増えつつある

# 通信コストの軽減

- 計算と通信のオーバーラップ
  - ノード内のデータのみで可能な計算とノード間通信を並行して実行
  - MDの場合、ノード内のcell/原子間の相互作用計算と隣接ノード間のcell/原子の転送
- XYZ順次通信
  - 軸方向の隣接通信を順次行なう。全ノードが同じ方向に通信することで、使用ネットワークの競合を避ける
  - 斜め方向や2段以上の隣接通信が多い場合に効果が期待される
    - 「京」のTOFUネットワークの場合、経路の冗長性等で、単純な3Dトラスより競合が少なくない

# cppmdでの選択

- PME : 参照用
- 単純カットオフ : 低精度で効率追求、  
最適化ノウハウの蓄積
- Zero-Dipole : 高精度かつ高性能  
適用条件の検証中
- FMM : 大規模並列で遠距離相互作用を陽に  
計算する場合

# その他の計算手法

- Multiple timestep
- 水分子、H-X結合のSHAKE/SETTLE/RATTLE
  - 全原子の拘束は並列化が困難と判断

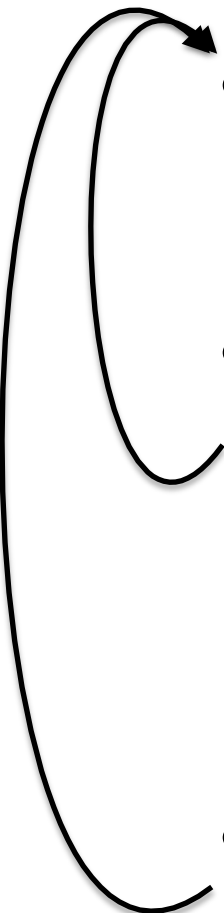
# その他の仕様

- AMBER互換入出力
- AMBER力場
  - LJパラメータ埋め込み
  - 結合力 AMBER topologyファイル
  - 共同研究者でAMBER利用者が多かったため。
- TIP3P water
  - 生体分子のMDでよく使われる
  - 拘束を適用しやすい。

# 二体相互作用カーネルの最適化

- ペアリスト／近接粒子リスト  
カットオフヒット率改善
- データパッキング変更  
キャッシュ効率改善
- Fortran カーネル
  - 現在はC++で同等の性能
- 構造体・STL Vector の排除(単純配列へのキャスト)：SIMD化・ソフトウェアパイプライン促進
  - コンパイラバージョンによっては不要
- if文の明示的マスク処理化：SIMD化促進
  - コンパイラによるマスク処理化がバージョン依存

# 「京」最適化手順

- 
- コード修正
  - コンパイラメッセージによる最適化適用状況の確認  
SIMD適用、ソフトウェアパイプラインング
  - プロファイル測定による障害要因の特定  
キャッシュヒット率、SIMD実行率



# SIMD化

- SIMD (Single Instruction Multiple Data)
  - 一つの命令で複数のデータを処理する
    - 「京」の場合一命令でデータ2セット

```
for(i=0;i<2*n;i++) a[i] += b[i]*c;
```

が

```
for(i=0;i<2*n;i+=2) {  
    a[i] += b[i]+b[i]*c; a[i+1] += b[i+1]*c; //同時実行  
}
```

のように半分のループ回数/命令数/時間で実行できる

# SIMD化の確認

- コンパイラメッセージ
  - オプション指定で適用状況が出力される
    - ループ単位およびループ内の行単位
- プロファイラによる実測
  - プロファイリング制御用の関数で測定区間を指定することで、目的の部分の詳細なSIMD実行比率等がわかる

# SIMD化阻害要因

- 不連続アクセス
  - メモリアクセスは連続の場合にSIMD化される
  - ペアリストのような間接参照はSIMD化されない
    - cppmdの場合カーネル部分の40%程度がSIMD化されていないメモリアクセス。演算命令だけなら99%SIMD化される。
- 条件分岐(if文)
  - 単純な計算ならmask付き演算に変換される
    - コンパイラのバージョンにもよる
  - cppmdの場合、二体相互作用カーネル部分のカットオフの真率が高いとわかっているので、明示的にmask処理のコードを書いた。

# ソフトウェアパイプライン

## ハードウェアパイプライン

1命令が複数の段階に分割されているので、複数の命令をずらしてオーバーラップして実行できる

依存関係があるとオーバーラップできない

ループをアンロールして依存関係のある演算待ちを軽減する。

```
for(i=0;i<n;i++){  
  a[i] += b[i]*c;  
  // a[i] の演算が終わるまで進  
  めない  
  d[i] = a[i]*e+f  
}
```

```
for(i=0;i<n;i+=2){  
  a[i] += b[i]*c;  
  a[i+1] += b[i+1]*c; // a[i]は関係  
  ないのでオーバーラップする  
  d[i] = a[i]*e+f; // a[i+1]のオー  
  バーラップで演算が終わっている  
  d[i+1] = a[i+1]*e+f;  
}
```

# オーバーラップの確認

- コンパイラメッセージ
  - オプション指定でソフトウェアパイプラインングが適用されたループがわかる。
- プロファイラによる実測
  - 当該ループ内の実行時間に占める演算待ちの比率をみる。
  - パイプラインングが効いていると演算待ちが少ない

# パイプライニング阻害要因

- レジスタ不足
  - レジスタに入り切らないとメモリアクセスが増えるのでアンロールは無制限にはできない
    - 「京」の場合演算レジスタは256個
      - ほかのCPUよりかなり多い
  - ループを分割して1ループ内の必要レジスタを減らす
    - 分割によって中間配列が増えすぎないように注意する
  - 二体相互作用カーネルの場合、演算量が多く、使い捨てる中間変数が多いのであまり問題にはならなかった。

# 開発中特有の問題点

- 8万並列の環境がない
  - 開始時はRICC(理研のクラスター)の8000コアで並列チューニング
  - 試験利用前は実機での実行は開発元に委託
  - 「京」の製造の進展とともに並列性の向上・確認
- コンパイラが開発中
  - 最適化性能が不十分、未実装機能・バグ
    - C++コンパイラの完成度が上がるまで、Fortran併用

# まとめ

- MDの最低限の機能を実装した。
- 大規模並列を前提として、設計・開発した
  - アルゴリズムの選択
  - 一般的並列化
- 二体相互作用計算を重点的に最適化した
  - 一般的な最適化
  - 「京」のCPU/コンパイラに特化した最適化



# 参考文献

- Ewald, PME  
T. Darden, D. York, L. Pedersen, Particle mesh ewald: An  $n \log(n)$  method for ewald sums in large systems, J. Chem. Phys. 98 (1993) 10089-10092.  
P. H unenberger, J. McCammon, Ewald artifacts in computer simulations of ionic solvation and ion-ion interaction: a continuum electrostatics study, J. Chem. Phys. 110 (1999) 1856-1872.
- FMM  
C. Lambert, T. Darden, J. Board Jr, A multipole-based algorithm for efficient calculation of forces and potentials in macroscopic periodic assemblies of particles, J. Comput. Phys 126 (1996) 274-285.  
R. Yokota, L. Barba, Hierarchical n-body simulations with autotuning for heterogenous systems, Comput. Sci. Eng. 14 (2012) 30-39.
- Cutoff  
D. Wolf, P. Keblinski, S. Phillpot, J. Eggebrecht, Exact method for the simulation of coulombic systems by spherically truncated, pairwise summation, J. Chem. Phys. 110 (1999) 8254-8282.  
X. Wu, B. R. Brooks, Isotropic periodic sum: a method for the calculation of long-range interactions, J. Chem. Phys. 122 (2005) 44107.  
I. Fukuda, Y. Yonezawa, H. Nakamura, Molecular dynamics scheme for precise estimation of electrostatic interaction via zero-dipole summation principle, J. Chem. Phys. 134 (2011) 164107.

# 参考文献

- その他

S. Miyamoto, P. Kollman, Settle: an analytical version of the shake and rattle algorithm for rigid water models, J. Comput. Chem. 13 (1992) 952-962.

H. Andersen, Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations, J. Comput. Phys. 52 (1983) 24-34.

S. Nosé, A unied formulation of the constant temperature molecular dynamics methods, J. Chem. Phys. 81 (1984) 511-519.

W. Hoover, Canonical dynamics: equilibrium phase-space distributions, Phys. Rev. A 31 (1985) 1695-1697.

- MDプログラム

D. A. Case, T. A. Darden, T. E. Cheatham, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Wang,

S. Hayik, A. Roitberg, G. Seabra, I. Kolossvary, K. F. Wong, F. Paesani, J. Vanicek, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke,

Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, V. Hornak, G. Cui, D. R. Roe, D. H. Mathews, M. G. Seetin, C. Sagui, V. Babin, T. Luchko, S. Gusarov,

A. Kovalenko, P. A. Kollman, B. P. Roberts, Amber 11, University of California, San Francisco, 2010. URL: <http://ambermd.org/#Amber11>.

D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. Berendsen, GROMACS: fast, flexible, and free, J. Comput. Chem. 26 (2005) 1701-1718.

- 教科書

分子シミュレーション — 古典系から量子系手法まで —、上田 顯、裳華房、ISBN 978-4-7853-1534-4

分子システムの計算科学 — 電子と原子の織り成す多体系のシミュレーション —、金田 行雄・笹井 理生監修・笹井 理生編、第2章 分子運動の計算科学、古明地 勇人、共立出版、ISBN 978-4-320-12271-0

## 2. cppmd 利用方法詳細

- 2.1 構築方法
- 2.2 使用方法
- 2.3 計算性能評価

## 2.1 構築方法

- GNU autotools 対応
  - 機能拡張等改変した場合
- configureスクリプトを生成済みの状態で配布
- PME・FMM対応等、有効にするにはconfigureに明示的に指定する必要がある。
- configureオプション化されていない機能選択はコンパイラへのマクロ定義オプション(-D)で指定する。

「京」のようなステージング運用の場合、実行バイナリが小さい方がジョブの負荷が軽減できるため必要な機能のみを有効にした方がよい

# PME

- configure オプション

- enable-pme (PME有効化、必須)

- with-fft=fftw2 (FFTW2選択、必須)

- with-boost=\${BOOST\_PATH} (BOOSTパス、非標準パスの場合)

- enable-simple-fft (推奨)

- マクロ

- CXXFLAGS+=" -I\${FFTW2INCLUDEDIR} " (FFTW2 include path)

- LDFLAGS+=" -L\${FFTW2LIBDIR} " (FFTW2 library path)

- LIBS+=' -lrfftw\_mpi -lfftw\_mpi -lrfftw -lfftw' (FFTW2 linker option)

- CXXFLAGS+=' -I\${BOOST\_INCLUDE\_PATH}' (BOOST include path)

- CXXFLAGS+=' -DCALCQ\_OPENMP' (Mesh関連処理のOpenMP対応、推奨)

# FMM

- configure オプション
  - enable-mr3exafmm (FMM有効化、必須)
- マクロ
  - CXXFLAGS+=' -DFMM\_CARTESIAN' (必須)
  - CXXFLAGS+=' -DPBOUND=1' (必須)

# カットオフ法

- マクロ

CXXFLAGS+=' -DUSE\_PAIRLIST -DINDEX\_PAIRLIST'

(必須)

CXXFLAGS+=' -DMAX\_PAIR=12000'

(ペアリスト最大サイズ)

CXXFLAGS+=' -DZERODIPOLE0' (係数0専用ZD)

CXXFLAGS+=' -DSIMPLE\_CUTOFF' (ポテンシャルエネルギーの絶対値が問題になる場合に必須)

CXXFLAGS+=' -DSIMPLE\_LJ' (ポテンシャルエネルギーの絶対値が問題になる場合に必須)

# 入出力

- **configure オプション**
  - with-static-hdf=\${HDF\_PATH} (HDF5有効化、無効の場合はnoを指定)
- **マクロ**
  - CXXFLAGS+=' -DCORRECT\_LEAPFROG\_RESTART'  
(restart fileがリープフロッグ用の場合)
  - CXXFLAGS+=' -I\${HDF5INCLUDEDIR}" (HDF5インクルードパス)
  - LD\_FLAGS+=' -L\${HDF5LIBDIR}" (HDF5ライブラリパス)



# 性能向上

- マクロ

CXXFLAGS+=' -DPRE\_CALC\_SF'

(カットオフ計算固定値の事前計算、推奨)

CXXFLAGS+=' -DTUNE\_CBMAP' (結合力最適化コード有効化、推奨)

CXXFLAGS+=' -DREDUCE\_COS' (結合力計算中のコサイン計算削減、推奨)

CXXFLAGS+=' -DOVERLAP -DOVERLAP\_PAIRLIST\_THREAD' (計算・通信のオーバーラップ有効化、推奨)

CXXFLAGS+=' -DMPIPARALLEL\_OPENMP' (MPI関連処理のOpenMP対応、推奨)

CXXFLAGS+=' -DORDERED\_RECV' (隣接通信順序の制御、推奨)

CXXFLAGS+=' -DNDEBUG' (デバッグコードの無効化、推奨)

# その他機能

- マクロ

CXXFLAGS+=' -DMT\_LONG=4' (PME,FMMのマルチプルタイムステップ間隔)

CXXFLAGS+=' -DUSE\_SHAKE -DSHARE\_LIST' (SHAKE有効化、推奨)

CXXFLAGS+=' -DTIMER\_DETAIL' (計算時間の詳細表示)

# 「京」固有

- configure オプション
  - host=sparc64-unknown-linux-gnu (ターゲットアーキテクチャ指定)
  - build=x86\_64-unknown-linux-gnu (ビルド環境指定)
- マクロ
  - CXX=mpiFCCpx (コンパイラ指定)
  - CXXFLAGS+=' -Kfast,openmp,array\_private,ilfunc,ocl,NOFLTLD,simd=2,preex'
  - CXXFLAGS+=' -Kprefetch\_indirect,prefetch\_cache\_level=1,prefetch\_iteration=8,prefetch\_iteration\_L2=16'
  - CXXFLAGS+=' -V -Koptmsg=2 -Kmfunc=3 -Nsrc,sta'  
(コンパイラ最適化オプション)
  - CXXFLAGS+=' -DK\_SIMD -DSPARC\_SIMD'  
(「京」SIMD用コード有効化)

# configure例

- 典型的な例はサンプルとして配布

```
VERSION=_209.4
#MPICXX=mpiFCCpx
MPICXX=mpiFCCpx
#CXX=mpiFCCpx
CXX=mpiFCCpx
CXXFLAGS+= ' -
Kfast,openmp,array_private,ilfunc,ocl,NOFLTLD,simd=2,preex'
CXXFLAGS+= ' -
Kprefetch_indirect,prefetch_cache_level=1,prefetch_iteration=8,p
refetch_iteration_L2=16'
CXXFLAGS+= ' -V -Koptmsg=2 -Kmfunc=3 -Nsrc,sta'
CXXFLAGS+= ' -DK_SIMD -DSPARC_SIMD'
#CXXFLAGS+= ' -DDEBUG_MOMENTUM'
CXXFLAGS+= ' -DORDERED_RECV'
CXXFLAGS+= ' -DOVERLAP -DOVERLAP_PAIRLIST_THREAD'
CXXFLAGS+= ' -DMT_LONG=4'
CXXFLAGS+= ' -DTUNE_CBMAP'
CXXFLAGS+= ' -DFMM_CARTESIAN'
CXXFLAGS+= ' -DK_SIMD'
CXXFLAGS+= ' -DNDEBUG'
CXXFLAGS+= ' -DUSE_SHAKE -DSHARE_LIST'
CXXFLAGS+= ' -DTIMER_DETAIL'
CXXFLAGS+= ' -DUSE_PAIRLIST -DINDEX_PAIRLIST'
CXXFLAGS+= ' -DMPIPARALLEL_OPENMP'
CXXFLAGS+= ' -DMAX_PAIR=12000'
CXXFLAGS+= ' -DPRE_CALC_SF'
CXXFLAGS+= ' -DPBOUND=1'
```

```
CXXFLAGS+= ' -DCALCQ_OPENMP'
CXXFLAGS+= ' -DREDUCE_COS'
CXXFLAGS+= ' -DZERODIPOLE0'
CXXFLAGS+= ' -DSIMPLE_CUTOFF'
CXXFLAGS+= ' -DSIMPLE_LJ'
CXXFLAGS+= ' -DCHECK_ENERGY'
#CXXFLAGS+= ' -DBINARY_DUMP'
#CXXFLAGS+= ' -DBINARY_RESTORE'
#CXXFLAGS+= ' -DCORRECT_LEAPFROG_RESTART'

#FFTW2DIR='/home/apps/fftw/2.1.5'
#CXXFLAGS+= " -I${FFTW2DIR}/include"
#LDFLAGS+= " -L${FFTW2DIR}/lib64"
#LIBS+= ' -lfftw_mpi -lfftw_mpi -lfftw -lfftw'
#CXXFLAGS+= ' -I/home/hp120068/k00155/boost-cross/include'
HDF5DIR='/data/hp120068/k00155/build_h5/hdf5'
CXXFLAGS+= " -I${HDF5DIR}/include"
LDFLAGS+= " -L${HDF5DIR}/lib"
../cppmd${VERSION}/configure \
--host=sparc64-unknown-linux-gnu --build=x86_64-unknown-
linux-gnu \
MPICXX="${MPICXX}" CXX="${CXX}" CXXFLAGS="${CXXFLAGS}" \
LDFLAGS="${LDFLAGS}" LIBS="${LIBS}" \
--enable-mr3exafmm \
--with-static-hdf=/data/hp120068/k00155/build_h5/hdf5
```

## 2.1 使用方法

- 実行時オプション
- 「京」のジョブスクリプトサンプル
- MD計算の手順

# 実行時オプション

- 並列関連
- 入出力
- 計算条件
- カットオフ関連
- 遠距離計算関連
- PME関連
- FMM関連

# 並列オプション

- 並列形状指定

`--nodediv3d={X}x{Y}x{Z}` 推奨

- ノード内セル分割指定

`--celldiv3d-in-node={x}x{y}x{z}` 推奨

- コピー指定

`--copynum3d={i}x{j}x{k}`

入力ファイルの計算対象を複数並べて大きな系として計算する

# 入出力オプション

- AMBER topology 入力ファイル  
--prmtop=\${paramtop} 必須
- AMBER restart 入力ファイル  
--restrt=\${restart} 必須
- AMBER restart 出力ファイル  
--mdrst=\${output\_restart}
- AMBER coordinate 出力ファイル  
--mdcrd=\${output\_coordinate}
- ファイル出力間隔  
--md{crd|rst}-interval=\${output\_interval}



# 計算条件オプション

- 計算ステップ数
  - m  $\{\text{ステップ数}\}$  必須
- 時間刻み
  - delta-t= $\{\Delta t(\text{fs})\}$  推奨
- アンサンブル指定
  - tempctrl-method={none | nose-hoover | andersen-hoover}
  - 温度・圧力制御
    - temperature= $\{\text{温度(K)}\}$  --tau-t= $\{\text{温度制御の速さ}\}$  --tau-p= $\{\text{圧力制御の速さ}\}$  --tempctrl-interval= $\{\text{制御間隔}\}$
- 統計量計算間隔
  - r  $\{\text{エネルギー、温度等の計算間隔}\}$
- 統計量出力間隔
  - p  $\{\text{エネルギー、温度等の出力間隔}\}$

# カットオフ関連オプション

- カットオフ距離
  - C  $\{\text{cutoff}(\text{\AA})\}$  必須
- 近距離ポテンシャルタイプ
  - coulomb-type={ewald | direct | zerodipole} 必須

# 遠距離計算オプション

- 遠距離計算有効化
  - L 遠距離計算の場合必須
- PME格子間隔
  - pme-grid-length=\${格子間隔(Å)}
- PME Parameter
  - pme-alpha=\${Ewald法の係数} --pme-order=\${charge assign の次数}
- 担当ノード割り当て方法
  - calc-space=\${DIM}\${share} 遠距離計算の場合必須
    - DIM : 0,1,2,3 遠距離担当ノード並列化次元数
    - share : 0 短距離計算ノードを兼ねる、1 独立ノード

# その他

- 固定 (全て必須)
  - B --citype=0
  - short-comm-pattern=0 --move-comm-pattern=0
  - shake-type=2
- メッセージ
  - v 1 (推奨)

# 実行スクリプト例(「京」)

```
#!/bin/sh
#PJM --rsc-list "rscgrp=huge"
#PJM --rsc-list "node=46656"
#PJM --rsc-list "elapse=01:00:00"
#PJM -s
#
#PJM --stg-transfiles all
#PJM --mpi "use-rankdir"
#PJM --vset "INTOP=vivo.top"
#PJM -x "INTOP=${INTOP}"
#PJM --vset "INRST=vivo.rst"
#PJM -x "INRST=${INRST}"
#PJM --vset "MAXSTEP=1000"
#PJM -x "MAXSTEP=${MAXSTEP}"
#PJM --stgin "rank=* ./cppmd %r:/"
#PJM --stgin "rank=0 ../ttha/${INTOP} 0:../"
#PJM --stgin "rank=0 ../ttha/${INRST} 0:../"
#PJM --stgout "rank=* %r:hfdump* ./vivo_n36x36x36_C12_cp2_1000/"

./work/system/Env_base_1.2.0-15

export OMP_NUM_THREADS=8

export NODEDIV3D="36x36x36"
export CELLDIV3DINNODE="2x2x2"
export COPYNUM3D="2x2x2"
```

```
export CUTOFF="12"

#export LONGMODE=""
#export COULOMB="zerodipole"
export LONGMODE="--calc-space=30 -L"
export COULOMB="direct"

export TEMPCTRL="--tempctrl-method=none"
export DELTAT="1.0"

export CRDINTERVAL="10"

export LARGEPAGE=32

export PRINT='4'

CMD="mpiexec ./cppmd ${LONGMODE} --citype=0 -m ${MAXSTEP} -p ${PRINT} -r
${PRINT} -B -C ${CUTOFF} --prmtop=../${INTOP} --restr=../${INRST} -v 1 --short-
comm-pattern=0 --move-comm-pattern=0 --coulomb-type=${COULOMB} --shake-
type=2 --nodediv3d=${NODEDIV3D} --celldiv3d-in-node=${CELLDIV3DINNODE} --
copynum3d=${COPYNUM3D} ${TEMPCTRL} --delta-t=${DELTAT} "

$CMD
```

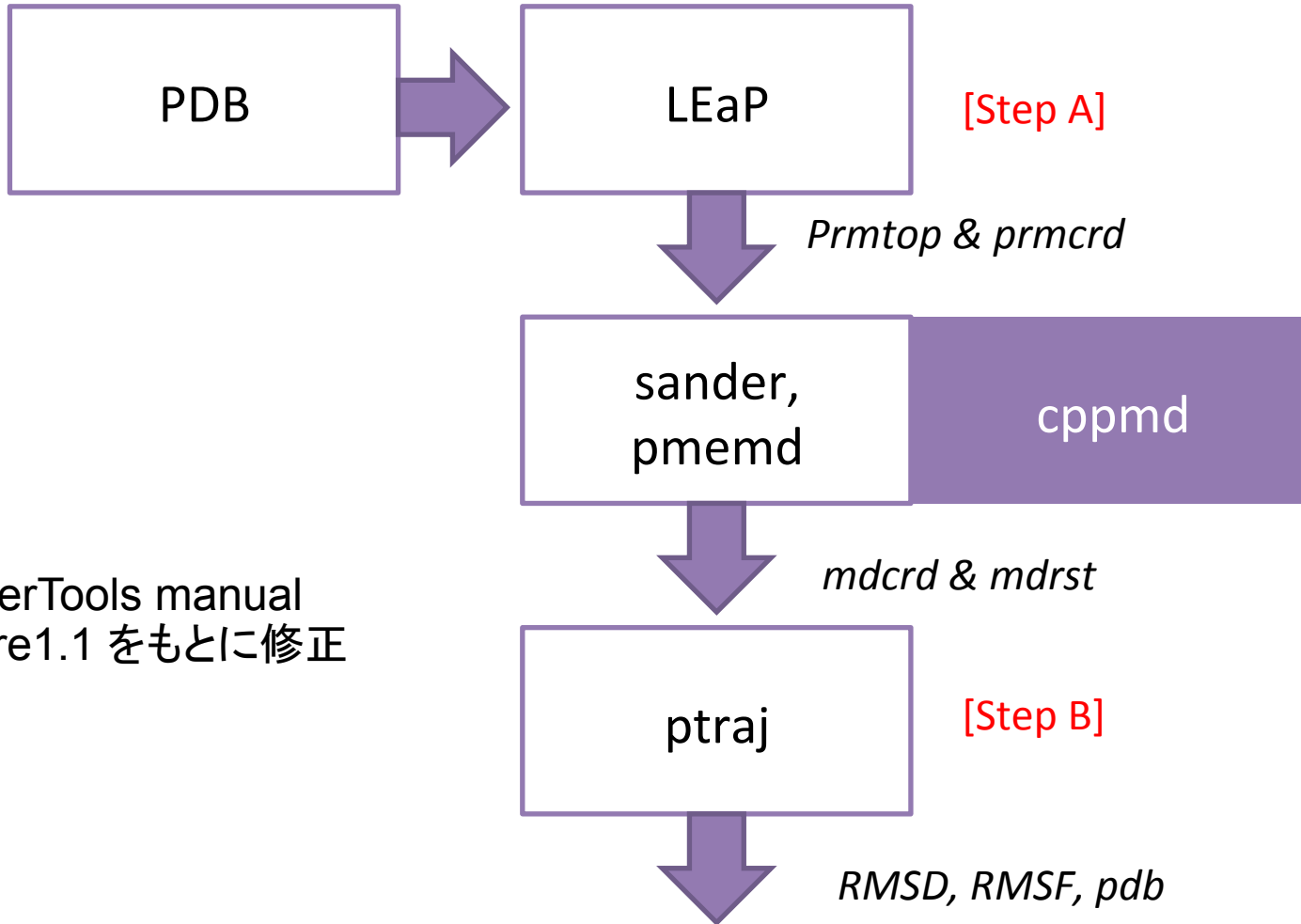
# 実行時標準出力

- 実行コマンドライン
- オプション値
- 自動決定パラメータ
- 統計量推移
  - タイムステップ、全エネルギー、運動エネルギー、ポテンシャルエネルギー、温度、圧力(圧力制御時)
- 計算時間
  - 全体
  - 主要機能内訳

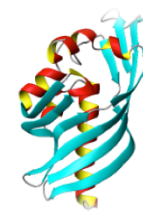
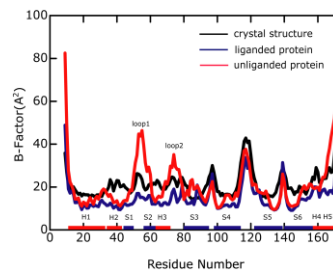
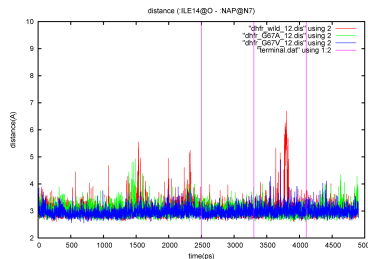
# MD計算の手順

- AMBERのsander, pmemdと置き換えて使う
- 入力データ作成、結果解析はAmberToolsを使う
  - 初期速度は別途MDコードで計算する
    - 初期速度0(温度0K)からの昇温計算等

# flow



AmberTools manual  
Figure1.1 をもとに修正





# 入力データ作成[Step A]

\$AMBERHOME/bin/tleap (tleap起動)

以下tleapのコマンドプロンプト

source oldff/leaprc.ff99 (基本設定の読み込み)

sample = loadPdb input.pdb (入力PDBを読む)

parm99 = loadamberparams parm99.dat (力場の設定)

addions sample Na+ 5.0 (イオンを付加)

addions sample Cl- 0.0 (イオンを付加)

solvatebox sample TIP3PBOX {15.0 15.0 15.0} 1.0 (箱状の水分子を発生)

saveAmberParm sample prmtop prmcrd (prmtopとprmcrdを出力)

savepdb sample out.pdb (pdbを出力)

quit

# 結果解析[Step B]

- `$AMBERHOME/bin/ptraj prmtop ptraj.in`

Ptraj.in:

`trajin comcrd`

(md結果の座標データの読み込み)

`reference prmcrd`

(rmsdを取る参照構造の読み込み)

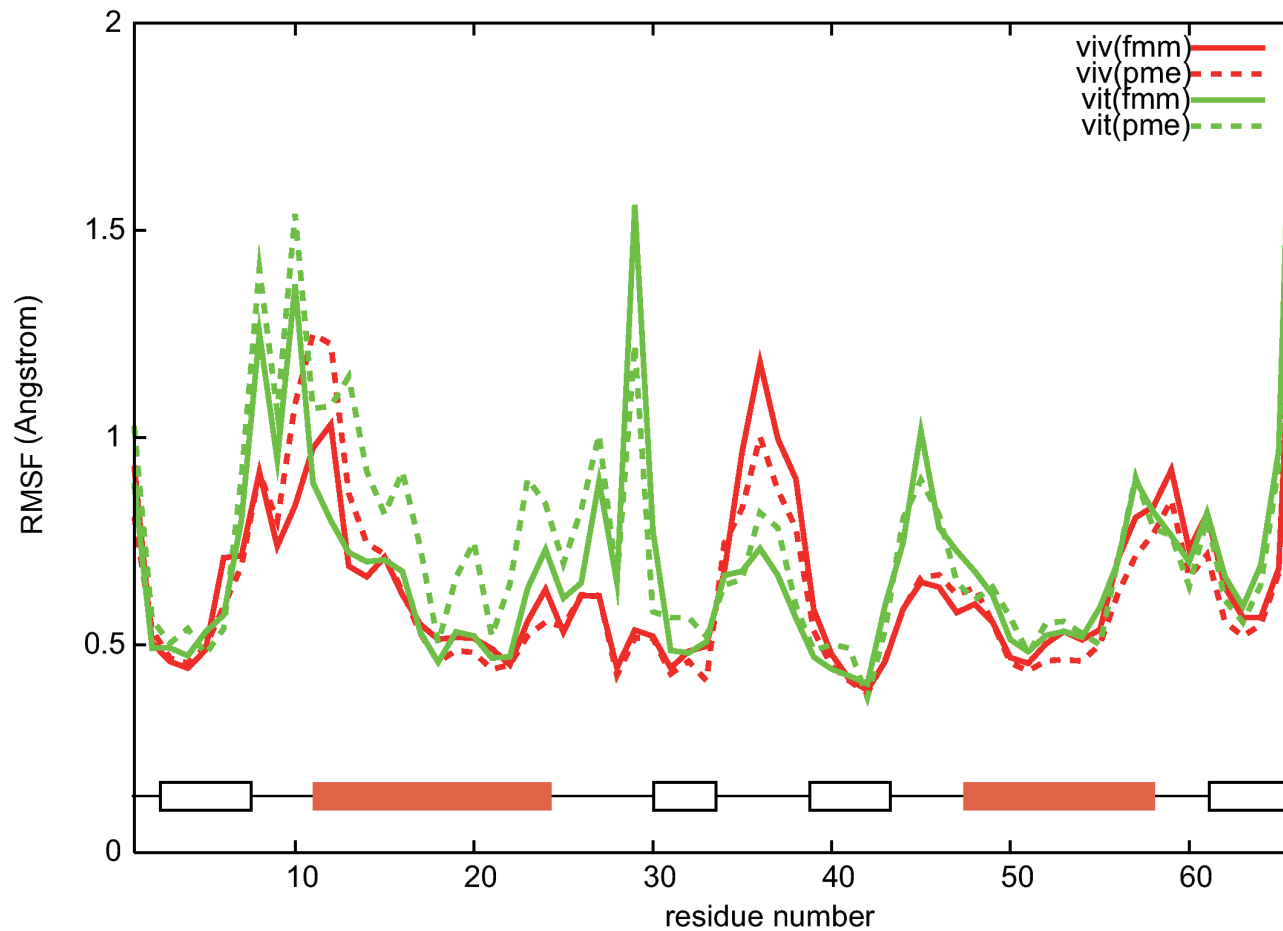
`Rms reference out rmsd.out time 10 :1-5@CA`

(1-5番の $\alpha$ 炭素のrmsdを測定し、rmsd.outに出力)

`atomicfluct out rmsf.out :1-5@CA byres`

(1-5番の $\alpha$ 炭素を使ってrmsfを測定し、rmsf.outに出力)

# RMSFの計算例



主鎖のゆらぎを、高濃度/低濃度、PME/FMMの組み合わせで比較

## 2.3 計算性能評価

- カットオフカーネル性能
- 並列化性能

ISLiMおよび、平成24-25年度「京」を中核とするHPCIシステム利用研究課題 一般利用課題  
hp120068の成果

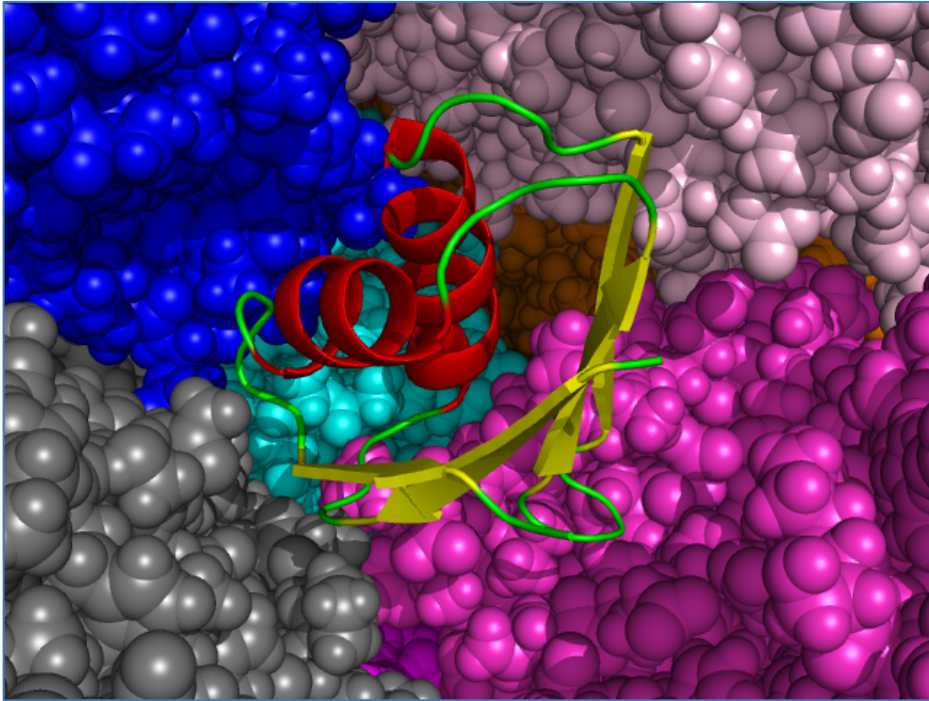
# カットオフカーネル性能

Number of atoms per CPU	6,542
Cutoff radius	28 Å + 2 Å margin
Number of pairs per atom	8,835( + 2,031 in margin)
Kernel FLOPS per CPU	80, 516 MFLOPS
Efficiency	62.9 %
SIMD ratio	57.2 % *

\* No-SIMD instructions are load/store

99 % of floating-point calculations are SIMD

# 大規模計算性能評価



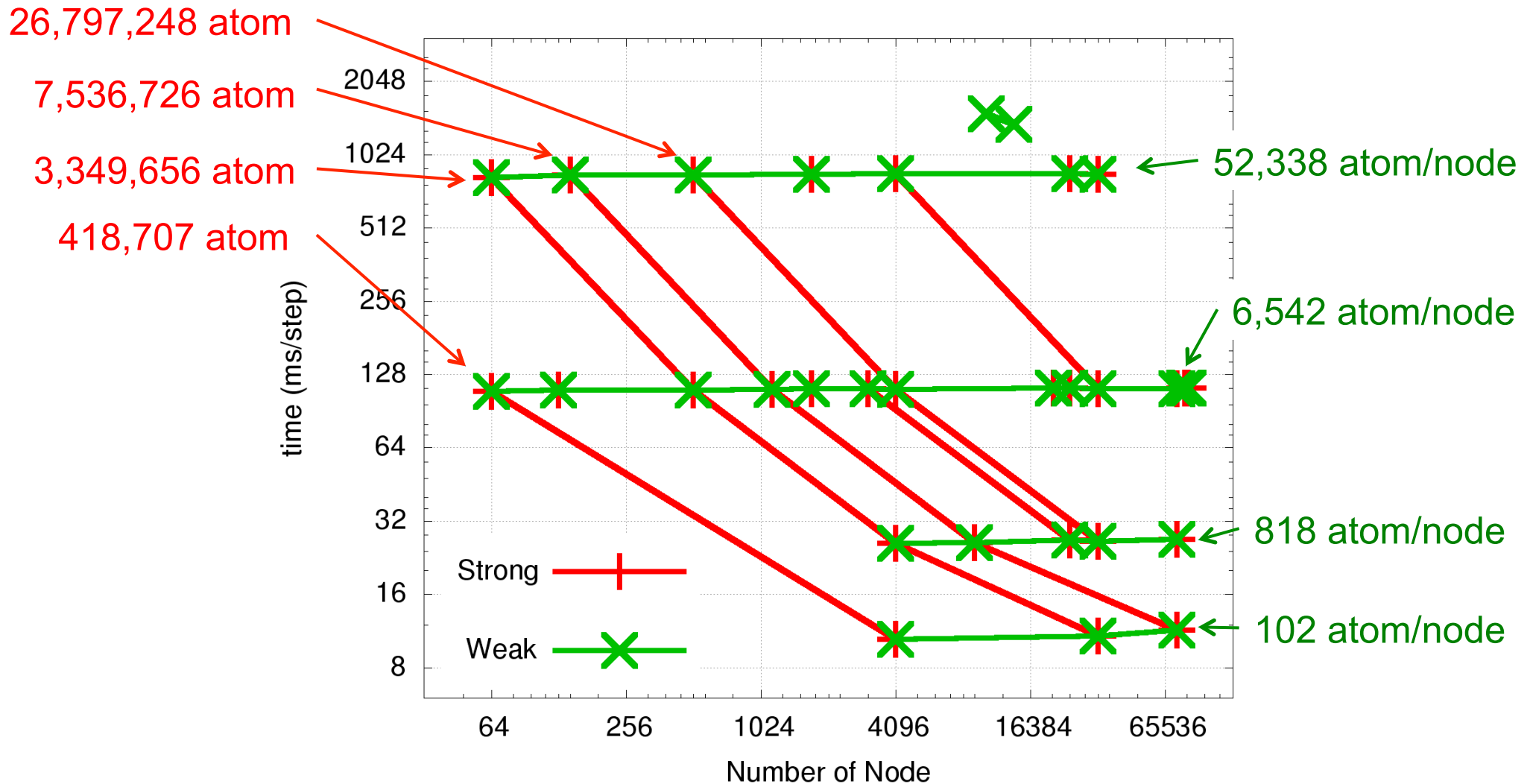
- 水中タンパク質
  - 低濃度、高濃度(左図)
  - 418,707原子、  
1,344,488原子
- 繰り返しコピーで  
542,644,272原子まで  
拡張

# カットオフ法の大規模性能

	Mar 2012			Oct 2012	Mar 2013
node3d	48x52x32	48x52x32	48x52x32	48x48x36*	48x48x36*
#node	79,872	79,872	79,872	82,944	82,944
#atom	522,546,336	522,546,336	522,546,336	542,644,272	542,644,272
ms/step	132.789	116.357	112.414	112.085	113.811
energy	1	1	1/4	1/4	1
PFLOPS	3.846	4.387	3.871	4.031	4.658
efficiency	0.376	0.429	0.379	0.380	0.439

通信オーバーラップ、コンパイラバージョンアップ \*TIFUのトーラス形状と異なる

# カットオフ法の並列性能



Cutoff 28Å, calculate energy every 4 step



# Weak Scaling

Number of nodes	64	512	4,096	32,768	79,872	82,944
Topology	4x4x4	8x8x8	16x16x16	32x32x32	48x52x32	48x48x36*
Number of atoms	418,707	3,349,656	26,797,248	214,377,984	522,546,336	542,644,272
Time (ms / step)	109.058	110.535	111.186	111.672	112.414	112.085
Force	91.622	91.528	91.601	92.329	91.262	91.641
Communication	12.124	13.694	14.011	14.012	15.820	15.067
Other	5.312	5.313	5.574	5.332	5.332	5.376
Performance (PFLOPS)	0.003	0.025	0.201	1.599	3.871	4.031
Efficiency (%)	39.0	38.5	38.3	38.1	37.9	38.0

Cutoff 28Å, 6,542 atom/node, calculate energy every 4 step

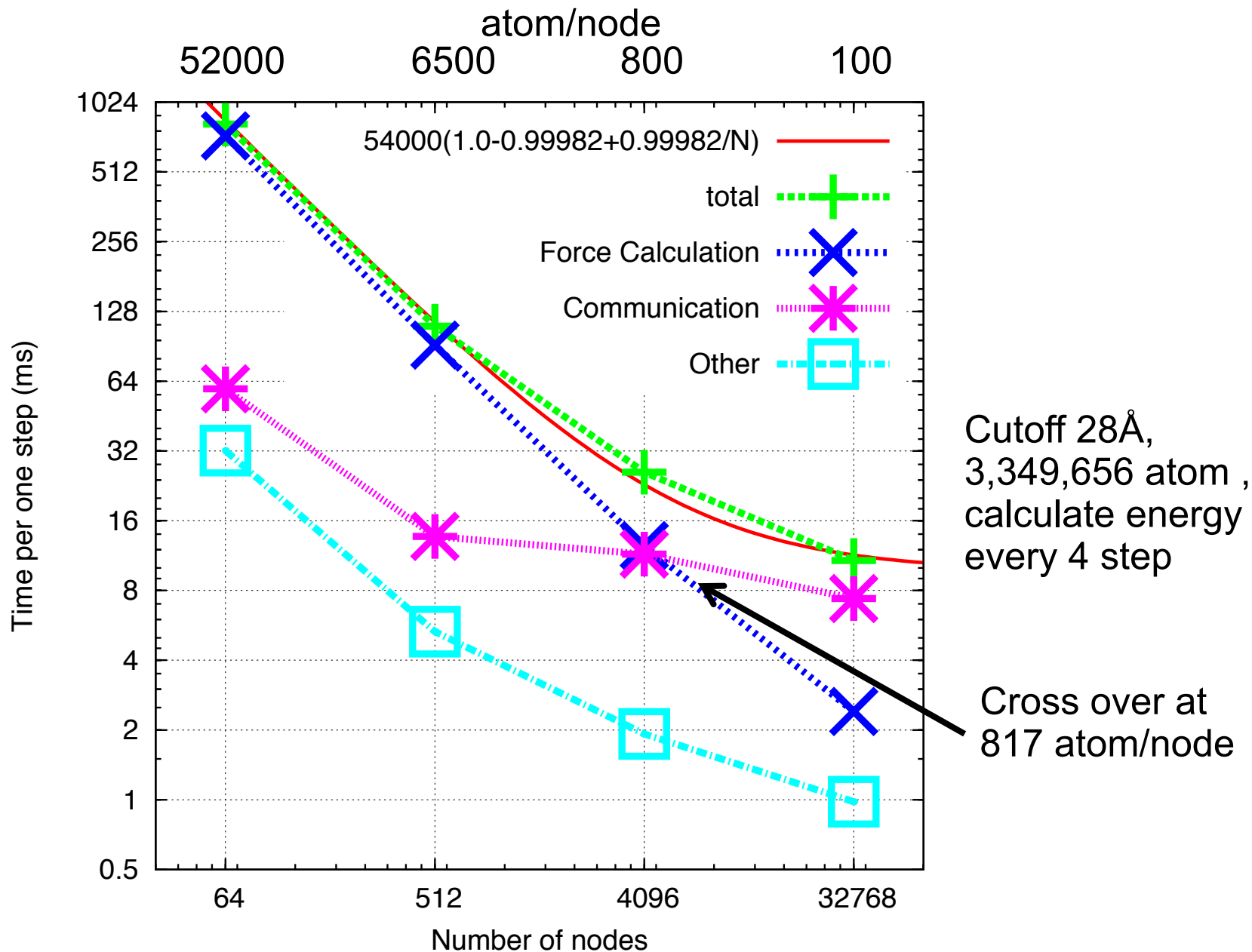
\* Different to physical geometry 48x54x32

# Strong Scaling

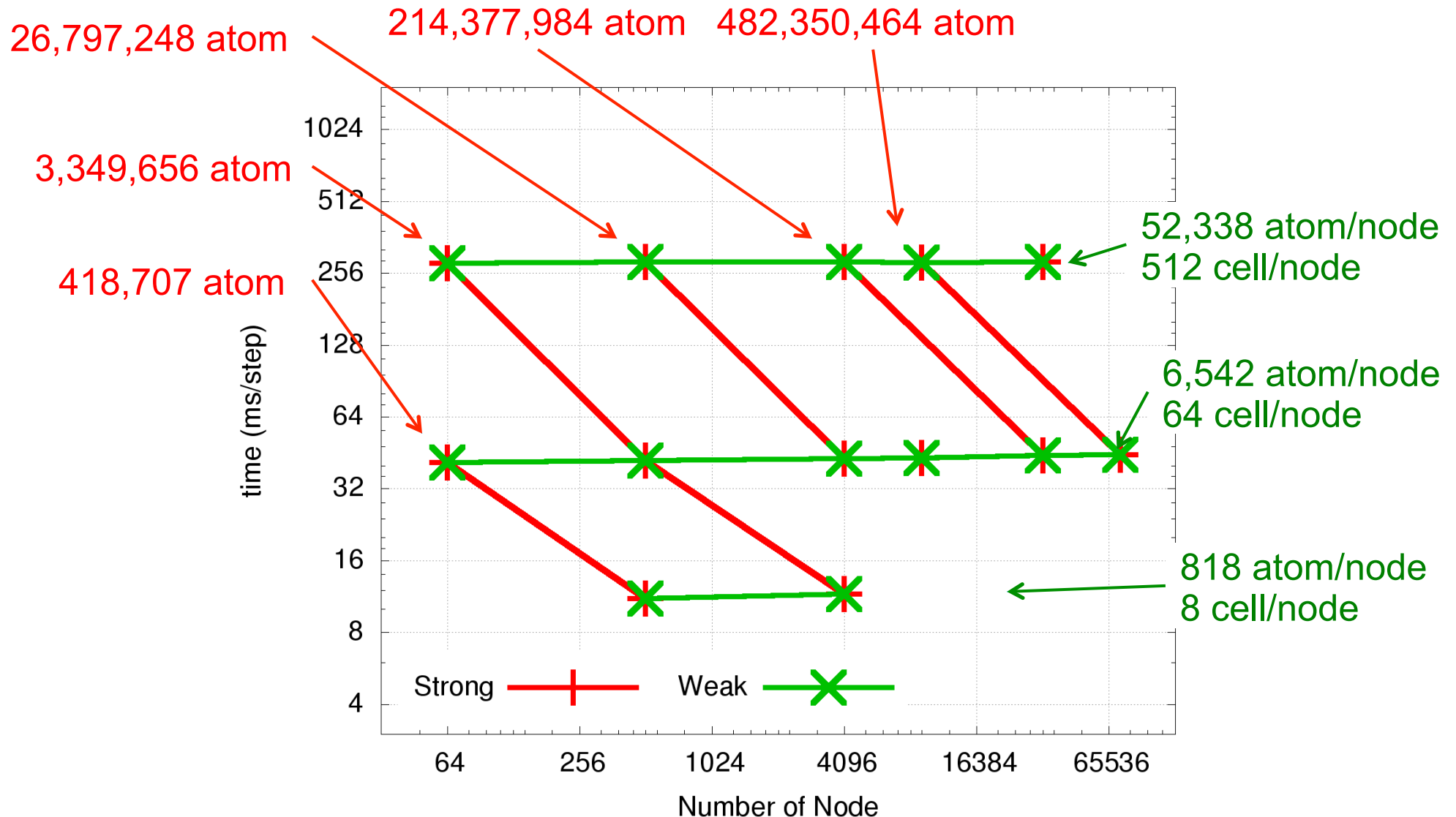
Number of nodes	64	512	4,096	32,768
Number of atoms per node	52,338	6,542	817	102
Number of transfer cells	2,232	936	504	342
Number of communication nodes	26	26	124	342
Time (ms / step)	823.07	110.54	25.90	10.77
Force	731.66	91.53	12.50	2.41
Communication	59.28	13.69	11.48	7.38
Other	32.13	5.31	1.92	0.98
Scalability ( to 64 nodes)	1.00	0.93	0.50	0.15

Cutoff 28Å, 3,349,656 atom , calculate energy every 4 step

# Strong Scaling (内訳)



# FMMの並列性能



# ZD法の大規模性能

京でなければ検証できない並列規模で確認した。

	Shift function	Zero-dipole				
	Mar 2013	May 2013				
node	48x48x36*	48x52x32		48x48x36*		
#node	82,944	79,872		82,944		
Target	Vitro	Vitro		Vivo		
#atom	542,644,272	522,546,336		516,283,392	64,535,424	
Cutoff(Å)	28	20	12	12	18	18
ms/step	113.811	68.569	30.568	35.336	63.518	20.195

\*TOFUのトーラス形状と異なる

# 性能限界

- 「京」では1ms/step (2fs/step として172ns/day)
  - サブミリ秒積分は1年単位
  - ネットワーク性能の限界
    - 100原子/ノード程度が限界
- ノード性能が高く並列度が低い計算機が必要

# 今後のMD

- 小規模長時間積分 ex ミリ秒計算
  - (準)専用計算機
    - MDGRAPE、ANTON
      - D. Shaw, R. Dror, J. Salmon, J. Grossman, K. Mackenzie, J. Bank, C. Young, M. Denero, B. Batson, K. Bowers, et al., Millisecond-scale molecular dynamics simulations on anton, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ACM, 2009, p. 39.
      - 次期スパコンのアクセラレータ型
- 大規模 ex 細胞(組織)サイズ
  - 大規模並列スーパーコンピューター

# 参考文献

- AMBER tools manual  
<http://ambermd.org/doc12/index.html>
- 平成24年度「京」を中核とするHPCIシステム  
利用研究課題 中間報告会 一般利用課題  
hp120068発表資料
- 平成25年度「京」を中核とするHPCIシステム  
利用研究課題 中間報告会 一般利用課題  
hp120068発表資料



# 謝辞

- 「京」でのコード開発・最適化・性能評価に、理化学研究所のスーパーコンピュータ「京」を利用しました。(「京」試験利用、一般利用(平成24,25年度 課題番号:hp120068))
- FX10(「京」互換機)での動作検証にSCSL計算機システムを利用しました。
- 「京」以外でのコード開発に理化学研究所情報基盤センターの RIKEN Integrated Cluster of Clusters (RICC) を利用しました。