

# OmniRPCによる 分子立体配座探索プログラム CONFLEXのグリッド化

中島佳宏\* 佐藤三久+ 朴泰祐+ 高橋大介+ 後藤仁志#  
\*筑波大学システム情報工学研究科  
+筑波大学電子・情報工学系  
#豊橋技術科学大学知能情報工学系



HPCS Lab.

High Performance Computing System Lab., Univ. of Tsukuba

## 発表の手順

- 背景
- OmniRPC
  - グリッドでの並列プログラミングのためのミドルウェア
  - 特徴・機能
- CONFLEX-G
  - 分子立体配座探索プログラムCONFLEXの概要
  - グリッド並列化
  - 性能評価実験
    - CONFLEX-G VS. CONFLEX with MPI
    - CONFLEX-G in Grid Environments
  - 考察
- おわりに



JpGrid Workshop in Osaka

HPCS Lab.

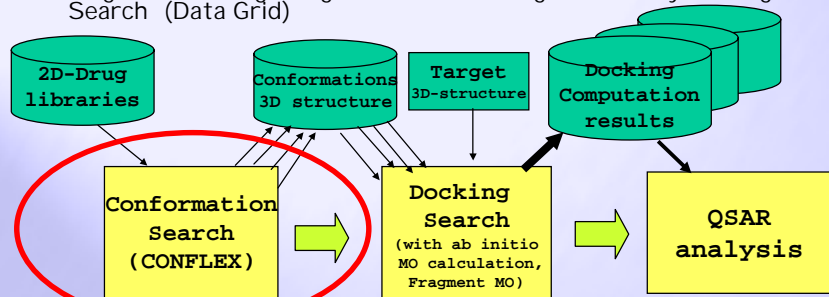
2003/7/24

2

High Performance Computing System Lab., Univ. of Tsukuba

## グリッドを用いた創薬プラットフォームの構築 (JST-ACT)

- 分子の配座探索
  - 分子の3次元構造をさがす
  - 2D drug database → 3D 構造 database
- Docking search: compute energy of combinations of molecules
  - Static jobs (dispatching by script), ... Dynamic job generation will be required
- Quantitative Structure-Activity Relationships (QSAR) analysis: finding rules of drug design from data-base generated by docking and conf. Search (Data Grid)



2003/7/24

JpGrid Workshop in Osaka

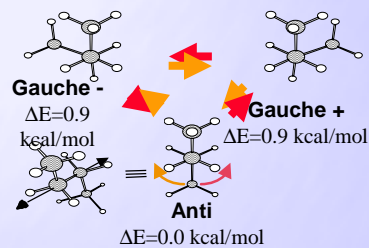
3

High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.

## CONFLEX

- 創薬のために配座を計算することが必要
  - 活性部位が3次元構造であるため、分子の3次元構造が必要
- 同じ分子でもエネルギー状態が異なる3次元構造をたくさん持つ
  - 大量の構造を高速に配座探索する必要性
- 今回はCONFLEXをグリッドで実行できるようにした



2003/7/24

JpGrid Workshop in Osaka

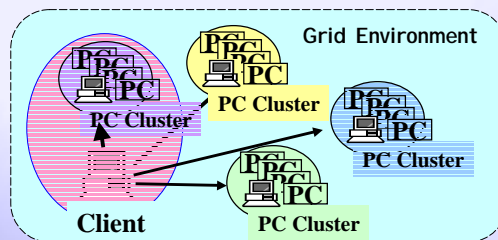
4

High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.

## OmniRPCの概要

- ローカルなクラスタからグリッドでのマルチクラスタまでシームレスなプログラミング環境を提供
  - ローカルな環境でプログラム開発、グリッドで大規模実行
- NinfのAPIをベースにしたthread-safeなRPC
- RPCクライアントが付加分散機能を持つ
  - 実行するホストはシステムに任せることができる
- ユーザごとのプライベートな環境の設定



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

5

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPCの特徴

- 主にMaster/Workerの並列プログラムを対象
  - パラメータサーチアプリケーションを効率よく処理できるようにサポート
- 簡便な並列プログラムインタフェース
  - 既存の(逐次)プログラムを簡単にGridの並列プログラムに移行できる
  - 非同期呼び出しを使い並列化
  - OpenMPを使っても並列化可能
- Gridのend-pointの計算環境としてクラスタを対象
  - グリッド環境では認証としてGlobusのほかにsshも可能
  - プライベートアドレスのクラスタについてもサポート
- 1000worker規模までの大規模な分散環境への対応



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

6

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPCの特徴(cont.)

- 簡便な環境設定: **プライベートなGrid計算環境構築のサポート**
  - 使える計算リソースの指定
  - リモート実行プログラムもユーザごとに管理
  - サーバーレス
- **認証, セキュリティに配慮**
  - 特定のポートを使用しない
- **動的な環境への対応**
  - 負荷の変動に対応
  - 障害ノードへの対応(現在実装中)
  - 実行中にノードを増やしたり、減らしたいする機能



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

7

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPCを使ったプログラミング

- **$B[i]=A * C[i]$ をRPCを使い並列に処理**

```
int main(int argc, char **argv){
    int i, A[100][100], B[100][100][100], C[100][100][100];
    OmniRpcRequest reqs[100];
    OmniRpcInit(&argc, &argv);

    /* set matrix A in worker modules*/
    for(i = 0; i < 100; i++)
        reqs[i] = OmniRpcCallAsync("mul", B[i], A, C[i]);
    OmniRpcWaitAll(100, reqs);

    OmniRpcFinalize();
    return 0;
}
```

非同期でRPC呼び出しで並列化

mulは $B[i] = A * C[i]$ の結果を返すリモート関数

RPC呼び出しの終了を待つ



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

8

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPCを使ったプログラミング(cont.)

- $B[i]=A * C[i]$ をRPCを使い並列に処理  
(モジュール初期化機能を使用した例)

```
int main(int argc, char **argv){
    int i, A[100][100], B[100][100][100];
    OmniRpcRequest reqs[100];
    OmniRpcInit(&argc, &argv);
    /* set matrix A in worker module
    OmniRpcModuleInit("matrix", A)
    for(i = 0; i < 100; i++)
        reqs[i] = OmniRpcCallAsync("mul2", B[i], C[i]);
    OmniRpcWaitAll(100, reqs);

    OmniRpcFinalize();
    return 0;
}
```

モジュール初期化機能を使用し  
Matrixモジュールにある行列Aの値  
をセット

ワーカーノードに初めてRPC呼び出し  
が行われたときに自動的にモジュールの  
初期化が行われる

非同期でRPC呼び出しで並列化

mul2は $B[i] = A * C[i]$ の  
結果を返すリモート関数

RPC呼び出しの終了を待つ



JpGrid Workshop in Osaka

HPCS Lab.

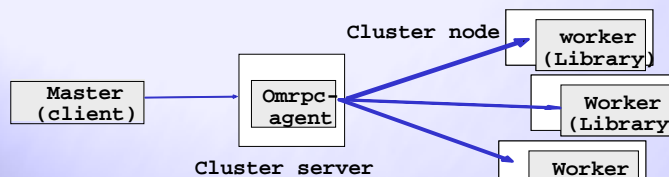
2003/7/24

9

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPCの動作

- ① クライアントプログラム起動
- ② リモートホストにssh, globusを使ってomrpc-agentを起動
- ③ omrpc-agentがレジストリ情報を取得
- ④ クライアントプログラムがRPC呼び出しを行うごとに  
omrpc-agentがリモート計算ノードにワーカーを起動し  
RPC呼び出しを行う



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

10

High Performance Computing System Lab., Univ. of Tsukuba

## OmniRPC:環境設定

- 個々のユーザのレジストリ(\$HOME/.omrpc\_registry)に環境を設定
  - 実行のための設定ファイル(hosts.xml)
    - 実行環境を変えるときに、プログラムを変更しなくてよい
  - スタッフデータベース
    - ホストでの利用可能なリモート実行プログラムと関数のデータベース

```
<? xml version="1.0 ?>
```

```
<OmniRpcConfig>
```

```
<Host name="dennis.omni.hpcc.jp" >
```

```
<Agent invoker="globus"
```

```
mxio="on"/>
```

```
<JobScheduler type="rr"
```

```
maxjob="20"/>
```

```
</Host>
```

```
</OmniRpcConfig>
```

omrpc-agentを起動させるリモートホスト

認証方式

通信の多重化

ジョブスケジューラ

同時に動作させる最大スタッフ数



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

11

High Performance Computing System Lab., Univ. of Tsukuba

## CONFLEX

- 分子の配座空間を探索するプログラム
  - 対象とする化合物が取り得るすべての立体配座を自動的に発生させ、化学的に重要な配座異性体の最適化構造を探索
  - 2D drug database → 3D structure database
- Molecular Mechanicsを使い構造最適化を行う
  - 全原子に対応した計算が可能
- 配座発生と構造最適化の機能を合わせ持つ
  - 配座発生の際に使用するアルゴリズムにより優れたパフォーマンス
  - Reservoir-Filling(貯水池注水)アルゴリズム
    - 見つかった配座異性体の中から、エネルギーの低い物から順に初期構造を選択
    - 探索領域は、速やかにエネルギーの低い方へと向う
    - 最安定配座が見つければ、探索領域は徐々に高い方へ拡大
  - 計算中頻繁に生成される既に見つかっている構造・鏡像異性体・幾何異性体・鞍点構造・極度に不安定な構造などは、自動的に削除



JpGrid Workshop in Osaka

HPCS Lab.

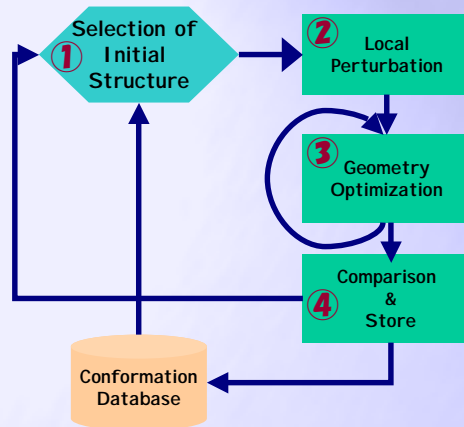
2003/7/24

12

High Performance Computing System Lab., Univ. of Tsukuba

## CONFLEXの配座探索アルゴリズム

- ① すでに保存されている構図の中から初期構造の選択
- ② 試行構造の生成
- ③ それぞれの構造を最適化
- ④ すでに得られている構造と比較し、新しい配座を保存



JpGrid Workshop in Osaka

HPCS Lab.

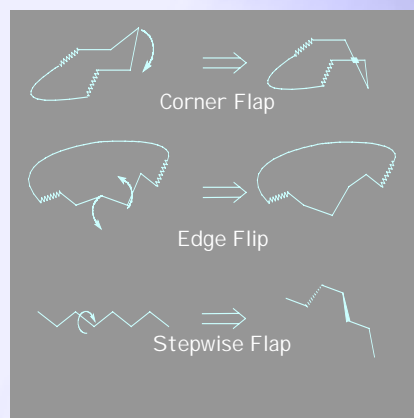
2003/7/24

13

High Performance Computing System Lab., Univ. of Tsukuba

## CONFLEXの構造生成

- Corner Flap
  - 初期構造の環の構成原子から一つを選び、環の平均平面に対して反対側に移動
- Edge Flip
  - 初期構造の環の構成原子から隣接する二つを選び、環の平均平面の反対側に移動、ねじる
  - 二つの原子を環の内側に動かし、へこませる
- Stepwise Rotation
  - 側鎖に対しては、単純な回転操作



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

14

High Performance Computing System Lab., Univ. of Tsukuba

## これまでのCONFLEXの問題点

- ・ 生体高分子をターゲットにするために試行構造の組み合わせの数が大きくなる
- ・ 1つの大きな立体構造を計算によって評価するためには、多大な時間を要する
- ・ 構造最適化の処理が全体の処理の90%以上を占める
- ・ これまではMPIを用いてPCクラスタで実行していた



巨大な計算機環境を使用できるGridで  
並列計算できるように改良



2003/7/24

JpGrid Workshop in Osaka

15

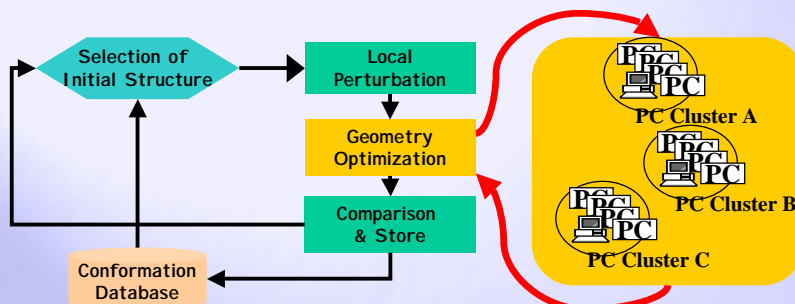
High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.



## CONFLEX-G:Grid enabled CONFLEX

- ・ 全体の処理の90%以上を占める構造最適化の処理をMaster/Workerパラダイムで並列化
- ・ 巨大な計算機資源を利用することができる
- ・ OmniRPCのモジュール再初期化機能を使用
  - 一度使った設定データを再利用
  - RPC呼び出しごとに初期化が不要



2003/7/24

JpGrid Workshop in Osaka

16

High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.



## Our Grid Platform for CONFLEX

Toyohashi Univ. of Tech.

### Toyo Cluster

Dual Athlon 1800+

8 nodes

Throughput: 0.55 MB/s

RTT: 13.0ms



Univ. of Tsukuba

### Dennis Cluster

Dual P4 Xeon 2.4GHz

10 nodes

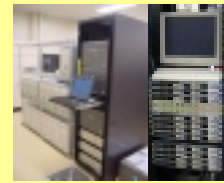
### Alice Cluster

Dual Athlon 1800+

14 nodes

Throughput: 11.22 MB/s

RTT: 0.18ms



Tokushima Univ.

### Toku Cluster

P3 1.0GHz

8 nodes

Throughput:

0.69MB/s

RTT:

24.4ms



AI ST

### UME Cluster

Dual P3 1.4GHz

32 nodes

Throughput: 2.12MB/s

RTT: 2.1ms

JpGrid Workshop in Osaka

Throughput, Round-trip  
time (RTT) はDennis Cluster  
間との値



2003/7/24

17

HPCS Lab. High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.

## クラスタの環境まとめ

Cluster	計算機の概要	ノード数	RTT* (ms)#	Throughput (MB/s)#
Dennis	Dual P4 Xeon 2.4GHz	10	-	-
Alice	Dual Athlon 1800+	14	0.18	11.22
Toyo	Dual Athlon 1800+	8	13.00	0.55
Toku	P3 1GHz	8	24.40	0.69
UME	Dual P3 1.4GHz	32	2.73	2.12

\* Round-Trip Time

# Dennis クラスタと各クラスタ間の値



2003/7/24

JpGrid Workshop in Osaka

18

HPCS Lab. High Performance Computing System Lab., Univ. of Tsukuba

HPCS Lab.

## 実験環境

- CONFLEXのバージョンは402q
- C17(51原子), AlaX08(91原子), AlaX16a(181原子)の分子をサンプルとして使用
- 認証にSSH
- 通信の多重化はなし
- クライアントプログラムはDennisのマスターノードから実行
- 基本的にクラスタにあるすべてのマシンを使用して計算



JpGrid Workshop in Osaka

HPCS Lab.

2003/7/24

19

High Performance Computing System Lab., Univ. of Tsukuba

## 使用したサンプル分子の概要

サンプル分子	一度に作成する試行構造数	1試行構造最適化にかかる平均時間* (s)	構造最適化を行った総試行構造数	Single CPUの計算機*で全試行構造最適化にかかる時間(s)
C17 (51 atoms)	48	1.6	522	835
AlaX08 (91 atoms)	80	20	800	16000 = 4.4 (h)
AlaX16a (181 atoms)	160	300	320	96000 = 26.7(h)

\*Dennisの1CPUでの場合



JpGrid Workshop in Osaka

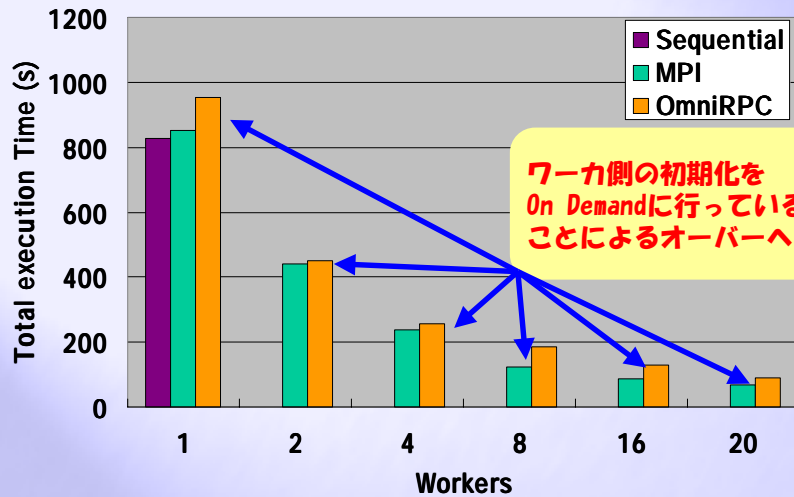
HPCS Lab.

2003/7/24

20

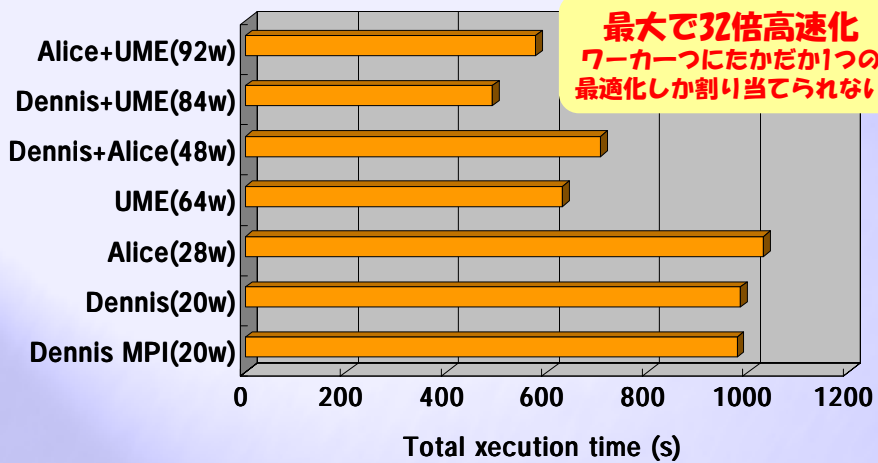
High Performance Computing System Lab., Univ. of Tsukuba

### Comparison between OmniRPC and MPI in Dennis Cluster with C17(51 atoms, 並列度48)



ワーカ側の初期化を On Demandに行っていることによるオーバーヘッド

### Execution time of Molecular Conformation with AlaX08(93 atoms, 並列度80)



最大で32倍高速化  
ワーカーつにたかだか1つの最適化しか割り当てられない

## Execution time of Molecular Conformation with AlaX16a(181 atoms, 並列度160)



2003/7/24

JpGrid Workshop in Osaka

23

HPCS Lab. High Performance Computing System Lab., Univ. of Tsukuba



## 考察

- OmniRPCはMPIと同程度の台数効果が得られている
  - OmniRPCではRPC呼び出しを行った時点でワーカーを初期化するのでそのオーバーヘッドがある
- 複数クラスタを使い高速化がはかれた
  - 最大で64倍の高速化が図れた  
(181 atoms molecular, total 112 Workers)
  - 高々ワーカー一つ当たり構造最適化が1or2しか割り当てられない
  - ジョブの処理時間のばらつきが大きい
- 分子の原子数が少ないときには並列化によって得られる台数効果が低い
  - 一回のループでの構造最適化処理の並列度が低いため
  - 一つの構造最適化にかかる時間が少なくなるため
    - ・ 構造最適化に20秒以上あるとよい台数効果がみこめる
  - 高分子をターゲットとしているためあまり問題ない(?)



2003/7/24

JpGrid Workshop in Osaka

24

HPCS Lab. High Performance Computing System Lab., Univ. of Tsukuba



## 考察 (cont.)

- **ワーカーを初期化するところが並列化されていないため効率よく初期化することができない**
  - ある程度の数いっせいにワーカーを初期化するAPIを考案中
- **もっと効率化するには**
  - **並列度を上げる**
    - 現在並列化を行っている外のループに対しても並列に実行するようにする。
  - **効率的なジョブスケジューリングを行う**
    - 重いジョブは速いマシンへ、軽いジョブは遅いマシンへ
      - 実行時間が予測可能か？
  - **Worker側でのスタブプログラムをSMPで(OpenMP)で並列化を行う**



## おわりに

- **OmniRPCを使って分子構造の配座探索を行うCONFLEXをグリッドで実行できるようにした**
- **広域のグリッド環境において高速化を図ることができた**
- **同じバイナリで、設定ファイルを変更することによって異なる環境に対応できる**
  
- **SSHは便利である**
- **Globusの設定はたいへん(Firewallとか)**



## ご静聴ありがとうございました

- OmniRPCのweb page

<http://www.omni.hpcc.jp/omnirpc/>



2003/7/24

JpGrid Workshop in Osaka

27

High Performance Computing System Lab., Univ. of Tsukuba

**HPCS Lab.**



2003/7/24

JpGrid Workshop in Osaka

28

High Performance Computing System Lab., Univ. of Tsukuba

**HPCS Lab.**

