



# REIN マニュアル

---

初版  
第 2 版

2012/11/06  
2013/02/04



# 目次

1 はじめに.....	1
動作環境.....	1
プログラム構成.....	2
ファイル構成.....	2
REIN の利用に際して.....	3
謝辞など.....	4
2 コンパイル手順.....	5
REIN プログラムのコンパイル.....	5
ツールのコンパイル.....	6
クリーン.....	7
3 REIN version 1.0 の仕様.....	8
REIN version 1.0 でできる REMD/REUS 計算.....	8
注意点.....	8
利用する MD プログラムに関して.....	9
4 レプリカ交換分子動力学法.....	10
動作.....	10
温度レプリカ交換分子動力学法.....	11
レプリカ交換アンブレラサンプリング法.....	11
5 チュートリアル.....	12
REIN の実行に必要となるファイル一覧.....	12
手順 1 : ファイルの準備.....	14
参考.....	14
手順 2 : インพุットビルダーの実行.....	15
手順 3 : バッチビルダーの実行.....	17
手順 4 : REIN の実行.....	19
手順 5 : REIN のリスタート実行.....	20
手順 6 : REIN のリスタート実行 (ステップ番号指定).....	21
手順 7 : トラジェクトリコンバータの実行.....	23
6 ファイルリファレンス.....	26
REIN インพุットファイル.....	26
Input File セクション.....	26
Output File セクション.....	27
MD Program セクション.....	27
Exchange Steps セクション.....	28
Replicas セクション.....	29
Machine Condition セクション.....	30
インพุットビルダーインพุットファイル.....	31
共通セクション.....	31
REIN セクション.....	32
MD セクション.....	34
バッチビルダーインพุットファイル.....	35
共通セクション.....	35
[BATCH]セクション.....	37
[REMD]セクション.....	39
トラジェクトリコンバータインพุットファイル.....	39
INPUT セクション.....	39

OUTPUT セクション .....	41
CONVERT セクション .....	42
SELECTION セクション .....	42
PBC セクション .....	43
レプリカ/エネルギートラジェクトリファイル.....	43
<b>7 テンプレートファイルリファレンス.....</b>	<b>45</b>
インプットビルダー用テンプレートファイル.....	45
バッチビルダー用テンプレートファイル.....	45
<b>8 REIN の設計思想と動作.....</b>	<b>46</b>
REIN のフローチャート .....	46
REIN の動作 .....	47

# 1 はじめに

Replica-exchange interface program (REIN)は、NAMD2 や MARBLE などの、既存の分子動力学(MD)ソフトウェアを用いて、多次元レプリカ交換分子動力学(REMD)シミュレーションを行うプログラムです。また、シミュレーション準備のためのツールおよび、シミュレーション後に得られるデータの変換ツールで構成されます。

## 動作環境

REIN のコンパイルおよび動作には以下の環境を推奨します。

想定するシステムは、Fujitsu コンパイラを備えた京コンピュータ、FX10(ステージング対応)共有ディスクドライブを持つ PC クラスタ等です。

アーキテクチャ	x86_64 SPARC64
オペレーティングシステム	CentOS 5 / 6 64bit 共有ディスクドライブを使用できる環境(ステージングにも対応)
MPI	Open MPI 1.4.4 以上 Fujitsu MPI
ジョブ管理システム	Univa Grid Engine / Sun Grid Engine Fujitsu ジョブ管理システム相当以上
コンパイラ	GNU コンパイラ 4.1 以上 / Intel Fortran Compiler 11 以上 Fujitsu Fortran
分子動力学(MD)ソフトウェア	NAMD 2.7 以上 / MARBLE 0.3.40 以上

## プログラム構成

REIN は以下のプログラムで構成されています。

プログラム名	概要
REIN (Replica exchange interface)	REIN プログラム本体です。 MD プログラムを用いた REMD 計算を実行します。
インプットビルダー input_builder	REIN のインプットデータを準備するためのツールです。
バッチビルダー batch_builder	REIN のジョブを投入する際に必要となるバッチスクリプトを生成するツールです。REIN をリスタート実行する際にも利用します。
トラジェクトリコンバータ remd_converter	REIN が出力したエネルギートラジェクトリおよび各 MD プログラムが出力した座標トラジェクトリ情報をコンバートし別形式のトラジェクトリファイルにまとめて出力するツールです。

## ファイル構成

REIN のファイル構成は以下のとおりです。

ディレクトリ	
rein/	
-- README.md	README ファイル
-- AUTHORS.md	開発者 ファイル
-- COPYING	GPL v3.0 ライセンス
-- notes.md	リリースノート + コンパイル方法
-- manual.pdf	マニュアル
-- rein_logo.pdf	REIN のロゴ
-- examples/	Example
-- src/	REIN プログラム本体のソースコード
-- lib/	ライブラリモジュール
-- libmarble/	MARBLE 用モジュール
-- libnamd/	NAMD 用モジュール
-- main/	メインモジュール
-- tools/	ツール
-- batch_builder/	バッチビルダー
-- converter/	トラジェクトリコンバータ
-- input_builder/	インプットビルダー
-- lib/	ツール共通ライブラリ
-- auto_submitter/	auto submission script

## REIN の利用に際して

---

REIN は別ファイルに詳細を記載しますが、GPL v.3.0 以上に準じるフリーソフトウェアです。プログラム内の README.md および COPYING を参照下さい。

REIN の利用は GPL v.3 に準じますが、利用後の成果発表に際しては以下の論文とダウンロードサイトを引用して下さい。

- Naoyuki Miyashita, Suyong Re, and Yuji Sugita, in preparation (2013)
- [http://www.islim.org/islim-dl\\_e.html](http://www.islim.org/islim-dl_e.html)

また、以下に REIN のロゴを記載します。



ロゴは、REIN の基幹となっている Watch Dog module を表す犬と、犬を制御する手綱 (REIN) で構成されています。REIN の文字は上下互い違いに隣の文字と結合しています。これはレプリカ交換を表しています。

## 謝辞など

---

REIN は、  
宮下尚之 (RIKEN CSRP, RIKEN QBiC, RIKEN AICS)と  
杉田有治 (RIKEN AICS, RIKEN, QBiC, RIKEN AICS)  
によって開発されました。

このプログラムを開発するにあたって、テスト計算やプログラムの改良などで助けていただいた、

高瀬規男 様 (磯子ソフト)

李秀栄 博士 (RIKEN ASI)

に感謝いたします。

テストを試して頂いた Pai-chi Li 博士 (RIKEN ASI)、Marble に関して教えて頂いた池口満徳 先生 (Yokohama city university) に感謝いたします。

その他お世話になりました、理化学研究所 杉田理論分子科学研究室、理化学研究所 生命システム研究センター 分子機能シミュレーション研究チーム、理化学研究所 計算科学研究機構 粒子系生物物理研究チーム、理化学研究所 分子スケール研究開発チームのメンバーの方々に感謝いたします。

本研究開発は、以下の支援を受けて行われたものです。

- 文科省 最先端・高性能汎用スーパーコンピュータの開発利用「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」
- 理化学研究所 生命システム研究センター
- 理化学研究所 計算科学研究機構
- JSPS 科研費 若手研究(B)24700299
- HPCI 「京」若手人材育成利用 課題番号 hp120173



## 2 コンパイル手順

REIN のコンパイル手順です。

### REIN プログラムのコンパイル

REIN プログラム本体のコンパイルは、ターミナルから以下のように入力してください。

```
(京コンピュータ, fx10 の例)
$ tar -ztf rein.tar.gz
$ cd rein
$ cd src
$ ./configure.sh sparc fujitsu    ← 構成の選択
$ make                          ← コンパイルの実行
    :
$ ls -l
Makefile
Makefile.comp
Makefile.lib
Makefile.machine
arch
configure.sh
lib
libmarble
libnamd
main
rein_sparc_fujitsu              ← 生成された REIN プログラム
$ make install                  ← REIN を rein/bin にインストールする
$ cd ../bin
$ ls
rein                            ← rein/bin にできた REIN の binary
```

上記例の京コンピュータ(K)以外では、`./configure.sh` の後に、〈プラットフォーム名〉、〈コンパイラ名〉を順番に指定します。

`src` ディレクトリ直下に以下のシンボリックリンクファイルが生成されます。

```
Makefile.comp
Makefile.machine
```

現状、〈プラットフォーム名〉、〈コンパイラ名〉は以下の組み合わせが可能です。

<i>Platform</i> \ <i>Compiler</i>	ifort	fujitsu
osx	○	×
sparc	×	○
x86_64	○	○

上記以外のコンパイル環境を別途追加する場合は、./src/arch ディレクトリ直下に、

```
Makefile.comp.<Platform>
Makefile.machine.<Platform>.<Compiler>
```

という2つのファイルを作成し、適宜編集をおこなってください。

K computer の場合には  
\$ ./configure.sh sparc k  
とすると、rein\_k という実行ファイルが作成されます。

\$ make install  
することで、rein のバイナリファイルは rein という名前で、rein/bin 以下に格納されます。

## ツールのコンパイル

REIN 付属のツールをコンパイルする手順です。

```
$ cd rein
$ cd tools
$ ./configure.sh x86_64 gfortran  ← 構成の選択
$ make                             ← コンパイルの実行
  :
$ make install                     ← rein/binにbinaryとtemplateがコ
                                   ピーされます。

$ cd ../bin
$ ls -l
batch_builder                    ← 生成されたバッチビルダー
input_builder                    ← 生成されたインプットビルダー
rein_convert                     ← 生成されたコンバータ
templates/                      ← ツールが使用するテンプレート
```

REIN プログラム本体と同様に、〈プラットフォーム名〉および〈コンパイラ名〉を指定して configure.sh を実行します。現状、以下の組み合わせが可能です。

<i>Platform</i> \ <i>Compiler</i>	gfortran	ifort
x86_64	○	○

K computer や FX10 ではフロントエンドで Linux が動いています。ツールはフロントエンドマシンで実行するプログラムですので、  
\$ ./configure.sh x86\_64 gfortran  
としてフロントエンド Linux 用の実行ファイルを作成して使用して下さい。

## クリーン

---

コンパイルしたファイル等の消去手順です。

\$ cd rein	
\$ cd src	
\$ make clean	← REIN プログラム本体、中間ファイルの消去
\$ make distclean	← bin から REIN のバイナリおよび、src 以下にある中間ファイルが全て消去
\$	
\$ cd ../tools	
\$ make clean	← ツールの中間ファイルの消去
\$ make distclean	← ツールの rein/bin からの消去
\$	

## 3 REIN version 1.0 の仕様

- REIN version 1.0 では以下に示すパラメータを用いたレプリカ交換分子動力学 (REMD) シミュレーションやレプリカ交換アンブレラサンプリング (REUS) シミュレーションができます。
- それぞれ自由に組み合わせて多次元レプリカ交換 MD (MREM) シミュレーションを行う事ができます。
- REIN のインプット自動作成ツール(input\_builder)とジョブ管理ソフトウェア用のバッチスクリプトを生成するツール(batch\_builder)があり、基本的な多次元 REMD 計算を行う際のインプットファイルを自動生成する事ができます。
- REIN のインプットファイルを直接修正する事で、複雑な REMD 計算にも対応します。
- **Version 1.0 では NAMD を子プログラムとして利用できます。K/FX10 用の NAMD が必要な場合は、 [yukimya+rein@gmail.com](mailto:yukimya+rein@gmail.com) までご連絡下さい。**

## REIN version 1.0 でできる REMD/REUS 計算

	NAMD2
温度	○
距離	○
距離(グループ)*	○
角度	○
角度(グループ)*	○
二面角	○
二面角(グループ)*	○

\*複数原子の選択ができます。複数原子の重心間の距離、角度、二面角を定義する事ができます。

## 注意点

1. 多次元レプリカ交換 MD シミュレーションを行う際、温度レプリカ交換を入れる場合には、温度軸を1次元目にして下さい。
2. REIN を用いた REMD シミュレーションを行う際には、事前に、使用する分子動力学 (MD) プログラムにて分子動力学計算を行い、使用する分子動力学プログラムのフォーマットの座標、速度、box、トポロジー情報など、通常の MD のリスタートの際に必要なインプットファイルを用意して下さい。
3. デフォルトでは10次元までの多次元レプリカ交換に対応しています。

4. 現 version では複数のパラメータを同一交換ステップ中に同時に交換する事はできません。
5. 実行の際には利用するコンピュータシステムの job scheduler の仕様を理解しておいて下さい。特に、hybrid 計算を行う際、job scheduler に依存します。
  - 京コンピュータ(K)や FX10 の場合、1つのノード内で2つの別の MD は流れません。(batch\_builder では自動的に job scheduler 用の batch file を作成します。)

## 利用する MD プログラムに関して

REIN version 1.0 では NAMD に対応しています。利用するプログラムは MPI でコンパイルしている必要があります。即ち、通常の MD 計算で、`mpiexec -n 8 namd2` などと記述する事で、動く状態であるという事です。MD プログラムの Source をダウンロードしてコンパイルをお願い致します。

NAMD:

<http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD>

---

x86\_64 アーキテクチャのマシンに限り、`mpi_comm_spawn` の機能を使わない計算ができます。この方式を用いると `Linux-x86_64-multicore` などのバイナリをそのまま利用できます。ただし、1つのレプリカで利用できる計算機のノード数は1ノード以下に制限されます。

nospawn オプションで REIN をコンパイルする場合

```
$ cd src
```

```
$ ./configure.sh x86_64 ifort nospawn もしくは RICC では ./configure.sh x86_64 ifort ricc
```

```
$ make; make install
```

また、`input_builder` や `batch_builder` を利用する場合には、それらのインプットの最初に `spawn = no` というオプションを入れて利用して下さい。利用の際には job scheduler の特徴を理解して利用して下さい。Grid Engine(GE)を用いる場合、通常はノード内のコア数を openMP のスレッド数と一致させて利用して下さい。

## 4レプリカ交換分子動力学法

レプリカ交換法(REM)は福島・根本によって 1992 年に提唱されました[1]。1999 年に杉田・岡本によって分子動力学法(MD)と組み合わせられ、レプリカ交換分子動力学法(REMD)[2]として発表されました。REMD は非常に簡便でかつ、効率良いサンプリング法の一つです。世界的にも非常によく使われる拡張アンサンブル法です。

2000 年に杉田・北尾・岡本によって、アンブレラサンプリング法と組み合わせたレプリカ交換アンブレラサンプリング法(REUS)と、温度 REMD と組み合わせた多次元レプリカ交換法(MREM)も開発されました[3]。

これらの方法を既存の MD プログラムで、既存の MD プログラムの改変無しに REMD/MREM シミュレーションを実現する為に、REIN は開発されました。

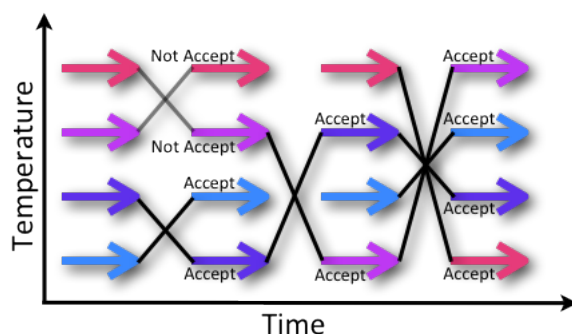
- [1] K. Hukushima and K. Nemoto J. Phys. Soc. Jpn. 65 (1996) 1604.
- [2] Y. Sugita and Y. Okamoto, Chem. Phys. Lett. 314, 141–151, 1999
- [3] Y. Sugita, A. Kitao and Y. Okamoto, J. Chem. Phys. 113, 6042–6051 (2000)

### 動作

レプリカ交換分子動力学法は、パラメータの違う複数の MD シミュレーションを同時に実行するプログラムです。あるステップだけ MD を実行した後に、メトロポリスクライテリアに従ってパラメータを交換します。これを必要回数繰り返します。

レプリカ交換の交換相手の選び方には様々な方法がありますが、REIN では、杉田・岡本のオリジナル REMD に従って、隣り合う値のパラメータを持つレプリカ間で交換評価を行います。

交換順序にも様々な方法があります。REIN のデフォルトでは杉田・岡本のオリジナル REMD と同様の交換手順になっています。例えば、4 レプリカの場合、 $i$  回目の交換ではパラメータ 1 (水色) – パラメータ 2 (青), replica 3 (紫) – replica 4 (赤)が交換し(交換 A)、 $i+1$  回目の交換では replica 2 (青) – replica 3 (紫)が交換します(交換 B)。これを繰り返します。



多次元の場合にも様々な方法があります。REIN のデフォルトでは以下の様な順に交換を行い、これを繰り返します。最初の数字が次元の番号を表しています。

1次元の場合: 1A, 1B

2次元の場合: 1A, 2A, 1B, 2B

3次元の場合: 1A, 2A, 3A, 1B, 2B, 3B

...

交換順は file\_ex.d で制御されており、この部分を変更する事で好みの交換を実現する事ができます。

以下、ハミルトニアンを以下の様に記述できるとします。

$$H_i = K(q_i) + V(q_i), \quad V(q_i) = E(q_i) + U(q_i) \quad .$$

$E(q_i)$ はポテンシャルエネルギー、 $U(q_i)$ は付加されるアンブレラポテンシャルです。

また、 $\beta_m = 1/(k_B T_m)$  .

ここで、 $i, j$  はレプリカ番号を表し、 $m$  はパラメータ番号。 $k_B$  はボルツマン定数、 $T_m$  は  $m$  番目のパラメータの温度。

## 温度レプリカ交換分子動力学法

温度交換を行う際、速度リスケールリングを行います。交換確率は以下です。

$$\min ( \exp(-\Delta_{\text{potential}}), 1 ) ,$$

$$\Delta_{\text{potential}} = ( \beta_m - \beta_n ) ( E(q_j) - E(q_i) ) .$$

## レプリカ交換アンブレラサンプリング法

将来的な拡張性の為に以下の交換確率を用いています。

$$\min ( \exp(-\Delta), 1 ) ,$$

$$\Delta = \Delta_{\text{potential}} + \Delta_{\text{harmonic}} \quad ,$$

$$\Delta_{\text{harmonic}} = \beta_m U_m(q_j) + \beta_n U_n(q_i) - \beta_m U_m(q_i) - \beta_n U_n(q_j) .$$

この version で想定しているアンブレラポテンシャルは二次形式  $1/2 k (x - x_0)^2$  です。

これらの交換法は `exchang.fpp` に記載されています。このモジュールを改変する事で様々なポテンシャルに対応する事ができます。

## 5 チュートリアル

REIN を使って実際にレプリカ交換分子動力学計算を実行します。  
シミュレーションの概要は以下のとおりです。例では京コンピュータ(K)を用います。

現状、K/FX10 では flat MPI の NAMD2 を用いて下さい。K/FX10 用の namd2 が必要であれば [yukimya+rein@gmail.com](mailto:yukimya+rein@gmail.com) にご連絡下さい。

1次元温度レプリカ交換分子動力学計算: (Flat MPI の NAMD を用います。)

・ジョブ管理システム	K
・プラットフォーム	フロントエンド: x86_64/計算ノード: sparco
・コンパイラ	フロントエンド: gfortran/計算ノード: fujitsu
・利用するデータ	REIN 付属のデータ examples/inputs
・次元数	1(温度)
・レプリカ数	8
・実行ステップ数	10(交換ステップ数)
・MD プログラム	NAMD
・MD プログラム MPI 並列数	8
・MD プログラム OpenMP 並列数	1

MD に、Hybrid 用の NAMD2.9 を利用する場合です。

REIN のインプットデータの準備は、REIN 付属のツールを利用します。これにより、最低限のデータを準備するだけで REIN 実行のための環境が整います。

## REIN の実行に必要なファイル一覧

はじめに REIN 実行時に必要となるファイルの一覧を以下に示します。  
ファイル名はツールを利用した場合のデフォルトの設定値であり、固定の名前ではありません。設定の変更については、「6 ファイルリファレンス - REIN インプットファイル p.26」を参照してください。



REIN NAMD 実行時		
1	file_axis_ex.d	軸交換リスト
2	file_coef.d	速度スケージングリスト
3	file_ex.d	交換パターンリスト
4	file_on.d	データ on/off リスト
5	file_rank.d	ランクリスト
6	file_value.d	パラメータ値リスト
7	initial.inp	NAMD インプットファイル
8	initial.colvar	NAMD colvar ファイル (拘束条件使用時)
9	initial.pdb	PDB ファイル
10	initial.psf	CHARMM PSF ファイル
11	namd2	NAMD プログラム
12	par_all27_prot_lipid.prm	CHARMM パラメータファイル
13	rein	REIN プログラム
14	rein.inp	REIN インプットファイル
15	rep. 1/	レプリカ 1 ディレクトリ
16	/rep. 1. 0. coor	NAMD restart coordinate ファイル
17	/rep. 1. 0. vel	NAMD restart velocity ファイル
18	/rep. 1. 0. xsc	NAMD restart extended system ファイル
19	rep. 2~rep. 8	レプリカ 2~8 ディレクトリ

## 手順1: ファイルの準備

必要なプログラムおよびデータを作業ディレクトリにコピーします。ここで作業ディレクトリの名前を test/ とします。

```
$ mkdir test ; cd test
$ cp -r <REIN_dir>/rein/bin/* .           ← REIN と tools のコピー
$ cp <NAMD_dir>/namd2 .                   ← NAMD のコピー
$ cp <REIN_dir>/rein/examples/inputs/* . ← インプットファイルの
                                           コピー
```

REIN プログラム、NAMD2、ツールおよびテンプレートディレクトリ(templates/)は、データと同じ位置に配置して下さい。

配置先を変更する場合の設定方法については「6.ファイルリファレンス - インプットビルダーインプットファイル - MD セクション p.34」を参照してください。尚、NAMD プログラムは REIN を実行する環境と同一の環境でコンパイルされている必要があります。

テンプレートディレクトリ(templates)は、通常はプログラムと同じパスに配置する必要があります。プログラムとは別のディレクトリに配置する場合は、ビルダープログラムにパスを設定する必要があります。詳細については、「6.ファイルリファレンス-インプットビルダーインプットファイル-共通セクション p.31」を参照してください。

## 参考

インプットファイルは、通常の MD シミュレーションに必要なインプットファイルと、MD を1度実行する事で得られるリスタートファイル群です。

実際に REIN を用いた多次元レプリカ交換シミュレーションを行う際には、事前に以下の準備を行って下さい。

1. 構造の準備 (PDB, もしくは CRD)
2. トポロジーファイルの作成 (psfgen, molx など)
3. Minimization (MD パッケージで可能)
4. REIN で使用する MD プログラムで、平衡化計算 (この際、リスタートファイルを出力)。

初期構造ファイル名は initial.pdb、トポロジーファイル名は initial.psf、リスタートファイル名は initial.XXX という形式にすると簡便に REMD 計算を進める事ができます。

また REIN では、プログラムをレプリカの数だけ実行するが交換をしない(アンブレラサンプリング)、というオプションがあります。rein.inp ファイルの equiv\_steps で制御されています。

REIN を用いて複数のレプリカを同時に平衡化する場合には

```
equiv_steps = 1
```

product\_steps = 0

とし、更に、initial.inp ファイル中の MD のステップ数を決める set finalstep の値を必要なステップ分取る事で、並列でレプリカの平衡化計算を行う事ができます。

インプットビルダーとバッチビルダーのデフォルト値は、これらのプログラムを順に両方実行する事を前提に設定されています。(単体での利用はそれぞれのオプションを参照して下さい。)

## 手順2: インプットビルダーの実行

インプットビルダーは、REIN の実行に必要な以下のファイルを生成します。

- ・レプリカ毎の初期構造、初期速度データ (rep.\*/\* ディレクトリ以下のファイル)
- ・レプリカ交換パターン、交換ルール、ランクリスト等のファイル (file\_\*.d ファイル)
- ・REIN インプットファイル

インプットビルダーを実行するにあたり、各 MD プログラムで以下の入力データが必要となります。作業ディレクトリには既にNAMD用の必要なファイルが配置されています。

NAMD	
1 PDB ファイル	initial.pdb
2 CHARMM PSF ファイル	initial.psf
3 NAMD coordinate restart ファイル	initial.coor
4 NAMD velocity restart ファイル	initial.vel
5 NAMD extended system ファイル	initial.xsc
6 CHARMM parameter ファイル	par_all27_prot_lipid.prm
7 インプットビルダーインプットファイル	-

インプットビルダー用のインプットファイルのテンプレートを以下の手順で生成します。

```
$ cd test
$ ./input_builder -h ctrl > ib.inp
```

テキストエディタ等を使って、生成された ib.inp ファイルの設定を変更します。(この例では設定変更の必要はありません。)

```
# control parameters in input_builder

rein_input          = rein.inp      # rein input file の名前

[REMD] #レプリカ交換に関するオプション
```

```

# total のレプリカ数は num_replicas001 x num_replicas002 x ...となる。

axis001                = temperature    # axis (TEMPERATURE/HARMONIC)
num_replicas001        = 8              # 1 番目の軸に対するレプリカ数
range001               = 300.0 304.0    # 温度の下限上限
division001            = exp            # (EQUIV/EXP/CYCLE) 分割ルール
    # equiv: 等間隔, exp: 指数間隔, cycle: 角度など等間隔で元に戻るもの

# 例は 1 次元なので、以下はコメントアウト
# 多次元 REMD を行う際にはコメントをはずす。
# axis002                = harmonic      # 2 次元目の指定
# num_replicas002        = 8            # 2 番目の軸に対するレプリカ数
# range002               = 2.0 8.0      # 軸に対する下限上限値
# division002            = equiv        # 分割ルール (等間隔)
# 軸の設定: NAMD の atom selection に従う
# distance002            = ADP CR 1-1 ADP CL 1-1 # 距離指定の場合
# angle002               =              # 角度指定
# dihedral002            = ADP NL 1-1 ADP CA 1-1 ADP CRP 1-1 ADP NR 1-1
#                               二面角指定
# spring_const002        = 2.0          # バネ定数

[MD] #それぞれのレプリカの MD に対するオプション

md_program              = namd          # md program (NAMD/MARBLE)
md_exe_file             = ./namd2       # md exe file

md_temperature          = 300.0         # md temperature (MD の初期温度)
md_steps                = 1000          # md steps / exchange
#一回のレプリカ交換で行う MD のステップ数

#システムサイズを入れると、PME のメッシュが 2, 3, 5 の倍数になる様に自動的に
#計算します。
system_size_x           = 23.4 # system size --automatically create pme grids
system_size_y           = 23.2 #
system_size_z           = 23.9 #

# pme_grid               = yes          # if you want to setup pme grids, (YES/[NO])
#直接 PME グリッド数を入れる場合に使います。
# pme_grid_size_x        = 24           # pme grid size x, y, z
# pme_grid_size_y        = 24           #
# pme_grid_size_z        = 24           #

```

次に、コマンドライン引数に ib.inp を与えてインプットビルダーを実行します。

```

$ ./input_builder ib.inp
:

```

```

Check_Files> check files for : namd
    rein                               : found.
    namd2                             : found.
    par_all27_prot_lipid.prm          : found.
    initial.pdb                       : found.
    initial.psf                       : found.
    initial.restart.coor              : found.
    initial.restart.xsc               : found.
    initial.restart.vel               : found.

Build_File_D> Build file_*.d files.

Build_Reininp> Build REIN input file.

Build_Mdinp> Build MD input files. type : namd

Copy_Files> copy files for : namd

```

実行に必要なファイル  
のチェック

エラーメッセージが表示されることなく処理が完了すると、  
以下のファイルが作成されます。

REIN	メインのインプット	rein.inp
	交換情報など	file_axis_ex.d, file_ex.d, file_on.d, file_value.d
	スタート時に必要な情報	file_rank.d, file_coef.d
MD	インプットのテンプレート	initial.inp
		initial.colvar (多次元の場合)

「REIN の実行に必要なファイル一覧 p.12」のファイルがすべて揃ったことになります。

file\_\*.d を作成し直す際には、  
\$ rm file\_\*.d  
を実行してから再度 input\_builder を実行して下さい。  
\$ ./input\_builder ib.inp

細かい設定も可能で input\_builder 単体利用も可能です。オプションの情報を参照して下さい。

## 手順3: バッチビルダーの実行

バッチビルダーは、REIN をジョブ管理システムへ投入するためのバッチスクリプトを生成します。

バッチビルダー用のインプットファイルを以下の手順で生成します。

```
$ ./batch_builder -h ctrl > bb.inp
```

生成された bb.inp ファイルの設定を以下のとおり(赤色)変更します。

```

# control parameters in batch_builder

rein_input          = rein.inp          # REIN input file name

# chain_job          = yes    # (YES/[NO]) It is for chain_job. For k: (NO)

[BATCH]
batchfile_name      = rein.sh           # batch filename
job_scheduler       = K                 # job scheduler type ([K]/GE/RICC)
queue_name          = small            # queue name, k: ([SMALL]/LARGE), fx10: REGULAR..

num_threads_md      = 1                 # number of threads in md
num_mpiproc_md      = 8                 # # of mpi process for md
cpu_time            = 01:00:00          # cpu time
num_core_cpu        = 8                 # number of core in cpu, for k = 8, for fx10 = 16, ...

[REMD]
replica_exchange    = yes # ([YES]/NO): replica exchange / umbrella sampling
num_of_exchange_steps = 10              # number of exchange steps

```

queue\_name は、ジョブ投入のための利用可能なキュー名を指定してください。今回のチュートリアルでは、全体で、33 ノード利用 (REIN プログラム1プロセス + NAMD プログラム 8 プロセス × 8 レプリカ) となるので small です。

K (FX10)の場合の利用ノード数は以下の計算で決まります。

K(FX10): 全ノード数 = REIN のプロセス数(自動生成であれば常に 1)  
+ MD のプロセス数(num\_mpiproc\_md) X 全レプリカ数

また、スケジューラが GE の場合には、実行時に全コア数を考慮する必要がありますがその場合の利用コア数は以下の計算となります。また通常、PC クラスタでは自動生成では flatMPI 構成となります。

(GE): 全コア数 = REIN のプロセス数(自動生成であれば常に1)  
+ MD のスレッド数(num\_threads\_md) X MD のプロセス数(num\_mpiproc\_md)  
X 全レプリカ数

次に、コマンドライン引数に bb.inp を与えてバッチビルダーを実行します。

```

$ ./batch_builder bb.inp
*****
*                                                                    *
*   BATCH_BUILDER: build batch script for REIN                       *
*                                                                    *
*****
:
output_replica_energy = replica_energy.trj

```

```
        program name = namd
        initial file name = initial
        I/O file name = rep.
# of total replicas = 8
        first step = 1
        equiv steps = 0
        product steps = 10

[STEP4] build batch script file

Batch> Batch queue file: ./rein.sh

$ ls rein.sh
rein.sh
$
```

エラーメッセージが表示されることなく処理が完了すると、バッチスクリプト rein.sh が生成され、rein.inp にも必要な情報が書き込まれます。

これで REIN の実行準備が整いました。

## 手順4: REIN の実行

以下の操作で REIN のジョブをキューに投入します。

```
$ pjsub rein.sh
[INFO] PJM 0000 pjsub Job 1481605 submitted.
$
```

REIN のジョブ終了後、STAGE\_OUT/ディレクトリ以下に以下のファイルが得られます。K や FX10 ではそれぞれのレプリカは tar されます。

注) PC クラスタ(GE)では STAGE\_OUT/ディレクトリは作成されず、rep.1/や rep.2/に直接データが記述されます。

出力ファイル		
1	file_out_coef. d	実行後の速度スケージングリスト
2	file_out_rank. d	実行後のランクリスト
3	replica. trj	エネルギー/ランクトラジェクトリファイル
4	rep. 1. tar	レプリカ 1 ディレクトリの tar ファイル
5	rep. 2. tar~rep. 8. tar	レプリカ 2~8 ディレクトリの tar

Current ディレクトリには以下のファイルが生成されます。

出力ファイル		
5	rein. sh. o1481628	ジョブ標準出力ファイル

6	rein. sh. e1481628	ジョブ標準エラー出力ファイル
7	rein. sh. i1481628	情報出力ファイル
8	rein. sh. s1481628	ステージングエラー出力ファイル

## 手順5: REIN のリスタート実行

前回の続きからシミュレーションを同じレプリカ交換ステップ数だけ再開する手順です。

```
$ ./batch_builder bb.inp restart          ← “restart”
*****                                を指定
*
*   BATCH_BUILDER: build batch script for REIN   *
*
*****

[STEP1] Read Control Parameters for batch builder
:
:
      I/O file name = rep.
# of total replicas = 8
      first step = 11
      equiv steps = 0
      product steps = 10

[STEP4] build batch script file

Batch> Batch queue file: ./rein.sh

$
```

この操作により以下のファイルが更新され、11 ステップ目からのシミュレーションの準備が整います。

更新されるファイル	
1	file_coef.d                      速度スケーリングリスト
2	file_rank.d                      ランクリスト
3	rein.inp                          REIN インプットファイル
4	rein.sh                          バッチスクリプト

また、前回のシミュレーションの出力ファイル（STAGE\_OUT/ディレクトリ内のファイル）が、results ディレクトリに格納されます。この際、tar の展開が行われます。results ディレクトリ直下に、開始と終了ステップ番号が割り振られたディレクトリが作成され、そこにファイルがコピー（tar の展開）されます。

京コンピュータと FX10 ではファイルの転送が1 ノードあたり 800 ファイル X ノード数 + 1000 ファイルと決まっている為に、tar をかけて転送する必要があります



ます。

```
$ ls results
result0000001_0000010
$
$ ls -l results/result0000001_0000010
file_axis_ex.d
file_coef.d
:
rein.inp
rep.1/
rep.2/
:
replica.trj
$
```

← 格納された  
前回の計算結果

作業ディレクトリ直下のレプリカディレクトリ(rep.1~8)以下は、リスタート実行の際に必要なステップ番号 10 番のファイルのみが格納されています。

```
$ ls -l rep.1
rep.1.10.inp
rep.1.10.out
rep.1.10.coor
rep.1.10.dcd
rep.1.10.vel
rep.1.10.xsc
rep.1.10.xst
rep.1.10.sh
$
```

再度 REIN ジョブをキューに登録します。

```
$ pjsub rein.sh
[INFO] PJM 0000 pjsub Job 1481740 submitted.
$
```

## 手順6: REIN のリスタート実行(ステップ番号指定)

REIN ステップ番号を指定して、そのステップからリスタート計算をおこなう手順です。これはしばしばシステムの問題で、途中でreinが落ちた時や、計算をやり直したい時に、簡便に計算のやり直しをする為の機能です。

```

$ ./batch_builder bb.inp restart 11
*****
*                                     *
*   BATCH_BUILDER: build batch script for REIN   *
*                                     *
*****

[STEP1] Read Control Parameters for batch builder
      :
      :
      I/O file name = rep.
# of total replicas = 8
      first step = 11
      equiv steps = 0
      product steps = 10

[STEP4] build batch script file

Batch> Batch queue file: ./REIN_que.GE.rein.inp.sh

$

```

← ステップ番号  
付で“restart”  
を指定

この操作により以下のファイルが results ディレクトリから復元され、11 ステップ目からのシミュレーション準備が整います。

更新されるファイル		
1	file_coef.d	速度スケージングリスト
2	file_rank.d	ランクリスト
3	rein.inp	REIN インプットファイル
4	rein.sh	バッチスクリプト
5	/rep.1.1~10.coor	NAMD coordinate ファイル
6	/rep.1.1~10.vel	NAMD velocity ファイル
7	/rep.1.1~10.xsc	NAMD extended system ファイル
8	rep.2~rep.8	レプリカ 2~8 ディレクトリ

results ディレクトリ以下に、以下の出力データが格納されている場合に指定可能な REIN ステップ番号は、1, 11, 21, 31, 41 です。それ以外の番号を指定した場合は、エラーとなります。

```

$ ls -l results
result0000001_0000010
result0000011_0000020
result0000021_0000030
result0000031_0000040
result0000041_0000050
$
$ ./batch_builder bb.inp restart 33
*****
*                                     *

```

```

*   BATCH_BUILDER: build batch script for REIN   *
*                                                    *
*****
:
        product steps = 10

ERROR: Cannot restart with REIN step number : 33
Restart_num> Batch-file restart was failed.

$

```

## 手順7:トラジェクトリコンバータの実行

REIN がステップ毎に断片的に出力するトラジェクトリファイルを、レプリカ番号別または条件別に1つのファイルにまとめ、別形式のトラジェクトリフォーマットに変換して出力します。

本プログラムでの解析では results/内のファイルを使いますので、必ず、解析の前に  
`$ ./batch_builder ib.inp restart`  
と restart コマンドを実行しておいて下さい。

変換対象となるデータは以下のとおりです。

変換対象	
エネルギートラジェクトリファイル	./replica.trj
座標トラジェクトリファイル	./rep.1~8/rep.1~8.1~10.dcd

まずはインプットファイルを作成します。

```
$ ./rein_convert -h ctrl > rc.inp
```

テキストエディタでファイル INPCNV を以下のように編集します。  
この例では、前節のリスタート実行を1度おこなっていると仮定しています。

```

# control parameters in remd_convert

[INPUT]

psffile          = initial.psf
reffile          = initial.pdb
trj_format       = DCD          # (DCD/MARBLE/PDB)

```

```

directory001      = ./results/result0000001_0000010
directory002      = ./results/result0000011_0000020
directory003      = ./results/result0000021_0000030

repeat001         = 2
trj_filename001   = rep. {rep_no}/rep. {rep_no}. {step}.dcd
energy_trj001     = replica.trj

energy_column     = # column of output energy, ex. = 1 2-4, empty: all
column

[OUTPUT]

out_pdbfile       = out.pdb
trj_filename      = out. {type}. {number}. {extension}
trj_format        = DCD      # (DCD/MARBLE/AMBER/TINKER/PDB/GROMACS)

[CONVERT]

# replica 毎の trajectory を作成する場合には REPLIC
# condition 毎の trajectory を作成する場合には CONDITIO
# と記載して下さい。
type              = REPLIC      # (REPLIC/CONDITIO)

rank              =          # ex. = 1 2-8, empty: output all rank

[SELECTION]

# select_atom     = all
# select_atom     = molname:protein | molname:lipid
# mole_name001    = protein  P1:1:TYR P1:5:MET
# mole_name002    = lipid    OLEO:PCGL:OLEO

```

#### [INPUT]セクション

入力ファイルの指定をします。

#### [OUTPUT]セクション

出力ファイルの設定をします。

#### [CONVERT]セクション

type の設定を“REPLIC”とすることで、レプリカ番号別にファイルをまとめて出力します。

また、[CONVERT]セクション list を空白とし、すべてのレプリカ番号をコンバートの対象とし出力するように指示します。

トラジェクトリコンバータを実行します。

```

$ ./rein_convert rc.inp
*****
*                               *
*   REIN_CONVERT: convert REIN trajectory format   *
*                               *
*****

[STEP1] Read Control Parameters for conversion
      :
tid:  0  dat:  8  dir:  2  fs:  1  ls:  5
tid:  0  dat:  8  dir:  3  fs:  1  ls:  5
tid:  0  dat:  8  dir:  4  fs:  1  ls:  5
tid:  0  dat:  8  dir:  5  fs:  1  ls:  1

[STEP4] Deallocate memory

$

```

← 実行

実行後、以下のファイルが得られます。

出力ファイル		
1	out.pdb	リファレンス座標
2	trj.replica.1~8.crdtrj	同一レプリカ番号で結合された座標トラジェクトリファイル
3	trj.replica.1~8.enetrj	同一レプリカ番号で結合されたエネルギートラジェクトリファイル

3 つの supporting software に関して

\$ XXX\_XXX -h ctrl all

を実行することで、全ての option を見る事ができます。

## 6 ファイルリファレンス

REIN プログラムおよびツールの入出力ファイルの解説です。

### REIN インプットファイル

REIN プログラムのメインのインプットファイルです。

プログラム	入力／出力
REIN プログラム	入力
インプットビルダー	出力
バッチビルダー	-
トラジェクトリコンバータ	-

使用例:

```
$ ./rein rein.inp          ← インプットファイルを指定して REIN を起動
```

### Input File セクション

REIN の実行に必要なインプットファイルパスを設定します。

#### input\_replica\_rank

入力するランクリストファイル名を指定します。

#### input\_replica\_values

入力するパラメータ値リストファイルを指定します。

#### input\_replica\_on

入力するデータ on/off リストファイル名を指定します。

#### input\_exchange\_pattern

入力するデータ on/off リストファイル名を指定します。

#### input\_axis\_exchange

入力する交換パターンリストファイル名を指定します。

### input\_restart\_tscale

入力する速度スケールリングリストファイル名を指定します。

## Output File セクション

---

### output\_restart\_rank

出力するランクリストファイル名を指定します。このリストは、最後のステップ実行後に得られるランクの状態です。

### output\_restart\_tscale

出力する速度スケールリングリストファイル名を指定します。このリストは、最後のステップ実行後に得られる速度スケールリングの状態です。

### output\_replica\_rank

出力するレプリカランク/エネルギートラジェクトリファイルを指定します。

## MD Program セクション

---

MD プログラムに関する設定をおこないます。

### program\_name

MD プログラム名を“namd”または“marble”の何れかで指定します。

デフォルト値: “namd”

### parallel\_exec

MD プログラムの実行ファイル名を指定します。

デフォルト値: “./namd2”

### velocity\_rescaling

速度リスケールリングの方法を以下の何れかで指定します。

デフォルト値: “v\_namd\_bin”

v_namd_bin	NAMD binary velocity ファイル
v_namd	NAMD ascii velocity ファイル
v_marble	MARBLE velocity ファイル

### **base\_temperature**

初期温度を設定します。温度レプリカ交換の場合、使用しません。

デフォルト値: 300K

### **parallel\_module\_md**

nonspawn オプションで mpi 以外の並列化コンパイルを用いる時ここに記載する。例えば “charmrun” など。

デフォルト値: “”

### **option\_md**

MD に必要なオプションを記載する。例えば “+ppn 7” など。

デフォルト値: “”

### **extend\_option\_md**

インプットの後に追加するオプションを記載する。

デフォルト値: “”

## **Exchange Steps セクション**

---

### **first\_step**

REIN 実行開始時のステップ番号を指定します。各レプリカのディレクトリ(rep.\*)にはそれより一つ前のステップ番号が付与された座標、速度、セル情報のファイルが格納されている必要があります。

デフォルト値: 1

### **equiv\_steps**

MD プログラムにて平衡化計算を実施する、REIN ステップ数を指定します。このステップ実行中は、レプリカ交換をおこないません。

デフォルト値: 0

### **product\_steps**

レプリカ交換を実施する REIN ステップ数を指定します。

デフォルト値: 10



## Replicas セクション

---

### initial\_file\_name

MD プログラムの初期入力ファイルに付加するプレフィックスを指定します。

デフォルト値: “initial”

### io\_file\_name

各レプリカのディレクトリ名に付加するプレフィックスを指定します。

デフォルト値: “rep.”

### number\_of\_axis

軸の数(次元数)を指定します。

デフォルト値: 1

### number\_of\_exchange

レプリカ交換の交換数を指定します。軸の数が 1 の場合、この値は 2 以上に設定する必要があります。

デフォルト値: 2

### axis\_index

各軸のインデックス番号を指定します。この設定は、number\_of\_axis の軸数分だけ記述する必要があります。このセクション中の以降の設定もすべて、軸数分だけ記述しなければなりません。

### axis\_type\_name

各軸のタイプ名を記述します。温度レプリカ交換の場合“temperature”、アンブレラポテンシャルのレプリカ交換の場合“harmonic”です。

### number\_of\_replicas

各軸のレプリカ数を指定します。

### axis\_control\_param

各軸の制御パラメータ番号を指定します。この値は、各ランクに割り振る制御パラメータ値を決めるための、[Input File]セクション input\_replica\_values で指定するファイル内でのカラム番号です。

### axis\_num\_of\_param

各軸のパラメータ数を指定します。ここで指定した数だけ後述の axis\_param を記述しなければなりません。

### axis\_param

各軸のパラメータ番号を指定します。この値は、各ランクに割り振るパラメータ値を決めるための、[Input File]セクション input\_replica\_values で指定するファイル内でのコラム番号です。

### periodic

軸が、dihedralなど周期的なパラメータの場合は“on”を指定します。それ以外は、“off”を指定します。

### periodic\_value\_min

周期的なパラメータの最小値です。

### periodic\_value\_max

周期的なパラメータの最大値です。

例： 0～180 を 4 レプリカにする場合

0    90    180    270

```
periodic          = "on"
periodic_value_min = 0
periodic_value_max = 270
```

## Machine Condition セクション

---

### sleep\_time

MDプログラムのファイル出力が完了したかどうか、ポーリング処理を行う際のスリープ時間を秒単位で指定します。完了状態になるまで、スレッドは繰り返しこの間隔でスリープします。

デフォルト値： 1

### number\_of\_cpus\_for\_exec

MDプログラムの MPI 並列数を指定します。

デフォルト値： 1

## インプットビルダーインプットファイル

プログラム	入力／出力
REIN プログラム	-
インプットビルダー	入力、出力 (雛形)
バッチビルダー	-
トラジェクトリコンバータ	-

使用例:

```
$ ./input_builder -h ctrl > INPIB      ← 雛形の作成
:
(編集)
:
$ ./input_builder INPIB                ← インプットビルダーの実行
```

## 共通セクション

### directory

出力先ディレクトリを指定します。  
このディレクトリ直下に、REIN の実行に必要なすべてのファイルが作成されます。

デフォルト値: “.” (カレントディレクトリ)

### rein\_input

REIN インプットファイルのファイル名を指定します。

デフォルト値: “rein.inp”

### template\_dir

インプットビルダーが使用するテンプレートファイルディレクトリへのパスを指定します。  
この設定が空白の場合、“<インプットビルダーのプログラムパス>/templates”をテンプレートディレクトリとします。

デフォルト値: “”

### spawn

通常は yes です。non spawn で計算を行う際に “no” とします。MPI-2 の mpi\_comm\_spawn に対応していない shared disk が利用できない計算機システムの場合

に、“no”とします。例えば理研 RICC。“no”の場合、rein が nospawn オプションでコンパイルされている必要があります。

デフォルト値: “yes”

## REMD セクション

---

### first\_step

REIN 実行開始時のステップ番号を指定します。各レプリカのディレクトリ(rep.\*)にはそれより一つ前のステップ番号が付与された座標、速度、セル情報のファイルが格納されている必要があります。

デフォルト値: 1

### equiv\_steps

MD プログラムにて平衡化計算を実施する、REIN ステップ数を指定します。このステップ実行中は、レプリカ交換をおこないません。アンブレラサンプリング計算となります。

デフォルト値: 0

### product\_steps

レプリカ交換を実施する REIN のレプリカ交換ステップ数を指定します。

デフォルト値: 0

### axis001-

軸のタイプを“harmonic”または“temperature”で指定します。001-は通し番号で、001, 002, 003 の順番で与えます。例えば 002 は 2 番目の軸に関する設定です。

デフォルト値: “”

### num\_replicas001-

軸のレプリカ数を指定します。3 以上の値を指定しなければなりません。

デフォルト値: 0

### range001-

軸の取りうる値の範囲を最小値、最大値の順で指定します。

例: range001 = 2.0 10.0

デフォルト値: “”

#### division001-

軸の値の分布タイプを以下の何れかで指定します。

equiv	等間隔に分割
exp	指数で分割
cycle	等間隔で分割かつ 0 に戻る。角度など。

デフォルト値: “exp”

#### distance001-

距離拘束を与える 2 つの原子グループまたは、原子番号を指定します。  
axis001-に“harmonic”を指定した場合に有効です。また、dihedral001-、angle001-との併用はできません。

デフォルト値: “”

<例: NAMD の場合>

“distance001 = ADP:NL:1-1 ADP:CA:1-1”

値の左から、

<残基名 1>:<原子名 1>:<開始残基番号 1>-<終了残基番号 1>

<残基名 2>:<原子名 2>:<開始残基番号 2>-<終了残基番号 2>

NAMD 実行時は、“atomNameResidueRange”での原子グループに翻訳されます。

#### angle001-

角度拘束を与える 3 つの原子グループを指定します。  
axis001-に“harmonic”を指定した場合に有効です。また、distance001-、dihedral001-との併用はできません。別の次元にして軸番号を変えて下さい。

MARBLE では利用できません。

デフォルト値: “”

#### dihedral001-

2 面角拘束を与える 4 つの原子グループまたは、原子番号を指定します。  
axis001-に“harmonic”を指定した場合に有効です。また、distance001-、angle001-との併用はできません。

MARBLE では、原子番号指定のみ対応しています。その場合、MARBLE 実行時は、“torsion\_harmonic”の設定に翻訳されます。

デフォルト値: “”

#### **spring\_const001-**

axis001-に“harmonic”を指定した場合のバネ定数を指定します。

デフォルト値: 2.0

## **MD セクション**

---

#### **num\_mpiproc\_md**

MD プログラムの MPI 並列数(プロセス数)を指定します。

デフォルト値: 8

#### **md\_program**

MD タイプを“namd”または“marble”の何れかで指定します。

デフォルト値: “namd”

#### **md\_exe\_file**

MD プログラムの実行ファイル名を指定します。

デフォルト値: “./namd2”

#### **md\_temperature**

初期温度を指定します。温度レプリカ交換では使用しません。

デフォルト値: 300K

#### **md\_steps**

MD プログラムの MD ステップ数を指定します。このステップ数は、REIN の 1 ステップあたりのステップ数です。

デフォルト値: 1000

#### **system\_size\_x**

系の X 軸方向のサイズ(Å)を指定します。自動的にふさわしい pme\_grid\_size を計算します。

#### **system\_size\_y**

系の Y 軸方向のサイズ(Å)を指定します。自動的にふさわしい pme\_grid\_size を計算し

ます。

#### **system\_size\_z**

系の Z 軸方向のサイズ (Å) を指定します。自動的にふさわしい pme\_grid\_size を計算します。

#### **pme\_grid**

Perticle Mesh Ewald のグリッドサイズを指定するかどうか決めます。“yes”の場合、以下の pme\_grid\_size\_x の値を用います。

デフォルト値: “no”

#### **pme\_grid\_size\_x**

Perticle Mesh Ewald の X 軸方向のグリッドサイズを指定します。

#### **pme\_grid\_size\_y**

Perticle Mesh Ewald の Y 軸方向のグリッドサイズを指定します。

#### **pme\_grid\_size\_z**

Perticle Mesh Ewald の Z 軸方向のグリッドサイズを指定します。

## バッチビルダーインプットファイル

プログラム	入力／出力
REIN プログラム	－
インプットビルダー	－
バッチビルダー	入力、出力 (雛形)
トラジェクトリコンバータ	－

使用例:

```
$ ./batch_builder -h ctrl > INPBB      ← 雛形の作成
:
(編集)
:
$ ./batch_builder INPBB                 ← バッチビルダーの実行
```

## 共通セクション

### **directory**

作業ディレクトリを指定します。  
REIN の実行に必要なすべてのファイルが格納されたディレクトリです。

デフォルト値: “.” (カレントディレクトリ)

### **rein\_input**

REIN プログラムのインプットファイル名を指定します。directory からの相対パスを指定しなければなりません。

デフォルト値: “rein.inp”

### **parallel**

MPI プログラム名を指定します。

デフォルト値: “mpiexec”

### **template\_dir**

バッチビルダーが使用するテンプレートファイルディレクトリへのパスを指定します。この設定が空白の場合、“<バッチビルダーのプログラムパス>/templates”をテンプレートディレクトリとします。

デフォルト値: “”

### **chain\_job**

Chain job (前の計算が終わり次第次の job を submit する手順) のオプション。GE でのみ利用できる。

デフォルト値: “no”

### **batch\_builder\_exe**

Chain job で利用するバッチビルダーの場所。

デフォルト値: “./batch\_builder”

### **batch\_builder\_input**

Chain job で利用するバッチビルダーの場所。

デフォルト値: “./INPBB”

### **spawn**



mpi\_comm\_spawn を利用するかしないか。“no”の場合、1レプリカ 1 ノード以下の利用となる。また rein を nospawn option でコンパイルする必要がある。namd はホームページにあるバイナリが利用できる。

デフォルト値: “yes”

### **openmp**

rein での openMP の有効／無効を指定する。(MD の openMP は num\_threads\_md で決まる。)

デフォルト値: “no”

### **gpgpu**

namd で gpgpu を使用する際の option を自動的に付ける為の option

デフォルト値: “no”

### **hybrid\_md**

K/FX10にて hybrid 版 namd を用いる際には”yes”とする。現在、hybrid 版 namd は不安定なのでデフォルトは”no”になっている。

デフォルト値: “no”

## **[BATCH]セクション**

---

以下のインプットで自動的に計算に必要なノード数などを計算します。

### **batchfile\_name**

REIN が生成する batch file の名前

デフォルト値: “rein.sh”

### **job\_scheduler**

ジョブを投入するマシンのスケジューラを“GE”または“K”で指定します。  
“GE”は UGE または SGE が利用できるマシン、“K”は、京コンピュータおよび FX10 のジョブ管理システム相当が利用できるマシンでそれぞれ指定します。

デフォルト値: “K”

### **queue\_name**

ジョブを投入するキュー名を指定します。

デフォルト値: “small”

#### **num\_threads\_md**

1つのレプリカの MD で用いる thread 数

デフォルト値: 1

#### **num\_mpiproc\_md**

1つのレプリカの MD で用いる MPI プロセスの数

デフォルト値: “8”

#### **num\_core\_cpu**

マシンのノード(もしくはプロセス)当りに割り当てられるコア数。K : 8, FX10: 16 など。

デフォルト値: 8

#### **num\_mpiproc\_rein**

REIN プログラムの実行プロセス数。

デフォルト値: 1

#### **stage\_out\_dir**

REIN ジョブ終了後のステージアウト先を指定します。

デフォルト値: “./STAGE\_OUT”

#### **cpu\_time**

最大実行時間を指定します。

デフォルト値: “01:00:00”

#### **direct\_describe**

node 数などのパラメータは通常自動的に生成されるが、direct\_describe = “yes”とする事で、パラメータを直接書き込む事ができる。

デフォルト値: “no”

#### **machine\_file**

MPI の machine ファイルを指定します。

デフォルト: 空白

## [REMD]セクション

---

### replica\_exchange

YES ならレプリカ交換、NO ならアンプレラサンプリング(交換無し)を行う。  
デフォルト値: “yes”

### num\_of\_exchange\_steps

レプリカ交換回数。  
デフォルト値: 1

## トラジェクトリコンバータインプットファイル

---

プログラム	入力／出力
REIN プログラム	-
インプットビルダー	-
バッチビルダー	-
トラジェクトリコンバータ	入力、出力(雛形)

使用例:

```
$ ./remd_convert -h ctrl > rc.inp      ← 雛形の作成
:
(編集)
:
$ ./remd_convert rc.inp                ← トラジェクトリコンバータの実行
```

## INPUT セクション

---

### psffile

CHARMM psf ファイルを指定します。この設定は省略可能(空白)です。その場合 reffile の情報を参照します。

デフォルト値: “initial.psf”

### reffile

PDB ファイルを指定します。

デフォルト値:

### trj\_format

トラジェクトリファイルのフォーマットを以下の何れかで指定します。

dcd
marble
amber
tinker
pdb
gromacs

デフォルト値: “dcd”

### energy\_column

エネルギートラジェクトリファイルの何カラム目の値を抽出し、出力するかを指定します。  
以下の書式で記述します。

例:

(空白)	すべてのカラム
1 2	1 番目と 2 番目のカラム
1 5-8	1 番目と 5~8 番目のカラム

デフォルト値: “ ”

### directory001-

座標トラジェクトリファイル、エネルギートラジェクトリファイルが置かれているディレクトリのパスを必要なだけ指定します。001-は通し番号で、001, 002, 003 の順番で与えます。

デフォルト値: “ ”

### repeat001-

以降の通し番号付きの設定を何回繰り返すかを指定します。

例えば、同一のトラジェクトリファイル名、エネルギーファイル名を持つディレクトリ指定を directory001-にて 3 つおこなっている場合、repeat001-の値に 3 を指定します。そうすると以降の通し番号付き設定を 1 回記述するだけで済みます。

デフォルト値: なし

### trj\_filename001-

directory001-で指定したディレクトリ以下に格納されている、座標トラジェクトリファイル

名を指定します。

以下のキーワードを指定でき、コンバート処理の際に実際の値に置き換えられ、そのファイル名のトラジェクトリファイルを読み込みます。

{rep_no}	レプリカ番号
{step}	REIN ステップ番号

デフォルト値: “rep. {rep\_no}/rep. {rep\_no}. {step}. dcd”

### **energy\_trj001-**

directory001-で指定したディレクトリ以下に格納されている、レプリカエネルギートラジェクトリファイルを指定します。

デフォルト値: “replica.trj”

### **step\_increment001-**

読み飛ばす REIN ステップ数を指定します。

デフォルト値: 1

## **OUTPUT セクション**

---

### **trj\_filename**

出力するトラジェクトリファイル名を指定します。この設定により、出力される座標トラジェクトリファイル、エネルギートラジェクトリファイルの両方のファイル名が決まります。

以下のキーワードを指定でき、コンバート処理の際に実際の値に置き換えられ、そのファイル名のトラジェクトリファイルが出力されます。

{type}	タイプ文字列“condition”または“replica”
{number}	条件番号またはレプリカ番号
{extension}	拡張子文字列“crdtrj”または“enetrj”

デフォルト値: “trj. {type}. {number}. {extension}”

### **trj\_format**

出力する座標トラジェクトリファイルのフォーマットを以下の何れかで指定します。

dcd
marble
amber
tinker

pdb
gromacs

デフォルト値: “dcd”

#### **out\_pdbfile**

参照座標ファイルを指定します。

デフォルト値: “out.pdb”

## CONVERT セクション

---

#### **type**

コンバートのタイプを以下の何れかで指定します。

replica	レプリカ番号別
condition	条件別

#### **rank**

番号のリストを指定します。type が“replica”の場合レプリカ番号、“condition”の場合は、条件番号を意味します。  
以下の書式で記述します。

例:

(空白)	すべての番号
1 2	1 番と 2 番
1 5-8	1 番と 5~8 番

デフォルト値: 空白

## SELECTION セクション

---

#### **select\_atom**

原子選択式を指定します。

デフォルト値: “all”

#### **mole\_name001-**

分子名を定義します。select\_atom で使用します。

デフォルト値: “”

## PBC セクション

### pbc\_correct

周期境界条件での座標補正を行うかどうかを“no”または“molecule”で指定します。

デフォルト値: “no”

## レプリカ/エネルギートラジェクトリファイル

プログラム	入力／出力
REIN プログラム	出力
インプットビルダー	-
パッチビルダー	-
トラジェクトリコンバータ	入力

REIN プログラムが出力するファイルです。REIN ステップ毎に MD プログラムから得られる各種エネルギー情報とレプリカのランクの情報が記録されています。  
温度レプリカの場合、カラムに出力される情報は以下の通りである。

### 1 次元 REMD の場合

replica 番号, condition 番号, 温度, total energy, kinetic energy, potential energy

### 多次元レプリカの場合

replica 番号, condition 番号, 温度, 距離 (角度…), total energy, kinetic energy, potential energy, extra energy

多次元 REMD の場合の距離情報などは、温度と total energy の間に記述される。

#===== EXCHANGE_STEP = 1						
1	1	290.14150	-1624.33470	370.15570	-1994.49040	
2	2	309.04970	-1677.37840	394.27840	-2071.65680	
3	3	314.56290	-1658.91060	401.31190	-2060.22250	
4	4	313.83160	-1657.91540	400.37900	-2058.29450	
5	5	304.91160	-1642.47860	388.99900	-2031.47760	
6	6	309.84760	-1635.80170	395.29630	-2031.09800	
7	7	299.44280	-1685.76920	382.02210	-2067.79130	
8	8	288.62440	-1683.47140	368.22020	-2051.69170	
#===== EXCHANGE_STEP = 2						
1	2	306.76840	-1664.28540	391.36800	-2055.65340	
2	1	317.18100	-1651.98420	404.65210	-2056.63630	
3	4	310.83750	-1629.92330	396.55920	-2026.48240	
4	3	284.34980	-1655.03960	362.76680	-2017.80640	
5	6	294.21380	-1680.86470	375.35110	-2056.21580	
6	5	336.64260	-1606.88210	429.48070	-2036.36280	
7	8	296.55280	-1698.13390	378.33510	-2076.46900	
8	7	299.56080	-1662.84080	382.17260	-2045.01340	

カラム番号	データ
1	replica 番号

2	condition 番号
3	システムの温度
4	Total エネルギー
5	Kinetic エネルギー
6	Potential エネルギー

レプリカ 1 が温度、レプリカ 2 が距離の場合、

カラム番号	データ
1	replica 番号
2	condition 番号
3	システムの温度
4	距離
5	Total エネルギー
6	Kinetic エネルギー
7	Potential エネルギー
8	extra エネルギー



# 7 テンプレートファイルリファレンス

ツールが利用するテンプレートファイルに関する解説です。

## インプットビルダー用テンプレートファイル

インプットビルダー用のテンプレートファイルは以下のとおりです。

テンプレートファイル		
1	t_rein.inp	REIN インプットファイル
2	t_rein_axis.inp	REIN インプット部分ファイル(軸定義)
3	t_namd.inp	NAMD インプットファイル
4	t_namd_copyfiles	NAMD インプットファイルコピースクリプト
5	t_namd.colvar	NAMD colvar ファイル
6	t_namd_colvar_ang	NAMD colvar 部分ファイル(角度グループ定義)
7	t_namd_colvar_dih	NAMD colvar 部分ファイル(二面角グループ定義)
8	t_namd_colvar_dis	NAMD colvar 部分ファイル(距離グループ定義)
9	t_namd_harmonic	NAMD colvar 部分ファイル(距離グループ定義)
10	t_marble.inp	MARBLE インプットファイル
11	t_marble_group	MARBLE インプット部分ファイル(グループ定義)
12	t_marble_atom_h	MARBLE インプット部分ファイル(拘束条件定義)
13	t_marble_group_h	MARBLE インプット部分ファイル(拘束条件定義)
14	t_marble_copyfiles	MARBLE インプットファイルコピースクリプト

## バッチビルダー用テンプレートファイル

バッチビルダー用のテンプレートファイルは以下のとおりです。

テンプレートファイル		
1	t_batch_ge	UGE/SGE バッチスクリプト
2	t_restart_ge	UGE/SGE 用リスタートセットアップスクリプト
3	t_batch_k	K バッチスクリプト
4	t_restart_k	K 用リスタートセットアップスクリプト
5	t_staging_k_marble	MARBLE 用ステージングファイルリスト
6	t_staging_k_namd	NAMD 用ステージングファイルリスト
7	t_restart_num	ステップ番号指定のリスタートセットアップスクリプト

## 8 REIN の設計思想と動作

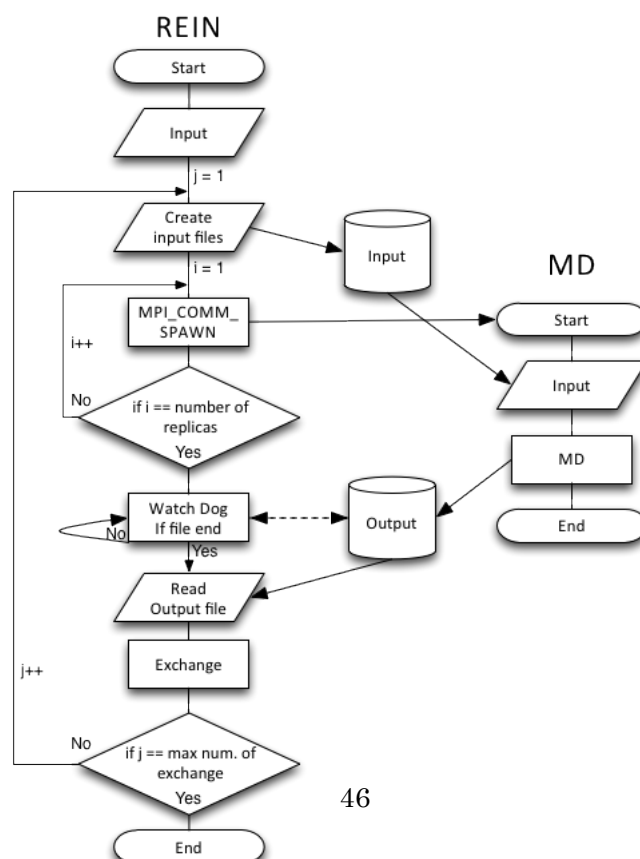
REIN は既存のバイナリプログラムを子プログラムとして制御して、多次元レプリカ交換シミュレーションを実行するインターフェースプログラムです。その為、REIN 単体では意味をなしません。NAMD2 や MARBLE などの子プログラムとし、REIN を親プログラムとする事でレプリカ交換シミュレーションを行う事ができます。子プログラムはバイナリでかつ MPI で並列化されている事を想定しています。REIN はバイナリの子プログラムの修正無しにそのインプットアウトプットを制御する事でレプリカ交換を行うプログラムです。また、子プログラムが MPI /OpenMP hybrid 並列に対応していれば、REIN も MPI/OpenMP hybrid 並列にも対応します。

親プログラムがバイナリの子プログラムを生成するプロセスには MPI-2 の動的プロセス生成機能を使っています。この動的プロセス生成は、簡単なプログラムの改変で複数のバイナリプログラムを組み合わせたり、複数のプログラムを制御する事ができるので将来性のある機能です。

REIN はユーザによって、他のプログラムを子プログラムとして使える様にする、様々なレプリカ交換に対応するなどの拡張を想定しています。新しい子プログラムを導入する場合には lib\_XXX というディレクトリを作成し、i/o 関連モジュールなど必要なプログラムモジュールを入れます。

### REIN のフローチャート

REIN のフローチャートを記載します。



REIN の主な機能はプログラムの流れ順に、1)子プログラムのインプットファイルを作る事、2)子プログラムを MPI\_COMM\_SPAWN で実行させる事、3)子プログラムの output を解析して子プログラムの終了を確認する事 (Watch Dog)、4)子プログラムの output を読んでデータを収集する事、そして 5)レプリカ交換判定をする事の5つです。このうち 3)のプロセスは mpi\_comm\_spawn の機能で返り値が無いためにプログラム終了確認をする必要があります。REIN では Watch dog module を使って子プログラムの終了判定を行っています。

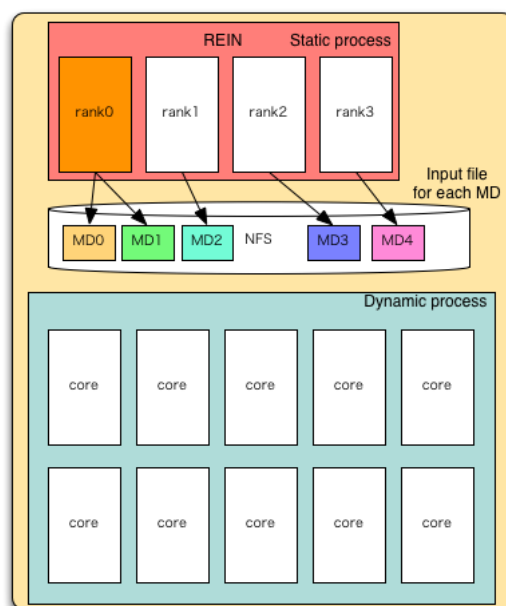
## REIN の動作

REIN の動作のイメージをつかむ為に、全体で 14 core を持つクラスタ上で、5 レプリカ、子プログラム 1 つあたり 2 core を使う場合について考えてみます。また、REIN の実行に 4 core 使う事とします。

REIN は複数の core や node で実行する事ができます。この場合、子プログラムのインプット作成や output の終了確認、output 情報の収集に複数の core と node が使われます。即ち、i/o に関しては MPI/OpenMP hybrid 並列化されています。一方、MPI\_COMM\_SPAWN を用いた子プログラムの実行は rank 0 のノードで、OpenMP を用いている場合には thread の 0 番の core から全て実行されます。これは現時点での MPI\_COMM\_SPAWN の仕様に依存しています。

また、REIN は複数の計算ノードに対して NFS などの shared HDD が必要です。これはレプリカの制御を slave-master 方式にしており、master node から slave node で実行される子プログラムの終了確認をする必要があるからです。即ちそれぞれのノードの local disk を用いた計算はできません。

REIN を実行する際には静的プロセスと動的プロセスという概念を知っておく必要があります。静的プロセスは mpiexec -n で指定する数だけ使う、通常の MPI プロセスです。一方、動的プロセスは mpi\_comm\_spawn など子プログラムの生成で利用するプロセスの事です。REIN が実行されると以下の図の赤色のところで REIN は起動します。REIN では予め、子プロセスの担当プロセッサを決めます。この場合、rank0 は MD0, MD1, rank1 は MD2, rank2 は MD3, そして rank4 は MD4 を担当します。



この場合、4 core 使っています。今回の例では5レプリカですので、子プログラムのインプットファイルが5つ共有ディスクに作成されます。この場合はレプリカ数と REIN で用いるコア数が一致していないので、rank0 で2つ生成しています。理想的には REIN に用いる core 数はレプリカ数の公約数であると良いです。

次に REIN の rank0 から子プログラムを実行します。

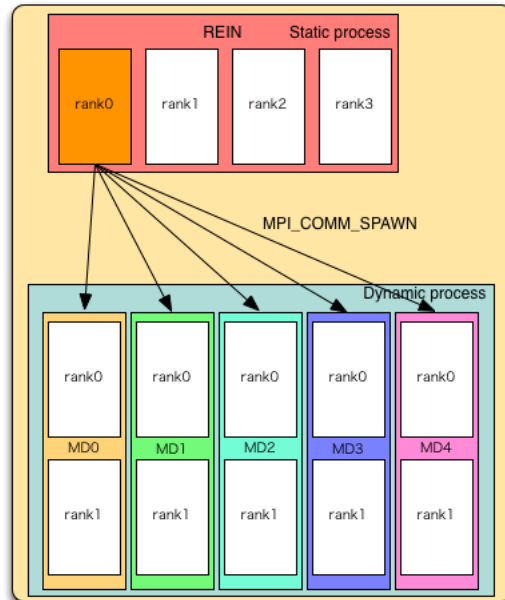


図 2：子プログラムの起動

この時、MPI\_COMM\_SPAWN により、動的プロセス(子プログラム)が REIN の rank 0 のノードから次々に起動されます。MPI\_COMM\_SPAWN の仕様からどのノードにそれぞれの子プログラムが割り当てられるかは一意ではありません。

それぞれの子プログラムはそれぞれのインプットファイルを読みます。

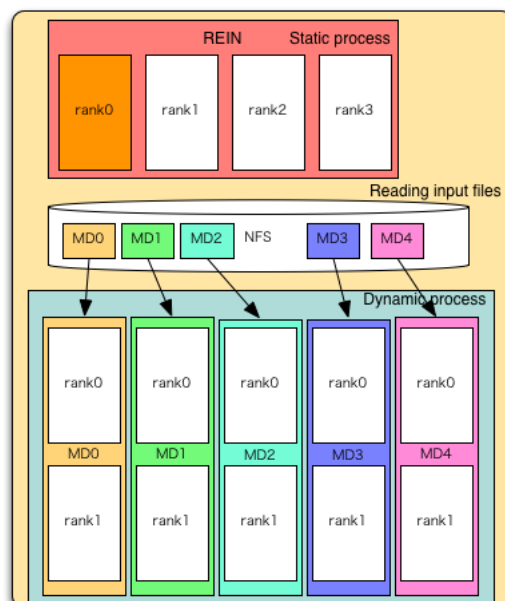


図 3：インプットファイルの読み込み

子プログラムはそれぞれ output を出力します。また、同時に REIN は監視モジュール (Watch dog module) で子プログラムが稼働中かどうかの監視をします。

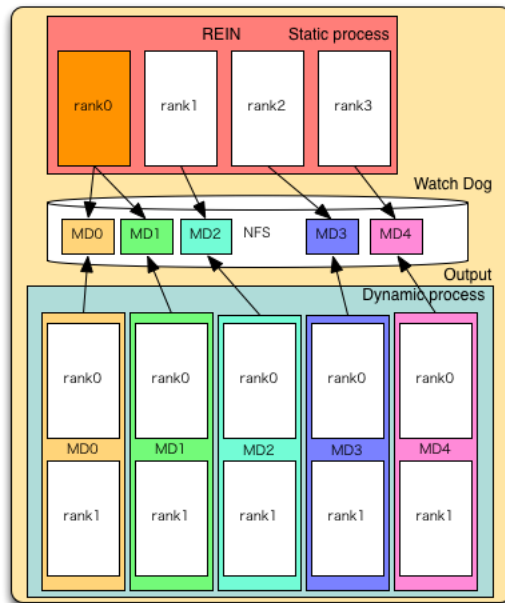


図 4：子プログラムの監視

監視は担当プロセスが行います。今回のケースでは REIN の rank 0 が、子プログラム MD0 と MD1 の2つの監視をします。1つのプロセスで複数の子プログラムの監視を行う場合は順番に監視を行います。この場合、最初に MD0 の監視をした後、MD1 の監視を行います。通常、全ての子プログラムはほぼ同時に実行され終了するので、MD1 の監視は一瞬で終了します。

最後にそれぞれのアウトプットを読み込み、レプリカ交換の評価を行います。この際、REIN の中で、MPI 通信を行います。それぞれの交換の評価はそれぞれの担当ノードで行います。

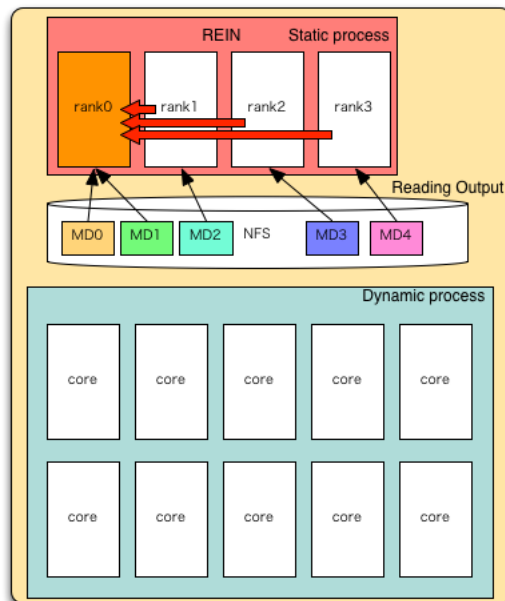


図 5：アウトプットの読み込みとレプリカ交換



