myPresto/psygene-G

利用マニュアル

Version 1.006

2018 年 1 月

真下 忠彰 [1, 3], 福西 快文 [2], 神谷 成敏 [3], 中村 春木 [3]

[1]次世代天然物化学技術研究組合
[2]独立行政法人産業技術総合研究所, 創薬分子プロファイリング研究センター
[3]国立大学法人大阪大学蛋白質研究所

1. インストール法

myPresto/psygene-G（GPU 版空間分割 MD）は、MPI と GPU の同時利用を前提としたプログラムである。このプログラムでは、複数の MPI プロセスが同じ GPU を共有できる。ここでは、以下の様な想定環境（PC-Cluster）でのインストール法について述べる。

1.1. 想定環境

対象となる計算機は、複数 GPU を搭載したノードから構成される Cluster 計算機を想定している。それらには以下 2 つの種類がある。

分離型： MPI と GPU のコンパイルノードがそれぞれ分かれている計算機環境
例えば、GPU が搭載されていない管理ノードに MPI、搭載されている計算ノードに GPU のコンパイラがある
現在（2018/01/13）、この様な環境は存在しない為、本マニュアルでは言及しない

一体型： MPI と GPU のコンパイルノードが同一である計算機環境
例えば、GPU が搭載された管理兼計算ノードに MPI 及び GPU のコンパイラがある
この場合、ソースコードは NFS マウント下の任意の位置に配置する

1.2. コンパイル

myPresto/psygene-G のコンパイルは、上記の想定環境の下で実施する。なお、GPU 機能を省けば CPU 版空間分割 MD：psygene としてコンパイル可能である（GPU 無搭載の CPU クラスタ用）。

1） ソースコードの配置・展開
MPI 及び GPU コンパイラの双方からアクセス可能な位置に配置・展開する。大抵の場合、NFS マウントされたディスク上に配置・展開する為、特に注意する必要は無い。

2） 一体型環境でのコンパイル
MPI コンパイル用 Makefile：
Makefile（src/Makefile）を環境に合わせて適宜書き換える。コンパイラは intel fortran を想定している。
① CUDA のバージョン指定
（リンク”cuda” 若しくは CUDA インストール先ディレクトリ”cuda-x.y（x,y は数字）”を指定）
② CUDA 環境の有効化：GPU を利用する psygene-G のコンパイル
（”# GPU Lib settings”ブロックの有効化＋”# GPU dummy”ブロックのコメントアウト）
③ CUDA 環境の無効化：GPU を利用しない psygene のコンパイル
（”# GPU dummy”ブロックの有効化＋”# GPU Lib settings”ブロックのコメントアウト）
④ MPI コンパイルコマンドの設定
（環境により、mpif90 若しくは mpiifort を選択）
⑤ 大規模系（100 万原子系）メモリサイズの設定（-D_LARGE_SYSTEM）

```
CUDA = cuda                                                          ← ①
…
# GPU Lib settings                                                   ← ②
PRGNAME = psygene-G
USE_GPU = 1
INS_PTH = /usr/local/$(CUDA)
CUDALOC = -Lcuda -L$(INS_PTH)/lib64
CUDALIB = $(CUDALOC) -lgpu_nonewald -lcudart #-cudart_static

# GPU dummy                                                          ← ③
#PRGNAME = psygene
#USE_GPU = 0
#CUDUMMY = -f Makefile.dummy
#CUDALOC = -Lcuda
#CUDALIB = $(CUDALOC) -lgpu_nonewald

# MPI settings
FC = mpif90                                                          ← ④
#FC = mpiifort
#PGFLAGS = -pg
#DBGFLAG = -Warn -fpe0 -traceback -g
OPTIONS = -assume buffered_io -D_SHKEX -D_SHK_THERM -D_LARGE_SYSTEM  ← ⑤
FASTSET = -O3 -xHost # # excluding -ipo -no-prec-div -static from -fast
FFLAGS = $(FASTSET) -c -cpp $(PGFLAGS) $(DBGFLAG) $(OPTIONS) -D_USE_GPU=$(USE_GPU)
LDFLAGS = -mcmodel=large -shared-intel #-i-static
MATHLOC = -L/usr/local/lib64
MATHLIB = -limf -lm -lstdc++
…
```

GPU コンパイル用 Makefile：

Makefile（src/cuda/Makefile）を環境に合わせて適宜書き換える。

⑥ CUDA のバージョン指定

（リンク”cuda” 若しくは CUDA インストール先ディレクトリ”cuda-x.y（x,y は数字）”を指定）

⑦ Compute capability の選択（参照：https://developer.nvidia.com/cuda-gpus）

（例えば compute capability 3.5 の場合、GEN_SM = -arch=sm_35 を選択）

⑧ CUB library の利用（利用しない場合、Thrust library を利用）

（デフォルトは Thrust library （USE_CUB=0）を利用する設定）

⑨ CUDA API 変更への対応

⑩ 詳細情報オプションの選択

```
CUDA = cuda                                                          ← ⑥
…
# --- compute capability ---
…
GEN_SM = -arch=sm_35                                                 ← ⑦
#GEN_SM = -gencode arch=compute_13,code=compute_13 ¥
         -gencode arch=compute_13,code=sm_13      ¥
         -gencode arch=compute_20,code=compute_20 ¥
         -gencode arch=compute_20,code=sm_20      ¥
         -gencode arch=compute_30,code=compute_30 ¥
```

```
        -gencode arch=compute_30,code=sm_30      ¥
        -gencode arch=compute_35,code=compute_35 ¥
        -gencode arch=compute_35,code=sm_35
# --- library ---
# USE_CUB = (cuda version < 5.0) ? 0 : 1
USE_CUB = 0                                                          ← ⑧
ifeq ($(USE_CUB),0)
THR_INC = -I/usr/local/$(CUDA)/include
else
CUB_INC = -Icub-1.7.0
…
# --- options ---
# USE_SYMB = (cuda version < 4.0) ? 0 : 1
USE_SYMB = 1                                                         ← ⑨
…
CUOPT = $(SYMBL) #-D_CUDA_INFO -D_CUDA_TMR -D_CUDA_CHKR2             ← ⑩
…
```

補足事項：

```
-D_CUDA_TMR          = GPU 計算時間の出力（デバッグモード）
-D_CUDA_INFO         = CUDA 関連情報の出力（デバッグモード）
-D_CUDA_CHKR2        = 衝突（< 0.5Å）原子の検出（デバッグモード）
```

コンパイルの実施:

一体型の場合には src/で"make"のみ実行する(リンクまで一括処理)。make の結果、psygene-G が生成される。

```
MPI + GPU ノード:
  $ cd src/
  $ make
```

2. 利用方法

myPresto/psygene-G は myPresto/cosgene 互換のプログラムである為、入出力・実行方法に大きな違いは無いが、機能が制限されている。現時点(2018 年 1 月)での実装状況は以下の通りである。(○:実装済み, △:一部実装, ×:利用不可)

Program			cosgene	psygene	psygene-G	Note
I/O	標準入力	制御	○	○	○	
	ファイル入力	トポロジー	○	△	△	ASCII のみ
		座標	○	△	△	PDB のみ
		SHAKE	○	△	△	HBON のみ
		固定原子	○	×	×	
		CAP	○	×	×	CAP 指定ファイルは未サポート
		拡張 CAP	○	×	×	
		位置拘束	○	○	○	
		距離拘束	○	○	○	

		二面角拘束	○	○	○	
		モニタ対象	○	×	×	
		GB/SA, ASA	○	×	×	
		リスタート	○	○	○	
		剛体モデル	○	×	×	
		UMBRELLA	○	×	×	
		重心合わせ対象	○	×	×	
	ファイル出力	SHAKE	○	×	×	
		トポロジー	○	×	×	
		座標	○	△	△	PDB のみ
		リスタート	○	○	○	
		エネルギー	○	×	×	
		総エネルギー	○	○	○	
		速度	○	×	×	
		トラジェクトリ	○	△	△	座標トラジェクトリのみ
		物理量モニタ	○	×	×	
		ベストフィット	○	×	×	
		RMSD	○	×	×	
	標準出力	ログ	○	○	○	
MD	温度設定	初期・目標温度指定	○	○	○	
		昇温機能	○	○	○	
	Thermostat	Hoover-Evans	○	○	○	
		Nose-Hoover	○	×	×	
	Barostat	Andersen	○	×	×	
		Berendsen	○	○	○	
		Parrinello-Rahmann	○	×	×	
	重心固定	重心固定法	○	○	○	
	分子力場	AMBER	○	○	○	
		CHARMm	○	×	×	
	アンサンブル	NVE	○	○	○	
		NVT	○	○	○	
		NPT	○	○	○	
	拡張アンサンブル	Force-bias	○	×	×	
		Simulated Tempering	○	×	×	
		GST	○	×	×	

	EFFE	○	×	×	
	Tsallis Dynamics	○	×	×	
	TTP-v-McMD	×	○	○	
	TTP-v-AUS	×	○	○	
相互作用	相互作用 Table	○	×	×	
	Spline 補間	○	×	×	
Cutoff 法	Atom	○	○	○	
	Residue center	○	×	×	
	Residue surface	○	×	×	
静電相互作用	Ewald	○	×	×	
	PME	○	×	×	
	FMM	○	×	×	
	FSw-Wolf	○	○	○	
	Zero-multipole	×	○	○	
積分器	Leap-frog	○	○	○	
	Velocity-verlet	○	○	○	
	RESPA	○	×	×	
拘束	SHAKE	○	○	○	
	RATTLE	○	×	×	
	位置	○	○	○	
	距離	○	○	○	
	二面角	○	○	○	
	CAP	○	△	△	原子ベースの CAP
	soft repulsion	○	×	×	
	Flow force	○	×	×	
	Range	○	×	×	
	拡張 CAP	○	×	×	
	frozen atom	○	×	×	
	剛体	○	×	×	
	UMBRELLA	○	×	×	
誘電体	GB	○	×	×	
	ASA	○	×	×	
計算系	孤立系	○	×	×	
	周期系	○	○	○	
計算セル	直方体	○	○	○	

		平行六面体	○	×	×	
		楕円体	○	×	×	
		球体	○	×	×	
Min	最小化	最急降下	○	○	○	
		共役勾配	○	○	○	
Analyze	解析	Force	○	×	×	
		静電ポテンシャル	○	×	×	
		Violation	○	×	×	
		エネルギー(Min)	○	×	×	
		エネルギー(MD)	○	×	×	
		Distance	○	×	×	
		Dihedral	○	×	×	
		Fluctuation	○	×	×	
GPGPU	二体相互作用	GPU 内空間分割法	×	×	○	
		原子総当たり法	×	×	○	
		B-Spline 補間	×	×	○	

※ myPresto/cosgene: 公開版(ver.5.0); myPresto/psygene: CPU 版(ver1.006); myPresto/psygene-G: GPU 版(ver1.006)

2.1. 利用方法

myPresto/psygene-G の基本的な設定・実行方法は既存の myPresto/cosgene_MPI と同じであり、空間分割の設定及び GPU 利用の設定についても制御ファイルで行う。他の利用可能な機能の設定については、myPresto/cosgene のマニュアルを参照する事。

2.1.1. 制御ファイル

制御ファイル(md.inp/min.inp)は基本的に myPresto/cosgene 互換であり、新たに空間分割および GPU の関連項目が追加されている。空間分割項目は"EXE> SPACEINPUT"タグで設定され、必須項目である。

GPU 項目は"EXE> GPU" タグで設定され、必須項目では無い。空間分割数は利用 CPU コア数に合わせるか(最も高速)、利用コア数の倍数になる様に指定する。倍数で無くとも実行出来るが、その場合空間数が最多のコアが律速となる。また、利用コア数が分割数よりも多い条件ではプログラムを起動できない。

GPU を共有利用する場合、CUDA の MPS(Multi-Process Service)を利用すると高速化される場合がある(MPS は"# nvidia-cuda-mps-control -d"で起動、"# echo quit | nvidia-cuda-mps-control"で停止)。

なお、GPU 機能を省いた myPresto/psygene として利用する場合、GPU 項目（"EXE> GPU"）は省略する事。

MD 計算のサンプル：

以下に NPT 計算制御ファイルのサンプルを示す。

```
; example md inputfile for zdip
    EXE> SPACEINPUT                                    ← 中心・セルサイズは"EXE> MD"でも指定
        CENTRX= 0.0D0       CENTRY= 0.0D0       CENTRZ= 0.0D0                ← 中心座標（Å）
        LXCELL= 84.3336 D0  LYCELL=   81.6389 D0  LZCELL=   69.4133 D0       ← セルサイズ（Å）
        DXCELL= 2 DYCELL= 2 DZCELL= 2              ← セル分割数の指定（2*2*2 が最小）
        QUIT
    EXE> GPU                                       ← GPU を利用する時のみ記述する
      ;;; device ;;;
        GPUALC= MPI         USEDEV= 0              ← GPU デバイスの設定法と使用 device ID リスト（例："0,1"）
      ;;; nonbond ;;;
        NBDSRV= GPU                                ← 非結合項（二体相互作用）計算のサービス選択
        NBDKNL= GRID                               ← GPU 計算カーネルの選択
        DBLKNL=NOCA                                ← 単精度計算カーネルの選択
      ;;; b-spline ;;;
        NBDBSP= NOCA                               ← GPU 計算カーネルでの b-spline 補間利用
        ;BSPMIN= 0.40D0     BSPMAX= 17.10D0        ← b-spline 補間範囲の指定（Å）
        ;BSPPNT= 512        BSPODR=   4            ← b-spline 補間の制御点数（区間数）、位数の指定
      ;;; hashed grid ;;;
        GRDSPC= 6.0D0                              ← GPU 内空間分割の分割サイズ（Å）
        GRDSCL= 1.10D0                             ← セルの最大拡張サイズ（セルサイズの 1.1 倍まで）
        QUIT
    EXE> INPUT
        TOPOLOGY= FORM      NAMETO= Pro.tpl
        COORDINA= PDB       NAMECO= Pro04.pdb
        REFCOORD= PDB       NAMERE= Pro04.pdb
        DISTANCE= READ      NAMEDI= Pro.dsr
        QUIT
    EXE> MD
      ;;; md control ;;;
        TIMEST= 0.5d0       LOOPLI= 10000
        OUTLOG= 100         LOGFOR= DETA
      ;;; md restart ;;;
        ;RESTAR= YES        NAMERI= md.rst
        NAMERO= md.rst                             ← restart は MD ループの最後に出力
        ;NAMERO= md.rst     OUTRSL=1000            ← 任意の step 間隔で restart を出力する事も可能
      ;;; ensemble & integration ;;;
        METHOD= NPT
        INTEGR= VELO
      ;;; npt contorl ;;;                          ← cosgene と指定方法が一部異なる
        BAROST= BERE
        MODIFI= ORTH ;ISOT
        SETPRE= 1.0D0
        COUPPI= 1000.0D0
        RELAXA= 100.0D0                            ← 緩和時間（fs）の指定（cosgene と異なる項目）
      ;;; temperature ;;;
        SETTEM= 300.0d0
        INITIA= SET         RANDOM= 111111
        STARTT= 300.0d0
```

```
      ;;; centering ;;;
            STOPCE= TRAN
       ;;; distance restraint ;;;
            CALDSR= CALC          WETDSR= 1.0
      ;;; short range interaction ;;;
            CUTLEN= 12.0D0
            DIEFUN= CONS
            DIEVAL= 1.0D0
            EWAPRM= 0.0D0                          ← α値の指定
            CALZMM= ZDIP                           ← Zero-multipole summation 法の選択
            ;FSWSFW= 1.0D0                         ← FSw-Wolf の Smooth Force width（Å）
      ;;; potential calulation ;;;
            CALV5N= NOCALC
            CALE5N= NOCALC
            CALH5N= NOCALC
            CALHYD= NOCALC
      ;;; boundary ;;;
            BOUNDA= PERI
            CENTRX= 0.0D0         CENTRY= 0.0D0        CENTRZ= 0.0D0        ←"EXE> SPACEINPUT"でも指定
            LXCELL= 84.3336 D0    LYCELL=   81.6389 D0 LZCELL=   69.4133 D0
      ;;; trajectory ;;;
            OUTCOO= 5000          MNTRCO= SING         NAMECO= md.crd       ← trajectory は指定した step で出
力可
            QUIT
      EXE> OUTPUT
            COORDI= PDB           NAMECO= md.npt.pdb                        ← coordinate は計算の最後で出力
            QUIT
      EXE> END
```

MIN 計算のサンプル:

以下に minimize 計算制御ファイルのサンプルを示す。

```
; example minimize inputfile for zdip
      EXE> SPACEINPUT                              ← 中心・セルサイズは"EXE> MD"でも指定
            CENTRX= 0.0D0         CENTRY= 0.0D0        CENTRZ= 0.0D0                ← 中心座標（Å）
            LXCELL= 84.3336 D0    LYCELL=   81.6389 D0 LZCELL=   69.4133 D0         ← セルサイズ（Å）
            DXCELL= 2 DYCELL= 2 DZCELL= 2              ← セル分割数の指定（2*2*2 が最小）
            QUIT
      EXE> GPU                                     ← GPU を利用する時のみ記述する
        ;;; device ;;;
            GPUALC= MPI           USEDEV= 0            ← GPU デバイスの設定法と使用 device ID リスト(例："0,1")
        ;;; nonbond ;;;
            NBDSRV= GPU                                ← 非結合項（二体相互作用）計算のサービス選択
            NBDKNL= GRID                               ← GPU 計算カーネルの選択
            DBLKNL=MANU                                ← 倍精度計算カーネルの選択
        ;;; b-spline ;;;
            NBDBSP= NOCA                               ← GPU 計算カーネルでの b-spline 補間利用
            ;BSPMIN= 0.40D0       BSPMAX= 17.10D0      ← b-spline 補間範囲の指定（Å）
            ;BSPPNT= 512          BSPODR=   4          ← b-spline 補間の制御点数（区間数）、位数の指定
        ;;; hashed grid ;;;
            GRDSPC= 6.0D0                              ← GPU 内空間分割の分割サイズ（Å）
            GRDSCL= 1.10D0                             ← セルの最大拡張サイズ（セルサイズの 1.1 倍まで）
            QUIT
```

```
EXE> INPUT
        TOPOLOGY= FORM      NAMETO= Pro.tpl
        COORDINA= PDB       NAMECO= Pro00.pdb
        REFCOORD= PDB       NAMERE= Pro00.pdb
        POSITION= READ      NAMEPO= MainChain.psr
        ;SETSHAKE= READ     NAMESH= Pro.shk
        QUIT
EXE> MIN
  ;;; min control ;;;
        LOOPLI= 2000
        MONITO= 10          LOGFOR= DETA
  ;;; minimize method ;;;
        METHOD= STEEP       CONVGR= 0.1d0       ← 最小化方法の指定および閾値の設定
  ;;; temperature ;;;
        TEMPER= 300.0d0
  ;;; bestfit ;;;
        BESTFI= YES
  ;;; position restraint ;;;
        CALPSR= CALC        WETPSR= 100.0D0
  ;;; shake method ;;;
        ;SHAKEM= HBON
  ;;; short range interaction ;;;
        CUTLEN= 12.0D0
        DIEFUN= CONS
        DIEVAL= 1.0D0
        EWAPRM= 0.0D0                           ← α値の指定
        CALZMM= ZDIP                            ← Zero-multipole summation 法の選択
        ;FSWSFW= 1.0D0                          ← FSw-Wolf の Smooth Force width（Å）
  ;;; potential calulation ;;;
        CALV5N= NOCALC
        CALE5N= NOCALC
        CALH5N= NOCALC
        CALHYD= NOCALC
  ;;; boundary ;;;
        BOUNDA= PERI
        CENTRX= 0.0D0       CENTRY= 0.0D0       CENTRZ= 0.0D0
        LXCELL= 82.144D0    LYCELL= 79.538D0    LZCELL= 66.008D0
        QUIT
EXE> OUTPUT
        COORDI= PDB         NAMECO= min.steep.pdb
        QUIT
EXE> END
```

なお、設定可能な GPU 関連項目は次の通り。これらの項目は GPU を利用する時のみ設定する（EXE> GPU 内）。

分類	項目	選択/デフォルト値	内容
Device	GPUALC	MPI	GPU の割り当てに MPI 関数を利用
		SMI	GPU の割り当てに nvidia-smi 機能を利用
		RNK	GPU の割り当てに machinefile を利用

			NAMERN で machiefile 名を指定
	ELMDEV	空白	未使用 Device ID リスト(コンマ区切りで指定)
	USEDEV	空白	使用 DeviceID リスト(コンマ区切りで指定)
	GPUPRC	1	1GPU を共有する MPI プロセス数
Nonbond	NBDSRV	NOCA	機能を利用しない
		CPU	CPU で計算
		GPU	GPU で計算
		COMP	CPU と GPU の結果を比較
	NBDKNL	GRID	GPU 内空間分割(grid)を利用
		TILE	総当たり計算(tile)を利用
	DBLKNL	NOCA	単精度計算カーネル使用
		AUTO	失敗時に倍精度計算カーネルによる再計算
		MANU	倍精度計算カーネル使用
B-spline	NBDBSP	NOCA	b-spline 補間を利用しない
		CALC	b-spline 補間を利用
	BSPMIN	0.40	補間区間の最小値
	BSPMAX	17.10	補間区間の最大値
	BSPPNT	512	補間点数(制御点数)
	BSPODR	4	補間位数(4 固定)
Hashed Grid	GRDSPC	6.0	GPU 内分割空間サイズ(1/2cutoff が適当)
	GRDSCL	1.10	最大膨張セルサイズ

また、Zero-multipole summation 法の選択肢(EXE> MD/EXE> MIN 内)は以下の通りである。

分類	項目	選択/デフォルト値	内容
Non-Ewald	CALZMM	NOCA	機能を利用しない(単なる cutoff 法)
		ZMON	Force-Switching Wolf 法を実施 [1)]
		ZDIP	Zero-Dipole summation 法を実施 [2)]
		ZQUA	Zero-Quadrupole summation 法を実施 [3)]
		ZOCT	Zero-Octupole summation 法を実施 [3)]
		ZDEX	Zero-Decaexipole summation 法を実施 [3)]

1) Ref: Yonezawa et al. (2011) *J. Chem. Theory Comput.* **7**, 1484-1493.

2) Ref: Fukuda et al. (2012) *J. Chem. Phys.* **137**, 054314.

3) Ref: Fukuda (2013) *J. Chem. Phys.* **139**, 174107.

2.1.2. MPI 実行

プログラムの起動法は cosgene_MPI の場合と変わらない。利用 CPU コア数・GPU 数は指定並列数(mpiexec オプション"-n")で決定される。

```
$ mpdboot -n $NODES -f $HOSTS                                  ← $NODES : ノード数
                                                                  $HOSTS : ホストファイル
$ mpiexec -machinefile $MCHNF -n $NCORE psygnene-G < md.inp > md.out
                                                               ← $MCHNF : マシンファイル
                                                                  $NCORE : 利用コア数
$ mpdallexit
```

以下、myPresto/psygene-G のマシンファイルの記述例(myPresto/psygene (CPU 版)の場合、GPU 数を考慮する必要は無い)。

```
$ cat machinefile                    4GPUs/node の計算機で 2node(8CPUcore+8GPU)利用の場合
gp01
gp01
gp01
gp01
gp02
gp02
gp02
gp02
```

※ 最高速にするにはマシンファイルは 1 ノード当たりの GPU 数 = 1 ノード当たりの利用コア数となる様に記述する。

2.1.3. LSF 実行

プログラムの起動法は myPresto/cosgene_MPI の場合と変わらない。利用 CPU コア数と利用 GPU 数は、指定並列数(bsub オプション"-n")で決定される。

(1CPUcore + 1GPU)× 8node の場合:

```
$ bsub -a mpich2 -q qgpu -n 8 -R "span[ptile=1]" -i md.inp -o md.out mpirun.lsf psygnene-G
```

(4CPUcore + 4GPU)× 2node の場合:

```
$ bsub -a mpich2 -q qgpu -n 8 -R "span[ptile=4]" -i md.inp -o md.out mpirun.lsf psygnene-G
```

2.1.4. PBS 実行

プログラムの起動法は myPresto/cosgene_MPI の場合と変わらない。利用 CPU コア数と利用 GPU 数は指定並列数(PBS オプション"-l nodes")で決定される。

(4CPUcore + 4GPU)× 2node の場合:

```
$ cat run.sh
#! /bin/bash
#PBS -l nodes=gp01:ppn=4+gp02:ppn=4
```

```
#PBS -N gp0102
#PBS -j oe
cd $PBS_O_WORKDIR
NCORE=`cat $PBS_NODEFILE | wc -l`
EXE=/work1/GPU/bin/psygene-G
mpirun -r ssh -machinefile $PBS_NODEFILE -n $NCORE $EXE < md.inp > $PBS_O_WORKDIR/md.out

$ qsub run.sh
```

3. GPU カーネル

myPresto/psygene-G では二体相互作用計算が GPU 移植されており、その GPU カーネルには総当たり(tile 法)及び空間分割法(grid 法)の二種類がある。また、Zero-multipole summation 法の特徴として Ewald 法の係数 αに相当する係数(dumping factor) : α= 0.0 の場合が最も精度が高くかつ式も単純になる(誤差関数を含まない)ため、α=0.0 専用の GPU カーネルが用意されている。その一方、Zero-monopole summation 法(Force Switching -Wolf 法) には α=0.0 専用の GPU カーネルは存在しない。

3.1. 総当たり法(tile 法)

二体相互作用計算において総当たり計算を実施する単純な方法である。1000 原子程度であれば空間分割法よりも高速である。 Zero-monopole summation 法(Force Switching -Wolf 法)を含む Zero-multipole summation 法に対応する GPU カーネルが用意されており、利用するには制御ファイル(md.inp)内の"EXE> GPU"にて"NBDKNL = TILE"を選択する。

3.2. 空間分割法(grid 法)

空間分割による二体相互作用計算を実施する方法である。計算系が大きい場合には総当たり法よりも高速である。Zero-monopole summation 法(Force Switching -Wolf 法)を含む Zero-multipole summation 法に対応する GPU カーネルが用意されており、利用するには制御ファイル(md.inp)内の"EXE> GPU"にて"NBDKNL = GRID"を選択する。

3.3. B-spline 補間

総当たり法・空間分割法に関わらず、α≠0.0 の場合には、GPU カーネル上での静電相互作用計算に誤差関数を含む処理が必要となる。その為、静電エネルギー・力の関数全体を b-spline 補間式で表現し、それを使って静電相互作用を計算する GPU カーネルを用意している。Zero-multipole summation 法の α≠0.0 及び Zero-monopole summation 法(Force Switching -Wolf 法の計算でこのカーネルを使用する。以下の様に制御ファイル(md.inp)内の"EXE> GPU"内にて b-spline 関連の設定を行う。

```
EXE> GPU
  …
  ;;; b-spline ;;;
      NBDBSP= CALC                                ← GPU 計算カーネルでの b-spline 補間利用
      BSPMIN= 0.40D0      BSPMAX= 17.10D0         ← b-spline 補間範囲の指定(Å)
      BSPPNT= 512         BSPODR=  4              ← b-spline 補間の制御点数(区間数)、位数の指定
```

```
    ...
```