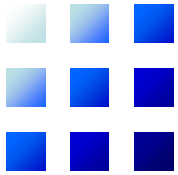


大規模並列量子化学計算プログラムSMASH
講習会資料(要約版)

石村　和也 (ishimura.smash@gmail.com)
(分子研、ポスト京重点課題5)

FOCUS講習会
2018年12月6日

資料内容

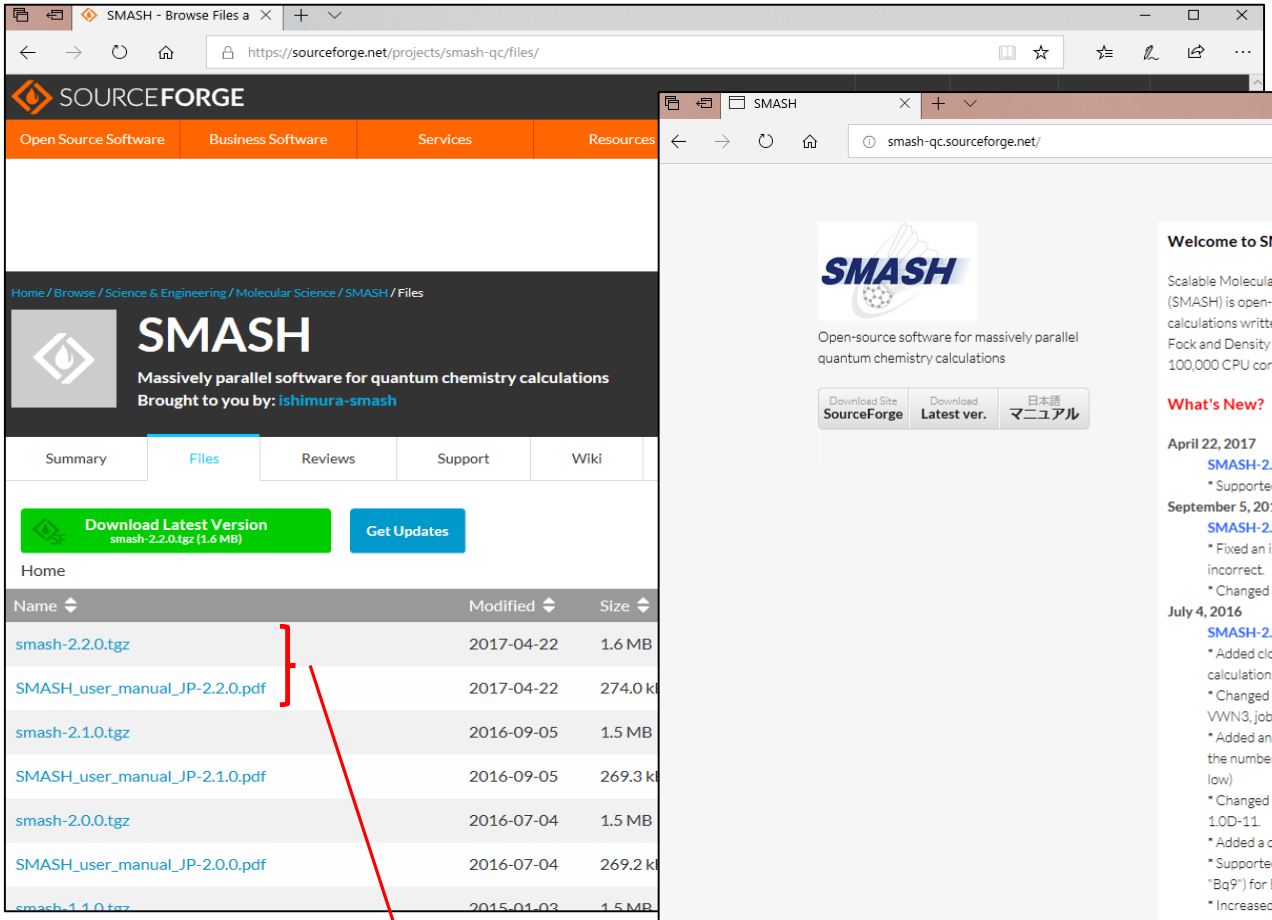
- SMASHプログラムの概要
- SMASHの実行性能
- SMASHのインプット及びアウトプットファイル
- 演習課題

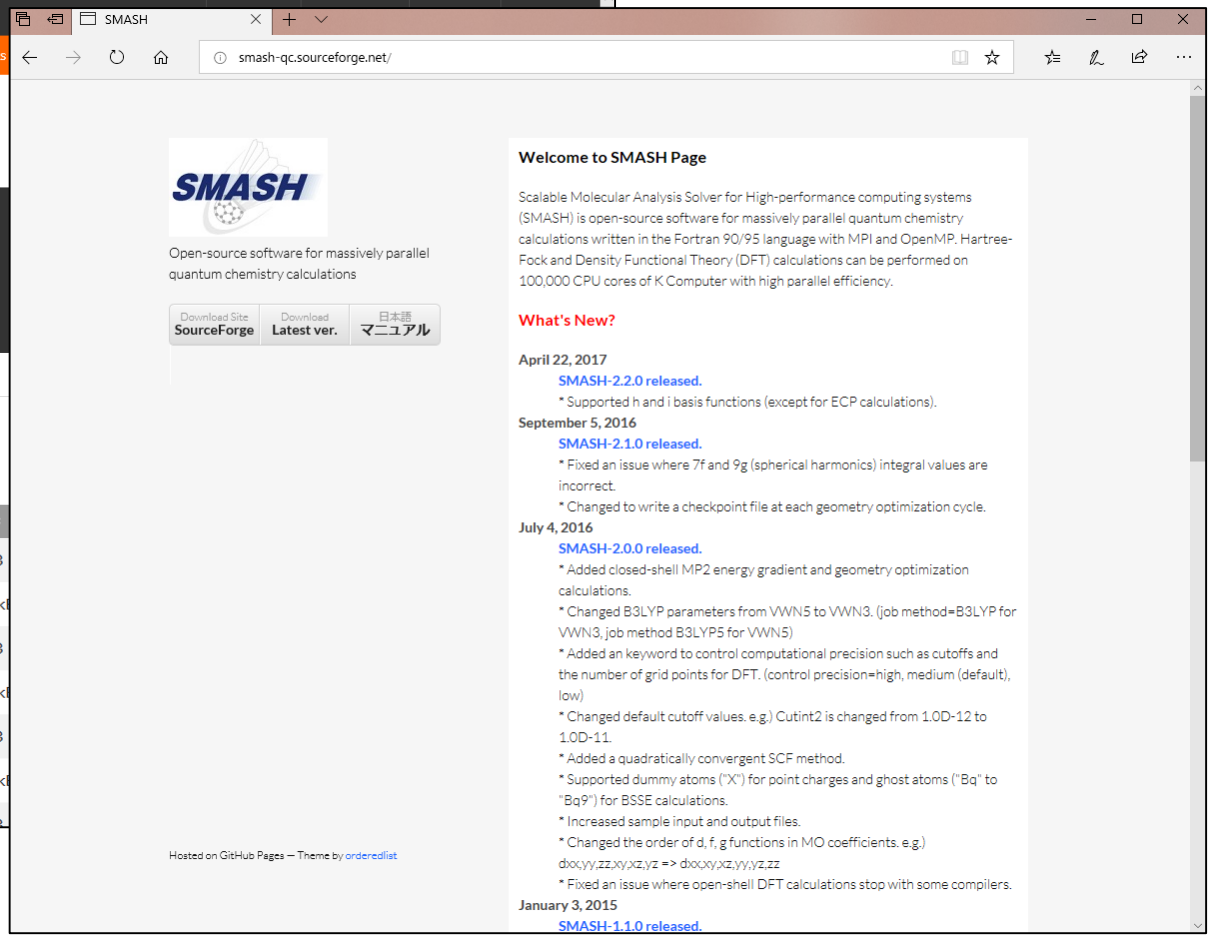
SMASHプログラム

- 大規模並列量子化学計算プログラムSMASH (Scalable Molecular Analysis Solver for High performance computing systems)
- オープンソースライセンス(Apache 2.0)
- http://smash-qc.sourceforge.net/
- キーワードはシンプル:ライセンス、入力形式、計算実行
- 対象マシン:スカラ型CPUを搭載した計算機(PCクラスタから京コンピュータまで)
- 高速と高並列を両立
- エネルギー微分、構造最適化計算を重点的に整備、現在2次微分並列計算、電荷解析計算アルゴリズムを開発中
- 現時点で、Hartree-Fock, DFT(B3LYP), MP2計算が可能
- 電子相関計算の大容量データをディスクではなくメモリ上に分散保存
- 言語はFortran90/95、並列化はMPI及びOpenMP
- 1,2電子積分など頻繁に使う計算ルーチンのライブラリ化で開発コスト削減
- 2014年9月から2018年11月までの約4年間で、ダウンロード数は1500程度

SMASH ウェブサイト

http://smash-qc.sourceforge.net/

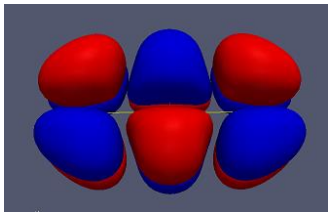
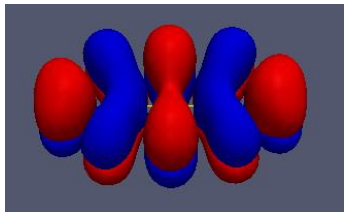




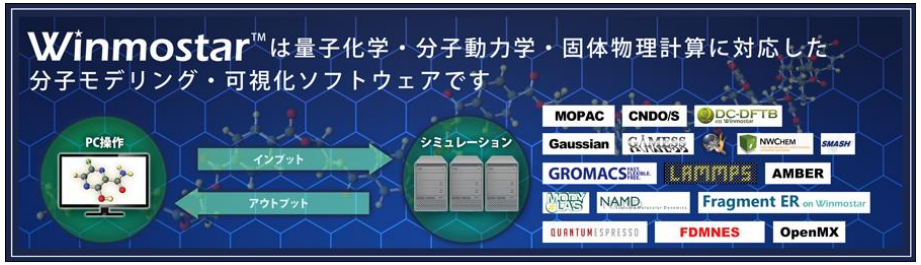
最新ソースコードと
日本語マニュアル

SMASH計算結果の可視化

- 可視化
  - Version1では、フリー可視化ソフトParaViewでMOを表示させるためvtkファイルを作るプログラムを用意

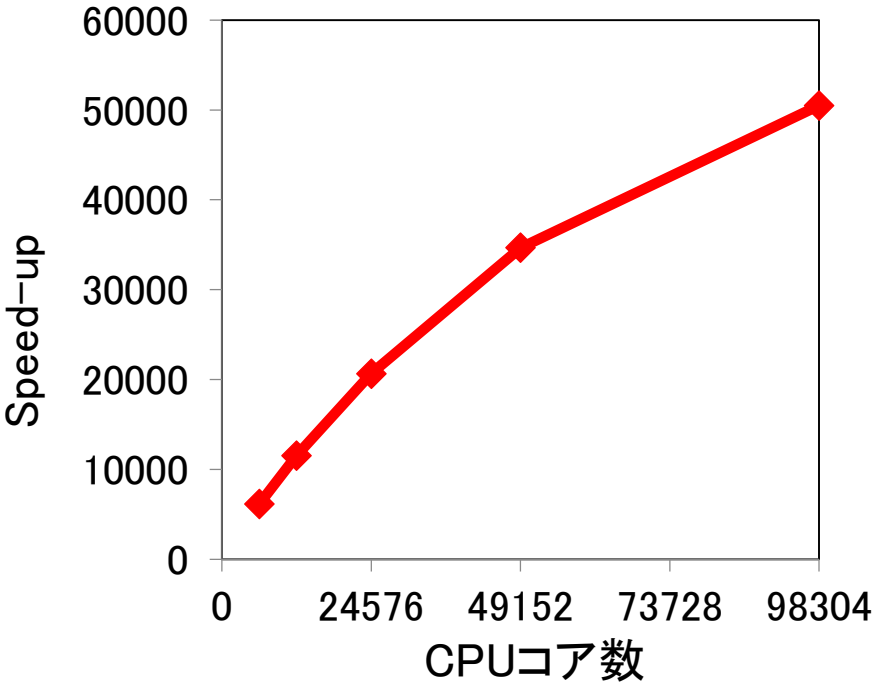



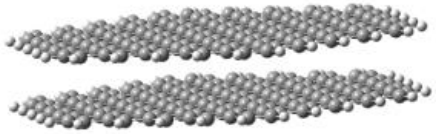
  - Version2からは、GaussViewなどでも表示できるようcubeファイルを作るプログラムも用意
  - 現在、X-Ability社のWinmostarでも対応を進めている



B3LYPエネルギー並列計算性能

- 10万コアで5万倍のスピードアップ、実行性能13%
- 10万コアで360原子系のB3LYP計算が2分半
- 行列対角化(LAPACK,分割統治法)3回分の時間は約35秒
  →ScaLAPACK、EigenExaなどプロセス並列化されているライブラリ導入が必要





計算機：京コンピュータ
分子　：$(C_{150}H_{30})_2$ (360原子)
基底関数：cc-pVDZ
(4500基底)
計算方法：B3LYP
SCFサイクル数：16

CPUコア数	6144	12288	24576	49152	98304
実行時間 (秒)	1267.4	674.5	377.0	224.6	154.2

Hartree-Fockエネルギー1ノード計算性能

- 1ノードでは、GAMESSよりHartree-Fock計算時間を最大40%削減
- SMASHではS関数とP関数を別々に計算するため、SP型の基底ではGAMESSとの差は小さい

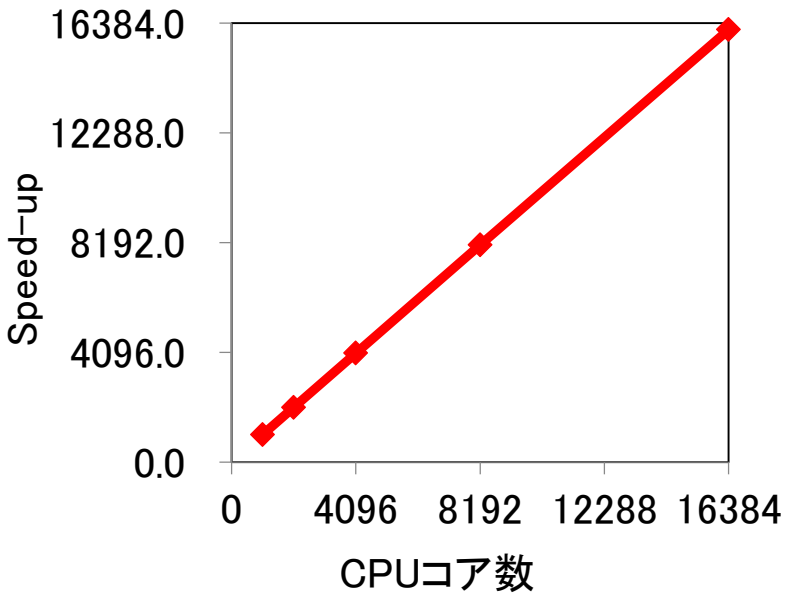
GAMESSとの比較

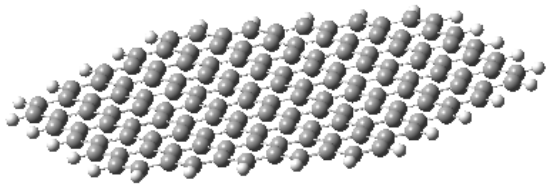
- Xeon E5649 2.53GHz 12core、1ノード利用
- Taxol($C_{47}H_{51}NO_{14}$)のHartree-Fock計算時間(sec)
- 同じ計算条件(積分Cutoffなど)

基底関数	GAMESS	SMASH
6-31G(d) (1032 functions)	706.4	666.6
cc-pVDZ (1185 functions)	2279.9	1434.3

B3LYPエネルギー微分並列計算性能

- エネルギー計算と異なり対角化計算が無いため、並列化効率はほぼ100%





計算機：京コンピュータ
分子　：($C_{150}H_{30}$)
基底関数：cc-pVDZ (2250 functions)
計算方法：B3LYP

Table B3LYPエネルギー1次微分計算時間(秒)と並列加速率(カッコ内)

CPUコア数	1024	4096	8192	16384
微分計算のみ	402.0 (1024.0)	101.2 (4067.7)	50.8 (8103.3)	25.5 (16143.1)

MP2エネルギー並列計算性能

- MP2エネルギー微分計算の並列加速率
  - 計算条件：$C_{150}H_{30}$、MP2/cc-pVDZ
  - 計算機：京コンピュータ
  - 使用ノード数が増えるほどトータルのメモリ量は増えて、占有軌道分割数が減るため、使用ノード数以上の並列加速率が得られた

計算時間(秒)と並列加速率

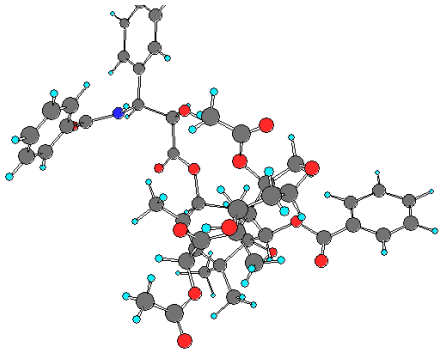
CPUコア数	1536	3072	6144	12288
占有軌道分割数	3	2	1	1
MP2微分実行時間	4253.6	2002.0	743.6	382.0
並列加速率	1536.0	3263.2	8785.6	17103.2

構造最適化回数の削減

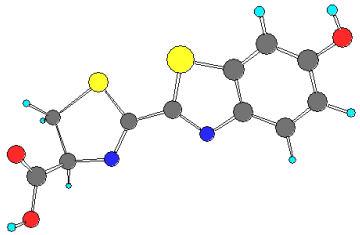
- Redundant座標と力場パラメータを使い、初期ヘシアンを改良することで、Cartesian座標に比べて最適化回数が1/5から1/6になった
- 2サイクル目以降のヘシアンはBFGS法で更新
- Redundant座標がSMASHではデフォルトになっている

Table B3LYP/cc-pVDZ構造最適化回数(初期構造HF/STO-3G)

	Cartesian 座標		Redundant 座標
Luciferin($C_{11}H_8N_2O_3S_2$)	63	⇨	11
Taxol ($C_{47}H_{51}NO_{14}$)	203		40



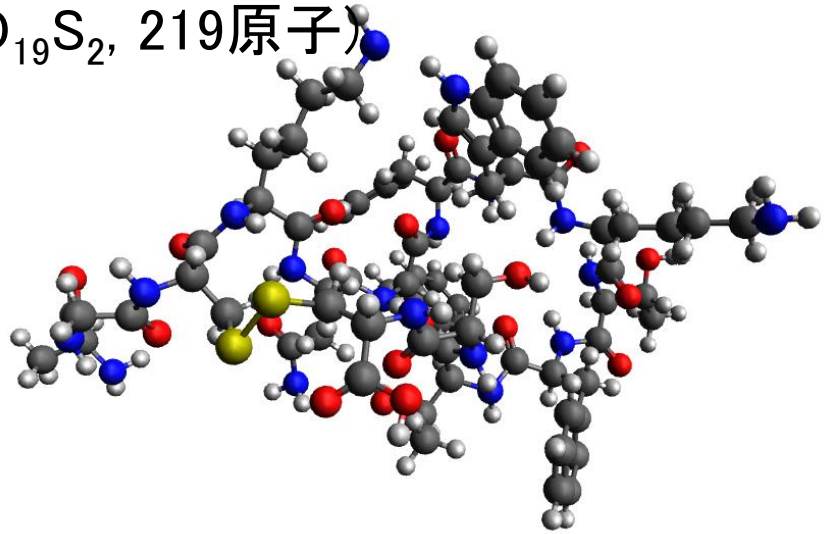
Taxol ($C_{47}H_{51}NO_{14}$)



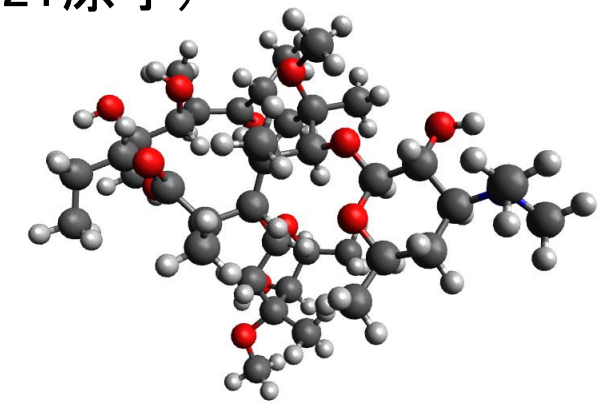
Luciferin($C_{11}H_8N_2O_3S_2$)

FOCUSスパコンでのSMASHの測定

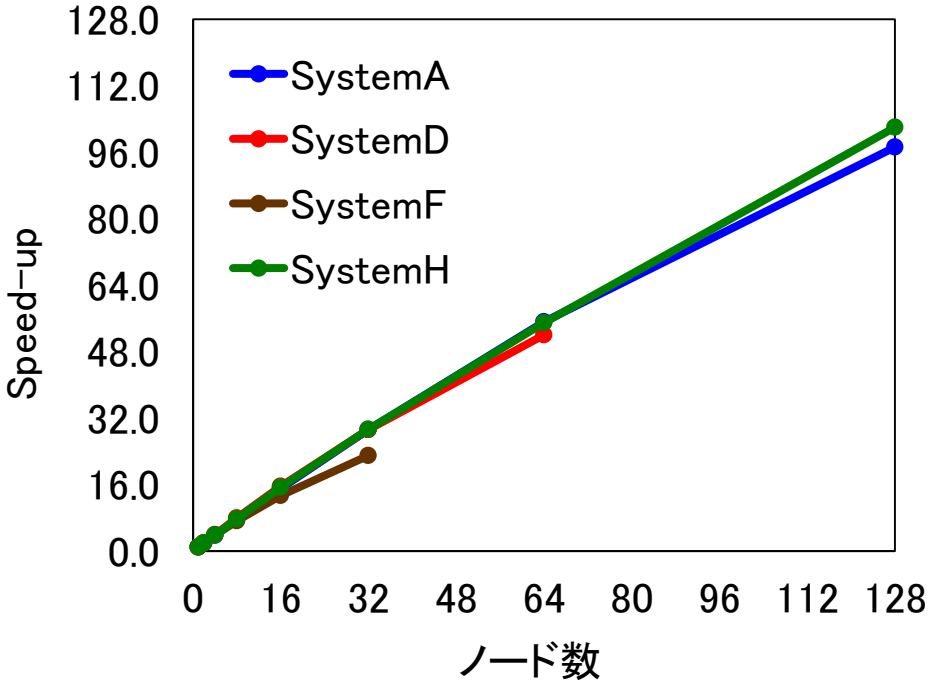
- DFT(B3LYP)構造最適化計算1サイクル
  - 分子：Somatostatin ($C_{76}H_{104}N_{18}O_{19}S_2$, 219原子)
  - 基底関数：6-31G* (1826次元)



- MP2構造最適化計算1サイクル
  - 分子：Clarithromycin ($C_{38}H_{69}NO_{13}$, 121原子)
  - 基底関数：6-31G* (866次元)



B3LYP構造最適化計算1サイクル(219原子)

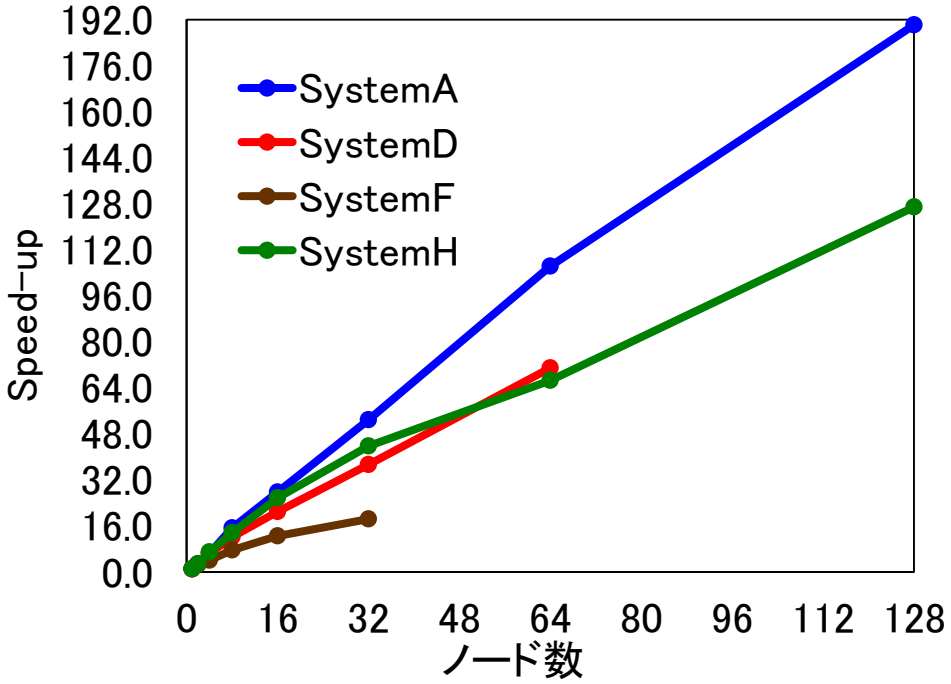


B3LYP構造最適化計算1サイクルの実行時間(秒)

ノード数	1	2	4	8	16	32	64	128
SystemA	10904.3	5604.9	2848.9	1429.3	724.4	372.5	197.4	112.1
SystemD	4073.9	2023.1	1017.3	510.1	260.8	139.1	78.3	
SystemF	1649.5	857.8	426.5	226.5	123.3	71.7		
SystemH	9142.2	4612.6	2319.1	1171.4	593.5	311.4	166.3	89.6

- ✓ どのシステムでもノード数が増えれば増えるほど速くなる
- ✓ 200原子系のDFT構造最適化計算は、1サイクル数分で実行可能

MP2構造最適化計算1サイクル(121原子)



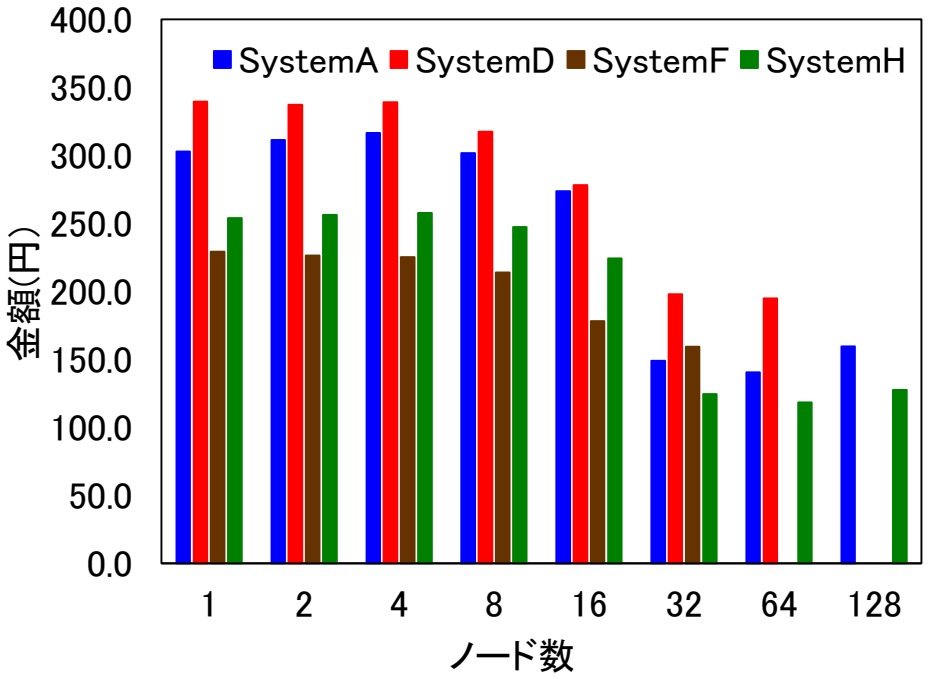
MP2構造最適化計算1サイクルの実行時間(秒)

ノード数	1	2	4	8	16	32	64	128
SystemA	18733.5	7123.8	2921.8	1230.4	677.2	354.3	176.3	98.5
SystemD	6339.1	2354.3	991.3	522.9	303.1	169.9	89.4	
SystemF	2312.7	921.4	574.0	307.4	185.4	125.8		
SystemH	14063.5	5033.9	2044.0	1031.6	547.7	321.6	211.3	110.9

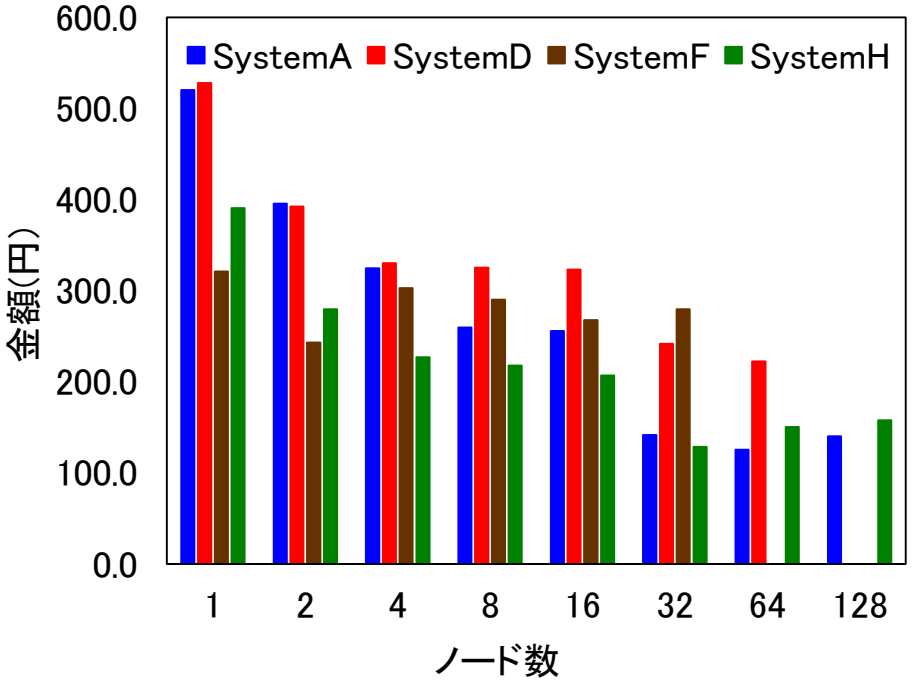
- ✓ ノード数が増えると総メモリ量が増加するため、使ったノード数以上のSpeed-upになる場合がある
- ✓ 100原子系のMP2構造最適化計算は、1サイクル数分で実行可能

1ジョブ当たりの金額(2018年度)

DFT



MP2



- ✓ DFT、MP2計算どちらもノード数が増えると、1ジョブ当たりの金額は小さくなる
- ✓ 少ないノード数ではSystemF、SystemHが安く、多いノード数ではSystemAも安くなる

SMASHのコンパイル方法

- 必要なコンパイラはFortran、ライブラリはBLAS・LAPACK
- 並列化はノード間がMPI(MPIライブラリが必要)、ノード内はOpenMP(コンパイラオプションで指定)
- FOCUSスパコンにはすでに実行ファイルが用意されている

1. ダウンロードしたファイルを展開

   `tar xfz smash-2.2.0.tgz`
2. 展開でできたディレクトリに移動

   `cd smash`
3. makeを実行

   ifortベースのmpif90を使う場合：`make`

   mpiifortを使う場合：`make -f Makefile.mpiifort`

   京やFX100を使う場合：`make -f Makefile.fujitsu`

   ifortを使う場合：`make -f Makefile.x86_64.noMPI`

SMASHの実行方法

- MPI(ノード間)並列

```
mpirun -np (プロセス数) ./bin/smash < (inputファイル名) > (outputファイル名)
```

- OpenMP(ノード内)並列
  - bashの場合：~/.bashrcファイルに次の行を追加して

```
export OMP_NUM_THREADS=(スレッド数)
ulimit -s unlimited
export OMP_STACKSIZE=1G
```

次のコマンドを実行

```
source ~/.bashrc
```

  - tcshの場合：~/.tcshrcファイルに次の行を追加して

```
setenv OMP_NUM_THREADS (スレッド数)
unlimit
setenv OMP_STACKSIZE 1G
```

次のコマンドを実行

```
source ~/.tcshrc
```

SMASHのインプット

- サンプルインプット・アウトプットファイルはsmash/example/を参照
- 計算方法や条件をjob, control, scf, opt, dft, mp2セクションで設定
- 分子座標はgeom行の次から記入、空行もしくはファイルの最後で座標読み込み終了
- 基底関数やECPを元素ごとに指定する場合、basis行、ecp行の次から記入
- 原子核の電荷(点電荷)の指定はcharge行の次から記入
- 大文字と小文字の区別は無し

H_2OのB3LYP/6-31G(d)構造最適化

```
job  runtype=optimize method=b3lyp basis=6-31g(d)
geom
 O   0.0000000    0.0000000    0.1423813
 H   0.0000000    0.7568189   -0.4626257
 H   0.0000000   -0.7568189   -0.4626257
```

インプット指定(jobセクション)

変数	内容	設定値
runtype	計算実行方法	energy: エネルギー計算 (default) gradient: エネルギー微分計算 optimizeもしくはopt: 構造最適化計算
method	計算方法	HF: Hartree-Fock計算 (default) b3lyp: B3LYP計算 b3lyp5: B3LYP計算 with VWN5 mp2: MP2計算
basis	基底関数	sto-3g (default), 6-31g, 6-31g(d), cc-pvdz, cc-pvtz, cc-pvqz, d95v, lanl2dz gen: 元素ごとに指定
memory	1ノード当たりメモリ使用量	(default 8GB) 単位:B, KB, MB, GB, TB
charge	系の電荷	(default ±0.0) Chargeセクションで指定された点電荷の値は含まない
multi	スピン多重度	1: singlet, 2: doublet, 3: triplet, ...
scftype	波動関数種類	RHF : Closed-shell (default) UHF: Open-shell
ecp	ECP (Effective Core Potential)	none (default) lanl2dz gen: 元素ごとに指定

インプット指定(controlセクション)

変数	内容	設定値
precision	計算精度の制御(原子軌道2電子積分計算のカットオフ、SCF収束判定、DFTのgrid点数)	high medium (default) low
spher	Spherical Harmonicsもしくは Cartesian基底指定	.true. : Spherical (5d, 7f,...) (default) .false. : Cartesian (6d, 10f,...)
guess	初期波動関数	huckel : 拡張Huckel計算 (default) check : チェックポイントファイル参照
check	チェックポイントファイル名	(default:空白)
cutint2	2電子積分のカットオフ	1.0d-11 (default)
bohr	距離の単位	.false. : Å(default) .true. : bohr
iprint	出力制御	1 : 最少出力 2 : 通常出力 (default) 3 : 詳細出力

インプット指定(scfセクション)

変数	内容	設定値
scfconv	SCF収束方法指定	DIIS: DIIS法 (default) SOSCF: Approximated Second-orderSCF法 QC: Quadratically convergent SCF法
maxiter	最大SCF回数	150 (default)
dconv	SCFでの電子密度収束判定	5.0D-6 (default)
maxdiis	最大DIIS回数	20 (default)
maxsoscf	最大SOSCF回数	20 (default)
maxqc	最大QC回数	15 (default)

インプット指定(opt,dft,mp2セクション)

opt

変数	内容	設定値
nopt	最大Opt回数	100 (default)
optconv	OptでのForce収束判定	1.0D-4 (default)
cartesian	構造最適化時の座標系	.true. : Cartesian coordinate .false. : Redundant coordinate (default)

dft

変数	内容	設定値
nrad	汎関数数値積分の動径点数	96 (default)
nleb	汎関数数値積分のLebedevグリッド角度点数	302 (default)

mp2

変数	内容	設定値
Ncore	Frozen core軌道数	自動的に計算 (default)
nvfz	Frozen virtual軌道数	0 (default)

インプット指定(geomセクション)

- “geom”の次の行から分子構造の読み込みが始まる。
- 1行に1原子で、元素記号とxyz座標を書く。
- 空行もしくはファイルの最後で分子構造の読み込みが終了する。
- チェックポイントファイルから読むときは、初めの行をgeom=checkと書く。
- 点電荷のためのdummy atomは”X”で指定する。
- BSSEなどで基底やECPを置いたり、DFTのgrid点を作成するghost atomは”Bq1(=Bq)”から”Bq9”まで利用可能。

```
job  runtype=optimize method=b3lyp basis=6-31g(d)
geom
 O   0.0000000    0.0000000    0.1423813
 H   0.0000000    0.7568189   -0.4626257
 H   0.0000000   -0.7568189   -0.4626257
```

インプット指定(basisセクション)

- jobセクションでbasis=genを指定して、”basis”の次の行から元素ごとに指定する
- 関数名(6-31G(d)、cc-pVDZ、LANL2DZなど)でも関数の直接記入でも可
- 関数名と関数の併用も可(LANL2DZにd関数追加など)
- 元素ごとの区切りは****で、basis指定の最後も****
- 個別の関数のフォーマットは次の通り(Gaussianと同じ)

```
(元素記号)
  (軌道角運動量 (S,P,D,F,G,H,I,SP)) (primitive関数の数)
    (Gauss関数の指数) (短縮係数)
    (Gauss関数の指数) (短縮係数)
    ...primitve関数の数繰り返し
  ...関数指定の繰り返し
  ****
```

```
basis
Se
LanL2DZ
D    1
   0.384     1.0
****
C
6-31G(d)
****
H
  S   3
     3.4252509        0.15432897
     0.6239137        0.53532814
     0.1688554        0.44463454
  ****
```

インプット指定(ecpセクション)

- jobセクションでecp=genを指定して、”ecp”の次の行から元素ごとに指定する
- 関数名(LANL2DZ)でも関数の直接記入でも可
- ecp指定の最後は空行もしくはファイルの最後
- 個別の関数のフォーマットは次の通り
  (Gaussianと同じ)

(元素記号)
(関数名(任意)) (最大軌道角運動量) (Core電子数)
(タイトル(任意))
(Gauss関数の数)
(Rの次数) (Gauss関数の指数) (Gauss関数の係数)
(Rの次数) (Gauss関数の指数) (Gauss関数の係数)
...Gauss関数の数繰り返し
...Gauss関数指定の繰り返し

```
ecp
Au
LanL2DZ
Cl
Cl-ECP     2     10
d-ul potential
  5
1      94.8130000           -10.0000000
2     165.6440000            66.2729170
2      30.8317000           -28.9685950
2      10.5841000           -12.8663370
2       3.7704000            -1.7102170
s-ul potential
  5
0     128.8391000             3.0000000
1     120.3786000            12.8528510
2      63.5622000           275.6723980
2      18.0695000           115.6777120
2       3.8142000            35.0606090
p-ul potential
  6
0     216.5263000             5.0000000
1      46.5723000             7.4794860
2     147.4685000           613.0320000
2      48.9869000           280.8006850
2      13.2096000           107.8788240
2       3.1831000            15.3439560
```

インプット指定(chargeセクション)

- ”charge”の次の行から原子の核電荷(小数点も可)を指定する
- 点電荷を置く場合は、dummy atomをgeomセクションでXを使って指定しておく
- Counterpoise計算などで基底関数やECPを置く場合は、ghost atomをBq(=Bq1)からBq9までを使って指定する
- 電荷指定のフォーマットは次の通り

(geomで書いた原子の番号) (電荷)
以下同様

```
geom
O    0.0    0.0    0.0
H    1.0    0.0    0.0
H    0.0    1.0    0.0
X    0.0    0.0    1.5
X    0.0    0.0   -2.0

charge
4   1.0
5   -0.5
```

SMASHインプット一覧

- example/ディレクトリに用意しているインプットファイル
- どのような設定をしているかわかるようなインプットファイル名になっている

b3lyp-energy.inp
b3lyp-high-precision.inp
b3lyp-opt.inp
basis-gen.inp
check-generation.inp
check-read.inp
ecp-gen.inp
ecp-lanl2dz.inp
large-memory.inp
mp2-energy.inp
mp2-opt.inp
point-charge.inp
uhf-opt.inp

SMASHアウトプット1

アウトプット冒頭(B3LYP/6-31G*のエネルギー計算)

```
The job started at Tue Sep 11 17:35:36 2018                    ← 計算開始時刻
Master node is a161                                            ← マスターノード
Number of processes =    16                                    ← MPI並列数
Number of threads   =    12                                    ← OpenMP並列数
------------------------------------------------------------
  Job infomation                                               ← インプットの設定内容
------------------------------------------------------------
  Runtype = ENERGY  ,  Method  = B3LYP    ,  Basis    = 6-31G*
  Memory  =      8000MB ,  SCFtype = RHF      ,  Precision= MEDIUM
  Charge  =        0.0      ,  Multi   =          1    ,  Spher    = T
  Bohr    = F               ,  Guess   = HUCKEL
------------------------------------------------------------
------------------------------------------------
  Computational condition                                      ← 計算条件一覧
------------------------------------------------
  Number of atoms                    =  219
  Number of alpha electrons          =  435
  Number of beta electrons           =  435
  Number of basis shells             =  902
  Number of basis contracted functions = 1826
  Number of basis primitive functions  = 2165
------------------------------------------------
--------------------------------------------------
        Molecular Geometry (Angstrom)
 Atom          X           Y           Z                       ← インプットの分子構造
--------------------------------------------------
  N        3.3644038    0.7678057   -0.6009326
  C        2.5680453    0.6959527    0.6572259
(省略)
-------------
  Basis set                                                    ← 基底関数
-------------
 N
   S  6
      4173.5114600      0.00183477
       627.4579110      0.01399463
```

SMASHアウトプット2

アウトプット半ば(B3LYP/6-31G*のエネルギー計算)

```
------------------------------------------------------------------------------
  Restricted DFT calculation                                    ← DFT計算条件一覧
------------------------------------------------------------------------------
  SCFConv  =    DIIS     , Dconv     =   5.00D-06,  MaxIter    =      150
  Cutint2    =   1.00D-11,  ThreshEx   =  3.00D+01,  ThreshOver = 1.00D-06
  Nrad       =          96,  Nleb       =        302,  ThreshRho  = 1.00D-05
  ThreshDfock= 1.00D-04,  Threshdftao= 1.00D-03,  ThreshWeight=1.00D-08
  MaxDIIS    =          20,  ThreshDIIS =  6.00D-01
------------------------------------------------------------------------------
====================
  SCF Iteration                                                 ← SCF計算状況
====================
Iter SubIt   Total Energy      Delta Energy      Delta Density     DIIS Error
  1    1   -6169.306374389   -6169.306374389      0.161485070      0.039836516
  2    2   -6169.791301888      -0.484927499      0.061139804      0.017591252
  3    3   -6169.747694463       0.043607425      0.055408096      0.025421474
  4    4   -6169.878210014      -0.130515550      0.017439698      0.006673351
  5    5   -6169.888565676      -0.010355662      0.003356739      0.001448374
  6    6   -6169.888980991      -0.000415315      0.001161040      0.000311985
  7    7   -6169.889006110      -0.000025119      0.000674240      0.000217031
  8    8   -6169.889013696      -0.000007586      0.000208345      0.000069770
  9    9   -6169.889014995      -0.000001299      0.000057789      0.000021609
 10   10   -6169.889015063      -0.000000068      0.000034944      0.000011911
 11   11   -6169.889015075      -0.000000012      0.000013343      0.000003255
 12   12   -6169.889015076      -0.000000001      0.000004112      0.000001070
------------------------------------------------------------------
  SCF Converged.
  DFT Energy =   -6169.889015076 a.u.                           ← SCF計算結果
  Exchange + Correlation energy =    -639.792183436 a.u.
  Number of electrons           =     870.001701482
------------------------------------------------------------------
```

SMASHアウトプット3

アウトプット半ば続き(B3LYP/6-31G*のエネルギー計算)

```
 ------------------------------------------------------------------------------
  Eigenvalues (Hartree)
 ------------------------------------------------------------------------------
   Alpha Occupied:   -88.87820  -88.87209  -19.20781  -19.17578  -19.16789
   Alpha Occupied:   -19.16537  -19.16416  -19.15864  -19.15125  -19.15037
   Alpha Occupied:   -19.14017  -19.13100  -19.12886  -19.12435  -19.12419
```

← 軌道エネルギー

アウトプット最後(B3LYP/6-31G*のエネルギー計算)

```
 ---------------------------------------
     Mulliken Population Analysis
    Atom     Population     Charge
 ---------------------------------------
    1  N       7.735445    -0.735445
    2  C       6.053215    -0.053215
(省略)
  218  O       8.612501    -0.612501
  219  H       0.579850     0.420150
 ---------------------------------------
    Total                  -0.000000
 ---------------------------------------
```

← Mulliken電荷

```
 -----------------------------------------------------
                Dipole Momemt
       X         Y         Z       Total
 -----------------------------------------------------
     5.9228    -2.1116    -8.7870    10.8051
 -----------------------------------------------------
```

← 双極子モーメント

```
(省略)
 Total CPU time :     8418.3 seconds
 Total Wall time:      724.4 seconds
              ( 0 days  0 hours 12 minutes  4.4 seconds)
 The job finished at Tue Sep 11 17:47:41 2018
 Used memory :    379 MB
 Your calculation finished with  0 warning(s).
```

← 計算実行時間

← 計算終了時刻

← ノード当たりメモリ使用量

← 計算中の警告数

可視化

- smash/visual/ディレクトリにチェックポイントファイルからvtkファイルとGaussian cubeファイルを作成するソースコードがある
- visualディレクトリでmakeを実行すると、vtk-generatorとcube-generatorファイルが作られる
- vtkファイルは、フリーソフトParaViewで表示可能
- 実行方法 (MOのタイプ MO:alpha MO、MOB: beta MO)
  (MO番号 数値、HOMO、LUMO)

```
./cube-generator (チェックポイントファイル) (cubeファイル) (MOのタイプ) (MO番号)
```

```
./vtk-generator (チェックポイントファイル) (vtkファイル) (MOのタイプ) (MO番号)
```

- 例) methaneの計算を行ってmethane.chkを作った後、HOMOのcubeファイルを作る場合:

```
./cube-generator methane.chk methane-homo.cube MO HOMO
```

SMASH演習に関する情報

- ff01.j-focus.jpに講習会用アカウントでログインしてください。

- システムごとに最適な実行ファイルを用意しています。

システム	ディレクトリ
SystemA	/home1/share/smash/smash-2.2.0/intel_sse4/bin/
SystemD	/home1/share/smash/smash-2.2.0/intel_avx/bin/
SystemF, SystemH	/home1/share/smash/smash-2.2.0/intel_avx2/bin/

- /home1/share/smash/smash-2.2.0/example/にサンプルインプットとアウトプット、ジョブ投入スクリプトsample_a.sh、sample_d.sh、sample_e.sh･･･があります。

演習内容

- SMASH演習1-7は(実習・基礎編)
- SMASH演習8-13は(実習・応用編)

SMASH演習1

- /home1/share/smash/smash-2.2.0/example/sample_a.shと/home1/share/smash/smash-2.2.0/example/b3lyp-energy.inpを自分のディレクトリにコピーしてください。

```
cp /home1/share/smash/smash-2.2.0/example/sample_a.sh ~
cp /home1/share/smash/smash-2.2.0/example/b3lyp-energy.inp ~
```

- このsample_a.shは同じディレクトリにあるb3lyp-opt.inpの計算をSystemA 2ノードで実行するスクリプトです。結果はb3lyp-opt.out_(ジョブID番号))に出力されます。
- sample_a.sh内にあるb3lyp-optを全てb3lyp-energyに変更してください。
- sample_a.shの3行目を次のように変更してください。

```
#SBATCH -p a001h_lec
```

- sample_a.shをSystemAにジョブ投入してください。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで確認してください。

SMASH演習2

- sample._a.shの4,5行目

```
#SBATCH -N 2
#SBATCH -n 2
```

- を

```
#SBATCH -N 4
#SBATCH -n 4
```

- に変更して、再度SystemAにジョブ投入してください。この変更で、2ノード利用から、4ノード利用に変更になります。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで、エネルギー値や計算実行時間を確認してください。

SMASH演習3

- b3lyp-energy.inpの1行目の基底関数をcc-pVDZから

```
basis=cc-pvdz
```

STO-3Gに

```
basis=sto-3g
```

変更して、SystemAにジョブ投入してください。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで、基底関数、計算条件、エネルギー値や計算実行時間を確認してください。

SMASH演習4

- 次の内容をファイル名water.inpで保存してください。

```
job  runtype=energy   method=hartree-fock  basis=6-31G*
geom
O   0.0   0.0   0.0
H   1.0   0.0   0.0
H   0.0   1.0   0.0
```

- sample_a.shの最後の行を次のように変更してください。

```
mpirun -np ${SLURM_NTASKS} $EXE < water.inp
```

- SystemAにジョブ投入してください。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで確認してください。

SMASH演習5

- water.inpを次のように変更してください。

```
job  runtype=opt  method=hartree-fock  basis=6-31G*
geom
O   0.0   0.0   0.0
H   1.0   0.0   0.0
H   0.0   1.0   0.0
```

- SystemAにジョブ投入してください。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで確認してください。job runtype=optにすることで、エネルギー計算から構造最適化計算に代わります。最適化構造は、アウトプットの最後に出力されます。

SMASH演習6

- water.inpを次のように変更してください。

```
job  runtype=opt  method=mp2  basis=6-31G*
geom
O   0.0   0.0   0.0
H   1.0   0.0   0.0
H   0.0   1.0   0.0
```

- SystemAにジョブ投入してください。

```
sbatch sample_a.sh
```

- 得られた結果をviやemacsなどで確認してください。method=mp2にすることで、計算方法がHartree-Fock法からMP2法に代わります。

SMASH演習7

- /home1/share/smash/smash-2.2.0/example/sample_h.shを自分のディレクトリにコピーしてください。

```
cp /home1/share/smash/smash-2.2.0/example/sample_h.sh ~
```

- sample_h.shのb3lyp-optを全てb3lyp-energyに変更してください。
- sample_h.shの3行目を次のように変更してください。

```
#SBATCH -p h001h_lec
```

- b3lyp-energy.inpの基底関数をcc-pVDZに戻してください。

```
basis=cc-pvdz
```

- SystemHにジョブ投入してください。

```
sbatch sample_h.sh
```

- 得られた結果をviやemacsなどで、SystemAと同じノード数で実行した場合とどの程度実行時間が違うか確認してください。

SMASH演習8

- /home1/share/smash/smash-2.2.0/example/large-memory.inpを自分のディレクトリにコピーしてください。MP2エネルギー計算です。

```
cp /home1/share/smash/smash-2.2.0/example/large-memory.inp ~
```

- large-memory.inpの基底関数を6-31Gに変更してください。

```
basis=6-31G
```

- sample_h.shの最後の行を次のように変更してください。

```
mpirun -np ${SLURM_NTASKS} $EXE < large-memory.inp
```

- SystemHにジョブ投入してください。

```
sbatch sample_h.sh
```

- 得られた結果をviやemacsなどで、MP2計算条件を確認してください。

SMASH演習9

- 演習8で変更したlarge-memory.inpのノード(プロセス)当たりのメモリ使用量を20GB、30GBに変更してください。

```
memory=20G(もしくは30GB)
```

- SystemHにジョブ投入してください。

```
sbatch sample_h.sh
```

- 得られた結果をviやemacsなどで、MP2計算条件や実行時間を確認してください。特に、MP2計算条件の後のMultiple passの数を見てください。Multiple passが少ないほど、実行時間は短くなります。

```
-----------------------------------------------
  MP2 calculation
-----------------------------------------------
    Ncore=  62,   Nvfz=   0
-----------------------------------------------
    Number of basis functions         =  660
    Number of basis shells            =  412
    Number of correlated occupied MOs =  164
    Number of active virtual MOs      =  434
-----------------------------------------------
== Multiple pass calculation ==
   Number of passes :    3
```

SMASH演習10

- Checkpointファイルの作り方・使い方、及び基底関数を少しずつ上げる演習です。
- /home1/share/smash/smash-2.2.0/example/check-generation.inp
  /home1/share/smash/smash-2.2.0/example/check-read.inpを自分のディレクトリにコピーしてください。

```
cp /home1/share/smash/smash-2.2.0/example/check-generation.inp ~
cp /home1/share/smash/smash-2.2.0/example/check-read.inp ~
```

- sample_h.shの最後の行を次のように変更して、SystemHに投入してください。

```
mpirun -np ${SLURM_NTASKS} $EXE < check-generation.inp
```

- sample_h.shの最後の行を次のように変更して、SystemHに投入してください。

```
mpirun -np ${SLURM_NTASKS} $EXE < check-read.inp
```

- 得られた結果をviやemacsなどで確認してください。さらに、checkpointファイル(この演習ではanthracene.chk)が作成されていることを確認してください。

SMASH演習11

- sample_h.shの最後の行

```
mpirun -np ${SLURM_NTASKS} $EXE < check-read.inp
```

を次のように変更してください。1行で書いてください。

```
/home1/share/smash/smash-2.2.0/intel_avx2/visual/cube-generator
anthracene.chk anthracene-homo.cube MO HOMO
```

- SystemHにジョブ投入すると、anthracene-homo.cubeが作成されます。このファイルを端末にコピーして、GaussViewで開いてください。HOMOの分子軌道図が表示されます。

- sample_h.shの最後の行を次のように変更して、同様の操作を行ってください。LUMOの分子軌道図が表示されます。

```
/home1/share/smash/smash-2.2.0/intel_avx2/visual/cube-generator
anthracene.chk anthracene-lumo.cube MO LUMO
```

SMASH演習12

- sample_h.shの最後の行

```
/home1/share/smash/smash-2.2.0/intel_avx2/visual/cube-generator
anthracene.chk anthracene-homo.cube MO HOMO
```

を次のように変更してください。1行で書いてください。

```
/home1/share/smash/smash-2.2.0/intel_avx2/visual/vtk-generator
anthracene.chk anthracene-homo.vtk MO HOMO
```

- SystemHにジョブ投入すると、anthracene-homo.vtkが作成されます。このファイルと/home1/share/smash/smash-2.2.0/intel_avx2/visual/mo-view.pvsmを端末にコピーしてください。ParaViewを起動した後、次の3つの手順でHOMOを表示してください。
  1. File -> Load State
  2. mo-view.pvsmを選択してOK
  3. anthracene-homo.vtkを選択してOK

SMASH演習13

- 計算方法と基底関数を少しずつ上げる演習です。
- home1/share/smash/smash-2.2.0/example/b3lyp-energy.inpを自分のディレクトリにコピーしてください。さらに、そのファイルをhf-energy.inp名でコピーしてください。

```
cp /home1/share/smash/smash-2.2.0/example/b3lyp-energy.inp ~
cp b3lyp-energy.inp  hf-energy.inp
```

- hf-energy.inpの初めを次のように変更してください。

```
job runtype=energy method=hf  basis=sto-3g
control  check=energy.chk
geom
```

- hf-energy.inpの計算をジョブ投入してください。どのシステムでも構いません。
- ファイルenergy.chkが作られているのを確認してください。

SMASH演習14

- 前ページの続きです。
- b3lyp-energy.inpの初めを次のように変更してください。

```
job runtype=energy method=b3lyp basis=cc-pvdz
control   check=energy.chk   guess=check
geom
```

- b3lyp-energy.inpの計算をジョブ投入してください。どのシステムでも構いません。
- アウトプットファイルの基底関数の出力直後の初期軌道計算の内容を確認してください。

```
--------------------
  Guess calculation
--------------------
  Guess MOs are read from checkpoint file and projected.
```

- 開殻系や大きな基底関数のDFT計算では、少しずつ基底関数や計算方法を上げると収束しやすくなります。