2.9 生体分子粗視化シミュレータ CafeMol の FX100 での性能評価

理化学研究所　情報基盤センター
検崎博生

2.9.1　はじめに

CafeMol は生体分子の粗視化シミュレータで、タンパク質や核酸のダイナミクスを調べることができる。生体分子である、タンパク質、核酸、脂質は分子レベルの生体機能を担うが、多くの生体分子は構造変化することによってさまざまな機能を実現する。粗視化モデルでは、全原子シミュレーションでは再現することが難しいような大規模な構造変化を簡単に計算することができる。また、CafeMol の特徴はタンパク質や核酸の構造変化のシミュレーションを簡単に行えるところにある。

CafeMol で用いる粗視化モデルでは全原子モデルに比べて粒子数が 10 分の 1 程度に減り、水分子もあらわに扱わないので粒子数が少数ですむ。結果として、計算量をかなり減らすことができるのであるが、これは並列性能を出しにくいということにも繋がっている。一方、今後の計算機の発展を考えると、ノード内のコアの数がますます増えると予想されるので、CafeMol でも 1 ノード内での並列性能を上げることが重大な問題になっている。

2.9.2　プログラム概要

分子動力学シミュレーションでは、分子の動きを時間積分することにより系を時間発展させるわけであるが、時間積分のステップを進める際の分子間に働く力の計算が最も時間が掛かる部分となっている。CafeMol では力の計算にネイバリングリスト方式を用いて計算量を減らしており、MPI と OpenMPI によるハイブリッド並列化を行っている。

粗視化された粒子間に働く相互作用はさまざまなものがあり、相互作用毎に異なるチューニングが必要になってくる。タンパク質の粗視化モデルでは比較的近距離の相互作用だけなのでステップ当たりの計算量は少なく、核酸では電荷の寄与が大きく遠距離の静電相互作用が必須で時間ステップ当たりの計算量は比較的多くなる。ここではタンパク質/DNA 複合体であるヌクレオソームを対象に性能測定を行う。静電相互作用はデバイヒュッケル理論に基づく近似を用いるため、イオン濃度が静電相互作用の到達距離を決める重要なパラメータとなっているが、ここでは生理条件と大きく変わらない 100mM で測定を行う。なお、ヌクレオソームは粗視化粒子数が 2,000 程度の系で、CafeMol を使って行われている粗視化シミュレーションでは中程度の大きさの系となっている。

2.9.3　測定環境

性能測定には、理化学研究所の RICC システム(Intel Xeon)、SCLS 計算機システム(FX10)、HOKUSAI システム(FX100)を使用した。それぞれの環境における CPU とコンパイラとデフォルトで用いたコンパイルオプションは以下の通りである。

- Intel Xeon：Intel Xeon X5570(2.93GHz)、インテルコンパイラ、-O3

- FX10：SPARC64™IXfx(1.65GHz)、富士通コンパイラ、-Kfast
- FX100：SPARC64™XIfx(1.975H)、富士通コンパイラ、-Kfast

2.9.4　測定結果

性能測定は、ヌクレオソームの 10,000step の計算時間を測定した。まず、MPI 並列は用いずに、OpenMP によるスレッド並列を用いた測定を行った。Xeon の 1 ノード 8 コアでの計算時間は 8.33 秒であり、FX10 での 1 ノード 16 コアでの計算時間は 15.5 秒、FX100 での 1 ノード 16 コアで 9.9 秒、32 コアで 8.19 秒であった。FX100 は FX10 と比べると同じ 16 コアで 1.5 倍程度、倍の 32 コアでは 1.9 倍程度速くなっており、FLOPS 比には足りないが悪くない性能向上となっている。ただし、Xeon と比べると 4 倍の 32 コアでわずかに速くなっている程度なので、このままでは物足りない数字となっている。

次に、FX100 で 1 ノード 32 コアを固定して、MPI 並列と OpenMP のハイブリッド並列の測定を行った。2MPI/16 スレッド並列では 6.93 秒と 1MPI/32 スレッド並列の 8.19 秒よりも速くなった。4MPI/8 スレッド並列と 8MPI/4 スレッド並列でも 2MPI/16 スレッド並列と同程度速くなったので、以下では 2MPI/16 スレッド並列で測定を行った。

最適化オプションの効果を調べるためにいくつかの最適化オプションを試した結果、”-Kvisimpact”を使うと 6.06 秒まで速くなり、自動並列化”-Kparallel”の効果であると考えられる。それ以外の最適化オプションは効果がなかった。

次に、チューニングを行うために計算時間の内訳を見ると、力の計算に 2.96 秒、乱数に 0.61 秒、ネイバリングリストの計算に 1.19 秒掛かっていた。全体の半分程度の時間が掛かっている力の計算については、以下で示すようにチューニングを試みた。なお、乱数については、組み込み関数を使うことで劇的な速度向上が見込まれる。

力の計算の内訳を見ると、もともと計算量が圧倒的に多い静電相互作用については SIMD 化によるチューニングが既にされていることもあり 0.38 秒しか掛かっておらず、SIMD 化されていない他の力の計算に時間が掛かっていた。

一般的に、分子動力学計算の力の計算では作用反作用の法則が成り立つため、SIMD 化が困難になる事情がある。例えば、2 体の相互作用を計算する際には 2 重ループを作って計算するが、外側のループ変数に依存する粒子 1 と内側のループ変数に依存する粒子 2 の力を計算した際、粒子 1 に掛かる力だけではなく粒子 2 に掛かる力も同時に計算できるのでこれを保存する必要があるが、そのままでは SIMD 化ができなくなる。ここで、粒子 2 に掛かる力を保存せずに捨てて、2 倍の量の計算を行うように変更するのが SIMD 化の 1 つ方法である。この方法では計算量が 2 倍になるというデメリットがあるが、静電相互作用の計算ではかなり性能が向上する。ところが、静電相互作用以外の相互作用では内側ループ長が短いためか、同程度か遅くなってしまう。3 体以上の相互作用では、この方法はさらに難しくなる。他にも SIMD 化するためのいくつかの方法で試したが、効果がないかむしろ遅くなった。

ここでは計算された力の保存が原因で SIMD 化が困難だったわけだが、アトミック操作などにより SIMD 化が可能になれば、性能が向上する可能性があると予想する。

2.9.5　まとめ

CafeMol を FX100 上で性能測定を行い、ハイブリッド並列におけるスレッド並列数やコンパイルオプションの影響を調べ、最後にチューニングを試みた。FX100 では FX10 に比べると性能が予想よりも向上していたが、Xeon と比べると少し物足りないものとなった。MPI と OpenMP のハイブリッド並列を行う際には、スレッド並列数 32 は性能が悪くなり、4,8,16 スレッド並列の方がいい性能が得られた。また、自動並列化"parallel"により少し性能が向上することを確認した。チューニングに関しては、静電相互作用以外の相互作用の力の計算では SIMD 化が困難もしくは性能向上ができなった。最後に、アトミック操作がハードウェアもしくはソフトウェアで対応すればより SIMD 化が容易に行えるようになり、チューニングに手数をあまり掛けなくても多くのアプリケーションで性能向上が実現できるのではないかと考える。

生体分子粗視化シミュレータCafeMol FX100でのチューニング

検崎博生
理研 情報基盤センター

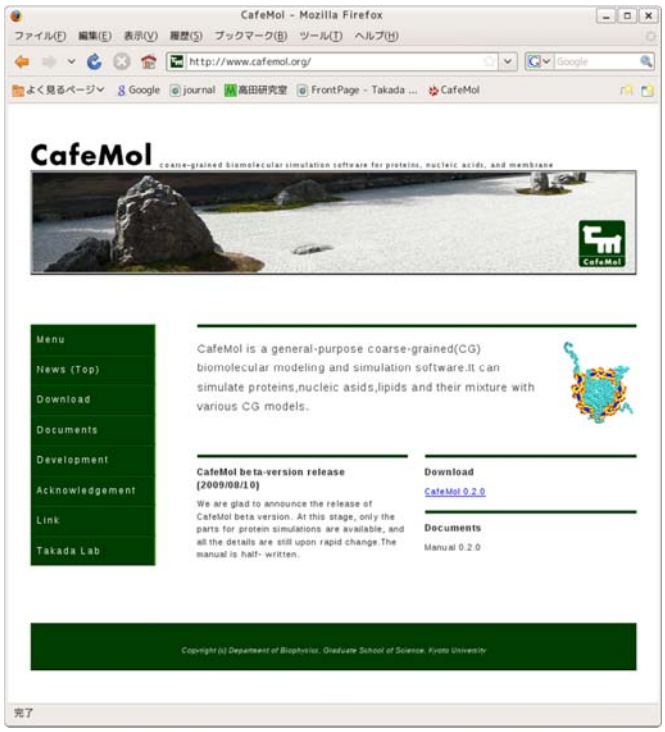
CafeMol (www.cafemol.org)

- Features are;
  - –Various CG models
    - –protein/DNA/RNA
    - –multiple basin model
    - –accurate CG model
  - –Simulating protein-at-work
    - –“switching”
- Under development
  - lipid
- Developer

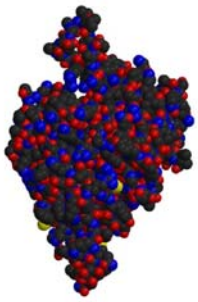
検崎博生、古賀信康、藤原慎司、堀直人、金田亮、李 文飛、岡崎圭一、姚 新秋、寺川剛、伊藤真志保、高田彰二

京都大学高田研究室

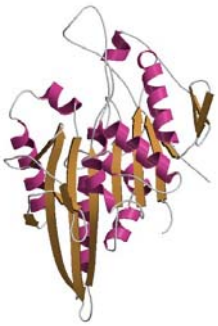
CafeMol 2.1 (2013/7)
source & manual released



Coarse-grained simulation



coarse-grained



Reducing the number of particles to 1/100

- 1 amino acid residue →1 particle ($C_\alpha$atom)
- No water molecule
  (included in potential energy)

Protein folding and
functional conformational change

10^6 times speed up

- Reducing the number of particles
- Enlarge time step
- Low friction

AA: 100PC 1week → CG: 1PC 1minute!!

計算規模と並列化

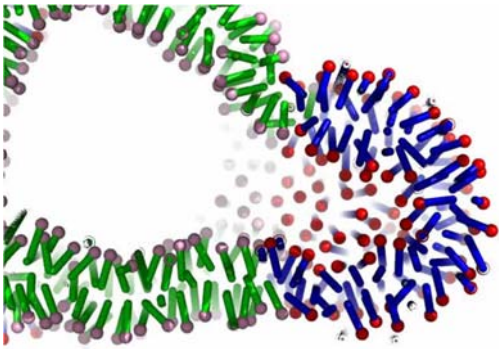
- 計算規模
  - 数十から数万粒子数程度
- 測定対象
  - ヌクレオソーム1個の系：1,854粒子
    - タンパク質/DNA複合体
- 並列化
  - 系の時間発展をMPIとOpenMPでハイブリッド並列化
    - 数十並列程度の並列化
    - 力の計算はネイバリングリスト方式
  - レプリカ交換によるMPI並列
    - 通信量が少ないので大規模並列化
    - 平衡量を求める計算に使える

ネイバリングリストの例 (2体の相互作用、i<jだけをリストアップ)

i	1				2		3			4		5			6		7		8
j	2	4	7	9	5	8	4	5	9	7	8	6	7	9	8	9	8	9	9

ネイバリングリスト

- 力場はネイバリングリスト方式で計算
  - 100stepに1回程度計算
  - ネイバリングリストはノード毎に分割
    - 最も大きな配列(粒子数x100～10,000)
  - 相互作用の到達範囲毎にネイバリングリストを作成
    - ローカルな相互作用
      - ボンド長、ボンド角、二面角
      - あらかじめネイバリングリスト作成しておく
    - 近距離相互作用(<20A)
      - 郷ポテンシャル、排除体積、塩基対
    - 中距離相互作用(20-50A)
      - 疎水相互作用
      - 多体のため複雑なネイバリングリストに
    - 遠距離相互作用(>50A)
      - 静電相互作用
      - 塩濃度により相互作用距離が変化



RICCでの性能測定　Intel Xeon5570(2.93Hz) nucleosome (10^4step, 100mM)

	1threads	%	2threads	%	4threads	%	8threads	%
force	41.68	88.3	21.79	85.8	11.18	81.5	6.23	74.8
_force(comm)	0.16	0.3	0.16	0.6	0.26	1.9	0.54	6.5
_force(local)	4.85	10.3	2.53	10.0	1.30	9.4	0.69	8.2
_force(go)	1.26	2.7	0.65	2.6	0.32	2.4	0.17	2.1
_force(pnl)	14.63	31.0	7.11	28.0	3.54	25.8	1.77	21.3
_force(ele)	20.67	43.8	10.87	42.8	5.41	39.4	2.72	32.7
random	2.94	6.2	1.61	6.3	0.88	6.4	0.55	6.6
_random(comm)	0.04	0.1	0.04	0.2	0.05	0.4	0.07	0.8
neighbor	1.30	2.8	1.12	4.4	0.87	6.3	0.74	8.8
_neighbor(comm)	0.00	0.0	0.00	0.0	0.00	0.0	0.00	0.0
_neighbor(pnl)	1.14	2.4	0.97	3.8	0.71	5.2	0.58	6.9
_neighbor(ele)	0.11	0.2	0.11	0.4	0.11	0.8	0.11	1.3
_neighbor(solv)	0.05	0.1	0.05	0.2	0.05	0.4	0.05	0.6
update	0.43	0.9	0.35	1.4	0.35	2.6	0.41	4.9
copyxyz	0.01	0.0	0.00	0.0	0.00	0.0	0.00	0.0
energy	0.38	0.8	0.20	0.8	0.11	0.8	0.07	0.8
_energy(comm)	0.00	0.0	0.00	0.0	0.00	0.0	0.01	0.1
output	0.26	0.6	0.25	1.0	0.25	1.8	0.25	3.0
radiusg_rmsd	0.02	0.0	0.02	0.1	0.02	0.1	0.02	0.2
others	0.05	0.1	0.03	0.1	0.03	0.2	0.03	0.3
ope	47.03	99.6	25.19	99.2	13.41	97.7	7.72	92.7
comm	0.20	0.4	0.21	0.8	0.31	2.3	0.61	7.3
main_loop	47.23	100.0	25.40	100.0	13.72	100.0	8.33	100.0

FX10での性能測定(-Kfast)
nucleosome (10^4step, 100mM)

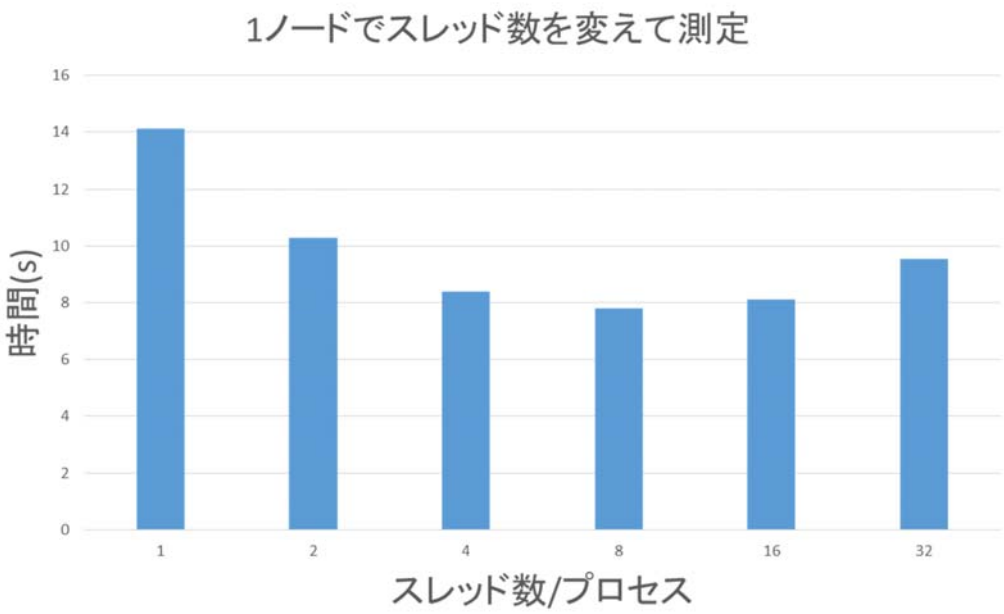
	1threads	%	2threads	%	4threads	%	8threads	%	16threads	%
force	86.64	80.8	46.35	75.8	23.45	68.9	12.9	61.3	8.5	54.8
_force(comm)	0.20	0.2	0.33	0.5	0.57	1.7	1.1	5.1	2.1	13.6
_force(local)	11.47	10.7	5.82	9.5	3.00	8.8	1.6	7.5	0.9	5.7
_force(go)	5.50	5.1	2.79	4.6	1.46	4.3	0.7	3.5	0.4	2.5
_force(pnl)	50.36	47.0	24.77	40.5	12.23	35.9	6.2	29.2	3.1	20.1
_force(ele)	18.81	17.5	12.34	20.2	5.89	17.3	3.1	14.5	1.7	11.0
random	7.90	7.4	4.24	6.9	2.37	7.0	1.4	6.7	1.0	6.2
_random(comm)	0.14	0.1	0.14	0.2	0.14	0.4	0.1	0.7	0.1	0.9
neighbor	9.46	8.8	7.76	12.7	5.68	16.7	4.3	20.5	3.7	23.7
_neighbor(comm)	0.00	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.0	0.0
_neighbor(pnl)	8.39	7.8	6.68	10.9	4.59	13.5	3.2	15.4	2.6	16.7
_neighbor(ele)	1.50	1.4	1.52	2.5	1.53	4.5	1.5	7.1	1.5	9.7
_neighbor(solv)	0.30	0.3	0.29	0.5	0.30	0.9	0.3	1.4	0.3	1.9
update	0.83	0.8	0.82	1.3	0.83	2.4	0.8	3.9	0.8	5.4
copyxyz	0.14	0.1	0.14	0.2	0.14	0.4	0.1	0.7	0.1	0.9
energy	0.98	0.9	0.55	0.9	0.30	0.9	0.2	0.8	0.1	0.8
_energy(comm)	0.00	0.0	0.00	0.0	0.01	0.0	0.0	0.1	0.0	0.2
output	0.68	0.6	0.68	1.1	0.68	2.0	0.7	3.2	0.7	4.4
radiusg_rmsd	0.04	0.0	0.04	0.1	0.04	0.1	0.0	0.2	0.0	0.3
others	0.10	0.1	0.10	0.2	0.10	0.3	0.1	0.5	0.1	0.6
ope	106.91	99.7	60.67	99.2	33.34	97.9	19.8	94.2	13.2	85.3
comm	0.33	0.3	0.48	0.8	0.72	2.1	1.2	5.8	2.3	14.7
main_loop	107.25	100.0	61.15	100.0	34.06	100.0	21.1	100.0	15.5	100.0

FX100での予備的な性能測定(-Kfast)
nucleosome (10^4step, 100mM)

	1threads	%	2threads	%	4threads	%	8threads	%	16threads	%	32threads	%
force	58.52	79.0	29.78	73.1	15.28	66.6	8.2	56.9	4.8	48.9	3.70	45.2
_force(comm)	0.11	0.2	0.16	0.4	0.25	1.1	0.4	3.0	0.8	7.8	1.45	17.7
_force(local)	8.90	12.0	4.51	11.1	2.30	10.0	1.2	8.4	0.6	6.5	0.36	4.5
_force(go)	4.20	5.7	2.14	5.3	1.10	4.8	0.6	3.9	0.3	3.0	0.16	1.9
_force(pnl)	36.06	48.7	17.41	42.7	8.71	38.0	4.4	30.5	2.2	22.4	1.14	13.9
_force(ele)	9.02	12.2	5.33	13.1	2.69	11.7	1.4	9.5	0.7	6.9	0.36	4.4
random	6.23	8.4	3.23	7.9	1.73	7.6	1.4	9.7	0.8	8.1	0.50	6.2
_random(comm)	0.06	0.1	0.06	0.1	0.06	0.3	0.1	0.4	0.1	0.7	0.06	0.7
neighbor	7.27	9.8	5.95	14.6	4.31	18.8	3.3	22.7	2.7	27.6	2.38	29.0
_neighbor(comm)	0.00	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0
_neighbor(pnl)	6.54	8.8	5.21	12.8	3.58	15.6	2.5	17.7	2.0	20.4	1.64	20.1
_neighbor(ele)	0.94	1.3	0.96	2.3	0.94	4.1	0.9	6.5	0.9	9.3	0.95	11.6
_neighbor(solv)	0.24	0.3	0.24	0.6	0.24	1.0	0.2	1.7	0.2	2.4	0.24	2.9
update	0.51	0.7	0.51	1.3	0.51	2.2	0.5	3.6	0.5	5.4	0.63	7.6
copyxyz	0.08	0.1	0.08	0.2	0.08	0.4	0.1	0.6	0.1	0.9	0.09	1.1
energy	0.68	0.9	0.38	0.9	0.20	0.9	0.1	0.8	0.1	0.8	0.07	0.9
_energy(comm)	0.00	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.0	0.1	0.02	0.2
output	0.35	0.5	0.35	0.9	0.35	1.5	0.4	2.4	0.4	3.6	0.35	4.3
radiusg_rmsd	0.02	0.0	0.02	0.0	0.02	0.1	0.0	0.1	0.0	0.2	0.02	0.2
others	0.08	0.1	0.08	0.2	0.08	0.3	0.1	0.5	0.1	0.8	0.08	0.9
ope	73.92	99.8	40.52	99.5	22.63	98.6	13.9	96.6	9.0	91.5	6.66	81.4
comm	0.17	0.2	0.22	0.6	0.31	1.4	0.5	3.5	0.8	8.5	1.52	18.6
main_loop	74.10	100.0	40.75	100.0	22.94	100.0	14.4	100.0	9.9	100.0	8.19	100.0

FX100でスレッド数を変えた時の性能測定(-Kfast) nucleosome (10^4step, 100mM)

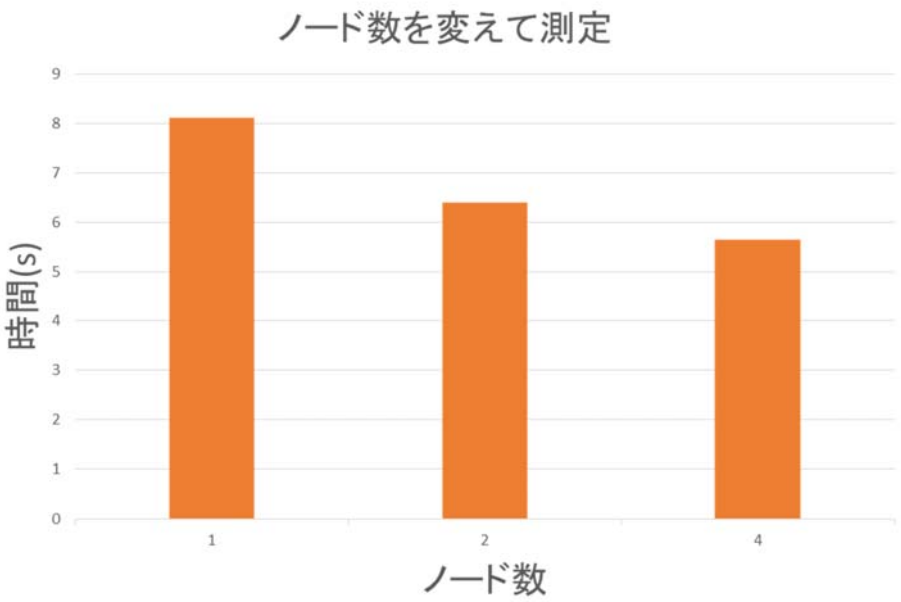
ノード内はMPI(32/nプロセス) x openMP(nスレッド)並列



スレッド数が4,8,16のときに高い性能

FX100でノード数を変えた時の性能測定(-Kfast) nucleosome (10^4step, 100mM)

ノード内はMPI(2プロセス) x openMP(16スレッド)並列



1ノードを超える並列性能は低い

FX100で最適化オプションを変えた時の性能測定 nucleosome (10^4step, 100mM)

ノード内はMPI(2プロセス) x openMP(16スレッド)並列

京で推奨されていた最適化オプションを使用(opt_kei)

-V –Kvisimpact,ocl,preex,array_private,auto,openmp,optmsg=2,simd=2 -Qt

	fast	%	opt_kei	%
force	3.44	49.59	2.96	48.87
_force(comm)	1.10	15.92	0.61	10.00
_force(local)	0.37	5.35	0.37	6.10
_force(go)	0.16	2.31	0.16	2.65
_force(pnl)	1.14	16.45	1.14	18.89
_force(ele)	0.37	5.38	0.38	6.19
random	0.62	8.89	0.61	10.11
_random(comm)	0.06	0.93	0.06	1.06
neighbor	1.39	20.07	1.19	19.58
_neighbor(comm)	0.00	0.00	0.00	0.00
_neighbor(pnl)	1.02	14.67	1.00	16.44
_neighbor(ele)	0.47	6.76	0.09	1.52
_neighbor(solv)	0.12	1.74	0.12	2.06
update	0.53	7.71	0.33	5.38
copyxyz	0.09	1.26	0.09	1.50
energy	0.06	0.94	0.06	0.95
_energy(comm)	0.02	0.22	0.01	0.15
output	0.34	4.86	0.35	5.71
radiusg_rmsd	0.02	0.24	0.03	0.50
others	0.08	1.10	0.08	1.26
ope	5.74	82.93	5.38	88.79
comm	1.18	17.07	0.68	11.21
main_loop	6.93	100.00	6.06	100.00

効果があったオプションは、fast→visimpact、つまりparallelの効果

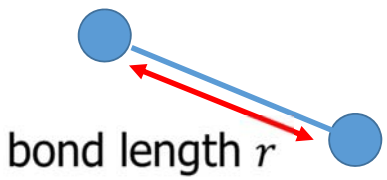
チューニングの方針

- force
  - eleはすでにSIMD化
  - localは、2-4体の相互作用
    - ボンド長:i,i+1
    - ボンド角:i,i+1,i+2
    - 二面角:i,i+1,i+2,i+3
  - goとpnlは基本2体だが、ネイバリングリストから変更する必要がある?
    - pnlはネイバリングリスト自体は1つだがさまざまな力で分割して利用
- random
  - 組み込み関数を使えば速くなる見込み

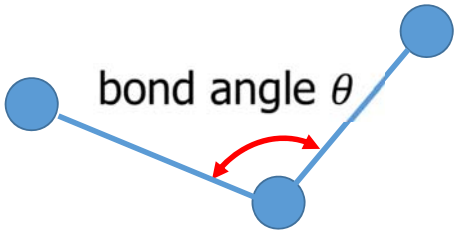
	opt_kei	%
force	2.96	48.87
_force(comm)	0.61	10.00
_force(local)	0.37	6.10
_force(go)	0.16	2.65
_force(pnl)	1.14	18.89
_force(ele)	0.38	6.19
random	0.61	10.11
_random(comm)	0.06	1.06
neighbor	1.19	19.58
_neighbor(comm)	0.00	0.00
_neighbor(pnl)	1.00	16.44
_neighbor(ele)	0.09	1.52
_neighbor(solv)	0.12	2.06
update	0.33	5.38
copyxyz	0.09	1.50
energy	0.06	0.95
_energy(comm)	0.01	0.15
output	0.35	5.71
radiusg_rmsd	0.03	0.50
others	0.08	1.26
ope	5.38	88.79
comm	0.68	11.21
main_loop	6.06	100.00

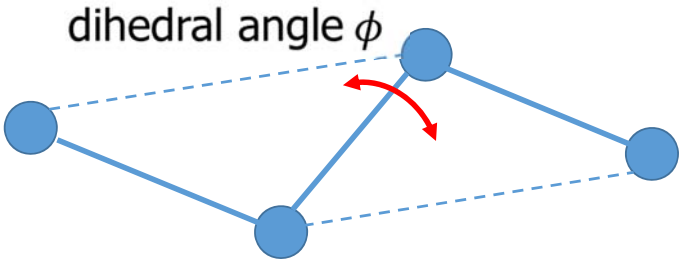
ローカルな力のチューニング

$$V_{local} = K_b \sum_i \left(r_{i,i+1} - r_{0i,i+1}\right)^2 + K_\theta \sum_i \left(\theta_i - \theta_{0i}\right)^2$$
$$+ K_\phi^1 \sum_i \left(1 - \cos\left(\phi_i - \phi_{0i}\right)\right) + K_\phi^3 \sum_i \left(1 - \cos 3\left(\phi_i - \phi_{0i}\right)\right)$$

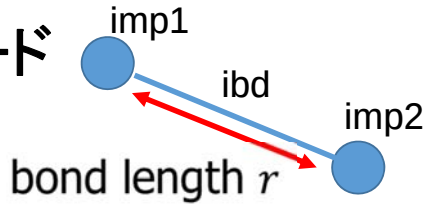


ボンド長：2体
ボンド角：3体
二面角：4体



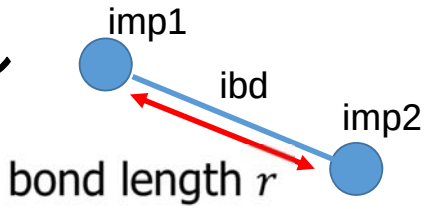


ボンド長の力の計算のソースコード




```
!$omp do private(imp1,imp2,v21,dist,ddist,ddist2,for, &
!$omp&            force,efull,iunit,junit,isys,istat)
  do ibd = ksta, kend
     imp1 = ibd2mp(1, ibd)
     imp2 = ibd2mp(2, ibd)
     v21(1:3) = xyz_mp_rep(1:3, imp2,irep) - xyz_mp_rep(1:3, imp1,irep)
     dist = sqrt(v21(1)**2 + v21(2)**2 + v21(3)**2)
     ddist = dist - bd_nat(ibd)
     ddist2 = ddist**2
     for = (coef_bd(1, ibd) + 2.0e0_PREC * coef_bd(2, ibd) * ddist2) * &
          (-2.0e0_PREC * ddist / dist)
     force(1:3) = for * v21(1:3)
     force_mp(1:3, imp1) = force_mp(1:3, imp1) - force(1:3)
     force_mp(1:3, imp2) = force_mp(1:3, imp2) + force(1:3)
  end do
!$omp end do nowait
```

ボンド長の力の計算のlstファイル




```
                    <<< Loop-information Start >>>
                    <<<  [OPTIMIZATION]
                    <<<    FULL UNROLLING
                    <<< Loop-information  End >>>
 64   1   p   6v          force(1:3) = for * v21(1:3)
 65   1
 66   1   p   6s          force_mp(1:3, imp1) = force_mp(1:3, imp1) - force(1:3)
 67   1   p   6m          force_mp(1:3, imp2) = force_mp(1:3, imp2) + force(1:3)
 68   1   p   2v        end do
 69                  !$omp end do nowait
```

排除体積相互作用の力の計算のソースコード

```
!$omp do private(imp1,imp2,v21,dist2,&
!$omp&          roverdist2,roverdist4, &
!$omp&          roverdist8,roverdist14,dvdw_dr,for)
  do ipnl=ksta, kend
     imp1 = ipnl2mp(1, ipnl, irep)
     imp2 = ipnl2mp(2, ipnl, irep)
     v21(1:3) = xyz_mp_rep(1:3, imp2, irep) - xyz_mp_rep(1:3, imp1, irep)
     dist2 = v21(1)**2 + v21(2)**2 + v21(3)**2
     if(dist2 > cutoff2) cycle
     roverdist2 = cdist2 / dist2
     roverdist4 = roverdist2 * roverdist2
     roverdist8 = roverdist4 * roverdist4
     roverdist14 = roverdist2 * roverdist4 * roverdist8
     dvdw_dr = coef * roverdist14
     if(dvdw_dr > DE_MAX) dvdw_dr = DE_MAX
     for(1:3) = dvdw_dr * v21(1:3)
     force_mp(1:3, imp2) = force_mp(1:3, imp2) + for(1:3)
     force_mp(1:3, imp1) = force_mp(1:3, imp1) - for(1:3)
  end do
!$omp end do nowait
```

排除体積相互作用の力の計算のlstファイル

```
62    1
                <<< Loop-information Start >>>
                <<<  [OPTIMIZATION]
                <<<    FULL UNROLLING
                <<< Loop-information  End >>>
63    1   p   6m          for(1:3) = dvdw_dr * v21(1:3)
64    1   p   6s          force_mp(1:3, imp2) = force_mp(1:3, imp2) + for(1:3)
65    1   p   6m          force_mp(1:3, imp1) = force_mp(1:3, imp1) - for(1:3)
66    1   p   2v        end do
67                  !$omp end do nowait
```

まとめ

- FX100でのベンチマーク結果
  - 同じ16コアでFX10に比べて1.5倍程度性能向上
  - 4,8,16スレッド並列で性能が出た
  - 自動並列化オプションが効果的だった
- FX100でのチューニングの方針
  - 力の計算
    - 静電相互作用以外でSIMD化用のコード作成
  - 結果として、チューニングは上手くいかなかった
    - 変数iのループで引数iだけでなく引数jの配列に計算結果を保存するときにもSIMD化ができるようにならないだろうか？
    - 変数kのループで引数i, jの配列に計算結果を保存するときにもSIMD化ができるようにならないだろうか？