2.3.2. 半古典分子動力学計算を用いたコンピュータ性能の測定

上智大学　　南部 伸孝

1. 概要

古典力学を基に分子の運動（粒子の運動）を記述する分子動力学シミュレーション（Molecular Dynamics simulation）が生体関連の分野で頻繁に利用され，最近では分子機械の解明等までも利用されている。特にそこで活躍されている数値計算法として速度ベルレ（Velocity-Verlet）法があるが，エネルギーの誤差が比較的大きい。一方，分子機能自体が量子現象に起因する場合もある。その場合は，分子の運動を古典論ではなく，量子論あるいは半古典論に基づき数値的に求めなければならない。ところが，計算精度の低い速度ベルレ法では破たんするため，より高精度な積分法を用いる必要が出てくる。

本計測では，ベンチマークプログラムとして古典トラジェクトリを半古典論に基づく経路積分法により量子効果を取り入れるため，Gray らが提唱する高精度な 4 次のシンプレクティック積分法［Brewer, Hulme, Manolopoulos, *J. Chem. Phys.* **106**, 4832-4839 (1996)の Appendix を参照］を用いた方法で実施した。また，【補足資料】(1)に実際のコードの一部を添付する。

2. 測定

測定を実施した計算機の CPU 型番は以下の 9 種類である。Intel と AMD と IBM が入り乱れた順番となっているが，リリースされた年代に沿って順番を振っていることから，年代とともに見ていただきたい。測定は 3 回実施し，その平均値を値とした。

①AMD Athlon™ 64 Processor 3500+
②AMD Athlon™ 64 X2 Dual Core Processor 4400+
③Intel® Pentium® 4 3.4GHz
④Intel® Xeon® 5160 3.0GHz（富士通 Primergy, PG）
⑤IBM Power5 1.9GHz　（日立 SR11000 J1 ノード）
⑥Intel® Itanium2-p9000 1.6GHz（富士通 Primequest, PQ）
⑦Quad-Core AMD Opteron™ Processor 2346 HE @ 1.8GHz
⑧Intel® Core™2 Duo CPU E6850 @ 3.00GHz
⑨Dual-Core AMD Opteron™ Processor 2214 HE @ 2.2GHz

また，使用したコンパイラは以下の通りである。

1. 富士通製 Fujitsu Fortran Driver Version 2.0 P-id: T05097-03 (Sep. 5 2007 17:37:52)
2. 日立製最適化 FORTRAN90 V01-05
3. Intel® Fortran Compiler for Intel® EM64T-based applications, Version 9.1
4. The Portland Group, Inc. pgf90 6.0-4 64-bit target on x86-64 Linux

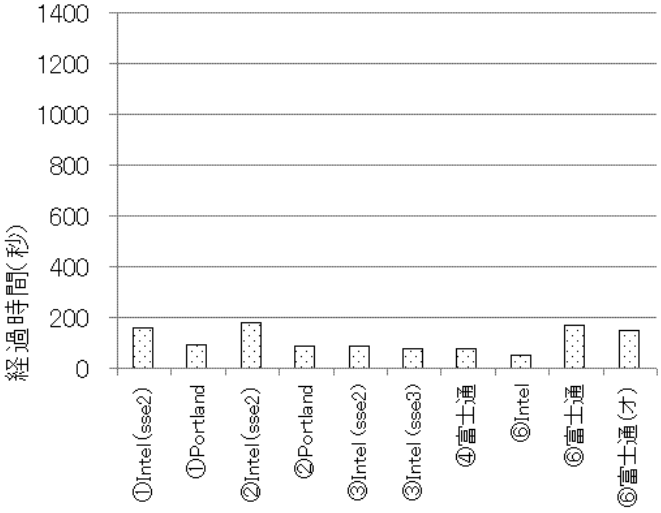

図 1. 非並列コンパイラの結果

3. 結果と考察

詳細なコンパイラオプション，計測経過時間等は【補足資料】(2)に列挙するので，それを参照されたい。ここでは，幾

つか特徴的に部分を取り上げて，結果を紹介する。まず，非並列コンパイルの結果を図 1 に示す。横軸は，使用したコンパイラのメーカーであり，例えば「⑥富士通(オ)」は富士通のコンパイルを使うが，オプションを最適化したものである。

具体的な結果であるが，非非並コンパイルと非並列実行の基，最も計算時間の短かったのは，Itanium2 の CPU 上で Intel コンパイラを用いた場合であった。50 秒弱であり，並列計算結果を含めても 2 番目である。②は AMD であるが，③の Intel との差異があまり見られない。但し，AMD では Portland 製を利用した方が良さそうである。一方，③と④はともに Intel の CPU であり混ぜて比較すると，型番が違うので明言はできないが，コンパイラの性能において富士通 vs. Intel は大差なしと思われる。

次に，それぞれのマシンを固定し，コンパイラ性能を比較する。図 2 は AMD Athlon の Dual Core の CPU である。東工大がみんなのスパコンとして導入した CPU より古い CPU であるが，結果の通り全く並列性能向上が見られない。その一方，Intel を除き，悪化も見られず，その性能を保持している。多分，オプションを選んでも全く並列化されなかったためだと思われる。

図 3 は Intel Xeon（別名：Nehalem コア）上での性能である。特に 1 ノード，4 コアまで共有メモリー型のマシンであることから，4 コアまで並列性能が期待できる。結果は，ご覧の通り，スレッド数が増えると悪化するのが分かる。そこで，最適オプションを富士通さんに選んで頂いくことにする。図 2 の結果と同様，自動並列を止める方向に動いていることがわかる。残念であるが，利用者からみた場合，悪化しなくなることも重要な要素なので，このコードでは，自動並列が期待できないと考えるべきかもしれない。

図 4 は日立 SR11000 上での性能である。もちろん，CPU は IBM Power5 であるが，4 コアで最短の 38.550 秒を記録した。ハードとソフトの同調性が見られる。日本は，ソフトウェアの開発において昔から才能がないようなことを言われているが，ゲームソフトとこのコンパイラの性能は，日本が誇るソフトウェアと自負すべきである。（富士通さん，頑張ってください！

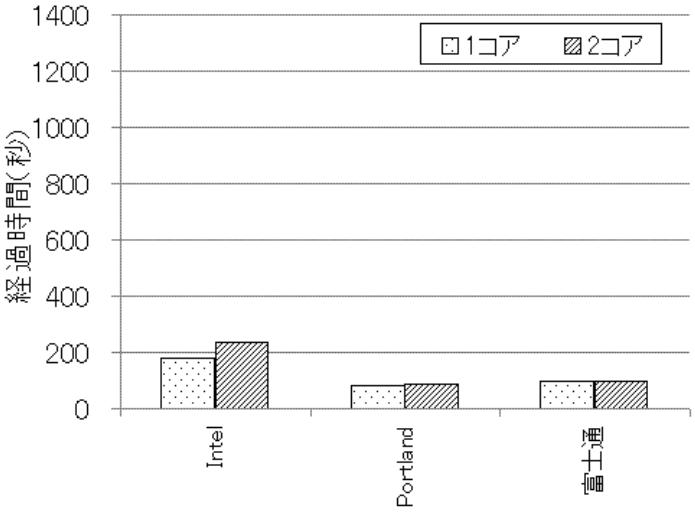

図 2. AMD 上における並列コンパイラ性能

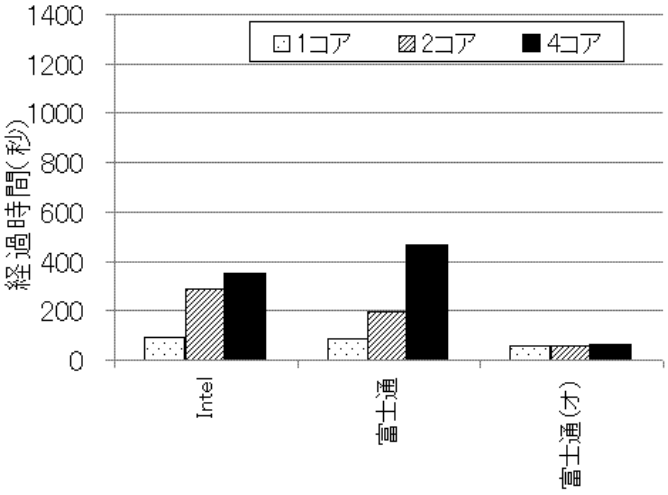

図 3. Intel Xeon 上における並列コンパイラ性能

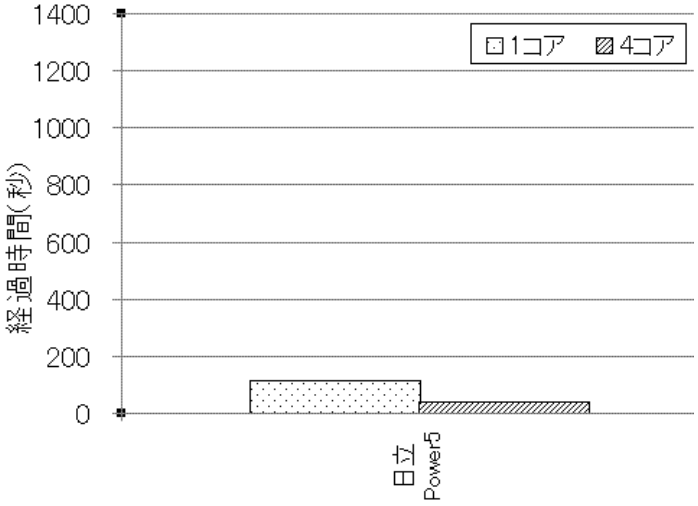

図 4. 日立 SR11000 上における並列コンパイラ性能

お願いします。また，可能だと期待しております。）

図 5 は Intel Intanium2 上での性能である。特に 1 ノード，64 コアまでの共有メモリー型マシンであることから，64 コアまで並列性能が期待できる。（気をつけなければならないことは,使用した計算機が SGI 社製 Altix 等とは異なり，物理的に共有メモリー型のマシンである。つまり，論理型の共有メモリー型マシンではない。）傾向は明らかに図 3 と同じである。並列性能が全く期待できない。そして，図 3 と比較するとかなり悪化する。無理に，共有メモリー型マシンを作成したのかもしれない。また上述と同様に，富士通の最適なオプションを選ぶと,悪化しなくなったが，性能の向上は見られない。

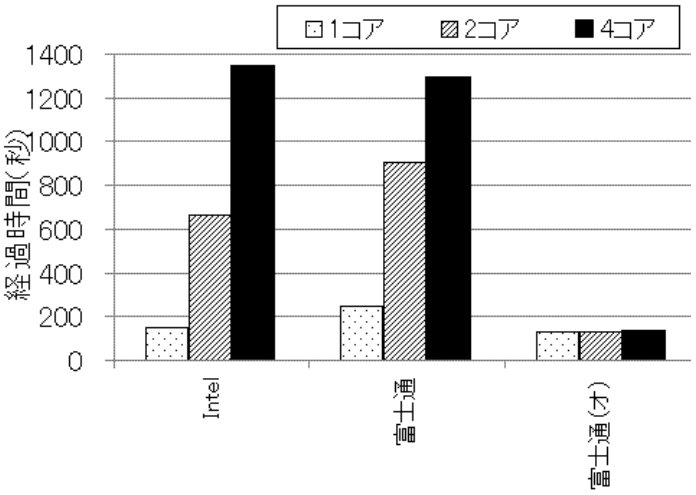

図 5. Intel Itanium2 上における
並列コンパイラ性能

その後，⑦，⑧，⑨のマシンでも富士通の最適なオプションを選べば，悪化せず。Intel は悪化し，Portland は富士通の最適なオプションと同様な傾向を示す結果となった。

最後に，自動並列化という視点でのハイライトを図 6（本文の最後に掲載）に示す。

4. まとめ

- 最短だったのは，日立コンパイル＆SR11000 モデル J1 上で，4CPUcore を用い並列実行した結果（38.550 秒）であった。1CPUcore の時が 113.329 秒から考えると自動並列化が機能しているように感じられる。
- 2 番目に短かったのは，49.978 秒を記録した Intel コンパイラ Version 9.1 を用いた非並列コンパイル＆富士通 Primequest 上での非並列実行であった。この値には少し驚いている。何故だろうか？
- 富士通さんが新たに行ったチューニングのうちオプションのみ使い再計測を行った。PG では約 30%，PQ では約 50%（ただし，並列実行可能なバージョンを用いたとき）の性能向上がみられた。また，どちらも並列実行時に性能向上が見られないが，悪化がなくなった。（AMD バルセロナでもそうかもしれない。）一方，以前のオプションでは悪化が見られた。オプションの説明をお願いしたい。
- 日立コンパイラ以外，自動並列性能がかなり悪いことが分かる．Many cores の時代が間近に迫っていることを考えると早急の対応が求められる。
- AMD のバルセロナは，富士通さんのコンパイラだと Portland Group を抜いているが，デュアルコアのオプテロンやアスロン 64X2 だと遅くなる。特異な命令を使っているのだろうか？　一方，512K のキャッシュは少なすぎる。Molpro を使った大行列の固有値問題では話にならない。Gaussian でも同様，困ったものだ。

5. 謝辞

これまで約 16 年間以上，分子科学の分野においてスーパーコンピュータの管理および調達に携わってきた。振り返ると，日立 S820, M682 (2CPU) から始まり, SR2201, SR8000, SR11000, SR16000, NEC SX-3, SX-4R, SX-5, SX-7, 富士通 VPP5000, PrimeQuest, IBM SP2, SGI Origin 2000, Origin2800, Altix3700, Altix4700 等まで相手に奮闘してきた気がする。余談だが，UNIX システムは DEC VAX-11/750, SONY NEWS 830 から利用し，恐らく日本で初めてインターネットを利用した研

究者の一人だと思う。朴さんには学生時代とてもお世話になった。そして縁があり，平成 21 年の春，上智大学 理工学部 物質生命理工学科に異動した。大学では，白衣を着て何とあの南部が実験の授業をやっている。ある意味，スーパーコンピュータから離れた立場となったが，実験研究者が実験装置に工夫をするように，理論研究者がコンピュータを意識してプログラムを開発することは，とても大切なことだと考えている。その一方，ここに至るまで様々な方々にお世話になった。この場をお借りして感謝申し上げる。ありがとうございました。

平成 22 年 3 月末　南部伸孝

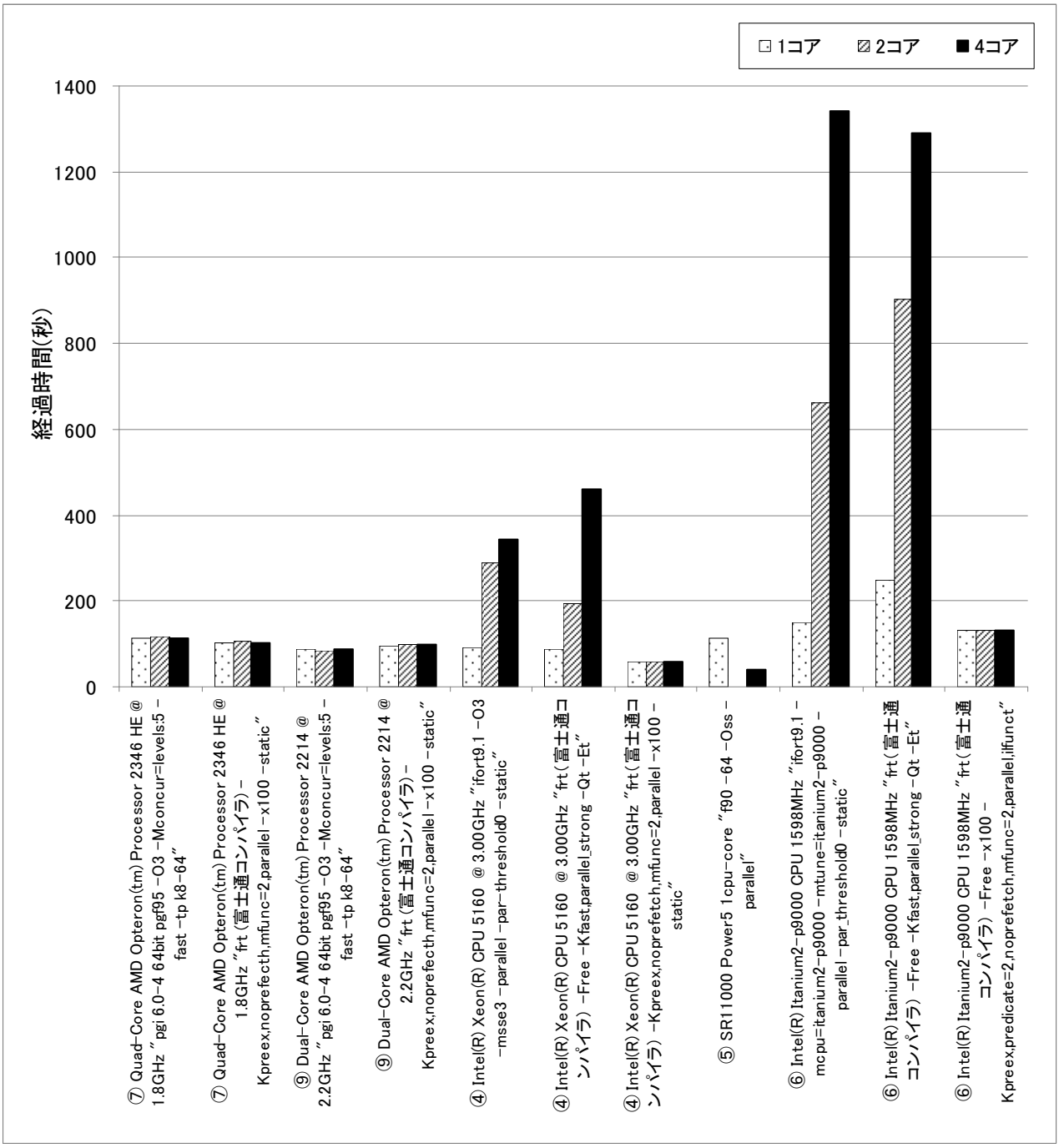

図 6. 様々なマシンと各社がリリースするコンパイラの自動並列化性能

【補足資料】

(1) 測定に用いたコード（4 次のシンプレクティク積分法）

```
! We use 4th order Gray sypmlectic integrator [JCP v106 p4832 (1997)]

! Coordinate integration coefficients
a(1) = 0.5*(1.0 - 0.577350269189625764509148780501 96)*m_traj%tstep
a(2) = 0.577350269189625764509148780501 96*m_traj%tstep
a(3) = -0.577350269189625764509148780501 96*m_traj%tstep
a(4) = 0.5*1.577350269189625764509148780501 96*m_traj%tstep

! Momentum integration coefficients
b(1) = 0.0
b(2) = 0.5*1.077350269189625764509148780501 96*m_traj%tstep
b(3) = 0.5*m_traj%tstep
b(4) = -0.5*0.077350269189625764509148780501 96*m_traj%tstep

! Make forward in time step for clasical trajectory

! Trajectory, monodromy matrixes and action are calculated
! simultaneosly to use advantage of local data cash
time = m_traj%tstep*m_traj%it
    ActionStep = 0.0
do k = 1, 4

    ! Data at q_{k-1}
    if(b(k) /= 0.0) then

        call tr_CashRightHandSides(m_traj, time)

        ! v_p = p_{k-1} -> p_{k}
        do i = 1, NUM_OF_DOF
            m_traj%v_p(i) = m_traj%v_p(i) + b(k)*m_traj%c_force(i)
        enddo

        ! From now v_q = q_{k-1}, v_p = p_{k}, time = t_{k-1}
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_dtmp(i, j) = 0.0
                do m = 1, NUM_OF_DOF
                    m_dtmp(i, j) = m_dtmp(i, j) &
                        & - m_traj%c_hess(i, m)*m_traj%m_QP(m, j)
                enddo
            enddo
        enddo
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_traj%m_PP(i, j) = m_traj%m_PP(i, j) + b(k)*m_dtmp(i, j)
            enddo
        enddo

        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_dtmp(i, j) = 0.0
                do m = 1, NUM_OF_DOF
                    m_dtmp(i, j) = m_dtmp(i, j) &
                        & - m_traj%c_hess(i, m)*m_traj%m_QQ(m, j)
                enddo
            enddo
        enddo
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_traj%m_PQ(i, j) = m_traj%m_PQ(i, j) + b(k)*m_dtmp(i, j)
            enddo
        enddo

    endif

    ! Calculate action step
    dtmp = 0.0      ! Get kinetic energy
    do i = 1, NUM_OF_DOF
        do j = 1, NUM_OF_DOF
            dtmp = dtmp + m_traj%v_p(i)*m_traj%v_p(j) &
```

```
                & *pes_GetKineticMatrix(i, j)
            enddo
        enddo
        if(b(k) /= 0.0) then
            ActionStep = ActionStep + a(k)*dtmp - b(k)*m_traj%c_pot
        else
            ActionStep = ActionStep + a(k)*dtmp
        endif

        ! v_q = q_{k-1} -> q_{k}
        do i = 1, NUM_OF_DOF
            v_dtmp(i) = 0.0
            do j = 1, NUM_OF_DOF
                if(i == j) then
                    v_dtmp(i) = v_dtmp(i) &
                        & + 2.0*m_traj%v_p(i)*pes_GetKineticMatrix(i, i)
                else
                    v_dtmp(i) = v_dtmp(i) &
                        & + m_traj%v_p(j)*pes_GetKineticMatrix(i, j)
                endif
            enddo
        enddo
        do i = 1, NUM_OF_DOF
            m_traj%v_q(i) = m_traj%v_q(i) + a(k)*v_dtmp(i)
        enddo

        ! From now v_q = q_{k}, v_p = p_{k}, time = t_{k-1}
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_dtmp(i, j) = 0.0
                do m = 1, NUM_OF_DOF
                    if(i == m) then
                        m_dtmp(i, j) = m_dtmp(i, j) &
                            & + 2.0*pes_GetKineticMatrix(i, m)*m_traj%m_PP(m, j)
                    else
                        m_dtmp(i, j) = m_dtmp(i, j) &
                            & + pes_GetKineticMatrix(i, m)*m_traj%m_PP(m, j)
                    endif
                enddo
            enddo
        enddo
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_traj%m_QP(i, j) = m_traj%m_QP(i, j) + a(k)*m_dtmp(i, j)
            enddo
        enddo

        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_dtmp(i, j) = 0.0
                do m = 1, NUM_OF_DOF
                    if(i == m) then
                        m_dtmp(i, j) = m_dtmp(i, j) &
                            & + 2.0*pes_GetKineticMatrix(i, m)*m_traj%m_PQ(m, j)
                    else
                        m_dtmp(i, j) = m_dtmp(i, j) &
                            & + pes_GetKineticMatrix(i, m)*m_traj%m_PQ(m, j)
                    endif
                enddo
            enddo
        enddo
        do i = 1, NUM_OF_DOF
            do j = 1, NUM_OF_DOF
                m_traj%m_QQ(i, j) = m_traj%m_QQ(i, j) + a(k)*m_dtmp(i, j)
            enddo
        enddo

        ! Time update time = t_{k}
        time = time + a(k)
        if(k == 4) time = m_traj%tstep*(m_traj%it + 1)
    enddo
```

(2) 詳細なコンパイラオプション及び計測経過時間

機種	分子動力学プログラム（自作）	経過時間（秒）
①	AMD Athlon(tm) 64 Processor 3500+ "ifort9.1-O3 -msse2"	159.029
①	AMD Athlon(tm) 64 Processor 3500+ "pgi 6.0-4 64bit pgf -O3 -fast -tp k8-64"	89.921
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "ifort9.1-O3 -msse2"	181.043
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "ifort9.1-O3 -msse2 -parallel -par_threshold0" & "setenv OMP_NUM_THREADS 2"	237.762
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "pgi 6.0-4 64bit pgf95 -O3 -fast -tp k8-64"	84.761
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 1"	83.529
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 2"	88.225
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 1"	96.620
②	AMD Athlon(tm)64 X2 Dual Core Processor 4400+ "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 2"	97.170
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 1"	114.180
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 2"	115.700
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 4"	114.710
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 1"	102.300
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 2"	104.310
⑦	Quad-Core AMD Opteron(tm) Processor 2346 HE @ 1.8GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 4"	102.210
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 1"	86.530
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 2"	85.690
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "pgi 6.0-4 64bit pgf95 -O3 -Mconcur=levels:5 -fast -tp k8-64" & "setenv NCPUS 4"	86.510
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 1"	96.180
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 2"	99.120
⑨	Dual-Core AMD Opteron(tm) Processor 2214 @ 2.2GHz "frt（富士通コンパイラ）-Kpreex,noprefecth,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 4"	98.290
③	Intel(R) Pentium(R) 4 CPU 3.40GHz "ifort9.1 -O3 -msse2"	88.807
③	Intel(R) Pentium(R) 4 CPU 3.40GHz "ifort9.1 -O3 -msse3"	78.825
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "ifort9.1 -O3 -msse3 -parallel -par-threshold0 -static" & "setenv OMP_NUM_THREADS 1"	89.800
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "ifort9.1 -O3 -msse3 -parallel -par-threshold0 -static" & "setenv OMP_NUM_THREADS 2"	287.730
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "ifort9.1 -O3 -msse3 -parallel -par-threshold0 -static" & "setenv OMP_NUM_THREADS 4"	343.050
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Free -Kfast -Qt -Et"	75.232
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 1"	86.670
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 2"	194.640
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 4"	461.120

④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Kpreex,noprefetch,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 1"	58.590
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Kpreex,noprefetch,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 2"	58.640
④	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz "frt（富士通コンパイラ） -Kpreex,noprefetch,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 4"	58.730
⑧	Intel(R) Core™2 Duo CPU E6850 @ 3.00GHz "ifort9.1 -O3 -msse3 -parallel -par-threshold0 -static" & "setenv OMP_NUM_THREADS 1"	77.900
⑧	Intel(R) Core™2 Duo CPU E6850 @ 3.00GHz "ifort9.1 -O3 -msse3 -parallel -par-threshold0 -static" & "setenv OMP_NUM_THREADS 2"	101.600
⑧	Intel(R) Core™2 Duo CPU E6850 @ 3.00GHz "frt（富士通コンパイラ） -Kpreex,noprefetch,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 1"	59.980
⑧	Intel(R) Core™2 Duo CPU E6850 @ 3.00GHz "frt（富士通コンパイラ） -Kpreex,noprefetch,mfunc=2,parallel -x100 -static" & "setenv OMP_NUM_THREADS 2"	58.600
⑤	SR11000 Power5 1cpu-core "f90 -64 -Oss -parallel" & "setenv HF_PRUNST_THREADNUM 1"	113.329
⑤	SR11000 Power5 1cpu-core "f90 -64 -Oss -parallel" & "setenv HF_PRUNST_THREADNUM 4"	38.550
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "ifort9.1 -mcpu=itanium2-p9000 -mtune=itanium2-p9000 -static"	49.978
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "ifort9.1 -mcpu=itanium2-p9000 -mtune=itanium2-p9000 -parallel -par_threshold0 -static" & "setenv OMP_NUM_THREADS 1"	148.610
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "ifort9.1 -mcpu=itanium2-p9000 -mtune=itanium2-p9000 -parallel -par_threshold0 -static" & "setenv OMP_NUM_THREADS 2"	661.780
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "ifort9.1 -mcpu=itanium2-p9000 -mtune=itanium2-p9000 -parallel -par_threshold0 -static" & "setenv OMP_NUM_THREADS 4"	1340.150
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -Kfast -Qt -Et"	170.916
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 1"	248.665
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 2"	902.400
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -Kfast,parallel_strong -Qt -Et" & "setenv OMP_NUM_THREADS 4"	1290.110
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -x100 -Kpreex,predicate=2,noprefetch,mfunc=2,ilfunct"	150.560
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -x100 -Kpreex,predicate=2,noprefetch,mfunc=2,parallel,ilfunct" & "setenv OMP_NUM_THREAD 1"	131.770
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -x100 -Kpreex,predicate=2,noprefetch,mfunc=2,parallel,ilfunct" & "setenv OMP_NUM_THREAD 2"	132.110
⑥	Intel(R) Itanium2-p9000 CPU 1598MHz "frt（富士通コンパイラ） -Free -x100 -Kpreex,predicate=2,noprefetch,mfunc=2,parallel,ilfunct" & "setenv OMP_NUM_THREAD 4"	132.350