Speeding Up NEC2++ by OpenBLAS

Yoshiyuki Takeyasu / JA6XKQ

OpenBLAS による NEC2++ の高速化

武安義幸 / JA6XKQ

NEC2++ [1] は数値演算ライブラリに ATLAS [2] を用いて高速化を図っている。NEC2++ を使い始めた当初 [3] は CPU に Core2 Duo を価格の観点から採用した。当時は高価であった同系列の Core2 Quad が中古市場で安価になったので、シミュレーションの更なる高速化を狙って CPU を換装した。単純な換装だけで CPU のコア数分の高速化を期待したが、話は単純ではなかった。

判れば簡単なことではあるが、その経緯を備忘録として本稿にまとめておく。数値演算ライブラリに OpenBLAS [4] を採用した結果、コア数増加分以上の高速化が得られた。

ATLAS によるマルチ・コア対応

コンピュータのマザーボードと OS (Debian 6.0.2 Squeeze) については、3.16GHz Core2 Duo を 3.0GHz Core2 Quad へ単純に換装できた。gcc-4.4 でビルドしていた NEC2++ コードもそのままで使用できるが、システム・モニタで CPU 使用率を見ると、4 コアのうち 2 コアしか利用されていないことが判明した。

使用している数値演算ライブラリは、Debian のパッケージとして配布されている ATLAS であった。ATLAS はその名称 Automatically Tuned Linear Algebra Software が示すように、CPU の特性(コア数、クロック周波数、インストラクションなど)に対応して自動チューニングするものである。しかし、そのチューニングはライブラリをビルド(コンパイル)するときになされるもので、パッケージとして配布されているものはビルド時に決定されているコア数にしか対応しない。

そこで、配布パッケージを使用せずに、ATLAS をソースコードからビルドしてみた。マニュアルに従ってビルドしているつもりだが、パラレル・スレッドのライブラリが生成されなかったり、スタティックのみで共有ライブラリが生成されないなどのトラブルに遭遇した。ATLAS と gcc のバージョンをそれぞれ変えて試行するも、バージョンごとに結果が異なり、トラブルは収束するどころか混乱の様相を呈した。

OpenBLAS によるマルチ・コア対応

ATLAS では自力で問題解決できないと判断し、他の演算ライブラリを試すことにした。ATLAS は、ビルドの際に各種パラメータを明示的に与えることで、CPU やプログラミング言語環境の相異に柔軟に対応できるようになっている。しかし、それが、初心者には取り扱いを難しくしている要因かもしれない。一方、OpenBLAS は各種パラメータを明示的に与えることもできるが、何もパラメータを与えずとも make だけで基本的なマルチ・コア対応のライブラリが生成される点は、初心者には重要なポイントであろう。

さらには、OpenBLAS はその前身である GotoBLAS (後藤BLAS)の時より、非商用ライブラリで最速と言われていることが、一番の魅力であろう。

OpenBLAS でのトラブル

後述するように、OpenBLAS を使用するために NEC2++ 側でビルド・ツールとソース・コードの変更を行なった。コンパイルはエラー無く終了するものの、実行すると NEC2++ が利用している zgetrs 関数でエラーを生じる。

OpenBLAS と NEC2++ をデバッグ情報付きでコンパイルして、問題を gdb デバッガでトレースした。

トレースの結果、OpenBLAS の zgetrs.c において、転置行列の判定でエラーを生じていることが判明した。コンパイラの文法としては正しいが、関数のロジックとして誤りを生じていた。誤りは下記のように生じている。誤りの発生が OpenBLAS のバグであるのか、使い方の誤りであるのかは判断できていない。

NEC2++ の matrix_algebra.cpp 内 solve 関数で LAPACK_zgetrs を使い、そこで転置行列に関する char 型の引数 CblasNoTrans を渡している。トレースの結果、OpenBLAS は、ヘッダー・ファイル cblas.h の enum CBLAS_TRANSPOSE で CblasNoTrans=111 と定義されている。これが zgetrs.c のローカル変数 trans_arg へ、文字列 'o' (ASCII 111)として渡る。

一方、zgetrs.c 内で trans_arg を判定するにあたっては、文字列 'N' と直接比較している。正しくは、CblasNoTrans である文字列 'O' ('o' を大文字へ変換)と比較すべきである。zgetrs.c 内での trans_arg の判定では、転置行列に関する四つのケース(No transposed、Transposed、Conjugate transposed、only conjugated)、文字列としては 'N' 'T' 'C' 'R' と比較されるので、'O' はいずれにも該当せずに誤った判定が下されていた。

ヘッダー・ファイル cblas.h 内で、

```
enum CBLAS_TRANSPOSE {CblasNoTrans=111, CblasTrans=112,
                      CblasConjTrans=113, CblasConjNoTrans=114};
```

と定義されているので、それぞれを文字列 'O' 'P' 'Q' ' R' として、zgetrs.c の変更前

```
if (trans_arg = = 'N') trans = 0;
if (trans_arg = = 'T') trans = 1;
if (trans_arg = = 'R') trans = 2;
if (trans_arg = = 'C') trans = 3;
```

を、

```
if (trans_arg = = 'O') trans = 0;
if (trans_arg = = 'P') trans = 1;
if (trans_arg = = 'Q') trans = 2;
if (trans_arg = = 'R') trans = 3;
```

と変更した。プログラミング手法とライブラリ全体としては統一性を欠いた解決法かもしれないが、NEC2++ を使う上で手っ取り早い解決策とした。

ビルド・ツールの変更点

NEC2++ のソースコードには、GNU Autoconf/Automake/Libtool を用いたビルドを前提として関連ファイルが添付されている。添付されている関連ファイルは ATLAS を対象とした記述となっているので、それを OpenBLAS に合致させる。

configure のルールを記述した configure.in において、LAPACK ライブラリをチェックしている部分(太字)

```
LIBLAPACK=
        AS_IF([test "x$with_lapack" != xno],
        [
                LDFLAGS="$LDFLAGS -L/usr/lib/atlas-base/atlas"
                AC_CHECK_LIB(
                [lapack], [clapack_zgetrf],
                [AC_SUBST([LIBLAPACK], ["-L/usr/lib/atlas-base/atlas
                -llapack -lblas -lpthread"]) AC_DEFINE([LAPACK], [1],
                [Define if you have liblapack])],
                [AC_MSG_FAILURE([lapack library test failed (--without
                -lapack to disable)])],
                [])
        ])
```

を、

```
LIBLAPACK=
        AS_IF([test "x$with_lapack" != xno],
        [
                LDFLAGS="$LDFLAGS -L/usr/lib/openblas-base/lib"
                AC_CHECK_LIB(
                [openblas], [LAPACKE_zgetrf],
                [AC_SUBST([LIBLAPACK], ["-L/usr/lib/openblas-base/lib
                -lopenblas -lpthread"]) AC_DEFINE([LAPACK], [1], [Define
                if you have liblapack])],
                [AC_MSG_FAILURE([lapack library test failed (--without
                -lapack to disable)])],
                [])
        ])
```

と変更した。この変更により、configure の過程で OpenBLAS ライブラリが正しく認識される。

(C) 2013, Yoshiyuki Takeyasu / JA6XKQ

NEC2++ の変更点

OpenBLAS ライブラリを使うために NEC2++ のソースコードを修整する。 matrix_algebra.cpp の

```
extern "C"
{
#include <atlas/clapack.h>
}

int info = clapack_zgetrs (CblasColMajor, CblasNoTrans,
        n, 1, (void*) a.data(), ndim, ip.data(), b.data(), n);

int info = clapack_zgetrs (CblasColMajor, CblasNoTrans,
        n, 1, (void*) a.data(), ndim, ip.data(), b.data(), n);
```

を、

```
extern "C"
{
#include </usr/lib/openblas-base/include/lapacke.h>
#include </usr/lib/openblas-base/include/cblas.h>
}

int info = LAPACKE_zgetrf((int) CblasColMajor, (lapack_int) n,
(lapack_int) n, (lapack_complex_double*) a_in.data(),
(lapack_int) ndim, (lapack_int*) ip.data());

int info = LAPACKE_zgetrs ((int) CblasColMajor,
        (char) CblasNoTrans, (lapack_int) n, (lapack_int) 1,
(const lapack_complex_double*) a.data(),
(lapack_int) ndim, (const lapack_int*) ip.data(),
(lapack_complex_double*) b.data(), (lapack_int) n);
```

と変更した。ライブラリとインクルード･ファイルの所在は、環境変数で PATH を通しておけば良いのだろうが、絶対パスでソースコード内に記述した。

以上の変更点が数値演算ライブラリを ATLAS から OpenBLAS へ変更するための修整点である。その他に、二点の修整を行なった。

一つは、[3] で判明した c_geometry.cpp の SC カード判定部分

```
if ( card_id != "GH") // gm_num != 10 )
        isct=0;
```

を、

```
if ( card_id != "SC") // gm_num != 10 )
        isct=0;
```

と訂正した。

二つ目は、実行時間計測に関するものである。シミュレーション自体には関係しないが、数値演算ライブラリの変更後に実行時間の改善具合を計測していて、NEC2++ に組み込まれている計測ルーチンの不具合に気づいた。CPU のコア数がシングルの場合は正しい実行時間を計測できるが、コア数がデュアル、クォッドと複数になると異常値（並列で実行された CPU タイムの合算か？）を示す不具合である。不具合と言うには語弊があるかもしれない、スピードアップを体感するのに不都合を感じた。

これは、misc.cpp にて secnds なる関数で時間を計測しており、そこで用いている sysconf(_SC_CLK_TCK) が原因のようである。対策を検索すると、実時間の計測には gettimeofday を用いるのが常套手段のようであり、例示 [5] からコードを借用した。

```
#include <sys/time.h>
double getETime()
{
struct timeval tv;
gettimeofday(&tv, NULL);
return tv.tv_sec + (double)tv.tv_usec*1e-6;
}
```

実行時間比較

[3] での計算モデルを用いて、3.16GHz Core2 Duo / ATLAS と 3.0GHz Core2 Quad / OpenBLAS での実行時間を比較する。前者については秒単位での正確な数値記録が無く、分単位での概略値となる。結果を下表に示す。

Model No. of surface-patch	3.16GHz Core2 Duo ATLAS (sec)	3.0GHz Core2 Quad OpenBLAS (sec)	Improvement
6,319	360	135	2.67
9,968	1,380	510	2.71
13,992	3,600	1,360	2.65

改善率は平均で 2.67 倍となる。コア数の単純な比率で 2 倍となるところ、それ以上の改善は OpenBLAS の性能に拠るところであろう。また、CPU のクロック周波数が 3.16GHz から 3.0GHz へ下がっていることを勘案すると、改善率は 2.81 倍に及ぶ。

まとめ

NEC2++ が用いる数値演算ライブラリを ATLAS から OpenBLAS へ変更することでマルチ・コア対応を行い、高速化を図ることができた。OpenBLAS の高性能により、コア数比率以上の改善を得ることができた。

シミュレーションの高速化により、アンテナの最適化設計での複数回演算に威力を発揮することが期待できる。

／／
☆

参考文献

[1] NEC2++
Timothy Molteno
http://elec.otago.ac.nz/w/index.php/Necpp

[2] Automatically Tuned Linear Algebra Software (ATLAS)
http://math-atlas.sourceforge.net/

[3] Simulation of a horn antenna using NEC2++
NEC2++ によるホーン･アンテナのシミュレーション
武安義幸、JA6XKQ
http://www.terra.dti.ne.jp/~takeyasu/Nec2pp3SecHorn.pdf

[4] OpenBLAS
http://xianyi.github.com/OpenBLAS/

[5] 時間計測の方法について
東京大学情報基盤センター
黒田久泰
http://www.cc.u-tokyo.ac.jp/support/press/news/VOL8/No5/data_no2_0609.pdf

(C) 2013, Yoshiyuki Takeyasu / JA6XKQ