GFDセミナー2011分科会 2011年8月21日

SJPACKのAVX利用について

石岡圭一

E-mail: ishioka@gfd-dennou.org

まずは球面のスペクトル法についてのおさらい

偏微分方程式中の従属変数を球面調和関数で展開.

$$f(\lambda,\mu,t)=\sum_{n=0}^{M}\sum_{m=-n}^{n}a_n^m(t)Y_n^m(\lambda,\mu)$$

ここに, λ: 経度, $\mu=\sin\theta$, θ: 緯度, t: 時刻, M: 切断波数.

$$Y_n^m(\lambda,\mu)=P_n^{|m|}(\mu)e^{im\lambda}.$$

$P_n^m(\mu)$は Legendre陪関数.

$$P_n^m(\mu)\equiv\sqrt{(2n+1)\frac{(n-m)!}{(n+m)!}\frac{1}{2^n n!}}\times(1-\mu^2)^{m/2}\frac{d^{n+m}}{d\mu^{n+m}}(\mu^2-1)^n\quad(0\le m\le n).$$

ルジャンドル陪関数による展開

球面スペクトル法には球面調和関数変換が必要．そのうち計算量の大部分を占めるのはルジャンドル陪関数に関する以下の変換である:

逆変換

$$g_j^m = \sum_{n=m}^{M} s_n^m P_n^m(\mu_j) \quad (m = 0, \cdots, M;\ j = 1, \cdots, J)$$

正変換

$$s_n^m = \sum_{j=1}^{J} w_j g_j^m P_n^m(\mu_j) \quad (m = 0, \cdots, M;\ n = m, \cdots, M)$$

ここに，M: 切断波数，J: ガウス緯度の個数，$P_n^m(\mu)$: ルジャンドル陪関数，μ_j: ガウス緯度，w_j: ガウス重み．

必要な計算量

$m \geq 1$に対しては実部・虚部を扱わねばならないことと，ルジャンドル陪関数の対称性から添字jに関する計算は半分の$J/2$でよいことを考慮すると，逆変換・正変換に必要な計算量はNはルジャンドル陪関数をすべて事前に計算しておくとしても(乗算と加算が必要なことを考慮して)

$$N = \frac{J}{2} \cdot (M+1)^2 \cdot 2 = J(M+1)^2.$$

さらに，正変換時の数値計算で誤差が出ないことを保証するためには $J > \frac{3}{2}M$ としておく必要があるので，

$$N \sim \frac{3}{2}M^3.$$

Mが大のときは，この変換計算をいかに効率化するかが全体の計算スピードを決める．

ISPACKにおける球面調和関数変換のこれまでの実装

stpack(1995年) 最も素朴な実装

smpack(1998年) ベクトル化を追求

snpack(1999年) ベクトル化を追求しつつ省メモリ化

sjpack(2009年) スカラー計算機での高速化(SSE2に対応)

sjpack-cuda(2010年) GPGPUへの対応

sjpack(AVX対応版)(2011年) AVX命令への対応(ispack-0.94 として公開済み).

AVX対応について

今年の1月に発売が開始された Intel の Sandy Bridge アーキテクチャの CPUでは AVX (Advanced Vector eXtensions) 命令が新たに使えるようになった.

AVX命令の主な特徴:

- 256bit 長の SIMD命令が使えるので, 倍精度変数4個に対する処理が一括でできる. それにより, 128bit長の SSE2命令に比べると理論的には倍速で処理ができる.
- 四則演算などに 3オペランド命令が使えるので, `C = A + B` のような形の計算がレジスタの余分なコピーなしで可能.

コンパイラはそれなりに賢くなってきているが，AVX命令を最大限活用しようとするならば(SSE2対応の際と同様に)，アセンブリ言語でのプログラミングは避けて通れない．

ということで，Legendre陪関数変換のコアの部分はアセンブリ言語で書いて最適化しておくべき．

実際コアとなる部分はどんなものか?

例: ljlswg-fort.f

```
      SUBROUTINE LJLSWG(JH,S,R,Y,QA,QB,W)

      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION W(JH,2),Y(JH),S(2)
      DIMENSION QA(JH),QB(JH)

      DO J=1,JH
        W(J,1)=W(J,1)+S(1)*QA(J)
        W(J,2)=W(J,2)+S(2)*QA(J)
        QB(J)=QB(J)+R*Y(J)*QA(J)
      END DO

      END
```

と短いサブルーチンなので, これを(AVX命令等を使って)ガシガシに最適化すればいい(というか, このコアの部分をサブルーチンとして独立させてあるところが設計の肝の一つ).

AVX命令を使って最適化した例
例: ljlswg-avx.s

```
.globl ljlswg_
ljlswg_:
        movl (%rdi), %edi
        vbroadcastsd (%rsi), %ymm0
        vbroadcastsd 8(%rsi), %ymm5
        vbroadcastsd (%rdx), %ymm1
        movq  8(%rsp), %r10
        shlq $3,%rdi
        xorq %rdx,%rdx
        subq %rdi,%rdx
        movq %r10,%r11
        addq %rdi,%r11
        subq %rdx,%r8
        subq %rdx,%r9
        subq %rdx,%r10
        subq %rdx,%r11
        subq %rdx,%rcx
```

```
.L0:
        vmulpd (%rcx,%rdx),%ymm1,%ymm4
        vmovapd (%r8,%rdx), %ymm2
        vmulpd %ymm2,%ymm4,%ymm4
        vmulpd %ymm0,%ymm2,%ymm3
        vmulpd %ymm5,%ymm2,%ymm2
        vaddpd (%r9,%rdx),%ymm4,%ymm4
        vaddpd (%r10,%rdx),%ymm3,%ymm3
        vaddpd (%r11,%rdx),%ymm2,%ymm2
        vmovapd %ymm4,(%r9,%rdx)
        vmovapd %ymm3,(%r10,%rdx)
        vmovapd %ymm2,(%r11,%rdx)
        addq $32,%rdx
        jnz .L0
        ret
```

ベンチマーク結果

とりあえず T170の普通の設定($J = 256, I = 512$)での snpack との速度比較(なお, sjpack では FFT として FFTJ を使っている). なお, Flops は実際の計算量(Legendre陪関数の毎回計算を含む)ではなく, Legendre陪関数をもし事前に計算してメモリに格納していたとする場合の変換そのものに絶対必要な計算量で算出している).

Intel Core-i7 2620M(Sandy Bridge 2.7GHz, turbo burst時 3.4GHz),
OS: Debian squeeze 64bit, Compiler: ifort 12.0.5, option xAVX
において
SJPACK: 6.7 GFlops
SNPACK: 2.9 GFlops

sjpack を SSE2対応のものにあえて変えた場合

SJPACK: 5.1 GFlops

評価

上記の GFlops は Legendre陪関数の on the fly での計算に必要な分を除いたものであるので，それも込みの実際の計算量で GFlops を見積るなら, 7/4 倍すべき．すると, AVX版の SJPACKでの実際のGFlopsは, $6.7 \times 7/4 = 11.7$ GFlops.

Intel Core-i7 2620M の理論性能は, $3.4 \times 4 \times 2 = 27.2$GFlos なので, AVX版の SJPACKでは理論性能の$11.7/27.2 \times 100 = 43\%$ はでていることになる.

なお, 次ページでコメントするが, FFT等を除いたLegendre陪関数変換の部分だけなら理論性能の約 50%程度はでている.

結論

・sjpack の AVX対応はそれなりにできた．球面調和関数変換全体としては SSE2版の 1.3倍速くらいにはなっている．倍率が 1.3倍止まりなのは，AVX対応にしたのが Legendre陪関数変換のところだけで，FFTJなどはAVXに対応していないことが一因． Legendre陪関数変換のところだけなら 1.5倍速にはなっている．

・AVX化した Legendre陪関数変換が 1.5倍速どまりなのは，AVX命令を使っても，load/store の部分がSSE2命令に比べて速くなるわけではないため．SSE2版では乗算のところが律速になっているが，AVX版では乗算の部分が倍速になっている分，load/storeが律速になり，その関係で全体で 1.5倍速どまりになっている．

課題

・FFTJの部分のAVX化．ただ，ここはあまり頑張ってもそんなに意味が無いかも (上に書いたようにAVX化しても load/store が速くなるわけではないので，多分 SSE2版の倍速には絶対にならない)．

・それよりは，もっと簡単にできる sjpack のMPI化が TODO スタックの一番上かな．

ネット上の情報源

- Agner Fog氏のページ(x86アセンブリでの高速化についての情報源)
(http://www.agner.org/optimize/)

- Intel のデベロッパーズマニュアル
(ftp://download.intel.co.jp/jp/developer/jpdoc/)

「ネットは広大だわ……」(by 草薙素子, 1989年)