Linux/Alpha 活用講座

清水尚彦　nshimizu@keyaki.cc.u-tokai.ac.jp

CXML活用法 ——ベクトル算術演算関数について

本屋に行ったら、Linuxを題材にしたビジネス書が出ていました。NTT出版「Linuxはいかにしてビジネスになったか　コミュニティ・アライアンス戦略」です。この本でもいろいろ分析していますが、Linuxが求心力を持ち続けているのは、開発の中心となったLinus自身がLinuxから利益を生もうとしなかったことが1つの要因となっていると思います。UNIX InternationalやOSFが成し得なかった「UNIX系のOSのハードウェアメーカーを超えた統合」が、Linuxによって実現していくのは、はたで見ていて楽しいものですね。OSなんて、アプリケーションを動かすための土台でしかないのですから、囲い込みを行って移植性を下げることをするよりも、同じベースでその上のミドルウェアやアプリケーションで勝負してもらいたいものです。

もちろん、OS自体の性能によってアプリケーションの性能に影響の出るものもあります。先月号で示したように、TLBミスが18%もの影響を与えるケースもあるし、マルチプロセッサの扱いやファイルのアクセス性能など、OSが頑張るべき所も多々あります。これらを個々の企業が囲い込んでいたのが既存のOSの世界ですが、それほど神秘的なことをしてきたわけではないのだろうと思っています。系統的にデータを取ったり対策したりという退屈な仕事を強いられるので、ペイベースでないとなかなかできないことなのですが、必要な企業がそれぞれ対策をオープンな議論の中で行うようにすればいいのだろうと思います。中にはSH4のように、ハードの出来が悪いために他のプロセッサにとって性能が悪化するような対策をカーネルに求める例もあるので、センス良く取捨選択するオーガナイザが必要となることは確かです。

こういった流れの中で、Linuxを扱う企業が、単なるディストリビュータとしてだけでなく、カーネルのメンテナンスにまでコミットしていくことは大切なことだと思っています。そういった意味から最近のいくつかの会社のやり方を注目しているところです。日本でもディストリビューションを作っているだけでなく、積極的にシステムの開発まで手を付けているDigital Factoryなどは要注目と思っています。ビジネス世界との潤滑油としてこういった会社が成功していくといいですね(私が全部の会社を知っているわけではないので、他にも注目すべき会社はどんどん生まれていると思います)。

A 研究室の近況

私の研究室で、ようやくAMDのAthlonを搭載したシステムを導入することができました。今年度は大学院生があまり学会発表しなかったので(これ自体は困ったことですが)、旅費に取っておいた予算を思い切ってPCに振り分けました。

主目的はCPLDの論理合成ツールがWindows用のものしかないため、その実行用ということになっていますが、Linuxを搭載して性能の検証もぼちぼちやっていこうと思っています。暇になったらAthlonのメモリ管理ユニットを使ったTLBミス低減のカーネルパッチも作ろうと思っていますが、あまり期待しないでいてください。

この主目的ですが、オンチップメモリを使ってHPCアプリケーションを高速に実行する「SCALT」と呼ぶプロセッサを1997年くらいから細々と作成していて、これをそろそろCPLD(Complex Programmable Logic Device)上だけでも実現できるかなと思っていて、大規模な合成ができるマシンが必要となってきたわけです。プロセッサができても、OSやコンパイラがないとお話にならないわけで、そこには当然ながらLinuxとGCCを利用しようと思っています。CPLDプロセッサで動くようになれば、コンパイラとライブラリの開発も比較的ス

トレスなく進むのではないかと期待しています。

PC1台買うのに窮々としている弱小研究室でハードウェアの開発をしようというのは、無謀と言えば無謀ですが(大体CPLD自体が高くてなかなか買えない)、毎年1人ずつ学生を割り当てる程度のリソースで32bitのパイプラインプロセッサが設計できるということを実証するのも面白いですよね。もっとも、新しく入った学生が始めるときにはゼロからのスタートとなるので、ずっと継続的な開発が続けられないところが大学の悩みではあります。

APIが新サーバ「CS20」を発表

さて、HPCのお祭り「SuperComputing 2000」がダラスで行われ、そこで恒例のTOP500が発表されました(記事末のRESOURCE[1]を参照)。今回も例によってLinux/Alphaのクラスタである「CPLANT」が100位以内に入っています。

ここでAPI(Alpha Processor, Inc.)が新しく1Uのラックマウントタイプのサーバ「CS20」を発表しました。1Uのサーバはすでに Compaqから「DS10L」が発表されていましたが、APIのサーバは833MHzのデュアルプロセッサというところが決定的に違います。これを使うと、42サーバが1ラックに載るということなので、1ラックで84プロセッサのマシンが実現できることになります。執筆時点のAlphaでは、最高クロックでSPECベンチマークの値も大変優れています。サーバには2つのFast Ethernetが搭載されているので、コンパクトなBeowulf型の高性能マシンを作りたい人達にとっては良い選択になることでしょう。台数を多く搭載する場合には、それなりに電源も用意する必要があるので、自宅で使うには躊躇するような代物になりますね。

ちなみに、1台の消費電力は約500Wとなるらしいですから、40台搭載すると20KW(100Vのラインで200A)となります。これは研究室でもちょっと遠慮させていただきたくなる電力消費量です。もっともこれは、IAサーバでも似たようなものでしょうから、Alphaだけの問題ではなく、大規模なシステムを組む時にはそれなりに電源を考えなくてはいけない、というものだと思ったほうがいいですね。

このシステムで得られるピーク性能は約130GFLOPSとなり、ちょっとした(?)スーパーコンピュータです。そういえば某国立大学に入っているマシンなどは、この程度の性能なら数百KW程度になるはずなので、許せる範囲かなと思いますね。どうですか? 一家に1台スーパーコンピュータ! 冬でも暖房いらず! なんてね。もっとも夏に「入気温度」を35度に保つのは至難の技かもしれません。

ベクトル算術関数とは?

さて、ヨタ話はこれくらいにして本題に入りたいと思います。今回からしばらくCompaqのリリースしているライブラリである「CXML」(Compaq eXtended Math Library)について、使い方の説明などをしていこうと思います。算術ライブラリであるCPML(Compaq Portable Math Library)は、libmの代わりにリンクするだけなので、使い方に困ることはありません。しかし、LAPACK互換のルーチンについてはLAPACKの説明書を見て頂ければよいとしても、それ以外のルーチンは英語の説明書しかなく分かりにくいと思いますので、簡単な解説とともに使い方を説明していきたいと思います。

さて、記念すべき第1回はベクトル算術演算関数(サブルーチン)です。この仲間は表1に示すように、全部で7種類あります。

「ベクトル」と付かなければ、聞いたことがあるような物ばかりだと思います。実はこれらの関数(サブルーチン)は、ベクトルと付いていない通常の演算を、配列に対してまとめて実行するだけのものです。

なんでこんな物が必要なのか、という理由を考えてみましょう。まず、通常のlibmなどで三角関数を計算している手順を復習します。三角関数は「級数」に展開して計算できることが知られています(級数という言葉に馴染みのない方は無視して読み飛ばしてください)。級数展開をそのまま計算すると計算量が膨大になることから、計算量を低減するために式をまとめたりして変形した式を基もとに計算を行います。

ところが、パイプラインプロセッサでこの計算を行うとき、データの依存性が発生して、どうしてもパイプラインがフルに使えないケースが出てきてしまいます。一方、行列で表されるアルゴリズムを基にした大規模な数値計算では、演算を行うときには配列されたデータ(ベクトル)にまとめて演算を行うケースが多く、特定のデータに対してだけ演算するというケースは少ないのです。そこで出てくるアイデアが「複数のデータに対する演算をいっぺんに行えば、パイプラインの空きを埋めることができるのでは?」というものです。

この技術は、もともとパイプラインのレイテンシが長いベ

表1 ベクトル算術関数

関数	意味
VSQRT	ベクトル平方根演算
VCOS	ベクトルコサイン演算
VSIN	ベクトルサイン演算
VRECIP	ベクトル逆数演算
VLOG	ベクトルログ演算
VEXP	ベクトル指数演算
VCOS_SIN	ベクトルサインコサイン演算

クトル計算機(従来のスーパーコンピュータ)では必然的に用いられてきた方式です。しかしこの方式は、最近の長いパイプラインのマシンでもそれなりに有効性が高いと考えられます。そこで、これらのベクトル演算のライブラリを用意して、通常のプログラムから呼び出すことで性能を向上させよう、というものが表1に示したサブルーチンなのです。

関数の使い方

それでは例を上げて使い方を説明しましょう。まずは「VSQRT」です。このサブルーチンを呼び出すには5つの引数を用意します。FORTRANからは次のように呼びます。

```
CALL VSQRT(X, IX, Y, IY, N)
```

ここでXは関数の計算を行う引数配列を示します。IXはこの配列上でデータを何個おきに利用するかを示す(ストライド)数値で、連続でアクセスするときには「1」を与えます。Yには計算結果を格納する配列を示し、IYはIX同様に結果配列のストライドを与えます。最後のNは演算個数を与えます。このサブルーチンは次のFORTRANのDOループと同じことをします。

```
DO I = 1, N
 Y((I-1)*IY+1) = SQRT(X((I-1)*IX+1))
ENDDO
```

ちょっと分かりにくいですね。よく使う「IX=IY=1」の場合に限定して書き直して見ると、

```
DO I = 1, N
 Y(I) = SQRT(X(I))
ENDDO
```

となります。つまりDOループでN回反復する代わりにサブルーチンを1回呼ぶようにするだけです。

VSQRT以外の関数も使い方はほとんど同じです。「VCOS」はCOS関数のベクトル版ですし、「VSIN」はSIN関数、「VLOG」はLOG関数、「VEXP」はEXP関数にそれぞれ対応します。ちょっと見慣れないものが「VRECIP」と「VCOS_SIN」でしょう。VRECIPは逆数を計算するものです。すなわち、

```
DO I = 1, N
 Y((I-1)*IY+1) = 1.0/(X((I-1)*IX+1))
ENDDO
```

という動作になります。

一方、VCOS_SINは、COSとSINをいっぺんに計算します。呼び出し方法は、

```
CALL VCOS_SIN(X, IX, Y, IY, Z, IZ, N)
```

となります。今までのサブルーチンに比べて引数の数が増えていますね。この動作は次のようになります。

```
DO I = 1, N
 Y((I-1)*IY+1) = COS(X((I-1)*IX+1))
 Z((I-1)*IZ+1) = SIN(X((I-1)*IX+1))
ENDDO
```

なんでこんな関数をわざわざ作る必要があるのでしょう？

単純に考えればVCOSとVSINを別々に呼べば済むことです。実は、Xの配列が大きい時にキャッシュミスやTLBミスを回避できるということのほかに、アルゴリズムとしてこれら2つの関数はほとんど同じで、前処理の多くの部分を共通化することができるため、両方の関数値を必要とするプログラムをより高速化できる可能性がある、というメリットがあるのです。

ベクトル算術関数の効果

さて、それでは実際にこれらの関数がどれだけ高速化に寄与しているのかベンチマークプログラムを作って計ってみましょう。

測定に使用したマシンは、Compaq「AlphaStation XP1000」(500MHz)で、OSは「Debian/potato」です。1要素当たりの通常のループによる関数の実行時間と、ベクトル関数の実行時間をマイクロ秒単位で測定し、ベクトル関数による加速率を算出しています。また、コンパイラやオプションに依存する可能性が高いため、Compaq Fortranとg77を用いて実行しました。

g77では、「-lcpml」を指定しないと標準のlibmがリンクされるため、基本関数の実行時間が長くなります。とは言ってもDebian/potatoにはglib-2.1が搭載されているため、旧来のlibmよりも速い、私がチューニングした関数が入っています。そのためSIN、COSなどは、それほど大きく見劣りしない程度の出来になっています。

一方、LOGやEXPのように、全然性能が出ていないものもありますね。このあたりはまじめにlibmをチューニングすると、Alpha以外のRISCのユーザーにも役に立つので、そのうちに手を付けたいところです。

表2を見ると、加速率が1を切っている関数が結構あることが分かります。SQRTやRECIPのように、もともとプロセッサ

に命令として備わっているものをベクトル化した関数では、性能を出すこと自体が難しいように思えるかもしれませんが、このあたりは本気でチューニングすれば解決するはずです。

とりあえず、ユーザーとしてはVSQRTやVRECIPは今のところ使わないのが良い選択なのかもしれません。データを細かく解析すると色々と面白い特性があるのですが、詳細は読者のお楽しみに取っておきます。

プログラムはリスト1に示しますが、比較的単純です。詳しい説明がなくてもFORTRAN初心者でも見れば分かるのではないでしょうか？ DTIME関数は、呼ぶたびに直前のDTIMEからの経過時間を返す関数です。

表2 ベンチマークの結果

COMPAQ FORTRAN -arch ev5			
sqrt	5.8594000E-02	7.2266005E-02	0.8108101
cos	0.1494150	5.9570000E-02	2.508226
sin	0.1503910	6.3475996E-02	2.369258
recip	2.4414001E-02	4.2969000E-02	0.5681770
log	9.2774004E-02	5.7617001E-02	1.610185
exp	9.2773005E-02	5.9570000E-02	1.557378
cos_sin	0.1787120	6.9335997E-02	2.577478
COMPAQ FORTRAN -arch ev6			
sqrt	6.0546998E-02	7.2266005E-02	0.8378353
cos	0.1328120	5.8594000E-02	2.266649
sin	0.1230470	6.0545996E-02	2.032290
recip	2.4414001E-02	4.1992001E-02	0.5813964
log	0.1064460	5.4687001E-02	1.946459
exp	9.7656004E-02	5.6641001E-02	1.724122
cos_sin	0.1748050	7.1289003E-02	2.452061
COMPAQ FORTRAN -fast -arch ev6			
sqrt	3.7108999E-02	7.2265998E-02	0.5135056
cos	0.1064450	5.8594000E-02	1.816654
sin	0.1064450	6.2500000E-02	1.703120
recip	2.4413999E-02	4.1992001E-02	0.5813964
log	8.5937999E-02	5.6639999E-02	1.517267
exp	8.4960997E-02	5.8594000E-02	1.449995
cos_sin	0.1474610	7.0312001E-02	2.097238
G77 -O6 -funroll-loops -lcxml			
sqrt	1.65722597	0.0722659826	22.9323101
cos	0.146484017	0.0595710278	2.4589808
sin	0.124022961	0.0625	1.98436737
recip	0.0263669491	0.0439450741	0.599997818
log	0.25097701	0.0566411018	4.431005
exp	2.33691406	0.0576171875	40.5593224
cos_sin	0.27148509	0.0712890625	3.80822921
G77 -O6 -funroll-loops -lcxml -lcpml			
sqrt	0.0605469979	0.072266005	0.837835193
cos	0.130859017	0.0585939884	2.23331809
sin	0.120116979	0.060547024	1.98386264
recip	0.0253909826	0.0439450182	0.577789843
log	0.0917969868	0.056641046	1.62067986
exp	0.09375	0.0576169491	1.62712538
cos_sin	0.284179986	0.0703129768	4.04164362

RESOURCE

[1] TOP500 Supercomputer Sites
http://www.top500.org

リスト1

```
      INTEGER N,N1,REP
      PARAMETER (N=10000,N1=10001,REP=100)
      DOUBLE PRECISION A(N1), B(N1), C(N1)
      REAL*4 DUMMY(2)
C SQRT vs. VSQRT
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = sqrt(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vsqrt(B,1,A,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "sqrt   ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
```

リスト1　続き

```
C COS vs. VCOS
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = cos(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vcos(B,1,A,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "cos    ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
C SIN vs. VSIN
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = sin(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vsin(B,1,A,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "sin    ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
C RECIP vs. VRECIP
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = 1/(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vrecip(B,1,A,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "recip  ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
C LOG vs. VLOG
      DO I = 1,N
```

リスト1　続き

```
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = log(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vlog(B,1,A,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "log    ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
C EXP vs. VEXP
      DO I = 1,N
         B(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         B(I) = exp(A(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vexp(A,1,B,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "exp    ", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
C SINCOS vs. VSINCOS
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
         C(I) = I*1.0/N
      ENDDO
      T1 = DTIME(DUMMY)
      DO K = 1, REP
       DO I = 1,N
         A(I) = cos(B(I))
         C(I) = sin(B(I))
       ENDDO
      ENDDO
      T0 = DTIME(DUMMY)
      DO I = 1,N
         A(I) = I*1.0/N
         B(I) = I*1.0/N
      ENDDO
      DO K = 1, REP
      call vcos_sin(B,1,A,1,C,1,N)
      ENDDO
      T1 = DTIME(DUMMY)
      WRITE(6,*) "cos_sin", 1E6*T0/N/REP,  1E6*T1/N/REP,  T0/T1
      STOP
      END
```