Linux/Alpha 活用講座

清水尚彦　nshimizu@keyaki.cc.u-tokai.ac.jp

CXML活用法(2)
——BLAS互換の基本関数

Pentium 4が出荷され、いろいろなところでベンチマーク評価が行われています。400MHzのフロントサイドバスや、800MHzのRDRAMの威力による高い性能が期待されているわけですが、ベンチマークによっては、クロック周波数の割にあまり性能が出ていないものもあります。

一方、高性能計算の分野では、Pentium 4の用意した機構はかなり効果的です。特にメモリスループットが性能を制するようなアプリケーションでは、マルチバンク構成のRDRAMの効果とバスの高速性によって、ものによってはクロック周波数を超える性能向上も見られるようです。その反面、メモリの価格低下が著しくなり、これからは大量のメモリを搭載するマシンが増えそうなことを考えると、32ビットのアーキテクチャとして登場したこのプロセッサは、ちょっと物足りなさを感じるのも事実です。

x86アーキテクチャで用意された拡張機構を使えば、Linuxでは一応64Gbytesまで物理メモリへのアクセスができるようになってはいます。しかし、1つのプロセスが「素直に」使える範囲は、あくまでも「4Gbytes」という論理空間の中でしかないのは、アドレスを指定するレジスタが32bitであるから仕方ありません。

AMDは、Athlonの64ビット拡張を考えていますが、IntelはIA-64で64ビットに進むことにしていることから、メモリアドレスの能力不足が大問題になる可能性があると私は思っています。

もちろん、「1台で大きなメモリを扱えなくても、クラスタにすればいい」とも言えますが、複数のパラメータで何度も流すような処理を行う場合には、クラスタのプログラムを書くよりは、1台ずつ別のパラメータで複数台のPCを動作させたいと思うユーザーのほうが多いのではないでしょうか？

このようなことを考えると、Intelがターゲットとしているユーザーがどこにあるのか、もう1つ分かりにくいような気がします。もちろん、ASCI(Accelerated Strategic Computing Initiative)プロジェクトのようなものを対象にするというなら、Pentium 4は素晴らしい素材だと思います。

Alphaは、いまだに0.25μmのEV67の世代のプロセッサしか出荷していませんから、クロック競争には完全に乗り遅れていますね。現在1GHzを超えるクラスのプロセッサも評価中とのことですが、0.18μmのEV7が製造ラインに投入されたという情報もあり、結局のところEV68が日の目を見ることなく、GHzプロセッサの登場は、今年後半か来年となりそうな感じがします。

この戦略は、32ビットプロセッサとは直接競合をせず、64ビットになってからゆっくり戦うというように見えます。だとすると、世の中の多くの人達が32ビットでも十分満足することを考えると、これは自らをニッチ市場に追い込むような戦略になりますね。

A 研究室の近況

長いこと愛用していたEB164が最近機嫌を損ねていて、ときどき突然プロセッサ負荷が増大して他のマシンへのセッションが切れてしまうようになりました。このマシンはディスクレスとして運用しているので、NFSやTELNETの接続が切断されると役に立たないので困っていました。

負荷の増大のときに何が起きているのか把握しようと、`top`コマンドなどを使ってモニタをしていても、負荷が増大すると`top`コマンドさえ止まり、それが収まると改めて表示が行われたりしますが、ログに何も残っていないので、結局この原因は何なのか分からずじまいです。

ディスクレスマシンが接続するサーバ機は、Debian GNU/Linux potatoの正式リリース後、アップグレードをしていないので、これを機会に(?)この辺で1回サーバ側のアップグレードを行っておこうと考えました。

Debian GNU/Linuxでは次のように、aptコマンドでパッケージ管理を行います。

```
# apt-get update
# apt-get upgrade
```

ネットワークに接続してパッケージのURLを登録しておけば、上の2つのコマンドでパッケージが最新版に入れ替えられるはずです。ところが、このコマンドを実行することによって、pLaTeX／pTeX関係のプログラムが動かなくなるという副作用が生じてしまいました。pLaTeXを日常の仕事に使っている私にとっては死活問題です。

問題は、ptex-binのバージョンとplatexの要求するバージョンが異なることでした。これは、platexのソースを取ってきて依存関係の調整をして、再構築を行うことで対応しました。

```
# apt-get source platex
```

と実行すると、パッケージのソースが展開されますから、展開されたディレクトリに移動して、debian/controlファイルを編集します。今回は、ptex-binのバージョンを2.1.8-9から2.1.8-8へ下げました。その後、

```
# debian/rules binary
```

としてパッケージを作成し、インストールします。これで無事TeX関係のファイルもアップグレードできました。

●主力マシンの変更

さて、サーバ側のアップグレードを行っても、やっぱり突然負荷が増大する現象は収まりません。原因も分からないため、取りあえず「気分も変えよう」と主力マシンを変更することにしました。使用するのは、UP1100のマザーボードに600MHzの21264Aを載せたマシンです。これは、HITという会社から購入したもので、忘れてしまうほど昔に発注を出していたのに、大学の予算執行の関係で遅れ、昨年末にやっと納品されたものです。

このマシンのマザーボードは、AMDがAthlon用に開発したチップセットを流用してコストダウンを計ったもので、Athlon並とは言わないまでも、それなりに競争力がありそうな販売価格となっています。

その意味でも、Athlonとの性能比較が興味深いところですが、現在メモリは64Mbytesしか載っておらず、取りあえずの用途に使用するための最低限の環境なので、ベンチマークはまたの機会とさせていただききます。64Mbytes程度のメモリがあれば、そこそこのことができた時代は過ぎ去り、今では、この程度のメモリが搭載されていても、ほとんど常にスワップしまくるような状態になってしまいます(もちろんXを使わなければいいのですが)。

このままでは、さすがの私も困るので、現在メモリを発注しています。UP1100は、PC100のECC付きSDRAMを使用することは分かっているので、秋葉原辺りで探してくれば、早くて安く上がるのですが、研究室で使用するマシンですから、そういうわけにもいきません。大学で購入する場合には、見積りなどを添付して依頼せねばならないため、高くついてしまいます。それだけでなく、発注後の連絡も何もないままなので、いつ入荷するのか見当もつきません。結局この原稿を執筆するまでには間に合いませんでした。このような状態なので、このマシンもしばらくはお遊び程度にしか使えませんね。

●Vine Linux 2.1 Alpha版

このマシンには、SuSE Linux 6.4がインストールされていたため、日本語環境は用意されていませんでした。そこで、「同じRPMでパッケージを管理しているKondara MNU/LinuxかVine Linuxから必要なファイルだけ持ってくればいいだろう」と気楽に考えていました。ところが、Kondaraのパッケージはftpサイトに見つからないし、VineのRPMは依存関係のトラブルでほとんどインストールできません。それでも、「rpm --nodeps -i xxx」と「rpm --force -i xxx」を使い分けて、無理矢理インストールして、普段使うものは何とか入ったようなのですが、残念なことにvi系のエディタjvimがどうもうまく動きません。

Debianの初期にも、今回と同じようにjvimのカーソルがうまく動かないことがあったのですが、原因がライブラリだったか、コンパイラオプションだったのか、そのとき直した経緯を忘れてしまったので、安直にjelvisのソースを取ってきてjelvisをインストールして済ませています。

Debianでは安定していたNetscapeも、SuSEではまともに動かないし、Mozillaも安定性が悪いので、これだけで仕事をするのはかなりしんどいところです。これが何に起因するのかは確かめていませんが、素直にVineに変更したほうが得策かもしれません。……と、ここまで書いていて、やっぱり実用的に使うのは難しいと判断し、思い切ってVineに変更しました。ずっと使ってきたDebian を使わなかったのは、前出の

pLaTeXの問題と、Compaqの出しているツール類を利用するにはRPMの方が簡単そうだからです。

Vineも、Alpha版は正式なリリースではないようなので、問題があってもVineプロジェクトの責任ではありません。あくまでも試用レベルと認識しています。私としては、Kondaraでも良かったのですが、FTP版がなかったので断念しました。

VineはCD-ROMからブートできないのですが、SuSEがハードディスクにあるのでこのカーネルを用いてCD-ROMをルートファイルとしてブートします。私のマシンではCD-ROMは/dev/hdbであることをSuSEのブートアップのメッセージから確認し、SRMコンソールのプロンプトから

```
boot -fl "root=/dev/hdb"
```

とします。ハードディスクにブートできるカーネルがなく、カーネルをフロッピーディスクから読み込みたい人は、他のLinuxと同様に、フロッピーディスクにカーネルを書き込んで、フロッピードライブにセットしてから、

```
boot -fl "root=/dev/hdb" dva0
```

としてください。Xベースのインストーラが立ち上がり、インストールが始まります。パーティションの切り方を考えていると、インストーラはその先に進んでディスクのフォーマットをしていました。

しかし、64Mbytesではいかにもメモリが少ないようで、Xが動き出すとスワップしっぱなしになります。ウィンドウマネジャを変えればいいのだとは思いますが、取りあえず我慢して先に進みます。インストールが終了して、リブートすると、ネットワークカードが自動認識されていないことに気付きました。これはlinuxconfコマンドを使って、ロードするモジュールを手動で指定して対策しました。

次にエディタのjvimですが、やはりまともに動きません。そこで、jelvisをソースからコンパイルしてインストールしました。ブラウザにはw3mやMozillaがあるのですが、Mozillaが動かなかったので、CompaqのWebページからNetscapeをダウンロードしてインストールしました。これはTru64互換パッケージとコンフリクトしますが、--forceオプションを付けて強制的にインストールしてしまいました。

このほかに、インストールしたいパッケージとして、当然ながらCompaq CとCompaq Fortran、CXML(Compaq eXtended Math Library)があります。CXMLは、執筆時点の最新版はVer. 5.0のベータ版となっています。このバージョンは、Cからのリンク時にFORTRANのランタイムライブラリを必要としなくなるなど、ポータビリティが向上しています。

試用したVineのAlpha版には、g77もgnuplotもoctaveもなく、普段の道具としては若干物足りないものでした。これらはRed Hat Linux 6.2からRPMパッケージを持ってきてインストールして対応します。正式リリースまでにはVine Plusを含めきちんと動くものが出荷されると思います。

Alpha版のディストリビューションは、ほかにもSlackwareなど増えつつあり、選択肢は多くなってきています。いろいろとパッケージ管理ツールが出てはいるのですが、同じツールを使っていても他のディストリビューションからのパッケージを使ったりするのは結構手間がかかります。また、ディストリビューションによっては、Alphaでは未確認のパッケージが収録されていることもあり、思ったように動かないケースもあるようです。それならいっそのこと、tarでまとめただけのSlackwareでもいいのではないかと思うことも多々あります。

A CXMLの使い方

さて今回は、CXML活用法の第2回をお送りします。第1回(本誌2001年2月号)では、ベクトル算術演算関数の使い方を説明したのですが、今回はBLAS(Basic Linear Algebra Subroutine)互換の基本関数(レベル1)を説明します。そして実際に、Netlib(記事末のRESOURCE[1]を参照)が配付するBLASとCXMLとの性能比較を行ってみます。BLASやLAPACK(Linear System and Eigenprobrem Solvers)は、演算の型の異なるルーチンがたくさん用意されていますが、本稿では、特に断らない限り倍精度の演算を対象にします。

なお、CXMLのオリジナルドキュメントはCompaqのサイト([2])から入手できます。

●IDAMAX(n, x, incx)

IDAMAXは、倍精度の1次元配列(ベクトル)の中から、絶対値の大きな要素の最初のインデックス番号を返します。単精度、単精度複素数、倍精度複素数の関数はそれぞれISAMAX、ICAMAX、IZAMAXになります。IDAMAXの引数を表1に示します。

この関数をベンチマークプログラムでテストしてみましょう。Compaq XP1000(Alpha 21264-500MHz搭載)でリスト1をg77で実行した結果は表2のようになりました。これは、1要素を検索するのにかかる時間をns(ナノ秒)単位で表したものです。IDAMAXでは、CXMLの性能は必ずしも優位とは言いがたい結果になりました。要素数が多くなると性能が劣化するのは、キャッシュミスとTLB(変換バッファ)ミスが多くなるからです。

表1　IDAMAX、DASUM関数の引数

引数	内容	
n	ベクトルの要素数(nが0か負の場合、関数は0を返す)	
x	倍精度の1次元配列 要素数は「1+(n-1)*|incx|」以上なければならない	
incx	要素の並びを制御する変数	
	incxが正数のとき	i=1,nに対して「x(1+(i-1)*incx)」の順に実行
	incxが1のとき	x(1)から連続して実行する
	incxが0のとき	x(1)に対してのみ実行するため、常に1を返す
	incxが負数のとき	「x(1+(n-i)*|incx|)」の順に実行

表2　IDAMAX関数のベンチマーク結果(単位ns)

要素数	BLAS	CXML
1000	10	12.3
10000	12.5	14.1
100000	18.4	17.2
1000000	30.9	33.3
10000000	33.5	36.6

表3　DASUM関数のベンチマーク結果(単位ns)

要素数	BLAS	CXML
1000	8.1	4.1
10000	8.2	4.1
100000	9.6	4.8
1000000	16.2	7.8
10000000	17.5	8.3

●DASUM(n, x, incx)

この関数は、倍精度の1次元配列(ベクトル)の要素の絶対値の和を返します。単精度、単精度複素数、倍精度複素数の関数は、それぞれSASUM、SCASUM、DZASUMになります。DASUMの引数を表1に示します。

リスト1をDASUM用に手直しして、こちらも要素ごとの計算時間を求めてみると、結果は表3のようになります。IDAMAXとは異なり、こちらはCXMLを呼び出すことによって約2倍高速になっています。要素数が多くなると性能が落ちるのは、同じ傾向にあります。

●DAXPY(n, alpha, x, incx, y, incy)

このサブルーチンは、倍精度のベクトルとスカラーの積和を計算します。このような計算の場合、通常「y=alpha*x+y」という演算を行うため、「AX Plus Y」を略して「AXPY」という名前になっています。単精度、単精度複素数、倍精度複素数の関数はそれぞれSAXPY、CAXPY、ZAXPYになります。DAXPYの引数を表4に示します。

例によって、要素ごとの計算時間を求めてみましょう(表5)。DAXPYも、要素数が多くなると、CXMLは倍以上の高速なライブラリになっていることが分かります。DAXPYは、xとyの2つのベクトルを読み出してyに書き込んでいるので、システムのメモリスループットを計算することができます。1要素の計算に24bytesのデータ移動があるので、1000000要素のときの値を用いると、1.1Gbytes/sのスループットが得られていることになります。

リスト1　ベンチマークプログラム

```
      PARAMETER (MAXN=10000000,NREP=10)
      DOUBLE PRECISION AA(MAXN)
      REAL*4 DUMMY(2)
C
      DO I = 1,MAXN
         AA(I) = I
      ENDDO
      NSTART = 1000
100   CONTINUE
      T1 = DTIME(DUMMY)
      DO J=1,NREP*MAXN/NSTART
            I=IDAMAX(NSTART,AA,1)
      ENDDO
      T0 = DTIME(DUMMY)
      WRITE(6,*) NSTART, T0*1E9/(NREP*MAXN), "nS"
      NSTART = NSTART * 10
      IF(NSTART .le. MAXN) go to 100
      STOP
      END
```

表4　DAXPY関数の引数

引数	内容	
n	ベクトルの要素数(0か負の場合、関数は0を返す)	
alpha	倍精度のスカラー数	
x	倍精度の1次元配列 要素数は「1+(n-1)*|incx|」以上なければならない	
incx	要素の並びを制御する変数	
	incxが正数のとき	i=1,nに対して「x(1+(i-1)*incx)」の順に実行
	incxが1のとき	x(1)から連続して実行
	incxが0のとき	x(1)に対してのみ実行するため、常に1を返す
	incxが負数のとき	「x(1+(n-i)*|incx|)」の順に実行
y	倍精度の1次元配列 要素の数は「1+(n-1)*|incy|」以上なければならない	
incy	要素の並びを制御する変数	
	incyが正数のとき	i=1,nに対して「y(1+(i-1)*incy)」の順に実行
	incyが1のとき	y(1)から連続して実行する
	incyが0のとき	y(1)に対してのみ実行するため、常に1を返す
	incyが負数のとき	「y(1+(n-i)*|incy|)」の順に実行

表5　DAXPY関数のベンチマーク結果(単位ns)

要素数	BLAS	CXML
1000	5.9	4.8
10000	20.0	11.3
100000	27.5	13.6
1000000	48.0	21.9
10000000	51.3	23.3

●DCOPY(n, x, incx, y, incy)

このサブルーチンは、倍精度のベクトルの(xからyへの)コピーを行います。単精度、単精度複素数、倍精度複素数の関

数は、それぞれSCOPY、CCOPY、ZCOPYになります。DCOPYの引数を表6に示します。

要素ごとの計算時間は表7の通りです。これを見ると、CXMLは要素数が多くなったときに高い性能を持つことが分かります。ベクトルのコピーではあまりプログラムを工夫する余地はないので、それほど大きな性能差にはなっていません。

●DDOT(n, x, incx, y, incy)

この関数は、倍精度のベクトルの内積を返します。単精度、単精度複素数、倍精度複素数の関数はそれぞれSDOT、CDOTC、ZDOTC、CDOTU、ZDOTUになります。CDOTCとZDOTCは共役を取った内積でCDOTUとZDOTUは共役を取らない内積の計算をします。また、異なる精度のベクトルとの内積も定義されていて、SDSDOT、DSDOTとなっています。DDOTの引数を表6に示します。

要素ごとの計算時間は表8の通りです。DDOTは16bytesのデータを読み出すので、CXMLでは、大きな配列に対して約1Gbytes/sのスループットになっているのが分かります。

表6　DCOPY、DDOT、DNRM2、DSWAP関数の引数

引数	内容	
n	ベクトルの要素数。0か負の場合には関数は0を返します	
x	倍精度の1次元配列 要素の数は「1+(n-1)*|incx|」以上なければならない	
incx	要素の並びを制御する変数	
	incxが正数のとき	i=1,nに対して「x(1+(i-1)*incx)」の順に実行
	incxが1のとき	x(1)から連続して実行
	incxが0のとき	x(1)に対してのみ実行するため、常に1を返す
	incxが負数のとき	「x(1+(n-i)*|incx|)」の順に実行
y	倍精度の1次元配列 要素の数は「1+(n-1)*|incy|」以上なければならない	
incy	要素の並びを制御する変数	
	incyが正数のとき	i=1,nに対して「y(1+(i-1)*incy)」の順に実行
	incyが1のとき	y(1)から連続して実行
	incyが0のとき	y(1)に対してのみ実行するため、常に1を返す
	incyが負数のとき	「y(1+(n-i)*|incy|)」の順に実行

表7　DCOPY関数のベンチマーク結果(単位nS)

要素数	BLAS	CXML
1000	3.7	4.4
10000	11.1	11.2
100000	16.8	13.9
1000000	24.9	17.5
10000000	25.7	16.8

表8　DDOT関数のベンチマーク結果(単位nS)

要素数	BLAS	CXML
1000	8.1	3.5
10000	9.1	6.4
100000	16.8	9.9
1000000	29.0	16.7
10000000	29.7	16.8

表9　DNRM2関数のベンチマーク結果

要素数	BLAS	CXML
1000	27.8	16.6
10000	27.9	16.2
100000	32.0	16.2
1000000	46.3	16.9
10000000	48.6	17.0

●DNRM2(n, x, incx, y, incy)

この関数は、倍精度のベクトルのユークリッドノルムを返します。単精度、単精度複素数、倍精度複素数の関数は、それぞれSNRM2、SCNRM2、DZNRM2になります。DNRM2の引数を表6に示します。

要素ごとの計算時間は表9の通りです。CXMLは、要素数が多い場合にはBLASの3倍近い性能を出していることが分かります。

●DROT(n, x, incx, y, incy, c, s)

この関数は、倍精度の2つのベクトルx、yの対応する要素を座標として、ギブンス平面上の回転を行います。cとsで表される引数は、それぞれ回転角のコサインとサインをとった数値とします。単精度、単精度複素数、倍精度複素数の関数は、それぞれSROT、CSROT、ZDROTになります。DROTの引数を表10に示します。

要素ごとの計算時間を表11に示します。cとsの値を何か決めないといけないので、ここでは45度の回転を行うことにし

表10　DROT関数の引数

引数	内容	
n	ベクトルの要素数。0か負の場合には関数は0を返します	
x	倍精度の1次元配列 要素の数は「1+(n-1)*|incx|」以上なければならない	
incx	要素の並びを制御する変数	
	incxが正数のとき	i=1,nに対して「x(1+(i-1)*incx)」の順に実行
	incxが1のとき	x(1)から連続して実行
	incxが0のとき	x(1)に対してのみ実行するため、常に1を返す
	incxが負数のとき	「x(1+(n-i)*|incx|)」の順に実行
y	倍精度の1次元配列 要素の数は「1+(n-1)*|incy|」以上なければならない	
incy	要素の並びを制御する変数	
	incyが正数のとき	i=1,nに対して「y(1+(i-1)*incy)」の順に実行
	incyが1のとき	y(1)から連続して実行
	incyが0のとき	y(1)に対してのみ実行するため、常に1を返す
	incyが負数のとき	「y(1+(n-i)*|incy|)」の順に実行
c	倍精度のパラメータ	
s	倍精度のパラメータ	

表11　DROT関数のベンチマーク結果

要素数	BLAS	CXML
1000	27.8	16.6
10000	28.3	16.2
100000	32.7	16.2
1000000	48.4	16.9
10000000	48.7	17.0

表12　DSCAL関数の引数(単位ns)

引数	内容	
n	ベクトルの要素数(0か負の場合には関数は0を返します)	
alpha	倍精度のスカラー数	
x	倍精度の1次元配列 要素の数は「1+(n-1)*|incx|」以上なければならない	
incx	要素の並びを制御する変数	
	incxが正数のとき	i=1,nに対して「x(1+(i-1)*incx)」の順に実行
	incxが1のとき	x(1)から連続して実行
	incxが0のとき	x(1)に対してのみ実行するため、常に1を返す
	incxが負数のとき	「x(1+(n-i)*|incx|)」の順に実行

表13　DSCAL関数のベンチマーク結果(単位ns)

要素数	BLAS	CXML
1000	2.9	2.6
10000	10.2	5.6
100000	11.7	8.6
1000000	21.1	11.9
10000000	22.9	12.4

表14　DSWAP関数のベンチマーク結果(単位ns)

要素数	BLAS	CXML
1000	7.5	7.9
10000	16.7	16.2
100000	22.3	17.5
1000000	37.3	29.5
10000000	38.7	32.3

ました。このサブルーチンは、「回転」といってもランク2の行列積を行うだけなので、実行時間の傾向はDNRM2とほとんど同じ結果になりました。

●DROTG(a, b, c, s)

この関数は、倍精度の2つの数値a、bを座標として、この点をX軸に移動するギブンス平面上の回転を行う回転の演算を求めます。cとsで表される引数は、それぞれ回転角のコサインとサインをとった数値となります。また、呼び出し後、aには回転後のX座標の値が入り、bには回転要素を再構成するための数値が入ります。 単精度、単精度複素数、倍精度複素数の関数は、それぞれSROTG、CROTG、ZROTGになります。

●DSCAL(n, alpha, x, incx)

このサブルーチンは、倍精度のベクトルのスカラー積を計算します。「x=alpha*x」という演算をするので前出のDAXPYのサブセットになっています。単精度、単精度複素数、倍精度複素数の関数は、それぞれSSCAL、CSCAL、ZSCAL、CSSCAL、ZDSCALになります。DSCALの引数を表12に示します。要素ごとの計算時間は表13に示す通りです。

●DSWAP(n, x, incx, y, incy)

この関数は、倍精度のベクトルの入れ替えを行います。単精度、単精度複素数、倍精度複素数の関数は、それぞれSSWAP、CSWAP、ZSWAPになります。DSWAPの引数を表6に示します。要素ごとの計算時間は表14の通りです。

まとめ

さて、これでBLASのレベル1の演算はほぼ説明しました。レベル1は、ベクトルに対する演算なので、演算量は少なく工夫する余地があまり多くはないのですが、こういった演算でもCXMLがDebianの配付パッケージBLASよりも、概ね高速であることが分かると思います。

原稿提出後に、EB164の不具合が判明しました。研究室の学生が、Windows 2000マシンに同じIPアドレスを勝手に付けたことが原因だったのです。こういった、運用面が原因となるトラブルは予想していなかったので、原因追求が難しくなってしまったようです。

R E S O U R C E

[1] Netlib
http://www.netlib.org/

[2] Compaq Documentation for Math Liblaries
http://www.compaq.com/math/documentation/index.html