Linux/Alpha活用講座

清水　尚彦　<nshimizu@keyaki.cc.u-tokai.ac.jp>

第13回
Compaq Cコンパイラを試す

AWindowsとAlpha

一瞬一瞬の先が読めないこの業界では月刊と言えども時事的な話題は難しいところがあります。アメリカの企業は業績や成長性をとても大事にするので利益の出ない事業に固執したりしません。ですから改革はドラスティックに行われます。

もういろいろな所でニュースを聞かれている方も多いと思いますが、マイクロソフトがWindows 2000の対応プロセッサからAlphaを落としました。これまでMIPSやPowerPCの対応を廃止してきた同社は、とうとう「IntelだけのOSとしてのWindows 2000」という位置付けに変えてきました。この背景にはCompaq社がWindows NT関係の技術グループを解散するという事情があるわけですが、選択肢が減っていくのは大変なことです。プロセッサ自体の開発そのものはファブ(工場)を持たないCompaq社にとってそれほどの負担にはならなくても、ソフトの技術グループの維持は見返りが少ないとなればさっさと撤退するクールさが、業績の低迷している現在は必要ということも理解できます。LinuxやTru64 UNIXに対する市場の要求はサーバ部門と数値計算と思われることが多いのですが、コンピュータの業界では大量に生産／出荷することで多大な開発費を賄うという場合も多く、一般ユーザー向けの製品なしでコストパフォーマンスを保ち続けるのは難しいのです。しかし、家庭用／業務用PCのハードウェアでの利益率はとことん下がっていることを考えると、ここに無理に留まるよりは、もう少し利益率の高い分野へと転身するCompaqの戦略は実は得策なのかもしれません。

ところが、一般のユーザーは数値計算もやらなければ家庭内に高性能なサーバも必要ありません。だとすると、Alphaの市場は縮退してしまうのかというと、EV7の時代には外部チップセットが少ないAlphaはゲーム市場を狙えるのではないかというような観測も出ていたりします。面白い見方だと思います。でも、市場があるところにIntelやAMDが手をこまねいているような気がしないので、どの市場も瞬く間に主戦場になるのでしょう。

そんな中、一歩退いた所にあるのが数値計算です。数値計算のユーザーは実はアーキテクチャにこだわることは少なく、安くて高速なプロセッサがあればどんどん使っていきます。ですから囲い込みがしにくく、ベンダーにはとっても手ごわい市場になります。日米の大手のコンピューターメーカーも、この市場で満足に利益をあげているところは少ないのではないかと私は疑っています。ここでは性能が高いことが付加価値になり、一番速いマシンは他社のマシンに比べコストパフォーマンスが悪くてもユーザーが付きます。その最たるものがベクトル機だったわけですが、クラスタシステムの普及によって、単体でのコストパフォーマンスが悪いマシンの需要が減る傾向になってきたので、競争はますます厳しいものとなってきています。コストパフォーマンスは利益率の低いPCと比較されることになってしまうので、IntelやAMDが頑張れば頑張るほどますます専用機メーカーは苦しくなります。

Windows 2000への対応が必要なければ、Alphaのマ

ザーボード上にAlphaBIOSやARC BIOSは不要なので、実は、LinuxやTru64 UNIXを中心にするという選択は、Alphaのインストール時の混乱を低減する可能性が高くなります。MILOはその役割を終えてSRMが標準コンソールとなる日が来るのでしょうか。

ニュースのあまりのタイミングの悪さに我ながらびっくりしますが、研究室のAlphaマシンのうち1台にWindows NT Terminal Server Edition(TSE)を導入しました。クライアントはWindows Based Terminal(WBT)とX端末を起動時に切り替えることができる、高岳製作所の「MiNT ACC」というマシンです。XFree86のサーバが対応していないグラフィックスカードを搭載しているXP1000のコンソールとして、X端末を主力に使っていますが、WBTもオフィスソフトが必要なときに役に立っています。実際、個人ユーザーが使う場合には、今や無料パソコンまで現れる時代ですからWindowsアプリケーション用に1台専用にマシンを購入しても金銭的には全然問題ありません。しかも、そのマシンにX端末のソフトをインストールしさえすれば、Linuxの端末として十分使い物になるので、サーバーにはコンソールもキーボードも接続しなくても十分役に立ちます。

CompaqのLinux/Alpha用Cコンパイラ

Linuxに注力するCompaq社からはFortranに続いてCコンパイラもβリリースされました。利用されたい方は次のURLをご覧ください。

```
http://www.unix.digital.com/linux/software.htm
```

今回はこのコンパイラをDebian GNU/Linuxで利用する方法を示し、簡単に評価してみたいと思います。

●インストール

Webページからダウンロードできる Compaq Cコンパイラのβ版はRed Hat Linux 5.2用になっています。その後、Red Hat Linux 6.0のパッチが出てパッケージのアップデートがされていますが、これをDebian GNU/Linuxで利用するには工夫が必要です。Red Hatのユーザーはこれを使うだけなので困ることはないでしょう。ここではDebianユーザーのために(私のことですが)インストールの方法を示します。今回のパッケージは、単純に"alien"コマンドを実行するだけでDebianのパッケージが作成できます。これは簡単だと思ってさっそく使って見ると、コンパイルが失敗します。実は、コンパイラの構成ファイルを変更する必要があったのでした。次に私がインストールした手順を示します。

Debianパッケージの作成

Compaqが配付するファイルはRed HatのRPMパッケージになっています。そこで、alienコマンドを実行してDebianパッケージに変更して、Debianで利用できるようにします。alienコマンドは、オプションを何も付けずに実行すると、RPMパッケージからDebianパッケージを作ってくれますが、バージョン番号が変わってしまいます。しかしローカルなパッケージなので、ここでは気にしなくて結構です。私がダウンロードしたパッケージの名前は"ccc-6.2.0-1.alpha.rpm"になっていました。バージョン番号はβリリース期間の間に変わる可能性が高いので、ダウンロードできるファイルの名前は、ここに挙げたものに限らないことに注意してください。alianコマンドは以下のように実行します。

```
$ alien ccc-6.2.0-1.alpha.rpm
```

パッケージができれば、次はdpkgコマンドでインストールしましょう。

```
$ dpkg -i ccc_6.2.0-2_alpha.deb
```

あっけないほど簡単にインストールが出来たと思います。ここまでの作業で一番時間がかかったのがWebページからのパッケージのダウンロードの時間でした。

コンパイラ構成ファイルの変更

変更するファイルは1つだけです。まず、コンパイラをインストールしたディレクトリに移動します。

```
$ cd /usr/lib/compaq/ccc-6.2/alpha-redhat-
linux/bin
```

この中に、コンパイラが利用するディレクトリを記述するファイル"comp.config"があります。変更はこのファイルに2つのディレクトリを付け加えることです。実はCompaqのCコンパイラは、システムにインストールされ

たgccのインクルードファイルやライブラリを利用するようになっていて、その場所を構成ファイルに明示的に示してやる必要があるのです。このファイルはテキストファイルなのですが、中身を見れば分かるように、インクルードファイルやライブラリを指示するコンパイラオプションを、ここにあらかじめ書いておくものです。このオプションには、直接インクルードファイルやライブラリがある絶対パスを記述してあります。絶対パスなので、インストールしている既存のコンパイラと、そのインストールの仕方に強く依存してしまうのです。Debian 2.1では"egcs-2.91.60"というバージョンのコンパイラを標準としていますから、このコンパイラの関連ファイルをcomp.configに記述します。

具体的には次の行をファイルの最後に付け加えます。comp.configは改行なしの1行のテキストファイルになっています。以下の行を付け加える場合も改行を入れないでください。

```
-SysIncDir  /usr/lib/gcc-lib/alpha-linux/
egcs-2.91.60/include -L/usr/lib/gcc-lib/
alpha-linux/egcs-2.91.60
```

リンカへのシンボリックリンクの変更

Compaq Cのリンカは RedHatをインストールした場合にリンカがある場所へのシンボリックリンクになっています。ところが、Debianにはこの場所にはリンカはないので実行できません。そこで、このシンボリックリンクを変更します。構成ファイルと同じディレクトリにおいて次のコマンドを入力してください。

```
$ ln -sf /usr/bin/ld ld
```

これで、インストールは終了です。コンパイラの起動はcccというコマンドで行います。

●コンパイラの試用

β版ということで、問題があればCompaqに報告する義務を持ちますが、使ってみないことには話は始まりませんから、試してみましょう。Cの数値計算ベンチマークとして、Alphaの世界で時々参照されるものとして、私が公開しているDGEMMのルーチンがあります。これはCで書かれているため、いろいろなアーキテクチャのマシンと比較するのに便利です。ソースコードはCD-ROMに収録しますが、他のプロセッサのベンチマーク結果は

http://shimizu-lab.et.u-tokai.ac.jp/pgm/gemm/

を参照してください。

このコードにはAlphaのgcc用に特別なアセンブラルーチンが入っています。Compaq Cではこのアセンブラルーチンが使えないので(文法が異なる)、ちょっとソースを変更してアセンブラを使わないプログラムとし、Compaq Cでコンパイルできるようにしました。

まずは、Compaq Cのコンパイルとベンチマークの結果です。同じルーチンを3回動作させてそれぞれ表示するようになっています。マシンはXP1000です。

```
$ ccc -DNOINLINECORE -DNOPREFETCH -O6 -tune
ev6 -Dev6 -DMAIN gemm.c -o gemm.ev6
$ ./gemm.ev6
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 581.988 MFLOPS( 3.436 sec)
Shimizu's DGEMM : 581.658 MFLOPS( 3.438 sec)
Shimizu's DGEMM : 581.823 MFLOPS( 3.437 sec)
```

次に同じソースコードをgcc(egcs-2.91.60)でコンパイル実行します。

```
$ gcc -DNOINLINECORE -DNOPREFETCH -O6 -Dev6 -
DMAIN gemm.c -o gemm.gccO6
$ ./gemm.gccO6
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 381.669 MFLOPS( 5.240 sec)
Shimizu's DGEMM : 381.669 MFLOPS( 5.240 sec)
Shimizu's DGEMM : 381.740 MFLOPS( 5.239 sec)
```

比較すると50%もの大きな差が出ています。

次にgccのインラインアセンブラを使ったプログラムをコンパイルして実行します。また、このプログラムにはソフトウェアによるデータプリフェッチも入っています。

```
$ gcc -O6 -Dev6 -DMAIN gemm.c -o gemm.xxx
$ ./gemm.xxx
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 592.763 MFLOPS( 3.374 sec)
Shimizu's DGEMM : 593.106 MFLOPS( 3.372 sec)
Shimizu's DGEMM : 592.934 MFLOPS( 3.373 sec)
```

驚くべきことにCompaq Cはこの手で最適化しているルーチンとほぼ同等の結果を出すではありませんか。

では手動で入れているプリフェッチを止めたらgccの性能がどうなるか見てみましょう。

```
$ gcc -O6 -DNOPREFETCH -Dev6 -DMAIN gemm.c -o
gemm.nopre
$ ./gemm.nopre
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 568.585 MFLOPS( 3.518 sec)
Shimizu's DGEMM : 568.427 MFLOPS( 3.518 sec)
Shimizu's DGEMM : 568.585 MFLOPS( 3.518 sec)
```

このようにプリフェッチを止めるとCompaq Cよりも性能が落ちてしまいます。

これらのことからCompaq Cのコード最適化の能力はかなりのものがあることが推測できます。さらに、得られた性能から、データプリフェッチを自動挿入していることも予想できます。私のコードではプリフェッチは単純なものと、ちょっとループの離れたものとの2種類を入れているのですが、単純なものだけにすれば、同等の性能になるのかもしれません。

次に、XP1000で高速な結果が得られたコードを21164A-600MHz(VT-5/600)で実行してみます。

```
et7fl: {3} ./gemm.ev6
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 321.945 MFLOPS( 6.212 sec)
Shimizu's DGEMM : 321.793 MFLOPS( 6.215 sec)
Shimizu's DGEMM : 322.249 MFLOPS( 6.206 sec)
et7fl: {4} ./gemm.xxx
m:1000 n:1000 k:1000
Ail_max 48, Blj_max 24, A_row_block 341
Shimizu's DGEMM : 337.258 MFLOPS( 5.930 sec)
Shimizu's DGEMM : 336.041 MFLOPS( 5.952 sec)
Shimizu's DGEMM : 336.704 MFLOPS( 5.940 sec)
```

この2つのコードから言えることは「Compaq C侮りがたし」ということでしょう。21164Aでは、アウトオブオーダー実行をサポートしていないため、命令の並びが極めて重要になります。その条件の厳しいプロセッサで、手で命令を並べたものとほぼ同等の結果を得られるというのは素晴らしいコンパイラと言えるでしょう。

ベンチマークプログラムについて補足

ここで評価に用いたベンチマークは、密行列の倍精度乗算を行うプログラムとコンパチブルなサブルーチンをCで作成したものです。このルーチンはLINPACKベンチマークのTPPなどで良く使われているため、ある種の演算にとっては非常に重要なものです。いくつかの実装がありますが、他のものに比べて絶対性能が低いと思われた方もいると思い若干の補足説明をします。

一般に数値計算ユーザーからみて、アドレス計算のバッファ(いわゆるTLB)で指定できるアドレス範囲が思いのほか狭いプロセッサがある、という点が見過ごされがちです。大容量のTLBを用意するのは今のプロセッサ技術では何でもないにも関わらず、こういう重要な点がほっておかれるのはおかしいので、このベンチマークではTLBが少ない、もしくはTLBの性能が悪いプロセッサではあまり性能が出ないようにしています。もし、TLBミスを減らしたければ、行列を一旦小さなサブ行列にコピーして計算すれば、明らかにミスは大幅に減って性能は向上します。しかし、従来のスーパーコンピュータの多くが仮想記憶をを持たなかったように、数値計算ユーザーにとって、TLBミスなんて邪魔物以外のなんでもなく、プロセッサとOSが賢ければ余計なコピーは不要なはずです。

おわりに

Compaq Cのベンチマークを実行していくと、思ったよりずっと優れた性能を出すのでびっくりしています。これだけの性能を持つコンパイラがあれば、アセンブラを直接触る必要はあまりなくなるでしょう。Linuxならくだらない管理の手間(例えば同時利用ユーザー数のライセンスで頭を悩ませるなど)をかけずに自由に使えるので、柔軟な環境を作ることができます。Cを中心とした数値計算のユーザーも、Compaqのますますの Linuxへの注力に本当に感謝していいと思います。

後は、この優れたプロセッサがいつまでもユーザーサイドに立って安価に安定した高性能を供給してくれることを願いましょう。