東海大学の清水です。

99年5月号の原稿になります。よろしくお願いします。
COMPAQから新しい発表があったことを受けて当初予定から
内容を変更しました。
また、できれば
http://www.jp.debian.org/debian/dists/slink/main/{disks-alpha,binary-alpha,source}
http://www.jp.debian.org/debian-jp/dists/slink-jp/main/{binary-alpha,source}
の内容をCDROMに載せて頂けますか?

性能解析ツールlprobeの紹介

清水尚彦 nshimizu@et.u-tokai.ac.jp

最近、学科の連絡なども電子メイルを中心に連絡を取り合うようになってきて
ようやく世間並みにペーパーレスになりつつあるのですが、困ったことに
MSWordやら一太郎やら果てはMacintosh binaryの添付ファイルが本文になっている
メイルが来ることも多くなり文句を言っても他に方法を知らないといわれたりします。
こういった場合に平文テキストを強く主張してもなかなか受け入れてもらえず
困ってしまいます。普段PC-Unixで生活していて自分の作る書類にはあまり困って
いないだけにWinodwsを使わないほうが悪いといわんばかりの風潮には反発を覚えて
しまいますね。とはいっても何とかしなくては仕事に支障が出てしまいます。
対策としては

1. Linux+WineでMSWordを動かす。(もしくはWordが読めるLinux用のワープロを導入する)
2. 日常の環境をMSWindowsに変更する
3. 二台のＰＣを切り替えて使う

などがありますが、どれもスマートではありません。1はだいぶましですが、仕事に
使うほどの安定度が確保できるか評価が難しいところです。
こんな時に大型コンピュータの世界ではＶＭを導入して同時に両方のＯＳを動作させる
のですが、i386アーキテクチャは自己仮想化をサポートしないのでＶＭの導入は困難です。
Alphaでも、ハードウェアリソースに書き込みのみのレジスタがあったりするのでそのまま
仮想化するのはオーバヘッドが大きそうですが、やってみる価値は十分にありそうですね。

私はすぐにでも必要なので、もう少し現実的な方法としてInterixをx86のWinodwsNTの
上にのせてそこに普段使うアプリケーションを乗せてしまうという方法を検討中です。
本当はWinFrameを使ってリモートのＮＴのアプリケーションを使った方がスマ
ートですが、WinFrameはとんでもない値段が付いているので個人では手が出ま
せん。ＮＴをメインとするとしても
普段使っているTeXやtgifやjvimなどを捨てたくはない
ですよね。TeXやjvimはNTでも動作するものがあるので、残りはtgifです。これを
ソースからＮＴで動かすように変更するのは大工事になるのですが、Interixなら
かなり簡単に移植できるのではないかと密かに期待しています。
tgifははまだですが少しだけNTを使いはじめました。しかし思った以上に安定度が悪く
一日に何回かはリセットボタンを押すはめになっています。本当にこんな
ひどいものをみんな使っているのだろうかととても不思議ですが、私の
システムだけが不安定なのだろうか？もっとも固まるのはNetscapeを使っているときが
ほとんどのような気がするのでマイクロソフト社の陰謀かもしれません :-)
InterixはAlphaでも動作しますが、ＮＴなのであくまでもアドレス空間は
３２ビットでしか動きません。（もちろん、プログラムは６４ビットレジスタを使えます。
ここでの３２ビットアドレスとは上位３２ビットが０のアドレスのことです。）そこで、
数ＧＢのメモリが必要な人がAlphaのマシンにＮＴを入れてもあまり嬉しいことはありません。

Linuxの2.2.Xのカーネルが出てきました。
Linux/Alphaのインストールという目でみると2.0.Xに比べカーネル作成時にGenericという
選択肢が増えたのが大きい変化です。これはマザーボードを
カーネルが自動認識するものでマザーボードに強く依存した従
来のカーネルに比べてインストール時のトラブルが減る可能性があります。
(もっともMILOはまだボード別になっているのでまだ意識は必要です。SRMからMILO
を使わずにインストールすればGenericだけですむかもしれません。)
Genericは2.1の系列にも入っていたようでRedHatのサイトにあるカーネル(2.1.124)には
Genericのものも置いてあります。
OpenBSDやNetBSDでは(Digital Unixでも)自動認識していたのでやればできる
ことだったはずですが、これまでなされていなかったのが不思議ですね。

年明け早々にCOMPAQからAlphaコミュニティに大きなプレゼントが来ました。
それはIprobeと呼ばれるパッケージです。このパッケージはAlphaの性能を
測定するための特殊なカウンタを使うためのフロントエンドになっていて
簡単な設定で様々な性能上の特性をとらえる事が出来るようになります。

ftp://metalab.unc.edu:/pub/Linux/devel

から入手できます。
ファイルは幾つかあり、

iprobe_suite-4.0-1.alpha.rpm
iprobe_suite-4.0-1.src.rpm
iprobe_suite-4.0-driver.o.gz
iprobe_suite-4.0.lsm
iprobe_suite-4.0.tgz

のようになっています。
今回はこのIprobeの使い方と何が出来るのかについてまとめてみたいと
思います。

1. Multia(UDB)のトラブルシューティングとDebian 2.1のインストール

後述のIprobeと
2.2.Xのカーネルを試すためにトラブル中のMultia(UDB)を復活させようとWEBサイトを
探していたところ(時期的に現役のマシンはAS200 4/100を含めて学生が常時
使っていて私が自由に使えるAlphaのマシンは少ないのです)

http://brouhaha.com/~eric/computers/udb.html

にMultiaの情報だけでなくマザーボードのマニュアルも置いてありました。
その中のTrouble Shootingの項目から私のトラブルに該当するものがあり
ました。現象としてはブートのモードを切り替える時にそのまま固まって
しまうというもので、私はBIOSのEEPROMイメージが壊れたものだと思って
いたのですが、マニュアルによるとバックアップ電池が切れた場合の現象
とのことでした ^_^;
(この程度の問題ならユーザーズマニュアルに載せておいて欲しいなぁ)
さっそくパソコンショップに出かけて「これと同じ型の電池は
ありませんか?」と尋ねたところMacintosh用のバックアップ電池と同型だそう
で案外簡単に手に入りました。
ということで、無事Multia君も生き返り新しいカーネルを試せると
思ったのですが、思わぬ所に伏兵がいました。
実は長いこと使わなかったうちにMultia君のフロッピードライブの調子が
すっかりおかしくなってしまったのです。
フロッピーからデータを読み出そうとするとウンウンうなってI/Oエラーと
なってしまいます ;_;
それでも何回かやっているうちにMILOだけは起動できたのでHDDにあった昔の
パーティションに新しいMILOを入れることができました。
(昔のパーティションからブートすればよいではないかと気が付いた人は鋭い
ですね。実はMILOのファイル名にちょっと細工していたのですっかりファイル
名を忘れてしまってブートまでこぎ着けなかったのでした。MILOが立ち上がっ
てファイル名が確認できたのでとりあえず古いバージョンのLinux/Alphaは動
くようになりました。)

RedHatのgenericカーネルも何回かに一回は立ち上がりペンギンの絵が
画面に出て来ました。ところが、ramdiskのフロッピーはどう頑張っても
読めないのです。
ということで、今回は残念ながら時間切れです。次回にSRMコンソールからデ
ィスクレスブートをするようにしてMultia君の再生に再挑戦します。

というはずだったのですが、フロッピーが読めないのも不便なのでいろいろ試して
みました。古いパーティションのカーネルを使って一旦Linuxを立ちあげた後
ddコマンドをシェルのループ機能で連続発行してフロッピードライブを動作させ
続けているとしばらくするとドライブの音が静かになってきたと思ったらちゃんと
読めるセクタが増えているではありませんか!
喜んでMILOからカーネルを読み込んでramdiskのイメージを読み込ませました。
調べてみるとどうにかこの弱ったドライブで読めるのはソニーのディスクだけで
マクセルや富士フィルム、ＤＹＳＡＮなど他のメーカー製はぜんぜん読めない

ようです。（Multia 以外のハードでは問題なく読み書きできるのでこれらの
メーカーが悪いわけではありません。）

フロッピーが読めたので RedHat 5.2 のインストールに挑戦です。
RedHat のミラーサイトからダウンロードした ramdisk を使って MILO から

```
boot sda1:generic.gz root=/dev/fd0 rw load_ramdisk=1 prompt_ramdisk=1
```

としてインストールを開始します。ここで sda1 としているのは前述のとおり
私は一旦古いシステムを立ちあげたのでその時に generic.gz を MILO のパーティションに
入れていたからこう指定しています。カーネルをフロッピーに入れている人は
「fd0:generic.gz」に読み替えてください。
カーネルが読み出されると ramdisk のフロッピーを要求してきますので入れ替えて
キーを何か押すと ramdisk の読み出しが始まり、インストール用のシステムが立ち上がり
ます。私の Multia には CDROM がついていないので、ftp を使ってインストールを試みました。
ところが、IP アドレスを直接指定しても DHCP を指定してもどちらもエラーになって
ネットワークを有効にできません。Alt-F3 でインストールの途中を確認すると ifconfig
はちゃんとできていそうなのになぜかスクリプトがエラーを検出します。
今回は時間がないので RedHat 5.2 のトライアルはこれでおしまいにしました。
落ち着いたらゆっくり原因を探ってみたいと思います。

次に Debian プロジェクトから出ている slink バージョン(Debian 2.1)の
インストールを試してみました。執筆時点ではまだ安定版にはなっていません
がフリーズ版になっていてリリースへの秒読み段階です。
この雑誌が販売されるころには大きく事情が変っているかもしれませんが、
御了承ください。
CDROM があれば簡単ですが、slink のアルファ版の CDROM なんてそうそう簡単に
は手に入らないので FTP によるインストールを行います。
インストーラが NFS で配布ファイルを含むディレクトリをマウントできる
ようにもう一台のマシン(x86 で OK)を設定しておいてください。
Alpha を持っている方はそれ一台しかない場合は少ないと思われるので
一台だけの場合について言及しませんが、もし単独でインストールしたい
場合、もしくは NFS のサーバーの設定が難しい場合にはフロッピーベースの
インストールを行ってください。

Debian ではパッケージの管理を行うツールが使われており最初にインストールするのは
ベースパッケージと呼ばれるものとカーネル・モジュールだけです。Debian のミラーサイト
(www.jp.debian.org など)の

```
debian/dists/slink/main/disks-alpha/1998-11-15
```

から自分のマシンに合わせて MILO とカーネルの入った
ファイルと root1440.bin と base2_1.tgz を取ってきましょう。
*.bin というファイルは 1.4MB の大きさですが、base2_1.tgz は 8MB 以上あります。
jensen だけは執筆時点で update が入っていますが、他のマシンは
すべてこのディレクトリの下から取ってきます。
新たにインストールをする場合には必ず update のディレクトリを確認して
自分のマシン用の update ファイルが出ているか確認してください。

base*.bin というファイルがたくさんありますが、こちらはフロッピーベース
のインストールで使うものですから今回は使いません。
ここには次のようなファイル/ディレクトリがならんでいます。

```
.               book1           eb164           root1440.bin
..              cabriolet       eb64p           ruffian
alcor           config.book1    eb66            sable
avanti          config.generic  eb66p           sable-g
base14-1.bin    config.jensen   jensen          sx164
base14-2.bin    config.miata    lx164           takara
base14-3.bin    config.miata-s  miata           xl
base14-4.bin    config.mikasa   mikasa          xlt
base14-5.bin    config.noritake noname
base14-6.bin    config.sable    noritake
base2_1.tgz     config.sable-g  pc164
```

各マシン毎のディレクトリには大体次の二つのファイルが入っています。

```
 drv1440.bin resc1440.bin
```

drv1440.binはARCコンソールかAlphaBIOS用のMILOをロードするためのもので

linload.exe
milo

のファイルが入っています。
一方、resc1440.binにはカーネルが入っています。
ブートの方法には大きく分けて三つあります。

1. jensen, noritake, sable, mikasa, book1の各マシンではカーネルとともに
aboot というブートマネジャが入っていてMILOを使わずにSRMからブートさせます。ブートのコマンドは次のようにします。

>>>boot dva0 -file linux -flags "root=/dev/fd0 rw ramdisk_size=5120
load_ramdisk=1 prompt_ramdisk=1"

2. ARCもしくはAlphaBIOSのマシンでは前述のようにlinload.exeからMILOを
起動させますので、BIOSの設定画面でフロッピーディスクからlinload.exeを
起動してOSファイルとしてmiloを読み出すようにします。

3. それ以外のSRMを持つマシンはresc1440.binのフロッピーにMILOとカーネルが
両方とも入っていてSRMからフロッピーをブートするとMILOが起動します。

>>>boot dva0

MILOを起動した場合にはMILOからLinuxをさらに起動する必要があります。
resc1440.binのフロッピーディスクをドライブにセットして次のように
ブートさせます。

MILO>boot fd0:linux root=/dev/fd0 rw ramdisk_size=5120 load_ramdisk=1
prompt_ramdisk=1

ramdiskのフロッピーを入れるように指示して来ますのでroot1440.binを
セットしてキーを押します。

ARCやAlphaBIOSのマシンでハードディスクからブートしたい場合には
インストール時にMILOをブートするための小規模(3MBほど)なFATのファイルシステム
を作っておきます。
また、SRMを使う場合には「必ず」BSDディスクラベルによってパーティション
の設定をしてください。BSDdisklabelはシェルプロンプトからfdiskを手動で
起動して"b"コマンドで行います。

ベースシステムさえ展開できれば残りのインストールはx86のインストールと
ほとんど同じです。
今回インストールしてみてつくづく感じたのですが、
回線が細い場合にはftpベースでのインストールは大変です。とても家から電話回線で
ftpインストールする気にはならないですね。安定板になったらCDROMが出回ると思われる
のでこの雑誌が出版されるころには簡単にCDROMが手に入ると思います。

2. lprobeの概要と利用方法

前述の通り、COMPAQからGPLにしたがってリリースされたlprobeはプロセッサ
の性能を様々な角度から測定するための有力なツールになります。
このようなツールを一般に公開して頂いたCOMPAQ社に拍手 \^_^/

さて、このツールを使うにはカーネルのバージョンが
2.1.132より大きいか2.2.xである必要があります。
テストはSRMでは行われていますが、MILOからブートしたマシンでは十分には
行われていないそうです。もっともMILOに問題があればMILOの担当者が直す
予定だと書かれているので最新のMILOを使えば問題ないかもしれません。
前章のMultiaのインストールは実はlprobeを使うために行ったのです。残念ながら
誌面の都合で今回はリリースノートから概要の説明に留めます。具体的なインストール
や利用方法は機を改めて書くつもりです。
RedHatを使っている人はrpmとコンパイル済みのカーネルモジュールが提供されるので
カーネルのバージョンさえ気を付けておけば簡単にインストールできます。

2.1 性能カウンタと lprobe

まずはこのツールがどんなものかを説明します。
ＣＰＵの設計は既存のプログラム（ＲＩＳＣでは特に有名なベンチマークに的を
絞って）の定量的な測定を行って設計上のさまざまなトレードオフを決定して
います。測定方法の一つはアドレストレースで、これは全部の命令をトレースする
ことで新しいプロセッサのシミュレーションデータを採取するものです。ところが、
この方法には大きな問題があります。というのはトレースを取るにはすべての命令に
割り込みをかけて必要な情報を採取するのですが、この時プログラムの動的な振る舞い
（特にキャッシュや割り込み）が変わってしまうのです。そこで、実動プログラムの
特性を動的に捕らえるためにハードウェアモニタもしくは性能カウンタと呼ぶ機構が
導入されることがあります。これはシステムのいろいろな事象をカウントするカウンタ
になっていてプログラムが動作したときの性能やイベントをカウントしてプログラムの
振る舞いを推定するために使われるものです。

この性能カウンタはＣＰＵの設計だけでなくソフトウェアのチューニングにも威力を
発揮します。たとえばキャッシュミスを低減するようなアルゴリズムを開発しても
どうも性能が上がらないなぁというような場合に性能カウンタを用いてキャッシュミスが
どの程度減ったのか定量的な評価が可能になります。また、ついでに他の性能要素を観測
することでキャッシュミスを対策したのは見当違いだったなんてことがわかる可能性も
あります。パイプライン動作や分岐予測、キャッシュの振る舞いや変換バッファのミス
など性能に関するパラメータならたいていは性能カウンタで取得できるように作ります。
Alphaにはプロセッサの実装に依存しますが複数の性能カウンタがあります。たとえば
今現役で売られている21164(A)のシリーズは３本のカウンタを持っています。
そこで、３つまでのパラメータは同時に測定できることになります。これ以上のパラメータ
は同時には取得できないので同じプログラムを測定パラメータを換えて何回か流して
全体の振る舞いを推定します。実はこのカウンタは一般に使うものではないので
ハードウェアの低減のために少しビットを減らしています。その代わり所定の値を超えた
ときに割り込みを発生することができます。メモリ上にデータを集計するカウンタや
ヒストグラムを作成しておき割り込みルーチンのなかでこれらの値を更新することで
カウンタの長さを超えた長時間の測定が可能になります。

lprobeはこの便利な性能カウンタをユーザーに開放するものです。カウンタの存在すら
企業秘密にする会社が多い中で性能カウンタの制御・解析ツールを一般に公開するのは
大英断だと思います。

2.2 採取イベント

21164(A)の世代のチップで採取できるイベントを示します。
見れば分かるようなイベントのほうが役に立つものが多いのですが、
それぞれのイベントの詳細はハードウェアリファレンスマニュアルを参照してください。

イベント名	カウンタ番号	説明
bcache_misses	2	外部キャッシュのミス数
bcache_hits	1	外部キャッシュのヒット数
bcache_victims	1	外部キャッシュビクティムエントリへのヒット数
branch_instructions	1	分岐命令数
branch_mispredicts	2	分岐予測失敗数
conditional_branches	1	条件分岐命令数
cycles	0	サイクル数
dcache_accesses	1	一次キャッシュアクセス数
dcache_misses	2	一次キャッシュミス
dtb_misses	2	データアドレス変換バッファ（ＤＴＢ）ミス
dual_issue_cycles	1	２命令実行のサイクル数
external	2	外部事象（ボードによってサポート）
float_operations	1	浮動小数点命令数
icache_accesses	1	命令キャッシュ参照数
icache_misses	2	命令キャッシュミス
integer_operations	1	整数演算命令
itb_misses	2	命令アドレス変換バッファミス
jsr_ret_instructions	1	サブルーチンリターン命令
ldu_replays	2	キャッシュミスを起こしたロードの結果を使う命令の再実行
load_instructions	1	ロード命令数
load_locked	2	ロック付きロード
loads_merged_in_maf	2	MAFにマージされたロード
long_stalls	2	12サイクル以上命令発行が中断されたストール
mem_barrier	2	メモリバリア命令

```
PC_mispredicts       2    PC予測失敗
pipeline_dry         1    命令の供給がなくパイプラインが止まるサイクル数
pipeline_frozen      0    資源競合による命令発行の中断
quad_issue_cycles    1    ４命令同時発行数
replay_trap          1    リプレイトラップ
scache_access        1    二次キャッシュ参照
scache_misses        2    二次キャッシュミス
scache_reads         1    二次キャッシュ読み出し要求
scache_read_misses   2    二次キャッシュ読み出しミス
scache_shared_write  2    二次キャッシュへのシェアード書き込み
scache_victims       1    二次キャッシュビクティムヒット
scache_write         2/1  二次キャッシュへの書き込み
scache_write_misses  2    二次キャッシュへの書き込みミス
single_issue_cycles  1    一命令のみ発行できたサイクル
split_issue_cycles   1    命令スロットの部分的命令発行
store_instructions   1    ストア命令数
system_invalidates   2    システムインバリデイト
system_read_requests 1    システム読み出し要求
system_requests      1    システム要求
triple_issue_cycles  1    ３命令発行
wb_maf_full_replays  2    MAFもしくはWBフルによるリプレイトラップ
```

2.3 実行コマンドと実行時オプション

すぐ使ってみたい人のために実行コマンドと実行時オプションを簡単に列挙します。
動作の詳細はドキュメントを参考に実際に動かしてみてください。

Iprobeは次のシンタックスで実行します。

```
iprobe [-option [-option..]] [event[.freq][.skip] [event[.freq][.skip]...]]
```

実行時オプションは次のようなものがあります。

```
-buffer_count [number]
-collect
-delay_start [seconds]
-duration [seconds]
-help
-input [file]
-interval [number]
-method [sampling-method *1]
-mode
-nocollect
-output [file]
-pc_range [address1-address2].granularity

*1 sampling-method count, sample, mode, ipl, address, total
```

イベント名は表にあげたものを使います。割り込み頻度をfreqで制御しますが、
比較的長いプログラムの評価をするときにはデフォルトのlowでＯＫでしょう。
もう少し精度の高い測定をしたいときにはfreqの部分にhighを指定してください。
ただし、割り込み頻度があがるとＣＰＵのタスクスイッチが増えてプログラムの
振る舞いが変わる可能性もあります。

使い方の例

```
$iprobe -method count -duration 300 dcache_misses
```

これは３００秒の間におきるキャッシュミスをカウントして出力します。
modeを指定するとＣＰＵのモードごとのデータをヒストグラムとして表示するので
カーネルとユーザーのデータを分けて把握できます。

3. おわりに

Debian-JPプロジェクトのおかげで比較的簡単に最新の日本語環境もそろえられる
可能性があります。現在 slink.jp にはすでにかなりの日本語ソフトが入っています。
パッケージ管理の自動化や性能関係のライブラリの積極的な取り込みなど注目すべき
配布パッケージになってきていると思います。

今回は細い大学の回線からftpでインストールしてみたのですが、やはりCDROMで
パッケージを持っていないとインストールは大変ですね。MSwordの件は悩ましいですが
きちんとインストールできたらDebian-JPをメインマシンに入れて使ってみたいと
思っています。

連載の読者の方から使っていないEB164のマシンを譲っていただけることになりました。
誌面を借りてお礼申し上げます。
これで遅れ馳せながらEV5の世代のマシンをようやく入手できます。
性能解析のツールIprobeも手に入ったので数値計算関連の性能についてさらに突っ込んだ
解析を含んだ話を展開できるだろうと思っています。