Universal Shell Programming as a glue language and technology

ISSN 2187-9664

USP MAGAZINE

Vol.1
400Yen

よりぬき版

for the sophisticated shell scripters

第1特集
シェルスクリプト部長の並列処理

第2特集
震災と開発者たち

シェルスクリプト大喜利

潜入! Linux Con Japan 2011

TechLION誌上中継
びぎねっと宮原徹が語る

オープンソースの未来はどっちだ!?

From Editor

震災から 3 ヶ月が経過し、世の中は徐々に平静を取り戻そうとしています。
だけど、何かがちょっと以前とは違うと思いませんか？
明日には何が起こるのかわからない、という現実を目の当たりにして、
「生きる上で本当に大切なものは何なのか？」という疑問を
日本人全員が突きつけられたからなのではないかと思います。

それは一瞬の興味を引くような、真新しいものや斬新な物や、
飾り立てられた何かやエンタテインメントではなくて、
生活の土台である、寝るところや食べるもの、着る物、
そして家族や仲間という、必要最低限のもの。

重厚長大な構造よりも、できるだけ簡素な構造へ。
それぞれの人が、世の中のシステムを知り、
自分たちの足でたくましく歩ける世の中へ。
何かが変わる時代が来たことをそこここに感じずにはいられません。

そんなときにひょっこりこの世の中へ登場した USP MAGAZINE。
まだまだひよっこではございますが、
技術に人生をかけたエンジニアの皆さんを、
シェルスクリプトという簡素な技術でつなぎ、
奥深い技術の世界へと誘う骨太な雑誌を目指したい！！と
心の奥から思う次第です。

どうぞ今後ともごひいきに。

USP MAGAZINE 編集部

Contents USP MAGAZINE 2011 summer

シェルスクリプト部長の並列処理…… 3
From My Tool Box …… -
震災と開発者たち…… -
cat コマンドを読み解こう！ …… -
潜入！ LINUX CON JAPAN 2011…… -
原色 Linux 美女図鑑…… -
今私たちは何を学ぶべきか　第 1 回 …… -
シェルスクリプト大喜利…… 10
オープンソースの未来はどっちだ？…… -
ベンチャー起業家のアドベンチャー日誌…… -
次号予告…… -

※ 薄字で記載している記事は、よりぬき版には収録されておりません。

特集１

スーパーマンでアイデアマン

シェルスクリプト部長の並列処理

執筆　リッチ・ミカン（松浦リッチ研究所）
技術協力　上田隆一（ユニバーサル・シェル・プログラミング研究所）

CPU はマルチコア化が進み、コンピューターはクラウド化が進み、使えるリソースが多数化・点在化する時代になった。これは、並列処理や分散処理に対応できなければ今後主流の座にはとどまれなくなるであろうことを示唆している。

我らがシェルスクリプトはどうなのだろうか？そこで今回、次回の２回に渡り、シェルスクリプトによる並列処理と分散処理の導入方法を紹介する。

記事を読めば、並列・分散処理にさして特殊な道具は要らないことがわかるだろう。多数をケアする能力と、少々の知恵が出せればそれでいい。そんな姿はまるで、部下を束ねる部長のようだ。

面倒見のよい、頼りがいのあるボス

「私は Perl が大好きだー!」……と、こともあろうにシェルスクリプトの雑誌を飾る特集記事の冒頭で、私は何を叫んでいるんだろう。でもそれは、Perl 作者のラリー・ウォールさんが、日本アニメのファンということで親近感を覚えたからだ。[*1] ……いや、それではなくて（それも捨てがたい理由ではあるが）、もちろんまじめな理由がちゃんとある。それは Perl が、何でもできて面倒見のよい、頼りがいのあるボスのような言語に感じられるからだ。

例えば、仕事で納期があって、今はその約束の 3 時間前だとしよう。納品物のディレクトリには必要なデータの他に、何故か関係無いゴミファイルが 5000 個……。急いで仕分けなければならない。でもそんな時、Perl は助けてくれる。「変数定義やコメント、綺麗な記述、そんなの書いてる暇など無いし、可読性とかどーでもいいから、とにかく動いて」という想いで作る書き捨てプログラムを受け入れてくれる。それでいて Perl は、本格的なプログラム作りにも付き合ってくれる。CPAN という豊富なライブラリーを持ち、オブジェクト指向プログラミングもしたければできるし、Ruby on Rails に代表されるような MVC スタイルの開発もサポートしているし……。実に頼もしい。Perl のこうした、何でもできて面倒見のよい様は、まるで頼りがいのあるボスだ。

さて、シェルスクリプトはどうか？こちらも書き捨てプログラムを動かしてくれるのは言うまでも無い。では一方、本格プログラムは作れるのか……。本誌 0 号の特集 1 で、シェルスクリプトを用いて本格的データベースを構築する話を書いたが、そこまででもかなり頼りがいあるボスに見えてくる。しかし今回さらに、もう一つその話題を提供したい。それが今回のテーマ、シェルスクリプト部長の並列処理なのである。

困っている部下には、とことん付き合う。

勿論、全員の面倒を見るのも忘れない。

それが、頼りがいのあるボス。

シェルスクリプトを擬人化してしまったが、それは並列処理の頭をとっている様子が、じつに人間っぽいからだ。人間が部下を使って事を成し遂げるのと同じドラマがそこにある。実際にシェルスクリプトを書くのはプログラマーであるから、つまりあなたが、このシェルスクリプト部長になって処理という名のプロジェクトを成功に導くのだ！

1 億行もの、途方も無いデータが立ちはだかった

「いち！じゅう！ひゃく！せん！まん！……いちおく！」残念ながら「なんでも鑑定団」ではない。[*2] この記事で、処理を挑もうとしているデータ（TESTDATA）の行数（レコード数）を数えた時に私が発したセリフだ。

1 億行とはなんとも暴力的な行数だ。しかしこのようなデータを実験材料としては用意したのは、並列処理の議論をするためだ。単独で処理するには途方の無さを感じてしまうくらいに膨大である方がいい。ちなみに中身はこんなかんじだ。

```
1534 高知県 -9,987,759 2001 年 1 月 5 日
3772 鹿児島県 5,689,492 1992 年 5 月 6 日
8238 大分県 1,099,824 2010 年 2 月 22 日
1292 秋田県 -4,497,866 1991 年 2 月 23 日
9812 新潟県 5,721,468 2001 年 2 月 28 日
:
```

*1 ウォールさんの好きな作品は「少女革命ウテナ」らしい。そして、じつは私も大のアニメ好きで……。化物語とかいいよねー、登場人物の中に MSX ユーザーがいるとかマニアックすぎだー。

*2 1 億円もの鑑定結果が付くシーンがあったら見てみたいが。

番号、都道府県名、金額、年月日と4つのレコードから成るが、こういうフォーマットであることにあまり深い意味はないので気にしないでほしい。

さて、このデータのうち、4列目の“yyyy年mm月dd日”という形式を分解してyyyyの列、mmの列、ddの列に独立させてくれと言われたとしよう。答えは簡単だ。

```
awk ´{gsub (/年|月|日/," ",$4) ;print}´ TESTDATA
```

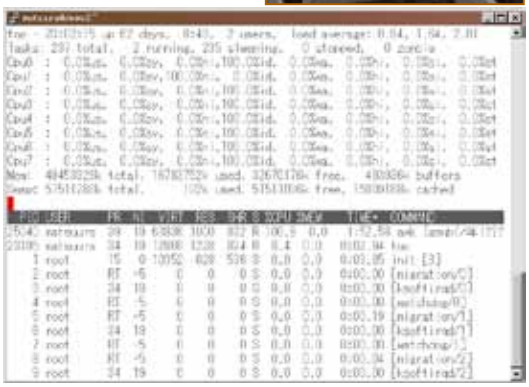
とでも打てばいい。あとは終わるのを待つだけだ。

ところが、1分、2分、3分、ある程度予想はしていたがやっぱりまだ終わらない。いくら1億行とはいえ、4コアのXeonを2基（つまり見かけ上8プロセッサ）積んだわりと贅沢なコンピューター（→図1）[*3]を使っているのに……。「ちゃんと働いてるのか？」と気になり、もう1つのコンソール画面でtopコマンドを使ってCPU稼働率を見てみたら、案の定、図2のような状態になっていた。

せっかく8プロセッサも積んでいながら1つだけが頑張っていて、残りの7つは休んでいる……。これでは時間の無駄だし、他にもいろいろな意味でコストの無駄だ。なんとかせねば。

図1.▶
今回使用するコンピューターのお姿。主なスペックは、CPUがXeon 3.2GHz, 4コア×2基、メモリが48GB…、と相当に贅沢!!

図2.▼
その贅沢なコンピューターで1億行のデータを普通に処理してる様子。うーん……



ポイントは、人間のグループ作業だったらどうするか

さあ、ボスの出番である。シェルスクリプト部長になるのだ。ターゲットとなるコンピューターが8プロセッサ相当なのだから、部下は部長自身を含めて8人ということになる。プランを練りやすくするために、ひとまず似たような例をモデルにして考えてみよう。

今、目の前に処理しなきゃならない800ページからなる一冊のバインダーがあった時、人間ならどうするか。

普通なら1人100ページずつ分け与えて処理させ、あとから回収して再び一冊にまとめようと考えるだろう。だから、その考え方に基づいて本題をシェルスクリプト向けに翻訳すればよい。そのようにして作ってみたのが“paraawk”だ。

早速リストを紹介したい。……したかったのだが、このプログラムを管理している方から「だ〜め(^^)(ライセンスの都合で)」とやんわり断られてしまった！どうしても「だ〜め(^^)」という……。「リストが載ってないんじゃ拍子抜けだよなー」と思ってしまうかもしれないが、まぁそう言わずに続きを読んでもらいたい。リストがダメでも、「リストを載せずに考え方を載せるのなら構わないよ」との

*3 ちなみにメモリは48GB。ちょっと贅沢すぎるか。

解説 名前付きパイプとは

名前つきパイプの生成にはmknodコマンドを使えばいいと本文中で述べた。しかし、そもそもその「名前付きパイプ」というものをご存知だろうか。実は、並列処理をさせるためには知っておくべき仕組みなのでおさらいしておく。

名前付きパイプを解りやすく例えると、テンポラリーファイルだ。データを渡したい側のプログラムがファイルリダイレクト等でこれに書き込み、受け取りたい側のプログラムが同様にこれから読み取る。ただし、渡す側、受け取る側のプログラムが、同時に読み書きしなければならないようになっている。送り側が先に一行分を送ろうとすれば、受け側がその一行を読みに来るまで待たされる。逆に、受け側が先に一行分読みに行こうとすれば、送り側が一行分書き終えるまで待たされる。

こうすることでデータを確実に受け渡せるようになっている。普通のファイルが「置き手紙」なら、名前付きパイプは「手渡し」だ。見た目はどちらもファイルだが、扱い方は正反対なのが面白い。

見た目は同じ、扱いは逆、「ファイルとパイプ」

	普通のファイル	名前付きパイプ
見え方	1つのファイル	1つのファイル
扱い方	同時読み書き**禁止**	同時読み書き**必須**
感覚	置き手紙	手渡し

ことだったので、是非そうさせてもらおうと思った。実は、paraawk の内容など十数行で書けてしまうのだ。考え方さえ聞けば、恐らく多くの人は書けるだろう。簡単なのだ。例えリストが載せられなくても、考え方を伝えるだけでも価値があると私は考えた。

では、その「考え方」を説明しよう。(ずいぶん具体的な考え方に見えますが「考え方」ですからねー)

1) 名前つきパイプを作る。
　A ライン (配下で働く awk に加工前データを渡す用)
　B ライン (働いた awk からの加工後データ受取り用)
　の 2 系統で、並列実行させたい awk の数の倍必要。
2) 標準出力から流れてきたデータを、A ラインのパイプ群に、一行ずつ順番に流す。
3) awk をバックグラウンド ("&") で、並列化したい数だけ生成する。
　awk への入力はさっきの A ラインから
　awk から出力はさっきの B ラインへ
4) B ラインのパイプ群から流れてくるデータを一行ずつ重畳し、標準出力に流す。

とこれだけだ。1) の名前つきパイプなど mknod コマンドで作ればいいので簡単だ。もし名前つきパイプについてよくわからないということであれば前ページの**解説**を読んでもらいたい。3) のバックグラウンド実行とてシェルスクリプトに入門したての方でもなければわかると思う。問題は 2) と 4) だが、これはこの後で解説する。

ボスはまず、スーパーマンでなければならない

さて、説明が後回しになった先程の「考え方」の 2) と 4) である。もう少し詳しく述べると、まず 2) は標準入力を複数のファイル（今回の場合は 8 つのパイプ）に 1 行ずつ順番に分け与えるコマンドである。ポーカー（カードゲーム）に例えれば、トランプを参加者に 1 枚ずつ順番に配るディーラーみたいなものだ。そして 4) はその逆で、複数のファイルから 1 行ずつ読み取って、標準出力に流す作業である。

もちろんどちらもシェルスクリプトやその周辺のコマンド (awk や sed 等) でできる作業だ。例えば 2) の作業を実現するには、次のように書けばよい。

```
awk ' {print $0 >>"/path/to/pipe."(NR-1)%8} ' /dev/stdin &
```

同様にして 4) についてもやっぱり書けるのだが、こちらはもう少し長めなので割愛させていただく。*4

シェルスクリプト等で書けることは書けるのだが、今回の paraawk では、C 言語で書き直して一つのコマンドとして独立させたものを使っている。**図 3** を見るとわかるので先に述べておくが 2) を C で書き直したものが "distribute"、4) を書き直したものが "interlace" である。

なぜそんなことをしてしまったのか？実は 2) と 4) というこの 2 つの箇所は、配下で働く awk の 8 倍のデータ転送速度が要求されるために、高速動作させる必要があったのだ。分け与えたり回収したりするという役を担うチームのボスは、部下とは違ってスーパーに仕事のできる人物でなければならないということである。こういうところまで人間もプログラムも共通しているのは、ある意味興味深い。

チームプレイの威力

さて、この awk を 8 つに並列処理したスクリプトは処理をどれほど高速化するのだろうか。年月日を独立させるという前述の処理を、Linux（CentOS 5.6）をインストールし

*4 こちらは公開しちゃだめというわけではいのだが……。その代わり、本誌後半の「シェルスクリプト大喜利」のお題にしてみてはという意見が出たのでそのようにさせていただいた。そちらの記事もよろしく！

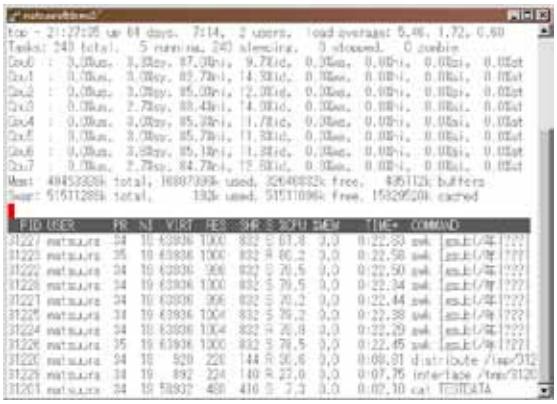


図 3．"paraawk" で 1 億行のデータを処理しているところ。

表 1．単独 awk と paraawk の処理速度と paraawk の実質的な速度向上

処理行数	1 億	1000 万	100 万	10 万	1 万	1000
単独の awk	256. 753 秒	25. 630 秒	2. 578 秒	0. 257 秒	0. 027 秒	0. 004 秒
paraawk	41. 429 秒	4. 161 秒	0. 436 秒	0. 060 秒	0. 020 秒	0. 015 秒
速度向上	6.2 倍	6.2 倍	5.9 倍	4.3 倍	1.4 倍	0.3 倍

※ awk, paraawk の両コマンドが、結果を /dev/null に送り終えるまでの秒数を time コマンドで 5 回計って平均したもの。行数の調整には head コマンドを使用。

た先の 8 プロセッサ相当コンピューターで試してみた。所要時間は相対的な意味しか持たないのでそれ以外のスペックの説明は割愛させていただく。

図 3 は、このようにして作った paraawk を動作させている時に top コマンドで見たコンピューターの稼働状況だ。最初に awk を動かした時の図 2 と比べれば違いは一目瞭然だろう。8 つの CPU が全て働いている。画面下半分にあるプロセスを見ると 8 つの awk、それに distribute と interlace が居るのがわかる。

表 1 は、単独の awk と paraawk の所要時間を図って比較したものだ。尚、処理結果を /dev/null に捨てさせて表示にかかる時間の影響を受けないようにしていることを補足しておく。

肝心の結果はというと、当初の目的である 1 億行のテストデータに対して、じつに 6.2 倍もの速度向上を達成した。

ただ興味深いことに、処理させる行数を 10 万行くらいまで減らすと徐々に効果が縮まり始め、1000 行では単独の awk が逆転した。つまり少々のデータならチームを組まずに一人で片付ける方が早いということで、これもやはり人間の世界と同じだ。

ソートはどうする。みんなで分担できるのか!?

シェルスクリプト（と少々のスーパーな C プログラム）で awk の並列化ができた。awk の部分を他のコマンドに差し替えれば、どれも同じ要領で並列化できる。……と思ったら、一つその方法では通用しないコマンドがあった! sort コマンドだ。なぜならソート（並べ替え）という作業は全てのデータを見てこそできる作業だ。複数人で手分けすることなんてできないんじゃないだろうか。

その疑問は当たっている。だが、半分だけだ。実は部分的に手分けできるのだ。さあ一体どうやって手分けするのか？そのアイデアを捻り出すのも人より優れた人物たるボスの仕事だ。

ボスはアイデアマンでもなければならない

アイデアマンと呼ばれるようなボスなら、まずソートという作業を 2 ステップに分解できることに気付くはずだ。

ここで、いかに分解するかという話をトランプを使って説明しよう。作業するのは「ボス」と「部下」の 2 人だったとする。そして今、目の前にハートの A から K まで 13 枚のカードがある。これをシャッフルし、ボスと部下で半分ずつ分ける。奇数枚なのでどちらかが一枚多くなるがそこは大した問題ではない。これで準備完了。そうしたら次の 2 ステップの作業をする（→図 4）。

1) ボスと部下それぞれが手持ちのカード 6 枚（または 7 枚）を小さい順に並べ替え、終わったら机の上に置く。（そうするとカードの山が 2 つできる）
2) ボスはその後、2 つのカードの山の一番上のどちらかから 1 枚カードを取り、手元に並べる。この時、小さい数字のカードが置いてある方を選ぶようにする。これを山が無くなるまで繰り返せば、カードが小さい順に並ぶことになる。

原理[*5]は理解できただろうか。こうすれば、全部ではないにしろ前半の作業を 2 人で分担することができる。前半は純粋なソート作業、後半は選択的なマージ作業と言えるわけで、このような分解を行うことで複数人を作業に参加させられるようになる。

*5 このようにソートを分解し、マージ作業化していく手法のソートは、実は「マージソート」と呼ばれている。

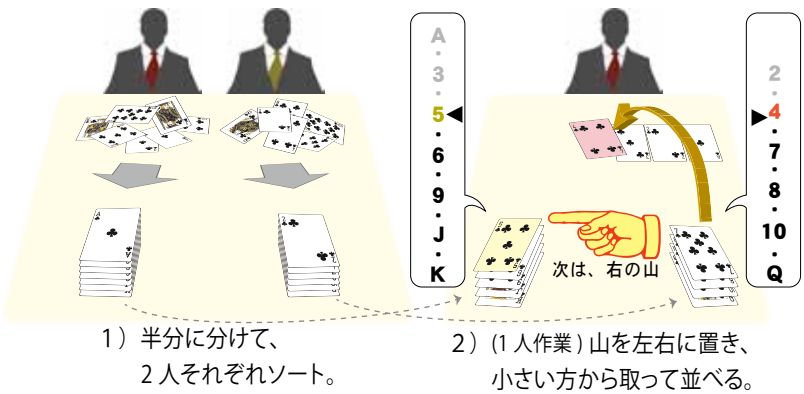


図 4. 1 つのソートは、小規模なソート作業とマージ作業に分解できる。

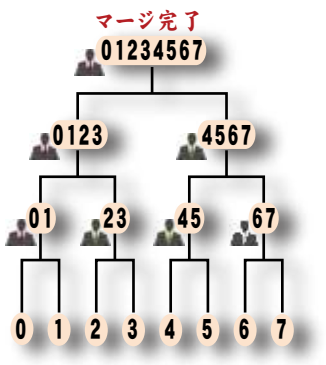


図 5. マージ作業のトーナメント化

ボスは応用力も持っていなければならない

もともとの話は 8 プロセッサー相当のコンピューターで並列化するという話なのでもう一捻りアイデアが必要だ。

前述のトランプの例では 2 人だったが、本題である 8 人で分担するにはどうすればよいか、応用して考えてみてもらいたい。「トランプを 8 人に配って 8 人にソートさせ、最後に 1 人が 8 つの山からマージを行えばいい」うーんオシい。前半はいいが、後半のマージ作業を 1 人でしかできないのが勿体無い。こうしたマージ作業の負荷の偏りを減らすいいアイデアはないだろうか……。

それはマージ作業のトーナメント化である（→**図 5**）。[*6] 先程説明したように、ソート作業は、ソート作業とマージ作業に分解することができる。だから分解してできたソート作業をまた分解するのだ。こうすると全体としてマージ作業の割合が増えるので、一見不利になるように思えるがそうではない。2 度目の分解で生じたマージ作業は合計 2 人でできる。しかも 1 度目の分解で生じたマージ作業では、選択元として見なければならないトランプの山が半分になるので 1 人でやるにしても負担は減ることになるからだ。

結局、8 人でソートを分担するためのシナリオはこうだ。

1) 8 等分したデータを 8 人それぞれがソートする。すると 8 つの小さなソート済データができる。
2) 4 人が 1 人 2 つのデータを集め、各自マージを行う（マージ準々決勝）。するとソート済データが 4 つになる。
3) 2 人が 1 人 2 つのデータを集め、各自マージを行う（マージ準決勝）。ソート済データが 2 つになる。
4) 最後に 1 人が 2 つのデータを集め、最後のマージを行う（マージ決勝）。データは 1 つになり、作業が完了する。

部長の出したプラン“parasort”

このアイデアをシェルスクリプトに翻訳することでできたものが “parasort” というスクリプトだ。既に述べたようにリストは公開できないので、[*7] 考え方のみ記す。といっても重要なアルゴリズムのほとんどは解説してしまったので補足だけ……。

補足は 3 つある。

その 1

まずはソート予選を開催するにあたっての各ソートへのデータの配布だ。しかしこれは paraawk と同様に distribute コマンドを使えば問題ないことがわかるであろう。

その 2

2 つ目は、マージ作業をどうやるかだ。ソートという作業は sort コマンドを使えばいいとして、**図 4** でも説明したマージ作業はどうやるのか？

でも実は簡単だ。sort コマンドに “-m” オプションを付ければよいだけなのだ。これまで “sort -k 1,1” などとして 1 列目をソートしていたのであれば “sort -m -k 1,1” と書くだけで、マージモードになる。もちろん “-m” オプション付きの sort コマンドに食わせるデータはそれぞれ通常のソートをしておかなければならない。これは**図 4** を理解していただけているなら納得できるところなはずだ。

ところでこの “-m” オプションがちゃんと用意されているところをみると、sort コマンド設計者もアイデアマンなのだろう。

その 3

3 つ目は、「待ち」である。トーナメントというのは上位の試合に進むためには下位の試合に決着がついてなければならない。ところが並列処理を行うため、各 sort コマンドは “&” をつけてバックグランド動作させる必要がある。すると、下位試合の決着がつく前に上位試合用の sort コマンドにたどり着いてしまう。これはマズい！そこで必要なのが「待ち」というわけだ。ではどうやって待たせるのか？

それには wait コマンドを使えばいい。wait コマンドは対象となるプログラムが終了するまで待つコマンドであるが、これを使うには目的の sort コマンドのプロセス ID を知る必要がある。しかしそれを知るのも難しくない。ps コマンド，grep コマンド，awk コマンドのあわせ技で

```
ps axww | grep sort | grep -v grep | grep 下位試合が処理しているはずのファイル名 | awk '{print $1}'
```

などと打てば一発でわかる。

部長のソースは 100 行足らず

これらのことさえ頭に入れておけば parasort 相当を自力で書くにしても 100 行も無しに書けてしまう。部長の考え方を盗み、あなたも頼れる部長になってみないか！

プランの実力

さて、部長が出したこのプランはどれほどの実力を発揮するのだろうか。早速実験してみた。

扱うデータは前と同じで、動作させる環境も、結果を /dev/null に送り終えるまでの時間を計るというところも全く一緒である。今回は 1 列目をソートしてみた。

*6 実はここまで含めて先程紹介したマージソートのアルゴリズムである。このアルゴリズムを知っていれば知恵を絞る必要はないのだが、その分幅広い知識を求められる。どのみちボスは頭がよくなくてはならないということだ。
*7 ピー音（自主規制音）ばかり入る放送みたいで申し訳ないが……。

表2．単独 sort と parasort の処理速度と、parasort の実質的な速度向上

処理行数	1 億	1000 万	100 万	10 万	1 万	1000
単独の sort	17202. 340 秒	1070. 217 秒	43. 643 秒	1. 617 秒	0. 044 秒	0. 003 秒
parasort	4590. 105 秒	349. 058 秒	19. 189 秒	1. 044 秒	0. 334 秒	0. 317 秒
速度向上	3.7 倍	3.1 倍	2.3 倍	1.5 倍	0.13 倍	0.01 倍

※ sort, parasort の両コマンドが、結果を /dev/null に送り終えるまでの秒数を time コマンドで 5 回計って平均したもの。行数の調整には head コマンドを使用。

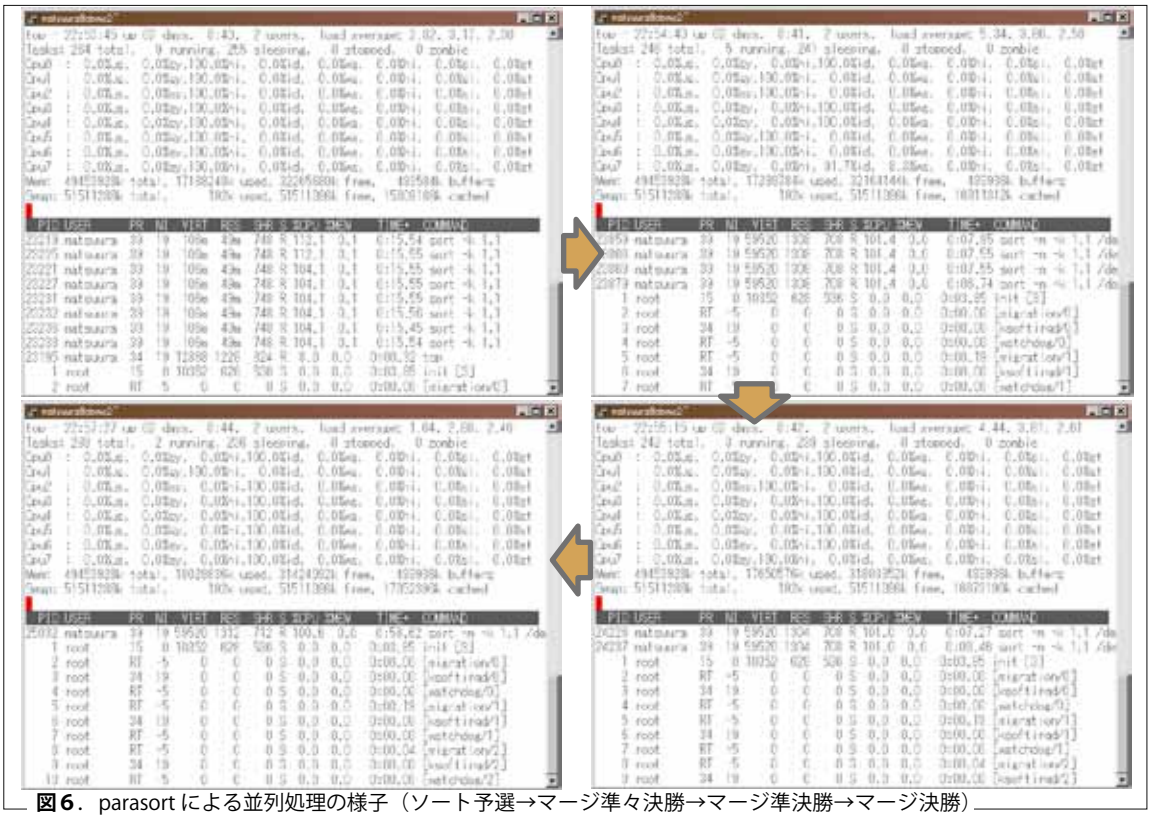


図6．parasort による並列処理の様子（ソート予選→マージ準々決勝→マージ準決勝→マージ決勝）

元の単独 sort と比べ実質的な速度比 3.7 倍を達成した（→表 2）。awk の時ほどのでないことに疑問を持つかもしれないが、それは図 6 のようにマージ作業に入ると分担できるプロセス数が次第に減ってしまうからだろう。しかしながら、ソートという作業は、表を見ればわかるようにもともと桁違いに時間のかかる作業であり、3.7 倍であってもとても大きな時間差となって現れている。その点を鑑みればやはりこのプランは実用的と言えるのではないだろうか。

次回、本部長に昇進&世界を股にかける

部長とかボスとか称して擬人化してきたが、並列化を実現するまでの考え方や、実行中の挙動、それに効果の現れ方、どれも実に人間っぽさを感じられなくはないだろうか。そして、並列化には、スーパーマンとアイデアマンの存在が大きな鍵を握っていることがわかったと思う。逆に言えば、重要なのはそこであり、専門的な知識やツールといったものはほぼ不要なのだ。本記事でもおさらいした名前付きパイプくらいなものである。ツールに関して言えば、ご覧の通りシェルスクリプトと簡単な C プログラムだけあれば事足りる。なんともお手軽な話ではないか。

さて冒頭でも述べたが、この特集記事には後編がある！シェルスクリプト部長は本部長に昇進してしまう。そして世界を股に書け、多くの人材を活用し、より壮大なプロジェクトに挑む！つまり、後編は複数台のコンピューターを操る分散処理の話に発展するのだ。処理速度も驚きの向上率を見せる……。もちろん後編でも特別なソフトウェア無しにやってのけるのだが、シェルスクリプトでそこまでやる姿をあなたは想像できるか？では、後編を乞うご期待！

シェルスクリプト大喜利

第一回

司会：『吹く』編集長・みかん

はじめまして、そして予めお題の回答を考えてくださった皆様、お待たせいたしました。今回から USP MAGAZINE で連載されますシェルスクリプト大喜利（略して sh 大喜利）のコーナーです。まずはメンバーのご挨拶から。

私は司会を務めます吹く編集長のみかんです。誤字ではありません。思わず吹いてしまうような作品をお待ちしてます。他のメンバーは……、読者の皆さんです。ここは読者参加型のコーナーなのです。ではシステムを説明します。

シェルスクリプト大喜利とは

購読している皆さんに毎号お題を出して回答を募り、面白いものを紹介します。そして、良い回答には段位を授与、悪い回答では段位を強制返還させてもらいます。面白さに応じて授与・返還される段数は様々です。そして、見事十段になりますと‼ **筆舌に尽くしがたいちんじゅうちゃんグッズ**をプレゼント！日曜夕方のアレとほとんど同じですね。ただ、扱うものがシェルスクリプトということでいくつか特有のルールがあります。以下にそれを記します。

シェルスクリプト大喜利特有のルール

一、sh 大喜利はクイズやテストではありません。つまり決まった答というものはありません。敢えて言うなら面白いスクリプトが正解です。

二、面白いスクリプトとは例えば、こんなものです。

イ、人が考えつかない意外性がある

ロ、美しい or 芸術的 or 記述がシンプル・短い or 高速

ハ、アイデア・こだわりが光る

ニ、ネタになるようなバカバカしさ、くだらなさがある

などなど、ただし最後のは段位強制返還の恐れが…。:-)

三、スクリプトの動作環境は Linux とします。そして、とくに断りがない場合は、Linux JM(http://linuxjm.sourceforge.jp/**) に記載されているコマンド及び機能のみ使用可能とします。これは多くの人が楽しめるようにするためです。(JM にあるという理由で、C シェル系での回答も OK！)**

四、sh 大喜利はシェルスクリプトを披露する場ですから、Perl や Ruby、Python などは使っちゃダメです。そもそも JM にも載っていません。逆にシェルスクリプトにとって不可欠な awk や sed 等は OK です。JM にもありますし。

それでは、本日のお題に進みましょう。

本番開始

<第一問>

yes コマンドで何か面白い使い方を教えてください。

yes コマンドは、yes でいいのかどうかを何回も聞いてくる他のコマンドに “y” の連打を浴びせるというとてもニッチな用途のコマンドですが、もっと使い道を開拓してあげたいのです。さて、どんな使い道がありますか。

◎青木剛理さんの回答

```
head -n $$ | tr -d "¥012" | awk ´length($1)!=´$$ | tr -d "¥171" | awk ´END{if(!NR)print "fall in love."}´
```

記念すべき本連載の初回答です。添えられていた説明によると、この記述に対して yes コマンドをパイプで流した時だけ “fall in love.” と告げるそうです。

うーん、正直いまいちかな……。と思ったのですが、echo で “y” とか “yes” とか作ってパイプに流し込んでみても騙されないのですよ。エライ！ちゃんと考えてるねー。初回答だし、景気よく行こうじゃないの。というわけで、**初段授与**！

◎U ○○友○○会 ○田会長の回答

```
yes > /dev/null && rm -Rf ~/*
```

添付コメント「隣の人が離席したときに慎重に打ちます。とりあえず実行しても事故は起こりませんが、どうやって止めても後味が悪いというのがポイントだと思います。」

動作検証のためとはいえ、打ち込むのがコワかったよ！打ち間違えたら大変じゃないか。それに間に head コマンドなど挟めば時限爆弾にもなる。これはけしからん‼ 段位を**二段強制返還**だ！まぁ無段位だからそのままだが……。

◇ ◇ ◇

他投稿者の回答として、dd コマンドの代わりに空ファ

イルを作る用途などが提案されましたが、受付窓口への投稿ではなかったので採用を見送りました。一番実用的な回答だったんですけどね。さて、次のお題。

<第二問>
BASH の RANDOM や /dev/random,/dev/urandom、awk の rand などなど、乱数を得るために用意されている機能を使わず、「それで得るか！」という方法で乱数を得る方法を教えてください。

これはまぁちょっとした頭の体操ですね。難しくはないと思いますが、肝心なのは面白いかどうかです。

◎りゅうち てつやさんの回答

```
printf "%d¥n" `echo -n ´0x´; ps aux | md5sum | cut -b 4-5`
```

ちなみに FreeBSD の場合は md5sum を md5 に替えてくださいとのこと。他の動作環境を考慮してもらえるのはありがたいです。なるほど、種は ps コマンドですか。そして MD5 化して一様分布化しているわけですが、ここらへんは投稿者してくれた皆さんが同じでした。しかしその後、整数化するとこまでケアしていながら他に比べてその記述がシンプルでしたので掲載しました。いいですね、**初段授与**！

他投稿者の回答としては、乱数の種を top にするもの等ありましたね。私も予想してましたが、これを使うと表示まで 1 秒待たされてしまうんですよね。他には、hwclock を種に使う回答を頂きましたが、root でなければパスが通っていないのみならず、

```
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access method.
```

と出て、使えない場所が多数あるので残念ながら採用にはなりませんでした。残念！では最後のお題。

<第三問>
1000 以下の素数を求めるプログラムを一行野郎（one liner）で書いてください。（ただし、面白くないのでこの問題では factor コマンドは使わないでください）

これは一行野郎シリーズです。こういうお題も毎回用意していこうと思います。今回は肩慣らしで素数問題です。

◎片山さんの回答

```
(for((i=2;i<=1000;i++))do((_$i))||cat<<<$i;for((j=i;(j+=i)<=1000;))do((_$j=1));done;done)
```

じつは、4 つの回答を投稿してくれたのですが、記述に拘ったこれを採用しました。スペースが全く入っていません。確かにこれは見た目が面白い。**初段授与**！ただ、その 4 つのうちの一つは高速性を狙った回答だったのですが、残念なことに次の回答の方が速かったのです。

◎鳥海7号さんの回答

```
seq $(dc -e 1000vp) | sed ´s/.*/seq $((&+&)) & 1000/´ | sh | sort -n | uniq -u
```

特に速さを追求したとも書いておらず、さりげないところがカッコいい！（誌面の都合で紹介できないが）jot コマンドを使った FreeBSD 版も添えてくれたし、うーん……よし、**二段授与**だ！

◇ ◇ ◇

というところで本日の大喜利はこれにてお開き！読者の皆様、投稿してくれた全ての皆様、ありがとうございました。

投稿大募集!!

シェルスクリプト大喜利は皆様からの投稿あってこそ成り立つ企画です。お題に対する回答、お題そのものを広く募集致します。さて、次号のお題はこちらです。

次回のお題

一、strings コマンドの意外な使い方を教えて下さい。

二、本誌 6 ページに出てきた interlace コマンドを distribute コマンドのごとくシェルスクリプトに移植してください。8 つのストリームを重畳する専用バージョンで構いません。なるべく簡潔・高速なコードがいいです。

三、USP 友の会に会員登録すると事務局に次のメールが自動送信されるそうです。

```
タイトル:USP友の会ご入会ありがとうございます。
本文:
XXXXXXXX
---以下、事務手続き用          （注）
name=山田太郎                ← この4項目の順番は
email=xxx@usptomonokai.jp    ← 仕様では未定義で、
place=東京都                 ← どの順番でくるか
how=twitter                  ← わかりません。
---
```

これを name, email, place, how の順番で /home/usp/members.csv に CSV で追記出力する 1 行野郎スクリプトを作ってください。なるべく短い記述で！このお題には軽い工夫が必要です。気づきますかな。

投稿のしかた

お題への回答は、お名前（ペンネーム）、回答したいお題番号、回答スクリプト、簡単な補足の四点セットで下記の宛先へ！一人何問でも何個でも回答可です。尚、次回締め切りは **8 月 22 日午前 0 時**とします。しかもその間は何度でも回答の修正を受け付けます。

それからお題も大募集。考えてくれた方にも段位を授与します。自分で出題して回答するもの、今のところ可！

どちらも宛先は mag@usptomonokai.jp です。

bashbash(バシバシ)tcshtcsh（トシトシ）投稿してください。はい、おあとがよろしいようで……。

編集・発行人　鹿野恵子　発行・発売 USP 友の会　〒105-0003 東京都港区西新橋 3-4-2 SS ビル 3 階
有限会社ユニバーサル・シェル・プログラミング研究所内　電話 03-3432-1174

定価 400 円（本体価格 381 円）