『VPP700/56 利用の手引』補遺

渡部 善隆 *

VPP700/56 の運用開始から一年近く経過しました．本稿は，1997 年 2 月に発行された『VPP700/56 利用の手引』の第 1.0 版 [3] から変更，追加された機能をまとめたものです．これらの追加記事をまとめた利用の手引第 2.0 版は，来年の春に発行したいと考えています．なお，VPP700/56 の対話型処理については [5] で既に紹介しましたので，ここでは省略しています．また，内容は [3] に書かれた手法，用語の知識を前提としています．

1 バッチリクエストの投入方法

センターニュース No.561 でお知らせした通り，1997 年 8 月 1 日から 1PE あたりのリージョンサイズの 省略値 を 0.5GB に設定しました． 上限 はこれまで通り 1.7GB です．省略値を設定したことにより，qsub コマンドでバッチリクエストを投入する際に -lM オプションを指定しない限り，リージョンサイズは 0.5GB に設定されます． 0.5GB 以上のリージョンサイズを確保したい場合は，-lM オプションによるサイズの設定が必要です．

省略値を設定した理由

運用開始から半年あまりの間に VPP700/56 で処理されたジョブを調べたところ，記憶領域が 0.5GB 以下のジョブがほとんどを占めることがわかりました．現在のシステムは，合計 1.7GB を越えない限り 2 つのジョブが同時に実行できるようになっています．このため，記憶領域が空いているにもかかわらず記憶領域が確保できないための実行待ちが起こるという事態を改善し，計算機の有効利用を推進する目的で省略値を設定することになりました．

1.1 0.5GB を超えるジョブの実行方法

• -lM オプション

VPP700/56(ホスト名 kyu-vpp, cf.[5])，および汎用計算機 M-1800/20U の UXP/M(ホスト名 kyu-cc) からジョブを投入する (cf.[3]) 場合には，qsub コマンドの -lM オプションでリージョンサイズを指定します．“-lM” の “l” は小文字の「エル」，“M” は大文字です．リージョンサイズの指定は -lM の後に空白をおいて

-lM␣ 整数.小数 gb

で指定します．“gb” はギガバイト単位を意味します[1]． 1.7GB の場合は “-lM 1.7gb”， 0.8GB の場合は “-lM 0.8gb” となります．

*九州大学大型計算機センター・研究開発部　E-mail : watanabe@cc.kyushu-u.ac.jp

[1] これ以外の単位でも指定可能です．詳しくは man qsub で参照下さい．

●バッチリクエストの投入例

以下はリージョンサイズを 1.7GB に設定したバッチリクエストの投入例です．なお，バッチリクエストファイル名は a.sh とします．

```
kyu-vpp% qsub -lM 1.7gb a.sh  ↵                    <---p1 キューに 1.7GB で投入
Request 30482.kyu-vpp submitted to queue: p1.
```

●並列ジョブの場合

並列ジョブの場合も -lM オプションでリージョンサイズを設定します．指定するサイズは合計のサイズではなく，1PE あたりの値を設定します．

以下はリージョンサイズを 1.7GB に設定し，p16 キューに投入する例です．

```
kyu-vpp% qsub -q p16 -lM 1.7gb a.sh  ↵          <---p16 キューに 1PE あたり 1.7GB で投入
Request 30483.kyu-vpp submitted to queue: p16.
```

●制限値を超えたジョブの運命

省略値の 0.5GB および qsub コマンドの -lM オプションで指定したリージョンサイズを超えてしまったジョブは，システムにより処理を打ち切られます．

Fortran プログラムであらかじめ (静的に) 領域を宣言している場合は，実行に入る前に以下の例のようなメッセージを標準エラー出力ファイルに書き出します．

```
Warning: no access to tty; thus no job control in this shell...
Mon Sep  8 11:35:07 JST 1997
UX:timex: ERROR: Unable to exec a.out: Not enough space
```

動的に領域を確保するプログラムの場合は，実行時にメモリーが足りなくなった旨のメッセージを出力し，実行が打ち切られます．以下は Fortran 90 で動的に配列を宣言するプログラムでの出力例です．

```
Warning: no access to tty; thus no job control in this shell...
 jwe0912i-u line 41 The dynamic area cannot be reserved because of insufficient
area. The used size is 00032768KB. The required size is 00032769KB.
```

1.2 リージョンサイズの見積もり

FORTRAN 77 の仕様に従い，配列の大きさをすべて宣言してあるプログラムから作成された実行ファイルは，size(/usr/ccs/bin/size) コマンドでおおよそのリージョンサイズを見積もることができます．

```
kyu-vpp% size a.out  ↵                   <--- 実行ファイル a.out のリージョンサイズを見る
15203128 + 278904 + 1605593060 = 1621075092
```

単位はバイトです．上の例では，約 1.6GB の記憶領域が必要なことがわかります．動的にメモリーを確保するプログラムの場合，正確な値は得られません．

また，Fortran 90/VPP で翻訳された実行ファイルに対し，グローバル変数の領域の大きさを表示するコマンドとして gsize(/usr/local/bin/gsize) コマンドがあります．gsize コマンドは kyu-vpp でのみ利用できます．

```
kyu-vpp% gsize a.out ↵                                  <---gsize コマンド
-------------------------------------------------------------------
  global array information :
    total size        -             1750175KB (   2) on   32 pe's
      partitioned     -             1750175KB (   2)
      non-partitioned -                   0KB (   0)
    max. size / pe    -               54692KB
-------------------------------------------------------------------
```

この実行ファイルの 1PE あたりに必要なグローバル変数の領域は約 54MB であることがわかります．

1.3 ジョブスクリプトに記述

いちいち qsub コマンドのたびに -lM オプションを指定することが煩わしい場合は，投入するバッチリクエストファイル名 (例では a.sh) にオプションを書き込むことができます．

バッチリクエスト内で "# @$" に続けて空白をおかずに指定された文字は qsub コマンドのオプションと見なされます．

```
#                                       <---csh で記述
# @$-lM 1.7gb                           <---qsub オプションの指定例
cd EXAMPLE                              <--- ディレクトリの移動
frt -Wx -Ps -Wv,-m3 test.f90            <--- 翻訳
a.out                                   <--- 実行
```

例では，1PE のリージョンサイズの値を 1.7GB に設定しています．

もちろん qsub コマンドの際に -lM オプションを指定する必要はありません．

1.4 MSP での設定方法

汎用計算機の MSP システム (ホスト名 kyu-msp) から VPP700/56 に 0.5GB 以上のジョブを投入する場合は，カタログドプロシジャ FORT の VREGION パラメータに **MB** 単位でリージョンサイズを設定して下さい．

```
//A79999AW JOB CLASS=Z
//  EXEC FORT,VPP=YES,OPTION='-Ps -Wv,-m3',VREGION=1740
//FORT.SYSIN DD DSN=A79999A.PROG.FORT,DISP=SHR
//
```

例では，リージョンサイズを 1.7GB(≈1740MB) に設定しています．

1.5 qps コマンド

バッチリクエストの実行状況を表示するコマンド qps(/usr/local/bin/qps) に，実行中・実行待ちのジョブ一覧を表示するオプション，並列ジョブの詳細を表示するオプションがサポートされました．

qps コマンドの形式は

qps␣*options*

です． *options* は次が指定できます．

`-a`	実行中のすべてのジョブを表示する．
`-q`	実行待ちジョブの一覧を表示する．
`-p`	並列ジョブの詳細を表示する．

なお，自分以外の利用者のジョブは user 名が “*******” で表示されます．

●使用例 I

kyu-vpp から`qps -q` を入力し，実行待ちジョブの一覧を表示します．

```
kyu-vpp% qps -q  ↵                    <--- 実行待ちジョブの一覧を表示
No  queue user    request                shell
 1     p8 a79999a 24964.kyu-cc            a.sh
 2     p8 ******* 31220.kyu-vpp    ns03_1_1.nqs
 1    p16 ******* 24495.kyu-cc           c.vpp
 2    p16 ******* 30556.kyu-vpp         sc.nqs
 3    p16 ******* 30611.kyu-vpp         move16
 4    p16 ******* 30750.kyu-vpp        test.sh
 1    p32 ******* 24508.kyu-cc         cuag3_a1
 2    p32 a79999a 24522.kyu-cc            b.sh
```

“No” はそのキュー内での順番を示します．

●使用例 II

kyu-vpp から`qps -a` を入力し，実行中のジョブの一覧を表示します．

```
kyu-vpp% qps -a  ↵                    <--- 実行中ジョブの一覧を表示
que  user    request            cpu       vu  vu/cpu cpu-limit  elapse  v-mem(MB)
 p1 a79999a 35193.kyu-vpp  18:02:31  17:09:26   95%  20:00:00  18:10:29 1240/1600
 p1 ******* 26462.kyu-cc   10:01:41   9:07:15   90%  20:00:00  17:25:08   48/ 512
 p1 ******* 35228.kyu-vpp  16:37:48  16:19:35   98%  20:00:00  16:45:40 1536/1704
 p1 ******* 35236.kyu-vpp   5:28:14   0:02:06    0%  20:00:00   6:10:03  104/ 512
 p8 a79999a 35243.kyu-vpp   waiting for allocation  20:00:00  --------    */1000
p32 ******* 35031.kyu-vpp   1:40:35   1:29:22   88%  20:00:00   2:03:22  936/1704
 mx ******* 35246.kyu-vpp   0:49:27   0:39:04   79%  19:59:46   1:39:33  160/ 512
 mx ******* 35249.kyu-vpp   0:29:31   0:14:44   49%  19:57:33   0:59:24  480/ 752
 d1 ******* 35231.kyu-vpp  15:17:28   5:06:56   33% 160:00:00  15:32:16  160/1792
```

2 Fortran 90/VPP

Fortran 90/VPP の機能追加として，添字式および部分列式の値を検査するデバッグ機能がサポートされました．

並列プログラムのデバッグを行なうためには，オプション -Wx,-Cf で翻訳して作成した実行ファイルを実行します．

以下は，翻訳から実行までのスクリプトの例です．

```
#                                        <---csh で記述
cd EXAMPLE                               <--- ディレクトリの移動
frt -Wx,-Cf -Ps -Wv,-m3 test.f90         <--- デバッグオプションを指定し翻訳
a.out                                    <--- 並列実行
```

例えば，宣言した配列をはみ出したアクセスが起きた場合には，実行時に以下のメッセージが出力されます．

```
 jwe2310i-w The subscript(a) is outside of the specified range. referece value is
 a(130:130:1,641:641:1), specification value is a(1:1024:1,0:0:1)(file=test.f90,
line=38,vpid=6).
 taken to (standard) corrective action, execution continuing.
```

●注意事項

並列デバッグオプションを指定して作成した実行ファイルは，オプションを指定しない場合に比べて大幅に実行時間が増加します[2]．

従って，デバッグオプションを指定して作成した実行ファイルと通常の実行ファイルは区別して利用して下さい．また，並列デバッグは小さいプログラムの規模で行なうことをお勧めします．

[2]今後のレベルアップで更にチューニングが進むとの報告は受けています．

3 Fortran 90/VP

3.1 機能追加

Fortran 90/VP の主な機能追加／強化部分を紹介します．

●翻訳時間の改善

[1] によれば，Fortran 90/VP は従来のベクトル化や最適化にプラスしてスカラープロセッサ向けの最適化を行っています．そのため翻訳時間は一般に VP2600/10 と比較して長くなります．

実際に VP2600/10[3]および汎用計算機 M-1800/20U と比較してどれくらい遅くなるかを，今回のスーパーコンピュータの調達で用いた性能試験プログラムを用いて測定しました．

プログラムの概要は表 1 の通りです．

表 1：性能試験プログラム群

番号	プログラムの概要	行数
job1	粘弾性流動解析	2571
job2	回転非同心円筒内の流体モード解析	2086
job3	シアー表面波の線形安定性問題	1218
job4	Volterra 型積分方程式の離散近似	486
job5	Navier-Stokes 方程式の有限要素解	4289
job6	Stokes 方程式の a priori 誤差評価	2090

プログラムはすべて Fortran の倍精度 (64 ビット) 実数データ型演算です．行数はコメント行も含みます．比較の計算機 VP2600/10, M-1800/20U のコンパイラは双方 FORTRAN77 EX/VP V12L10 です．各測定日時は VP2600/10 が 1996 年 3 月，M-1800/20U は 1997 年 9 月 2 日，VPP700/56 の "Dec. 1996" は，1996 年 12 月 5 日に富士通沼津工場で，"Sep. 1997" は 1997 年 9 月 1 日です．オプションは VPP700/56 の RISC 命令の最適化を抑止するオプション以外は標準値を採用しています．

表 2 に翻訳時間を，図 1 にその値をグラフ化したものをそれぞれ示します．翻訳時間は実行プログラムを作成するまでの CPU 時間を UXP の `timex` コマンドで計測しました．従って，編集・結合の時間も含みます．単位は秒です．有効数字を 2 桁としそれ以下は切捨てています．

表 2：翻訳性能の比較

job 名	VPP700/56		VP2600	M-1800
	(Sep.1997)	(Dec.1996)		
job1	46sec.	58sec.	24sec.	15sec.
job2	43sec.	58sec.	23sec.	15sec.
job3	26sec.	36sec.	14sec.	9.2sec.
job4	5.4sec.	7.9sec.	3.6sec.	2.3sec.
job5	51sec.	74sec.	43sec.	28sec.
job6	21sec.	32sec.	13sec.	7.5sec.

[3] VPP700/56 が導入される前のスーパーコンピュータです．1997 年 2 月末日で運用停止しています．

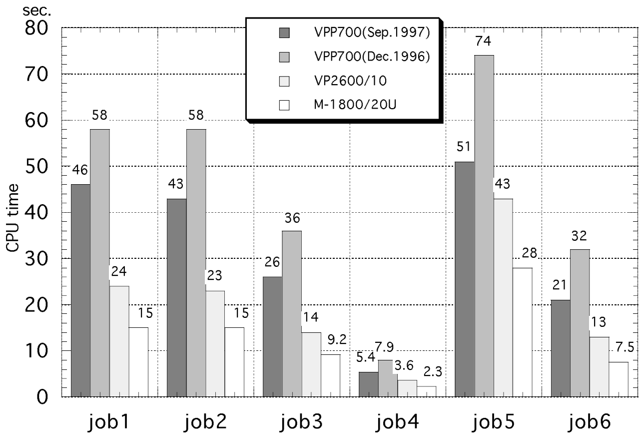

図 1： 翻訳性能の比較 (表 2 のグラフ)

運用開始直前の測定結果では平均で VP2600/10 の約 2.3 倍の翻訳時間でしたが，最近の測定では約 1.6 倍にまで短縮しています．これは約 1.4 倍の高速化です．

●最適化オプションと翻訳時間

翻訳時間は最適化のレベルによって大きく異なります．表 3 は，最適化オプション別の翻訳時間です．測定は 1997 年 9 月 10 日に行ないました．センターの推奨オプション (`-Oe`) は何も最適化オプションを指定しないことですが，例えばデバッグ時に小規模なプログラムを何度も翻訳する場合，`-On` オプションを指定することでかなりの翻訳時間が節約できます．

表 3： 最適化オプション別の翻訳時間 (VPP700/56)

job 名	-On	-Ob	-Oe	-Oe,-R	-Of	-Of+ *opts**
job1	12.62	63.08	62.81	47.64	82.71	104.94
job2	12.62	52.01	58.65	44.36	75.99	99.55
job3	7.65	33.41	34.97	28.62	45.57	61.38
job4	2.38	6.42	6.76	6.12	8.82	11.54
job5	30.88	128.24	133.11	54.94	388.29	395.67
job6	8.28	27.67	30.66	22.59	50.98	68.57

**opts* は “`-KVPP700 -Wv,-Ov`”，単位は秒．

実行時間は，一般に最適化レベルを上げるに従って短くなります．しかし，特に `-Oe` より強い最適化を行なった場合には，翻訳時間が大きく増えたわりに実行時間がほとんど変わらなかったり，逆に最適化による副作用が生じたりする可能性があります．従って，最適化レベルを `-Of` 以上にする場合は，十分に注意して下さい．各最適化の内容は [3], [1]，`man frt` を参照願います．

●デバッグ機能の高速化

デバッグオプションとして，ベクトル実行での検査を可能とする翻訳時オプション `-Dx` が追加されました．`-Dx` は，添字式および部分列式の値を検査する `-Ds`，および未定義データの引用を検査する `-Du` オプションと同時に指定することで，従来阻害されていた DO ループのベクトル化を促進する働きがあります．従って，`-Dsux` などと指定することにより，従来より高速なデバッグが可能になりました．

```
kyu-vpp% frt -Dsux -Wv,-ad test.f90 ↵
```

●べき乗の演算範囲緩和

従来，実数型 `X1`, `X2` に対するべき乗の演算 `X1**X2` では，`X1`$<$ 0.0 かつ `X2`$\neq$ 0 の場合実行時にエラーとなっていました．数学的にはこれで正しい処理ですが，エラーとしてしまうのは少し厳しいという意見があったようで，特に `X2` が整数表現可能な場合，即ち `X2=INT(X2)` の場合は演算可能 (値を返すよう) になりました．

その他，型変換オプションの追加，ベクトル化機能の強化，入出力装置番号の拡張，入出力統計情報の出力などが機能追加されました．

4　C/VP

C/VP は，ベクトル化機能を中心に大幅にオプションが追加になりました．機能の詳細は [2](第 2 版が発行になっています)，`man vcc` を参照して下さい．

ここでは，追加された主な機能，オプションを簡単に説明します．

4.1　while ループのベクトル化

従来 C/VP のベクトル化の対称ループは for ループのみでしたが， while ループのベクトル化も可能になりました．

また，翻訳時オプション `-Wv,-eloop` オプションの指定により，回転数の確定していない while ループもベクトル化することができます．ただし，このオプションは無理矢理ベクトル化を促進するものですので，実行時に異常終了する可能性があります．指定する場合は注意が必要です．

4.2　ループ融合，インライン展開

繰り返し回数が同じである 2 つ以上のループを合わせて最適化を図るオプション `-Wv,-fusion`，およびベクトル化された組み込み関数を引用箇所に展開するオプション `-Wv,-ilfunc` が追加されました．

ともに実行性能を向上する目的のオプションですが，逆に性能が低下したり翻訳時間が増大したりする可能性があります．

4.3　多重 for ループのベクトル化

for ループが入れ子になっている多重ループでは，通常， for 文と for 文の間に実行文がない場合に限って多重ループをベクトル化します．`-Wv,-Mi` オプションを指定するとすべての多重ループをベクトル化します．

4.4　高速な数学関数

高速な sqrt 関数を使用する `-Wv,-Of` オプション，また引数の範囲が $[-\pi/2, \pi/2]$ である場合に限定した sin, cos 関数を使用する `-Wv,-Op` オプションが指定できます．

ただし，高速 sqrt 関数は精度が少し悪くなります．また高速 sin, cos 関数は引数の範囲が $[-\pi/2, \pi/2]$ 以外になると性能が低下します．

以上のオプションをまとめて指定すると次のようになります．

```
kyu-vpp% vcc -Wv,-eloop,-fusion,-ilfunc,-Mi,-Of,-Op sample.c ↵
```

4.5 翻訳情報の出力

frt と同様の機能として，翻訳情報を指定したファイルに出力する -Z オプションが追加されました．-Z の後空白に続けて任意のファイル名を指定します．

```
kyu-vpp% vcc -Z cout sample.c ↵        <--- 翻訳情報の出力
```

5 SSL II

SSL II/VP，SSL II/VPP ともにバージョンアップが行なわれ，幾つかの機能が追加されています．機能の一覧は kyu-vpp の man ssl2vp，man ssl2vpp で，各サブルーチンもそれぞれ man コマンドで検索できます．

5.1 機能追加 (SSL II/VP)

SSL II/VP では拡張機能 II として表 4 のサブルーチンが追加されました．なお，倍精度用は先頭に "D" が付きます．

表 4：SSL II/VP の新機能

サブルーチン名	機能
VMRFT	多重・多次元離散型実 Fourier 変換 (2,3 および 5 の混合基底)
VSRFT	1 次元・多重離散型実 Fourier 変換 (2,3 および 5 の混合基底)

5.2 機能追加 (SSL II/VPP)

SSL II/VPP では表 5 のサブルーチンが追加されました．倍精度用です．

表 5：SSL II/VPP の新機能

サブルーチン名	機能
DP_VTFQD	非対称または不定値のスパース行列の連立 1 次方程式 (TFQMR 法，対角形式格納法)
DP_VTFQE	非対称または不定値のスパース行列の連立 1 次方程式 (TFQMR 法，ELLPACK 形式格納法)
DP_VQMRD	非対称または不定値のスパース行列の連立 1 次方程式 (QMR 法，対角形式格納法)
DP_VQMRE	非対称または不定値のスパース行列の連立 1 次方程式 (QMR 法，ELLPACK 形式格納法)
DP_VSEVPH	実対称行列の固有値・固有ベクトル (3 重対角化，マルチセクション法，逆反復法)
DP_VHEVP	Hermite 行列の固有値・固有ベクトル
DP_VTDEVC	実 3 重対角行列の固有値・固有ベクトル
DP_VLAND	実対称スパース行列の固有値・固有ベクトル (Lanczos 法，対角形式格納法)
DP_VRANN3	正規乱数の生成

なお，実 3 重対角行列の固有値・固有ベクトルを求めるサブルーチン DP_VTDEV は DP_VTDEVC に移行しています．

6　Analyzer

プログラムの性能測定，デバッグ支援ツールの Analyzer が 1997 年 9 月にバージョンアップされました．マニュアルも [4] に改版されています．

6.1　新機能

- 単一 PE で動作する Fortran プログラムの翻訳時と実行時に収集した情報を解析して実行回数やループの回転数などのベクトル化情報を出力するツールである Counter がサポートされました．
- 性能解析ツール Sampler に手続きごとのベクトルヒット率を測定する機能が追加されました．また，動的にリンクされたプログラムへの対応も可能になりました．
- PEPA による C プログラムのデータ採取が環境変数で設定できるようになりました．
- マニュアルに記述されている並列プログラムの対話型並列デバッガ pfdb は運用上の問題から当面利用できません．

6.2　Counter の利用方法

Counter は，単一 PE で動作する Fortran プログラムに対し，手続き，ループ，文の実行回数，ループの回転数，IF 文の正しかった割合などの詳細な実行状況をプログラムリストと共に表示するツールです．

Counter を使用するためには翻訳時オプション `-Wc` の指定が必要です．ただし，`-Wc` オプションの指定により Counter 用のオブジェクトコードが挿入されるため，通常より実行性能が 劣化 します[4]．

•バッチリクエストの記述例

Fortran プログラム "`example.f90`" に対し，翻訳から Counter による解析まで行なうバッチリクエストを記述すると，次の例ようになります．

```
#                                       <--- csh で記述
cd EXAMPLE                              <--- ディレクトリの移動
frt -Wc example.f90                     <--- -Wc オプションを付け翻訳
setenv FJCNT file:counter.data          <--- 環境変数の設定
a.out                                   <--- 実行情報の採取
fjsamp example.ainf                     <--- Counter による解析
```

`frt` コマンドに `-Wc` オプションを付加した翻訳の結果，実行ファイル `a.out` と翻訳情報ファイル `example.ainf` が生成されます[5]．次に `csh` の `setenv` サブコマンドによる環境変数を設定します．環境変数名は "`FJCNT`" です．"`file:`" に続くファイルに実行情報が出力されます．ファイル名は任意です．ここでは "`counter.data`" という名前にしています． Counter による解析は Sampler と同じ `fjsamp` コマンドです．翻訳情報ファイル (例では `example.ainf`) を引数に指定します．`fjsamp` コマンドの機能は kyu-vpp の `man fjsamp` でも検索できます．

[4] この点が Sampler との大きな違いです． Sampler は通常の実行ファイルに対し解析ができるため，実行性能はほとんど変わりません．ただし， Counter の方がより詳細なチューニング情報を収集しますので，状況に応じて使い分けることをお勧めします．

[5] 拡張子 "`ainf`" が自動的につきます．翻訳を対話的に行なうことも可能です．

●解析結果の出力例

Counter の解析結果は標準出力に返却されます．

```
Status                      : Serial
Number of Processors        : 1

Synthesis Information
   Exec-cnt| Loop-leng| Name
      24096|       362|  LAPLACE_
          1|       147|  STRLINE_
          1|      1184|  MAIN__
          -|         0|  VAL_

           |       361|  TOTAL

Program Unit Information(LAPLACE_)++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Procedure List:
   Exec-cnt| Loop-leng| Name
      48192|       241|  laplace.calc_
      24096|     96641|  laplace_

           |       362|  PROCEDURE_TOTAL

 Loop & Array Expression List:
   Loop-exe| Loop-leng| V_Mark| kind  | Line
   19228608|       241|   V   | DO    |  00000060 - 00000065
      24096|     96641|   V   | ARRAY |  00000053 - 00000053
      48192|       399|   V   | DO    |  00000059 - 00000066

 Vectorize Statement List:

  Line         Exec-cnt  true  v o a
  00000046                          !
  00000047        24096                  SUBROUTINE LAPLACE(DELTA)
  00000048                               USE VAL
  00000049                               REAL*8 DELTA
  00000050        24096                  CALL CALC(P1, P2)
  00000051        24096                  DELTA = 0.0
  00000052        24096                  CALL CALC(P2, P1)
  00000053        24096        v   v     D = ABS(P2 - P1)
  00000054        24096                  DELTA = MAXVAL(D)
  00000055                          !
  00000056                               CONTAINS
  00000057        48192                    SUBROUTINE CALC(PIN, POUT)
  00000058                                 REAL*8 PIN(IXMESH,IYMESH), POUT(IXMESH,IYMESH)
  00000059        48192        v           DO J=2, IYMESH-1
  00000060     19228608        v             DO  I=1, IXMESH
  00000061     4.63E+09  83.1  v               IF (MBOUND(I, J)) THEN
  00000062     3.85E+09        v                POUT(I,J)=(PIN(I+1,J)+PIN(I-1,J)+PIN(I,J+1)&
  00000063                                      +PIN(I,J-1))*0.25
  00000064     4.63E+09        v               ENDIF
  00000065     4.63E+09        v             END DO
  00000066     19228608        v           END DO
  00000067        48192                    END SUBROUTINE CALC
  00000068        24096                  END

Message List:

 vectorization messages:
 Internal subprogram name(laplace.calc_)
    jpc1101i-i  ``sampler-spe.f90'', line 61 - 65: Vectorized by DO variables J and I.
    jpc1002i-i  ``sampler-spe.f90'', line 62 - 62: Vectorized with masked operation method.
                                   :
```

もしバッチリクエストを sh で記述している (即ち先頭の # をつけない) 場合，環境変数の設定は

```
FJCNT=file:counter.data
export FJCNT
```

となります．

6.3 環境変数の設定による PEPA の利用 (C/VP)

PEPA は PE の動作に関する情報を採取するツールです．従来 PEPA の環境変数の設定による使用は Fortran のみでしたが，今回のバージョンアップにより C/VP プログラムにも適用可能となりました．

環境変数は Fortran と同じ FJPEPA，変数は ON です．バッチリクエストの記述例は以下の通りです．a.out は C/VP により作成した実行ファイルです．

```
#
cd EXAMPLE
setenv FJPEPA ON                  <--PEPA 環境変数の設定
a.out                             <-- 実行
```

採取された情報は標準エラー出力に書き出されます．採取項目については [3] または [4] を参照して下さい．

参考文献

[1] UXP/V Fortran 90/VP 使用手引書 V10 用, J2U5-0050, 富士通株式会社 (1995).

[2] UXP/V C/VP 使用手引書 V10 用, J2U5-0120, 富士通株式会社 (1997).

[3] VPP700/56 利用の手引 (第 1.0 版)， 九州大学大型計算機センター・プログラムライブラリ室 (1997).

[4] UXP/V アナライザ使用手引書 V12 用, J2U5–0131, 富士通株式会社 (1997).

[5] 渡部 善隆：VPP700/56 での対話型処理, 九州大学大型計算機センター広報, Vol.30, No.2, pp.167–185 (1997).