サイエンティフィック・システム研究会
科学技術計算分科会　第１回会合資料

平成 11 年 8 月 20 日

原研における原子力コード高速化への取り組みと プログラミング事例

日本原子力研究所
計算科学技術推進センター
情報システム管理課
久米悦雄，根本俊行
kume@brian.tokai.jaeri.go.jp
toshi@aries.tokai.jaeri.go.jp

1．はじめに

日本原子力研究所（以下原研）には、富士通を初めとする、IBM、インテル、NEC、日立等、世界有数のスーパーコンピュータが運用されており、一部特殊計算専用機を含め、その数は、各地区（東海、那珂、中目黒、寝屋川）合わせて 13 システムに上る。

これらシステムは、東海地区に VPP500/42、AP3000/24、GS8400/20（平成 11 年 8 月より GS8300/10N へ移行）、Monte-4（平成 10 年度で運用終了）の 4 システム、那珂地区に Paragon/256、SP2/4 の 2 システム、中目黒地区に VPP300/16、SX-4/２Cx3、T94/4、SR2201/64、SP2/48 の 5 システム、寝屋川地区に VPP300/12、Paragon/834 の 2 システムと、各地区に分散配置されている。各システムは、主に、東海地区では計算科学や基礎基盤、安全性研究の分野で、那珂地区は核融合研究分野、中目黒地区は計算科学（並列処理研究）分野、寝屋川地区は光量子研究分野に利用されている。システム全体では、計算科学分野で約 32%、光量子約 30%、核融合約 27%、基礎基盤約 9%の CPU 時間使用率となっており、年間約 4,700 千 CPU 時間の利用がある（平成 10 年度実績）。

原研計算科学技術推進センター情報システム管理課では、これら各種のスーパーコンピュータの効率的な運用とコンピュータ資源の有効利用を促進するため、ユーザの利用が多く、計算機資源をより多く必要とする原子力コードを、ユーザに代わってスーパーコンピュータ上に整備し、それぞれのコードに最適な高速化を施す作業を実施している。

これら高速化作業は年間十数件ほど行っており、個々のコードに合わせた機種選定を含め、解析規模の拡大や高速化のための計算アルゴリズムの見直しをも視野に入れた、ベクトル化及び並列化チューニングを実施し、ユーザに提供している。これらのチューニングにより、コードやその計算規模によって様々ではあるが、数倍から数十倍の高速化が実現されている。

本報は、このうち東海地区の VPP500 を中心とした、高速化作業への取り組みについての現状報告である。高速化作業体制とここ数年の成果の概要を中心にプログラミングの具体例を交えて紹介する。

2. 計算機システム

原研が保有する計算機システムについて述べる。

2.1　構成

原研の各研究所に配置されている計算機の一覧を表 2.1 に示す。

表 2.1　原研における計算機システム（平成 11 年 8 月現在）

東海研究所

VPP500/42	分散メモリ型ベクトル並列計算機（富士通）
VPP500 増設部	分散メモリ型ベクトル並列計算機（富士通）
AP3000/24	共用 UNIX サーバ（富士通）
GS8300/10N	汎用計算機（富士通）

那珂研究所

Paragon/256	分散メモリ型並列計算機（インテル）
SP2/4	解析/DB サーバ（ＩＢＭ）

東京（中目黒）

VPP300/16	分散メモリ型ベクトル並列計算機（富士通）
T94/4	共有メモリ型ベクトル並列計算機（クレイ（SGI)）
SX-4/2Cx3	共有/分散メモリ型ベクトル並列計算機（ＮＥＣ）
SR2201/64	多次元クロスバー型スカラ並列計算機（日立）
SP2/48	多段結合型スカラ並列計算機　（ＩＢＭ）

関西研究所

VPP300/12	分散メモリ型ベクトル並列計算機（富士通）
Paragon/834	分散メモリ型並列計算機　（インテル）

2.2　スペック

各計算機におけるスペックを表 2.2 に示す。

表 2.2　各計算機のスペック

	機種	1PE ﾋﾟｰｸ性能(MFLOPS)	PE 数	1PE 主記憶(MB)/総主記憶(GB)	PE 間結合方式	ﾃﾞｰﾀ転送速度(MB/S)	ディスク容量(GB)
東海	VPP500/42	1600	42	256/10.5	クロスバ	400	180
	VPP500 増設部	2400	12	2048/24.0	クロスバ	570	60
	AP3000	600	24	2048/23.5	2 次元ﾄｰﾗｽ	200	450
	GS8300/10N	20	1	0.25	-	-	117
東京	VPP300/16	2200	16	512/9.5	クロスバ	570	100
	T94/4	1800	4	共有型 1.0	-		51
	SX-4/2Cx3	2000	6	512/1.5	ｸﾛｽﾊﾞ接続	8000	34
	SR2201/64	300	64	256/16.0	2 次元ｸﾛｽﾊﾞ	300	39
	SP2/48	266	48	128/6.0	多段結合	40	27
那珂	Paragon/256	75	256	32/8.0	2 次元ﾒｯｼｭ	175	52
	SP2/4	308	4	256/1.0	多段結合	150	68
関西	Paragon/834	150	834	128/104.3	2 次元ﾒｯｼｭ	175	416
	VPP300/12	2200	12	512/6.0	クロスバ	570	60

2.3　運用

ここでは主にコード作成、コード実行及び高速化作業が行われる東海研究所の VPP500/42、および VPP500 増設部の運用について述べる。シングルジョブ、並列ジョブ共に NQS によるバッチ処理方式で運用している。プログラムの作成、修正、ジョブの投入等は VPP500、VPP500 増設部のフロントエンドマシンである GSP によって行っている。コードの実行はすべてバックエンドマシンである VPP500、または VPP500 増設部で行う。

バッチジョブは、使用する計算資源（CPU、メモリ、PE 数）によってキュークラスが設定されている。ユーザはこれらの要素でジョブを投入するクラスを決定する。

また、一人の利用者が計算機を占有しないよう、各キュークラスには“重み”が設定されている。利用者は重みの合計がある一定の値までならば、自由にキュークラス選んでジョブを投入することができる。なお並列クラスについては、申請があれば設定クラス以上の PE を使用できることになっている。

2.4　利用状況

東海研究所における平成 10 年度の利用状況を表 2.3 に示す。

表 2.3 原研大型計算機システムの利用状況（平成 10 年度）

地区	機　種	CPU 使用時間(H)	CPU 使用率(%)	利用人数	処理件数
東海	VPP500/42	237,117	63	399	207,570
	AP3000/24	169,995	58	336	149,838
	GS8400/20	4,617	27	948	301,760
	Monte-4	11,960	35	62	74,477

3．高速化作業体制

3.1　要員

高速化作業の実施に当たっては、１名の管理者、１名の主任技術者及び５名の技術者が担当する。管理者はスケジューリング及び全体の進捗管理等のマネージメントが主であり、実際の高速化作業は主任技術者以下６名が実施する。

3.2　対象コードの選定

高速化対象となる原子力コードを選定するに当たり、まず候補となるコードが次の４つの場合から挙げられる。

1)年１回の所内一般公募
2)CPU 利用時間が長いコード
3)プログラム相談窓口からの依頼
4)プログラム開発整備作業からの受託

1)は毎年 12 月に次年度高速化希望コードを所内ユーザを対象に一般公募するものであり、例年 10 件ほどの応募がある。2)は毎月の計算機利用状況を踏まえ、CPU 使用時間が長く高速化の可能性のあると思われるコードをピックアップし、運用／管理者側から当該コードの利用者に対し高速化への要望を伺うもので、年間数本のコードが対象となる。3)は情報システム管理課が設けるプログラム相談室からの依頼によるもので、ユーザがプログラム開発中に発生した問題解決のために利用する、プログラム相談室での相談内容から、高速化等のプログラミングに係わるコードが候補となる。4)は 1)と同時に公募し情報システム管理課で別途実施している「プログラム開発整備作業」の作業結果に基づき、高速化の必要のあるコードを受託するものである。

候補となったコードに関しては、ユーザにコード高速化に関する調査用紙への記入をお願いし、その内容を元に直接ユーザとミーティングを持ち、それらの調査結果から高速化対象コードを選定する。なお、当該高速化作業には高速化に関する調査等も含まれる。

3.3　作業指針

対象コードの高速化チューニングは、極力その計算内容の把握に勤めコードの分析を十分に実施することから始める。ユーザの希望だけではなく、コードの特性や計算機環境に合わせ、他の計算機での高速化の可能性の検討も行っている。また、ベクトル化や並列化の施行も、解析規模や実行回数を考慮し、それらに合わせたいくつかの高速化コード（バージョン）を作成する場合もある。

並列化に関しては今後のより高性能な並列型計算機への交換をも考慮し、移植性・汎用性を重視した並列化技術である MPI（Message Passing Interface）の利用を推進している。

また、高速化チューニングに限らず、コードの利用・実行に際しての使いやすさをも考慮した、ユーザインタフェースや出力グラフ等の整備も併せて実施している。

これらの詳細については４章で述べる。

3.4　作業の進捗及び成果

高速化作業は、プログラム解析ツールを用いたプログラム分析に始まり、チューニング指針の策定、基本チューニング（プログラム分析に基づいたチューニング作業）、高度チューニング（アル

ゴリズムレベルを考慮したチューニング指針の作成とプログラム変更)、そしてチューニング後の性能評価という流れになる。一連の作業は、数ヶ月から１年に及ぶものもあり、月１度の定例会でその進捗状況を報告／議論している。

また、作業成果に関してはユーザへの作業報告書とは別途、年次報告書として、原研報告書（JAERI-Data/Code）に毎年の成果をまとめると同時に、所内向けホームページでその成果概要を公開している。これらの詳細については５章で述べる。

4. 高速化プログラミング事例

ここでは、原研において高速化技術者が実際に行っている高速化作業の手順及び事例等を紹介する。

4.1　作業の流れ

VPP500 における高速化作業の進め方について説明する。

4.1.1　作業の分類

ユーザより入手した原子力コードは、ユーザの要望に沿って作業を行うことになるが、使用状況等により表 4.1 のような作業分類を行う。

表 4.1　作業の区分け

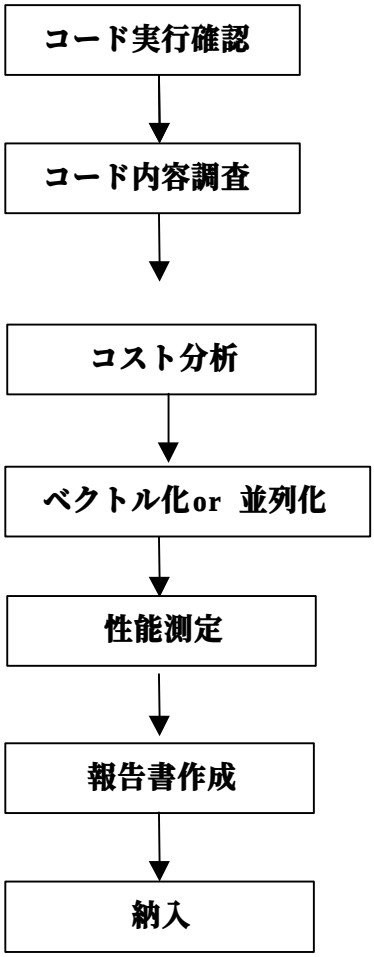
要望	作業
解析規模の拡大（メモリの拡大）	**（ベクトル）並列化**
高速化	実行回数が少ない、または実行時間が比較的短い場合　→　**ベクトル化**
	実行回数が多い、または実行時間が極めて長い場合 →　**ベクトル並列化**

解析規模を大きくする場合は、1PE のメモリでは実行不可能なため、メモリ分割による並列化が必然となる。一方、高速化の場合、原則的にコードの性質、運用上の条件を加味し、表 4.1 に準じて作業を行っている。これは、CPU の有効利用にも関係するものである。

さらに、コードによってはベクトル化が困難な場合がある。このような場合には、原研で導入している AP3000 へのインストール、さらに並列化を行う場合もある。

4.1.2　作業手順

コードを入手してからベクトル化・並列化までの一般的な手順を示す。



正しいオリジナルコードかチェック。

コードの計算内容・構造等のチェック。

ANALYZER、SAMPLER（参考文献(1)(2)）等を用い CPU 時間分布の測定。

実際のチューニング。

性能を測定し、不十分なら再度チューニングを加える。

コードのチューニング内容、使い方等をまとめる。

コード、報告書の納入、および使い方等の説明。

4.2　手法について

ベクトル化・並列化を行うにあたり、その手法について以下にまとめる。

4.2.1　コード変更に関する注意

ベクトル化・並列化のためにコードを変更することになるが、手法によっては、オリジナルのコードを大幅に変更することがある。一方、原研では高速化チューニングを施したコードに対して、後にユーザが（高速化とは無関係な）機能拡張等のための変更を加えることがある。このような場合、チューニングされたコードをユーザ自身が変更することは困難である。このため、作業前にユーザの確認を取り、高度チューニングの範囲を限定する場合がある。

4.2.2　並列化言語

並列化を行う場合、VPP FORTRAN と MPI のどちらかを使用することになる。原研では前述のとおり複数の並列計算機を所有しているため、VPP500 以外でコードを実行する場合がある。このため、最近は MPI によって並列化を行う傾向が高い。

4.2.3　並列軸について

ベクトル化、並列化を行っていく上で重要なポイントとなるのは、どの軸でベクトル化、または並列化をしていくかということである。特に並列化の場合、並列軸の選択次第で各 PE の負荷分散や通信の発生に影響を与えるため、作業前に慎重な調査を要する。また、ベクトル軸との兼用をした場合、（ベクトル長が短くなることで）ベクトル性能の低下を招くことがあるので、なるべく避

けるようにしている。

4.3　原研における原子力コードの高速化の例

ここでは、実際に原子力コードをベクトル並列化した例を示す。対象としたコードの概要を表 4.2 に示す。

表 4.2　ベクトル並列化した原子力コードの概要

コード	非線形フォッカープランクコード
内　容	フォッカープランク方程式を解くことにより、プラズマ加熱過程における粒子の散乱・エネルギー解析等を行う。
コードの特長	元々CRAY 計算機にて開発されたコードで、一部原研で変更が加えられている。
ユーザの希望	高速化および解析対象の大規模化

4.3.1　コード調査

当該コードは CRAY 計算機で開発されたため、基本的にベクトル処理できる構造になっている。また、2 次元コードであり、ベクトル軸としては X 方向、もしくは Y 方向となる。

しかし、オリジナル版でのベクトル化状況はあまりよくない。当該コードの VPP500 へのインストール時に、CRAY 特有の組み込み関数（CVMGM）が VPP500 では用意されていないため、これを模擬した外部関数を作成し代用している。このため、この関数は CRAY 上ではベクトル化対象組み込み関数であったが、VPP500 では一般の外部関数となり、この呼び出し部分はベクトル化対象外となってしまった。この他、細かい所でベクトル化されない要素を含んでいる。

規模拡大のための並列化を検討した結果、このコードでは領域分割が有効であることが判明した。コード内で使用される配列（array）は、領域（メッシュ）に関するものが大部分であり、これを効果的に分割することで、解析対象の大規模化が実現できる。

これらの結果、原則的に X 方向をベクトル軸、Y 方向を並列軸に採用することにした。

4.3.2　コスト分析

CPU コスト分布を調査するため、SAMPLER 及び ANALYZER を実行した。SAMPLER は実際の VPP500 上での CPU 分布を、ANALYZER は副プログラムの詳細な情報を得るためのものである。表 4.3、表 4.4 にそれぞれの実行結果を示す。表 4.3 から分かるように、上位 5 ルーチンがコストの約 98%を占めており、これらを高速化の対象とした。

表 4.3　SAMPLER の結果（抜粋）

```
Synthesis Information
     Count|    Percent|     VL| Name
      6187|       44.5|    445|  sub1
      4820|       34.7|      -|  cvmgm_
      1234|        8.9|      -|  sub2
      1091|        7.9|     61|  sub3
       290|        2.1|    101|  sub4
                    ・
                    ・
                    ・
     13888|           |    149|  TOTAL
```

表 4.4　ANALYZER の結果（抜粋）

name	ex-count	v-cost(%)	s-cost(%)	v-leng	v-rate	v-effect
sub1	2000	62.2	46.5	92	41.0	1.6 - 1.7
sub4	2000	15.5	13.9	76	51.4	1.9 - 2.0
sub	0	10.0	13.0	2747	66.5	2.8 - 2.9
>sub2	16013					
cvmgm	.1228E09	8.2	3.7	1	0.0	1.0 - 1.0
sub3	13	4.0	22.5	61	96.8	10.7 -14.1
			・ ・ ・			
(total)	----------	100.0	100.0	----	57.0	2.2 - 2.2

sub1:フォッカープランク方程式を解きながら粒子の物性値を解く。
sub2:配列の操作を行う。
sub3:ある計算ルーチンであるが、後にユーザのコード修正で廃止。
sub4:フォッカープランク方程式解法ルーチンのひとつ。

4.3.3　ベクトル化

当該コードにベクトル化チューニングを施した一例を以下に示す。

前述したように、CRAY 計算機で開発したコードが元になっているので、ベクトル化を意識したコーディングになっている（図 4.1）。CRAY 組み込み関数 CVMGM を模擬した関数（図 4.2）を、コンパイルオプションによるインライン展開をすることでベクトル化が実現できた（図 4.3）。

参考として、ベクトル化による性能向上率の測定結果を表 4.5 に示す。

```
      DO 33 I=I1LIM,I2LIM
      EG(I,ITEMC1(I))=CVMGM(0.,ALPB(XCON1,CA(I,2),XCON2,CB(I,1),CB(I,2)
     C )*TEMC2(I),JMIN(I,K) .GE. 1)
      FG(I,ITEMC1(I))=CVMGM(FJMIN(I,K),(DELB1(RBGN,TRT(I,1),TEMP1(I,1),
     C CDH(I,1))+TEMC8(I)*DELB2(XCON3,CC(I,1),TEMP1(I+1,1),
     C TEMP1(I-1,1),CC(I,2),TEMP1(I+1,2),TEMP1(I-1,2),TEMC9(I)))*
     C TEMC2(I),JMIN(I,K) .GE. 1)
   33 CONTINUE
```

図 4.1　　オリジナル版 CVMGM 引用部分

```
      FUNCTION CVMGM(AXYZ,BXYZ,XXYZ)
      LOGICAL XXYZ
      CVMGM=BXYZ
      IF (XXYZ) CVMGM=AXYZ
      RETURN
      END
```

図 4.2　　CRAY 組み込み関数 CVMGM を模擬した関数

```
v        DO 33 I=I1LIM,I2LIM
vi       EG(I,ITEMC1(I))=CVMGM(0.,ALPB(XCON1,CA(I,2),XCON2,CB(I,1),CB(I,2)
         C )*TEMC2(I),JMIN(I,K) .GE. 1)
vi       FG(I,ITEMC1(I))=CVMGM(FJMIN(I,K),(DELB1(RBGN,TRT(I,1),TEMP1(I,1),
         C CDH(I,1))+TEMC8(I)*DELB2(XCON3,CC(I,1),TEMP1(I+1,1),
         C TEMP1(I-1,1),CC(I,2),TEMP1(I+1,2),TEMP1(I-1,2),TEMC9(I)))*
         C TEMC2(I),JMIN(I,K) .GE. 1)
v    33 CONTINUE
```

図 4.3　コンパイラオプションによるインライン展開でベクトル化できた例

表 4.5 ベクトル化版の性能（参考）

バージョン/実行形態	経過時間	CPU 時間	VU 時間	倍率(CPU)
オリジナル版スカラ(0e)	7m17.12s	6m56.25s		1.0
オリジナル版ベクトル(0e)	2m49.11s	2m21.90s	14.01s	2.9
オリジナル版ベクトル(0f)	50.53s	39.11s	16.31s	10.6
ベクトル化版ベクトル(0f)	29.55s	19.20s	16.08s	21.7

4.3.4　並列化

当該コードに Y 方向の領域分割を行う並列化を施した一例を以下に示す。

領域分割の概念を図 4.4 に示す。

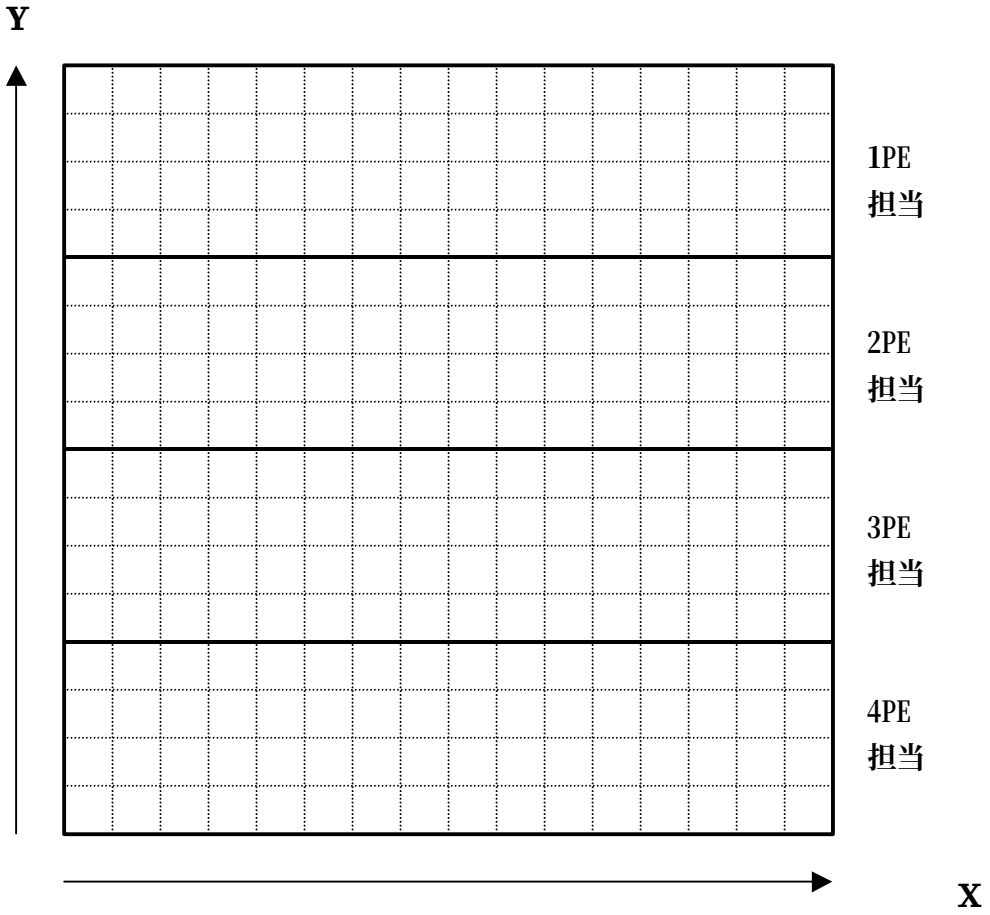


図 4.4　領域分割の概念

(1)メモリ分割

領域の大きさに関連を持つ配列すべてに対し、分割を行う。Y 方向で分割するが、ある物性値を持つ格子に関する配列においては、ある格子点におけるその値が、隣の位置の物性値の影響を受ける場合がある。このため、場所によっては PE をまたいで参照する場合があり、袖を持たせている。オリジナルの配列宣言文を図 4.5 に、袖を持たせて分割した配列宣言文を図 4.6 に示す。

```
      COMMON /MCOEF/ EG(IYDMA,JXA),FG(IYDMA,JXA)
      COMMON/MTEMP/TEMP1(IYDMA,JXA),TEMP2(IYDMA,JXA)
```

図 4.5　オリジナルの配列宣言の例

```
!XOCL INDEX PARTITION Q=(PROC=P,INDEX=1:JXA,PART=BAND,OVERLAP=(1,1))
!XOCL INDEX PARTITION Q3=(PROC=P,INDEX=1:JXA,PART=BAND,OVERLAP=(2,2))

      COMMON /MCOEFP/ EGG(IYDMA,JXA),FGG(IYDMA,JXA)
      DIMENSION       EG(IYDMA,JXA),FG(IYDMA,JXA)
      COMMON/MTEMPP/TEMP1G(IYDMA+1,JXA),TEMP2G(IYDMA,JXA)
      DIMENSION TEMP1(IYDMA+1,JXA),TEMP2(IYDMA,JXA)

!XOCL LOCAL EG(:,/Q),FG(:,/Q)
!XOCL GLOBAL EGG,FGG
!XOCL LOCAL TEMP1(:,/Q3),TEMP2(:,/Q)
!XOCL GLOBAL TEMP1G,TEMP2G
      EQUIVALENCE    (EG,EGG),(FG,FGG)
      EQUIVALENCE    (TEMP1,TEMP1G), (TEMP2,TEMP2G)
```

図 4.6　分割した配列宣言の例

(2)命令の分割

配列の分割に伴い、正しく並列処理が行えるよう命令に対しても分割を行う。

一例として、図 4.7 に示した DO ループを対象とし、その処理を分割し並列化したものを図 4.8 に示す。

```
      DO 8061 J = 1, JJMAX(K)
      DO     61 I = 1, IYDM
        EG(I,J)=CVMGM(0.,TEMP1(I,J),TEMP1(I,J) .LT. 0.)
        TEMP2(I,J)=CVMGM(0.,-TEMP1(I,J),TEMP1(I,J) .GT. 0.)
   61 CONTINUE
 8061 CONTINUE
```

図 4.7　オリジナル版並列化対象ループの例

```
!XOCL                    INDEX                    PARTITION
Q2=(PROC=P,INDEX=1:JXA,PART=BAND)

!XOCL SPREAD DO/Q2
        DO 8807 II = 1,JJMAX(K)
            IF( ISTA0.EQ.0) ISTA0 = II
            IEND0 = II
 8807 CONTINUE
```

← 別ルーチンで DO ループ・Y 方向(J)の範囲を設定

```
      DO 8061 J = ISTA0, IEND0
      DO     61 I = 1, IYDM
         EG(I,J)=CVMGM(0.,TEMP1(I,J),TEMP1(I,J) .LT. 0.)
         TEMP2(I,J)=CVMGM(0.,-TEMP1(I,J),TEMP1(I,J) .GT. 0.)
   61 CONTINUE
 8061 CONTINUE
```

図 4.8　並列化した例

4.3.5　ベクトル並列化版の性能

ベクトル並列化版を使用して速度向上率、解析可能サイズの調査を行った。

(1)速度

速度向上率についての測定結果を表 4.6 及び図 4.9 に示す。

表 4.6　ベクトル並列化版速度性能

	経過時間	倍率
ベクトル化版	1h18m02.36s	1.0
並列化版(4PE)	26m38.11s	2.93
並列化版(8PE)	19m16.61s	4.04
並列化版(16PE)	16m51.93s	4.62

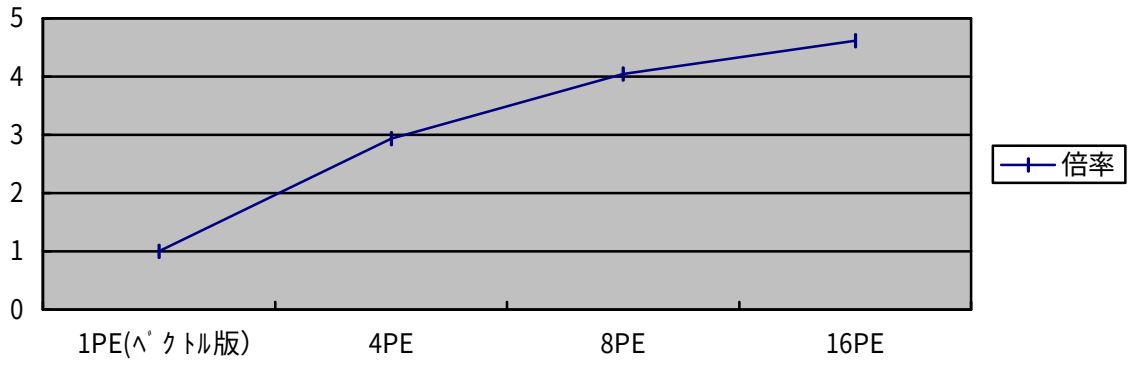


図 4.9　ベクトル並列化版速度向上率

並列化による台数効果は使用 PE が多くなるにつれ低下する。これは、コード内で並列化不可部分があること、データ転送時間によるオーバーヘッド等が原因である。

(2)解析可能サイズ

解析可能サイズについての測定結果を表 4.7 に示す。

表 4.7 ベクトル並列化版解析可能サイズ

バージョン	最大メッシュサイズ (X×Y)	比率
シングル版	751×501	1.0
並列化版(4PE)	1348×1348	4.8
並列化版(16PE)	2696×2696	19.3

解析可能サイズが台数分より大きくなっているのは、不要な配列の削除、配列の兼用のチューニングを行ったためである。

4.4 大規模計算

大規模計算は

1)メモリを多く使用することで解析対象を大きくする or メッシュを細かくする。

2)高速化によって時間発展シミュレーションを長時間にわたって行う or 時間ステップを小さくして詳細な解析を行う。

の２つに大別される。

原研においても粒子（分子）系コードや流体系コードに関しては、速度、メモリの要求は年々高くなり、現状の VPP500 ではパワー不足になりつつある。また、原子炉事故時等のシミュレーションを行う安全性コードに関しては、メモリはともかく速度を要求される場合が多く、より高速な CPU が望まれている。

4.5 高速化作業のまとめ

原研における高速化作業において一番の長所は、高速化技術者が実際に使用するユーザと同じ原研内に常駐していることである。これは、

1)高速化作業中にユーザと密に接触できることで、作業が比較的スムーズに進めやすい。

2)コード納品後、コードの使い方、ユーザによるコード改良及び問題発生等において、高速化作業者が緻密なフォローができる。

という大きな利点がある。

また、作業を行ったコードに対し、すべて報告書を作成・保存しており、これらの高速化のノウハウを蓄積することで、代々の作業者の参考となっている（詳細は 5 章)。

しかしながら、問題点もいくつか存在する。コードをベクトル化・並列化する場合、コード内容を把握してから実施することが望ましい。原研においては、ごくまれにコードに関する資料等が無い場合がある。ユーザがコード内容について把握していれば、さほど問題にはならないが、ブラックボックス的に使用しているユーザもおり、コードの内容を理解していない場合がある。このよう

な時においては、コード理解に長時間を要したり、表面的なチューニングしか行えない場合がある。
次に高速化、大規模化に伴う問題点を述べる。これらの作業の結果、（時間発展シミュレーションの長時間化等により）ディスク上に書き出すファイル（計算結果）が極めて大きくなる場合がある（この計算結果は図形化や統計計算等に用いられる）。この場合、ファイルの書き出しに長時間を要する。VFL-FS（Very Fast and Large File System）を使用しても、格段に速くなるわけではない。コードがベクトル並列化により高速になっても、I/O に多くの時間を費やしては CPU の有効利用にはならない。改善が望まれる点である。

5．成果

5.1　成果一覧

平成 9 年度及び 10 年度に実施した高速化作業の成果一覧を付録 1 に示す。
付録 1 からも分かるように各コードとも並列化（ベクトル化）による高速化効果が顕著に表れており、CPU 時間の大幅な短縮がなされている。また、並列化によるメモリ分割の効果や計算時間の短縮により、大規模ジョブの実行が可能になっている。さらに、高速化及び計算結果の検証過程で潜在的なバグが発見・修正されるなどの副次的な効果もある。
一方、一部コードに関しては、まだ残された課題、改良すべき点、すなわち高速化の余地が残されているものもある。今後これらの点を見極め、各コードの計算アルゴリズムの見直し、I/O の効率化等を進めることで、より一層の高速化が期待できるものと思われる。

5.2　成果の公開と管理

３章で述べたように、高速化作業の成果はユーザへの作業報告書とは別途、年次報告書として、原研報告書（JAERI-Data/Code 等）に毎年の成果をまとめると同時に、原子力学会等での発表、所内ユーザ向けのコンピュータ情報誌及び所内向けホームページでの成果概要公開を行っている。
また、高速化されたコードのいくつかは、所内「コンピュータプログラム等取扱規程」に則り、プログラム等審査会を経て、情報システム管理課で一元管理すると同時に、（財）高度情報研究機構を通じて所内外へ配布されている。

(1)JAERI-Data/Code
年度毎の成果を作業内容によって 3 つに大別し、それぞれベクトル化編、並列化編及び移植編の 3 分冊を刊行している。これらは一般公開用の刊行物であり、入手希望者は筆者まで連絡されたい。原子力学会等での発表及び所内ユーザ向けのコンピュータ情報誌への寄稿等を合わせて、ここ数年のリストを参考文献(3)～(14)に示す。

(2)ホームページ
ホームページの利用は、これらの成果（情報）の有効利用と効率的な維持管理を目的としている。報告書等にまとめられた成果は、今後の作業実施に関し非常に有用な情報であり、これを有効に活用するため、すなわち情報／資源の共有化を計り業務管理効率の向上のため、これら成果を電子化しそれを管理する DB システムとして、Web を利用したシステムの構築を行っている（イントラネット用）。
図 5.1 に平成 10 年度の高速化作業一覧を示す。ここでは、コード名、依頼元及び対象計算機に

関する情報のみを表示し、概要ボタンをクリックすると図 5.2 に示したような、個々のコードに関する作業概要が表示されるようになっている。また、作業者専用のページでは電子化（PDF）された報告書の閲覧も可能である。

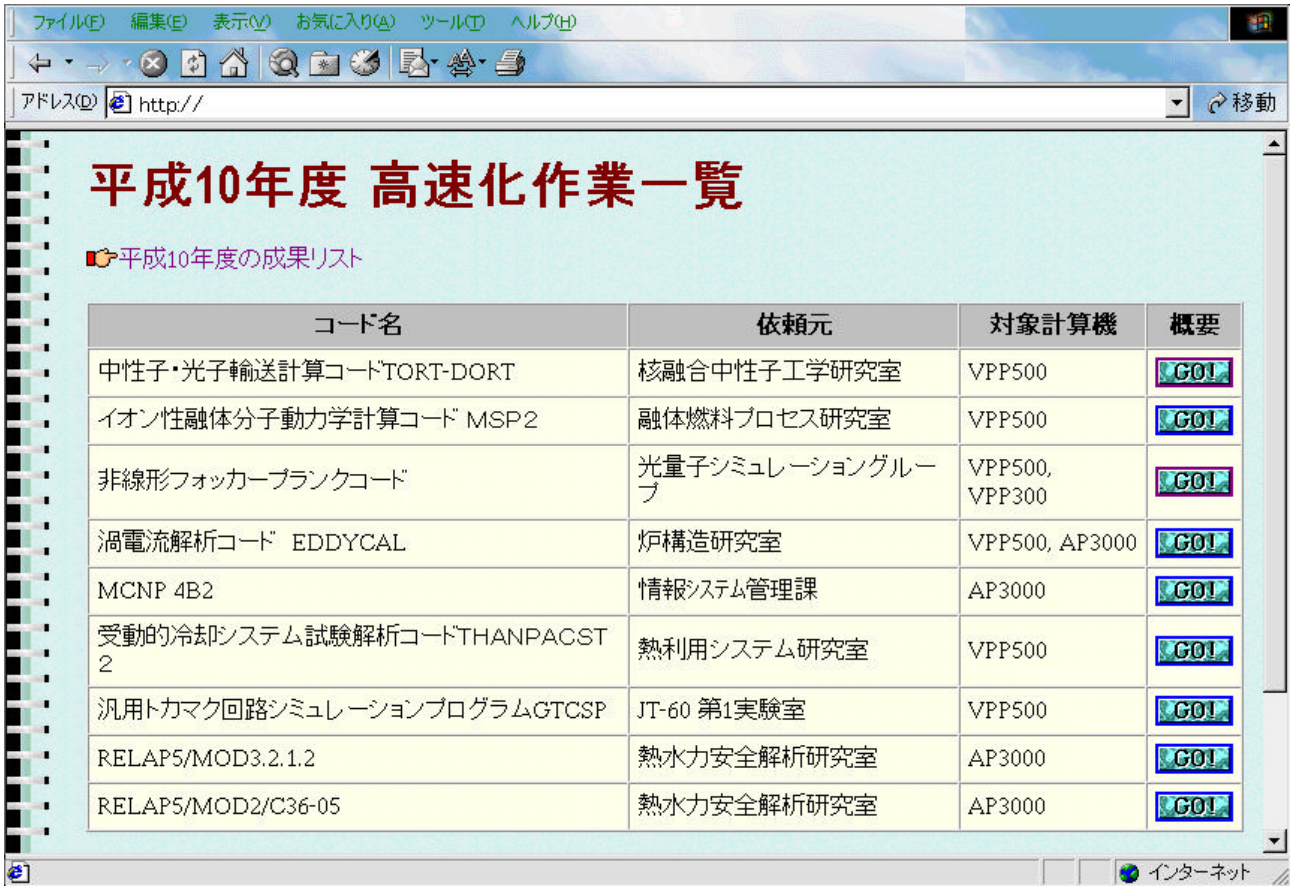


図 5.1　平成 10 年度高速化作業一覧

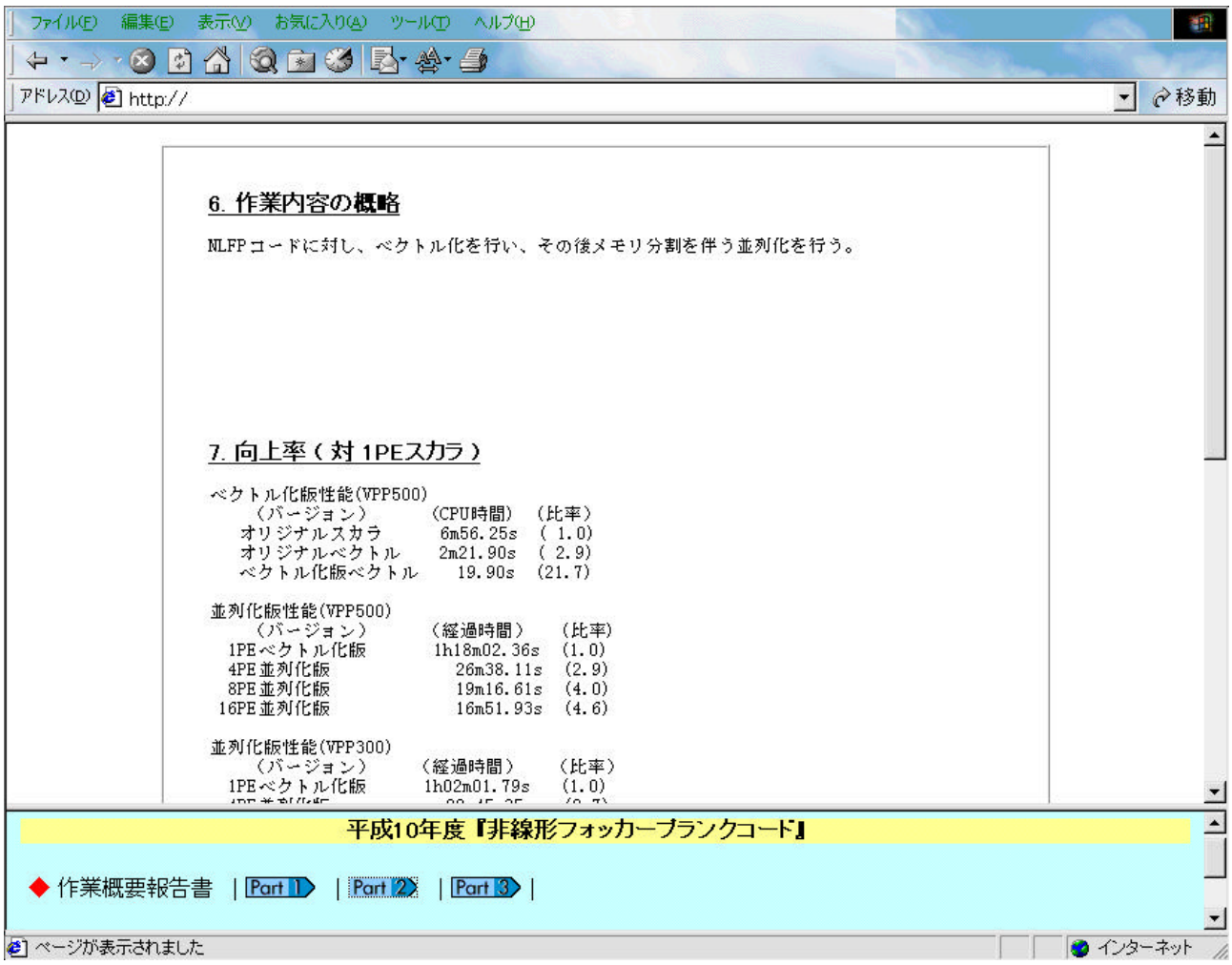


図 5.2　高速化作業概要報告書

これらホームページで示される情報は、作業者毎に Web からデータの登録が行えるようになっており、付録 1 のような一覧表の作成から作業概要報告書まで、自動的に作成されるシステムとなっている。

その他、当該ホームページにはスケジュール表や一般応募フォーム等も掲載されている。なお、所内向けのページであり外部からの閲覧は出来ない。

6. おわりに

原研計算科学技術推進センター情報システム管理課で実施している原子力コードの高速化作業は、毎年十数件を順調にこなし、平成 9 年度に 14 件の作業を、平成 10 年度には 9 件の作業を完了し、平成 11 年度にも 14 件の作業が計画されている（VPP500,VPP300,一部 AP3000)。これら作業は、ユーザからの要望に応じ、原子力コードを 各種スーパーコンピュータ向けに最適なベクトル化、並列化を施すチューニングを行うものであり、コード実行時間の大幅な短縮に寄与している。さらに、単一プロセッサ上ではメモリ不足から実行できないようなジョブを並列化効果により可能にするなど、計算機のスループットの向上、ターンアラウンドタイムの短縮、それによるユーザの仕事の効率化、計算可能なジョブの範囲の拡大など、計算機の効率的な運用と計算機資源の有効利用に大いに貢献するものと考えている。

本報では、原研における原子力コード高速化への取り組みの現状を、高速化プログラミングの具体例（非線形フォッカープランクコード）を交えて紹介した。チューニングによる高速化効果はいずれのコードにおいても顕著であり、計算機資源の有効利用につながっていると推察されるが、こと「並列化による効果」と言う観点から見れば必ずしもそうとは言えない部分もある。並列処理は、基本的に複数の PE を利用することにより高速化をねらうものであり、多くの PE が利用可能でなければならない。ユーザ数が多く、その利用が多岐に渡る利用形態においては、シングルジョブや他の大型ジョブとの兼ね合いで、PE そのものが利用できない「待ち」状態であることが多い。これはジョブクラスの運用方針に依存するものであるが、ターンアラウンドタイムからすればシングルジョブの方が、高速化された並列ジョブよりも早い場合もある。トータル的に見て、本当の高速化がなされているかどうか、という点では疑問が残る。また、並列ジョブ実行時にはどうしても PE の利用率の低下（占有しているが実際に動作していない状態）が発生する。計算機の効率的な運用という点からも十分に考慮しなくてはならない問題である。これらの事を見極めた上でのコード選択、高速化が重要ではあるが、根本的な解決のためには、効率的で柔軟な並列ジョブクラスの運用が可能な計算機システムの出現が望まれる。

[参考文献]

(1) UXP/M アナライザ使用手引書(FORTRAN,VP 用)V10L20 用，富士通（株）(1994)．

(2) UXP/M VPP アナライザ使用手引書 V10 用，富士通（株）(1994)．

(3) 鈴木；MCNP4.A の AP3000 への整備について，Computer 情報 No.64，pp.43-47 (1997.6).

(4) 山極，根本，広瀬，ELIA；運動論的バルーニング・シューティングコード KBSHOOT の並列化，JAERI-Data/Code 09-032 (1997.7).

(5) 渡辺，功刀，小瀬；MARS 法による多相流数値解析手法の並列化，原子力学会(平成 9 年度秋) (1997).

(6) 功刀，小瀬，渡辺；MARS 法を用いた Kelvin-Helmholz 不安定起因の破砕波解析，原子力学会(平成 9 年度秋) (1997).

(7) 渡辺，川井，田辺，根本，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植 - 平成 8 年度作業報告(ベクトル化編) -，JAERI-Data/Code 97-051 (1997.12).

(8) 根本，川井，川崎，田辺，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植 - 平成 8 年度作業報告(並列化編) -，JAERI-Data/Code 97-052 (1997.12).

(9) 根本，川崎，田辺，渡辺，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植 - 平成 8 年度作業報告(移植編) -，JAERI-Data/Code 97-055 (1998.1).

(10) 渡辺，佐竹，功刀；円筒座標系直接数値シミュレーションコードの VPP500 向き並列化，第 11 回数値流体力学シンポジウム (1997.12).

(11) 箭竹；MCNP-4B の Paragon（那珂研）への整備，Computer 情報 No.65，情報システム管理課 (1998.6).

(12) 川井，川崎，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植(並列化編) - 平成 9 年度作業報告 -，JAERI-Data/Code 99-020 (1999.3).

(13) 川崎，石附，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植(ベクトル化編) - 平成 9 年度作業報告 -，JAERI-Data/Code 99-026 (1999.5).

(14) 石附，田邊，他；原子力コードの VPP500 におけるベクトル化，並列化及び移植(移植編) - 平成 9 年度作業報告 -，JAERI-Data/Code 99-027 (1999.5).