並列化プログラム自動生成システムの開発

The Development of Automatic Parallelized Program Translator PPGEN（Parallel Program Generator）

社会の様々な分野で重要な役割を担っている科学技術計算では，高速化の手段として並列化が重要な要素である。共有メモリ型計算機でユーザがプログラムの並列実行を実現するには，原始プログラム（ソース）中に必要な並列化の内容を記述する方法，またはコンパイラの自動並列化機能を使用する方法がある。両者には一長一短がある。㈱日立東日本ソリューションズが開発を担当した，㈱日立製作所の製品である「PPGEN（Parallel Program Generator）」は，両者の長所をあわせもち，並列化でのユーザの負担を軽減するというコンセプトでリリースされた。本報告では，PPGENの強力な並列化機能およびその性能について述べる。

川崎　真基　Kawasaki Masanori

1　はじめに

科学技術計算は，気象予報・原子力・金融・学術研究など，社会の様々な分野で重要な役割を担っている。スーパコンピュータをはじめとして，ワークステーションやPCで構成される計算サーバなどの計算機資源を有効に活用し，高速な計算を実現するには効果的な並列実行による演算が不可欠である。

並列化の方式はDMP（Distributed Memory Processing）とSMP（Symmetric Multi Processing）の２つに大別される。前者はグリッド，PCクラスターなどに代表される分散メモリ型の並列化で，後者は主に共有メモリ型計算機上でマルチスレッド・マルチプロセサによる共有メモリ型の並列化である。

共有メモリ型計算機は，現在のスーパコンピュータおよびワークステーションの主流となっており，日立のスーパテクニカルサーバSR8000およびEP8000も共有メモリ型アーキテクチャである。PCでもマルチプロセサの計算機が増えている中，SMP型並列化の重要性は年々高まってきている。

こうした背景のもと㈱日立製作所は，高度な自動並列化解析およびプラットフォーム共通のポータビリティの双方を実現する製品であるPPGEN（Parallel Program Generator）を2003年６月にリリースした。

㈱日立東日本ソリューションズはPPGENの開発を担当した。本報告では，PPGENに実装した強力な並列化機能およびその性能について述べる。

2　並列化指示文と自動並列化

科学技術計算プログラムは一般にFORTRAN言語で記述されるが，共有メモリ型計算機上で並列実行するには，おもに次の２つの方法がある。

①コメント形式の並列化指示文をプログラム中に用いて並列化プログラムを記述する。

②コンパイラがもつ自動並列化機能を用いる。

上記２つの方法のうち，①では各ベンダが独自仕様の並列化指示文をサポートしているが，プラットフォーム

```
C$OMP PARALLEL
C$OMP DO
C$OMP&   PRIVATE(I)
        DO I=1,N
          A(I)=I
        ENDDO
C$OMP ENDDO NOWAIT
C$OMP DO
C$OMP&   PRIVATE(J)
        DO J=1,N
          B(J)=B(J)+J
        ENDDO
C$OMP ENDDO NOWAIT
C$OMP END PARALLEL

    DO ループを並列化
```

図１　OpenMP 指示文による並列化プログラム

共通の並列化指示文のデファクト・スタンダードはOpenMP[1)]である。OpenMPは，OpenMP委員会[2)]により策定された並列化指示文仕様である。OpenMP指示文を用いて並列化プログラムを記述し（図１参照，赤字がOpenMP指示文），OpenMPをサポートしているコンパイラを用いることにより，ベンダ・プラットフォームを意識する必要なくプログラムの並列実行が可能となる。

一方，②の自動並列化機能は，Intel，IBMなどの限られたベンダが提供するコンパイラがサポートしている機能であり，入力ソースの並列性を自動で解析し，並列実行プログラムを生成する。日立がSR8000およびEP8000向けに提供している最適化ＦＯＲＴＲＡＮコンパイラOFORT90（Optimizing FORTRAN90）も自動並列化機能をサポートしている。

表１に上記２つの並列化手法の特徴を示す。

表１　OpenMP 指示文と自動並列化の特徴

項番	項　目	並列化手段	
		OpenMP指示文	自動並列化
1	並列化指定方法	OpenMP指示文をソース中に記述する。	コンパイル時にオプションを指定する。
2	コンパイラが行う並列化	ユーザソースに記述されたOpenMP指示文の内容に従った並列化を行う。	並列化解析により並列化可能箇所を検出し並列化を行う。
3	要求される並列化の知識	指示文を記述するために並列化範囲・同期の位置および変数属性などの並列化に関する情報が必要。	コンパイラが自動で解析を行い各種並列化情報を決定するので基本的には要求されない。
4	並列化ソースのポータビリティ	ほとんどのコンパイラがOpenMPをサポートしており，プラットフォームを意識する必要はない。	自動並列化機能をサポートしているコンパイラは限られており，プラットフォームは限定される。

すなわち，OpenMPで記述した並列化ソースは高いポータビリティを持つ反面，各種並列化に関する指示をユーザが行う必要がある。一方，自動並列化では煩雑な並列化指示をユーザが行う必要はないが，利用可能なプラットフォームは限られている。

2.1　PPGENの位置付け

PPGENは，両者の欠点を解消し，長所をあわせもつという位置付けのもとで開発された製品である。

PPGENはLinux上で動作し，入力されたFORTRANソースに対し並列化解析を行い，OpenMP指示文によって記述された並列化ソースを出力する。PPGENの持つおもな機能は，並列化解析機能，OpenMP並列化表現機能およびソース生成機能である。並列化解析機能でプログラムの並列性を解析し，OpenMP並列化表現機能で，OpenMPを用いて表現可能な並列化表現を構成する（図２）。

並列化解析機能は日立OFORT90の自動並列化機能を使用しており，SR8000およびEP8000で培ったOFORT90の並列化能力を継承している。

3　PPGENによる並列化

PPGENでは，マルチスレッドによりループを並列実行する並列化ソースを出力する。

3.1　PPGENによる並列化の表現

PPGENで並列化表現に使用するOpenMP指示文は下記のとおりである（図３）。

①並列区間

並列区間は，マルチスレッドで実行する区間である。この区間内で各スレッドを制御することにより，ループの負荷分散を行い並列実行させる。並列区間は「並列起動（スレッド生成）」と「並列終了（スレッド終結）」で囲まれ，区間先頭でスレッドを生成する。区間最後でスレッドを終結する。OpenMP指示文では並列区間をOMP PARALLEL指示文とOMP END PARALLEL指示文で囲まれた区間で表現する。

②並列ループ

並列区間内で,各スレッドが処理分散を行い実行する部分である。並列ループに対して,各スレッドは繰返しを分担して並列に実行する。OpenMP指示文では，OMP DOまたはOMP PARALLEL DO指示文で指定されたループが並列ループである。

③逐次実行区間

並列区間内で，単一スレッドのみで実行する区間である。OpenMP指示文では逐次区間をOMP MASTER指示文とOMP END MASTER指示文で囲まれた区間で表現する。

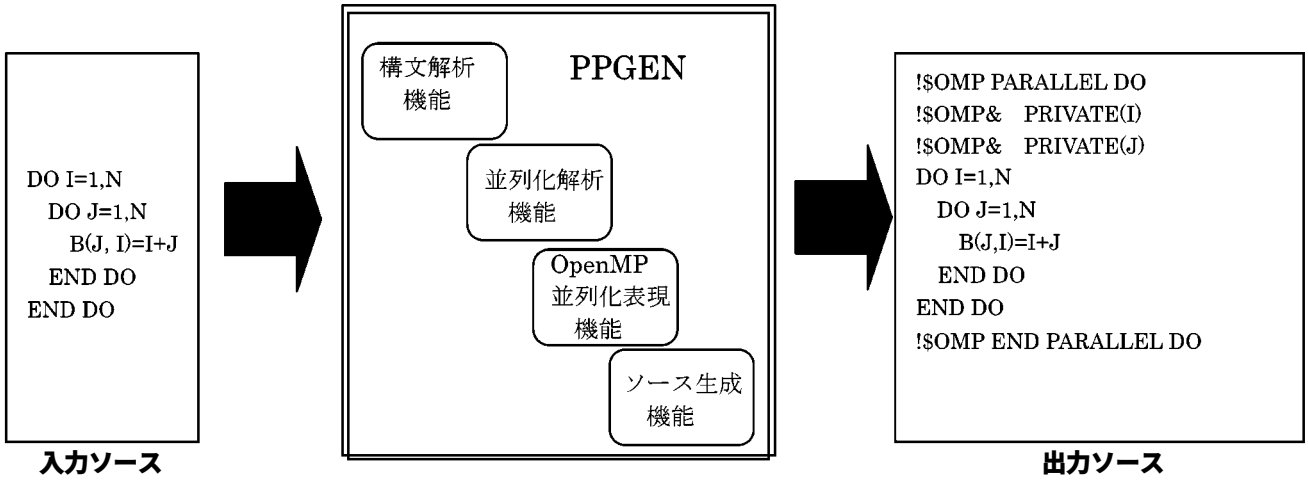

図２　PPGEN の構成と並列化ソース生成

④バリア同期

スレッド間で同期をとることを意味する。OpenMP指示文ではバリア同期をとる位置にOMP BARRIER指示文を記述する。

3.2　変数属性

並列ループ内の変数は，スレッドセーフまたはスレッド共有の属性をもつ。前者はスレッド間で独立した記憶域をもつ変数であり，後者はスレッド間で共有した記憶域をもつ変数である。変数がスレッドセーフの場合，並列ループ先頭/最後で，初期値／終値を設定する場合がある。これら変数属性を指定するためにPPGENが使用するOpenMP指示節（OpenMP指示文に付加して指定する補助的指示文）は表２のとおりである。

表２　並列ループ内の変数属性

項番	変数属性	意　　味	OpenMP指示節
1	スレッドセーフ	スレッド毎独立の変数領域	PRIVATE
2	スレッド共有	スレッド共有の変数領域	SHARED
3	初期値設定付スレッドセーフ	初期値設定+スレッドセーフ	FIRSTPRIVATE
4	終値設定付スレッドセーフ	終値設定＋スレッドセーフ	LASTPRIVATE
5	リダクション	足しこみ演算などのリダクション演算を行うスレッドセーフ変数	REDUCTION

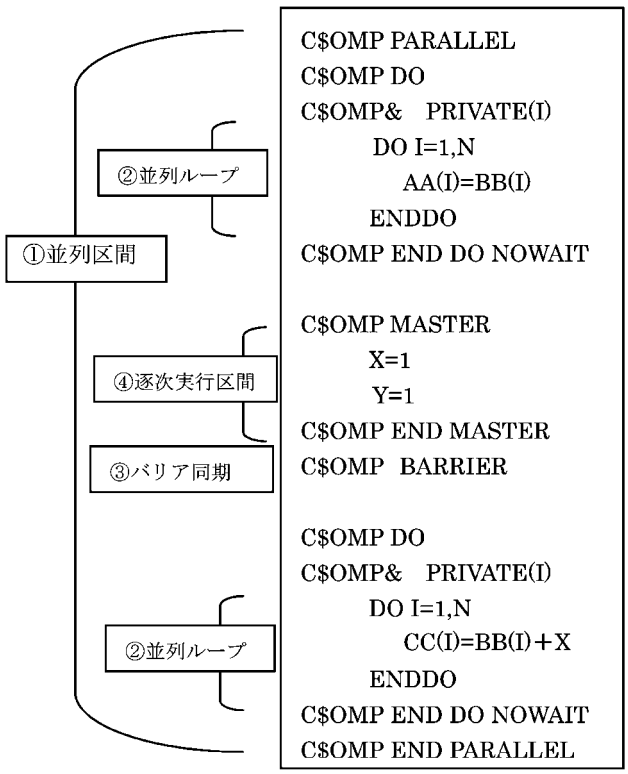

図３　OpenMP 指示文による並列化の表現

3.3　PPGENによる並列化の特徴

効果的な並列化を実現するには，並列ループ数の増加により並列性を上げ，並列実行開始・終了およびスレッド間の同期など，実行にともなうオーバヘッドを削減することが重要である。PPGENではこれらを実現するために，OFORT90から継承した自動並列化能力により次のような解析を行っている。

①並列区間の拡大

並列ループ内外の変数依存を解析し，並列区間を並列ループの外側に拡大して並列起動・終了オーバヘッドを削減する。図４では内側ループは並列化可能だが，外側ループは，配列変数ＡＡにループ繰返しにまたがる依存があり並列化できない。そのため，本来は内側ループの範囲が並列区間であるが（図５），並列区間を内側ループから外側ループに拡大している（図６）。これにより，並列起動・終了オーバヘッドを（外側ループ回数(N)－１回）分削減することができる。図３では二つの並列ループ間に存在する文を逐次実行区間に指定することで，並列区間を拡大している。

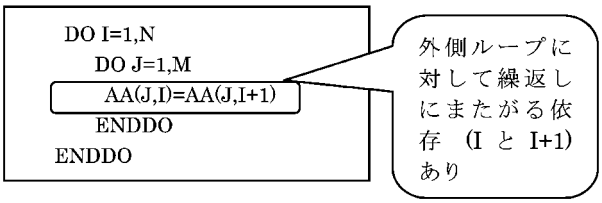

図４　内側が並列化されるループ

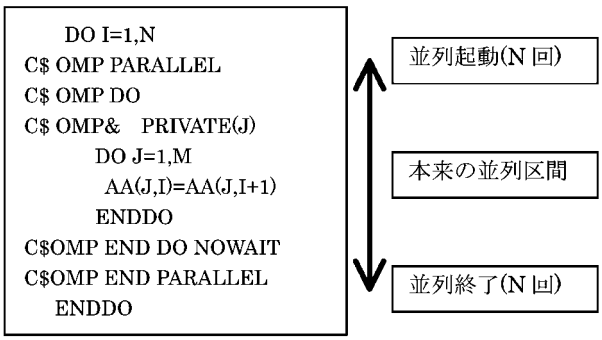

図５　本来の並列区間

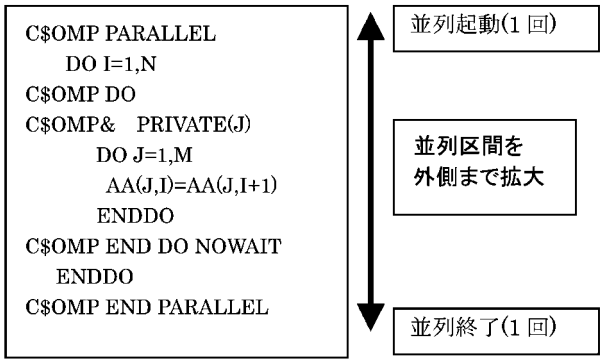

図６　解析による並列区間の拡大

②並列ループ数の増加

ループ内に存在する配列の添字を解析し，ループ繰返しにまたがる依存を詳細にチェックする。図７では次のような配列添字の解析を行う。

［DO 90 ループ］

式（３）の左辺のZ(K)に対して，右辺に繰返し依存Z(K+１)があるので並列化できない。

［DO 85 ループ］

式（２）の左辺XX(I,J,K)に対して，式（１）の右辺に繰返し依存XX(I,J-1,K)，XX(I,J+1,K)などがあるようにみえるが，繰返し幅が４なので，XX(I,J,K)とXX(I,J-1,K)およびXX(I,J,K)とXX(I,J+1,K)などでJのとる値は重ならな

```
      DO 90 K=1,10
        DO 85 J=3+K,1000+K,4
          DO 80 I=3+K,1000+K,2
            Y= XX(I-1,J+1,K)+XX(I,J,K)+  &     ･･･ (1)
                XX(I-1,J+1,K)+XX(I,J+1,K)+  &
               XX(I,J-1,K)+XX(I+1,J,K)*XX(I-1,J,K)
            XX(I,J,K)=Y                        ･･･ (2)
80      CONTINUE
85      CONTINUE
        Z(K)=Z(K)+Z(K+1)+XX(K+3,K+3,K)         ･･･ (3)
90      CONTINUE
```

図７　配列依存ループ（PPGEN 適用前）

```
C$OMP PARALLEL
        DO 90 K=1,10
C$OMP DO
C$OMP&　PRIVATE(J)
C$OMP&　PRIVATE(I)
C$OMP&　PRIVATE(Y)
        DO 85 J=3+K,1000+K,4
          DO 80 I=3+K,1000+K,2
            Y= XX(I-1,J+1,K)+XX(I,J,K)+  &
                XX(I-1,J+1,K)+XX(I,J+1,K)+  &
               XX(I,J-1,K)+XX(I+1,J,K)*XX(I-1,J,K)
            XX(I,J,K)=Y
80      CONTINUE
85      CONTINUE
C$OMP END DO NOWAIT
C$OMP BARRIER
C$OMP MASTER
        Z(K)=Z(K)+Z(K+1)+XX(K+3,K+3,K)
C$OMP END MASTER
C$OMP BARRIER
90    CONTINUE
C$OMP END PARALLEL
```

図８　配列依存ループ（PPGEN 適用後）

い。したがって依存はなく並列化可能である。

［DO 80 ループ］
式（２）の左辺XX(I,J,K)に対して,式（１）の右辺に繰返し依存XX(I-1,J,K)，XX(I+1,J,K)などがあるようにみえるが，繰返し幅が２なので，XX(I,J,K)とXX(I-1,J,K)およびXX(I,J,K)とXX(I+1,J,K)などでIのとる値は重ならない。したがって依存はなく並列化可能である。

この解析結果により，ループ繰返しの依存が並列化を妨げないと判定し，並列化可能な一番外側のループであるDO 85 ループを並列ループに適用している（図８）。

③バリア同期の削減
　並列ループで値を更新する変数が並列ループ後に使用されている場合，変数更新の同期を取るために並列ループ直後の位置にバリア同期が必要である。バリア同期を削減するため，並列ループ前後での変数の使用状況を解析し，不要なバリア同期を出力しないようにする。図９では，二つの並列ループで使用されている配列の依存を解析してバリア同期不要であることを検出し，バリアを削除している。

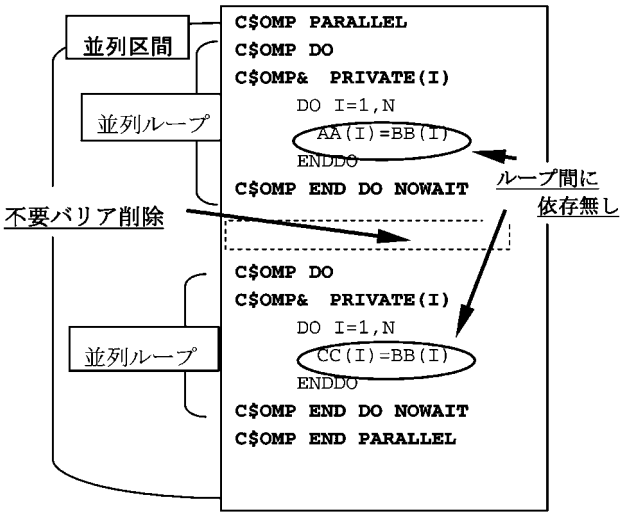

図９　バリア同期削減

④条件付並列化
　並列実行には，並列起動・終結などのオーバヘッドがかかるため,並列ループに適用可能なループであっても，回数が短く，ループ内の演算量が少ない場合は並列化の効果がみこめない。そこで，並列ループにOpenMP指示節のIF指示節を指定して，並列化の効果がある場合のみ並列実行するように指定する。
図10では，ループ内の演算に対し並列化の効果がのぞめるループ回数の場合は並列実行するようにIF指示節によって指定している。

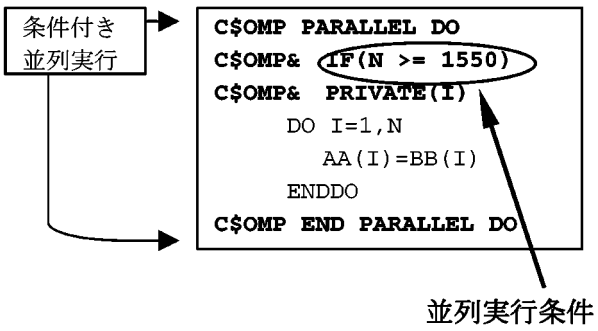

図10　IF 指示節による条件付並列化

⑤スレッド間の負荷分散ループスケジューリング
　多重ループで内側のループ回数が外側ループの制御変数に依存する「三角ループ」を並列化すると，通常は特定のスレッドに負荷が集中する。そこで，スレッド間の負荷を分散し効率的な並列実行を実現するため，三角ループに対して，各スレッドに割当てるループ繰返しの分割単位を小さくするスケジューリングを指示する。OpenMPではSCHEDULE（STATIC,1）指示節をループに指定しする。これにより,特定スレッドに対する負荷の集中を緩和することができる（図11，図12）。

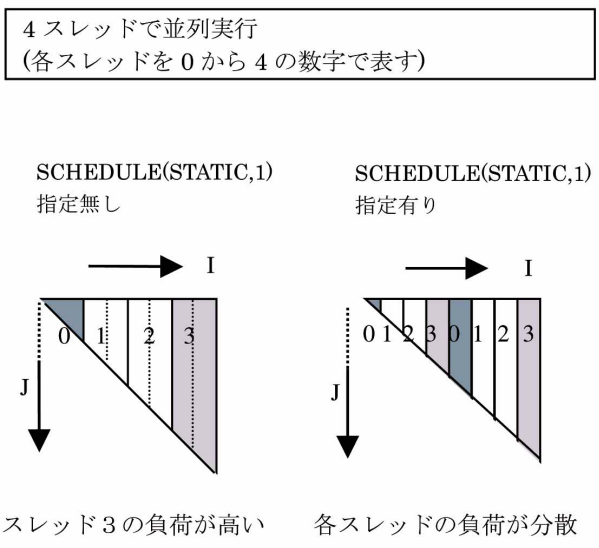

図11　三角ループのスレッド間負荷

```
C$OMP PARALLEL
C$OMP DO
C$OMP&  SCHEDULE(STATIC,1)
C$OMP&  PRIVATE(I)
C$OMP&  PRIVATE(J)
      DO I=1,N
       DO J=1,I
        AA(J,I)=BB(J,I)
       ENDDO
      ENDDO
C$OMP END DO NOWAIT
C$OMP END PARALLEL
```

STATIC,1　スケジューリング指定による負荷分散

内側のループ長が外側ループの制御変数に依存する「三角ループ」

図12　三角ループのスレッド間負荷分散並列化

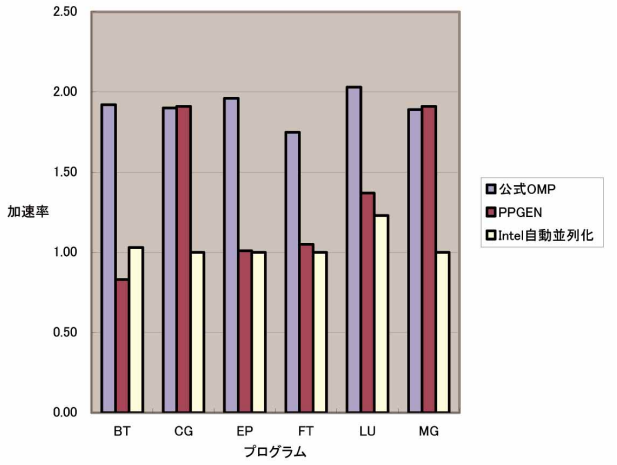

図13　PPGEN の並列化加速性能

4　PPGENの並列化性能

前節で述べたPPGENの強力な並列化機能により，高い並列化性能を実現することができる。NPB3.0をPPGENで変換した並列化ソースを用いて，並列化性能の目安である加速率（=逐次実行時間/並列実行時間）を測定した。

NPB3.0は，並列化性能を測定する代表的ベンチマークである。NPB3.0は，流体力学計算のBT，行列の固有値計算のCGおよびポアソン方程式を数値的に解くMGなど，各分野で実際に行われている数値計算を題材にした７題のプログラムで構成される[4)]。そのうちの６題のプログラムを用いて測定した結果を図13に示す。比較対象は，公式OMP版（公式にOpenMPで並列化されたベンチマーク）およびIntel FOTRANの自動並列化機能により並列化したものである。

［測定環境］
測定マシン：HA8500
（Itanium2 900MHz x 2way）
OS：RedHat7.2
使用コンパイラ：Intel FORTRAN V7.0
コンパイルオプション
①公式OMP：-openmp
②PPGEN出力ソース：-openmp
③Intel自動並列化使用：-parallel

PPGEN出力ソースを用いて並列化した場合，Intel自動並列化に比べて平均29%高い加速率である。また，CGおよびMGは公式版OpenMPソースによる並列化と同等以上の加速率となっている。総合的に，PPGENによる並列化性能が優れていることを示している。

5　今後の展開

PPGENの次期バージョン以降はよりいっそうの並列性能向上のため，つぎのようなサポートおよび機能追加を検討している。

5.1　C言語対応

PC上では，Hyper-Threadingテクノロジ[3)]などマルチスレッドプログラミングによる並列実行に効果的なアーキテクチャをもつハードウエア構成が主流になりつつあり，PC上での主要開発言語の１つであるC言語をもちいた並列化の重要性も高まっている。

OpenMPはFORTRAN上での並列化指示文規格であるが，C言語に対しても規格が存在し，FORTRAN同様並列化指示文のデファクト・スタンダードになっている[5)]。PPGENは今後，C言語ソースをターゲットに拡大する。

5.2　OpenMP指示文の効果的使用による並列化オーバヘッド削減

OpenMP指示文をより効果的に使用することにより，PPGENで出力する並列化ソースに次のような並列化オーバヘッド削減を行う。次のように，リダクション更新演算を並列区間終了時に行うことにより，バリア同期を削減する。

並列ループに対する変数のリダクション更新演算は，並列ループ直後の位置で行っている。リダクション更新演算を行った場合，更新位置直後にバリア同期を出力する。並列ループに対する変数のリダクション更新演算を並列区間終了時にまとめて行うことができるケースで

は，並列区間終了時にリダクション更新演算を行うことで，バリア同期およびリダクションのオーバヘッドを削減することができる。OpenMPではREDUCTION指示節をDOまたはPARALLEL指示文に指定することでリダクションを指示する。DO指示文にREDUCTION指示節を指定すると，並列ループの直後の位置でリダクション更新を行う。PARALLEL指示文にREDUCTION指示節を指定すると，並列区間の最後の位置でリダクション更新を行う。図14ではDO指示文にREDUCTION指示節を指定しているが，図15ではＰＡＲＡＬＬＥＬ指示文にREDUCTION指示節を指定して，リダクション更新を並列区間最後の位置に移動し，リダクション更新を１つおよびバリア同期を２つ削除している。

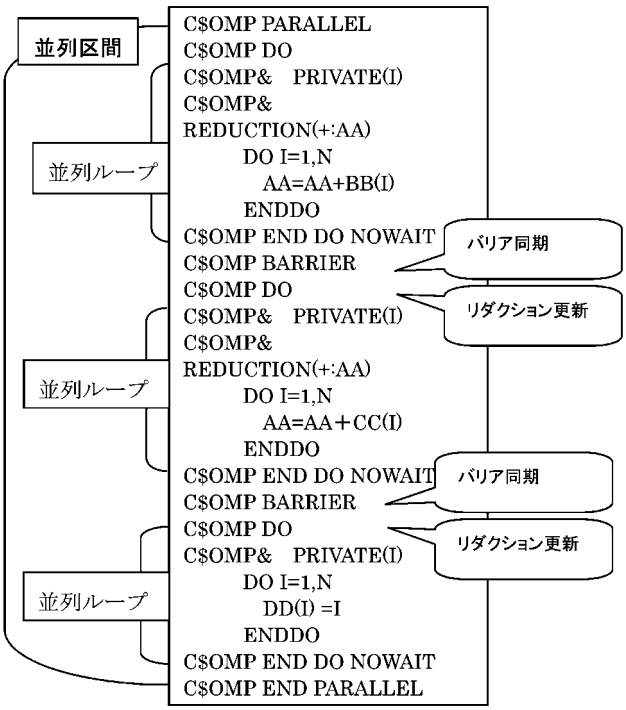

図14　リダクション更新とバリア同期

6　おわりに

2003年６月にリリースされたPPGENは，既にいくつかの大学および研究機関に納入され，ユーザの好評を得ている。今後は，よりいっそうの並列化性能向上が課題である。特に，図13のBTなど，性能測定で十分な並列化性能を発揮できない場合について，詳細に調査し性能改善を行っていく。

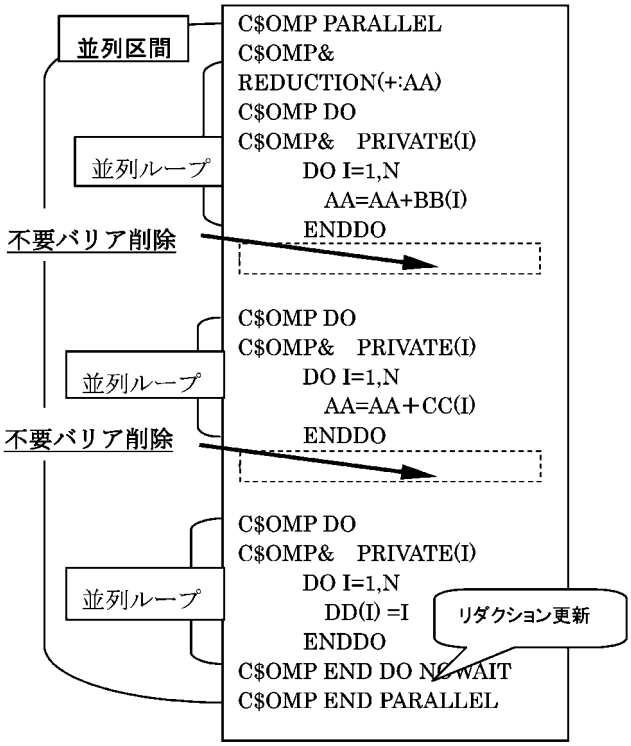

図15　リダクション更新位置移動によるリダクション更新とバリア同期の削除

参考文献

1）OpenMP ARB, "OpenMP Fortran Application Program Interface Version 2.0, November 2000"

2）OpenMP ARB, http://www.openmp.org

3）D.T.Marr,et.al., "Hyper-Threading Technology Architecture And Microarchitecture", Intel Technology Journal Volume 06 Issue 01 Pub. February 14, 2002 ISSN 1535766X, http://www.intel.com/technology/itj/2002/volume06issue01/vol6iss1_hyper_threading_technology.pdf

4）http://www.nas.nasa.gov/Software/NPB

5）OpenMP ARB, "OpenMP C and C++ Application Program Interface Version 2.0, March 2002"

川崎　真基　　1998年入社
金融基本ソフト開発グループ
PPGEN，並列化コンパイラの開発
m-kawa@hitachi-to.co.jp