SX-9の言語処理系

横谷 雄司・工藤 淑裕・田村 正典
早坂 武・林 康晴

要　旨

SX-9では、優れた最適化・ベクトル化・並列化機能を持つFortranコンパイラFORTRAN90/SX及びC/C++コンパイラC++/SX、分散並列処理用のデファクト標準言語であるHPFコンパイラHPF/SX V2、更に分散並列処理プログラム・インタフェースMPI1.2、MPI-2.0仕様に完全準拠したMPI/SX、MPI2/SXを提供しています。本稿ではこれらの言語処理系プログラムにおけるSX-9向けの高速化技術に関する機能や特長について紹介します。

キーワード

●SXシリーズ ●スーパーコンピュータ ●コンパイラ ●MPI ●HPF ●Fortran
●SX-9 ●C ●C++ ●言語 ●共有並列 ●分散並列

1. はじめに

スーパーコンピュータSX-9は、科学技術計算分野、特に数値シミュレーション分野における大規模超高速計算のニーズに応えるための強力なハードウェア機構を提供しています。この強力なハードウェアを使いこなし、高性能を引き出すためには、コンパイラ、実行時ライブラリの能力が非常に重要になります。

SX-9では、旧SXシリーズから定評のある優れた最適化、自動ベクトル化、自動並列化機能を更に進化させたコンパイラFORTRAN90/SXとC++/SX、分散並列プログラム・インタフェースとして、SX-9向けにチューニングを行ったHPF/SX V2、MPI/SX、MPI2/SXを提供しており、ハードウェアの持つ高い性能を容易に引き出すことができます。

2. FORTRAN90/SXとC++/SX

FORTRAN90/SXは、Fortran規格ISO/IEC 1539、JIS X3001:1998に完全準拠し、更に最新のFortran規格（通称Fortran2003）におけるCとの相互運用、IEEE算術及び例外処理などの機能もサポートしたコンパイラです。

C++/SXは、CコンパイラとC++コンパイラの2つの機能を含んだコンパイラ製品です。Cコンパイラは、C言語の国際規格ISO/IEC 9899：1999に準拠しており、規格に新たに追加された複素数型も利用できます。また、C++コンパイラは、C++言語の国際規格ISO/IEC 14882：1998に準拠しており例外処理、実行時型同定機能、標準テンプレートライブラリなどを利用できます。

FORTRAN90/SXとC++/SXともに共有メモリ型並列処理向け標準APIであるOpenMP Version 2.5をサポートしています。

以下では、これら2つのコンパイラのSX-9向けに強化された機能や特長を紹介します。

2.1 分岐を越えた命令並べ換え

SX-9は乗算と加算をそれぞれ2個ずつ、論理演算1個、除算・平方根演算1個の合計6個のベクトルパイプラインを備えており、2つの乗算と2つのベクトル加算、ベクトル論理演算、ベクトル除算（または平方根）の同時実行が可能です。SX-9の性能を引き出すにはこれらのベクトルパイプラインを可能な限り並列動作させることが重要となります。

FORTRAN90/SXとC++/SXは、SX-9のハードウェアの動作をシミュレートし、ハードウェア資源を最大限活用できるように命令列を最適な順番に並べ換えます。その際には、分岐命令をまたいだ広範囲の命令列を対象に命令を並べ換え、より多くの同時実行可能な命令を探し出すことが可能となっています。

例として、**図1** にFortranの配列代入文とそれに対して生成されるベクトル命令列のイメージを示します。

この例では、配列代入文に対し絶対値ABSを求めるための2個のベクトルAND、3個のベクトル乗算、2個のベクトル加算を含むベクトル命令列が生成されています。

コンパイラは、SX-9のベクトルユニットの動作をシミュレートし可能な限り多くのベクトルパイプラインが同時に動作できるようにベクトルレジスタを割り当て命令を並べ換え、**図2** に示すように1つのAND演算と2つの乗算、2つの加算

```
X(:)= 2.0*ABS(A(:))+3.0*ABS(B(:))+ 4.0*C(:)
```

ベクトル命令列のイメージ

```
vand   vt1,0x7fffffff,A
vfmul  vt2,2.0,vt1
vand   vt3,0x7fffffff,B
vfmul  vt4,3.0,vt3
vfadd  vt5,vt2,vt4
vfmul  vt6,4.0,C
vfadd  X,vt5,vt6
```

※ vfmul :ﾍﾞｸﾄﾙ乗算
vfadd :ﾍﾞｸﾄﾙ加算
vand :ﾍﾞｸﾄﾙAND（ABSに対応）

図1 Fortranの配列代入文とベクトル命令列

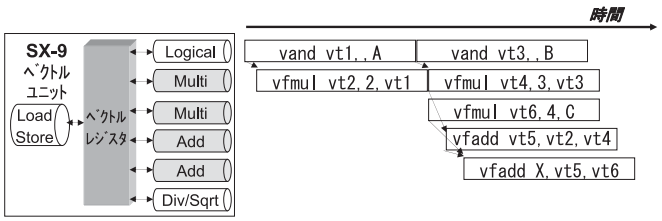

図2 SX-9でのベクトル命令列の動作イメージ

の合計5つのベクトル演算が同時に動作できるような命令列を生成します。

このように、SX-9のコンパイラは、ベクトルパイプラインが並列動作する機会を高められるような命令の並べ換えを広範囲にわたって適用し、SX-9の高い演算性能を最大限まで引き出します。

2.2 ベクトルデータ・バッファリング機能

気象や流体解析などのプログラムの実行では、メモリ上の配列に高速にアクセスすることが要求されます。このようなプログラムに対応するためには、メモリとレジスタの間にメモリ・キャッシュを設けるのが一般的ですが、HWはすべてのデータをキャッシュに載せようとするため、配列が巨大である場合や他のデータが大きい場合、その容量が足りず、本当に必要な配列がキャッシュから追い出されるなど、キャッシュの効果を発揮できませんでした。

SX-9では、この状況に対応するため、ADB (Assignable Data Buffer)と呼ばれるハードウェア機構が、メモリとベクトルレジスタの間に追加されました。ADBには、コンパイラのベクトルデータ・バッファリング機能により選択的にデータをバッファリングできます。例えば、頻繁にアクセスされる配列のみをADBにバッファリングしておくことで、その配列に長期間にわたって高速にアクセスできるようになります。

ある配列、ポインタの指示先をon_adbコンパイラ指示オプションで指定したとき、それらのベクトルロード、ベクトルストアはADB経由で行われます。そのとき、それらの最初のベクトルロード、またはベクトルストアにおいて、データがADBにバッファリングされます。そして、次にそれらのデータを利用するときには、ADBにバッファリングされたデータがロードされます。ADBからは、メモリからロードするより高速にデータをロードできるため、2回目以降のベクトルロードを高速化できます。

図3 にFortranプログラムでのon_adbコンパイラ指示オプションの指定例を示します。どちらのプログラムの処理も同じです。ベクトルデータ・バッファリング機能も同じように動作します。

図3のプログラムを実行したときのベクトルデータ・バッファリング機能の動作を **図4** を使って説明します。

1) 1番目のループの直前にon_adb(a)と指定されているため、配列aのベクトルロード時に配列aのデータがADBにバッファリングされます。

2) on_adb(c)と指定されているため、配列cのベクトルストア時に配列cのデータがADBにバッファリングされます。

3) 配列aの2度目の参照ではADBにバッファリングされた配列aのデータがベクトルロードされます。

```
      subroutine sub
      common a(4096),b(4096),c(4096)
!cdir on_adb(a)
!cdir on_adb(c)
      do i=1,4096
            = a(i) + ...          1)
        c(i) = ...                2)
            = a(i) * ...          3)
      enddo
      ...
!cdir on_adb(a)
!cdir on_adb(c)
      do i=1,4096
        b(i) = a(i) + c(i)    4)/5)
      enddo
      end subroutine
```

図3 Fortranプログラムでの指定

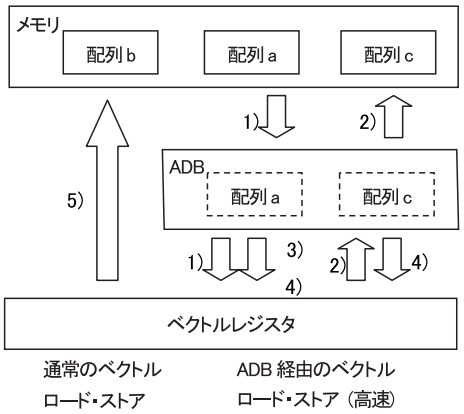

図4　ADBの動作

4) 2番目のループの配列a、配列cの参照では、ADBにバッファリングされた配列a、配列cのデータがベクトルロードされます。

5) 配列bはon_adb指示オプションで指定されていないため、ADBを経由せず、メモリに直接ベクトルストアされます。このとき配列bのデータはADBにバッファリングされません。

ベクトルデータ・バッファリング機能を効果的に使用することで、リストベクトル（間接参照ベクトル）のロードや配列要素のスカラロードの性能も改善できます。

(1) リストベクトルのロード性能改善

リストベクトルとして用いられている配列をon_adbコンパイラ指示オプションで指定しADBにバッファリングしておくと、2回目以降のロードがより高速になります。

図5 は、Fortranプログラムでの例です。この例では、配列a、bを連続ベクトルとしてストアする際に、それらのデータをADBにバッファリングしています。これにより、2番目のループのリストベクトルa(idx(i))、b(idx(i))を参照するときのメモリアクセス性能が改善されます。

C/C++プログラムでも同様の手法で性能改善できます。

(2) 配列要素のスカラロード性能改善

ある配列要素をスカラロードするとき、その配列要素がADBにバッファリングされているのであれば、そのデータはADBからロードされます。例えば、ベクトル化されたループでロード、ストアされた配列の多数の要素をループの直後で使用するとき、その配列をon_adbコンパイラ指示オプションで指定しADBにバッファリングしておくことで、それら配列要素をより高速にロードできます。

図6 は、Cプログラムでの例です。この例では、配列aをベクトルストアする際にそのデータをADBにバッファリングしています。このため、その配列要素をベクトル化不可の2番目のループで参照するときには、ADBにバッファリングされたデータがスカラロードされるので、2番目のループでのa[i]のメモリアクセス性能が改善されます。

Fortranプログラムでも同様の手法で性能改善できます。

このように、ADBにバッファリングするデータを適切に選択し、バッファリングされたデータをできるだけ繰り返し利用するよう心掛けることで、ベクトルデータ・バッファリング機能を効果的に利用でき、メモリアクセス性能を改善できます。将来は、ADBにバッファリングすべきデータ

```
      ...
!cdir on_adb(a)
!cdir on_adb(b)
      do i=1,10000
         a(i) = ... 連続ベクトルとしてストア
         b(i) = ... 連続ベクトルとしてストア
      enddo
      ...
!cdir on_adb(a)
!cdir on_adb(b)
      do i=1,10000
         x(i) = a(idx(i)) + b(idx(i))
         y(i) = a(idx(i)) - b(idx(i))
      enddo
      ...
```

図5　リストベクトルのロード性能改善

```
    ...
   for (j = 0; j < 10000; j++) {
#pragma cdir on_adb(a)
       for (i = 0; i < 256; i++) {
           a[i] += b[j][i];
       }
   }
   ...
   i = 0;
   while (x[i] >= 0) { //    ベクトル化不可
       x[i] = a[i] + ...
       i++;
   }
   ...
```

図6　配列要素のスカラロード性能改善

をコンパイラが自動的に選択できるようにし、より容易にプログラムが高速化されるようコンパイラを強化していきます。

3. MPI/SX、MPI2/SX

MPI/SX及びMPI2/SXは、MPIフォーラムにおいて策定されたメッセージパッシング型プログラミングインタフェース仕様であるMPI-1.2及びMPI-2.0に準拠しています。MPI仕様で定められた機能は、手続き呼び出しのインタフェースで提供され、Fortran、C及びC++言語のプログラムコードから利用することにより、メッセージパッシング型の分散並列処理を行うことができます。

分散並列処理においては、プログラムの並列化に伴い発生するデータ通信などが、並列化処理効率を低下させる要因となることから、高速なデータ転送機能を提供するMPI処理系が重要となります。SX-9においては、ノード間通信用のIXS装置が強化され最大16RCUの構成が可能となっており、ハードウェアの最大スループット性能が大幅に向上しています。以下では、MPI/SX及びMPI2/SXがこの強化されたIXSを活用し実施している高速化について紹介します。

3.1 スループット性能向上

2つのプロセス間でデータ転送を行う1対1通信においては、ある一定サイズ以上の通信を行う場合、転送データをいくつかのブロックに分割し処理します。各ブロックの転送処理は複数のRCU上に最適に分散させ、同時に実行することにより通信を高速化できます。その結果、通信サイズが拡大するほど、ハードウェアピーク性能に応じてスループット性能が向上します（**図7**）。

3.2 集団通信アルゴリズム強化

複数のプロセスが参加してデータ転送を行う集団通信においては、高速化の一手法として、各プロセスがツリー構造を構成するとみなし、そのツリー構造に沿ってデータ転送処理を実施しています。一般的な通信アルゴリズムとしては二分木が知られていますが、SX-9ではマルチRCUを生かし、二分木には限定せずN分木(N≦16)のアルゴリズムを採用しています。N分木の場合、二分木と比較してツリー構造の最大段数（＝通信処理ステップ数）が削減され、通信時間の短縮が可能となります（**図8**）。また、一対一通信に限らず、集団通信においてもデータ転送処理を複数のRCUに最適に分散させ同時に実行することにより、転送サイズが小さなレイテンシ領域から転送サイズが大きなスループット領域に至るまで、ハードウェア特性を十分に活用した高速化を実現しています。

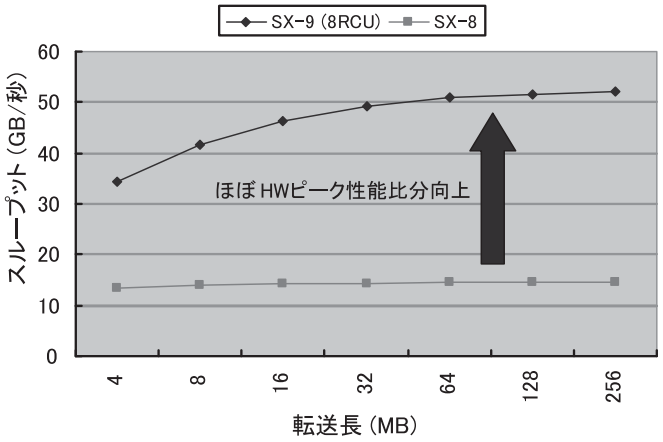

図7　ノード間通信スループット性能比較

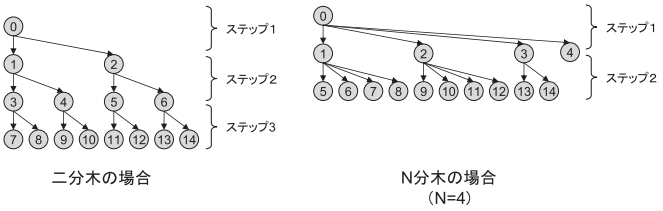

図8　プロセスツリー構造とデータ転送処理（ブロードキャスト通信の例）

4. HPF/SX V2

HPF (High Performance Fortran)は、Fortranの拡張仕様であり、分散並列処理のデファクト標準言語です。

分散並列プログラミングにおいては、データの分散、処理の分割、通信の3要素を考慮する必要がありますが、HPFを使えば、既存のFortranプログラムにコメント形式の指示文を挿入し、データの分散方法を指定するだけで、自動的に処理の分割と通信の生成が行われます。そのため、初心者でも容易

に分散並列プログラムを開発できます。

HPF/SX V2は、HPF2.0及びHPF公認拡張、HPF/JA拡張の主要な仕様をサポートするコンパイラであり、SX-9シリーズに向け、HPFの特長である生産性、保守性を更に向上させる機能を強化しています。以下では、そのうちの主な機能を紹介します。

4.1 チューニング支援機能

(1) 並列化情報リスト

従来の診断メッセージに加え、翻訳時オプションにより、並列化状況、 通信発生状況を知らせる並列化情報リストを生成します。並列化情報リストにより、並列化されたループだけでなく、通信の発生個所や、コンパイラが並列化できないと判断したループや並列化しなかったループを簡単に抽出することができるため、ユーザは、チューニングすべき個所を簡単に特定できます（**図9** 参照）。

(2) 手続境界における通信検出機能

仮引数または実引数に対して分散指定を適切に行わないと、手続呼出し時に通信を伴うコピーが発生し、性能低下を招く場合があります。HPF/SX V2では、実行時オプションにより、手続境界で通信が発生した場合、引数名や手続名などの情報を出力できます。

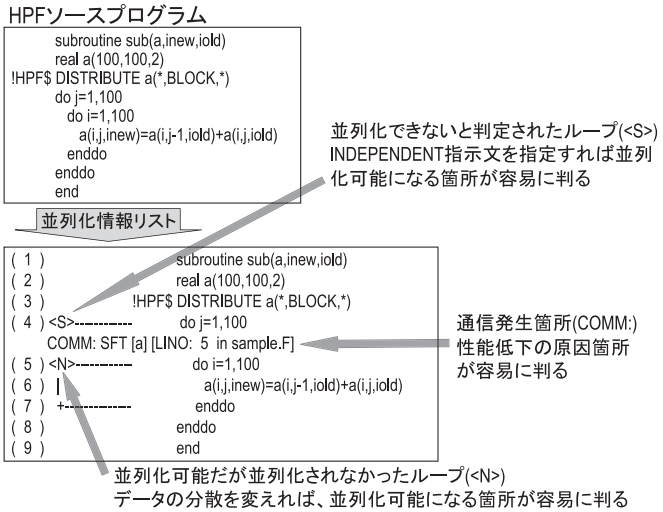

図9　並列化情報リスト

4.2 デバッグ支援機能

(1) 宣言範囲外アクセス検出機能

オプションにより、宣言範囲外アクセスを行っている配列名やプログラムの行番号を出力できます。本機能を利用しても、ベクトル化や並列化は原則として阻害されないため、実用的な実行時間で宣言範囲外アクセスのチェックができます。

(2) 手続間での配列属性の矛盾検出機能

HPFでは手続をまたがって結合する変数―つまり共通ブロック変数と引数―の形状や型等の属性は、原則として一致していなければなりません。オプションにより、実行時にこれらの変数の不整合を検出し、変数名や手続名を出力できます。

5. まとめ

本稿では、SX-9におけるコンパイラ、MPIライブラリについて紹介しました。今後も、SX-9のハードウェア性能をさらに引き出すためのFORTRAN90/SX、C++/SX、HPF/SX V2の最適化及び言語機能の強化、MPI/SX、MPI2/SXの性能チューニングによる高速化などを行っていく所存です。

執筆者プロフィール

横谷 雄司
コンピュータソフトウェア事業本部
第一コンピュータソフトウェア事業部
エキスパート

工藤 淑裕
コンピュータソフトウェア事業本部
第一コンピュータソフトウェア事業部
エキスパート

田村 正典
コンピュータソフトウェア事業本部
第一コンピュータソフトウェア事業部
エキスパート

早坂 武
コンピュータソフトウェア事業本部
第一コンピュータソフトウェア事業部
主任

林 康晴
コンピュータソフトウェア事業本部
第一コンピュータソフトウェア事業部
主任