解 説 HPF言語の動向

3. SP2のためのHPFコンパイラにおける最適化技術

Code Optimization in HPF Compiler for SP2 by Hideaki KOMATSU, Toshio SUGANUMA, Takeshi OGASAWARA, Kazuaki ISHIZAKI and Osamu GODA (IBM Tokyo Research Laboratory).

小松秀昭[1] 菅沼俊夫[1] 小笠原武史[1]
石崎一明[1] 郷田修[1]
1 日本IBM(株)東京基礎研究所ソフトウェアテクノロジー

1. はじめに

我々のHPF処理系[1]はblock-cyclic以外は完全にHPF-subsetに準拠したものである．本システムはFortran 90コンパイラの最適化部の一部を変更することによって作成されている．本システムにおける最適化部の構成は以下のようになっている．

- 静的単一代入化（Static Single Assignment）
- 誘導変数解析（Induction Analysis）
- ループ再構成
- データ依存性解析
- ループ分散（Loop Distribution）
- リダクション最適化
- 通信解析とループ並列化
- メッセージ最適化
- ストレージ局所化
- SPMDコード生成

このフェーズの後に，通常のスカラプロセッサのための最適化が行われる．本システムは既存のプログラムをいかに書き換えることなしで（directiveを付加するだけで），並列化できるかというアプローチをとっている．たとえば，DOループのない，IF文だけのプログラムであっても，静的単一代入化および誘導変数解析によって，ループを再構成し，この再構成したループによって，ループ伝播依存をもとめ，ループを並列化できる．また，この誘導変数による配列の実行アドレス計算はストライド化（update命令）の最適化も施される．

ループ分散は並列化やベクトル化の基本的な技術であり，これなしではほとんどの並列化の効率を落とし，実行性能を著しく低下させる．本システムでは，ループ融合を採用していないが，メッセージ最適化によって，複数のループに関したメッセージは融合させる．

本システムでは，精密なデータ依存解析によって，通常の並列実行だけでなく，DOACROSSループのパイプライン実行もサポートしている．また，実際のアプリケーションを並列実行するために，強力なリダクション検出および並列化の機構を開発している．

本システムの生成するSPMDコードの実行モデルは，バリア同期をいっさい行わない生産者-消費者間のメッセージによるデータフロー実行を行っている．HPFでは，プロセッサの数が実行時に決定されたり，配列やループの範囲が実行時に決定されるため，コンパイラが静的に行うのとまったく同様の，通信解析やメッセージ最適化を実行時に行っている．

2. 最適化技術

2.1 ループ並列化

HPFコンパイラにおけるループ並列化の目的は以下のとおりである．

（1） プログラムの実行時間の多くを占めるループの実行時間を，各プロセッサで並列実行することによって短縮する：分散メモリ並列計算機のプロセッサ台数効果を利用するために必須である．

（2） ループ内で発生する通信を，ループ実行前後で一括化して行う：分散メモリ並列計算機では通信のオーバヘッドが大きいので通信の発生回数を減らし高速に実行する．

本コンパイラのループ並列化フェーズは，上記の目的を満たし，以下の最適化を行う．

（1） 通信一括化とループ並列化
（2） 通信遅延隠蔽を行うループ変形

2.1.1 通信一括化とループ並列化

これまでの並列化コンパイラの研究から，通信一括化において以下の最適化が最低限必要である．

●一括化可能なプリフェッチ通信：DOALL型のループにおいて，あらかじめ必要となるデータを自プロセッサのメモリに取得する．

●一括化可能なパイプライン通信：DOACROSS型のループにおいて，パイプライン実行によって台数効果を得る．

本コンパイラでは，この2つの通信を検出し，同時に並列実行可能なループネストを検出するアルゴリズムを開発した[2]．

まずループのイタレーション空間を分割し通信解析を行い，通信を必要とする配列領域を求める．これからループインデックスの値が変化したときに通信が発生する，ループインデックス空間における通信情報を生成する．この通信情報とループに対するデータ依存解析結果から，我々が通信依存ベクタと呼ぶデータ依存関係によって発生する通信を表す情報を生成する．

この値に基づき，ループネストから一括化可能なプリフェッチ通信と一括化可能なパイプライン通信を検出する．同時に，イタレーション空間の分割可能性と一括化不可能な通信の有無から，並列実行可能なループネストの範囲を決定する．

最終的に，データ依存ベクタと通信解析の情報から，データ依存に基づく通信を行うプロセッサ間でのみ同期を行う通信生成のための情報を出力する．これにより，バリア同期に基づくモデルよりスケーラビリティのよいコードを生成できる．

2.1.2 通信遅延隠蔽を行うループ変形

通信の一括化を行っても，データが到着するまでプロセッサは演算を開始することができない．この場合に，通信時間と計算時間をオーバラップすることによって，通信遅延を隠蔽する最適化が研究されている．我々のHPFコンパイラでは1つのループに着目し通信と計算のオーバラップを行う．この際，通信解析の情報と配列添字から得られる再使用情報を元に，冗長な通信を発生させないタイリングとループ交換方法を求めるアルゴリズムを開発した[3]．

2.2 通信の最適化

本システムはコンパイル時，実行時ともに通信の融合（coalescing），集約化（aggregation）を行うが，ほかにも独自の通信最適化を行っている．ここではこの内からリダクションおよび集合通信について述べる．

2.2.1 リダクション

リダクションは，数値計算プログラムなどにおいて，きわめて頻繁に現れるループパターンであり，これを検出し効率よく実行させることは全体の性能向上からも非常に重要である．とくに分散メモリ並列計算機においては，これを検出できない場合全プロセッサ間で通信しながら計算を行わねばならず，そのペナルティはほかの並列化の効果を打ち消してしまうまでになる．これまでリダクションの検出は，コンパイラがパターンデータベースをもち，プログラムのある特定の形をもつ断片をイディオムとして認識し，これを対応する実行時ライブラリへ変換することで処理されており，一般に複雑なループにおけるリダクションの検出は不可能であった．これに対し本コンパイラでは，解析的な方法により，条件文や代入文が複数ある任意にネストされたループ中に現れるリダクションも検出し並列化することを可能にしている[4]．

リダクションは本質的に，リダクションを構成する一連の実行文の間にデータの依存サイクルがあることに着目する．このデータ依存グラフの強連結成分を検出対象とすることで，ループ本体から実行文を抽出してこれを表現木として内部構成し，次にそれらが実際にリダクションを構成しているかを，この表現木をトラバースすることで決定している．リダクションの認識においては，表現木そのものが交換法則，結合法則の成り立つ演算子で成り立ってリダクションを構成している場合と，表現式とこれをガードする条件式が組み合わされてリダクションを構成しているケースとがある．後者の場合，条件式中の論理表現式と，リダクション関数と呼ばれる副作用のない関数を特定することでリダクションの存在を決定している．

リダクションが検出され，そのループが並列化された場合，各プロセッサにおけるリダクションの中間結果を集計するためリダクション通信が生

成される．このリダクション通信は，通常1回のパケット通信内で1つのスカラーデータが送受信されるのみであり，しかも全プロセッサ間で同期をとらねばならないという，実行性能上のボトルネックとなりやすい，効率の悪い実行形態である．この通信オーバヘッドを軽減させ，プログラム中における同期ポイントを減らすため，複数のリダクションをベクトル化し，異なるプロセッサグループに対するものも含め，1回の通信ですべてのリダクションを行うよう変換する最適化を施している．

2.2.2 集合通信

並列化された数値計算アプリケーションでは，集合通信と呼ばれる決まったパターンの通信がよく行われることが知られている．集合通信は多数のプロセッサが同時に通信する状況の1つである．集合通信に分類される通信の特徴は，プロセッサが行う通信に規則性があることである．代表的な集合通信に，放送，収集，分散などがある．こうした規則性をもった通信に対して，各通信ハードウェアごとにさまざまな最適化が行われている．多くの場合最適化した集合通信ライブラリは，並列計算機によって提供される．本コンパイラは，通信コストを減らすために，プログラムで生じる通信が集合通信のパターンであるかどうかを調べ，できるだけ集合通信ライブラリのルーチンを呼び出すコードを生成する．従来から，静的に分割された配列に対して，コンパイル時にループ実行で必要な通信から集合通信を識別する手法が研究されてきた．しかしこうした手法は，通信する時点でプロセッサ分割が静的に決まってる場合にのみ有効であるが，HPFでは(1)条件つき分岐などで複数の分割方式が指定可能，(2)サブルーチンの実引数が複数の分割方式が可能，であることから，通信が必要なプログラムのある地点に複数の分割方式が到達する可能性がある．このような場合，従来の方式では集合通信ルーチン呼び出しコードを生成できない．我々はこの問題に対し，実行時に集合通信認識を行う方式をとっている[5]．

MPIによる集合通信では，集合通信に参加するプロセッサ群と通信するデータのアクセス規則性を指定する．我々は，配列に $a*i+b$ のように線形にアクセスする規則計算に対して，送信/受信プロセッサの集合的表現，通信するデータアクセスのパラメタ表現を実行時に高速に構成できるようにし，それらをMPIがサポートする集合通信にマップした．ループ実行時に生じるプリフェッチだけでなく，サブルーチン境界などで生じる配列再分割でも活用している．集合通信を含めこうした実行時に計算される通信とその最適化は，結果を再利用可能であればキャッシュする[6]．そのため，実行時計算が必要でもその後不変なことが検出されれば再計算されず，HPFに特有な実行時オーバヘッドを削減している．

2.3 記憶域関連の最適化

本処理系では，分割された配列に対して，各プロセッサの持ち分のみを主記憶上に割りあてる．ただし，block分割された配列次元に対しては，オーバラップ領域を割りあてる．この方法では，実行中必要となるほかのプロセッサからのデータをどこに置くかという問題があるが，これに関してバッファの融合とオペランド参照の動的切替を行っている．

バッファ領域の融合は，ループ中の代入文で，同一配列に対する複数オペランドの参照領域が重複する場合に行う．重複領域の占める割合が大きい場合，これらのオペランドに別々のバッファを割りあてるのは無駄である．本処理系では，これらのバッファ領域の融合を試みている．バッファの融合の決定はコンパイル時および実行時の両方で行う．コンパイル時に静的解析によって，バッファが融合可能と判断されれた場合，コンパイラは複数のオペランド参照に対して，融合されたバッファを参照するコードを生成する．また，可能ならばこれらのオペランドに対する通信を融合する．バッファが融合可能かどうかの判断がコンパイル時に決定できない場合は，コンパイラは実行時に融合可能性を判定し，これに基づいて融合したバッファまたは独立したバッファを動的に選択して使用するコードを生成する．

オペランド領域の動的切替は，他プロセッサからのデータをできるだけオーバラップ領域に収容するための最適化である．一般に，バッファを使用する場合でも，大部分のデータはそのプロセッサが所有している場合が多く，これらのデータを参照される前にバッファにコピーする必要がある．これは大きなペナルティとなるため，できる

限りオーバラップ領域を使用することが好ましい．このため本処理系では，静的解析によってオーバラップ領域に収容可能かどうかを判断できない場合でも，実行時に判断を行い，可能ならばオーバラップ領域に対する参照に変更するコードを生成する．

3. 評　　価

処理系の評価を，分散メモリ並列計算機SP 2を使って行った．使用プログラムにはApplied Parallel Reserch 社の作成したHPFベンチマークからtomcatv，shallow 77，grid，x 42を選んだ．配列の分割はgridが（block，block，*），そのほかは（*，block）である．図-1は本処理系でコンパイルした場合の実行時間（Tp）と，同じプログラムを逐次コンパイラでコンパイルした場合の実行時間（Ts）を比較したもので，横軸にプロセッサ数(p)，縦軸にスピードアップ（Sp＝Tp/Ts）を示した．また，図-2はshallow 77について，同じプログラムをメッセージ通信ライブラリを使った手書きプログラムと比較したものである．

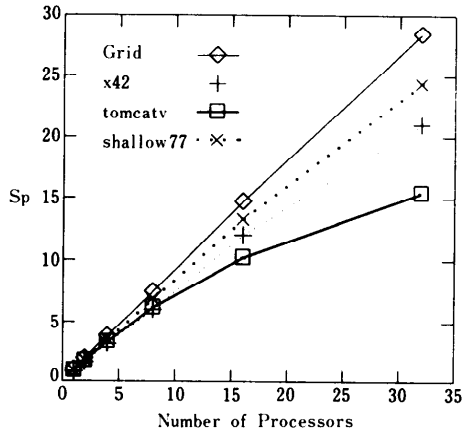

図-1　逐次プログラムとの比較

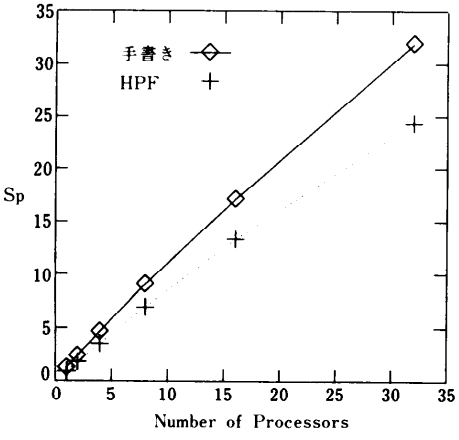

図-2　手書きプログラムとの比較

4. おわりに

SP 2のためのHPF処理系について，主に本処理系固有の最適化について述べた．ベンチマークによる評価結果は並列化および最適化の有効性を示している．

参考文献

1) 郷田，大澤，小松，菅沼，小笠原，石崎，中谷：HPF処理系の実現と評価，情報処理学会研究報告，95-HPC-57，pp. 115-120 (1995).
2) Ishizaki, K. and Komatsu, H.: A Loop Parallelization Algorithm for HPF Compilers, in Proc. of 8th Workshop on Languages and Compilers for Parallel Computing, LNCS 1033, pp. 176-190, Springer (1995).
3) 石崎，小松：分散メモリ並列計算機のためのコンパイラによる通信遅延隠蔽アルゴリズム，並列処理シンポジウムJSPP'96，pp. 339-346 (1996).
4) Suganuma, T., Komatsu, H. and Nakatani, T.: Detection and Global Optimization of Reduction Operations for Distributed Parallel Machines, in Proc. of ACM SIGARCH Internaltional Conference on Supercomputing, pp. 18-25 (1996).
5) 小笠原，小松：HPFにおける集合通信の実行時認識手法，並列処理シンポジウムJSPP '96，pp. 323-330 (1996).
6) 小笠原，小松：HPFにおける実行時の通信解析オーバヘッドの削減手法，情報処理学会研究報告97-HPC-57，pp. 109-114 (1995).

（平成8年9月26日受付）

小松　秀昭（正会員）

1985年早稲田大学大学院理工学研究科電気工学専攻修了．同年日本IBM(株)東京基礎研究所入社．コンパイラ，アーキテクチャの研究に従事．

菅沼 俊夫

1982年京都大学大学院数理工学専攻修士課程修了．1992年ハーバード大学計算機学科修士課程卒業．日本IBM(株)入社以来，東京基礎研究所において，HPFコンパイラの研究に従事．

小笠原武史

1991年東京大学大学院理学系研究科情報科学専攻修了．同年日本IBM(株)東京基礎研究所入社．主にHPFなどのコンパイラの研究に従事．

石崎 一明

1992年早稲田大学大学院理工学研究科修士課程修了．同年日本IBM(株)入社．以来，東京基礎研究所において，HPFなどのコンパイラに関する研究に従事．

郷田 修

1978年電気通信大学大学院電気通信学研究科修士課程修了．同年三菱総合研究所入社．1984年日本IBM(株)入社．現在，東京基礎研究所にて，主にコンパイラの研究開発に従事．