氏名	イ　サン　ホン 李　相　憲
授与学位	博士（工学）
学位授与年月日	平成16年3月25日
学位授与の根拠法規	学位規則第4条第1項
研究科，専攻の名称	東北大学大学院工学研究科（博士課程）電気・通信工学専攻
学位論文題目	Automatic Code Generation Method for Parallel Processing (並列処理プログラム自動生成法に関する研究)
指導教官	東北大学教授　阿曽　弘具
論文審査委員	主査　東北大学教授　阿曽　弘具　東北大学教授　阿部　健一 東北大学教授　丸岡　章　東北大学助教授　大町　真一郎 (情報科学研究科)

論文内容要旨

Parallel computers that built out of the latest microprocessors are becoming widely used as high-powered computer systems. To utilize parallel computers effectively, the program must be expressed as a parallel code for the target machine. In general, however, most algorithms that used in scientific applications are written as sequential codes, for the portability and simplicity of programs. Thus programs must be translated with respect to the target machine, for the efficient use of parallel computers. The translation of a sequential code to an equivalent and efficient parallel program is a difficult and error-prone work. The translator must identify which portions of a program can be run in parallel, and he must specify how the computation should be distributed and communicated between the processors. Depending on the memory system and the connection of components of the given parallel computer, the translator must do other translations that lighten problems related to the data distribution, the communication cost, the synchronization cost and the locality of data reference. To relieve users from this translator's non-trivial work, a parallelizing compiler can be used to automatically translate sequential codes to parallel programs.

There are different levels of parallelism that can be exploited, such as task-level parallelism, loop-level parallelism and instruction-level parallelism. In this paper, we deal with loop-level parallelization and loop-level optimization. For large scientific applications, exploiting loop-level parallelism can be especially effective, because they make heavy use of loops to operate on large array data structures.

In general, the memory systems of modern computers are organized as a hierarchy in which the

latency of memory accesses extremely increases from one level of the hierarchy to the next level. Thus, to realize high performance computing on the modern computers, it is necessary that data being reused are kept in the fast memory closer to the processors. Tiling, as one of the most important compiler optimization loop transformation, is useful for improving data locality. Tiling is a program transformation that a compiler can use to automatically generate a blocked program.

Tiling is very important loop transformation for parallel processing, but still has some restrictions. To achieve high performance in a parallel computer system, the restriction must be solved. Also, the relationship between tiling, scheduling and mapping is not well understood. This paper deals with these problems to optimize synchronization cost of tiled codes. The main goal of the thesis is to give automatic generation methods to generate blocked parallel codes.

Chapter 1 is introduction, and showed the backgrounds of loop transformations.

In Chapter 2, backgrounds for this thesis are introduced. The condition for parallel processing of tiles is introduced. From the condition, a new tiled code generation method for parallel processing is proposed, called order-preserving code generation method. Since proposed method preserves execution order, if original loop nests have parallelism, the result codes have parallelism as well. It is a sequential program code that is generated by conventional methods and then the code is further transformed for parallel processing. Proposed tiled code generation method is equivalent in a sense to the one that combines the two phases of conventional methods, tiled code generation and parallelization, but the synchronization cost may be reduced. The proposed code generation method is convenient and efficient to the optimization of DOALL parallel processing for minimizing synchronization.

A new algorithm which determines shape of tiles is proposed in Chapter 3. The tiles that are determined by proposed algorithm fulfill the condition introduced in Chapter 2. It means that, the tiled codes which are obtained by proposed methods (the tile shape determining algorithm and order-preserving code generation method) preserve parallelism. When the preserved parallelism is the optimal parallelism for a property, the result code of parallelism preserving tiling may have the optimal parallelism for the property. The example loop program in the paper shows that the synchronization cost by the proposed method is less than the one by the conventional

communication-minimal tiling.

In Chapter 4, a new loop fusion algorithm called perfect loop fusion is proposed. Loop fusion is a loop transformation which is used to merge series of loops into a single loop, for improving data locality and reducing synchronization. Tiling is a locality enhancing program transformation for individual loop nest, and loop fusion is a locality enhancing program transformation between loop nests. Many useful loop transformations are restricted to perfectly nested loops. Tiling is restricted to apply to perfectly nested loops as well. Since loop fusion merges series of loops into a single loop, it can be used as preprocessing of tiling. Fusion is not always legal in the presence of dependences between the loops to be fused. Fusing two parallel loops is legal if the resulting loop produces the same result as the one by the original sequence of loops. Moreover, if the original loop is a parallel loop, the fused loop is restricted as a parallel loop. The result of conventional loop fusion is not always a single loop nest. When, however, the result codes are not a single loop nest, tiling can not be performed to it directly. Perfect loop fusion fulfills following three qualities.

1) The result codes maintain parallelism.

2) Locality of data reference is improved by the transformation.

3) The result code is a single perfectly nested loop nest.

For an example, perfect loop fusion is performed to Jacobi loop program. Jacobi loop program is a typical program that solves a partial differential equation by an explicit method. According to proposed method, the synchronization cost of tiled Jacobi program is not dependent on array sizes. The proposed method is useful and effective for parallel architectures of which each computational element has its own parallel processing ability, and they require some different parallel processing schemes, since for such cases, it is important that the result of transformation preserves parallelism and is a perfectly nested loop nest.

In Chapter 5, a new loop-synthesizing algorithm that is represented as unimodular transformation is proposed. Loop-synthesizing combines two loops into one loop, but it does not preserve parallelism. Presented loop-synthesizing algorithm is simple and the result of our synthesizing algorithm is compact than previously known method.

Chapter 6 is the conclusion and states the main contribution.

論文審査結果の要旨

並列処理プロセッサ利用のためのプログラミングは逐次処理の場合に比べ大きな負担となる。そのため，作成が容易な逐次処理プログラムを並列処理プログラムに自動変換するコンパイラが開発されている。既存の自動変換技術は，階層的メモリの効率的利用，同期コストを低減する並列処理化，複数のループプログラムを一つにする融合処理などの個々の変換技術を組み合わせたもので，変換処理間の整合をとるための付加的な処理を必要としている。これらの技術の中で，階層的メモリの効率的利用についてはメモリ参照を局所化するブロック化－タイリング変換－が有効であるが，その変換では並列性の保存は考慮されてこなかった。著者は，タイリング変換に注目し，付加的な処理を不要とする単一の変換方式で並列性を保存する変換手法について研究するとともに，タイリング変換に基づいた並列処理可能なループプログラムの融合手法を考案し，並列処理プログラムの効率的自動生成法を開発した。本論文はその成果をとりまとめたもので，全編６章よりなる。

第１章は序論で，研究の背景・目的を述べ，ループ変換・タイリング変換に関する基本概念を紹介している。

第２章では，並列処理可能なタイルの条件を導き，逐次処理の実行順序を保存するタイリング手法とそれに基づくコード生成法を提案している。提案手法は，処理ブロックを与えるタイルの実行順序として従来はタイル座標の順序であったものを，タイル配置座標の順序で動作するプログラムに変換するもので，従来必要であったタイリングの後の並列化変換を不要にするものである。また，提案手法により生成されたプログラムの並列実行において，同期コストが従来手法によるものより低減されることを示している。これらは逐次処理の並列化を効率的に実現するもので，重要な成果である。

第３章では，最外ループ制御変数を時間軸として並列実行可能なDOALLループに対して，その並列性を保存するタイリング手法を提案している。この手法では，並列処理可能なタイルの構成法を与え，構成されたタイルに対して実行順序保存タイリング手法を適用している。また，提案手法が，プロセッサ間通信量最小化を目的とした従来のタイリング手法に比べて，大規模問題に対して同期コストの面で有利であることを明らかにしている。これらは実用上有用な成果である。

第４章では，二つのループプログラムの系列を一つにするループ融合法について，並列性を保存し，メモリ参照を局所化し，得られるループネストが完全ネストになる完全ループ融合法を提案している。従来のループ融合では融合後に並列化変換が必要であったことに対して，一度の融合変換で並列処理プログラムを生成するものとなっており，有用な成果である。

第５章では，ループ融合法をループ変換で実現する方法を与えている。この方法は従来の融合法で得られるプログラムに比べて構造が単純な完全ネストループプログラムを生成するもので，その生成の結果得られるものに対して直接的にさらなる並列化変換，タイリング変換を適用することができ，興味深い成果である。

第６章は結論である。

以上要するに本論文は，実行順序保存タイリング手法を提案するとともに，そのもとで並列処理可能なタイルの条件を明らかにし，それを満たすタイル構成法を与え，DOALL並列性を保存するタイリング手法，完全ループ融合法を考案して，メモリ参照の局所化を実現し，同期コストを低減する並列処理プログラムが自動生成できることを示したもので，情報通信工学並びに並列処理理論の発展に寄与するところが少なくない。

よって，本論文は博士（工学）の学位論文として合格と認める。