| | |
|---|---|
| 氏　　名（本籍） | 峰 尾 昌 明（東京都） |
| 学 位 の 種 類 | 博 士（工学） |
| 学 位 授 与 番 号 | 甲 第4号 |
| 学 位 授 与 日 付 | 平成１７年3月２５日 |
| 専　　攻 | システム工学専攻 |
| 学 位 論 文 題 目 | 分散メモリアーキテクチャのためのデータ自動分散機能を有する並列化コンパイラＭＩＲＡＩの開発 |
| 学位論文審査委員 | （主査）教 授　吉 本 富士市<br>（副査）教 授　瀧　　寛 和<br>助教授　齋 藤 彰 一<br>教 授　國 枝 義 敏（立命館大学） |

## 論文内容の要旨

**研究説明**

本研究では，分散メモリ型並列アーキテクチャ(以下，分散メモリアーキテクチャ)のためのデータ自動分散機能を有する自動並列化コンパイラ MIRAI(以下，MIRAI コンパイラ)の依存解析部とコード生成部の開発を主に行った． MIRAI コンパイラは Entry Consistency (EC)モデルに基づいたソフトウェア分散共有メモリシステムに対応した初めての自動並列化コンパイラである．分散メモリアーキテクチャでは各 PE(Processor Element)間の通信コストが大きいため，これまで自動並列化はあきらめられていた．MIRAI コンパイラは分散メモリアーキテクチャと実行時環境である Fagus に対応するために他の自動並列化コンパイラにはない以下の機能を持つ．

(1) データ自動分散
(2) 共有変数と同期変数の関連付け
(3) 一貫性制御関数の挿入

これらの機能を実現するために必須の機能である依存解析部とコード生成部を本研究で新たに開発した．依存解析部では厳密かつ高速な配列データ依存解析手法である **Laputa テスト**を提案し，実用化した．コード生成部では中間表現の情報と Laputa テストの解析結果を用いて上記 3 項目を実現するための **4 種類の最適化**を開発した.性能評価の結果,MIRAI コンパイラはデータの自動分散を的確に行い,Fagus の性能を引き出すことができた．これにより，分散メモリアーキテクチャに対しても自動並列化の有効な例があることを示すことができた．

並列計算機はこれまでに様々な並列アーキテクチャが提案されているが，近年はマルチプロセッサの研究開発が主流となっている．マルチプロセッサは複数の PE をバス，クロスバスイッチなどのネットワークを用いて接続したものである．このマルチプロセッサはさらに 2 つに分類することができる．まず 1 つ目は各 PE から直接読み書き可能な共有メモリがネットワーク上に存在する共有メモリ型並列アーキテクチャである.複数の PE が同等な立場で処理を分担する対称型マルチプロセッサ(SMP: Symmetrical MultiProcessor)もこの共有メモリアーキテクチャの一種である．2 つ目はネットワーク上に共有メモリが存在せず，各 PE のローカルメモリのみ存在する分散メモリアーキテクチャである．分散メモリアーキテクチャのうち，各 PE をパーソナルコンピュータ(PC)やワークステーション(WS)で構成したものが PC/WS クラスタである．近年 PC/WS クラスタに関する研究が盛んに行われている．その理由としてはプロセッサ単体の高性能化や，メモリ動作速度の高速化，ネットワークの広帯域化があげられる．これらの 3 要素の高性能化により，PC/WS クラスタとベクトル型スーパーコンピュータとの差が縮んでいる．

しかし，これらの並列計算機上で動作する並列処理用のプログラムの開発は，従来の逐次処理用のプログラムと比較して一般に難しい．並列処理用のプログラムを開発するには，各並列計算機に用意された制御用の専用命令を利用し，各並列計算機のアーキテクチャの特徴を熟知した上で，その特徴を有効に利用できるようにプログラムを作る必要がある．そのため，並列処理の専門家が必要となる．ところが，実際に並列処理を必要としているユーザの多くは，並列処理の専門家ではない．その結果，並列計算機を有効に活用できる応用プログラムの開発，ひいては同計算機の有効な利用が困難になっており，逐次処理用のプログラムを自動的に並列処理用のプログラムに変換する自動並列化コンパイラが必要とされている．

自動並列化コンパイラは商用では，各スーパーコンピュータのメーカーが自社のスーパーコンピュータ用に開発を行っている．研究用ではこれまでにイリノイ大学の Polaris，Parafrase-2，PROMIS，早稲田大学の OSCAR マ

ルチグレイン並列化コンパイラなどの自動並列化コンパイラが開発されているが，そのほとんどが共有メモリアーキテクチャ，特に SMP に対応したものである．分散メモリアーキテクチャに対応した自動並列化コンパイラは知られていない．分散メモリアーキテクチャでは各 PE 間の通信コストが大きいため，事前に「データ分散」を行い，必要なデータを各 PE に割り付けることによる通信の最適化が重要となる．しかし，この「データ分散」をプログラマやコンパイラが行うことは一般に難しく，これまで分散メモリアーキテクチャに対応した自動並列化はあきらめられていた．MIRAI コンパイラは自動並列化だけでなく，この「データ分散」も自動的に行い，幅広い分散メモリアーキテクチャへの対応を目指した自動並列化コンパイラである．

しかし，実行時環境となる様々な分散メモリアーキテクチャの差異にコンパイラだけで対応するのは困難である．そのため，MIRAI コンパイラではこの差異をソフトウェア分散共有メモリシステム Fagus を実行時環境とすることで吸収する．分散共有メモリシステムとは，本来共有メモリを持たない分散メモリアーキテクチャにおいて，共有メモリの一部の複製を各 PE のローカルメモリに配置することで，仮想的に共有メモリを実現するシステムである．しかし，この複製に対して各 PE が個別に書き込みを行うと通常メモリ内容の一貫性が保たれなくなり，処理に矛盾が生じる．したがって，書き込みによって内容が更新された複製間で一貫性を維持する一貫性制御機構が必要となる．分散共有メモリシステムはこの一貫性制御の方式により分類でき，Fagus は EC モデルに基づいている．このモデルでは，共有変数と同期変数を関連付けることによって，1 度に同期を取るメモリの量を必要最小限にし，一貫性制御に必要な通信コストを減少させている．そのため，一貫性制御方式の中でもっとも性能の高いモデルとなっている．しかし，このモデルでは，誤りなく，かつ，最適に「① 共有変数と同期変数の関連付け」，「② 一貫性制御関数の挿入」を行う責務をユーザに課す．これが，このモデルを用いたプログラム作成の大きな障害となっており，これらを自動的に行うコンパイラはない．そこで，本研究では，①と②を MIRAI コンパイラによって自動的に行う方式を提案し，実装・評価した．

具体的には，MIRAI コンパイラの中間表現の変数情報を利用することにより①，②が可能となる．さらに，変数へのアクセス情報から従来のシステムでは予測不可能であった共有変数へのアクセスを事前に予測することが可能となる．これにより，「データ分散」が可能となる．しかし，動的なアクセスを静的な解析のみから完全に予測するのは困難であり，その意味で「データ分散」を完全に自動化するのは難しい．そのため，他のシステムでは「データ分散」は行われていない．MIRAI コンパイラでは，静的に解析できる所は MIRAI コンパイラが主導的にデータ再配置を，さらに必要ならば積極的にデータ転送に関与する．逆に静的に解析できない所については，実行時環境である Fagus にデータ転送をゆだねる．この方式は，筆者が所属するプロジェクトで従来から提案している cc-COMA(compiler controlled Cache Only Memory Architecture)の考え方を具体化したものである．

本研究では主に MIRAI コンパイラの依存解析部とコード生成部の開発を行ったが，コンパイラはパーザ，中間表現，依存解析部，最適化部，コード生成部の一連の流れによって成り立つものである．コンパイラの各部は基本的に中間表現にあわせて実装されており，部分的に切り離すことはできない．そのため，パーザ，中間表現，最適化部の設計や仕様策定にも携わった．新たに開発した依存解析部とコード生成部は MIRAI コンパイラが自動並列化，データ自動分散を行うために必須の機能である．依存解析部は詳細で厳密なデータ参照に関する解析を行い，プログラム中の並列実行可能性を判別する．また，特に配列型の共有変数のアクセスパターンを厳密に解析するためにも，必須となる．コード生成部では分散メモリアーキテクチャと実行時環境である Fagus の特性に対応するために後述の 4 種類の最適化を行う．以下，各部について解説する．

依存解析部では，独自の厳密かつ高速な配列データ依存解析手法である **Laputa テスト**を新たに開発し，実用化した．配列要素間のデータ依存解析手法はこれまで種々提案されており，各手法には解析の厳密性と速度の間にトレードオフが存在する．解析の速度を重視した手法である GCD テストと Banerjee テストでは厳密に依存関係の解析ができず，実際には並列化可能なループを並列化不可と判定する場合がある．厳密性を重視した依存解析手法としては Omega テストがよく知られている．しかし，Omega テストは解析にかかる時間が長く，また実装が困難であるという欠点を持つ．そこで，GCD テストと Banerjee テストと同等の機能を前処理として組み込み，核となるアルゴリズムは一般的なシンプレックス法を用いた新たな依存解析手法である Laputa テストを提案し，実装した．これまでにもシンプレックス法に基づいたものは存在するが，それらは分枝限定法によりシンプレックス法を反復的に行っているため非効率である．それに対し，Laputa テストでは解空間の頂点座標を求めるためにシンプレックス法を用い，その解空間の整数格子点を全探索することで厳密な解析を行う．性能評価の結果，Laputa テストは Omega テストと同等の厳密な解析が行え，ほとんどの場合 Omega テストより高速に解析が可能であった．

コード生成部は統一的中間表現である HTG の情報と Laputa テストによる依存解析結果を用いて，「① 共有変

数と同期変数の関連付け」と「② 一貫性制御関数の挿入」を行う．その際，分散メモリアーキテクチャに対応するために，「通信回数，時間の削減」，「データ分散」，「動的な再分散」を考慮しなければならない．これらに対応するために「オーナーの決定」を提案，実装した．また，既存のループ再構築手法をこれらのために適切に選択し，適用することを提案した．さらに，実行時環境である Fagus の固有の特性に対応するための 3 つの最適化を提案，実装した．以下，「オーナーの決定」と 3 つの最適化について順に解説する．

- オーナーの決定
  各 PE がループの各イタレーションでアクセスする共有変数，または共有変数(配列)の一部を静的に決定する．これにより，一度分散したデータが無駄に再分散することを防ぐ．さらに，オーナーが変化しないため，Fagus の一貫性制御関数を省略でき，PE 間の通信を削減することができる．
- データ量の最適化
  Fagus では OS の仮想記憶にならい，ページ単位でデータを管理するため，共有変数の先頭アドレスがページ境界でなければならない制約がある．そのため，MIRAI コンパイラは共有変数のサイズがページ単位(4KB)の整数倍でない場合は，ページ単位の整数倍になるように変数のサイズを拡張し，割り付ける．

**表 1 ： PE 数 8 の PC クラスタ環境における台数効果**

| | 行列積演算 | LU 分解 | 姫野ベンチマーク |
|---|---|---|---|
| 理想的最高レベルの最適化 | 9.8 倍 | 7.3 倍 | 7.6 倍 |
| MIRAI コンパイラによる自動並列化 | 8 倍 | 6.7 倍 | 7.4 倍 |
| 最適化を行わずに並列化 | 1.7 倍 | | |

- ループ独立依存時の並列化
  基本的に依存のあるループは並列化を行うことはできない．しかし，依存がループ独立依存である場合，Fagus ではそのループを並列実行することが可能である．そのため，Laputa テストは依存がループ独立依存かどうかの細かな判定を行えるように設計開発した．
- 配列の転置
  並列処理部内の多次元配列に対するアクセスの方向が Fagus のためのページ割り付けの方向と異なる場合，実行時にページ転送が頻発し，その通信によるオーバヘッドが急激に増える．共有変数と同期変数の関連付けはプログラム中で変更することができない．そのため，当該ループの直前で配列に対して転置を行うことでアクセス方向をページ割り付けの方向に一致させ，通信によるオーバヘッドを軽減する．MIRAI コンパイラではこの転置のためのオーバヘッドと，軽減される通信オーバヘッドを概算で見積もり，転置すべきか否かを判定させる設計とした．

これらの依存解析部とコード生成部の開発を行ったことにより，MIRAI コンパイラは入力されたソースプログラムを目的プログラムに変換するというコンパイラとしての最低限の動作が行えるようになった．PE 数 8 のクラスタ環境での性能評価の結果(表 1)，MIRAI コンパイラで自動並列化を行った場合，行列積演算では，台数効果が約 8 倍となった．LU 分解に対しては約 6.7 倍，姫野ベンチマークに対しては約 7.4 倍となった．人手による理想的最高レベルの最適化を行った場合，行列積演算に対して約 9.8 倍，LU 分解に対して約 7.3 倍，姫野ベンチマークに対して約 7.6 倍であり，人手による並列化に近い自動並列化が行えていることがわかった．さらに行列積演算，LU 分解に対しては「オーナーの決定」による「データ分散」の効果が見られた．また，本研究で提案した最適化を行わずに並列化を行った場合，行列積演算に対して約 1.7 倍の高速化にとどまり，LU 分解と姫野ベンチマークはプログラムが終了せず，最適化を行わないと性能が出ないことがわかる．これらのことから，MIRAI コンパイラは誤りなく，かつ，最適に「① 共有変数と同期変数の関連付け」，「② 一貫性制御関数の挿入」を行い，Fagus の性能を引き出しているといえる．本評価で用いたテストプログラムは比較的並列化の効果の出やすい例ではあるが，これまで自動並列化があきらめられていた分散メモリアーキテクチャに対しても自動並列化の有効な例があることを示すことができた．これにより，分散メモリアーキテクチャに対する自動並列化の第一歩を踏み出すことができたと考える．

## 論文審査結果の要旨

本論文は、分散メモリ型並列アーキテクチャのためのデータ自動分散機能を有する自動並列化コンパイラ

MIRAI について、データ依存解析部とコード生成部を中心として研究開発した成果をまとめている。MIRAI は、Entry Consistency モデルに基づいたソフトウェア分散共有メモリシステムに対応した、はじめての自動並列化コンパイラである。その主要な特徴は、１）データの自動分散機能、２）共有変数と同期変数の関連付け機能、３）一貫性制御関数の挿入機能、である。また MIRAI は、厳密かつ高速な解析と分散メモリ型並列アーキテクチャ向けの高性能な目的コード生成を可能としている。本論文は、MIRAI の主要部分に関する研究をまとめたものであり、分散メモリ型並列アーキテクチャ上での自動並列化に関する先駆的な研究であると考えられる。

以上の研究成果に鑑み、本論文はシステム工学専攻博士後期課程における博士学位授与のための審査に合格しているものと判断した。

## 最終試験結果の要旨

公聴会（2005 年 2 月 24 日開催）において、本研究の成果について発表させ意見を求めた。その結果、特に問題点の指摘はなく、本研究は博士論文に値するものとして評価された。また、聴講者との質疑応答も適切であり、申請者は博士学位授与に値する学識を有するものと考えられる。

以上に基づき、2005 年 2 月 24 日開催した審査委員会において、本申請は最終審査に合格したものと判断した。