# マルチプラットフォーム・マクロデータフローコンパイラの開発

1L－3

安田 泰勲, 合田 憲人, 岩井 啓輔, 岡本 雅巳, 笠原 博徳
早稲田大学理工学部電気工学科

## 1 はじめに

従来、マルチプロセッサスーパーコンピュータ上でのFORTRANプログラムの並列処理では、マルチタスキングとループ並列化[1][2]が用いられてきた。マルチタスキングではソースプログラム中に記述された並列処理のための拡張命令によってプログラム中の並列性抽出が行われ、タスクの並列実行はOSのタスク生成機能によって実現されている。この場合OSの実行時オーバーヘッドが大きく、さらにユーザがプログラムの並列性を意識して記述しなければならない。また、ループ並列化処理手法は従来より最も広く用いられているが、複雑なデータ依存や条件分岐により並列化できないシーケンシャルループが存在する。このような問題点を解決するため、筆者等はマルチプロセッサシステム上でのプログラムの粗粒度並列処理手法として、マクロデータフロー処理手法[3]を提案してきた。本手法ではコンパイラがプログラムの粗粒度タスク(マクロタスク)への分割、タスク間の並列性抽出、スケジューリングコードの生成などを自動的に行なうことにより効率の良い並列処理が可能である。本稿では筆者等が開発した、複数のマルチプロセッサシステムを対象としたマクロデータフローコンパイラ(以下 マクロデータフローコンパイラ)について述べる。

## 2 マクロデータフロー処理手法

本章では、FORTRANプログラムのマクロデータフロー処理について概説する。

### 2.1 マクロタスク生成

マクロデータフロー処理では、コンパイラがFORTRANプログラムをマクロタスク(MT)と呼ぶ粗粒度タスクに分割する。MTは、基本ブロック(BB)あるいは複数の基本ブロックからなるブロック(BPA)、繰り返しブロック(RB)、サブルーチンブロック(SB)から構成される。

ここで、RBがDo allループである場合、Do allループ内の並列性を抽出するためにDo allループに処理時間とスケジューリングオーバーヘッドを考慮してループ分割を適用する。また、RB, SBでは内部を階層的にマクロタスク(サブマクロタスク)に分割し、サブマクロタスク間で並列性が抽出できる場合は階層型マクロデータフロー処理を適用する。

### 2.2 マクロタスク間並列性抽出

MT生成後、MT間の制御フロー解析・データ依存解析を行ない、これらの情報を表現するマクロフローグラフを生成する。次にマクロフローグラフから、各MTの最早実行可能条件を求め、これを表現するマクロタスクグラフを生成する[3]。

### 2.3 スケジューリングコード生成

マクロタスクグラフの情報をもとに、Dynamic-CP法[3]により実行時にMTをPEに割り当てるスケジューリングコードを生成する。一般的なOSコール等によるダイナミックスケジューリングは実行時のオーバーヘッドが大きいが、本手法ではコンパイラが生成したスケジューリングコードがMTのダイナミックスケジューリングを行なうため、オーバーヘッドを小さく抑えることができる。また、ダイナミックスケジューリング方式として、プログラムの並列性・使用するマシンの同期(排他制御)オーバーヘッド・プロセッサ台数・メモリ構成などを考慮して、スケジューリングコードの実行を単一PE上に集中させる集中スケジューラ方式または全PEに分散させる分散スケジューラ方式のどちらかを用いる[4]。

## 3 コンパイラの構成

本章ではマクロデータフローコンパイラの構成について述べる。

マクロデータフローコンパイラはマルチプラットフォームを実現するため、図1に示すようなFront End, Middle Path, Back Endの3つのステージから構成される。各ステージ間では、FE, MPから生成される中間言語(Intermediate language)によりコード授受が行なわれる。

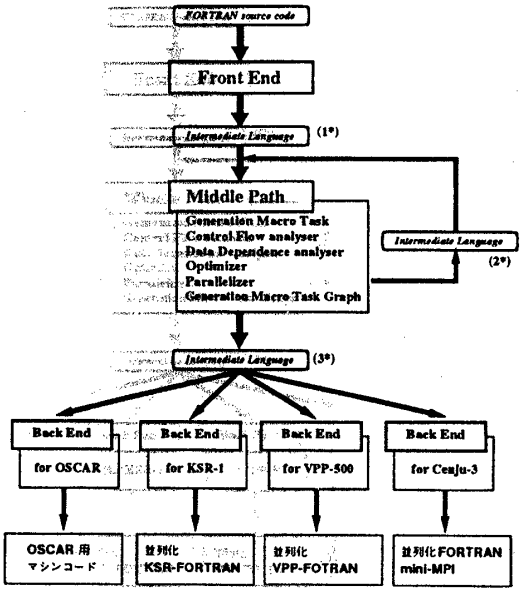


図 1: マクロデータフローコンパイラの構成

### 3.1 中間言語

マクロデータフローコンパイラでは、各ステージ間のコード授受のために以下の3種類の中間言語(Intermediate language)が用いられる。

**plain 中間言語** FEが生成し、プログラム中の変数、定数及び命令コードのみの情報を持つ中間言語。MPへの入力となる。(図1(1*))

---

Developement of Multi-platform Macro-dataflow Compiler.
Yasunori YASUDA, Kento AIDA, Keisuke IWAI, Masami OKAMOTO, Hironori KASAHARA
Dept. of Electrical Engineering, Waseda University

**構造化中間言語** MP 内で plain 中間言語の情報に MT を階層的に生成した結果が付加された中間言語。(図 1(2*))

**並列化中間言語** MP 内で構造化中間言語に MT の最早実行可能条件や PE 間同期コードなどの並列化情報が付加された中間言語。BE への入力となる。(図 1(3*))

### 3.2 Front End

FE(Front End) では FORTRAN ソースコードの字句解析・構文解析を行ない、plain 中間言語を生成する。

### 3.3 Middle Path

MP(Middle Path) では plain 中間言語を解析することにより、主にプログラムの並列化および最適化が行なわれる。MP 内で行なわれる具体的な処理は、

- マクロタスク生成
- 変数定義・参照解析
- 制御フロー解析
- データ依存解析
- Direction/Distance ベクトルを用いた Do all ループ判定
- マクロタスク処理コスト計算
- マクロフローグラフ生成
- 最早実行可能条件解析
- 粗粒度並列処理用マクロタスクグラフ生成
- 中粒度並列処理用中間語生成
- 近細粒度並列処理用中間語生成

である。

また、粗粒度・中粒度・近細粒度を組み合わせたマルチグレイン並列処理 [5] を行なう場合、ループ並列化、近細粒度タスクの並列化を MP において行なう。

### 3.4 Back End

BE(Back End) は対象とするマルチプロセッサ毎に専用のものが用意され、MP の生成する共通の並列化中間言語を解析することにより、各アーキテクチャ用のスケジューリングコード及びプログラムコードを生成する。

現在、筆者等が開発した主記憶共有型マルチプロセッサ OSCAR [6] 用 BE 及び KSR-1 用 BE が用意されている。

#### 3.4.1 OSCAR 用 BE

OSCAR 用 BE は MP が生成した並列化中間言語を解析し、その情報をもとに OSCAR 用マシンコードで記述されたスケジューリングコードとプログラムコードを生成する。

#### 3.4.2 KSR-1 用 BE

KSR-1 用 BE は MP が生成した並列化中間言語を解析し、その情報をもとに並列化された KSR FORTRAN で記述されたスケジューリングコードとプログラムコードを生成する。

## 4 コンパイラの性能評価

本章では、マクロデータフローコンパイラの性能評価を KSR-1 上で行なった結果について述べる。

### 4.1 KSR-1 のアーキテクチャ

KSR-1 は、32 MB のローカルキャッシュを持つ PE がリング状ネットワーク (ALLCACHE Engine) に接続されており、1PE のピーク性能は 40MIPS, 40MFLOPS である。1 リングあたりに接続できる PE 数は 32PE で、リングを階層的に接続することにより、最大 1088PE まで接続可能である。

各 PE のローカルキャッシュは ALLCACHE Memory System によって全 PE から参照・更新可能であり、論理的には共有メモリとして使用できる。ALLCACHE Memory System の概略図を図 2 に示す [7]。

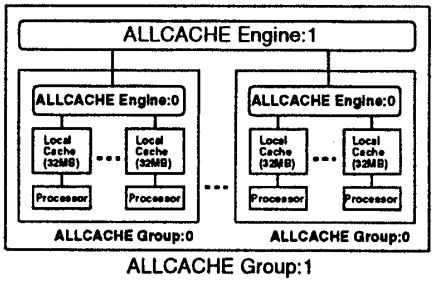


図 2: KSR-1 ALLCACHE Memory System

### 4.2 アプリケーションプログラムによる評価

本章では、NAS Parallel Benchmark[8] の kernel benchmark の一つであるモンテカルロ法を用いた統計処理を行なう EP benchmark 及び 対称帯状マトリクス係数行列を持つ連立方程式の解法である CG(Conjugate Gradient) 法プログラムを用いたマクロデータフローコンパイラの性能評価について述べる。これらのプログラムをマクロデータフローコンパイラによって自動並列化を行ない、性能評価を行なった。

PE 数はマクロタスクのダイナミックスケジューリング方式として、集中スケジューラ方式を用いる場合ではタスクの実行に 16 PE、スケジューリング用に 1 PE の合計 17 PE を用い、分散スケジューラ方式を用いる場合では合計 16 PE を用いた。マクロデータフローコンパイラによる自動並列化を行なったマクロデータフロー処理の結果、分散スケジューラ方式の場合では EP Benchmark はシーケンシャル実行時の 9.921 倍、CG 法はシーケンシャル実行時の 10.815 倍、集中スケジューラ方式の場合では EP Benchmark はシーケンシャル実行時の 9.957 倍、CG 法はシーケンシャル実行時の 11.016 倍という結果が得られた。

## 5 まとめ

本稿では、筆者等が開発したマルチプラットフォーム・マクロデータフローコンパイラについて述べた。筆者等は現在、VPP-500, Cenju-3 用 BE を開発中であり、今後は複数のマルチプロセッサシステム上で本コンパイラの性能評価を続ける予定である。

なお、本研究にあたり KSR-1 を使用させて頂いたキャノンスーパーコンピューティング S.I.(株) の皆様に感謝致します。

## 参考文献

[1] M.D.Guzzi, D.A.Padua, J.P.Hoefinger, D.H.Lawrie : Cedar Fortran and Other Vector and Parallel Fortran Dialects, Proc.Supercomputing '88, pp 114-121, (Mar.1988)

[2] A.H.Karp, R.G.Babb II : A Comparison of 12 Parallel Fortran Dialects, IEEE Software Vol.5 no.5, pp52-67, (Sep.1988)

[3] 本多, 合田, 岡本, 吉田, 尾形, 笠原 : Fortran macro-dataflow compiler, Proceedings of Fourth Workshop on Compilers for Parallel Computers, pp.265-286, 1993.12

[4] 合田, 岩崎, 松本, 岡本, 笠原, 成田:主記憶共有マルチプロセッサシステム上でのマクロデータフロー処理の性能評価, 情処研報 ARC105-9, (1994-3)

[5] 吉田, 岡本, 合田, 尾形, 本多, 笠原 : OSCAR Fortran マルチグレインコンパイラ, 情処研報 PRG-10, (1992-10)

[6] 笠原, 成田, 橋本 : OSCAR のアーキテクチャ, 信学論 (D), J71-D, 8(1988-08)

[7] KENDALL SQUARE RESERCH TECHNICAL SUMMARY, Kendall Square Research Corporation, (1992)

[8] D. Bailey, E. Barszcz, J. Barton, "NAS Parallel Benchmarks RNR Technical Report RNR-94-007", *NASA Ames Research Center*, March. 1994