タスク分割による Cell 用の自動並列化コンパイラの開発

漆尾有史† 桑原寛明†† 國枝義敏††
† 立命館大学大学院理工学研究科 †† 立命館大学情報理工学部

1 はじめに

Cell[1][2] は正式名称を Cell Broadband Engine といい，ソニーコンピュータエンタテインメント，IBM，東芝によって開発されたヘテロジニアスなマルチコアプロセッサである．

図 1 に Cell の構成を示す．Cell は，その特異なアーキテクチャとして，以下の特徴を持つ．メインプロセッサである PPE(PowerPC Processor Element) を 1 つ，サブプロセッサである SPE(Synergistic Processor Element) を 8 つ搭載している．SPE は SIMD(Single Instruction Multiple Data) 型の演算器であるため，スカラー演算を行うと性能を充分に発揮できない．各 SPE は，メインメモリから独立した 256KB のメモリ領域 (LS:Local Store) を持ち，メインメモリには直接アクセスできない．そのため，DMA(Direct Memory Access) 転送を用いてメインメモリと LS 間でデータ転送をする必要がある．DMA 転送には，データのアラインメントとサイズの制限がある．Cell は，複雑なスケジューリング機構を排除することで，高クロック化を実現し，ピーク時の演算性能が非常に高い [3][4]．上記の特性から，Cell でのプログラム開発は，プログラマにかかる負担がとても大きい．

そこで，本研究では Cell プログラムを簡単に作成するための，自動並列化コンパイラを開発する．現在のバージョンでは，プログラムをタスクに分割し，各タスクは SPE で実行し，タスクのスケジューリングを PPE で行う形で並列処理をするコードを生成する．

この前提に基づき Cell 用の自動並列化コンパイラ C.C.(Coins based Cell compiler) を開発し，評価を行った．C.C. の実装には COINS(a COmpiler INfra Structure)[5] を用いた．

2 タスク分割とスケジューリング

2.1 タスク分割

並列化を行うために，プログラムを複数のタスクに分割する．タスク分割は，粗粒度と中粒度の 2 つの粒度で行う．

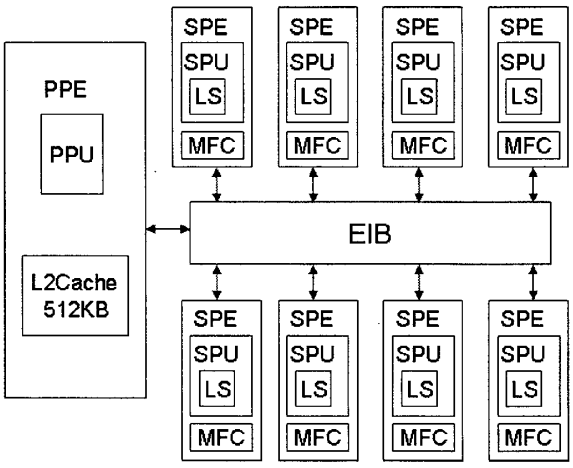

図 1: Cell の構成

粗粒度分割では，基本ブロック毎に分割を行う．また，各タスク毎に依存解析を行うことによって，並列実行の可否判定を行う．

中粒度分割では，ループ並列化を行う．ただし，LS のサイズと，DMA 転送におけるデータのアラインメントとサイズの問題を同時に考慮する．

LS のサイズの上限の問題には，ループ分割数を増やすことによって対処する．

アラインメントとサイズの問題に対処するために，128byte 境界にアラインメントされ，128byte の倍数のサイズになるように各タスクの担当範囲を増やし，前後のタスクと重複するループの回転を担当させる．これにより，無駄な処理が増えることになるが，DMA 転送を可能にするため，敢えて試みることにする．

2.2 スケジューリング

分割したタスクのスケジューリングは PPE が以下のように行う

1. 最初のタスクの実行を SPE に指示
2. 制御依存しているタスクがないタスクを実行待ちキューに追加
3. 実行待ちキューの中から，データ依存しているタスクがないタスクの実行を開始．実行待ちキューが空ならスケジューリング終了
4. タスクが終了すれば，そのタスクが呼び出したタ

Development of an Automatic Parallelizing Compiler for Cell by Task Partitioning
† Yushi Urushio
†† Hiroaki Kuwabara and Yoshitoshi Kunieda
Graduate School of Science and Engineering, Ritsumeikan University (†)
College of Science and Engineering, Ritsumeikan University (††)

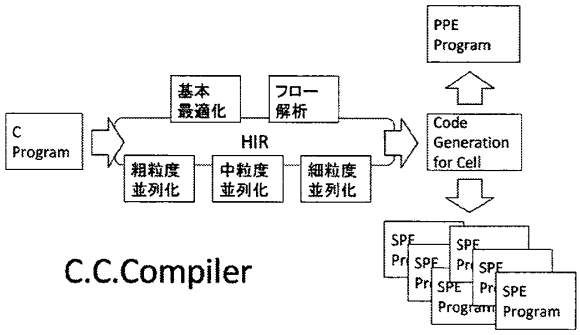

図 2: C.C の概要

スクを実行待ちキューに追加，終了したタスクにデータ依存している全てのタスクの終了したタスクに対する依存フラグを 0 にする．

5. 3 に戻る

3 実装方式

C.C. は上記の並列化手法に基づき，COINS を用いて実装された，Cell 向けの自動並列化コンパイラである．逐次の C 言語のプログラムを入力とし，C 言語の PPE プログラムと SPE プログラムとそれらの Makefile を生成する．C.C. の概要を図 2 に示す．

図中の HIR は COINS の中間表現である．C.C. は HIR 上でフロー解析，依存解析を行い，それらの情報を元に前述の手法を用いて，並列化を行う．その後，PPE プログラムと SPE プログラムを出力する．

C.C. の処理手順は，以下のとおりである．

1. COINS のフロントエンド部を呼び出し HIR 木を作成
2. 関数のインライン展開
3. 粗粒度で分割し，タスクグラフを生成
4. 更に中粒度で分割し，タスクグラフを生成
5. 分割したタスク毎に SPE プログラムを生成
6. スケジューリングを行う PPE プログラムを生成

4 評価

C.C. でコンパイルを行ったプログラムの実行時間に基づいて，提案手法の評価を行う．今回は，次の 2 種類のプログラムで評価を行った．サンプル 1 は，四則演算，シフト演算，剰余演算を 10 万回程度繰り返したプログラムである．サンプル 2 はヒープソートによるソートプログラムである．ただし，ヒープソートの方は LS のサイズの問題のため，大きな数のソートはできていない．プログラムの実行は PS3 で行った．実行結果を表 1 に示す．

表 1: 実行結果

	サンプル 1	ヒープソート
PPE のみ	89sec	9msec
SPE1 基のみ	51sec	11msec
C.C. のコード:6SPE	17sec	18msec
手動並列化	16sec	並列化できず

この表より，サンプル 1 の場合は，PPE の約 5 倍，SPE1 基のみを使用した場合の約 3 倍の速度向上が得られた．手動で並列化した場合とほとんど同じ実行速度が得られていることがわかる．

しかし，ソートプログラムでは C.C. による速度向上は得られなかった．この主な原因は，このソートプログラムは一つの，並列化できないループ文から構成されているためである．さらに，全体の実行時間が短いため，SPE を起動させるためのオーバーヘッドが顕在化している．

5 おわりに

本研究では，Cell の性能を簡単に引き出すための自動並列化コンパイラ C.C. の開発を行った．今後の課題として，ループ分割で対処できない場合の LS のサイズの上限を越えるデータが存在する場合の対応，短すぎるタスクの結合，データ転送の最適化の組み込みなどが挙げられる．

参考文献

[1] Sony Computer Entertainment Inc. Cell broadband engine architecture version 1.01. http://cell.scei.co.jp/pdf/CBE_Architecture_v101_j.pdf.

[2] Dac Pham and et al. The Design and Implementation of a First-Generation CELL Processor. In ISSCC 2005, pp. 184-185, IEEE Press, 2005.

[3] Carsten Benthin, Ingo Wald, Michael Scherbaum, and Heiko Friedrich. Ray Tracing on the CELL Processor. In Interactive Ray Tracing 2006, pp. 15-23, IEEE Press,2006.

[4] Akihiro Asahara, Munehiro Doi, Yumi Mori, Hiroki Nishiyama, and Hiroki Nakano. Cell Broadband Engine Based Realtime Wavelet Decomposition for HDTV Video Images and Beyond. In Multimedia and Expo 2006, pp. 445-448, 2006.

[5] Coins. http://www.coins-project.org/