5L-03

階層統合型粗粒度並列処理のための OpenMP/Task 構文を用いた並列コード生成手法

A Parallel Code Generation Scheme Using OpenMP/Task-Construct for Layer-Unified Coarse Grain Parallel Processing

渡辺 智之 †
Tomoyuki Watanabe

吉田 明正 †‡
Akimasa Yoshida

1 はじめに

マルチコアシステムにおける並列処理手法として，ループ並列性に加えて，粗粒度タスク並列性 [1][2] を最大限に利用する階層統合型粗粒度並列手法 [1][3][4] が提案されている．階層統合型粗粒度並列処理の並列コードは，現在までに Java Fork/Join Framework による実装 [4] や，OpenMP による独自スケジューラを伴う実装 [5] が提案されている．

最近では，OpenMP4.0[6] に OpenMP/Task 構文 depend 節が導入されており，本稿ではそれらを用いて並列コード生成手法を提案する．本並列コードでは，粗粒度タスクの最早実行可能条件 [1] を，depend 節の指示文で指定し，その依存関係に基づいて OpenMP/Task 構文のスケジューラが粗粒度タスクをコアに割り当てる．性能評価では，Intel Xeon サーバ上でヤコビ法プログラムにより提案手法の有効性を示す．

2 階層統合型粗粒度並列処理

階層統合型粗粒度並列処理では，階層型マクロタスクグラフ（MTG）[7] を生成し，粗粒度タスク（マクロタスク，MT）を階層的に定義する．その後，最早実行可能条件 [1] を満たした全階層のマクロタスクを，ダイナミックスケジューラが統一的にコアに割り当てて実行する．例えば，図 1 のような階層型マクロタスクグラフで表されるプログラムを 4 コアで実行したイメージは図 2 のようになる．この場合，複数階層のマクロタスク間の並列性が最大限に利用されていることがわかる．ここで，図 1 の MT8 の最早実行可能条件は MT5∧MT6∧MT7 と求めることができる．これは，MT5 と MT6 と MT7 の実行が終了した後に，MT8 の実行が可能になるということを表している．

3 OpenMP/Task 構文を用いた並列コード

本章では，OpenMP/Task 構文を用いた階層統合型粗粒度並列処理コードについて述べる．

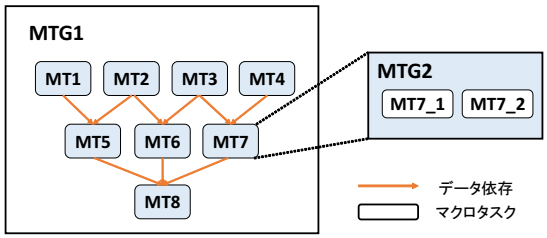

図 1: 階層型マクロタスクグラフ（MTG）.

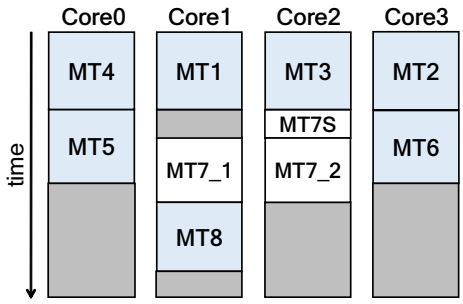

図 2: 4 コア上での階層統合型粗粒度並列処理の実行イメージ.

3.1 OpenMP/Task 構文

タスク並列を利用したプログラミングモデルとして OpenMP がある．OpenMP におけるタスク並列は OpenMP3.0 で実装された task 指示文で利用可能となり，while ループや再帰処理などで用いられる．OpenMP4.0 では task 指示文の節として depend 節が追加され，タスク間の依存関係を記述できるようになった．depend 節は in，out，inout の依存タイプを持ち，in の依存タイプを指定したタスクは，out または inout のどちらかの依存タイプ内のリストを参照する．out または inout のどちらかの依存タイプを指定したタスクは，in，out，inout の依存タイプ内のリストを参照する．本稿では，マクロタスクの最早実行可能条件を depend 節で指定することでマクロタスクは動的にコアへ割り当てられる．

3.2 OpenMP/Task 構文を用いた階層統合型粗粒度並列処理コード

OpenMP/Task 構文を用いて階層統合型粗粒度並列処理コードを生成した．並列コードの構成を図 3 に示す．

図 3 の main() 関数では，まず，2 行目で OpenMP の parallel 指示文により並列領域を指定する．単一のスレッドで粗粒度タスクの実行させるために 4 行目の single 指示文により指定を行う．例えば，図 1 の MT8 の最早実行

† 明治大学大学院先端数理科学研究科
Graduate School of Advanced Mathematical Sciences, Meiji University
‡ 明治大学総合数理学部
School of Interdisciplinary Mathematical Sciences, Meiji University

```
1  int main(){
2  #pragma omp parallel
3    {
4  #pragma omp single
5      {
6        ...
7
8  #pragma omp task depend(out:MT5) depend(in:MT1, MT2)
9        {MT5 の処理;
10       }
11 #pragma omp task depend(out:MT6) depend(in:MT2, MT3)
12       {MT6 の処理;
13       }
14 #pragma omp task depend(out:MT7) depend(in:MT3, MT4)
15       {MT7 の処理;
16 #pragma omp task depend(out:MT7_1)
17        MT7_1 の処理;
18 #pragma omp task depend(out:MT7_2)
19        MT7_2 の処理;
20       }
21 #pragma omp task depend(out:MT8) depend(in:MT5, MT6, MT7)
22       {MT8 の処理;
23       }
24     }
25   }
26   return 0;
27 }
```

図 3: 階層統合型粗粒度並列処理コード.

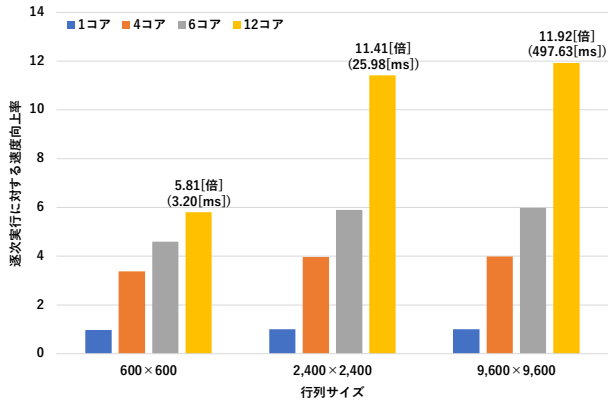

図 4: ヤコビ法プログラムによる性能評価.

可能条件である MT5∧MT6∧MT7 を 21 行目の depend 節で指定する場合，depend 節は depend(out:MT8) depend(in:MT5,MT6,MT7) となり，MT5，MT6，MT7 の処理が終了次第 MT8 が実行される．

4 マルチコア上での OpenMP/Task 構文による並列コードの性能評価

本章では，Intel Xeon サーバにおいて連立 1 次方程式反復解法のヤコビ法プログラムを用いて性能評価を行う．

4.1 性能評価の環境

本性能評価では，Dell PowerEdge R730 サーバで並列実行を行う．本サーバは，Xeon E5-2680（12 コア）の CPU を 2 個，メモリ 64GB を搭載している．OS は CentOS 6.9，処理系は GCC 5.3.1 である．

4.2 ヤコビ法プログラムを用いた性能評価

性能評価プログラムとして連立 1 次方程式反復解法のヤコビ法を用いた．ヤコビ法の反復計算は，収束条件を満たすまで繰り返され，行列サイズは 600 × 600，2,400 × 2,400，9,600 × 9,600 とした．Intel Xeon サーバで実行した結果を図 4 に示す．

まず，行列サイズ 600 × 600 において CPU の 1 コアの実行時間は 18.92[ms]，4 コアの実行時間は 5.46[ms] となり逐次プログラムの実行時間 18.50[ms] と比べて 0.98 倍，3.38 倍の速度向上が得られている．また，6 コアの実行時間は 4.00[ms] で 4.59 倍，12 コアで 3.20[ms] となり 5.81 倍の速度向上が得られている．

行列サイズ 2,400 × 2,400 では，12 コアで 25.98[ms] という実行結果となり逐次実行の 296.55[ms] に対して 11.41 倍の速度向上が得られている．行列サイズ 9,600 × 9,600 では，12 コアで 497.63[ms] となり，逐次の実行結果 5932.26[ms] に対する速度向上率は 11.92 倍となった．それゆえ，提案手法は小さい粒度から大きい粒度のマクロタスクに対して，低いオーバーヘッドでダイナミックスケジューリングが行われることが確かめられた．

5 おわりに

本稿では，階層統合型粗粒度並列処理のための OpenMP/Task 構文を用いた並列コード生成手法を提案した．提案手法では，マクロタスクの最早実行可能条件を depend 節で指定することでマクロタスクはコアへ動的に割り当てられる．

ヤコビ法プログラムにおける性能評価では，Xeon サーバ上で実行した結果，行列サイズ 9,600 × 9,600 において最大 11.92 倍の速度向上率が得られ，提案手法の有効性が確かめられた．

今後の課題としては，提案する並列コードを自動生成する並列化コンパイラの開発が挙げられる．

参考文献

[1] 吉田明正: 粗粒度タスク並列処理のための階層統合型実行制御手法，情報処理学会論文誌，Vol.45，No.12 pp.2732-2740，2004.

[2] 林明宏，和田康孝，渡辺岳志，関口威，間瀬正啓，白子準，木村啓二，笠原博徳: ヘテロジニアスマルチコア向けソフトウェア開発フレームワークおよび API，情報処理学会論文誌，Vol.5，No.1 pp.68-79，2012.

[3] Yoshida, A., Ochi, Y., Yamanouchi, N.: Parallel Java Code Generation for Layer-unified Coarse Grain Task Parallel Processing, IPSJ Transactions on Advanced Computing Systems, Vol.7, No.4 pp.56-66，2014.

[4] Yoshida, A., Kamiyama, A., Oka, H.: A Task-Driven Parallel Code Generation Scheme for Coarse Grain Parallelization on Android Platform, IPSJ Transactions on Advanced Computing Systems, Vol.10, No.1 pp.1-12, 2017.

[5] 渡辺智之，吉田明正: マルチコア/GPU 環境における階層統合型粗粒度タスク並列処理，情報処理学会研究報告，Vol.2018-ARC-232，No.25 pp.1-7，2018.

[6] OpenMP，https://www.openmp.org/，2018.

[7] 笠原博徳，小幡元樹，石坂一久: 共有メモリマルチプロセッサシステム上での粗粒度タスク並列処理，情報処理学会論文誌，Vol.42，No.4，pp.910-920，2001.