1J－3

入出力命令間の依存関係を考慮したプログラムの並列化手法

朝井 義久　　渋沢 進

茨城大学工学部情報工学科

1　はじめに

最近、周辺機器の増加によりプログラム内にも計算だけでなく、周辺機器への入出力も多く含まれるようになってきている。しかし、入出力命令を含むプログラムの並列化の際には入出力命令の前後で同期をとり、入出力命令については逐次で行なうという手法をとっていた。そこで、入出力命令間の依存関係を明確にし、一般のタスクで考えられている依存関係である制御依存やデータ依存 [1] にこの依存関係も含めて並列化を行った。

2　プログラムの表現

プログラム内のタスクをノードで表し、タスク間の制御の流れをノード間のエッジで表す制御フローグラフ (CFG) を使用する (図1)。1つのノードから2つ以上のエッジが出ているものは制御の分岐を表し、それらのエッジのうちの1本だけ通る。図1でタスク番号が白抜きで表現されているものは、そのタスク内に入出力命令を含んでいることを意味する。

3　入出力命令

3.1　入出力命令の定義

入出力命令とは周辺機器 (デバイス) と変数間でデータのやりとり (読み書き) を行う命令である。

3.2　並列化の際の入出力命令の制約

並列化不可能なのは、同一デバイスへのアクセスを同時に行った時であり、異なるデバイスへのアクセスの場合は並列化可能。これより、同一デバイスに対する入出力命令群は逐次と同じ順序で行い、重複は許さないものとする。つまり、入出力命令は直前に実行される同一デバイスの入出力命令が終了してから開始することができる。この依存関係をデバイス依存とする。

A method of program parallelization which considers dependence between Input/Output instructions

Yoshihisa ASAI
Susumu SHIBUSAWA
Faculty of Engineering, Ibaraki University
Hitachi, Ibaraki 316, Japan

4　デバイス依存

4.1　デバイス依存の定義

デバイス依存とは逐次実行と同じ結果を出すための、同一デバイスの入出力命令間の依存関係である。

このデバイス依存が生じる時は、デバイス依存の対象となっているタスクの入出力命令がすべて終了した時に、目的のタスクの入出力命令が行なえる。例えば a、b を入出力命令あるいは入出力命令を含むタスクとした時に、a が終了して初めて b が実行できるというようなときは、b は a にデバイス依存するといい、$a\delta_{dev}b$ で表す。

4.2　デバイス依存グラフの生成

デバイス依存グラフ $DevDG(V_{Dev}, E_{Dev})$ は、以下に基づいて $CFG(V, E)$ から生成することができる (図2)。ただし、V_{Dev}、V をノードの集合、E_{Dev}、E をエッジの集合とする。

- $V_{Dev} = \{x | x \in V, \text{入出力命令} \in x\}$
- x から $y (x, y \in V_{Dev})$ へ到達可能なとき、x から y への CFG 内でのパスを $P = \langle x, \cdots, y \rangle$ とすると、

$$E_{Dev} = \{(x, y) | \forall z \in P, \quad x, y \notin z, \quad z \notin V_{Dev}\}$$

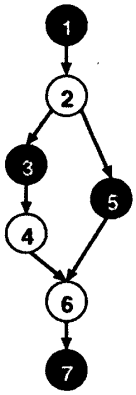

図1.CFG

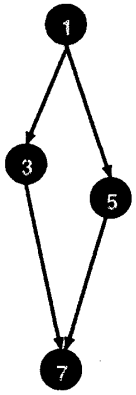

図2.DevDG

4.3　デバイス依存による実行可能条件

実行可能条件を表すための方法として以下の方式を採用する。

- タスク番号

  その番号のタスクが終了した

- (タスク1–タスク2)

  タスク1からタスク2への分岐が決定した

- ∧,∨

  記号の両側の条件の論理積と論理和

デバイス依存グラフより、入出力命令を含むタスクkのデバイス依存による実行可能条件は上の表記法を用いると、

$$\text{タスク}k\text{の実行可能条件} = \bigwedge_{\{y|y\delta_{dev}k\}} \left(y \bigvee_{(a-b)\in Neg(y)} (a-b) \right)$$

となる。ただし、ここに現れる$Neg(y)$はyの実行を迂回するようなCFG内のすべての枝の集合である。

5 一般のタスクの実行開始条件

4節でのデバイス依存に加え、一般のタスクで考えられる制御依存とデータ依存を使用する。

- 制御依存

  あるタスクが実行されることが確定する条件

  形式：$(a-b)\vee(c-d)\vee\cdots$

- データ依存

  変数の参照・更新を正しい順序で行うための条件

  形式：$(e\vee(a-b)\vee(c-d))\wedge(f\vee(g-h))\cdots$

これらの論理積により、タスクの実行開始条件が求められる。

タスクの実行開始条件
= 制御依存∧(デバイス依存∧データ依存)

デバイス依存とデータ依存は同じノードに対する依存が同じ形式であり容易に1つにまとめることができる。このためデータ依存と制御依存だけの実行開始条件と考えることができ、最適化も行いやすい。

6 実行開始条件を用いた動的スケジューリング法

1. 実行開始条件がTrueのタスクを空いているプロセッサに割り当てる。
2. もしあるタスクが終了したか、ある条件分岐が決定したら、まだ割り当てていないタスクすべてについて実行開始条件を調べ、Trueとなっているタスクを空いているプロセッサに割り当てる。
3. 割り当てられているタスクの実行がすべて終了し、これ以上割り当てられるタスクがないとき終了となる。
4. 2. 3. の繰り返し

タスクをプロセッサに割り当てる際に空いているプロセッサがない場合は、キューに貯めておきプロセッサが空き次第キューの先頭から割り当てるようにする。

7 実行例

図1のCFGをデバイス依存を含めた実行開始条件に基づいてスケジューリングした例を図3.に示す。また、デバイス依存を考慮せずに実行開始条件を求め、スケジュール時に入出力命令の前後で同期をとったものを図4.に示す。図3. 図4.で数字がタスク番号で、網掛けの行われているタスクは入出力命令を含むタスクである。

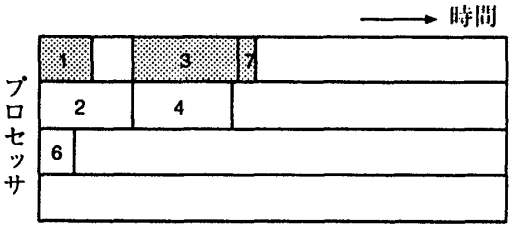

図3. デバイス依存を用いたスケジューリング

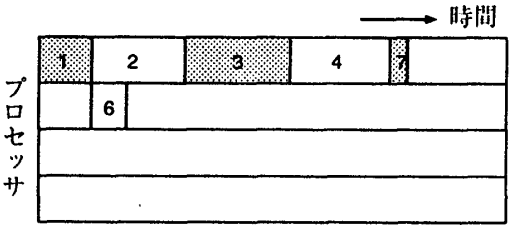

図4. デバイス依存を用いないスケジューリング

8 おわりに

入出力命令間の依存関係を実際にスケジュールを行う以前に求め、タスクの実行開始条件に含めることにより、動的スケジュール時には入出力命令を意識せずに並列化が行えるようになった。またこの方法では同期をとる方法の際には並列化されなかった入出力命令を含むタスクとそうでないタスクの並列化も行えるようになった。

参考文献

[1] M. Girkar and C. D. Polychronopoulos: Automatic Extraction of Functional Parallelism from Ordinary Programs, IEEE Transanctions on Parallel and Distributed Systems, Vol.3, No.2, pp.166-178, March 1992.