

5X-6 並列処理システムにおける同期のためのオーバーヘッドの削減とスケジューリングについて

永松 正博 佐々木 貴康 尾崎 敦夫 有田 五次郎
九州工業大学

1. まえがき

MIMD型の並列処理システムでは、タスクを細分し、多重並列処理を行おうとしても、同期のためのオーバーヘッドが大きくなり、並列処理の効果が上がらなくなる。各プロセッサがFIFOキューを持ち、実行可能となった各タスクは、これらのFIFOキューに投入され、各プロセッサは自分のFIFOキューから先着順にタスクの実行をしていくような並列処理システムにおいては、先行制約に関する同期のオーバーヘッドの削減に関していくつかの結果が知られている¹⁾²⁾³⁾。本論文では、これらの結果を考慮に入れたスケジューリングについて述べる。

2. 並列プログラムとその実行

先行制約を表わす半順序関係 R が与えられたタスクの集合を T とする。他のすべてのタスクに先行するタスク t_i を開始タスクと呼ぶ。有向閉路をもたず、かつ、タスク t_i から他のすべてのタスクに有向道が存在するような有向グラフ $G=(T, E)$ を並列プログラムと呼ぶ。入次数 $d^-(t)$ が2以上であるタスク t を合流タスクと呼ぶ。合流タスクのない並列プログラムを待ちなし並列プログラムと呼ぶ。

次に、並列プログラム G を実行する機構について説明する。タスク t を実行するプロセッサを $pr(t)$ で表わす。 pr をプロセッサ割当て関数と呼ぶ。また、各タスク t に対して、 t から出る枝の間には、順序がついているとする。枝 (t, t') の順位を $f(t, t')$ で表わす。 f を枝順序関数と呼ぶ。

[アルゴリズム FCFS(G, pr, f)]

- 1) 各タスク t に対して $V[t]$ を t の親の個数にする。
- 2) t_i を $pr(t_i)$ のFIFOキューに投入する。
- 3) 各プロセッサは、並列に、以下を実行する。
 - 3.1) FIFOキューが空であるならば、タスクが投入されるまで待つ。
 - 3.2) FIFOキューの先頭からタスク t を取り出し、その実行を開始する(この時刻を t の実行開始時刻と呼ぶ)。
 - 3.3) t の実行が終了する(この時刻を t の実行終了時刻と呼ぶ) と、 t の各子 u に対して、 $f(t, u)$ の小さいものから順に以下の同期操作を実行する。
 - 3.3.1) u が合流タスクである場合、 $V[u]$ から1引き、0になれば、 u を $pr(u)$ のFIFOキューに投入する。
 - 3.3.2) u が合流タスクでない場合、 u を $pr(u)$ のFIFOキ

ューに投入する。

- 3.4) t の処理を終了する(この時刻を t の処理終了時刻と呼ぶ)。

3. FCFSにより誘導される先行関係

$G=(T, E)$ とする。次のように定義される T 上の2項関係 $R_{IPR}(G, pr, f)$ を FCFS(G, pr, f) により誘導される先行関係 (IPR, Induced Precedence Relation) という。

$R_{IPR}(G, pr, f) = \{(t_1, t_2) \mid \text{任意の実行例において、} t_1 \text{ の実行終了時刻よりも} t_2 \text{ の実行開始時刻の方が後である}\} \cup \{(t, t) \mid t \in T\}$

$E_{IPR}(G, pr, f) = \{(t_1, t_2) \mid \text{任意の実行例において、} t_1 \text{ の処理終了時刻よりも} t_2 \text{ の実行開始時刻の方が後である}\}$

$E^* \cup E_{IPR}(G, pr, f) = R_{IPR}(G, pr, f)$ であることが知られている³⁾。FCFS(G, pr, f) において、 E の要素が同期操作に対応する。与えられた先行制約 R を満足するためには、 $R \subseteq E^*$ である必要はなく、 $R \subseteq R_{IPR}(G, pr, f) = E^* \cup E_{IPR}(G, pr, f)$ であればよい。従って、同期操作を減少させるには、 $R \subseteq E^* \cup E_{IPR}(G, pr, f)$ であるような E の中で、同期操作のオーバーヘッドがなるべく小さいものを作ればよい。とくに、合流タスクの個数を少なくする、または、なくす(すなわち、待ちなし並列プログラムに変換する)ことは、同期のためのオーバーヘッドの削減に役立つ。以上のことを行うには、 $E_{IPR}(G, pr, f)$ を計算する必要があるが、この計算方法はまだ知られていない。 $E_{IPR}(G, pr, f)$ のかわりに次のように $D_{IPR}(G, pr, f)$ の定義を行う。

[D_{IPR} の定義]

タスク t_i からタスク t_k への各道 $(x_0=t_i, x_1, x_2, \dots, x_k=t_k)$ ($k \geq 1$) に対して、次の条件を満足する整数 j ($0 \leq j < k$) および、タスク $y_0, y_1, y_2, \dots, y_{k-j}$ が存在するならば $(t_1, t_2) \in D_{IPR}$ である(図1参照)。

- ① $(y_0, y_1), (y_1, y_2), \dots, (y_{k-j-1}, y_{k-j}) \in E^*$
 $(x_j, y_0) \in E, (y_{k-j}, t_2) \in E^*$
- ② $(x_{j+1}, y_1), (x_{j+2}, y_2), \dots, (x_{k-1}, y_{k-j-1}) \in D_{IPR}$
- ③ $pr(x_k) = pr(y_{k-j})$
- ④ (x_j, x_{j+1}) よりも (x_j, y_0) の方が枝順序が後であるか、
 $pr(x_j) = pr(y_0)$ である。

一般に、 $E^* \cup D_{IPR} \subseteq R_{IPR}$ であり、 $G=(T, E)$ が待ちなし並列プログラムである場合は、 $E^* \cup D_{IPR} = R_{IPR}$ であること

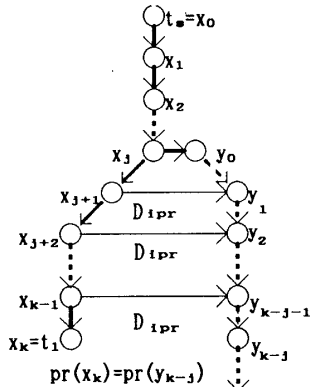
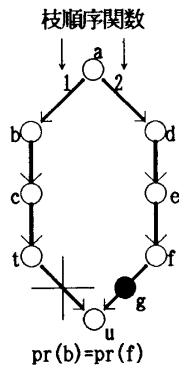
図1 $D_{i,pr}$ の定義のための説明図

図2 制御タスク挿入の説明図

が証明されている³⁾。

4. 待ちなし並列プログラムへの変換と

スケジューリング

タスクの集合 T 、 T 上の先行制約 R 、および、各タスク t の平均実行時間 $w(t)$ が与えられているとする。並列プログラム G の各タスク t に対して、 t を始点とする有向道のうち、各タスクの平均実行時間で重みをつけた重み付長さが最長であるものの、重み付長さを $wlength(t)$ で表わす。

次のような並列プログラム $G=(T \cup C, E)$ 、プロセッサ割当て関数 pr 、枝順序関数 f を作ることを考える。ここで、 C は実行時間が 0 である制御タスクの集合である。

- 1) G は待ちなし並列プログラムである。
- 2) FCFS(G, pr, f) による実行は R に違反しない。
- すなわち、 $R \subseteq (E^* \cup D_{i,pr}(G, pr, f)) \cap (T \times T)$ である。
- 3) T 全体の実行終了時刻をできるだけ早くする。

[アルゴリズム SCHED(T, R, w)]

- 1) E を R の被覆関係 ($E^*=R$ である最小の関係) とし、並列プログラム $G=(T, E)$ を作る。
- 2) 各タスク t に対して、 t_1, t_2 が t の子供で、 $wlength(t_1) < wlength(t_2)$ ならば、 $f(t, t_2) < f(t, t_1)$ であるように枝順序関数を決定する。
- 3) 各タスク t に対して $V[t]$ を t の親の個数にする。
- 4) $pr(t_*)$ を適当に決定し、 t_* を $pr(t_*)$ の FIFO キューに投入する。
- 5) 各プロセッサは、並列に、以下を実行する。
 - 5.1) FIFO キューが空であるならば、タスクが投入されるまで待つ。
 - 5.2) FIFO キューの先頭からタスク t を取り出し、その実行を開始する。
 - 5.3) $w(t)$ の時間の後、 t の実行を終了し、 t の各子 u に対して、 $f(t, u)$ の小さいものから順に以下の操作を実行する。
 - 5.3.1) u が合流タスクである場合、 $V[u]$ から 1 引き、0 になれば、制御タスクの挿入と適当なプロセッサ割当てを行い、 u に入る複数本の枝のうち、1 本だけを残し、他を削除した後、 u を $pr(u)$ の FIFO キューに投入する。枝を削除する原理は、 $R \subseteq E^* \cup D_{i,pr}(G, pr, f)$ であればよいことと、 $D_{i,pr}$ の定義

によっている (図 2 の例では、枝 (f, u) の間に制御タスク g を挿入し、 $pr(g)=pr(c)$ 、 $pr(u)=pr(t)$ にすることにより、枝 (t, u) を削除することができる)。

- 5.3.2) u が合流タスクでない場合、 $pr(u)$ を適当に決定し、 u を $pr(u)$ の FIFO キューに投入する。

- 5.4) t の処理を終了する。

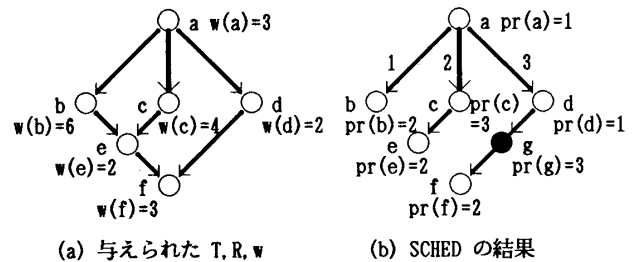
アルゴリズム SCHED において、「 pr を適当に決定して」と記述されている部分は、各タスクの $wlength$ 、および、各プロセッサがかかっているタスクの量に依存して、ヒューリスティックに決定する。なお、十分な台数のプロセッサがある場合、 $R=(E^* \cup D_{i,pr}(G, pr, f)) \cap (T \times T)$ となる。図 3 に SCHED を適用した例を示す。

5. むすび

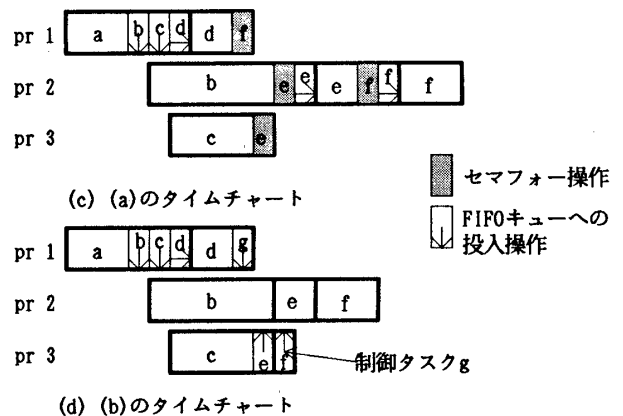
本論文では、タスクの集合と先行制約が与えられたとき、先行制約に関する同期のためのオーバーヘッドの削減を考慮に入れたスケジューリングを行うアルゴリズム SCHED を示した。SCHED はタスクの集合全体の実行終了時刻をできるだけ早くするような、待ちなし並列プログラムを作り出すことができる。

文 献

- 1) 有田: "FIFO キューを同期手段とする並列プログラムについて (I) - 待ちなし並列プログラム -", 情処学論, 24, 2, pp. 221-229, (1983).
- 2) 有田, 荒木: "FIFO キューを同期手段とする並列プログラムについて (II) - 並列プログラムの待ちなし変換 -", 情処学論, 24, 2, pp. 230-237, (1983).
- 3) 永松, 有田: "並列処理システムにおける同期のためのオーバーヘッドの削減について", 信学論 (A), J72-A,

(a) 与えられた T, R, w

(b) SCHED の結果



(d) (b) のタイムチャート

図3 SCHED の適用例