4B－10

重複可能なバリア型同期のための最適バリアスケジューリング

高木浩光　有田隆也　川口喜三男
名古屋工業大学

1. はじめに

並列計算機のための同期方式として，専用ハードウェアを用いた，重複可能なバリア型同期が提案されている[1, 2]．本稿では，タスク集合とタスク集合上の先行制約，プロセッサ台数，タスクのプロセッサ割り当てが与えられたとき，重複可能なバリア型同期以外の同期操作を併用しないでプログラムの正しい実行順序を保証するような，バリア挿入位置を求める同期スケジューリングアルゴリズムを示し，このアルゴリズムが任意の入力に対して最適なバリア挿入位置を出力することを示す．

2. 諸定義

2.1 並列プログラム

並列プログラムを $G=(T, \prec_C, e)$ で表す．T はプログラムが含むタスクの集合であり，$T=\{t_1, t_2, \ldots, t_{|T|}\}$ とする．$\prec$ は T 上の2項関係であり，任意の実行タイミングにおいて，$t \in T$ の実行終了時刻 $\tau_{fin}(t)$，$t' \in T$ の実行開始時刻 $\tau_{st}(t')$ について $\tau_{fin}(t) \leq \tau_{st}(t')$ が成り立つとき，$t \prec t'$ と書く．またこのとき，$\prec$ を順序対の集合とみなして $(t, t') \in \prec$ とも書く．この順序対 (t, t') はタスク t と t' の間の先行制約を表している．タスク集合 T 上で $\prec_C$ の関係が成り立たてばプログラム G の正しい実行が保証される．

e は $e: T \to \{1, 2, 3, \ldots\}$ なる関数であり，$e(t)$ はタスク t の予測処理時間を表す．

同期操作などに要する時間は無視できるものと仮定し，先行制約 $\prec$，予測処理時間 e のもとでタスク t が実行を開始する時刻 $\tau_{st}^{\prec,e}(t)$，及び終了する時刻 $\tau_{fin}^{\prec,e}(t)$ を次のように定義する．

$$\tau_{st}^{\prec,e}(t) = \begin{cases} \max\limits_{t' \in T, t' \prec t} \{\tau_{fin}^{\prec,e}(t')\} & \{t' \in T \mid t' \prec t\} \neq \phi \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_{fin}^{\prec,e}(t) = \tau_{st}^{\prec,e}(t) + e(t)$$

また，プログラム G の処理時間を $\tau_{total}^{\prec,e} = \max\limits_{t \in T} \{\tau_{fin}^{\prec,e}(t)\}$ で表す．

2.2 実行系 M_0

実行系 M_0 は，個々のデータ依存関係に基づいた同期手段を持つ並列計算機のモデルであり，$M_0=(P, S)$ と定義する．P は，実行系が持つ逐次型プロセッサの集合であり，$P=\{p_1, p_2, \ldots, p_{|P|}\}$ とする．$p_1, p_2, \ldots, p_{|P|}$ は能力の等しいプロセッサである．S は，プロセッサのスケジュールであり，$S=\langle s_{p_1}, s_{p_2}, \ldots, s_{p_{|P|}} \rangle$ である．各 s_p は，実行プロセッサとして $p \in P$ が割当てられているタスクの列であり，

$$s_p = \langle t_{i_1}, t_{i_2}, \ldots, t_{i_{|s_p|}} \rangle, i_j \neq i_k (j \neq k)$$

である．$|s_p|$ は，実行プロセッサとして p が割当てられているタスクの総数を表す．また，$s_p = \langle t_{i_1}, t_{i_2}, \ldots, t_{i_{|s_p|}} \rangle$ であるときの t_{i_j} を，単に $s_p[j]$ と書く．

各プロセッサ $p \in P$ は，s_p の要素であるタスクを $t_{i_1}, t_{i_2}, \ldots, t_{i_{|s_p|}}$ の順に逐次的に実行する．したがって，

$$\prec_S = \{(s_p[i], s_p[j]) \mid p \in P, 1 \leq i < j \leq |s_p|\}$$

の先行制約が保証される．

実行系 M_0 は，異なるプロセッサが割り当てられた2つのタスク間の先行制約のそれぞれを，共有変数を用いた同期操作などで保証する．この同期操作により保証される先行制約を $\prec_V$ とすると，このとき実行系 M_0 全体が保証する先行制約の集合 $\prec_{M_0}$ は $\prec_{M_0} = \prec_S + \prec_V$ であり，実行系 M_0 の正しい実行を保証するためには $\prec_V$ を $\prec_V = \prec_C \cap \overline{\prec_S}$ とすれば十分である．

[1] An Optimal Scheduling Algorithm for Ultimate Barrier Synchronization
Hiromitsu Takagi, Takaya Arita, Kimio Kawaguchi
Nagoya Institute of Technology

2.3 重複可能バリア

重複可能なバリア型同期機構を持つ実行系を M_{B_u} とし，$M_{B_u}=(P, S, B_u)$ とする．B_u は挿入されているバリア領域の集合であり，$B_u=\{b_1, b_2, \ldots, b_{|B_u|}\}$ である．バリア $b \in B_u$ は，$b=\langle I_{p_1}, I_{p_2}, \ldots, I_{p_{|P|}} \rangle$ であり，I_p はバリア領域のプロセッサ p における境界位置を表す．また，このときの I_p を単に $b_u[p]$ と書く．I_p はバリア領域入口と出口の位置を示すもので，$I_p = (i_{entr}, i_{exit})$ である．i_{entr} は入口，i_{exit} は出口の位置を示す．このときの i_{entr}, i_{exit} をそれぞれ単に $b[p].entr, b[p].exit$ と書く．$b[p].entr = j$ であることは，タスク $s_p[j]$ と $s_p[j+1]$ との間にバリア領域入口が位置することを意味している．

重複可能バリア同期は，その同期ハードウェアの構造上の特徴から，バリア領域に重複することが許されている．ただし，領域が他の領域の内部に含まれることは許されない．また，同一のバリア領域について入口より先に出口に到達してはならない．したがって，

$$\forall_{p \in P}\, \forall_{i (1 \leq i \leq |B_u|)};\ b_i[p].entr \leq b_i[p].exit$$
$$\forall_{p \in P}\, \forall_{i,j (1 \leq i < j \leq |B_u|, b_i[p].entr \leq b_j[p].entr)};\ b_i[p].exit \leq b_j[p].exit \quad (1)$$

でなければならない．

重複可能バリア同期は，すべてのプロセッサの実行がバリア領域入口を越えるまで，どのプロセッサもそのバリア領域の出口を越えて実行しないという同期方式であるので，バリア領域集合 B_u 全体により保証される先行制約の集合を $\prec_{B_u}$ とすると，

$$\prec_{B_u} = \{(s_p[i], s_{p'}[j]) \mid b \in B_u, p, p' \in P, 1 \leq i \leq b[p].entr,\ b[p'].exit < j \leq |s_{p'}|\}$$

である．そして，実行系 M_{B_u} がプロセッサの逐次実行とバリア同期の組み合わせにより保証する先行制約の集合 $\prec_{M_{B_u}}$ は，$\prec_{M_{B_u}} = \prec_S + \prec_{B_u}$ である．

実行系 M_{B_u} が，重複可能なバリア型同期以外の同期操作を用いないで，与えられたプログラム G を正しく実行するためには，

$$\prec_C \subseteq \prec_{M_{B_u}} \quad (2)$$

でなければならない．また，実行系 M_{B_u} がデッドロックを引き起こさないためには，$\prec_{M_{B_u}}$ が先行関係であれば十分である（先行関係の定義については文献[3]参照）．

3. 最適バリア領域スケジューリングアルゴリズム

$T, \prec_C, P, S$ が与えられたとき，式(1),(2)を満たし，$\prec_{M_{B_u}}$ が先行関係となるような B_u を求めるアルゴリズムを以下に示す．

アルゴリズム1 INSERT-BARRIER-REGION

INSERT-BARRIER-REGION $(T, \prec_C, P, S, e)$ returns B_u is
　$B_u \leftarrow \phi$;
　$\prec_{ne} \leftarrow \prec_C \cap \overline{\{(s_p[i], s_p[j]) \mid p \in P, 1 \leq i < j \leq |s_p|\}}$;
　$\prec_{M_0} \leftarrow \prec_C + \prec_{ne}$;
　$i \leftarrow 0$;
　while $\prec_{ne} \neq \phi$ do ……(1)
　　$i \leftarrow i + 1$;
　　$(t, t') \leftarrow (t, t') \in \prec_{ne}$ where
　　　$\tau_{fin}^{\prec_{M_0},e}(t) = \max\limits_{(t'', t''') \in \prec_{ne}} \{\tau_{fin}^{\prec_{M_0},e}(t'')\}$; ……(2)
　　$b_i \leftarrow \langle (0, |s_{p_1}|), (0, |s_{p_2}|), \ldots, (0, |s_{p_{|P|}}|) \rangle$; ……(3)
　　$b_i[p].entr \leftarrow j$ where $s_p[j] = t$; ……(4)
　　$b_i[p'].exit \leftarrow j'$ where $s_{p'}[j'+1] = t'$; ……(5)
　　$\prec_{ne} \leftarrow \prec_{ne} \cap \overline{\{(t, t')\}}$ ……(6)
　end ;
　$n \leftarrow i$;
　for $i \leftarrow 2$ to n step 1 do ……(7)

```
        b_i[p] ← max{b_i[p], b_{i-1}[p]}
    end ;
    for i ← n − 1 to 1 step −1 do  ..... (8)
        b_i[p] ← min{b_i[p], b_{i+1}[p]}
    end ;
    B_u ← ⟨b_1, b_2, ..., b_n⟩
end
```

$\prec_{ne}$ は，(1) の while ループの各繰り返しでの B_u において，まだ保証されていない先行制約の集合を意味する．各繰り返しごとに，$\prec_{ne}$ からひとつの先行制約 (t,t') が選択され (行 (2))，これをひとつのバリア領域 b_i で保証する．そしてそのバリアによって保証される先行制約の集合が $\prec_{ne}$ から取り除かれる (行 (6))．ループは，$\prec_{ne}$ が空となったとき，すなわち与えられたプログラムが要求する先行制約のすべてがプロセッサの逐次実行及びバリア同期により保証されたとき，終了する．

定理 1 アルゴリズム INSERT-BARRIER-REGION は正当である．
(証明) 定理を証明するために次の補題を用いる．

補題 1 アルゴリズム中のループはすべて必ず終了する．
(証明) (1) の while ループは，行 (2),(6) より，繰り返しごとに $|\prec_{ne}|$ を 1 つずつ減少させることがいえるので，いつか必ず $\prec_{ne}=\phi$ となりループは終了する．このとき n は有限な値となっているので，(7)(8) の for ループも必ず終了する． □

補題 2 最終結果の B_u は式 (2) を満たす．
(証明) 行 (4),(5) により決定された p,p' におけるバリア領域の入口出口の位置により，先行制約 (t,t') は保証される．この位置は (7),(8) の for ループ実行後も移動しないので，最終結果の B_u は $\prec_C \cap \overline{\prec_S}$ を保証する．よって補題は成立する． □

補題 3 最終結果の B_u は式 (1) を満たす．
(証明) (7) の for ループにより決定されるバリア領域の入口には

$$\forall_{p\in P}\ \forall_{i(1<i\le|s_p|)}\ ; b_{i-1}[p].entr \le b_i[p].entr$$

の関係があり，(8) の for ループにより決定されるバリア領域の出口には

$$\forall_{p\in P}\ \forall_{i(1\le i<|s_p|)}\ ; b_i[p].exit \le b_{i+1}[p].exit$$

の関係があるので，式 (1) を満たす． □

以上の補題より定理が示された． □

定理 2 アルゴリズム INSERT-BARRIER-REGION は，任意の入力に対して，$\tau_{total}^{\prec_{M_{B_u}},e}$ を最小とする B_u を出力する．
(証明)

補題 4 (下界) 任意の B_u について，

$$\tau_{total}^{\prec_{M_{B_u}},e} \ge \tau_{total}^{\prec_{M_0},e} \qquad (3)$$

が成立する．
(証明) 一般に，$\prec_A \subseteq \prec_B \Rightarrow \tau_{total}^{\prec_A,e} \le \tau_{total}^{\prec_B,e}$ であり，正しい実行が保証されるような任意の $\prec_{M_{B_u}}$ について $\prec_{M_{B_u}} \supseteq \prec_{M_0}$ であるので，補題は成立する． □

補題 5 (上界) 任意の G,P,S について，アルゴリズム INSERT-BARRIER-REGION が出力する B_u は，

$$\tau_{total}^{\prec_{M_{B_u}},e} \le \tau_{total}^{\prec_{M_0},e} \qquad (4)$$

を満す．
(証明) (1) の while ループの各繰り返しにおける $(t,t'),b_i$ について，$pred(b_i)=t$，$succ(b_i)=t'$ と定義する．ここで，行 (2) より，ループの繰返しごとに $\tau_{fin}^{\prec_{M_0},e}(pred(b_i))$ が単調増加することがいえるので，

$$\forall_{b_j,b_k\in B_u(j<k)}\ ; \tau_{fin}^{\prec_{M_0},e}(pred(b_j)) \le \tau_{fin}^{\prec_{M_0},e}(pred(b_k)) \qquad (5)$$

が成立する．

ループの各繰り返しにおいて，バリア領域 b_i の入口出口は，t に割り当てられているプロセッサ p に対する $b_i[p].entr$ は，$s_p[j]=t$ なる j (行 (5)) と，t' に割り当てられているプロセッサ p' に対する $b_i[p'].exit$ は，$s_{p'}[j'+1]=t$ なる j' (行 (6)) と決定される．p 以外の p'' に対する $b_i[p''].entr$ は 0 (行 (3)) と，p' 以外の p'' に対する $b_i[p''].exit$ は $|s_{p_{|P|}}|$ (行 (3)) とされる．そしてその後，(7) の while ループでバリア領域の入口を決定 ($b_i[p].entr=0$ となっているものを $b_{i-1}[p].entr$ に変更) し，(8) の while ループでバリア領域の出口をを決定 ($b_i[p].exit=|s_{p_{|P|}}|$ となっているものを $b_{i+1}[p].exit$ に変更) するため，

$$b_i[p].entr = \begin{cases} j \text{ where } s_p[j]=pred(b_i) & pred(b_i)\in s_p \\ b_{i-1}[p].entr & \text{otherwise} \end{cases}$$

$$b_i[p].exit = \begin{cases} j \text{ where } s_p[j+1]=succ(b_i) & succ(b_i)\in s_p \\ b_{i+1}[p].exit & \text{otherwise} \end{cases} \qquad (6)$$

の関係が成立することとなる．

補題 6 任意の $b_i\in B_u$ について，

$$\max_{p\in P, b_i[p].entr>0}\{\tau_{fin}^{\prec_{M_0},e}(s_p[b_i[p].entr])\} \le \min_{p\in P, b_i[p].exit<|s_p|}\{\tau_{st}^{\prec_{M_0},e}(s_p[b_i[p].exit+1])\} \qquad (7)$$

が成立する．
(証明) 式 (7) が満されないとすると，

$$\tau_{st}^{\prec_{M_0},e}(s_p[b_i[p].exit+1]) < \tau_{fin}^{\prec_{M_0},e}(s_{p'}[b_i[p'].entr]) \qquad (8)$$

なる p,p' が存在する．この p,p' について，式 (6) より，$pred(b_j)$ にプロセッサ p' が割り当てられている $j(j\le i)$，および $succ(b_k)$ にプロセッサ p が割り当てられている $k(i\le k)$，が存在する．一般に $\tau_{fin}^{\prec_{M_0},e}(pred(b_k)) \le \tau_{st}^{\prec_{M_0},e}(succ(b_k))$ であるので，

$$\begin{aligned} \tau_{fin}^{\prec_{M_0},e}(pred(b_j)) &= \tau_{fin}^{\prec_{M_0},e}(s_{p'}[b_i[p'].entr]) \\ &> \tau_{st}^{\prec_{M_0},e}(s_p[b_i[p].exit+1]) \\ &= \tau_{st}^{\prec_{M_0},e}(succ(b_k)) \\ &\ge \tau_{fin}^{\prec_{M_0},e}(pred(b_k)) \end{aligned}$$

となり，式 (5) に矛盾する．よって補題は満たされる． □

補題 6 は，どのバリア領域においても，入口よりも早い時刻に出口に到達するプロセッサは存在せず，$\prec_{B_u}$ によってプロセッサが実行を休止させられることがないことを示している．したがって $\tau_{total}^{\prec_{M_{B_u}},e} \le \tau_{total}^{\prec_{M_0},e}$ である． □

補題 4,5 よりアルゴリズム INSERT-BARRIER-REGION が，任意の入力に対して，$\tau_{total}^{\prec_{M_{B_u}},e}$ を最小とする B_u を出力することがいえる． □

4. むすび

重複可能なバリア型同期以外の同期操作を併用せずとも，プログラムの正しい実行順序が保証されるようなバリア挿入位置を求める同期スケジューリングアルゴリズム INSERT-BARRIER-REGION を示し，このアルゴリズムが任意の入力に対して最適なバリア挿入位置を出力することを示した．また最適なバリア挿入を行なった場合の実行系 M_{B_u} によるプログラム G の処理時間が，実行系 M_0 による実行の処理時間に一致することも示されている．このことは，個々のデータの依存関係に基づいた同期方式と同等の性能を，バリア型同期のような単純な同期機構でも達成できることを示している．

今後の課題としては，アルゴリズムの計算コストの見積もり，および計算コストの改善などが挙げられる．

参考文献

[1] 高木浩光, 有田隆也, 曽和将容: 細粒度並列実行を支援する種々の静的順序制御方式の定量的評価, 並列処理シンポジウム JSPP'91 論文集, pp.269–276 (1991).

[2] 松本 尚: 細粒度並列実行支援マルチプロセッサの検討, 情報処理学会論文誌, Vol.31, No.12, pp.1840–1851 (1990).

[3] 高木浩光, 有田隆也, 川口喜三男, 曽和将容: バリアを唯一の同期手段とした場合のタスクスケジューリング, 電子情報通信学会技術研究報告 CPSY93-22, pp.73-80 (1993).