3U−04

スケジューリングにおける利用プロセッサ制限とその一時解除による稼働率改善の実現

柏木 紘一　樋上 喜信　小林 真也

愛媛大学工学部

1　はじめに

マルチプロセッサシステムでジョブを高速に実行するには，ジョブをいくつかのタスクに分割し，更に，タスク間の依存関係を考慮して各タスクの実行プロセッサや実行順序を決定しなければならない．タスク間の通信時間を無視できる場合のスケジューリング法としてCP/MISF法 [1] が提案されている．しかし，CP/MISF法等のリストスケジューリングの多くは，利用可能プロセッサ数に余裕がある場合，処理時間の短縮が微小，あるいはできないにもかかわらず，必要以上にタスクを多くのプロセッサに分散させてしまうものがある．その結果，プロセッサ稼働率は低下する．

我々はこの問題に対して、まず，利用プロセッサ数に制限をかける方法を提案し稼働率の改善を実現した [2]．このとき問題となったタスク集合の総処理時間増加に対して，各タスクに設定したデッドラインによって処理時間に影響を与えない方式 [3] を提案し，問題の解決を図った．本論文ではこの方式を改良し，より高い稼働率実現を目指す．

2　利用プロセッサ数の制限

スケジューリングでは一般的に，利用プロセッサ数の増加は，ある台数までは処理時間を効果的に短縮する．しかしそれ以上の利用プロセッサ数の増加は，処理時間の短縮効果は少なく，稼働率の低下につながる．

例えばCP/MISF法では，実行可能となったタスクは処理を開始するまでに余裕がある場合でも，空きプロセッサがあるとその時点で割り当てられてしまい，上記の問題を引き起こす．

そこで我々はタスク集合毎に利用プロセッサ数を制限することで必用以上にプロセッサを用いない方式である，利用プロセッサ数制限法 [2] を提案している．この方式により稼働率の改善が実現されたが，同時にわずかではあるが処理時間増加が引き起こされた [2]．

これに対して，処理時間に影響を与えずに稼働率を改善する方式として，タスク毎のデッドラインを考慮した制限法（デッドライン法）を提案している [3]．タスク毎のデッドラインは，タスク集合の総処理を最短で終えるために，少なくとも処理を開始しなければならない時刻と定義する．制限法により利用プロセッサ数を制限した場合，あるタスクが自身のデッドラインをむかえても割り当てられないことがあり，そのときタスク集合の総処理時間は増加する．そこでデッドライン法では，デッドラインをむかえたタスクが制限内のプロセッサに割り当てられなかった場合のみ，制限を一時的に解除して制限外のプロセッサへ割り当てることで，タスク集合の総処理時間増加をなくした．

しかし文献 [3] による制限外プロセッサへの割当ては，デッドラインの時刻に割当て対象となるタスクの数だけ制限外プロセッサを用いておりプロセッサの利用効率が悪い．そこで本論文では，タスクの割当てをデッドラインの時刻よりも可能な限り早い時刻とし，各タスクの割当て時刻をずらして1つのプロセッサで複数のタスクを処理できるようにすることで，プロセッサの稼働率を高める．

3　制限値の設定

利用プロセッサ数を制限する場合，タスク集合が本来持つ並列性を損なわず，かつ十分生かしきることのできるような適切な利用プロセッサ数を制限値として設定する必要がある．そこで，利用プロセッサ数の制限を行うために，利用可能なプロセッサ数が十分にあるという条件下でタスク集合を割り当てると仮定して，以下のように制限値を計算する．

まず，割当て開始時刻から最短の総処理終了時刻までの各瞬間毎において，各タスクが処理されている確率を計算する．そして，ある時刻において各タスクが処理されている確率の和が，ある時刻において処理されているタスク数の期待値と表すことができる．このようにして求められる各瞬間毎の期待値は，そのタスク集合を最適に並列処理するために，その時点において利用される理想的なプロセッサ数になると考えられる．そして，時間軸における期待値の最大値を，そのタスク集合の並列度と呼び，その値を基に制限値を定める．一般に並列度は実数であり，制限値は自然数であるので，従来は並列度の小数点第一位を切り上げた値を制限値としていた．本論文では，小数点第一位を切り捨てた値も含めて2通りの制限値を考える．次に具体的な計算式を説明する．

タスク集合を $\{T_1, T_2, \cdots, T_N\}$，タスク T_i から直後タスクが存在しないタスクへ至る経路のうちで最長のものの経路長（最長パス長）を cpl_i，各 i における cpl_i の最大値（最大最長パス長）を $lcpl$，T_i の処理を最も早く開始できる時刻（最早開始時刻）を est_i，最大最長パス長で全処理を終えるために T_i の処理を遅くとも終えておかなければならない時刻（最遅終了時刻）を lct_i，タスク T_i のタスクサイズを s_i とする．これらは，次のように求められる．ここで $P(T_i)$ は T_i の直前タスク集合とする．

Improvement of processors operating ratio with restriction and momentary release of used processors for scheduling

Koichi Kashiwagi, Yoshinobu Higami and Shin-ya Kobayashi
Faculty of Engineering, Ehime University
{kashiwagi,higami,kob}@koblab.cs.ehime-u.ac.jp

$$est_i \equiv \begin{cases} \max\limits_{T_j \in P(T_i)} \{est_j + s_j\} & : P(T_i) \neq \emptyset \\ 0 & : P(T_i) = \emptyset \end{cases}$$

$$lct_i \equiv lcpl - cpl_i + s_i$$

これらを用いて，時刻 t に タスク T_i が処理される確率 $f_i(t)$ を次式のように算出する．

・ $lct_i - est_i \leq 2s_i$ の場合

$$f_i(t) \equiv \begin{cases} 0 & : t < est_i,\ lct_i \leq t \\ \frac{t-est_i}{lct_i-est_i-s_i} & : est_i \leq t < lct_i - s_i \\ 1 & : lct_i - s_i \leq t < est_i + s_i \\ \frac{lct_i-t}{lct_i-est_i-s_i} & : est_i + s_i \leq t < lct_i \end{cases}$$

・ $lct_i - est_i > 2s_i$ の場合

$$f_i(t) \equiv \begin{cases} 0 & : t < est_i,\ lct_i \leq t \\ \frac{t-est_i}{lct_i-est_i-s_i} & : est_i \leq t < est_i + s_i \\ \frac{s_i}{lct_i-est_i-s_i} & : est_i + s_i \leq t < lct_i - s_i \\ \frac{lct_i-t}{lct_i-est_i-s_i} & : lct_i - s_i \leq t < lct_i \end{cases}$$

そして，時刻 t における処理中タスク数の期待値 $F(t)$ は，$\sum_{i=1}^{N} f_i(t)$ で与えられ，並列度は $\max\limits_{0<t\leq lcpl} \{F(t)\}$ となる．並列度の小数点第一位を切り上げた値を制限値 l_c，と切り捨てた値を制限値 l_f とする．

4 デッドライン法のアルゴリズム

ある割当て時刻 t でデッドラインをむかえて制限内のプロセッサに割り当てられなかったタスクが複数あった場合を考える．制限外のプロセッサには一意の番号を付ける．このとき，文献 [3] の “デッドライン法 Method A” では，タスクサイズの最も小さいタスクと，プロセッサ番号の最も小さい空きプロセッサの組を選択し，それらのタスクをデッドラインの時刻で割り当てる．この方式において制限値 l_c と l_f を用いたものをそれぞれ Method A (ceiling), Method A (floor) と呼ぶ．

提案する新しい方式では，まず 1 つのタスクとプロセッサの組を上と同様に撰択し，割当て時刻を，そのタスクの割当て可能時刻と，そのプロセッサで最後に処理をしたタスクの終了時刻との大きい方の時刻とする．このようにタスクの割当てをデッドラインの時刻より可能な限り早めることで，今割り当てたタスクがデッドラインの時刻よりも早く処理が終了した場合，そのタスクを処理したプロセッサに別のタスクを割り当てることが可能となり，制限外のプロセッサを利用する個数を減らすことができる．この方式に対して制限値 l_c と l_f を用いたものをそれぞれ Method A’ (ceiling), Method A’ (floor) と呼ぶ．

5 疑似タスクによる評価

割当て対象タスク集合を乱数により生成したもので評価する．タスク集合生成の条件は，タスク数 1000 個，平均タスクサイズ 100 [u.t] ([u.t] は単位時間)，依存確率と依存範囲は，(0.1, 60), (0.2, 30), (0.3, 20), (0.4, 15), (0.5, 12) の 5 通りとした．平均依存個数（依存確率 × 依存範囲）はいずれの場合も 6 となっている．さらに，全タスクに対して 0.3 の確率で平均 3 個の複製を付加した．これら 5 通りのタスク集合を乱数によりそれぞれ 10 組生成し，プロセッサ数 32，完全網のシステム上で，CP/MISF 法と CP/MISF 法に前節の 4 種類のデッドライン法を適用した方式によりスケジューリングを行ったときの平均稼働率を表 1 に示す．ここで稼働率はタスク集合の全タスクサイズを，利用したプロセッサ数と総処理時間の積で割った値である．表 1 より，デッドライン法により稼働率が改善されているのが分かる．また，Method A’ の稼働率が高いことから，今回提案したアルゴリズムが効果的であったことも分かる．また，制限値 l_f を用いたデッドライン法の方が l_c を用いるよりも稼働率が高い．これは，制限値が小さい方が，処理を開始できるのにも関わらず制限によって制限内のプロセッサに割り当てられなったタスクが多くなり，制限外のプロセッサに割り当てられるタスクも多くなるが，制限値が 1 小さい分，制限外プロセッサに 1 台余裕があり，その 1 台にそれらのタスクが効率良く割り当てられたために，最終的に利用したプロセッサ数が少なくなるためである．

表 1: 各方式における稼働率の平均

	CP/MISF	Method A		Method A’	
		ceiling	floor	ceiling	floor
稼働率の平均	0.207	0.257	0.265	0.258	0.267

6 まとめ

利用可能プロセッサ数が十分ある場合，リストスケジューリングが必要以上にプロセッサを利用しシステムの稼働率低下をもたらす問題に対して，利用プロセッサ数制限法があったが，タスク集合の総処理時間増加をなくすために，デッドライン法を提案した．デッドラインは各タスク毎に設定され，デッドラインの時刻をむかえても制限内のプロセッサに割り当てられなかったタスクのみを制限外プロセッサへ割り当てることで，総処理時間増加を引き起こさずに稼働率の改善を図った．本論文では制限外プロセッサへの割当てアルゴリズムを新たに提案し，乱数による擬似タスク集合に対して評価を行った結果，稼働率の改善が実現された．また，制限値として 2 通りの値を設定したが，小さい制限値の方が高い稼働率を示した．これは，デッドライン法による割当てアルゴリズムの効率が良かったためと考えられる．

今後は，制限外プロセッサへの割当てにおける，タスクとプロセッサ対の選択アルゴリズムを改良し，さらなる稼働率改善を目指す．

参考文献

[1] 笠原博徳，“並列処理技術”，コロナ社, 1991.

[2] 柏木紘一，森麻衣子，小林真也，“利用プロセッサ数制限によるスケジューリングのプロセッサ稼働率改善の評価”，情報処理学会 第 61 回全国大会 Vol. 1, pp. 89-90.

[3] K. Kashiwagi and S. Kobayashi, “Consideration of task’s deadline for scheduling method with used processors limitation,” in *Proc. ACS’2001*, Oct. 2001, pp. 487-496.