利用プロセッサの制限によるタスクスケジューリングの改善

1L－3

小林 真也　木股 洋
金沢大学 工学部 電気・情報工学科

1 はじめに

マルチプロセッサシステムで高速に並列処理を行うためにはタスクの実行プロセッサ・実行順序を決定するタスクスケジューリングが重要である。しかし，従来のスケジューリングアルゴリズムはプロセッサ台数が多い場合に過度なタスクの分散により処理時間の増加をもたらすことがある．そこで本稿では，タスク集合の特性に基づいて利用プロセッサを制限することで処理時間の改善を行う方法について述べる．

2 制限値の設定

通信の影響が無視できる場合，プロセッサ数が十分あり，かつ最適な割当てが可能であれば実行時間はタスクグラフの最大最長パス長となる．この場合以下の1~3によって各瞬間毎の処理中タスク数の期待値を求めることができる [3]．本稿ではこの期待値により各タスクの利用プロセッサの制限値を設定する．

1. タスク T_i の最早開始時刻 est_i・最遅終了時刻 lct_i を計算.
ここで est_i は T_i が最も早く処理を開始できる時刻，lct_i は最大最長パス長で全処理を終えるために T_i が遅くとも処理を終えておかなければならない時刻を表す.

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

$$lct_i \equiv lcp - cp_i + \sigma_i$$

ただし，σ_i：タスク T_iの処理時間，$P(T_i)$：T_iの直前タスク集合，cp_i：T_iの最長パス長，lcp：タスクグラフの最大最長パス長.

2. 時刻 t に T_i が処理される確率 $f_i(t)$ を計算.

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

3. 処理中タスク数の期待値 $F(t)$ を計算.

$$F(t) = \sum_{i=1}^{N} f_i(t)$$

ただし，N：タスク集合のタスク数.

4. T_i の制限値 p_i を決定.

$$p_i \equiv \max_{est_i < t \leq lct_i} \{F(t)\}$$

3 利用プロセッサの制限

スケジューリングでは，その直前タスクが全て終了したタスクは割当て可能となり，これらのタスクの中からプライオリティに従って順次割当てタスクが決定される．

利用プロセッサに制限を加えるために，本研究では以下の２つの条件を満たすタスクを割当て可能タスクとする．

条件１：直前タスクの全てが終了している

条件２：$p_i \geq p_{busy}$
p_i：各タスクの利用プロセッサ制限値
p_{busy}：現在稼働中のプロセッサ台数

4 シミュレーション

4.1 ランダムタスク

利用プロセッサ制限による処理時間の改善効果を調べるため，乱数により作成したランダムタスク集合に対してシミュレーションを行った．なお今回はCP/DT/MISF[1],CP/RCO[2]に対してプロセッサ制限を適用した．

タスク集合はタスク数:500，1タスクの平均実行時間:100[unit time]，平均通信時間:20,60[unit time]，各タスクが20個のタスクに対して0.3の確率で依存関係を持つものとした．ただしタスク集合に並列性の変化をもたせるために全タスク中のd%のタスクは特定のタスクのみに依存する，つまり並列処理可能なタスクであるとし，ここではdの値を5,10,20,30%と変化させて各100組生成した．

図１，２にプロセッサ数を64台とした場合の結果

Improvement of Task Scheduling by Limiting Used Processors
Shin-ya KOBAYASHI, Hiroshi KIMATA
Department of Electrical and Computer Engeneering, Kanazawa University

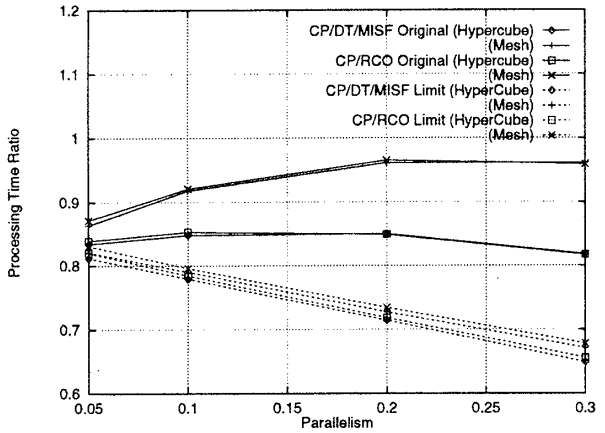

図 1: 並列タスク割合 vs 処理時間比 (平均通信時間 20)

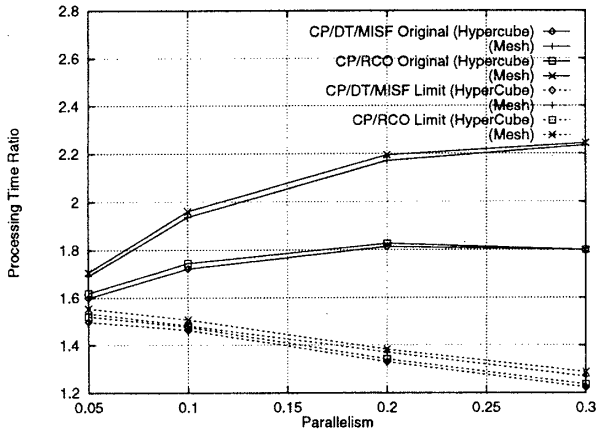

図 2: 並列タスク割合 vs 処理時間比 (平均通信時間 60)

を示す．縦軸は逐次処理の処理時間に対する処理時間比の平均値，横軸は並列処理可能なタスクの割合 (並列タスク割合) を示す．Original は従来法での結果，Limit はそれにプロセッサ制限をかけたときの結果を表し，それぞれプロセッサ間結合形態が Hypercube 結合, Mesh 結合の場合について示してある．並列処理可能なタスクの割合が大きいほど，また通信時間の大きいほど Original の処理時間増加は大きく，結果からこのようなタスグ集合に対して特に利用プロセッサ制限による処理時間の抑制効果が大きいことが分かる．

4.2 Runge-Kutta

文献 [2] に取り上げられている 4 次の Runge-Kutta 法による常微分方程式の数値解計算を行うタスク集合

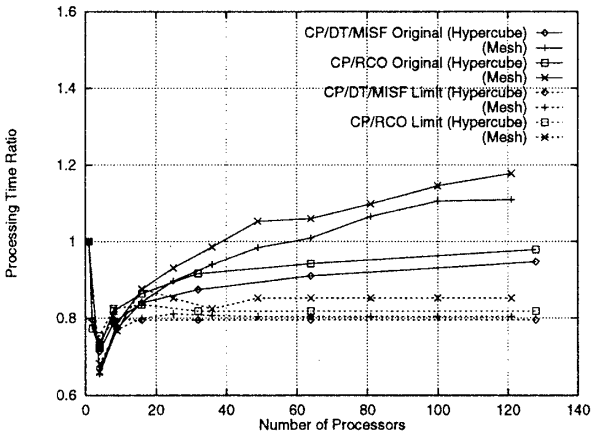

図 3: プロセッサ数 vs 処理時間比

に対して同様にシミュレーションを行った．図 3 に結果を示す．縦軸は逐次処理の処理時間に対する処理時間比，横軸はプロセッサ数を示す．従来法 (Original) ではいずれもプロセッサ数の増加に従って処理時間が増加するのに対し，制限を加える (Limit) ことで処理時間増加が抑制されている．

5 おわりに

本稿ではタスク集合の特性に基づく利用プロセッサ制限による処理時間の改善方法について述べ、シミュレーションにより評価を行った．その結果、並列処理可能なタスクが多いほどその改善効果が高いことを示した．今後はより多くのタスク集合を対象に評価を行う予定である．

参考文献

[1] 笠原博徳, “並列処理技術”, コロナ社, (1991)

[2] 小林真也, “不完全結合マルチプロセッサシステムに対するタスク割当て法の提案と評価”, 信学論 D-I, Vol.J79-D-I, No.2, pp.69-78 (1996)

[3] 小林真也, “タスクスケジューリングにおける利用プロセッサ数の制限法”, 信学論 D-I, Vol.J81-D-I, No.3, pp.353-355 (1998)

[4] 木股洋, 小林真也, “タスクスケジューリングにおける利用プロセッサ数の制限法に対する考察”, 情処研報 98-HPC-73, pp.67-72(1998)