計算機アーキテクチャ 132－4
システムソフトウェアとオペレーティング・システム 80－4
ハイパフォーマンスコンピューティング 75－4
（1999. 3. 4）

通信時間を削減するためのタスク複製の手法

高木 秀樹† 李 鼎超‡ 石井 直宏†

†名古屋工業大学知能情報システム学科 ‡名古屋工業大学情報処理教育センター

†名古屋工業大学知能情報システム学科
‡名古屋工業大学情報処理教育センター
†E-mail: {takagi, ishii}@egg.ics.nitech.ac.jp
‡E-mail: liding@center.nitech.ac.jp

本稿では，並列プログラムを分散メモリ並列アーキテクチャ上にスケジューリングする際に発生する通信遅延を削減するためのタスク複製の手法を提案する．提案する手法において，タスク複製による通信コストの減少を定量的に分析し，選ばれたタスクを複製すべきかどうかの条件を導く．本手法をよく知られているいくつかのスケジューリングアルゴリズムに対して実装し評価を行ない，その有用性を確認した．

Task Duplication Techniques for Reducing Communication Delays

Hideki Takagi† Dingchao Li ‡ and Naohiro Ishii †

†Department of Intelligence and Computer Science, Nagoya Institute of Technology
‡Educational Center for information processing, Nagoya Institute of Technology
†E-mail: {takagi,ishii}@egg.ics.nitech.ac.jp
‡E-mail: liding@center.nitech.ac.jp

This work discusses how to use task duplication to reduce communication delays for assigning parallel programs onto distributed memory multiprocessors. Through an instantaneous performance gain analysis, we establish the conditions under which our task replication policy is beneficial. Experimental study shows that incorporating the proposed strategies in several well-known priority-based scheduling algorithms Dose improve their performance with very low computational Costs.

1 はじめに

分散メモリ並列アーキテクチャにおける並列プログラムの実行効率は，プログラムの並列性を抽出する並列コンパイラ，特にタスクスケジューリング時に使用される戦略に大きく依存する．タスクスケジューリングにおいてプロセッサ間の通信時間を減少させる手法の一つに，タスク複製の手法がある．タスク複製の手法とは，メッセージを送るタスクを受け取るタスクと同じプロセッサ上に複製することにより，通信時間を削減する手法である．

いくつかのタスク複製を取り入れたスケジューリングアルゴリズムが研究されている [1, 2, 3, 4]．しかしながら，それらのアルゴリズムは全てプロセッサ数に制限がないと仮定している．したがって，アルゴリズムが必要とする数のプロセッサが利用可能でない場合，いくつかの問題が発生する．本稿では，利用可能なプロセッサ数が制限されている場合のタスクスケジューリングのためのタスク複製の手法を提案する．同様の手法として，Kwok らが提案したタスク複製の手法 [5] がある．この手法では，スケジューリング前にクリティカルパス上のタスクを決定する，そして，クリティカルパス上のタスクをすべてのスケジューリングステップにおいて複製することを試みる．しかしながら，同じプロセッサに割り付けられた 2 つのタスクの間の通信時間は 0 であるため，新たなクリティカルパスがス

ケジューリング中に発生する．したがって，本稿の手法においては，複製するタスクをクリティカルパス上のタスクに限定せず，プログラムの実行時間の増加を動的に計算し，タスクの複製が可能な場合に先行タスクを複製する．

また，一般的に，タスク複製を行なうスケジューリングアルゴリズムは，行なわないアルゴリズムと比較して短いスケジューリング長を得ることができる．一方，タスク複製を行なう場合，複製するタスクの選択，複製するタスクを割り付けるスケジューリングホールの発見などいくつかの問題を考慮する必要がある．そのため，タスクの複製を行なうアルゴリズムは行なわないアルゴリズムと比較して複雑になる．このように，タスク複製を実行することによるスケジューリング長の改善と，スケジューリングアルゴリズムの処理時間の間にはトレードオフが存在する．そのため，本稿では単純に一回だけタスク複製を実行する手法，(アルゴリズム A と呼ぶ) と，タスク複製を再帰的に実行する手法，(アルゴリズム B と呼ぶ) の 2 つのアルゴリズムを提案し，2 つのアルゴリズムのスケジューリング長の改善と処理時間の増加を検討する．

本稿は，以下 4 章で構成される内容は以下の通りである．第 2 章では，本稿で対象としたマシンモデルとプログラムモデルについて述べる．第 3 章では，本稿で提案するタスク複製の手法について説明する．第 4 章では，逐次計算機上での実験及び，考察を行なう．第 5 章では，本稿のまとめと今後の課題について述べる．

2 モデル定義

2.1 分散メモリアーキテクチャ

本稿は対象とする計算機モデルは，任意の方法で相互結合された均一のプロセッサにより構成される並列実行が可能な計算機である．プロセッサはそれ自身のプライベートメモリを持ち，共有のメモリを持たない．プロセッサ間の通信はすべて相互結合網を通して一台のプロセッサから一台のプロセッサに明確にメッセージを送ることで行なわれる．同じプロセッサ上で実行されるタスク間においてメッセージの交換が必要な場合，データはローカルメモリから直接読み込まれる．

$\mathcal{M} = (\mathcal{P}, \lambda)$ はターゲットマシンを表す．$\mathcal{P}$ はプロセッサ $P_i (i = 1, \cdots, n)$ の有限の集合を表す．関数 $\lambda(P_i, P_j)$ はプロセッサ P_i からプロセッサ P_j へ 1 ユニットのメッセージを送る場合に必要な時間 (転送率) を表す．$P_i = P_j$ の場合の通信時間は，タスクの実行時間と比較して非常に小さいため，通信時間は無視できる大きさであるとする．なお，実行時間の単位は time unit である．

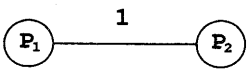


図 1 マルチプロセッサシステム

図 1 は 2 台のプロセッサにより構成されるプロセッサシステムの例である．ノードはプロセッサを表し，ノード内の数字はプロセッサ番号を表す．また，アーク上の数字は，各プロセッサ間の転送率 λ を表す．

2.2 タスクグラフ

本稿では，タスクグラフと呼ばれる有向無サイクルグラフを扱い，これを用いて並列プログラムを表現する．タスクグラフはノードとエッジによってタスクとタスク間の先行制約を表現する．

$G = (\Gamma, A, \mu, \eta)$ はタスクグラフを表す．Γ はタスク $T_i (i = 1, \cdots, n)$ の有限の集合を，A はアーク $(T_i, T_j)(i, j = 1, \cdots, n, i \neq j)$ の有限の集合を表す．関数 $\mu(T_i)$ は T_i の実行時間を，関数 $\eta(T_i, T_j)$ はタスク T_i の実行終了時に T_i から T_j に送られるメッセージの数を返す関数である．$P(T)$ は，タスク T を実行するプロセッサを表す．アーク (T_i, T_j) は，T_i の実行が完了し T_i からのメッセージが T_j に到着するまで T_j を実行開始できないことを意味する．この時，T_i を T_j の直接先行タスクと呼び，T_j を T_i の直接後続タスクと呼ぶ．そして，$Pred(T_i)$ を T_i の直接先行タスクの集合，$Succ(T_i)$ を T_i の直接後続タスクの集合とする．プロセッサ間の通信は，直接先行タスク T_i のメッセージを直接後続タスク T_j が必要とし，かつ T_i, T_j がそれぞれ異なるプロセッサに割り付けられた場合に発生する．

T_i, T_j 間の通信時間は以下の式で与えられる．

$$\begin{cases} 0 & (P(T_i) = P(T_j)) \\ \eta(T_i, T_j) \times \lambda(P(T_i), P(T_j)) & otherwise \end{cases}$$

図 2 は並列プログラムをタスクグラフ表現した例である．

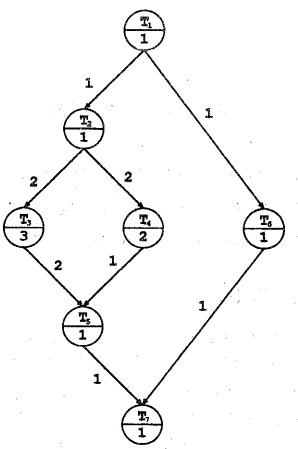


図 2 タスクグラフ

2.2.1 粒度

タスクグラフの粒度はタスクスケジューリングアルゴリズムの性能評価をする際に重要な要素である．粒

度は，並列コンパイラによりプログラムを最適化する際に，プログラムをどのレベルでタスクに分割したかにより決定される．本稿では [9] で与えられた粒度の定義を採用する．粒度は以下の式で計算される．

$$g_1(T_j) = \frac{\min\{\mu(T_i)|(T_i,T_j) \in A\}}{\max\{\lambda(T_i,T_j)|(T_i,T_j) \in A\}}, T_j \in \Gamma$$

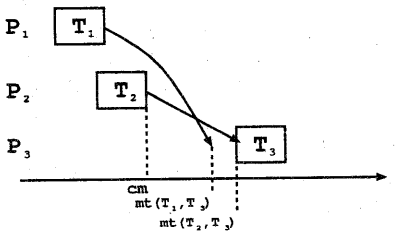
$$g_2(T_j) = \frac{\min\{\mu(T_k)|(T_j,T_k) \in A\}}{\max\{\lambda(T_j,T_k)|(T_j,T_k) \in A\}}, T_j \in \Gamma$$

T_j の粒度 $g(T_j)$ は $\min\{g_i(T_j), g_2(T_j)\}$ である．
タスクグラフ G の粒度 $g(G)$ は次式で与えられる．

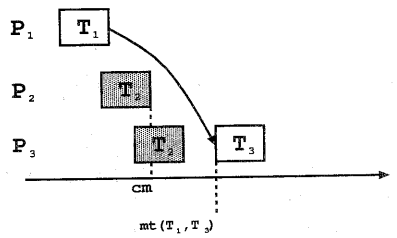
$$g(G) = \min\{g(T_j)|T_j \in \Gamma\}$$

タスクグラフ G は，$g(G) \leq 1$ の場合，粗粒度である．

3 タスク複製



(a) タスク複製実行前



(b) タスク複製実行後

図 3(a) タスク複製前．(b) タスク複製後

以下に本節で使用する表記を説明する．$st(P,T), ct(P,T)T$ をプロセッサ P に割り付ける．はそれぞれ P 上で T を実行した時の T の実行開始時刻と実行完了時刻を表す．cm はイベントクロックのカレント時刻を，T_{ready},P_{idle} はそれぞれ cm 時に利用可能なプロセッサとタスクの集合を表す．また，$mt(T_i,T_j)$ をプロセッサ $P(T_j)$ にプロセッサ $P(T_i)$ からメッセージが到着する時刻とし，以下の式で計算する．

$$mt(T_i,T_j) = ct(T_i) + \lambda(P(T_i),P(T_j))$$

図 3 にタスク複製の簡単な例を示す．

T_3 は 2 つの直接先行タスク，T_2 と T_1 を持つ．図 3(a) はタスク複製を実行しない場合のスケジュール結果である．T_2 から T_3 へのメッセージが P_2 から P_3 に到着する時刻は $mt(T_2,T_3)$ である．T_1 から T_3 へのメッセージが P_2 から P_3 に到着する時刻は $mt(T_1,T_3)$ である．2 つのメッセージの到着時刻 $mt(T_2,T_3)$ と $mt(T_1,T_3)$ を比較する．この場合，$mt(T_2,T_3)$ がより遅い時刻に P_3 に到着する．このため，P_3 上の T_3 の実行開始時刻は $mt(T_2,T_3)$ となる．つまり，T_2 からのメッセージの到着時刻により T_3 の実行開始時刻が決まる．

図 3(b) はタスク複製を実行する場合のスケジュール結果である．T_2 からのメッセージが最も遅く P_3 上の T_3 に到着するため，T_2 を P_3 に複製する．タスク複製の結果，T_3 の新たな実行開始時刻は $mt(T_1,T_3)$ となる．したがって，T_3 の実行開始時刻はタスク複製を行なわない場合と比較して，$mt(T_2,T_3) - mt(T_1,T_3)$ 早くなる．

3.1 アルゴリズム A

本節では，タスクの複製を一回のみ行うタスク複製の手法，アルゴリズム A について述べる．

あるスケジューリングステップ cm において最も優先度の高いタスクとプロセッサの組み (T,P) が選択されたと仮定する．

この時，アルゴリズム A,algoA(T,P) は次の手順で行なう．

1. 遅延の計算

タスク T の直接先行タスクの集合 $Pred(T)$ の全てのタスクに対して $mt(T_i,T)(T_i \in Pred(T))$ を計算する．

タスクの複製を実行しない場合のタスク T の実行開始時刻 $st_{nodup}(P,T)$ は以下の式で計算される．

$$st_{nodup}(P,T) = \max_{(T_i) \in Pred(T)}\{mt(T_i,T), cm\}$$

このため，タスク複製を実行しない場合のタスク T によるプログラム全体の実行時間に対する遅延 $delay_{nodup}(P,T)$ は次式のようになる．

$$delay_{nodup}(P,T) = st_{nodup}(P,T) - lst(T)$$

遅延が存在しない場合タスク T は全体のスケジュール長にたいして影響を及ぼさないため，タスク複製は行わず，スケジューリングアルゴリズムに従いタスク

2. 複製タスクの選択

遅延が存在する場合，タスク T に最も遅く到着するメッセージを送るタスク T' を以下の式で求める．

$$mt(T',T) = \max_{(T_i) \in Pred(T)}\{mt(T_i,T)\}$$

タスク T の実行開始時刻はこのタスク T' によって決定されるため，タスク T' を複製タスクとして選択する．

3.Gain の計算

プロセッサ P 上でタスク T' が実行完了する時刻 $ct(P,T')$ は以下の式で求められる．

$$ct(P,T') = \max_{(T_j \in Pred(T')}\{mt(T_j,T')+\mu(T'), ct(P)+\mu(T')\}$$

また，T に2番目に遅く到着するメッセージを送るタスク T'' は以下の式で求められる．

$$mt(T'',T) = \max_{(T_i)\in Pred(T),T_i\neq T'}\{mt(T_i,T)\}$$

この時，タスク T' をプロセッサ P に複製した場合のタスク T の実行開始時刻 $st_{dup}(T,P)$ は以下の式で与えられる．

$$st_{dup}(T,P) = \max\{mt(T'',T), ct(T',P(T')), cm\}$$

タスク複製を行った場合のタスク T 遅延 $delay_{dup}(T)$ は以下の式で計算される．

$$delay_{dup}(T) = st_{dup}(P,T) - lst(T)$$

そのため，複製を行ったことよる全体のプログラム実行時間における性能の改善 $Gain$ は以下の式で計算される．

$$Gain = delay_{dup}(T) - delay_{nodup}(T)$$

タスク複製は $Gain > 0$ の場合に実行する．

図4に図1のタスクグラフを，図2のプロセッサシステムにたいして DL 法 [7] によりスケジューリングした結果を示す．全体のスケジューリング長は 9 time unit である．また，図5に，アルゴリズム A を適用した DL 法によるスケジューリング結果を示す．図5においては，T_4 の実行開始時刻を早めるために T_2 の複製が実行される．タスク複製を実行しない場合のタスク T_4 の実行開始時刻は 4 time unit である．スク T_2 を複製することによりタスク T_4 の実行開始時刻は 3 time unit となり，1 time unit 早くなる．その結果全体のスケジューリング長はアルゴリズム A を適用しない場合と比較して 1 time unit 短くなる．

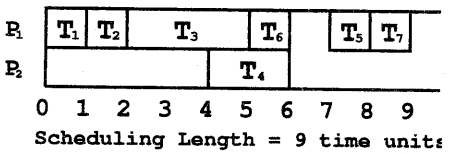


図4 DL 法

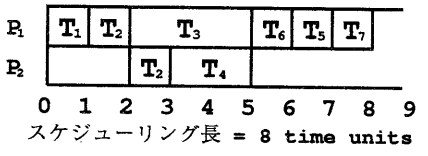


図5 アルゴリズム A

3.2 アルゴリズム B

本節では，タスク複製を再帰的に行なうタスク複製の手法，アルゴリズム B を説明する．アルゴリズム B においては，タスク複製を再帰的に行ない，依存関係の流れが存在する複数のタスクを複製する．

アルゴリズム B は次の手順で行なう．アルゴリズム A と同様に，スケジュールタスクを T，複製するタスクを T' とし，2番目に遅く T に到着するメッセージを送るタスクを T'' とする．また，遅延の計算，複製タスクの選択，複製による Gain の計算はアルゴリズム A と同様であるため，1 ～ 3 の手順はアルゴリズム A と同じである．

4. 複製の選択

タスク T' の複製が決定された時点で，さらに T' の直接先行タスクを複製を実行するかの選択を行なう．$st_{dup}(T,P) > mt(T'',P)$ の場合，タスク T' の直接先行タスクを複製対象としアルゴリズム B を行なう．それ以外の場合は，タスク T' の複製を実行し，タスク複製を終了する．

5. 実行開始時刻の再計算

いま，複製が再帰的に n 回実行されたとする．この時プロセッサ P 上には，n 個 $(T_{x_1}, T_{x_2}, \cdots T_{x_n})$ の複製タスクが複製される．最後に複製されたタスク T_{x_n} の複製が終了した時点で，タスク T_{n_1} のプロセッサ P 上の実行完了時刻 $ct(P,T_{x_n})$ が決まる．この時，$T_{x_{n-1}}$ の T の P 上の実行開始時刻 $st_{redup}(T_{x_{n-1}},P)$ は以下のように計算される．

$mt(T''_{x_{n-1}},P) > st_{redup}(T'_{x_{n-1}},P)$ の場合

$$st_{redup}(T_{x_{n-1}},P) = mt(T''_{x_{n-1}},P)$$

else

$$st_{redup}(T_{x_{n-1}},P) = ct(P,T'_{x_{n-1}})$$

この時，$T_{x_{n-1}}$ の P 上での実行終了時刻 $ct(T_{x_{n_1}},P)$ は $st_{redup}(T_{x_{n-1}},P)$ となる．

以上の計算を再帰的に行なうことで，すべての複製タスクの実行開始時間及び実行終了時刻が決定し，タスク T の実行開始時刻が決定する．

また，図6に，アルゴリズム B を適用した DL 法によるスケジューリング結果を示す．アルゴリズム A と同様に T_4 のスケジューリング時に T_2 が P_1 に複製される．この時，さらに，タスク T_2 の実行開始時刻を早めるために T_1 が P_1 に複製される．T_1 を複製したことにより，T_4 の実行開始時刻は 2 time unit となり，タスク T_1 を複製しない場合と比較して 1 time unit 早くなる．その結果，全体のスケジューリング長はアルゴリズム A と比較して 1 time unit 短くなる．

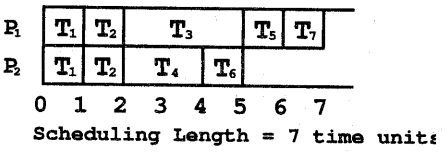


図6 アルゴリズム B

4 評価

4.1 性能評価

本節では，逐次計算機上において，ランダムに生成したタスクグラフにたいして，タスク複製の手法の性能評価及び時間増加量の評価を行なった．

性能評価及び，時間量の評価は逐次計算機上においては，ETF 法 [6]，DL 法 [7]，MH 法 [8]，アルゴリズム A 及びアルゴリズム B を適用した場合のそれぞれの手法をそれぞれ C 言語を用いて実装し，実験を行なった．

性能評価においては，ETF 法によるスケジューリング長 (ETFScheduleLength) を 1 とした場合との比を用いた．

$$Speed\_up = \frac{ETFScheduleLenght - ScheduleLength}{ETFScheduleLength}$$

同様に時間評価は，アルゴリズム A 及びアルゴリズム B を適用した場合と，適用しない場合の DL 法，ETF 法，MH 法のスケジューリング時に掛かる CPU 処理時間を比較した．

スケジュール対象とするマシンモデルは，均質の 8 台のプロセッサにより構成されるハイパーキュブアーキテクチャを用いる．

タスクグラフはランダムに生成したものを用いた．タスクグラフをランダムに生成する際のパラメータは，以下の表に示す値を用いた．

Parameter	Value
Anchor Out-Degree	3
Node Weight Range	3-7
Number of Messages Range	1-15
Number of Nodes	50,120,240

ランダムタスクグラフの粒度の値が細粒度，中粒度，粗粒度となる 3 つの場合に対して実験を行なった．粒度の値は，Number of Messages Range の値によって変化させた．Number of Messages Range の値は，細粒度の場合 2-4, 中粒度の場合 3-7，粗粒度の場合 10-15 とした．

図 7 は細粒度における，それぞれの手法の ETF 法にたいする速度向上比である．アルゴリズム A により，ETF 法においては 2%程度，DL 法，MH 法においては 1%程度の改善を得ることができた．アルゴリズム B においては，アルゴリズム A と比較してほとんど改善を得ることができなかった．

図 8 は中粒度における，それぞれの手法の ETF 法にたいする速度向上比である．ルゴリズム A により，ETF 法においては 4%程度，DL 法，MH 法においては 2%程度の改善を得ることができた．アルゴリズム B により，ETF 法においてはアルゴリズム A と比較して 1%程度の改善を得ることができた．

図 9 は粗粒度における，それぞれの手法の ETF 法にたいする速度向上比である．アルゴリズム A により，ETF 法においては 6%程度，DL 法，においては 5% 程度，MH 法においては 7%程度の改善を得ることができた．アルゴリズム B により，ETF 法，MH 法においては，アルゴリズム A と比較して，1% 程度の改善を得ることができた．粗粒度においては，ほとんどの場合において細粒度の場合と比較してよい結果を得ることができた．これは，通信時間が実行時間と比較して大きいため，タスクの複製を行なうための空間がプロセッサ上に多く存在するためと考えられる．

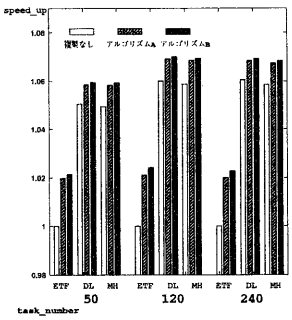


図 7 細粒度の評価結果

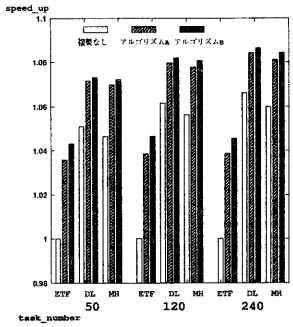


図 8 中粒度の評価結果

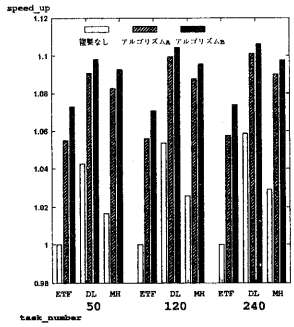


図 9 粗粒度の評価結果

図 10 は，ランダムタスクを ETF 法，アルゴリズム A を適用した ETF 法，アルゴリズム B を適用した ETF

法のそれぞれの手法により Alpha-Compatible 上でスケジューリングした場合の CPU 処理時間である．時間単位はマイクロセカンドである．同様に，図 11，図 12 はそれぞれ DL 法，MH 法における CPU 処理時間である．タスク複製の手法はそれぞれのスケジューリングアルゴリズムの CPU 処理時間をほとんど増加させない．

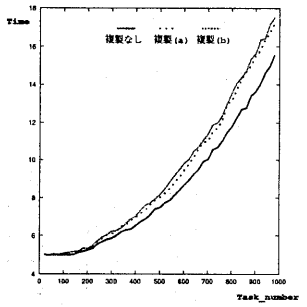


図 10 ETF 法の時間評価結果

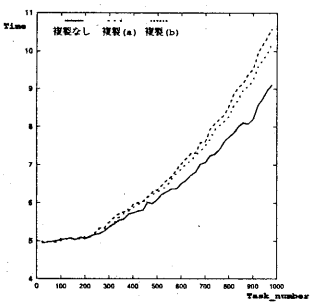


図 11 DL 法の時間評価結果

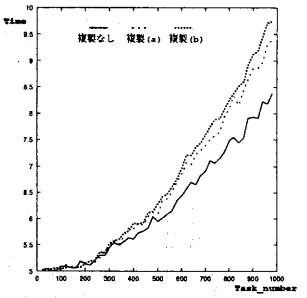


図 11 MH 法の時間評価結果

5 まとめ

本稿では，タスクスケジューリング時に通信遅延を削減するためのタスク複製の手法の提案を行なった．タスク複製の手法として，単純に一回のみタスク複製を行なう手法，アルゴリズム A と，再帰的にタスク複製を行なう手法，アルゴリズム B の 2 つの手法を提案した．ランダムタスクグラフの評価においては，粒度が細粒度，中粒度，粗粒度の場合にたいして実験を行なった．粗粒度の場合，アルゴリズム A においては，ETF 法に対して 6%程度，DL 法にたいして 5%程度，MH 法にたいして 6%程度の改善を得ることができた．アルゴリズム B においては，アルゴリズム A と比較して，MH 法，ETF 法，DL 法において 2%程度の改善を得ることができた．そして，タスク複製の手法の処理時間の評価を行ない，タスク複製のアルゴリズムがほとんどスケジューリングアルゴリズムの CPU 実行処理時間を増加させないことを示した．今後の課題として，通信リンクの衝突を考慮したモデルへの拡張，同期プリミティブ挿入方式の拡張などが挙げられる．

参考文献

[1] J.Y. Colin and P. Chritienne, C.P.M. Scheduling with Small Communication Delays and Task Duplication, Operations Research, vol. 39, no. 4, pp. 680-684, July, 1991.

[2] H.B. Chen, B. Shirazi, K. Kavi, and A.R. Hurson, Static Scheduling Using Linear Clustering with Task Duplication, Proc. of ISCA International Conference on Parallel and Distributed Computing and Systems, pp. 285-290, 1993.

[3] J. Siddhiwala and L.F. Chao, Path-Based Task Replication for Scheduling with Communication Costs, Proc. of the 1995 International Conference on Parallel Processing, vol. II, pp. 186-190, 1995.

[4] S. Darbha and D.P. Agrawal, Optimal Scheduling Algorithm for Distributed-Memory Machines, IEEE Trans. on Parallel and Distributed Systems, vol. 9, no. 1, pp. 87-95, Jan. 1998.

[5] K.K. Kwok and I. Ahmad, Exploiting Duplication to Minimize the Execution Times of Parallel Programs on Message-Passing Systems, Proc. of the sixth IEEE Symposium on Parallel and Distributed Processing, pp. 426-433, 1994.

[6] Jing-Jang Hwang,Yuan-Chiet Chow, "Scheduling Precedunce Graphs in System with Interprocessor Communication Times "Siam J Comput.vol 18,No.2,pp.244-257.

[7] G.C.Sih adn E.A.Lee, "Schedule to account for interprocessor communication within interconnection-constrained processor networks" Proc.Int Conf.parallel Proc.1990

[8] Heshan Dl-Rewini ,T.G.Lewis "Schedule Parallel Program Tasks onto Arbitrary Target Machines"IEEE Transaction on Parallel and Distributed Systems,Vol.5,No.3,March 1994

[9] A-A-Khan,C-L-McCreary "A Comparison of Multiprocessor Scheduling Heuristics" *International Conference on Parallel Processing*,II-243-250(1994).