2003－MPS－47　(17)
2003／12／12

通信遅延を考慮したタスクスケジューリングアルゴリズム

野口智史† ，大下福仁† ，増澤利光†

近年，グリッドなど通信遅延の大きな並列計算環境が用いられている．そこで本報告では，通信遅延の大きな並列計算環境において効率のよいタスクスケジュールを求めるアルゴリズムの考察を行う．まずクロスクラスタリングというクラスタリングのクラスを提案し，このクラス内に 1.5-近似スケジュールを導くクラスタリングが存在することを示す．さらに，タスクの複製を認めないという仮定の下でクロスクラスタリングを行う発見的アルゴリズムを提案し，本アルゴリズムが通信遅延の大きな並列計算環境において，効率的なタスクスケジュールを導くことをシミュレーションによって示す．

A Scheduling Algorithm for Task Graphs with Large Communication Delays

Satoshi Noguti† ，Fukuhito Ooshita† ，Toshimitsu Masuzawa†

For most of the parallel systems available today,the large communication delay is a crucial factor of performance. Thus,we consider an efficient algorithm for scheduling tasks on a parallel system with large communication delays. We first establish a general result on a class of clustering algorithms with specific properties called cross clustering. The class can theoretically be proved to be within a factor of 1.5 from the optimal. Then we propose an algorithm for building cross clustering. This algorithm is assessed by some simulations run on rondom task graphs.

1　はじめに

並列計算環境でアプリケーションを効率的に実行するためには，効率的なタスクスケジュールを求めることが必要である．そのため，効率的なタスクスケジュールを求めるための様々なアルゴリズムが提案されている [1],[2]．一方，グリッドのように通信遅延の大きな並列計算環境が近年よく利用されている．そのため，通信遅延の大きな並列計算環境で効率的なタスクスケジュールを求めるアルゴリズムが重要となる．しかし，これまでに考案されているアルゴリズムの多くは，並列計算機や PC クラスタなど通信遅延が大きくない並列計算環境を対象にしている．そのため，これらのアルゴリズムでは通信遅延の大きな並列計算環境における効率的なタスクスケジュールを求めることができない．また，タスクスケジューリングには複製を許す場合と許さない場合がある．タスクの複製を許すと，より効率的なスケジュールを求められる可能性があるが，システム全体の作業量が増大し，多数のユーザが同時に利用するグリッドのような並列計算環境では望ましくない．そのため，本稿では複製無しのタスクスケジューリングについてのみ考察する．

R.Lèpere ら [3] は，通信遅延の大きな並列計算環境においてタスクの複製なしで効率的なスケジュールを求めるためのクラスタリングアルゴリズムについて考察している．クラスタリングアルゴリズムとは，まずタスクを幾つかのクラスタと呼ばれる集合に分割し (クラスタリング)，それぞれのクラスタをプロセッサに割り当てることでスケジューリングを行う手法のことをいう．R.Lèpere らは，まずコンベックスクラスタリングというクラスタリングのクラスを提案している．そして，このクラス内に 2-近似スケジュール‡ を導くクラスタリングが存在することを証明し，コンベックスクラスタリングを行う発見的アルゴリズムを提案している．また通信遅延が大きいときに，代表的なタスクスケジューリングアルゴリズムである DSC[4] よりも効率的なタスクスケジュールを出力できることを示している．

本報告では，通信遅延が大きな並列計算環境においてタスクの複製なしで効率的にクラスタリングを行うアルゴリズムについて考察する．まずコンベックスクラスタリングの制約を緩めたクロスクラスタリングというクラスを提案する．そして，そのクラス内に 1.5-近似スケジュールを導くクラスタリングが存在することを証明し，クロスクラスタリングを行う発見的アルゴリズムを提案する．最後に，提案

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science,Graduate School of Engineering Science, Osaka University

‡ 2-近似スケジュール: スケジュール長が最適なスケジュールの 2 倍以内で抑えることのできるスケジュール

するアルゴリズムが通信遅延が大きな並列計算環境で効率的なスケジュールを出力できることことをシミュレーションによって示す．

2 諸定義

タスククラスタリングの対象となるアプリケーションは，有向非循環グラフ (DAG) ($G = (V, E)$) で与えられるものとする．ここで V は頂点集合で各頂点がタスクを表し，E は辺集合で各辺がタスク間の依存関係を表す．依存関係とは，例えばタスク v_1 からタスク v_2 に辺があるなら，v_2 は v_1 の結果を用いなければ実行を開始できないことを表す．$x \prec y$ は，タスク x からタスク y へ有向経路が存在することを表し，$x \not\prec y$ は，x から y に有向経路が存在しないことを表す．また $x \not\prec y \bigwedge y \not\prec x$ なとき，x と y は独立であるといい $x \sim y$ と表す．

本報告で考える並列計算環境のモデルについて述べる．タスクの実行時間はタスクの違いや，実行するプロセッサの違いに関わらず 1 単位時間とする．通信遅延については，異なるプロセッサ間での通信遅延は $\rho(\geq 1)$ とする．なお，同一プロセッサ内での通信には時間がかからないものとする．

クラスタリングは，k 個のクラスタに分割する場合 $R = \bigcup_{i=1}^{k}\{C_i\}$ と表記する．C_i は i 番目のクラスタを表しタスク集合からなる．DAG G に対しクラスタリング R を行ったときに，根のタスクが時刻 0 で実行を開始可能とし，依存関係やタスクがどのクラスタに含まれるかを考慮して全てのタスクの実行開始時刻を求める．そのうち，最大の実行開始時刻を $time(R)$ とする．本研究では，できるだけ $time(R)$ が小さくなる R を求めることが目的となる．

あるクラスタリング R において，タスク $t(\in C_i)$ の根からの経路長を $s(t)$，葉までの経路長を $f(t)$ とする (t が根なら $s(t) = 0$，そうでなければ $s(t) = \max\{s(u) + 1 + \alpha | (u,t) \in E, u \in C_i$ なら $\alpha = 0$, そうでなければ $\alpha = \rho$ $\}$,t が葉なら $f(t) = 0$, そうでなければ $\max\{f(v) + \rho + 1 | (t,v) \in E, v \in C_i$ なら $\alpha = 0$, そうでなければ $\alpha = \rho$ $\}$)．そして t の最大経路長を $L(t)(= s(t) + f(t))$ とする ．

3 クロスクラスタリング

[3] において，コンベックスクラスタリングというクラスタリングのクラスが定義されている．コンベックスクラスタリングには，任意の相異なるクラスタの組 C_i, C_j の間に

$\forall a \in C_i, \forall a' \in C_i, \forall b \in C_j, \forall b' \in C_j,$

$a \prec b \Rightarrow b' \not\prec a'$

という制約がある．本節では，この制約を緩めたクラスであるクロスクラスタリングを定義する．

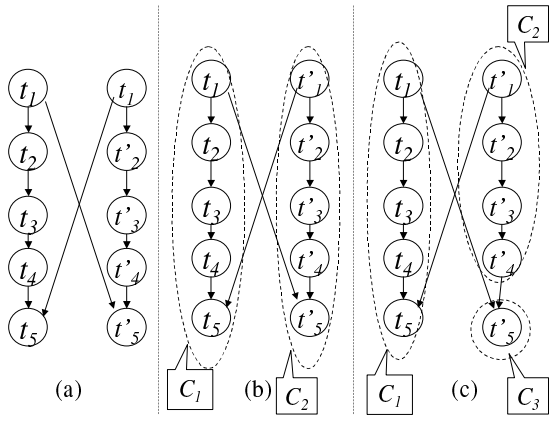

図 1: クロスクラスタリングの有効性

定義 **1** 任意の相異なるクラスタの組 C_i, C_j の間に次の式が成り立つクラスタリングをクロスクラスタリングという．

$\forall x \in C_i, \forall y \in C_j, \forall z \in C_i, x \prec y \Rightarrow y \not\prec z$ □

コンベックスクラスタリングとの違いを見るため，図 1(a) のグラフに対するクラスタリングを考える．クロスクラスタリングでは図 1(b) のようにクラスタリングすることが可能である．例えば $\rho = 3$ とすると，クラスタ C_1 では，t'_1 からの通信を行っている間に t_2, t_3, t_4 を実行でき，通信による待ち時間を生じさせずに t_5 を実行できる．同様に，C_2 でも t_1 からの通信による待ち時間を生じさせずに t'_5 を実行できる．このように双方向 (C_1 から C_2 と C_2 から C_1) に通信を行っても待ち時間が生じないため，全タスクを実行時間 5 で終了するスケジュールを求めることができる．コンベックスクラスタリングでは，図 1(b) のように 2 つのクラスタ間に双方向の通信があると定義に反してしまうため，(c) のような 3 つ以上のクラスタへの分割となる．そのため通信による待ち時間が生じてしまい，実行時間 8 で終了するスケジュールしか求めることができない．

また，クロスクラスタリングは次の特性を持つ．

定理 **2** 任意のタスクグラフ G に対して，1.5-近似スケジュールを導くクロスクラスタリングが存在する．

紙面の都合上，証明は省略する．

4 クロスクラスタリングアルゴリズム

前節において，効率的なスケジュールへと導くクロスクラスタリングが，任意のタスクグラフに対し

アルゴリズム **1:**$CrossCL_\rho(C)$
(入力:クラスタ C，出力:クラスタ集合)

while(複数回繰り返し，最も効率的な分割を選択){
- 独立したタスクの組 $(task_1, task_2)$ を選択
- C を $C_1, C_2, C_T, C_B, C_{Other}$に分割

}
if(分割した方が効率的)
-$R\_C_1 = CrossCL_\rho(C_1)$
-$R\_C_2 = CrossCL_\rho(C_2)$
-$R\_C_T = CrossCL'_\rho(C_T)$
-$R\_C_B = CrossCL'_\rho(C_B)$
-$R\_C_{Other} = CrossCL'_\rho(C_{Other})$
-return $R\_C_1 \bigcup R\_C_2 \bigcup R\_C_T \bigcup R\_C_B \bigcup R\_C_{Other}$
} else{
-return $\{C\}$
}

て存在することを証明した．本節では，そのようなクロスクラスタリングを行うための発見的アルゴリズムを提案する．アルゴリズムの概要をアルゴリズム 1に記す．DAG $G = (V, E)$ をクラスタリングする際には，V をクラスタ C としてアルゴリズムの入力とする．アルゴリズムはクラスタ C に対する再帰的なアルゴリズムとなっている ($CrossCL'$については後述)．以下では，このアルゴリズムの詳細について説明していく．

分割方法　独立したタスクの組 $task_1$,$task_2$を見つけることで，次のように 5つのクラスタに分割する．(タスクの組の発見方法は後述)

$C_1 : task_1 \bigcup Y_1 \bigcup Z_1$
$(Y_1 = \{y_1 \in C | y_1 \prec task_1 \wedge y_1 \not\prec task_2\},$
$Z_1 = \{z_1 \in C | task_1 \prec z_1 \wedge task_2 \not\prec z_1)\})$
$C_2 : task_2 \bigcup Y_2 \bigcup Z_2$
$(Y_2 = \{y_2 \in C | y_2 \prec task_2 \wedge y_2 \not\prec task_1\},$
$Z_2 = \{z_2 \in C | task_2 \prec z_2 \wedge task_1 \not\prec z_2)\}$
$C_T : \{x \in C | x \prec task_1 \wedge x \prec task_2\}$
$C_B : \{x \in C | task_1 \prec x \wedge task_2 \prec x\}$
$C_{other} : \{x \in C | x \sim task_1 \wedge x \sim task_2\}$

このように分けることで図 2のように，Y_1内のタスクから Z_2内のタスクへ，更に Y_2内のタスクから Z_1内のタスクへの依存関係を許容し，C_1と C_2間にクロスクラスタリングの条件を満たす双方向な通信が起こるクラスタリングが行える．また，C_1と C_2は C_{other}との間にも同様の関係がありうる．ただし，上記の分割をそのまま行うと $y \prec x \prec z$ となる $y, x, z (y \in Y_1, x \in C_{Other}, z \in Z_1$または $y \in Y_2, x \in C_{Other}, z \in Z_2)$ の存在を許すため，クロスクラスタリングの定義から外れてしまう．そこでその場合は y を C_Tに入れるか,z を C_Bに入れるかのどちらかを行うことでクロスクラスタリングの定義に沿うようにする．C_1と C_2は大きい方が全体の並列性が高まり望ましいため，$Y_1 \cdot Y_2$内でのクロスクラスタリングに反するタスクの総数と，$Z_1 \cdot Z_2$内でのクロスクラスタリングに反するタスクの総数を比較して，少ない方の移動を行う．

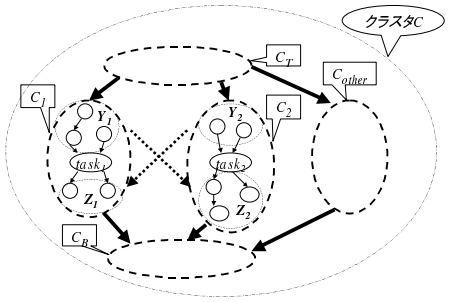

図 2: クラスタリング

独立したタスクの組の選択方法　タスクグラフにおいて $L(t)$ の最大値はスケジュール長の下界となる．そのため，その最大経路に含まれる辺の両端のタスクを優先的に同一クラスタに入れることで効率的なスケジュールが得られることがある．そこで $task_1$は，$L(t)$ が最大値を取るタスクの中からランダムに選択することにする．これにより，そのタスクの最大経路上の辺は C_T, C_1間と C_1, C_B間の高々2回しか通信遅延が生じなくなる．ただし，アルゴリズムを再帰的に実行していく際に毎回 $L(t)$ を計算すると実行時間がかかる．そのため V を一つのクラスタとしたときの $L(t)$ を継続して利用することにする．$task_2$の選択も同様のアイデアに基づき,$task_1$と独立しているタスクのうち，$L(t)$ が最大のものを $task_2$とする．

効率的な分割の判定基準　クラスタリング $R =\{ C_1, C_2, C_T, C_B, C_{Other}\}$ に対し $time(R)$ を求める．これをこのクラスタリングの優先度とする．独立したタスクの組の選択方法がランダムであるため，選択を複数回行い最小の優先度をとる選択を採用する．なお，実行開始時間は単純に根から幅優先探索によって求めていくこととする．

また，その分割が，分割前より優れているかの判定にもこの値を利用する．具体的には，複数回の試行によって得られたクラスタリングの優先度が，元のクラスタ C に含まれるタスク数より大きければ，逐次に実行したほうが効率的であると考え C に対する分割は行わない．分割を行う場合は，分割されたクラスタを入力とし再び $CrossCL$ を実行する．ただし C_1, C_2以外のクラスタは，互いに依存関係のない複数のクラスタに分割できることがある．そのとき $CrossCL'$はまず分割を行い，それぞれのクラスタに対して $CrossCL$ を実行する．

5 実験と評価

実験 本実験では，提案アルゴリズム (Cross) とコンベックスクラスタリングアルゴリズム (Convex) との比較を行う．比較は $time(R)$ の値で行い，DAG は [5] で提供されている物を今回のモデルに合わせて用いる．[5] には，タスク数ごとに 180 種類ずつのランダムなタスクグラフが提供されている．比較を行う両者は共にランダムアルゴリズムである．そこで，各グラフに対してそれぞれ 10 回の実行を行い，その平均値と最小値を用いて比較を行う．また独立したタスクの選択回数も両アルゴリズム共に 10 回行うこととする．

表 1 はタスク数とρの変化による比較を行ったものである．縦軸がタスク数，横軸がρの値の変化を表している．比較はそれぞれの項目に対して全グラフの $time(R)$ の合計値で行う．項目ごとに 2 種類の値を記しているが，上段は (Cross での最小値/Convex での最小値), 下段は (Cross での平均値/Convex の最小値) を表す．

Min/Min	1.5	3.0	5.0	8.0	10.0	14.0
102	0.944	0.936	0.932	0.939	0.935	0.951
302	0.925	0.922	0.914	0.904	0.895	0.910
752	0.923	0.918	0.902	0.884	0.880	0.870
1252	0.924	0.915	0.9000	0.876	0.871	0.859
Avg/Min	1.5	3.0	5.0	8.0	10.0	14.0
102	0.989	0.990	0.987	0.994	0.981	0.980
302	0.958	0.959	0.954	0.942	0.936	0.944
752	0.948	0.945	0.930	0.913	0.907	0.895
1252	0.941	0.936	0.919	0.897	0.891	0.877

表 **1** 実験結果

評価 最小値同士の比較結果より，Cross が Convex よりも，効率的なスケジュールを出力できることがわかる．Cross の平均値と Convex の最小値の比較結果より，Cross はランダムな実行による誤差が生じても，Convex よりも効率的なスケジュールを求める事が期待できる．また，表には記していないが Cross の実行時間は Convex の実行時間の 2 倍で充分抑えることができる．以上の結果から，通信遅延が大きい時に効率的なタスクスケジュールを出力できる Convex 以上に，Cross は効率的なタスクスケジュールを出力できることがいえる．

6 むすび

本報告では，クロスクラスタリングというクラスタリングのクラスを提案し，このクラス内に 1.5-近似スケジュールが存在することを証明した．また，クロスクラスタリングを行う発見的アルゴリズムを提案し，通信遅延が大きなときに効率的なクラスタリングが行えることをシミュレーションにより示した．今回は DSC アルゴリズムより，通信遅延が大きいときに効率的なスケジュールを出力できるコンベックスアルゴリズムとの比較しか行わなかったが，DSC アルゴリズムを改善したアルゴリズムは他にも幾つか提案されている [1]．今後はそういったアルゴリズムとの比較を行っていく．

参考文献

[1] Jing-Chiou Liou and Michael A.Palis ,"A New Heuristic for Scheduling Parallel Programs on Multiprocessor",IEEE Intl. Conf. on Parallel Architectures and Compilation Techniques,1998

[2] Yu-Kwong Kwok and Ishfaq Ahmad,"Dynamic Critical-Path Scheduling:An Effective Technique for Allocating Task Graphs to Multiprocessors" IEEE Transactions on Parallel and Distributed Systems,1996

[3] R.Lèpere and D.Trystram "A new clustering algorithm for scheduling task graphs with lare communication delays" ,International Parallel and Distributed Processing Symposium,2002

[4] A. Gerasoulis and Tao Yang. "DSC:Scheduling parallel tasks on an unbounded number of processors",IEEE Transactions on Parallel and Distributed Systems,1994

[5] 早稲田大学 笠原研究室,"Standard Task Graph Set",http://www.kasahara.elec.waseda.ac.jp/schedule/