順序制約のあるタスクのスケジューリング方法について

2H－4

小池祐二　塩田佳明　渋沢進

茨城大学工学部情報工学科

1　はじめに

複数のタスクからなるジョブを並列に短時間で処理するには，タスクの適切なスケジューリングが必要である．最適スケジューリング問題はNP完全であるが，多項式時間で実用的なスケジュールを求めるアルゴリズムも開発されている[1][2]．しかし，並列処理において常に問題になるのがプロセッサ間通信による待ち時間である．

そこで本研究では，通信のオーバヘッドを考慮したスケジューリング方法を考える．ここで考えたアルゴリズムは全タスクの実行時間の短縮を目的とし，最長経路の長さによってタスクに優先順位をつけ，プロセッサへ割り当てていく．プロセッサネットワークとしてハイパーキューブ結合を想定したときの計算量は$O(n^2N\log N)$である．

2　スケジューリング問題のモデル

順序制約のあるn個のタスクをスケジュールする問題を考える．データ依存関係とそれに起因する先行制約はタスクグラフで表される（図1）．

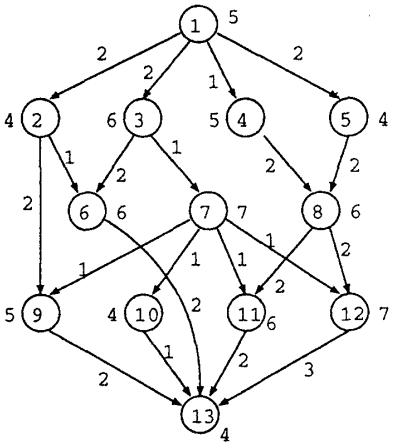

図1: タスクグラフ

図は，タスクiからタスクjに有向辺があるとき，jはiの処理が終了すれば実行可能となることを意味する．各ノードはタスクを表し，ノード内の数はタスク番号を，ノードの横の数はタスクの実行時間を，辺の横の数は通信量を表す．

A Scheduling Method for Ordered Tasks
Yuji Koike, Yoshiaki Shiota, Susumu Shibusawa
Ibaraki University, Hitachi, Ibaraki 316, Japan

プロセッサaからbへ通信量cのデータを送るのにかかる時間は，プロセッサ間の距離をd_{ab}として，$c\cdot d_{ab}$時間とする．ここで，プロセッサ間距離とはハイパーキューブでいういわゆるハミング距離のことである．従って，プロセッサ内タスク間通信にかかる時間は0となる．

3　スケジューリングアルゴリズム

ここで述べるアルゴリズムは終端タスクまでの最長経路の長さが大きいタスクから優先的にプロセッサへ配置していく．またプロセッサ間の距離を考慮し，データ依存関係にあるタスクどうしは同じプロセッサに配置することでプロセッサ間通信による待ち時間が少なくなるようにした．

スケジューリングアルゴリズムは大きく分けて，最長経路の計算とタスク配置のふたつの段階から成る．

3.1　最長経路の計算

各ノードから終端ノード（子を持たないノード）への最長経路の長さを次のように計算する．各ノードvについてd[v]をvからの最長経路の長さ，t[v]をvの実行時間とする．各ノードvとその親pについてc[p,v]をpからvへの通信量とする．

1. 各ノードvが持つ子の数child[v]を計算する．
2. 全てのd[v]を0に初期化する．
3. child[v]=0を満たすノードvは各親pにd[v]+t[v]+c[p,v]を渡し，親pは受けとった値とd[p]とを比較し大きい方をd[p]とする．
4. child[p]=child[p]−1とし，新たにchild[v]=0となるvがあれば3.以降を繰り返し，なければ終了する．

3.2　タスク配置

タスク配置は，d[v]の大きいタスクvから順に行なう．

1. 各ノードについて親の数parent[v]を計算する．
2. parent[v]=0を満たすvの中でd[v]が最大のタスクをuとする．d[v]が同じものがある場合はc[$\cdot,v$]の大きい方を優先する．
3. タスクuを最も早く実行開始できるプロセッサPを求める．
4. uをPに割り当てる．

5. u の各子 c について parent[c]=parent[c]−1 とする.
6. u を削除し, 未配置のタスクがあれば 2. 以降を繰り返し, なければ終了する.

プロセッサ Pを求めるところでは, ネットワークアーキテクチャに応じてプロセッサ間の距離を計算し, データ転送による待ち時間を考慮してタスクの実行開始可能時刻を求める. この時刻が最も早いプロセッサを Pとする. プロセッサ間距離を計算するモジュールを変更すれば, 種々のネットワーク（ハイパーキューブ, 木, 格子結合）に対応できる.

4 評価

4.1 アルゴリズムの計算量

ノードの数を n, 辺の数を m, プロセッサの数を N とする. グラフのデータを保持するために図 2, 図 3 に示すデータ構造を用いる.

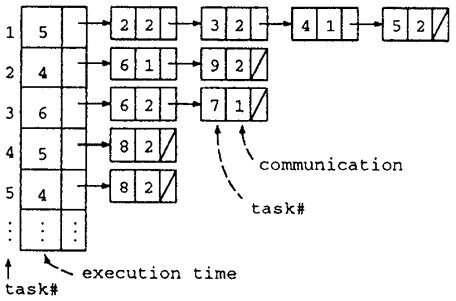

図 2: グラフのデータ構造

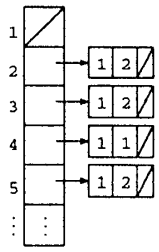

図 3: 順序制約のデータ構造

ここではハイパーキューブにタスク配置する場合の計算量を調べる. 各ノードの子の数を計算するには, 図 2 でリンクをたどって構造体の数を数えればいいので $O(m)$ で計算できる. 最長経路の重みを求めるには, 図 3 のデータ構造を用いて親のノードへデータを送る操作を辺の数だけ繰り返せばいいので, $O(m)$ で計算できる. 各ノードの親の数を求めるには, 図 3 を用いて子の数の計算と同様に $O(m)$ で計算できる.

タスク配置のところは n 回繰り返されるループである. これは 1 回のループでひとつのタスクをスケジュールするからである. ループの中では, d[v] が最大となるタスクを求めるのに $O(n)$, Pを求めるのに $O(nN\log N)$, parent[ ] の更新に $O(n)$ の計算が必要であるから, ループ全体として $O(n^2N\log N)$ の計算が必要である. $n^2 > m$ なので, アルゴリズム全体としては $O(n^2N\log N)$ である.

この中の $O(\log N)$ はハイパーキューブでのプロセッサ間距離を求めるのにかかる計算量である. 木結合の場合も同じく $O(\log N)$ の計算が必要であるが, 格子結合の場合には $O(1)$ の計算で済む.

4.2 生成されるスケジュール

このアルゴリズムによって求めた図 1 のタスク群のスケジュールを図 4 に示す. これはプロセッサネットワークとしてハイパーキューブを想定したものである. 1 プロセッサの場合は 69 単位時間で処理は終了し, 2 プロセッサのときは 42 単位時間で, 4 プロセッサのときは図のように 33 単位時間で終了する. 2 プロセッサと 4 プロセッサの速度向上比はそれぞれ 1.6 と 2.1 で, プロセッサ利用率はそれぞれ 0.82 と 0.52 である. この例の場合, 2 から 4 へプロセッサ数を増やすとプロセッサ利用率が低下することが分かる.

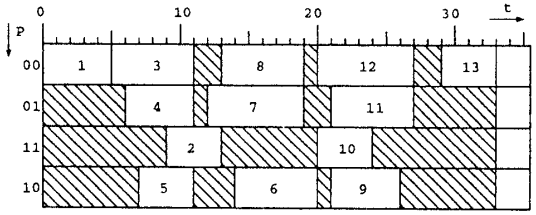

図 4: 生成されたスケジュール

5 おわりに

このスケジューリング方法では, あらかじめ全てのタスクの実行時間とタスク間で渡されるデータの量が分かっている必要がある. しかし実際には, タスクの実行時間を実行前に正確に知るのは難しい. スケジューリング問題には, 個々のタスクの実行時間が少し変わっただけでスケジュール全体が大きく変わり, 悪化してしまう場合がある [3]. ここで考えたスケジューリング方法では, 最長経路を計算するなどしてそのような変化がスケジュールに与える影響を少なくしている. また, プロセッサ間の通信で渋滞が起こり, データの到着が遅れる可能性がある. このあたりを考慮し, 第 2 節で述べたモデルが現実の計算機を正確に反映するように拡張することが今後の課題である.

参考文献

[1] 富田眞治, 末吉敏則 : 並列処理マシン, オーム社, p.236 (1989). [2] 坂井修一 : 並列計算機におけるスケジューリングと負荷分散, 情報処理, Vol.27, No.9, pp.1031-1038 (1986). [3] 秋山仁, R.L.Graham : 離散数学入門, 朝倉書店, p.198 (1993).