数理モデル化と問題解決 43－15
（２００３．３．４）

グリッド環境における計算ノードの故障を考慮した独立タスクのスケジューリングアルゴリズム

譚　　林[†]　藤本典幸[††]　萩原兼一[††]

様々な計算機と異種のネットワークからなるグリッド環境では計算ノードの故障，計算ノードの速度の変化およびネットワークのトラフィックの変化により，スケジューリングしたタスクの計算結果が返されるのが遅れる場合がある．本研究では，グリッド環境において，計算ノードの故障を考慮し，タスク複製を用いた独立タスクのスケジューリング手法を提案する．グリッドの模擬環境を用いて評価実験を行ったところ，計算ノードの故障がある場合，スケジューリングした順序でタスクを複製するとスケジュールの実行時間が短くなることがわかった．

Scheduling Independent Tasks onto a Computational Grid in the Presence of Processor Failures

Lin Tan[†] ,Noriyuki Fujimoto[††] and Kenichi Hagihara [††]

Since grid environment is composed of various kinds of computers and networks, task occasionally results in a delayed return or even no return due to processor failures and the variation of computer speed and communication delay.In this report,we give an independent task scheduling algorithm using task replication to discard faulty computers.Some experimental results show that in the presence of processor failures tasks should be replicated in the order where the tasks were scheduled.

1. はじめに

近年，ネットワークの高速化およびハードウェアの発展により，インターネットを通じて数多くの計算機を計算ノードとして結び，各計算機の余剰計算能力を利用して，分散処理を行うグリッド計算についての研究が盛んになってきた．莫大な計算量を必要とする問題でもグリッドを用いれば，インターネット上の何万何十万台の計算機が同時に計算を行うことにより，短時間で解ける可能性がある．

グリッドの特徴の一つは計算ノードが企業や大学のサーバマシンや個人のパソコンなど多種多様なことである．原則として，計算機の本来の利用者の通常の使用に影響してはいけないため，各計算機の使用状況によって，グリッドの計算ノードとして使える計算能力が異なってくる．そのため，グリッドでは各計算ノードの速度の差が大きい．またインターネットによりデータの転送を行うため，通信遅延が非常に大きい．そのため，粗粒度の独立タスク集合になる計算問題がグリッドに対して一番適切である．そのような計算問題の例としては，データマイニング，パラメータ検索，モンテカルロ計算などの問題がある．これらの問題はタスクが完全に独立し，タスク間の依存関係がなく，タスク間の通信がないという特徴を持つ．また１タスクの普通の計算機上での実行時間が数時間程度であるのに対して，タスクの入出力データの転送時間は数分程度なので，通信遅延は無視できる．

本研究ではホモジニアスな並列計算機に対するワークキューという古典的な独立タスクのスケジューリングアルゴリズムを拡張し，全体の計算効率に悪影響を与えるタスクに注目し，それらのタスクを他のマシンに複製する手法を提案する．特に，計算ノードの故障がある場合のタスク複製の順番について検討する．さらに，Simgrid[2] というツールでグリッドの模擬環境を作り，複製回数の効率への影響と，計算ノードの故障がある場合の二種類の複製の順番について評価を行う．

† 大阪大学大学院基礎工学研究科情報数理系専攻
Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University
†† 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 Information Science and Technology, Osaka University

2. ワークキューアルゴリズム

タスクスケジューリングアルゴリズムはスケジュール生成のタイミングという点からみると，静的と動的の二種類に分類できる．タスクを実行する前にスケジュールを全て生成し，そして，生成したスケジュールに従い，タスクを計算ノードに配って，実行を行う場合を静的と言う．一方，タスクを実行しながら，スケジュールを生成する場合を動的と言う．グリッド環境では各計算ノードのスピードが時間につれ変化し，予測しにくいため，本研究では動的なスケジューリングアルゴリズムを使う．ワークキューアルゴリズム (以下，WQ アルゴリズム) は古典的な動的タスクスケジューリングアルゴリズムである．

WQ アルゴリズムはまず一つのサーバでタスクのキューを作り，タスクを全てキューに入れる．次に，キューからタスクを順番に取り出し，各計算ノードに一つずつ配る．そして，計算ノードがタスクの実行を行う．もし一つの計算ノードでのタスクの実行が終わって，結果がサーバに返されたら，キューからタスクを取り出し，再びその計算ノードに配り，実行させる．このように，キューの中のタスクを全部実行完了されるまで繰り返す．WQ アルゴリズムの特徴の一つはタスクのサイズやマシンの速度などの情報を使わずに動的にスケジュールを作ることである．WQ アルゴリズムの利点としては，速度が速いマシンに多くのタスクを配って実行させる傾向があることが挙げられる．しかし，計算ノードの速度の差が大きいグリッド環境では，もしキューの中に最後に残ったタスクを遅い計算ノードに配ったら，他のタスクの実行が終わっても，そのタスクの実行が終わらないと計算問題の実行が完了できないため，全体の実行時間が長くなり，効率が悪くなる欠点がある．また，実際の状況を考えると，タスクを実行中に計算ノードの故障が発生する場合では，故障した計算ノードが回復しないと，計算問題の完了ができないおそれがある．

3. 提案アルゴリズム

本研究では WQ アルゴリズムの効率の向上を目指し，またグリッド環境で計算ノードの故障を考慮し，タスク複製を用いた独立タスクのスケジューリングアルゴリズムを提案する．

3.1 タスクの複製

提案アルゴリズムは WQ アルゴリズムを拡張し，タスクキューに残った最後のいくつかのタスクの複製を行う．そのタスクの複製により，WQ アルゴリズムの最後に遅い計算ノードに配ったら効率が悪いという欠点を避ける．例えば，タスクが t 個，計算ノードが m 台の場合では，まず，WQ アルゴリズムと同じように，キューを作って，タスクを計算ノードにスケジューリングする．次に，タスクキューの最後のタスクを計算ノードに配った時に着目すると，全体から見ると m 台の計算ノードがタスクを実行しているため，完了していないタスクの数は計算ノードの台数とおなじ m 個である．そして，もし一台の計算ノードが配られたタスクの実行を完了したら，タスクキューの中に，スケジューリングしていないタスクがもう残っていないため，WQ アルゴリズムではスケジューリングせず，最後まで待つしかない．これに対して，提案アルゴリズムではまだ完了していないタスクを複製し，その手が空いている計算ノードに配る．このように，全部のタスクの実行を完了するまで手が空いている計算ノードがあったら複製を行う．ここで，同じタスクが多数の計算ノードで同時に実行されている時，一つの計算ノードでの実行が完了したら，別のマシンの同じタスクを殺す．

実行時間の面からみると，もし複製したタスクが元のタスクより早く完了したら，全体的に実行時間が短くなる．その一方，元のタスクが先に完了した場合，実行時間が複製しない場合と同じとなる．

3.2 複製の回数

複製の回数とは，一つのタスクに対して，許される複製の最大数である．例えば，一回複製は複製を高々 1 回しか行わない．キューの中に残っているタスクを全部一回複製した時，たとえ手が空いている計算ノードがあっても，複製をしないでそのまま全タスクの実行完了まで待つ．二回複製は，キューに残っているタスクを全部一回複製したうえで，手が空いている計算ノードがあったら，複製したタスクをもう一回複製し，その計算ノードに配る．このように，残っているタスクを全部二回複製されるまで複製を許す．つまり，一つのタスクに対して，最大三つの計算ノードでの同時実行を許す．三回複製，四回複製も同様に定義する．提案アルゴリズムは複製の回数を制限しない．ここで，複製の回数が多くなると，タスクが速い計算ノードに配る可能性が大きくなり，完了までの時間が短くなるが，複製したタスクによる無駄な計算量も多くなる．

3.3 複製の順番

複製の順番として正順と逆順の二種類を考える．正順ではタスクを配る順番と同じ順で複製する．つまり先に配ったタスクを先に複製する．逆順では配る順番と逆の順で複製する．つまり最後に配ったタスクを先

に複製する．

計算ノードの故障を考慮しない場合では，正順と逆順で複製の効果はあまりかわらない．ここで，故障とはマシンが一旦停止したら，最後まで回復しないことである．計算ノードの故障を考慮する場合，正順の方が効率がいいと考えられる．理由は，故障した計算ノードに割り当てられたタスクは必ず他の計算ノードに再割り当てする必要があるのに対して，その他のタスクには必ずしも複製の効果があるとは限らないからである．

3.4 関連研究とのちがい

Paranhos ら [3] と Anderson ら [1] もグリッド上での独立タスクのスケジューリングにタスク複製を用いている．Paranhos らは計算ノードの故障を考慮しておらず，また複製の回数を最大 4 回に制限している．複製の順序などの詳細は文献 [1] には示されていない．Anderson らは各タスクに対して複製の機会を与える．これに対して我々は実行完了が最後から $(m-1)$ 番目以降のタスクに対してのみ複製の機会を与える．

4. 評価実験

本実験では，まず提案アルゴリズムと古典的な WQ アルゴリズムの性能を比較する．次に，タスク複製の回数の増加につれて，実行時間が短くなり，効率が向上することを示す．最後に，計算ノードの故障がある場合の二種類の複製順序の効果を比較する．

4.1 実験環境

C 言語の API として無償提供される Simgrid を使って実験環境を作る．Simgrid はグリッド環境上でのタスクスケジューリングを研究するためのツールである．Simgrid は一台のマシンでグリッド環境を模擬できる．そのため，実験を簡単に何回も実行し，アルゴリズムを評価できる．図 1 と図 2 と図 6 は 100 回の模擬の平均値である．図 3 と図 4 は 10 回の平均値である．

4.2 複製回数と対 WQ 速度比

提案アルゴリズムの WQ アルゴリズムに対する速度向上の比率を対 WQ 速度比と呼ぶ．計算ノードの速度のちがいによる対 WQ 速度比のちがいを**図 1** に示す．実験の条件は以下の通りである．計算ノードが 100 台，各計算ノードの元の速度が 1 から 2，1 から 4，1 から 8，1 から 16 までそれぞれの範囲でランダムできまる．実際に実行するとき，元の速度に対して，各計算ノードのプロセッサの利用率が時間につれて 0 から 100%までランダムに変化する．タスク総数は 1000，サイズは全て 100 とする．ここで，サイズ 100 のタスクを速度 1 の計算ノードで実行すれば，100 単位時間

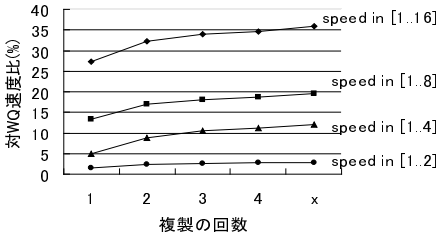

図 1 複製の回数と対 WQ 速度比の関係
(計算ノードの速度によるちがい)

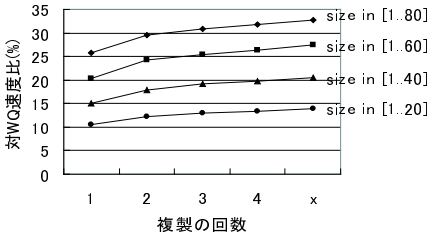

図 2 複製の回数と対 WQ 速度比の関係
(タスクサイズによるちがい)

かかる．図 1 の縦軸は対 WQ 速度比，横軸は複製の回数を表す．x は複製回数を制限しないことを意味する．この図から，複製の回数が増えると，対 WQ 速度比があがることがわかる．また計算ノードの速度の差が大きい場合，対 WQ 速度比が大きくなる．**図 2** は計算ノードの台数と速度を固定し，タスクのサイズがそれぞれ 1 から 20，1 から 40，1 から 60，1 から 80 までランダムで決まる場合の対 WQ 速度比を示す．この図から，タスクサイズの差が大きい場合，対 WQ 速度比が大きくなることがわかる．

図 3 はタスクの数とサイズを固定する場合，つまり総計算量が一定の場合の，計算ノードの台数のちがいによる対 WQ 速度比のちがいを示す．サイズ 100 のタスクが 10000 個，計算ノードの速度が 1 から 16 まで，台数が 100，200，300，400 とする．結果は対 WQ 速度比の最大値が 24．75%，最小値が 0 である．計算ノードの台数が増えるにつれて，対 WQ 速度比が上がる．これは計算ノードの台数が増えると，全体の計算時間が減少し，複製による短縮の時間が相対的に大きくなり，対 WQ 速度比が上がるためと考えられる．**図 4** は計算ノードの台数を固定する場合の，タスク数のちがいによる対 WQ 速度比のちがいを示す．計算ノードが 100 台，速度が 1 から 16 まで，タスクのサイズが 100，数が 10000，20000，30000，40000 とする．対 WQ 速度比の最大値は 6．66%，最小値は 0．05%である．図 4 から，計算量が増えるにつれ

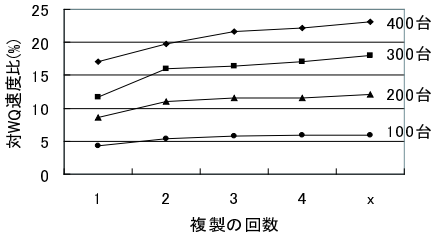

図 3 複製の回数と対 WQ 速度比の関係
(計算ノード数によるちがい)

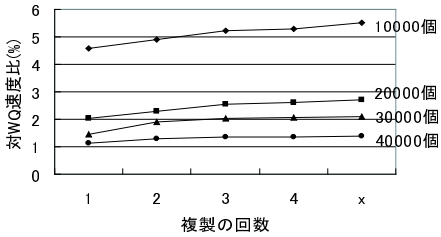

図 4 複製の回数と対 WQ 速度比の関係
(タスク総数によるちがい)

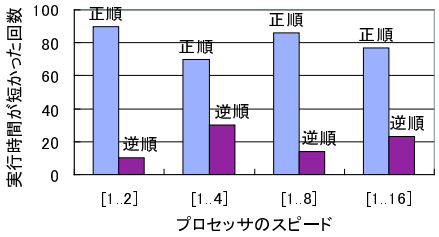

図 5 正順複製と逆順複製による実行時間のちがい

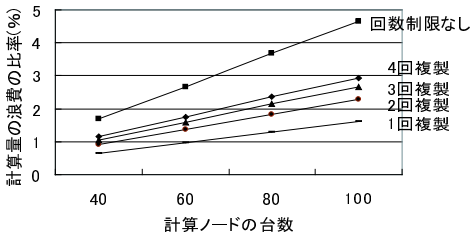

図 6 複製による CPU の計算量の浪費

て，対 WQ 速度比が減る．理由は図 3 と同じと考えられる．

4.3 複製順序の効果

サイズ 100 のタスクが 1000 個，計算ノードの台数が 100 台，速度がそれぞれ 1 から 2，1 から 4，1 から 8，1 から 16，計算ノードの故障率が 10%の場合を考える，ここで，故障率が 10%とは，最初のタスク実行開始から最後のタスクの実行完了までの間に，10%の計算ノードが故障することである．故障する計算ノードと故障発生時刻はランダムに決める．このような場合の正順複製と逆順複製のスケジュールの実行時間の比較を**図 5** に示す．図 5 は 100 回比較して実行時間が短かった回数である．この図から正順で複製すれば効率がいいと言える．

複製をしない場合の総計算量に対する複製タスクの計算量の比率を**図 6** に示す．実験条件は，サイズ 1000 のタスクが 2000 個，計算ノードの速度が 1 から 16 まで，台数が 40，60，80，100 である．この場合，最大の浪費量は元の総計算量の 5%未満である．

5. ま と め

本研究では古典的な WQ アルゴリズムを拡張し，グリッド環境で計算ノードの故障を考慮し，粗粒度の独立タスク集合の計算問題に対して，タスクの複製を用いたスケジューリングアルゴリズムを提案した．また，グリッドの模擬環境でタスクの複製により性能が向上することを確認した．さらに，計算ノードの故障がある場合，配る順番と同じ順でタスクを複製すると効率がいいことがわかった．

謝辞 本研究の一部は，日本学術振興会未来開拓学術研究推進事業（JSPS-RFTF99I00903），科学研究費補助金基盤研究 (C)(2)(14580374)，NEC ネットワークス開発研究所および栢森情報科学振興財団の補助による．

参 考 文 献

1) Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. : “SETI@home: An Experiment in Public-Resource Computing”, *Communications of the ACM*, Vol. 45, No. 11, pp. 56–61, November (2002)
2) Casanova, H. : “Simgrid: A Toolkit for the Simulation of Application Scheduling”, IEEE International Symposium on Cluster Computing and the Grid (CCGrid2001), pp. 430-437, (2001).
3) Paranhos, D., Cirne, W., and Brasileiro, F. : “Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids”, http://www.lsd.dsc.ufpb.br/papers/WQReplication.pdf, Submitted for publication, October (2002).