並列分枝限定法における耐故障アルゴリズムの評価

川 上 健 太[†]　合 田 憲 人[†]

本稿では，マスタ・ワーカモデルを用いた並列分枝限定法における耐故障性を実現するためのアルゴリズムの性能を議論する．計算に耐故障性を持たせるための手法としては，タスクの多重化による手法が提案されている．しかし，タスクを多重化させた場合，実行時間の増大が大きな問題となる．そのため，本稿では実行時間の増大を抑えるために，マスタの不要な処理を中断させる機構の評価を行うとともに，新たにワーカの監視による手法を提案し，既存の手法との比較を行う．

1. はじめに

計算機クラスタやグリッドで大規模な処理を行った場合，計算ノードで障害が発生すると計算全体が停止してしまう恐れがある．そのため，並列計算における耐故障性は非常に重要なテーマである．本稿では，マスタ・ワーカモデルを用いた並列分枝限定法における耐故障性を実現するための手法について議論する．

並列分枝限定法において耐故障性を付加するための手法として，タスクを多重実行する方法が提案されている[1)]．しかし，この手法を使用した場合，多重度（同じ処理を実行させるワーカの台数）が増大するに従って実行時間が増大してしまう問題がある．そのため，本稿ではこの実行時間のオーバヘッドを削減するために，マスタの不要な処理を中断させる機構を実装し，PC クラスタ上での実行時間を計測する事によりこの機構の有効性を確認した．また，タスクを多重化せずに耐故障性を実現する方法として，ワーカの状態を監視することにより障害に対応するアルゴリズムを提案し，既存のアルゴリズムとの比較を行う．

2. 並列分枝限定法とタスクの優先度

分枝限定法では，親問題の解の探索範囲を分割して子問題を生成し，各子問題ごとに目的関数の下界値，上界値，解を計算するという操作を繰り返すことにより最適解を探索する．マスタ・ワーカ方式で分枝限定法を並列実行する場合，マスタが解の探索範囲である探索ツリーを管理し，探索ツリー上の子問題を複数のワーカに割り当てることにより並列処理を実現する．

マスタがワーカにタスクを割り当てる方法の一つに最良下界値優先探索がある．この手法は，探索ツリー上の各節点に対して下界値を優先度として設定し，優先度の高い節点から順に処理を行うというものである．下界値を優先度として使用するのは，下界値が小さい節点の方が最適解の存在する可能性が高いためである．本研究でも優先度として下界値を使用する．

3. 耐故障アルゴリズム

本節では，本稿で議論する２つの耐故障アルゴリズムについて説明する．これらは，タスクの多重化による手法と，ワーカの監視による手法からなる．

3.1 タスクの多重化による手法

タスクの多重化による手法では，マスタが同一のタスクを複数のワーカに多重に割り当てる事により，任意のワーカに障害が発生した場合でもタスクの実行を他ワーカ上で続ける．また，任意のワーカ上でタスクが正常終了した場合は，同じタスクを実行している他のワーカ上の処理は不要となるためそれらを中断する．即ち，マスタは以下の様な流れで処理を行う．

(1) マスタは事前に設定した多重度のリストに従って，同一のタスクを複数のワーカに割り当てる．
(2) 任意のワーカ上でタスクが正常終了した場合は，同一のタスクを実行している他のワーカの処理を中断させる．

3.2 ワーカの監視による手法

ワーカの監視による手法では，タスクを多重化しない代わりにマスタがワーカの処理状況を常に監視し，ワーカの計算状況が全く変化しない場合は，それらに障害が発生していると推測する．本稿では，その様な障害が推測されるワーカを，障害が疑われるという意味でサスペクト (suspect：容疑者) と呼ぶ事にする．以降では，ワーカタスクの処理は二等分する事が可能であると仮定し，ワーカが前半の処理を行っている状態を状態 1，後半の処理を行っている状態を状態 2 と呼ぶ．マスタ及びワーカは以下の様な流れで処理を行う．

† 東京工業大学
Tokyo Institute of Technology

(1) マスタがアイドルワーカにタスクを割り当てる．
(2) ワーカがタスクの前半部分を実行する（状態1）．
(3) ワーカはタスクの前半部分が終了した時点で，それをマスタに伝える．
(4) ワーカがタスクの後半部分を実行する（状態2）．
(5) マスタは任意のワーカ上でのタスク処理が正常終了した場合，そのワーカ上でのタスク実行中に状態が全く変化しなかった他のワーカをサスペクトとする．
(6) マスタは，サスペクトとなったワーカに割り当てられていたタスクを，他のアイドルワーカに割り当てる．

この操作によって，あるワーカが一つのタスクを終了した時間で，その半分の処理すら完了していないワーカはサスペクトと見なされる．

4. アルゴリズムの評価

4.1 不要な処理の中断による実行時間の変化

タスクの多重化による手法の実行時間の評価を行うために，PC クラスタを用いて以下の実験を行った．PC4 台で構成した PC クラスタに，3.1 節の手法をマスタワーカ型の並列プログラミング環境を提供する Ninf[2] を用いて実装した．各計算ノードについて，CPU は Dual Intel Xeon 2.4GHz，メモリは 512MB，OS には Red Hat Linux 7.1(Kernel 2.4.7-10) である．以上の環境で，マスタ 1 台，ワーカ 8 台を用い，BMI 固有値問題アプリケーション[3] を使用して性能評価を行った．

図 1 に不要な処理を中断する場合と，中断しない場合の実行時間の遷移を示す．本実験では，下界値が最も小さい節点のタスクのみを多重化した．グラフの横軸は，その多重度を示す．両者とも，多重度が増加するに従って実行時間は増加した．これは先程述べた様に，多重度が増加する程，ワーカ全体の処理能力が低下するためである．しかし，図 1 より不要なタスクを中断することにより，実行時間の増加を小さく抑えられることがわかる．表 1 に不要な処理を中断した事による実行時間の削減率を示す．この表から，多重度が増加するに従って，不要な処理の中断による実行時間の削減率も増大していく事がわかる．これは，多重度の増加は冗長な処理の増大を招くため，不要な処理の削減によって得られる効果も大きくなるためである．

4.2 ワーカの監視による耐故障性の実現

この手法はサスペクトが検出された場合にのみタスクの多重化を行うため，先程のタスクの多重化と比較して，複数ワーカ上での冗長処理は大幅に削減されると期待される．また，サスペクトは障害が発生した場合だけでなく，ワーカ同士の処理能力の差が 2 倍以上である場合に検出されるため，性能の低いワーカ上の処理がボトルネックとなることを防ぐことにも用いられる．例えば，性能の劣ったワーカを処理から切り離したい場合は，一定の回数以上サスペクトとなったワーカには一切タスクを投げないようにすればよい．逆に，性能差が 2 倍のワーカが多数あり，それらがサスペクトとなるのを防ぎたい場合は，それを判断するタイミングを変えればよい．

この手法は，ワーカのタスクの処理時間に閾値を設けて障害を検出する方法と類似している．しかし，閾値を設けた場合，ある時点で全てのワーカの処理能力が低下した場合に全てのワーカが障害と判断されてしまう可能性がある．一方，本手法では各ワーカの処理時間を比較して障害を検出するため，その様な状況により柔軟に対応する事が出来ると考えられる．

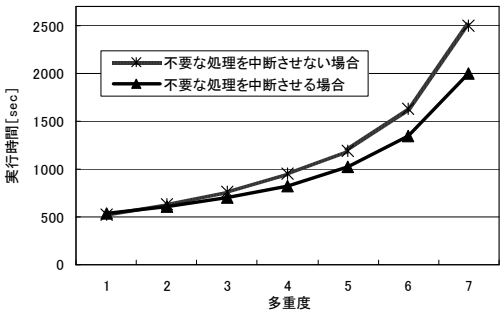

図 1 不要な処理の中断による実行時間の変化（横軸は下界値が最小のタスクの多重度を意味する）

表 1 不要な処理の中断による実行時間の削減率
削減率 [%]=(中断無しの実行時間−中断有りの実行時間) ／ 中断無しの実行時間

多重度	1	2	3	4	5	6	7
削減率 [%]	-1.0	4.6	9.0	13.8	14.5	17.7	19.7

参 考 文 献

1) 久保田和人, 仲瀬明彦:耐故障／耐高負荷を考慮した並列分枝限定法と基本性能の評価，情報処理学会論文誌, Vol.45, No. SIG11(ACS 7), pp. 171-181(2004).
2) Ninf: A Global Computing Infrastructure, http://ninf.apgrid.org/.
3) Kento Aida, Yoshiaki Futakata, Shinji Hara, "High-performance Parallel and Distributed Computing for the BMI Eigenvalue Problem," Proc. 16th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2002), Apr. 2002