B-029

ボランティアコンピューティングにおける 功績値評価に基づいた探索問題解決手法
Search Problem Solving based on Credit Evaluation for Volunteer Computing

伊草拓哉 † 松澤智史 ‡ 武田正之 ‡
Takuya Igusa Tomofumi Matsuzawa Masayuki Takeda

1. はじめに

近年，インターネット上の多数の有志ユーザから提供される遊休計算資源を集約し，大規模科学技術計算を行うボランティアコンピューティング (Volunteer Computing: VC) が研究され，ミドルウェアである Berkeley Open Infrastructure for Network Computing(BOINC)[1] を用いた多くの計算プロジェクトが運用されている．一般に VC は単純並列計算を扱うことが多いが，TANPAKU[2][3] のような実行順序依存な探索問題を扱うこともある．
BOINC では m-first 多数決を採用しているが，過剰な冗長計算によるシステムの性能低下と低い冗長性による計算結果の信頼性低下というトレードオフな問題がある．探索問題の場合，誤った結果のまま計算を続けると，それ以降のタスクの結果にも影響を及ぼし，最悪の場合無駄な計算を行なってしまったということになりかねない．
そこで本稿では前述のような探索問題において，BOINC で各ノードに割り当てられていく功績値という指標を用いてタスクの冗長度を決定する手法を提案する．

2.BOINC

2.1. システム概要

BOINC はマスタ・ワーカモデルを採用し，マスタがプロジェクトの管理とタスク配布を担い，ワーカはマスタに対しタスクを要求し，受け取ったタスクの処理を行う．タスクは要求した順序でワーカに割り当てられる．
ワーカはユーザから提供された遊休ノードであるため，これらの計算機が正しい計算結果を返してくる保証はない．そのためマスタはタスクに冗長性を持たせ同じタスクを複数のワーカに配布し，計算結果を m-first 多数決により決定することで信頼性を保証している．
VC では，ワーカとなる PC がシャットダウンや他の処理を行うなどして、タスクの処理が中断することが度々ある．それによりタスクの返却時刻が伸びてしまい，全体の処理効率が低下してしまったり，最悪の場合，タスクが消失してしまう可能性がある．そのため BOINC では各タスクに計算結果の提出期限が設けられている．提出期限を過ぎてしまった場合，マスタはそのタスクの処理が失敗したとみなし，他のワーカに再配布する．これにより，処理効率低下の問題を解決している．

2.2. 功績値

BOINC では，タスクを完了した各ワーカに対して功績値を与える．功績値は各ワーカがタスク完了時に，タスク処理にかかった CPU 時間や CPU ベンチマークの結果から算出し，マスタに対して申請する．マスタは各ワーカから返却されたタスクの処理結果について正当性などを検証し，申請された功績値を与えるかを決定する．マスタは各ワーカの功績を，総功績値と最近の平均功績値 (Recent Average Credit: RAC) という 2 つの値で管理する．RAC は最近の功績値獲得時刻からの経過時間に応じて値が減衰するもので，1 週間で値が半減する．算出方法の詳細は以下のとおりである．

$$RAC_n = \begin{cases} \frac{work}{t/86400} & (n=1) \\ w \times RAC_{n-1} + \frac{(1-w)work}{t/86400} & (n \geq 2) \end{cases}$$

$w = \exp(-\frac{t \log 2}{86400 \times 7})$
RAC_n：平均功績値
$work$：仕事量
t：処理開始から終了までの時間 (sec)
RAC_{n-1}：前回算出した平均功績値
w：時間経過による重み

3. 提案手法

3.1. 想定する探索問題

以下に今回対象とする探索問題の概要を示す．

- タスクは独立な複数のサブタスクから成る．
- 計算プロジェクトはゲーム木の構造を持ち，各節がタスクにあたる．
- タスクは処理終了時に次の処理タスクと，自らの検証タスクを生成する．
- 検証タスクの結果，処理タスクの計算結果が誤っていると判断した場合，その処理タスクと以降の部分木を削除し，再びその処理タスクを生成する．

つまり処理タスクの計算結果は求められた時点では仮の計算結果でしかなく，検証タスクの計算結果によって正しいと判断されたときに確定することになる．上記についてまとめた探索問題のモデルを図 1 に示す．

3.2. ノードのクラスタリング手法

VC システムのノードをクラスタリングする手法としては，福士らの文献 [4] において，Banerjee らが文献 [5] で提案した階層型クラスタリング手法を VC システムに適用した手法が提案されている．

†東京理科大学大学院　理工学研究科　情報科学専攻
‡東京理科大学　理工学部　情報科学科

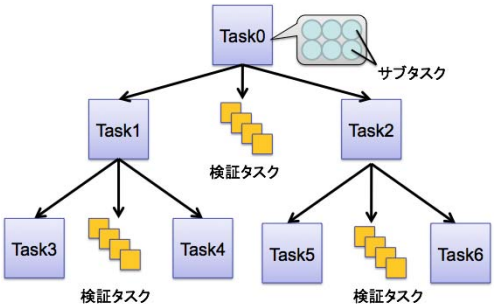

図 1: 想定する探索問題のモデル

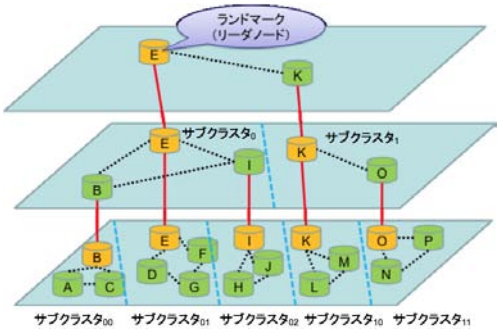

図 2: 階層型クラスタの基本構造

Banerjee らの手法は，特定のノードがランドマーク (目印) として他のノードをクラスタリングし，それらをさらに上位のランドマークノードがクラスタリングすることで階層的な構造を形成する．文献 [5] ではランドマークのことをリーダノード，最上位以外のリーダノードが束ねるクラスタ内の小さなクラスタをサブクラスタと呼称している．リーダノードは 1 つ上の層のサブクラスタにも属し，上位層に対するタスク要求や受け取り，下位のノードの管理，サブタスク配布などの役割を担う．Banerjee らが提案した階層型クラスタの基本構造を図 2 に示す．

福士らはノード間のネットワーク距離から Banerjee らの階層型クラスタを形成することで，通信が発生する VC プロジェクトにおける通信コストを最小化できることを示した．

本手法では 2.1 節で紹介した探索問題を対象とするため，プロジェクトに参加するノードを RAC によって階層的にクラスタリングすることで信頼性の高い計算結果を効率的に求めることを目的とする．サブクラスタ内のリーダノードは各ワーカと一定時間ごとに RAC や処理中のタスク，サブタスクなどの情報を交換する．このときに反応が返ってこないワーカは処理を休止しているものとみなす．リーダノードは所属しているワーカと自身の RAC の平均を求め，これをサブクラスタ全体の RAC として上位層のリーダノードに通知する．

3.3. クラスタの功績値とタスクの関係

BOINC では各タスクに優先度を割り振ることが出来る．優先度の高いタスクは計算プロジェクト内において重要なものであることが多い．探索問題のように

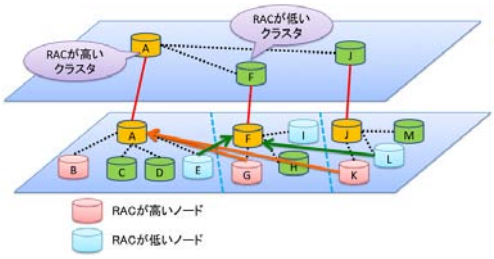

図 3: ノード移動の仕組み

階層形式になっている計算処理では上位層のタスクは下位のタスクに影響を与えるため，上位のタスクに優先度を高めに割り振っていることが多い．これらを早く正確に処理するため，優先度の高いタスクは功績値の高いクラスタを優先して割り当てるようにする．

また，功績値の高いクラスタが担当したタスクは正しい計算結果である可能性が高く，逆に功績値が低いクラスタが担当したタスクは誤っている可能性が高い．これらのことから，功績値が高いクラスタが担当したタスクについては検証タスクの生成数を少なくし，功績値の低いクラスタが担当したタスクは検証タスクを多めに設定することで，高確率で信頼できる結果を効率的に求めることが出来るようにする．

3.4. ノードのクラスタ間移動

RAC は時刻によって変化していくため，RAC によってクラスタリングを行う場合，その変化に応じてノードをクラスタ間移動させる必要がある．本節ではノードのクラスタ間移動について説明する．

同一クラスタ内において，RAC が上位 α 位までのノードと下位 α 位までのノードを移動の対象とする．移動先のクラスタもノードの RAC 順位によって決まり，ノードの RAC 順位を上位 x 番目とすると，移動先は RAC 順位が上位 x 番目のクラスタとなる．下位 x 番目のときも同様である．ただし各クラスタにはノード数の上限 W_{max} と下限 W_{min} が設けられており，ノードが移動することによって移動先のクラスタのノード数が W_{max} を上回ってしまう場合や移動元のクラスタのノード数が W_{min} を下回ってしまう場合，既に RAC 順位が上位 x 番目のクラスタに属している場合はノードの移動は起こらない．これにより，α 個のクラスタが他のクラスタと比べて信頼性と性能の高いものとなる．ノード移動についてを図 3 に示す．

4. 評価と考察

4.1. シミュレーションの概要

提案手法の有用性を示すため，ノードのクラスタリング機能を持つ独自のシミュレータを用いて実験を行い，提案手法と BOINC のマスタ・ワーカモデルについて完了タスク数と正確性の観点から評価を行った．完了タスクとは検証まで終了したタスクのことを表す．

クラスタの階層数は 2，つまりシステム全体ではマスタ, リーダノード, ワーカの 3 層構造とし，リーダノー

表 1: 実験パラメータ

パラメータ名	値
ワーカ数	1000
資源提供率 $P(\%)$	100,50,25,12.5
資源提供率ごとのノード比	1:6:13:20
クラスタ内階層数	2
クラスタ数	25
クラスタ間移動の対象ノード数 α	2
クラスタ内のノード数上限 W_{max}	100
クラスタ内のノード数下限 W_{min}	10
タスク内のサブタスク数	10
サブタスク粒度 (ターン)	20
サブタスク冗長度	5
処理タスク生成数	2
検証タスク生成数	2,4,6,8,10
悪意のノードの存在率 $a(\%)$	2,4,6,8,10
悪意のノードが誤結果を返す確率 $b(\%)$	90
処理終了ターン	10000

ドの離脱や休止は考慮しないものとする．ワーカは全ての時刻で計算能力を提供するのではなく，資源提供率 P に応じて計算を行うか否かを決定する．資源提供率ごとのノード比は TANPAKU のログデータをもとに決定した．また，ワーカには悪意のあるノードが a%含まれており，これらは b%の確率で誤った結果を返す．a の最大値は 10 としたが，これは TANPAKU のログデータから，誤った結果のタスクや紛失したタスクの割合が 10%近く占める期間が存在したためである．
その他の詳しい実験パラメータについては表 1 に示す．

以下にシミュレーションの流れを示す．

1. 全ワーカを各クラスタに対して，合計台数，資源提供率ごとの台数共に偏りが出ないように均一に配置する．これを初期状態とする．
2. マスタは構成されたクラスタに対して RAC の値に基づいて適当な優先度のタスクを割り当て処理を開始する．
3. タスクを割り当てられたリーダノードは，タスク内のサブタスクを下位のワーカに対して割り当てる．
4. サブタスクを完了したワーカは自身の RAC の値から，他のクラスタに移動すべきかを 3.4 節で説明したアルゴリズムに基づいてリーダノードに判断させる．完了したサブタスクはリーダノードに返され，手順 3 に戻る．
5. 完了したタスクをマスタに返却し，計算プロジェクトの処理が進行する．この際に完了したタスクが検証タスクでなければ，次の処理タスクと検証タスクを必要数だけ生成し，手順 2 に戻る．
6. 終了ターンになったらシミュレーション処理を完了する．

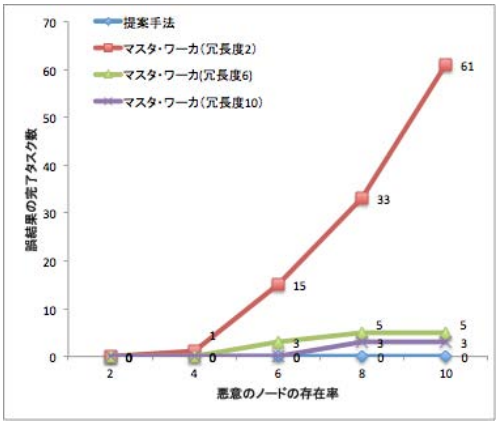

図 4: 悪意のノードの存在率に対する誤結果の完了タスク数

4.2. 悪意のノードの存在率と誤結果の完了タスク数の関係

悪意のノードが存在する割合と，誤結果の完了タスク数の関係を図 4 に示す．
通常，処理タスクの計算結果が誤っていた場合は検証タスクの結果から誤りを検知するが，検証タスクを担当したクラスタにも悪意のノードが含まれていた場合は誤りに気づかない可能性がある．本稿で処理タスクで誤った結果を求め，その検証タスクでも誤りに気付かなかったタスクのことを誤結果の完了タスクと呼ぶ．
提案手法とタスク冗長度が 2，6，10 のマスタ・ワーカモデルとを比較すると，提案手法が悪意のノードの存在率が増加しても誤結果の完了タスク数が 0 であるのに対し，その他の結果は悪意のノードの存在率が増えるにつれ，何らかの誤結果が生じていることが確認できる．
これは提案手法の場合，功績値に応じてクラスタリングしたことで悪意のノードが同一のクラスタにまとまり，その他のクラスタは悪意のノードが排除できたため信頼性が高まったことと，悪意のノードが集まったクラスタが処理した場合でもその他のクラスタが検証を行うことで誤結果を検知できたためと考えられる．一方，マスタ・ワーカモデルはどのノードがどの処理を担当するかはタスクの優先度とタスクを要求した順序で決定するため，処理タスクも検証タスクも悪意のノードが担当してしまい，処理タスクの誤結果を検知できない事態が発生したと考えられる．

4.3. 悪意のノードの存在率と処理効率の関係

次に悪意のあるノードの存在する割合と，完了タスク数の関係を図 5 に示す．
提案手法は悪意のノードの存在率が増加しても処理効率に大きな変化がないことが分かる．それに対しマスタ・ワーカモデルは，冗長度の値に関わらず，悪意のノードの存在率が増加するに連れて性能低下が起こっている．
提案手法の場合，功績値の高いノードは同一のクラスタに集まり同じタスクを処理するため，その計算結果

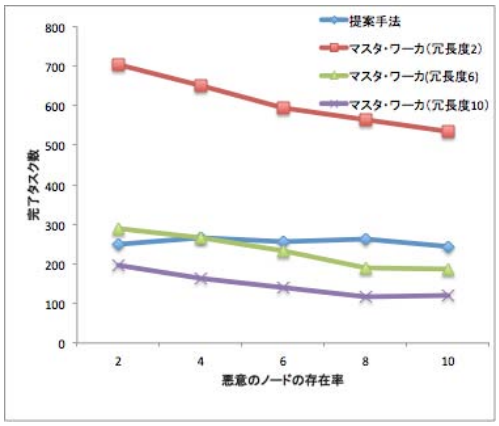

図 5: 悪意のノードの存在率に対する完了タスク数

が誤ったものである可能性は低い．悪意のノードが集まったクラスタは誤った結果を返す確率が高いが，このクラスタや以降の処理タスクを担当していたクラスタのノードは悪意のあるものか性能の低いものがほとんどであり，仮に検証により破棄されたとしても全体に与える影響は少なく済む．しかしマスタ・ワーカモデルの場合，各ノードが担当するタスクは要求順に決定されるため，功績値の高いノードと悪意のノードが同じタスクを担当してしまったり，誤結果のタスクの部分木にあたる処理を功績値の高いノードが担当してしまうことがある．その場合，後の検証の結果，部分木が破棄され功績値の高いノードの計算結果が無駄になってしまう．悪意のノードの存在率が高いほど，このことが起こる可能性は高まるため，図 5 のようなマスタ・ワーカモデルにおける性能低下が起こったと考えられる．
また，タスク冗長度 2 のマスタ・ワーカモデルが他の手法に比べて性能が良いことが示されているが，これは検証を行う回数が他の手法よりも少ないためである．しかし 4.2 節にあるように，タスク冗長度 2 のマスタ・ワーカモデルは計算結果の信頼性が低く，悪意のノードの存在率が高いほど誤結果を検出できない可能性が高まる．
以上のことから，悪意のノードが一定の割合以上含まれる環境において，提案手法は計算結果の信頼性と処理性能の面から有用であると言える．

5. まとめと今後の課題

本稿では実行順序依存関係のある VC 計算プロジェクトを高い信頼性と処理性能で実行することを目的に，功績値に基づいたクラスタリング手法を提案した．またクラスタの功績値により検証タスクの冗長度を調整することにより，悪意のノードが多数存在した場合でも高い信頼性と処理性能が維持できることをシミュレーション実験により確認した．
今後の課題としては功績値と冗長度の最適な関係を求めることや，ノードの新規参加，離脱に対応したクラスタリング手法に改良することなどが挙げられる．また，功績値は計算機の処理性能と正確性を合わせた尺度であるため，これらを分離した評価尺度を導入し，その関係性についても評価する必要がある．

参考文献

[1] David P. Anderson, “BOINC: A System for Public-Resource Computing and Storage”, Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on In GRID ’04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 4-10, 2004.

[2] Naoki Kobayashi, Tadashi Ando, Masayuki Takeda，“TANPAKU: Protein Folding Simulation Using Volunteer Computing”, TUS-NPU Bilateral Seminar 2008, pp.169-174, 2008.

[3] 安藤格士，山登一郎，小林直記，旭悠哉，武田正之，“分散コンピューティング環境でのタンパク質立体構造予測シミュレーション”，第 43 回 日本生物物理学会年会，2005.

[4] 福士将, 菅原雅也, 堀口進, “ボランティアコンピューティングにおけるノードの動的クラスタリング”, 第 8 回情報科学技術フォーラム (FIT2009), N0.4, pp.157-164, 2009.

[5] S.Banerjee, S.Parthasarathy, B.Bhattacharjee, “A protocol for scalable application layer multicast”, Technical Report CS-TR-4278, Department of Computer Science, University of Maryland College Park, USA, 2001.