Vol.2013-ARC-207 No.19
Vol.2013-HPC-142 No.19
2013/12/16

信頼度に基づく多数決法を用いたボランティアコンピューティングシステムの実装

土居 俊也[†1]　渡邊 寛[†1]　福士 将[†2]　舩曳 信生[†1]　中西 透[†1]

概要：ボランティアコンピューティング (VC) では，通常，誤った計算結果に含まれる誤りを排除するために，多数決法が採用される．先行研究では，各参加者が正答を返す信頼度を用いて重み付き多数決を行う信頼度に基づく多数決法が提案されている．シミュレーションにより，少ない冗長数で効率的に誤りを排除可能なことが示されているが，本手法を実際に動作させた際に発生する処理のオーバーヘッドはまだ明らかにされていない．そこで本稿では，実環境における有効性を確認するために，同手法を実装した VC システムを構築し，性能評価実験を行った．実験の結果，システムの規模が大きくなると，信頼度に基づく多数決法を用いた場合では，単純な多数決を用いた場合に比べて性能低下の割合が大きいことが確認された．また，多数決処理に要した時間の内訳を調べた結果，信頼度計算を伴う一部の多数決処理が，性能低下の割合を大きくする原因であることがわかった．

キーワード：並列分散処理，デスクトップグリッド，妨害者対策，計算信頼性，評価実験

An Implementation of Volunteer Computing System with Credibility-based voting

Abstract: In volunteer computing(VC) systems, a management node distributes a computation to multiple participants, collects result candidates, and then preforms voting to eliminate incorrect ones. To enhance the efficiency of voting methods, "credibility-based voting" is proposed, which performs weighted-voting based on each participants' credibility. In this paper, we implement this voting method to actual VC systems and evaluate the performance through experiments with varying scales of the system. As a result of the experiment, we confirmed that the performance degradation of credibility-based voting is proportional to the number of the workers. We also found that the performance degradation comes from the overhead of the management node, i.e. the computation of credibility.

Keywords: Parallel Computing, Desktop Grids, Sabotage-tolerance, Reliability, Evaluation Experiment

1. はじめに

ボランティアコンピューティング (VC) とは，ボランティア参加者からインターネットを通じて無償で提供されるアイドル CPU サイクル等の遊休計算資源を利用して，高性能な大規模並列計算システムを安価に構築する手法である．例えば，大規模な科学技術計算を取り扱う SETI@home[1] や Folding@home[2] では，数十万から数百万人のボランティア参加者を集め，スーパーコンピュータを上回る計算能力を獲得することに成功している．また，Kondo ら [3] による並列分散システムの構築方法の違いによるコスト差を調べた研究では, 計算資源をレンタルすることができる Amazon EC2 クラウドと比べて，ある一定以上の時間がかかる大規模な計算に対しては，VC を用いる方が低コストであるという結果が報告されている．

VC では不特定多数の PC を用いて計算を行うという性質上，誤った計算結果が返されるという問題がある．計算誤りは，PC のオーバークロックや故障，ソフトウェアのエラー，意図的な計算妨害行為など，様々な原因で，しか

†1 岡山大学大学院自然科学研究科
Presently with Graduate School of Natural Science and Technology, Okayama University
†2 山口大学大学院理工学研究科
Presently with Graduate School of Science and Engineering, Yamaguchi University

も，ボランティア参加者側の都合で発生する．この問題に対して，現行の VC システムでは，計算を冗長化して多数決を行うことで計算の信頼性向上が図られている．現在主流となっている VC ミドルウェア BOINC[4] では，M 個の同一の計算結果を集める M-first 多数決法が採用されている．

先行研究 [5] では，M-first 多数決法の改良として，計算結果に対して信頼度という厳密な確率的指標を導入することで，確率的に信頼性の保証が可能な，信頼度に基づく多数決法が提案している．本手法では，信頼度を重みとした重み付き多数決により，計算の冗長度を 2 以下に抑えることが可能であり，M-first 多数決法 (最も冗長度が低い 2-first 多数決法) よりも高速に計算を終えることが可能である．

しかし, 信頼度に基づく多数決法は有効な手法ではあるものの，実装上の性能は十分に評価されていない．すなわち，本手法は単純な多数決を行う M-first 多数決法とは異なり，信頼度計算を伴う複雑な多数決処理が必要となるが，この処理がネックになり，VC システム全体の性能に影響を及ぼす可能性がある．また，想定する多数決処理を適切に行うために，処理自体のオーバーヘッドに加えて，ディスク I/O や排他制御などのオーバーヘッドも発生するが，これらは実装を通じてしか得られない性能である．

そこで本稿では，信頼度計算のオーバーヘッドがシステム全体の性能に与える影響を検証するため，実際に信頼度に基づく多数決法を VC システムに実装し，その性能を評価した．評価実験では，実装した信頼度に基づく多数決法と M-first 多数決法との性能比較を行うことで，信頼度計算のオーバーヘッドによる性能低下の割合がどの程度であるかを明らかにする．また，多数決処理に要した時間の内訳を分析することで，実環境において具体的に何が大きなオーバーヘッドとなっているかを検証する．

以下では，まず 2 章で VC のモデルと高信頼化手法を示した後に，実環境において考えられるオーバーヘッドについて述べる．次に，3 章で実装した VC システムについて述べ，実装上，具体的にどのような形でオーバーヘッドが発生するのかを述べる．4 章では，性能評価としてワーカ数 W などを変化させた時の全体の実行時間の変化と，多数決処理の内訳を分析した結果を示す．最後に，5 章でまとめと今後の課題について述べる．

2. VC のモデルと高信頼化手法

2.1 計算モデル

実際に運用されているほとんどの VC では，計算モデルとして図 1 に示すマスタ・ワーカモデルが採用されており，本研究においてもこれを適用する．以下にこのモデルの概要を示す．

- マスタ 1 台とワーカ W 台から成るシステムで計算プロジェクトを実行する．

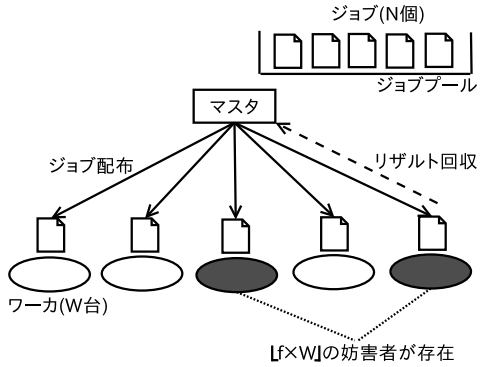


図 1 VC におけるマスタ・ワーカモデル

- 計算プロジェクトは独立した計算問題 (ジョブ)N 個から成り，マスタはワーカからのジョブ要求に応じて個々のワーカにジョブを配布する．
- ジョブを配布されたワーカはこれを実行し，生成された計算結果 (リザルト) をマスタへ返却する．
- 計算プロジェクトは N 個のジョブ全てが終了すれば完了となる．

また，誤りを返す妨害者が存在する VC を考える．ここでは，妨害者存在率 f を用いて VC に参加するワーカのうち $\lfloor f * W \rfloor$ の妨害者が存在するものとし，個々の妨害者は確率 s(妨害率) で誤った計算結果を返すとする．マスタにとって，妨害者存在率 f や妨害率 s は未知の値であり，事前に知ることはできない．マスタは推定値として妨害者割合の最大想定値 f_{max} を定義し，ワーカ W 台のうち最大で $\lfloor f_{max} * W \rfloor$ の妨害者が存在すると仮定する [6].

2.2 高信頼化手法

誤りを返す妨害者が存在する VC において，得られる計算結果の信頼性を向上させるための手法として，妨害者検出を目的とした spot-checking と，多数決による計算の冗長化がある．以下では本稿で用いる手法の概要を示す．

2.2.1 spot-checking

spot-checking では，確率 q(抜取検査率) で，通常のジョブの代わりに，予め正しい計算結果のわかっている検査問題をワーカに配布する．検査問題に対して誤ったリザルトを返したワーカは妨害者とみなされ，以下の 2 つの対策が施される．

- backtracking
  妨害者と判断されたワーカがそれまでに生成したリザルトを全てを無効化する．
- blacklisting
  妨害者を隔離し，以降このワーカに対してジョブの配布やリザルトの回収を行わない．

ただし，blacklisting が全ての妨害者に対して有効であるとは限らない．VC では，より多くのボランティア参加者を集めるために，参加の際に参加者個々の情報をあまり多

く要求しないことで参加障壁を下げているため，妨害者が別のワーカとして再参加しやすい仕組みとなっている場合が多い．例えば，参加に必要な情報が E メールアドレスのみだった場合，検出された妨害者は登録する E メールアドレスを変更することで，別のワーカとして簡単に VC に再参加することができてしまう．ここでは，blacklisting が有効な VC 環境である”with blacklisting”と，blacklisting が無効な VC 環境である”without blacklisting”の 2 つを考える．”without blacklisting”では, 検出された妨害者は即座に他のワーカとして VC に再参加するものと仮定する．

2.2.2 M-first 多数決法

M-first 多数決法は，1 つのジョブを複数のワーカに配布し，最初に同じ値のものが M 個集まったリザルトを対応するジョブの計算結果として採択する手法である．冗長度 M を小さくし過ぎると計算の誤り率が増大してしまうため，実際に運用されている VC システムでは冗長度が $M = 3$ 程度に設定されていることが多い．しかし，例えば $M = 3$ の場合，計算プロジェクトの完了に必要な計算が，本来の 3 倍になってしまうため，VC システム全体の性能が大幅に低下してしまう．

2.2.3 信頼度に基づく多数決法

信頼度に基づく多数決法は，得られた個々のリザルトなどに対して信頼度による重み付けを行い，重み付き多数決によって採択するリザルトを決定する手法である．この手法は，計算全体の誤り率が一定値 (許容誤り率 ϵ_{acc}) 以下になることを確率的に保証する．許容誤り率 ϵ_{acc} は，計算プロジェクトが要求する計算精度に従って，マスタが計算開始前に予め設定する値である．

以下では，信頼度の計算方法と採択するリザルトの決定方法について説明する．ワーカ w の信頼度は，with blacklisting と without blacklisting の場合で計算式が分かれており，検査問題に対して正しいリザルトを返した回数 k と妨害者割合の最大想定値 f_{max} を用いて，式 (1)，式 (2) で表される [5]．ただし，e は自然対数の底である．

with blacklisting の場合

$$C_W(w) = \begin{cases} 1 - \frac{f_{max}}{(1-f_{max})\times ke} & \text{if } k \neq 0 \\ 1 - f_{max} & \text{if } k = 0 \end{cases} \quad (1)$$

without blacklisting の場合

$$C_W(w) = \begin{cases} 1 - \frac{f_{max}}{k} & \text{if } k \neq 0 \\ 1 - f_{max} & \text{if } k = 0 \end{cases} \quad (2)$$

個々のリザルト r の信頼度 $C_R(r)$ は，そのリザルトを生成したワーカ w の信頼度 $C_W(w)$ と等しいものとする．

$$C_R(r) = C_W(w) \quad (3)$$

ジョブ j に対して複数のリザルトが返されている場合は，その値毎に複数のグループに分類し，個々のグループをリザルトグループと呼ぶ．ジョブ j のリザルトが g 個のリザルトグループ $G_1, G_2, ..., G_g$ に分けられたとすると，リザルトグループ G_a の信頼度 $C_G(G_a)$ は式 (4)-式 (6) によって計算される．

$$C_G(G_a) = \frac{P_T(G_a)\prod_{i\neq a} P_F(G_i)}{\prod_{i=1}^{g} P_F(G_i) + \sum_{n=1}^{g} P_T(G_n)\prod_{i\neq n} P_F(G_i)} \quad (4)$$

$$P_T(G_a) = \prod_{r\in G_a} C_R(r) \quad (5)$$

$$P_F(G_a) = \prod_{r\in G_a} 1 - C_R(r) \quad (6)$$

ジョブ j の信頼度 $C_J(j)$ は，式 (7) のようにリザルトグループ $G_1, G_2, ..., G_g$ の中で最大の信頼度を持つグループである，Gx の信頼度と定義される．

$$C_J(j) = C_G(G_x) = \max_{1\leq a\leq g} C_G(G_a) \quad (7)$$

ジョブ j の信頼度が閾値 $\theta = 1 - \epsilon_{acc}$ を超えた場合，マスタは G_x のリザルトを採択し，ジョブ j を終了する．

2.3 高信頼化手法を用いた管理ノードのオーバーヘッド

実環境において，高信頼化手法を用いる際には，シミュレーションでは現れない実装上のオーバーヘッドについて考慮する必要がある．管理ノード (マスタ) で発生するオーバーヘッドとして，以下のようなものが考えられる．

- ディスク I/O
  巨大な計算問題を管理する VC システムでは，必要なデータの全てをメモリ上で管理することは現実的ではない．よって，ジョブ配布，リザルト回収，多数決処理などのタイミングで，必要な情報の取得や記録のために，マスタにおいて少なからずディスク I/O が発生することとなり，そのオーバーヘッドが無視できない．
- 共有資源に対する排他制御
  計算プロジェクトがマスタによって一元管理されている環境下では，複数のワーカからの要求を並列に処理する際に，データベース等の共有資源へのアクセスに対して，排他制御 (ロック) を施す必要がある．大量のワーカからの要求を処理する VC では，共有資源への同時アクセスが頻繁に発生する可能性があり，その場合に増大する各要求処理のロック獲得待ち時間は，そのまま実装上のオーバーヘッドとなるため，VC システム全体の性能が大きく低下してしまうことがある．
- ネットワークの転送遅延
  実際の VC システムでは，インターネットを通じてマスタとワーカがやり取りするため，ネットワークを経由した際に通信の遅延が生じる．例えば村田ら [7] による研究では, ジョブのデータサイズが大きい場合に, マスタのネットワーク帯域幅がボトルネックとなり,VC システム全体の性能が大きく低下してしまう問題が報告されている．

高信頼化手法におけるオーバーヘッドの中でも，多数決処理は，ジョブの配布などの処理と比べてデータベース上

の多くの情報を必要とする分，大きなオーバーヘッドになりやすい．また，個々の多数決処理のオーバーヘッドが大きくなると，排他制御によるロック占有時間も増大するため，ディスク I/O のオーバーヘッドがそのまま排他制御のオーバーヘッドの増大に繋がる．

3. VC システムの実装

実環境上で多数決法を用いた場合のオーバーヘッドを検証するために，本稿では，M-first 多数決法と信頼度に基づく多数決法の各手法を C 言語で実装した VC システムを構築した．

以下では，まず 3.1 節で各多数決法のアルゴリズムを説明する．次に，3.2 節でこれらを実装した VC システムの概要を示し，3.3 節でリクエスト処理の流れを示す．また，3.4 節では，実装上の各オーバーヘッドがアルゴリズム上のどこで発生するか述べる．

3.1 多数決処理アルゴリズム

Algorithm1,2 に，M-first 多数決法，信頼度に基づく多数決法のアルゴリズムをそれぞれ示す．

Algorithm1 に示す M-first 多数決法では，通常ジョブのリザルトの場合 (2 行目) と，検査問題が誤答であった場合 (5,6 行目) に，多数決処理に伴うデータベース入出力が発生する．

一方，Algorithm2 に示す信頼度に基づく多数決法では，通常ジョブのリザルトの場合 (2,3 行目) と，検査問題のリザルトが誤答であった場合 (6-8 行目) に加えて，検査問題のリザルトが正答だった場合 (10-12 行目) にもデータベース入出力が発生する．特に，検査問題の場合は，正答，誤答の両方で複数のジョブに対する多数決を行うため，信頼度計算を伴う多数決に必要なデータベース入出力が何度も発生することとなる．

3.2 実装の概要

図 2 の上部に実装した VC システムの物理的構成を，下部にワーカからのリクエスト処理の概要を示す．

図上部に示すように，ワーカ PC 上ではワーカプロセスを複数起動し，これを仮想的なワーカとして扱う．マスタ PC 上では計算プロジェクトを管理するマスタプロセスを 1 つ起動する．個々のワーカプロセスは任意のタイミングでマスタプロセスにアクセスし，ジョブ配布やリザルト回収といった処理要求 (以下，リクエスト) をマスタに送信する．リクエストは常にワーカからマスタへ送信され，マスタからワーカに対して能動的にリクエストを送信することはない．複数のワーカ PC を用いる場合には，各ワーカ PC の IP アドレスと個々のワーカプロセスに割り振る ID の組をワーカの識別子として用いることで，マスタは個々のワーカプロセスを識別する．

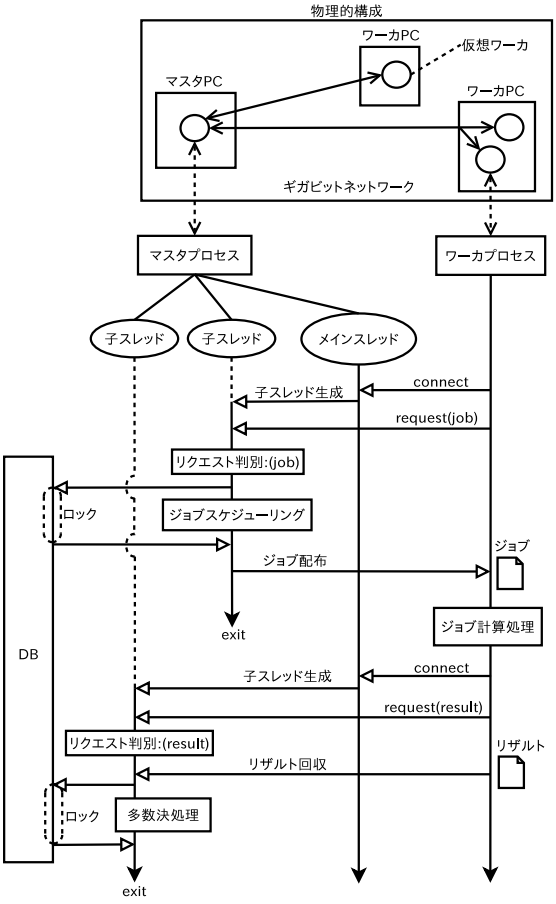


図 2 実装の概要

図 2 の下部に示すように，マスタ側では，マルチスレッドによる並列処理を行なっている．ワーカからのアクセスを受け付けるメインスレッドは，アクセスの度にリクエス

Algorithm 1 M-first 多数決法の多数決処理

```
1: if 通常ジョブのリザルト then
2:     DB からリザルト情報を取得し，多数決を実行
3: else // 検査問題のリザルト
4:     if 誤答 then
5:         妨害者のリザルトを無効化
6:         無効化したリザルトを含むジョブ全てに対する多数決
7:     end if
8: end if
```

Algorithm 2 信頼度に基づく多数決法の多数決処理

```
1: if 通常ジョブのリザルト then
2:     DB からワーカの検査問題通過回数とリザルト情報を取得
3:     信頼度計算を伴う多数決を実行
4: else // 検査問題のリザルト
5:     if 誤答 then
6:         妨害者のリザルトを無効化
7:         無効化したリザルトを含むジョブ全てに対する多数決
8:         各多数決のために DB から信頼度計算に必要な情報を取得
9:     else // 正答
10:        ワーカの信頼度上昇
11:        ワーカが返したリザルトを含むジョブ全てに対する多数決
12:        各多数決のために DB から信頼度計算に必要な情報を取得
13:    end if
14: end if
```

ト処理を担当する子スレッドを生成する．ワーカからの具体的なリクエストは生成された子スレッドが受け取り，リクエストが「ジョブ配布」か「リザルト回収」かを判別した後に各処理に移る．

3.3 リクエスト処理の流れ

各子スレッドがジョブ配布要求 (図 2 中，request(job)) を受信した場合，ジョブスケジューラを起動してどのジョブを配布するかを決定する．どのジョブを配布するかは，計算プロジェクトの進捗とジョブスケジューラのジョブ選択手法によって決定される．この際，ジョブを管理するデータベースにアクセスする必要があるが，データベースには，同時に 1 つの子スレッドからしかアクセスできないため，ロックを獲得する必要がある．

ジョブを受け取ったワーカは即座にジョブの計算処理を開始し，リザルトを生成する．ワーカは，生成したリザルトを返却するためにリザルト返却リクエスト (図 2 中，request(result)) をマスタに送信する．リクエスト処理を担当する子スレッドは，リザルトを回収したら，対応するジョブの多数決処理を開始する．この多数決処理の際にも，処理を行う子スレッドはデータベースにアクセスする必要があるため，ロックの獲得を行う．

3.4 実装上のオーバーヘッド

図 2 における「ジョブスケジューリング」と「多数決処理」では，データベースへの入出力が伴うため，各処理に対して，mutex ロックを用いてアクセスを制限している．また，共通の mutex 変数を用いているため，例えばある子スレッド A が多数決処理を行なっている最中に，他の子スレッド B のジョブスケジューリングが発生した場合，A の多数決処理が完了するまで B のジョブスケジューリングを進めることはできない．よって，ある処理のデータベースアクセスでロック占有時間が増大すると，他の全てのリクエストが待たされることになる．つまり，各ジョブスケジューリングや，多数決処理に時間がかかるほど，VC システム全体の性能が低下してしまうことになる．

また，マスタへのアクセスが集中しすぎた場合に，VC システム全体が停止してしまわないように，タイムアウト制御によって，一定時間内に処理できなかったリクエストの破棄を行なっている．ワーカ混雑時にはタイムアウトが頻発するため，リクエストを破棄されたワーカが，一斉にリクエストの再送を行う可能性が高い．その場合，マスタへのアクセス集中が連続的に発生し，混雑状態が継続してしまうため，VC システム全体の性能が著しく低下してしまう．

信頼度に基づく多数決法では，3.1 節で述べたように，M-first 多数決法よりも多くのデータベース入出力が伴うため，多数決処理における mutex ロック占有時間が長くなりやすく，実環境上ではリクエストのタイムアウトを誘発する悪循環に陥りやすいと考えられる．

表 1 実験パラメータ

ジョブ数 N	ワーカ数 W の 10 倍
ジョブの計算量	100 秒 (sleep 100)
ワーカ数 W	100 - 1000
抜取検査率 q	0.25
許容誤り率 ϵ_{acc}	0.05
妨害者割合の最大想定値 f_{max}	0.4
妨害者存在率 f	0.4
妨害率 s	0.1 - 1.0

4. 評価実験

3 章で示した実装を用いて，各多数決法の性能評価を行った．表 1 に実験に用いたパラメータを示す．実験環境として，マスタ PC1 台 (Ubuntu 12.04, CPU:Intel Core i7-3770, メモリ:8GB) とワーカ PC1 台 (Ubuntu 12.04, CPU:Intel Core2 Quad Q6700, メモリ:4GB) を用いた．また，データベースとして CSV 形式のテキストを用いている．

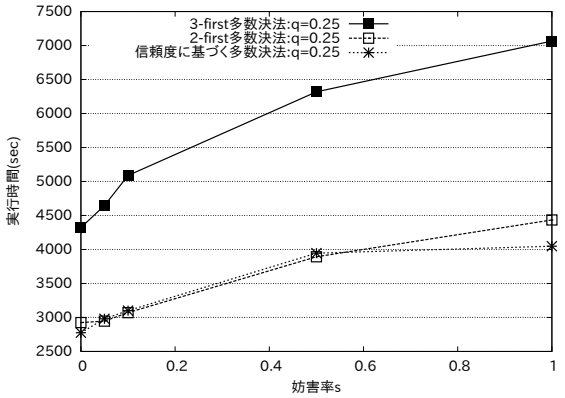
評価実験では，ジョブスケジューリング手法としてラウンドロビン法を用いた．ラウンドロビン法では，各ジョブに割り振られたジョブ ID 順に配布するジョブを選択する．もしジョブが完了していれば次の ID のジョブを選択し，ジョブ ID が一巡したら再度ジョブ ID の 1 番から順にジョブを選択する．

実験は”without blacklisting”環境下で行い，妨害者は検査問題に誤答した場合，即座に他のワーカとして VC システムに再参加する．また，妨害者はジョブのリザルトとして予め決められた正答か誤答のどちらかを生成するものとした．

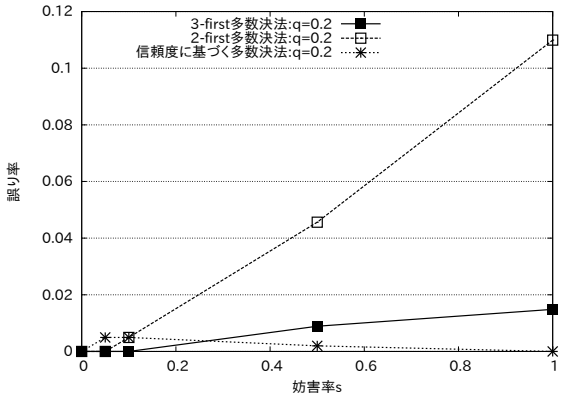
4.1 妨害率 s

図 3 に，妨害率 s を変化させた場合の実行時間と誤り率を示す．図 3(a) から，信頼度に基づく多数決法は，3-first 多数決法と比べて常に高速に動作し，2-first 多数決法と比べても同程度の速度で動作していることが分かる．全体的に，妨害率 s が増加する程，実行時間が増大傾向にある．これは，妨害者が抜取検査に誤答する確率が上昇し，リザルトの無効化によって計算プロジェクトの進捗が遅れるためである．

また，図 3(b) から分かるように，M-first 多数決法では，妨害率 s が高くなるほど誤り率が上昇するのに対して，信頼度に基づく多数決法では常に低い誤り率を維持している．特に，3-first 多数決法と比べた場合，信頼度に基づく多数決法は高速に計算プロジェクトを終了させつつ，誤り率を低く抑えている．これは，正しい計算結果を返すワーカのリザルトの重みを大きくすることで，効率的にジョブを完了させることができるためである．以上の実験によって，実環境においても信頼度に基づく多数決法が M-first



(a) 実行時間



(b) 誤り率 s

図 3 妨害率 s に対する実行時間と誤り率 ($W = 100, N = 1000$)

多数決法よりも有効に働くことが確認できたと言える．

4.2 ワーカ数 W

図 4 に，ワーカ数 W を変化させた場合の，実験における実行時間 (左軸) とシミュレーションにおける実行時間 (右軸) を示す．ここで，シミュレーションにおける単位「ターン」は，ジョブの配布からリザルトの回収までを 1 ターンとした場合の実行時間の単位である．

実験パラメータとして，ジョブ数 N を常にワーカ数 W の 10 倍に設定しているので，計算プロジェクト完了までに各ワーカが実行しなければならないジョブの平均個数は,W の値に関わらず常に一定となる．よって，図のシミュレーション結果 (右軸) が示すように，各ワーカに対するジョブ配布やリザルト回収が完全並列処理で，実環境で発生するようなオーバーヘッドが存在しない環境下では，ワーカ数とジョブ数の変化に関わらず実行時間は常に一定となる．

しかし，実験 (左軸) ではシミュレーションとは異なり，どちらの多数決法においてもワーカ数が増大するほど実行時間が増大していることが分かる．M-first 多数決法では，ワーカ数が 700 から 1000 に増大した時に性能が大きく低下しているのに対して，信頼度に基づく多数決法では，300 から 500 になった時に性能が大きく低下していることが確認できる．また，ワーカ数が 300 以下と少ない間は，信頼度に基づく多数決法は実環境においても高速に動作するが，ワーカ数が 500 の時点で M-first 多数決法との性能が逆転してしまい，以降の性能低下の割合も M-first 多数決法より大きいことが分かる．

4.3 多数決処理に要した時間の内訳

図 4 に示したように，用いる多数決法によってワーカ増大時の性能低下の割合は異なる．この原因を調べるために，Algorithm1,2 の実行に要した時間の内訳を調べた．

4.3.1 M-first 多数決法の内訳

図 5 に 2-first 多数決法における多数決処理に要した時間の内訳を示す．図中の各項はそれぞれ，Algorithm1 の 2 行目 (normal-voting)，4-7 行目 (backtracking) を指している．ただし，update-k-voting は Algorithm1 の 4 行目にあたる if 文にかかった時間を指す．

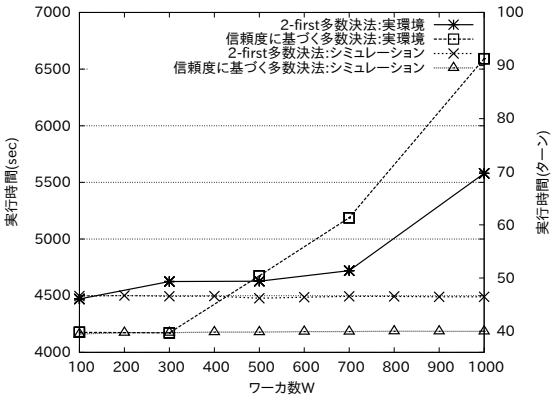


図 4 ワーカ数 W に対する実行時間 ($s = 1.0$)

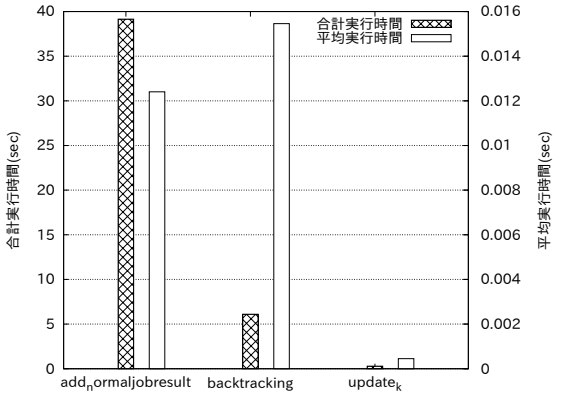
図 5(a),(b) の平均実行時間 (右軸) を比べると，ワーカ数 W やジョブ数 N が増えた場合，normal-voting と backtracking の平均実行時間が増大していることが分かる．これは，扱うジョブ数 N が 10 倍に増えたことで，データベース入出力にかかる時間が増大したためである．一方，Algorithm1 に示したように，検査問題正答時は多数決を行わないため，update-k-voting の処理にかかる時間がほぼゼロであることが分かる．

4.3.2 信頼度に基づく多数決法の内訳

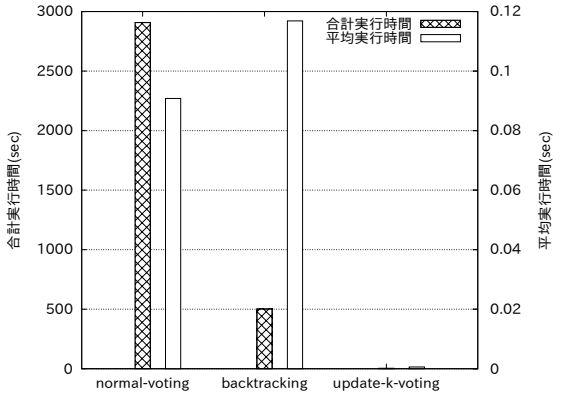
図 6 に，信頼度に基づく多数決法における多数決処理に要した時間の内訳を示す．図中の各項はそれぞれ，Algorithm2 の 2,3 行目 (normal-voting), 6-8 行目 (backtracking), 10-12 行目 (update-k-voting) を指す．

図 6(a)(b) から，ワーカ数増大時における normal-voting と backtracking の平均実行時間 (右軸) の増加割合は，M-first 多数決法とあまり変わらないことが分かる．

しかし，update-k-voting の平均実行時間は，他の多数決処理と比べてかなり長いことが分かる．例えば $W = 1000$

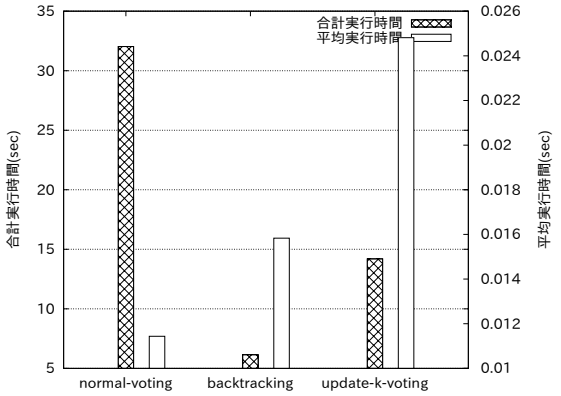


(a) ワーカ数 $W = 100$, ジョブ数 $N = 1000$

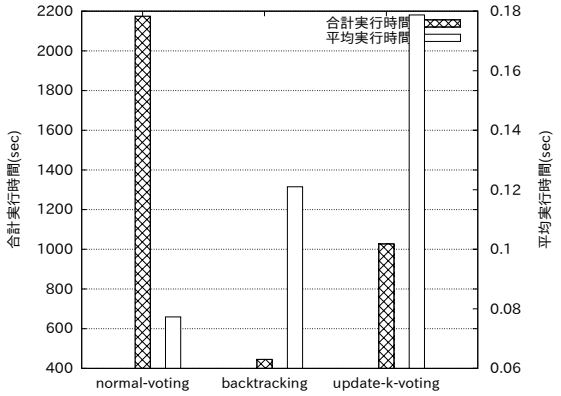


(b) ワーカ数 $W = 1000$, ジョブ数 $N = 10000$

図 5 2-first 多数決法における多数決処理に要した時間の内訳 ($s = 1.0$)



(a) ワーカ数 $W = 100$, ジョブ数 $N = 1000$



(b) ワーカ数 $W = 1000$, ジョブ数 $N = 10000$

図 6 信頼度に基づく多数決法における多数決処理に要した時間の内訳 ($s = 1.0$)

の場合，normal-voting が平均 0.0773 秒，backtracking が平均 0.121 秒かかるのに対して，update-k-voting は平均 0.179 秒かかっている．3.4 節で述べたように，個々の多数決処理にかかる時間が大きいほど，VC システム全体の性能は著しく低下してしまう．また，図 6(b) の合計実行時間 (左軸) から分かるように，update-k-voting が多数決処理に占める時間の割合は比較的大きく，無視できない．そのため，ロック占有時間の長い update-k-voting がある分，信頼度に基づく多数決法は M-first 多数決法よりも，ワーカ増大時の性能低下の割合が大きくなったと考えられる．

5. 今後の課題

本稿では，M-first 多数決法と信頼度に基づく多数決法の各手法を実装した VC システムを構築し，多数決法の違いによる性能差を比較した．評価実験の結果から，ワーカ数やジョブ数が少ない間は，信頼度に基づく多数決法は M-first 多数決法よりも有効に動作するが，それらが増大した場合には，シミュレーションと異なり性能が逆転してしまうことがわかった．また，各多数決処理の実行時間の内訳を分析した結果，実環境において信頼度に基づく多数決法を用いた際に，VC システム全体の性能が著しく低下してしまう原因が，検査問題正答時の多数決処理で発生するオーバーヘッドにあることがわかった．

今後の課題としては，多数決処理の際のオーバーヘッドを低減するための，データベースの高速化や，実アプリケーションを用いた評価などが挙げられる．

参考文献

[1] SETI@home. http://setiathome.berkeley.edu
[2] Folding@home. http://folding.stanford.edu
[3] D. Kondo, B. Javadi, P. Malecot, F. Cappello and D. P. Anderson, "Cost-Benefit Analysis of Cloud Computing versus Desktop Grids", In Proc. of the 2009 IEEE International Symposium on Parallel and Distributed Processing, pp.1-12, 2009.
[4] BOINC. http://http://boinc.berkeley.edu
[5] K. Watanabe, M. Fukushi and S. Horiguchi, "Expected-credibility-based Job Scheduling for Reliable Volunteer Computing", IEICE Transactions on Information and Systems, Vol.E93-D, No.2, pp.306-314, 2010.
[6] Luis F. G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems", Future Gener. Comput. Syst., vol.18, Issue.4, pp.561-572, 2002.
[7] 村田 善智, 石杜 佑記, 滝沢 寛之, 小林 広明, " 動的負荷分散機能を持つ高性能ボランティアコンピューティングの実現", 情報処理学会論文誌, vol. 52, no. 2, pp. 401-414, 2011.