Vol.2012-ARC-202 No.16
Vol.2012-HPC-137 No.16
2012/12/13

デスクトップグリッドにおけるワーカの性能差を考慮した信頼度計算式の拡張

渡邊 寛[1,a)]　福士 将[2]　舩曳 信生[1]　中西 透[1]

概要：近年，インターネットに接続された多数のコンピュータ（ワーカ）の遊休計算資源を用いることで，スーパーコンピュータ並の計算性能を実現するデスクトップグリッド（DG）が注目されている．本稿では，誤った計算結果を返すワーカ（妨害者）が存在する DG において，計算結果の信頼性を保証する手法である，信頼度に基づく多数決法の拡張を行う．本手法では，各ワーカの信頼性（信頼度）を条件付き確率として計算することで，計算結果の誤り率を常に許容値以下に抑えることが可能であるが，この際，「全ワーカの性能が同じ」であることを前提としていた．そこで本稿では，各ワーカの性能が異なり，かつ，その性能が未知であるといった，より実環境に近い状況を想定して，信頼度計算式の拡張を行う．最悪ケースを想定した，妨害者の性能が非妨害者よりも 10 倍高い場合などのシミュレーション結果から，拡張した計算式を用いることで，誤り率を常に許容値以下に収められることを確認した．

キーワード：妨害者対策，計算信頼性，並列分散処理，数学モデル化，デスクトップグリッド

An Extension of Credibility Formula in Desktop Grid to Different Workers' Performance

Kan WATANABE[1,a)]　Masaru FUKUSHI[2]　Nobuo FUNABIKI[1]　Toru NAKANISHI[1]

Abstract: To efficiently improve the sabotage-tolerance of Desktop Grid (DG) systems, *credibility-based voting method* is proposed. Assuming that every worker has the same performance, this method can guarantee the condition that the error rate is less than the specified acceptable value. In this paper, we extend the credibility formula so as to afford more realistic DG situations where workers may have different performances. Even if performances of the attackers are unknown, our extended formula can calculate the credibility by considering the worst case where their performances are higher than others. Through simulations, we confirm that the error rate is always less than the acceptable value.

Keywords: Sabotage-tolerance, Reliability, Parallel Computing, Mathematical Modeling, Desktop Grids

1. まえがき

デスクトップグリッド (DG) は，廉価なパーソナルコンピュータ (PC) を計算資源として用いた，グリッドコンピューティング環境の構築手法である．DG では，スーパーコンピュータと同等以上の高い性能を，数万台規模の多数のデスクトップ PC を用いて実現する．また，インターネットを利用している一般の PC 利用者から，その PC の遊休計算資源をボランティアで（無償で）DG システムのノードとして提供してもらうことが可能であり，これにより計算環境を非常に安価に構築することが可能となる（このような DG は，ボランティアコンピューティングとも呼ばれる）．現在，世界規模で運用されている DG システムの例として，SETI@home[1] などがある．

DG の問題点として，ボランティア参加者の中に，誤った計算結果を故意に返すといった悪意ある者（**妨害者**）が

[1] 岡山大学　大学院自然科学研究科
[2] 山口大学　理工学研究科
[a)] can@cne.okayama-u.ac.jp

混じる可能性があるため，計算結果の信頼性が低いことが指摘されている [2], [3], [4]．DG では，参加者から提供される各計算資源（**ワーカ**）で，小規模に分割された計算（**ジョブ**）をそれぞれ独立に実行してもらうため，多くの参加者を集めるほど高い性能を得ることが可能となる．よって，より多くの参加者を集められるよう，参加者に個人情報の提供を強制することなどが難しく，実質的には，妨害者を含む「誰でも」が計算に参加できるようになっている．その結果，妨害者の数が多い場合は，計算結果の信頼性が著しく低下するため，何らかの対策を行う必要がある．

基本的な妨害者対策手法として，ミドルウェア BOINC[5] で用いられている**多数決法**が挙げられる．この手法では，同じジョブを複数のワーカに計算させて冗長に計算結果を集め，単純にそれらの多数決を取る．しかし，単純な多数決法では，冗長度を m とした場合に性能が $1/m$ となり，性能が著しく低下する．また，経験則等に基づいて手動で冗長度を決定するため，計算結果に含まれる誤りの割合（**誤り率** ϵ）がいくらになるか分からない，といった問題がある．

この単純な多数決法の改善として，各ワーカに**信頼度**と呼ばれる重みを与えて重み付き多数決を行う，**信頼度に基づく多数決法** [3] が提案されている．本手法では，各ワーカに対して，**抜取検査**と呼ばれる正答が既知のジョブを定期的に与え，その結果に基づいて，正しい計算結果を返す確率としての信頼度を求める．この信頼度が高いワーカほど大きな重みを持ち，その計算結果が多数派として採用されやすくなることから，小さな冗長度で効果的に誤り率を低減することが可能となる．また，本手法では，確率的な解析に基づいて計算した信頼度を用いることで，誤り率 ϵ を常に所望値（**許容誤り率：** ϵ_{acc}）以下とする冗長度が自動的に決定される．本グループでは，これまで，同手法の効率を改善するスケジューリング法 [6], [7] や，抜取検査を行う頻度の最適化法 [8] などの研究を行っている．

ここで，信頼度に基づく多数決法の問題点として，**全ワーカの性能が均一**と仮定して信頼度を計算している点が挙げられる．実際の DG 環境では，各参加者の PC のスペックはそれぞれ異なり，ワーカの性能としては不均一となるのが自然である．このため，従来の信頼度計算式を実際の DG でそのまま適用した場合，正しい信頼度が計算できない恐れがある．特に，妨害者の性能が非妨害者の性能よりも高い場合には，より多くの誤りが生成されるため誤り率が増大し，「常に誤り率 $\epsilon \leq$ 許容誤り率 ϵ_{acc}」の条件を満足できなくなる可能性がある．

そこで本稿では，**各ワーカの性能が不均一**であることを考慮して信頼度を計算するために，信頼度計算式の拡張を行う．この拡張により，各ワーカの性能が異なる場合，特に妨害者の性能が非妨害者の性能よりも高いような場合においても，「常に $\epsilon \leq \epsilon_{acc}$」の条件を満足できるようになる．また，拡張した計算式では，各妨害者の性能が分からない場合にも，ワーストケースとして全ワーカの中で性能の高い順に妨害者と見なすことで，信頼度の計算を可能とする．

また本稿では，拡張した計算式の有効性を評価するために，各ワーカの性能が不均一となる DG 環境のシミュレーションを行う．まず，従来の計算式を用いた場合に，「$\epsilon > \epsilon_{acc}$」となる場合があることを示す．次に，そのような場合でも，拡張した計算式により，「$\epsilon \leq \epsilon_{acc}$」とできることを確認する．

本稿の構成は以下の通りである．まず，2 章で想定する DG のモデルを述べ，3 章で信頼度に基づく多数決法について述べる．次に，4 章でワーカの性能差を考慮した信頼度計算式の拡張を行う．5 章では，従来の信頼度計算式と，提案する信頼度計算式を用いた場合それぞれに対して，シミュレーションによる評価を行い，最後に 6 章で結論と今後の課題を述べる．

2. DG モデルと多数決法

2.1 計算モデル

DG の計算モデルとして，図 1 に示す，マスタワーカモデル [5] を仮定する．本モデルの詳細は以下の通りである．

- **計算プロジェクト**は，N 個の独立したジョブから成り，全てのジョブが完了したら計算終了となる．
- **マスタ** 1 台と，ボランティアにより提供される**ワーカ** W 台で，計算プロジェクトを実行する．
- 各ワーカは，アイドル状態になるとマスタに**ジョブ**を 1 つ要求し，ジョブを実行後，その計算結果（**リザルト**）を返却する．

2.2 妨害者モデル

DG における妨害者は，次のようにモデル化される [3]．まず，全ワーカ W 台のうちの $\lfloor f \times W \rfloor$（$\lfloor\rfloor$ はフロア関数）台のワーカを妨害者（f:妨害者割合）とする．各妨害者は誤ったリザルトを確率 s（妨害率）で返却する．f と s はマスタにとって未知の値である．

妨害者の存在する DG では，マスタによりリザルトの検査等が行われる場合がある．このため各ジョブは，ワーカ

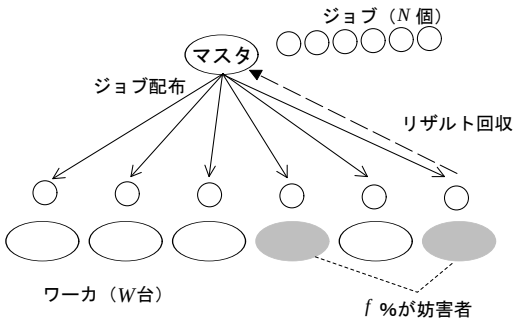


図 1 DG の計算モデル

からリザルトが回収され，マスタによる多数決等の検査を経て最終的なリザルトが確定した時点で，初めて完了ジョブとなる．全てのジョブが完了ジョブとなるまでに要した時間を**計算時間** T とする．また，完了ジョブのうち，誤ったリザルトにより完了したジョブ（誤りジョブ）が占める割合を**誤り率** ϵ と定義する．

ここで，1 つのジョブに対してリザルトを 1 つ生成するだけでは，リザルトの誤りが直接ジョブの誤りとなり計算結果の信頼性が著しく低下する．例えば，各ワーカの性能やジョブの計算量が均一の場合，誤り率は $\epsilon = \frac{N \times f \times s}{N} = f \times s$ となる．すなわち，妨害者数や妨害率が大きいほど，誤り率 ϵ が大きくなるため，1 つのジョブに対して複数のリザルトを集めたり，妨害者を検出し排除するなどして誤り率 ϵ を小さくする，妨害者対策が不可欠となる．

2.3 多数決法による妨害者対策と問題点

基本的な妨害者対策手法として，DG ミドルウェア BOINC[5] で用いられている**多数決法**がある．これは，同じジョブを複数のワーカに計算させて冗長にリザルトを集め，単純にそれらの多数決を取る手法である．しかし，本手法では，リザルトを集める個数 (**冗長度**) を m とした場合に，計算性能が $1/m$ に低下するといった問題がある．また，経験則等に基づいて手動で冗長度を決定するため，誤り率 ϵ が予測できない．このため，必要以上に高い冗長度を設定してしまい，システム全体の性能を損なったり，逆に，誤り率が許容値を超え計算結果を利用できないといった問題が生じる可能性がある．誤り率の許容値の例としては，例えば 3 次元映像のレンダリングでは誤り率 1%[3] などがある．

3. 信頼度に基づく多数決法

3.1 概要

信頼度に基づく多数決法 [3] は，より効率的に妨害者対策を行うために，新しく**信頼度**と呼ばれるパラメータを導入し，信頼度を重みとした重み付き多数決を行うことで，計算結果の信頼性を確保しながら冗長度を下げる手法である．信頼度は，ワーカやリザルトなどの DG 環境内の各要素に対して与えられる，その要素の正しさを表す条件付確率である．各ワーカの信頼度は，そのワーカに対する抜取検査の結果を評価することで求める．高い信頼度は，そのワーカやリザルトがより高い確率で正しいことを意味するため，信頼度を重みとする重み付き多数決では，正しいリザルトが多数派となりやすくなり，効率的に誤り率を低減することが可能となる．

3.2 特徴

信頼度に基づく多数決法 [3] の特徴として，各ジョブの冗長度が一定でなく，信頼度の値に基づいて動的に変化する点が挙げられる．これは，ジョブを完了するための基準が，集まったリザルトの数ではなく，各リザルトの信頼度から動的に計算される「ジョブの信頼度」が閾値 θ に達するまで，と定められているからである．ある時刻におけるジョブの信頼度は，その時点までに集まったリザルトの中の多数派が，実際に正しい確率の下限として計算される．ここで，確率の下限として信頼度を計算する理由は，実際に誤りである確率が未知パラメータである s や f に依存するためである．下限として信頼度を与えると，信頼度 $1-x$ のジョブを完了とした場合にこのジョブが正しい確率は，s や f によらず常に $1-x$ 以上であり，誤りである確率は x 以下である．閾値 θ は任意の値に設定することが可能であるため，任意の許容誤り率 ϵ_{acc} について $\theta = 1 - \epsilon_{acc}$ とすれば，常に $\epsilon \leq \epsilon_{acc}$ とすることができる．

3.3 使用するパラメータと仮定

信頼度に基づく多数決法 [3] では，信頼度を計算する際に，以下の前提をおいている．

- 全ワーカの性能は均一で，1 つのジョブを計算するのに 1 単位時間を要する．
- 妨害者割合 f に関して，実際の f は未知数であるため，その上限として f_{max} を仮定する ($f \leq f_{max}$)．そのため，f_{max} が正しくない場合（$f > f_{max}$ となるような場合），$\epsilon \leq \epsilon_{acc}$ が満足されない可能性がある．
- 抜取検査を行う頻度として確率 q を用いる．各ワーカは，ジョブを受け取る際に確率 q で抜取検査用のジョブを渡され，検査を受けることとなる．
- 抜取検査により妨害者として検出された時点で，そのワーカがそれ以前に生成したリザルトは全て無効化される．

  妨害者として検出された後のワーカの振る舞いとして，そのワーカがマスタの**ブラックリスト**に登録され以後の計算に参加できなくなる場合（ブラックリスト有効の場合）と，検出された時点で新しいワーカとして ID 等を再度取得し，計算に参加し続ける場合（ブラックリスト無効の場合）がある．

3.4 信頼度の計算式

信頼度に基づく多数決法における，信頼度の計算方法 [3] を説明する．3.3 節で説明した仮定のもとで，ワーカ w の信頼度 $C_W(w)$ は，抜取検査に通過した回数 k を用いて，ブラックリスト有効の場合には式 (1)，無効の場合には式 (2) で求められる．

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

$$C_W(w)_{no-bl} = \begin{cases} 1 - f_{max}, & \text{if } k = 0, \\ 1 - \frac{f_{max}}{k}, & \text{otherwise.} \end{cases} \quad (2)$$

ただし，e は自然対数の底である．

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 に対するリザルトは，値の等しいリザルトごとに複数のグループ（**リザルトグループ**）に分けられる．ジョブ j のリザルトが g 個のリザルトグループ $G_1, G_2, \ldots, G_a, \ldots, G_g$ 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)-(6) で計算される．$C_G(G_a)$ は，リザルトグループ G_a のみが正しいリザルトの集合であり，他の全てのリザルトグループが誤りである確率となる．

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

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

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

ジョブ j の信頼度 $C_J(j)$ は，$G_1, \ldots, G_g$ の中で最大の信頼度を持つグループ G_x 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 のリザルトをジョブ j のリザルトとして受け入れ，ジョブ j を完了とする．

4. ワーカの性能差を考慮した信頼度計算式の拡張

4.1 計算式拡張の意義

3.3 節で述べたように，従来の信頼度計算式は，「全ワーカの性能が均一」であることを前提としている．そのため，実際の DG 環境のように各ワーカの性能が異なる場合，特に妨害者の方が非妨害者よりも性能が高い場合には，正しい信頼度の値が計算できず，5.2 節のシミュレーション結果にも示すように，結果として $\epsilon \leq \epsilon_{acc}$ が満足されない場合がある．

本稿では，各ワーカの性能が不均一の場合にも，常に $\epsilon \leq \epsilon_{acc}$ を満足できるようにするため，信頼度計算式の拡張を行う．ここで，各ワーカの性能値は，単位時間当たりに実行可能な計算量で表し，DG ミドルウェア BOINC で行われているように，ベンチマークプログラムを各ワーカに実行させることで事前に得られるものとする．また，各ジョブの計算量はそれぞれ同程度となるよう分割されているものとする．例えば，ある計算がそれぞれ計算量１０のジョブに分割された場合，これを性能２のワーカが実行すると，１ジョブあたり５単位時間でリザルトが得られるものとする．

4.2 信頼度計算式の拡張

ワーカ W 台の性能を，それぞれ $p_1, \ldots, p_W$ と定義する．各ワーカは，その性能に応じて一定期間内に $p_i (i = 1, \ldots, W)$ 個のリザルトを生成する．また，ワーカ W 台の中で，$\lfloor f\times W \rfloor$ 台存在する妨害者の性能を，$p_{sb_1}, \ldots, p_{sb_{\lfloor f\times W\rfloor}}$，残りの非妨害者の性能を，$p_{nosb_1}, \ldots, p_{nosb_{\lceil (1-f)\times W\rceil}}$ とする．

信頼度に基づく多数決法において，k 回の抜取検査を通過したワーカの信頼度は，そのワーカが返す任意のリザルト r が正しい確率の下限として与えられる．ワーカが返すリザルトは，正しいか誤りのどちらかなので，リザルト r が正しい確率の下限（信頼度）は，１からリザルト r が誤りである確率 $\epsilon_r(s, f, k)$ の上限を引いて得ることが出来る．以下，$\epsilon_r(s, f, k)$ を求め，さらに $\epsilon_r(s, f, k)$ の上限を求めることで，拡張した信頼度計算式を得る．また，$\epsilon_r(s, f, k)$ の上限は，任意の s, f に対する上限として求めることで，どのような s, f の値の場合であっても $\epsilon \leq \epsilon_{acc}$ を満足できるようにする．

まず，ブラックリストが有効の場合，リザルト r は，非妨害者か，抜取検査を k 回，確率 $(1-s)^k$ で通過した妨害者のどちらかにより生成されている．よって，$\epsilon_r(s, f, k)$ は次式で与えられる．

$$\epsilon_r(s, f, k) = s \times \frac{(1-s)^k \times \sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i}}{\sum_{i=1}^{\lceil (1-f)\times W\rceil} p_{nosb_i} + (1-s)^k \sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i}} \quad (8)$$

以下，$k \neq 0$ の場合について，任意の s, f についての $\epsilon_r(s, f, k)$ の上限を求める．まず，$(1-s)^k$ 及び $p_{sb_1}, \ldots, p_{sb_{\lfloor f\times W\rfloor}}$ が全て 0 以上であることから，式 (8) の上限として以下を得る．

$$\epsilon_r(s, f, k) \leq s \times \frac{(1-s)^k \times \sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i}}{\sum_{i=1}^{\lceil (1-f)\times W\rceil} p_{nosb_i}} \quad (9)$$

次に，全妨害者の性能値と，全非妨害者の性能値の合計は，全ワーカの性能値の合計であることを利用する．

$$\sum_{i=1}^{\lceil (1-f)\times W\rceil} p_{nosb_i} + \sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i} = \sum_{i=1}^{W} p_i \quad (10)$$

式 (10) を用いて式 (9) を変形することで，以下が得られる．

$$\epsilon_r(s, f, k) \leq s(1-s)^k \times \frac{\frac{\sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i}}{\sum_{i=1}^{W} p_i}}{1 - \frac{\sum_{i=1}^{\lfloor f\times W\rfloor} p_{sb_i}}{\sum_{i=1}^{W} p_i}} \quad (11)$$

ここで，式 (11) には，妨害者の性能 p_{sb_i} が含まれているが，実際にはどのワーカが妨害者か分からないため，それぞれの p_{sb_i} の値を知ることはできない．そこで本稿では，全ワーカを性能順に並べ替え，性能が高い順に $\lfloor f_{max} \times W \rfloor$ 台を妨害者とみなすことで，p_{sb_i} の合計値の上限を求めることとした．すなわち，全ワーカを性能の高い順に $p_{ordered_1}, \ldots, p_{ordered_W}$ とすると，$\sum p_{sb_i} \leq \sum p_{ordered_i} (i = 1, \ldots, \lfloor f \times W \rfloor)$ かつ $f \leq f_{max}$ であることから以下が成立する．

$$\sum_{i=1}^{\lfloor f \times W \rfloor} p_{sb_i} \leq \sum_{i=1}^{\lfloor f_{max} \times W \rfloor} p_{ordered_i} \tag{12}$$

ここで，表記の簡略化のため，全ワーカの性能値の合計に占める，妨害者の性能値の合計の比率を考え，その上限 r_{max} を以下のように定義する．

$$r_{max} = \sum_{i=1}^{\lfloor f_{max} \times W \rfloor} p_{ordered_i} / \sum_{i=1}^{W} p_i \tag{13}$$

式 (11) と (12) から，r_{max} を用いて以下が得られる．

$$\epsilon_r(s, f, k) \leq s(1-s)^k \times \frac{r_{max}}{1 - r_{max}} \tag{14}$$

また，$s(1-s)^k$ は $0 \leq s \leq 1$ の範囲で $s = \frac{1}{(1+k)}$ で最大となる事と，任意の自然数 k に対して $(\frac{k}{1+k})^k$ の上界が $\frac{1+k}{ke}$ であること [7] から，以下が得られる．

$$\begin{aligned} s(1-s)^k &\leq \frac{1}{(1+k)}(1 - \frac{1}{1+k})^k \\ &= \frac{1}{1+k}(\frac{k}{1+k})^k \\ &\leq \frac{1}{ke} \end{aligned} \tag{15}$$

よって式 (14)，(15) から，$\epsilon_r(s, f, k)$ の上限は以下で与えられる．

$$\epsilon_r(s, f, k) \leq \frac{1}{ke} \times \frac{r_{max}}{1 - r_{max}} \tag{16}$$

また，$k = 0$ の場合，式 (8) に $k = 0$ を代入して式 (17) が得られる．

$$\begin{aligned} \epsilon_r(s, f, 0) &= s \times \frac{\sum_{i=1}^{\lfloor f \times W \rfloor} p_{sb_i}}{\sum_{i=1}^{W} p_i} \\ &\leq s \times r_{max} \\ &\leq r_{max} \end{aligned} \tag{17}$$

式 (16)，(17) を用いて，ワーカの信頼度は，1 から $\epsilon_r(s, f, k)$ の上限を引いた値として，式 (18) で与えられる．

$$C_W(w)_{bl_{proposed}} = \begin{cases} 1 - r_{max}, & \text{if } k = 0, \\ 1 - \frac{r_{max}}{1 - r_{max}} \times \frac{1}{ke}, & \text{otherwise.} \end{cases} \tag{18}$$

次に，ブラックリスト無効の場合も，同様の手順で $\epsilon_r(s, f, k)$ の上限を計算することで，ワーカの信頼度が式 (19) で与えられる．

$$C_W(w)_{no-bl_{proposed}} = \begin{cases} 1 - r_{max}, & \text{if } k = 0, \\ 1 - \frac{r_{max}}{k}, & \text{otherwise.} \end{cases} \tag{19}$$

式 (18)，(19) を，従来の信頼度計算式 (1)，(2) と比較すると，それぞれ f_{max} を r_{max} に置き換えたものになっていることが分かる．r_{max} の定義式 (13) において，全ワーカの性能を均一 $(p_1 = p_2 =, \ldots, = p_W)$ と仮定すれば従来式が得られ，提案式が，ワーカの性能差を考慮した場合の拡張となっていることが分かる．なお，リザルト，リザルトグループ，ジョブの信頼度の計算式は従来式と同様である．

5. シミュレーションによる評価

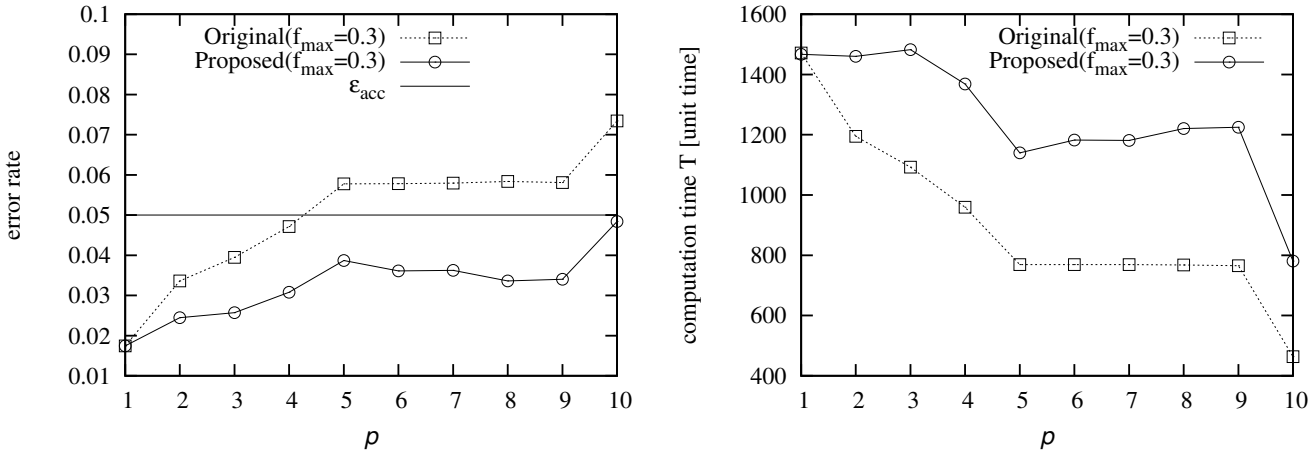
5.1 本シミュレーションの狙い

シミュレーションにより，本稿で拡張した信頼度計算式の有効性を検証する．ここでは，ワーカの性能が不均一である場合に，従来の信頼度計算式を用いた場合，および，拡張した信頼度計算式を用いた場合の誤り率 ϵ，計算時間 T を評価する．特に，誤り率がより大きくなると考えられる，妨害者の方が非妨害者よりも性能が高い場合に注目し，そのような場合にも，拡張した信頼度計算式により信頼性条件 $\epsilon \leq \epsilon_{acc}$ を満足することができることを確かめる．

表 1 にシミュレーションに用いたパラメータの値を示す．各ワーカの性能として，非妨害者の性能を基準として 1 と設定し，これに対して各妨害者の性能をその p 倍とした（p:性能比）．また，ワーストケースを考慮して，ブラックリストは無効とし，抜取検査で検出された妨害者は別のワーカ ID を用いて再参加するようにした．更に，ジョブの実行順序が計算時間に与える影響を小さくするため，ジョブスケジューリング法として**ランダム法**を用いた．ランダム法では，ワーカがマスタにジョブを要求した際，未完了ジョブの中からランダムで 1 つを受け取る．

表 1 シミュレーションパラメータ
Simulation parameters

ジョブ数 N	10000
ワーカ数 W	100
チェック率 q	0.1
許容誤り率 ϵ_{acc}	0.001 ～ 0.1
妨害者割合 f	0 ～ 0.95
妨害者割合の上限 f_{max}	0.1, 0.3
妨害率 s	0 ～ 1
非妨害者の性能	1
妨害者の性能 p	1 ～ 10
1 ジョブの計算量	10



(a) 誤り率 ϵ　　(b) 計算時間 T

図 2　性能比 p に対する結果 ($\epsilon_{acc} = 0.05$, $s = 0.1$, $f = f_{max} = 0.3$)

5.2　シミュレーション結果

5.2.1　異なる性能比に対する結果

図 2 に，横軸に性能比 p を取った場合の誤り率と計算時間を示す．図中の Original は従来の信頼度計算式を用いた場合，Proposed は拡張した計算式を用いた場合の結果を示す．図 2(a) から，従来式，提案式のどちらを用いた場合でも，妨害者の性能 p が大きくなるにつれて誤り率も増大する傾向が分かる．１つのジョブの計算量が１０であることから，ジョブの計算を完了するまでの時間は，性能１の場合は１０単位時間，性能２の場合は５単位時間，性能５～９の場合は２単位時間となる．このため，性能 p が大きくなるほど，妨害者によってより短い単位時間で誤りが生成され誤りの数が増大し，誤り率が高くなると考えられる．

図 2(a) から，従来式を用いた場合，p が５以上になると誤り率が許容誤り率を上回ってしまい，信頼性条件 $\epsilon \leq \epsilon_{acc}$ が満足されていないことが分かる．これに対して，提案式を用いた場合では，p の値に関わらず，常に信頼性条件が満足されている．特に，p が１０の場合では，全ての妨害者がそれぞれ１単位時間でリザルトを生成するため，リザルトの大多数が妨害者によって生成される状態となる．提案式では，このような場合においても適切に信頼度を計算することで，誤り率を許容誤り率以下に抑えることができている．

しかし，図 2(b) に示すように計算時間では，提案式を用いた場合，従来式を用いた場合と比べて約１．５～２倍と，大幅に増加することが分かる．これは，提案式を用いると，各ワーカに与えられる信頼度の値が従来式の場合よりも小さくなり，各ジョブの完了により多くのリザルトが必要となるためである．例えば，$f = f_{max} = 0.3$ で $p = 10$ の時，式 (13) より $r_{max} = 0.3 \times 10/(0.3 \times 10 + 0.7) = 0.81$ となり，f_{max} の代わりに r_{max} を用いる提案式 (19) では，ワーカが従来式と同じ信頼度を得るためには $0.81/0.3 = 2.7$ 倍の大きさの k が必要となる．f_{max} と r_{max} の差は，妨害者の性能 p と共に増大するため，p が大きくなるほど，従来式と提案式を用いた場合の計算時間の違いも大きくなってしまう．そのため，今後の課題として，信頼性条件を満足しつつ計算時間を小さくできるよう，スケジューリング方法や検査頻度 q のパラメータを適正化することが挙げられる．

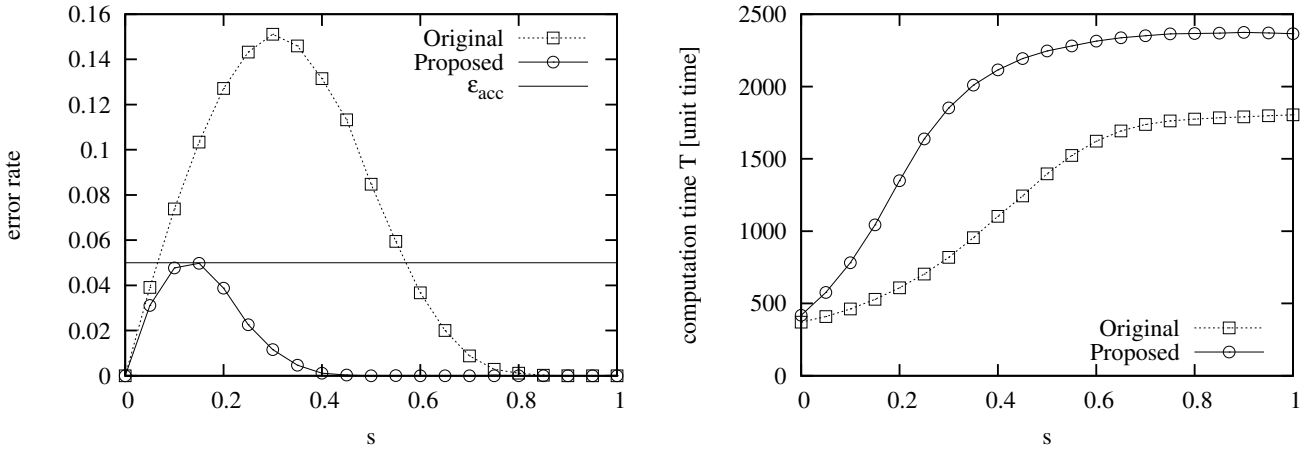
5.2.2　異なる妨害率に対する結果

図 3 に，横軸に妨害率 s を取った場合の誤り率と計算時間を示す．ここで性能比として，図 2(a) において誤り率が最も大きくなる最悪ケースである，$p = 10$ を用いている．

図 3(a) から，従来式，提案式のいずれを用いた場合でも，妨害率に対して誤り率が山なりに変化していることが分かる．これは，妨害率が小さい場合は生成される誤りが少なく，妨害率が大きい場合は抜取検査によって妨害者が検出されやすくなるためである．各妨害者は，抜取検査に対しても確率 s で誤りを返すため，例えば $s = 1$ の場合には１回の抜取検査によって確実に検出されてしまい，それまでに生成したリザルトが無効化される．このため，従来式では $s = 0.3$ 付近，提案式では $s = 0.1$ 付近と，妨害率が比較的小さい，生成する誤りの数と検出されやすさのバランスが取れた所に誤り率のピークが生まれる．

妨害率 s の値は，マスタにとって未知の値であり，また上記のように「どこに誤り率のピークが来るか分からない」という性質を持つパラメータである．このような未知パラメータに対して，取りうる範囲のどのような値の場合であろうと，常に条件 $\epsilon \leq \epsilon_{acc}$ を満足する必要がある．図 3 から，提案式を用いることで，妨害者がどのような妨害率の値を用いても，常にこの条件を満足できることが分かる．

また，図 3(b) で示されるように，提案式を用いた場合には，計算時間が大幅に増大することが分かる．特に，$s \geq 0.7$ の範囲では，従来式においても，誤り率が大きいために妨



(a) 誤り率 ϵ　　(b) 計算時間 T

図 3　妨害率 s に対する結果 ($\epsilon_{acc} = 0.05$, $p = 10$, $f = f_{max} = 0.3$)

害者が容易に抜取検査で検出され，妨害者の生成したリザルトはほぼ全て無効化されている（このため，誤り率もほぼ 0 となっている）．よってこの範囲では，妨害者による影響はほぼ無いにも関わらず，図 2(b) の場合と同様の理由で，従来式と提案式で大きな計算時間の差が生まれている．実際の DG では，このような妨害率が大きい場合は稀と考えられるが，このような場合に特化した計算時間の削減も，今後の課題として挙げられる．

5.2.3　異なる許容誤り率に対する結果

図 4 に，横軸に許容誤り率 ϵ_{acc} を取った場合の誤り率と計算時間を示す．許容誤り率 ϵ_{acc} は DG のユーザが任意に与えるパラメータであり，どのような値の場合であっても $\epsilon \leq \epsilon_{acc}$ を満足しなければならない．図 4(a) に示されるように，提案式を用いることで常に $\epsilon \leq \epsilon_{acc}$ が満足され，信頼性保証が実現できていることが分かる．

また，図 4(b) に示されるように，許容誤り率 ϵ_{acc} の値は計算時間に大きな影響を与える．このため，ϵ_{acc} の値は，計算の種類やアプリケーションに応じて，許容可能な範囲でできるだけ大きく設定するべきである．

5.2.4　異なる妨害者割合に対する結果

図 5 に，横軸に妨害者割合 f を取った場合の誤り率と計算時間を示す．これらの図では，f の上限としてそれぞれ $f_{max} = 0.1$ と $f_{max} = 0.3$ を想定した場合の結果を示している．信頼度に基づく多数決法では，条件 $f \leq f_{max}$ の下で信頼性保証を行うことを想定しているが，従来式を用いると，例えば $f_{max} = 0.3$ の場合の誤り率は $f = 0.1$ で 0.048, $f = 0.15$ で 0.058, $f = 0.2$ で 0.065 と，$f \leq f_{max}$ であるにも関わらず，許容誤り率 $\epsilon_{acc} = 0.05$ を超えていることが分かる．

これに対して，提案式を用いると，例えば $f_{max} = 0.3$ の場合の誤り率は $f = 0.2$ で 0.04, $f = 0.3$ で 0.049 と，誤り率が $\epsilon_{acc} = 0.05$ 以下となっていることが分かる．また，$f_{max} = 0.1$ の場合の $0.2 \leq f \leq 0.95$ の範囲のように，提案式を用いても誤り率が $\epsilon_{acc} = 0.05$ を超えている場合があるが，これは，f が前提条件である $f \leq f_{max}$ を満たしていないためである．この図から，$f \leq f_{max}$ の範囲では，提案式を用いることで常に $\epsilon \leq \epsilon_{acc}$ となっていることが確認できる．

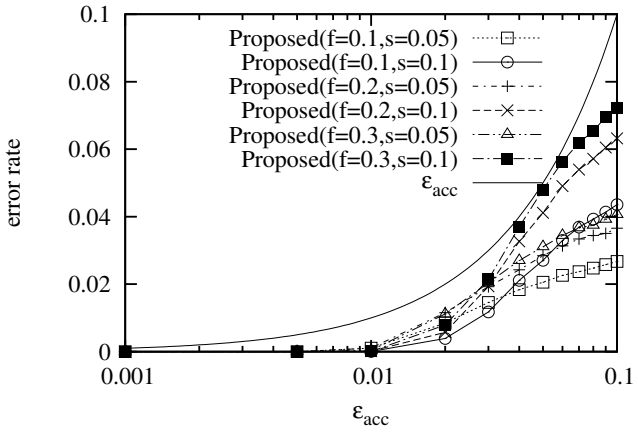
また，提案式では，f が 0.5 を超えた範囲で，f が大きくなるにつれて誤り率が減少するといった現象が起きていることが分かる．この理由は，f が大きくなるにつれ，性能の高い妨害者が増えて計算時間が小さくなり，結果として妨害者によって生成される誤りの絶対数が減少するためであると考えられる．

図 5(b) を見ると，$f_{max} = 0.1$ と $f_{max} = 0.3$ で計算時間の違いがそれほど大きくないことが分かる．実際の f の値は未知であるため，$f \leq f_{max}$ を確実に満たせるよう，統計的な結果から推測されるよりは少し大きめの f_{max} を設定するのが良いと考えられる．

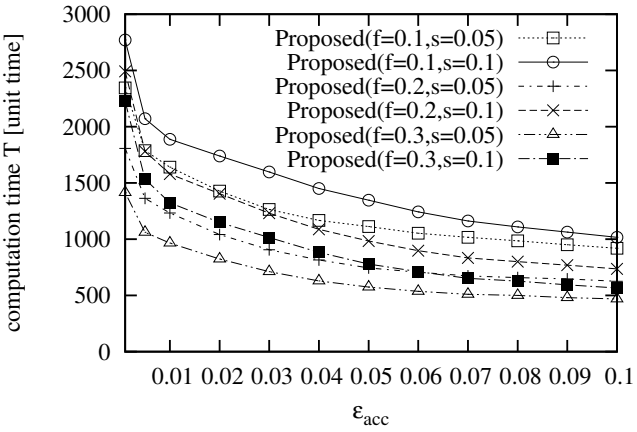
6.　まとめと今後の課題

本稿では，各ワーカの性能が不均一となるデスクトップグリッドにおいて，信頼度に基づく多数決法を適用可能とするため，信頼度計算式の拡張を行った．拡張した計算式では，各妨害者の性能が分からない場合でも，その性能合計比率の上限 r_{max} を用いることで，信頼度計算を可能としている．シミュレーションにより，従来の計算式では，計算結果の誤り率を許容値 ϵ_{acc} 以下に抑えることができない場合にも，拡張した計算式を用いることでこれが可能となることを示した．今後の課題には，誤り率を許容値 ϵ_{acc} 以下に収めつつ計算時間を削減するよう，ジョブスケジューリングや抜取検査の頻度を適正化することが挙げられる．

謝辞　本研究の一部は科学研究費補助金の研究活動スタート支援（23800041）の助成による．

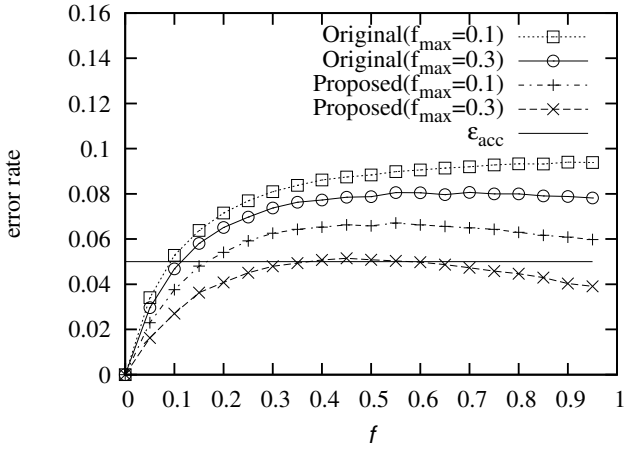


(a) 誤り率 ϵ

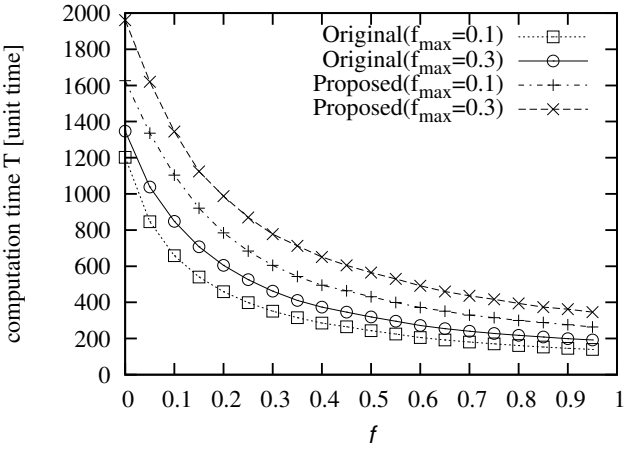


(b) 計算時間 T

図 4 許容誤り率 ϵ_{acc} に対する結果 ($p = 10$, $f_{max} = 0.3$)



(a) 誤り率 ϵ



(b) 計算時間 T

図 5 妨害者割合 f に対する結果 ($\epsilon_{acc} = 0.05$, $p = 10$, $s = 0.1$)

参考文献

[1] SETI@home http://setiathome.ssl.berkeley.edu/

[2] D. Kondo, F. Araujo, P. Malecot, P. Domingues, L. M. Silva, G. Fedak, and F. Cappello, “Characterizing Error Rates in Internet Desktop Grids”, 13th European Conf. Parallel and Distributed Comput., pp. 361–371, 2007.

[3] L. F. G. Sarmenta, “Sabotage-Tolerance Mechanisms for Volunteer Computing Systems”, Future Generation Computer Systems, Vol. 18, Issue 4, pp.561-572, 2002.

[4] F. Araujo, J. Farinha, P. Domingues, G.C. Silaghi, and D. Kondo, “A Maximum Independent Set Approach for Collusion Detection in Voting Pools”, J. Parallel and Distributed Comput., Vol. 71 (10), pp. 1356 – 1366, 2011.

[5] BOINC http://boinc.berkeley.edu/

[6] K. Watanabe, M. Fukushi and S. Horiguchi, “Expected-credibility-based Job Scheduling for Reliable Volunteer Computing”, IEICE Trans. Inf.& Syst., Vol.E93-D, No.2, pp.306 – 314, 2010.

[7] K. Watanabe, M. Fukushi, and M. Kameyama, “Adaptive Group-Based Job Scheduling for High Performance and Reliable Volunteer Computing”, J. Information Processing, Vol.19, pp.39 – 51, 2011.

[8] K. Watanabe, M. Fukushi and S. Horiguchi, “Optimal Spot-checking for Computation Time Minimization in Volunteer Computing”, Journal of Grid Computing, Vol. 7, Issue 4, pp.575 – 600, 2009.