A-005

逐次計算の検証を分散処理する手法の実装
An implementation of a distributed verification scheme for serial computation

豊住 洋史† 山本 宙‡
Toyosumi Hiroshi Yamamoto Hiroshi

1. はじめに

近年では単体計算機の能力の限界を高める金銭コストに比べ，低速な計算機を多数製造する金銭コストの方が著しく低い．本研究では高速かつ金銭的に高コストな計算機群と低速かつ金銭的に低コストな計算機群が利用でき，その計算速度と金銭コストはそれぞれ極端に差がある場合を考え，これを仮定する．金銭コストの高い高速な単体計算機を利用する代わりに著しく金銭コストの低い低速計算機を多数利用することで金銭的，計算量的に低いコストで高い総合計算能力を得られる可能性がある．

本研究では効率的に分散処理できない問題で高速計算機を利用しなければ現実的な時間で解くことができない問題に対して検証計算に分散処理を用いて総合的な計算コストを低減させる手法について実装を行った．また計算時間について検証を行った．

2. 提案手法

著者らの研究グループでは計算の正しさを検証する作業に分散処理を用いる手法を提案してきた[1][2]．まずシステムの構成について概説する．計算の検証作業を制御する計算機をシステムサーバとよび，システムサーバは信頼できる低速計算機であると仮定する．利用者はシステムサーバに全体の計算を依頼する．必ずしも信頼できない計算機を用いた計算の正しさを検証する方法として，複数の計算機に同じ計算を依頼して結果が一致すれば正しいと判断する手法が現在用いられている．この手法では正しくない計算が偶然一致することはないという仮定が置かれており，本研究でもこれを仮定する．システムサーバは必ずしも信頼できない高速計算機群，低速計算機群に対して計算の依頼と報告を受ける．これを適宜行い，検証作業が完了すると利用者に結果を返す．計算機の記憶装置，メモリ，変数の内容などある時点の計算機内の状態を表す情報を計算状況とよぶ．システムサーバが高速計算機に依頼する計算を主計算とよぶ．そのとき，一区間あたりの計算ステップ数を決めるなどしてあらかじめ計算にいくつかのチェックポイントを設けておき，チェックポイントごとに計算状況を出力してシステムサーバへ報告させるようにする．システムサーバは高速計算機から逐次得られる計算状況に基づいて低速計算機に対して計算状況と計算すべき区間を入力として与え，小さな区間に対して高速計算機と同じ計算を行わせ，結果の計算状況を報告させる．システムサーバが適当に低速計算機に検証計算を依頼することで，低速かつ金銭的に低コストの計算機を用いたパイプライン処理で検証作業を効率的に処理することができる．これらの技術を使うことで結果検証作業も含めた全体の処理を効率的に行う．提案手法の構成を図 1 に示す．なおシステムサーバ，高速計算機群，低速計算機群はグリッドコンピューティングで構成する．

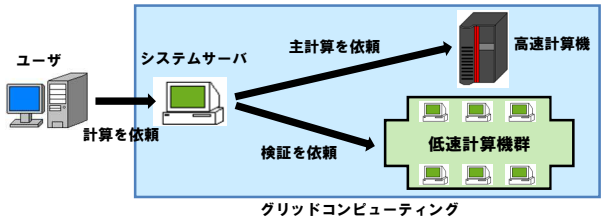

図 1: 提案手法の構成

つぎに具体的な方法を示す．概要説明のため，主計算が正しい場合の説明を最初に行う．システムサーバは計算全体を計算ステップ数によっていくつかの区間に分割し，分割点をチェックポイントとする．全体の計算開始から最初のチェックポイントまでの計算を区間 1 とよび，次のチェックポイントまでの区間を順に区間 2，3，...とよぶ．区間iの始点のチェックポイントでの計算状況と終点でのチェックポイントの計算状況をそれぞれその区間iの始点計算状況，終点計算状況とよぶ．システムサーバはまず高速計算機へ計算を依頼する．このとき，チェックポイントに関する情報も与え，チェックポイントごとにそのチェックポイントを終点とする区間名，始点計算状況，終点計算状況からなるデータ構造体を報告するように指示する．このデータ構造体を報告データとよぶ．同時に複数の低速計算機へ最初の区間の計算を依頼し，高速計算機と同様に区間名，始点計算状況，終点計算状況からなる報告データを送信するように指示する．以降，高速計算機から報告データを受け取るたびにその終点計算状況を始点計算状況とする区間の計算を複数の低速計算機に依頼し，区間の計算を終了後，先述のものと同様の報告データを送信するように指示する．

システムサーバは高速計算機，低速計算機から非同期に種々のチェックポイントの報告データを受信する．システムサーバにはそれぞれの報告データについて，それをもとにそのデータの検証状況を保持するために，始点計算状況が検証済みかどうか，その区間の計算が検証済みかどうか，という 2 つのフィールドを付加し，それぞれの初期値を偽に設定する．これを検証データとして保存する．システムサーバは報告データを受信し，検証データを追加，保存するたびに以下の手続き 1，2 を順に行う．

1．報告を受けた区間の計算に関する検証

追加される検証データと，すでに記録されている同じ区間の検証データをチェックする．チェックした検証データの

† 東海大学大学院工学研究科情報理工学専攻，
〒259-1292 神奈川県平塚市北金目 1117
Graduate School of Engineering Tokai University,
1117 Kitakaname Hiratsuka Kanagawa ZIP: 259-1292
‡ 東海大学情報通信学部通信ネットワーク工学科，
〒108-8619 東京都港区高輪 2-3-23
Dept. of communication and Network Engineering,
School of Information and Telecommunication Engineering,
Tokai University,
2-3-23 Takanawa Minatoku Tokyo ZIP: 108-8619

うち，ある 2 つの検証データについてそれぞれの開始計算状況が等しくかつそれぞれの終点計算状況が等しいものがあれば，この区間の計算を検証済みとして検証データを変更する．このような検証データの組がなければその区間の計算の検証過程については何も処理を行わない．

2．区間を通した検証

- 区間 1 のすべての検証データの始点計算状況を検証済みとする．
- $i=1$から最後の区間番号（Nとする）まで以下の処理を行う．
  - 区間iについて始点計算状況と区間の計算の両方が検証済みのデータがあれば，
    - その区間のそれ以外のデータを破棄．
    - $i<N$のとき，区間iの終点計算状況と等しい始点計算状況をもつ区間$i+1$の始点計算状況を検証済みとする．

手続き 1，2 の終了後，区間Nの始点計算状況と終点区間の計算が検証済みであれば全体の計算が検証済みとなり，ユーザに報告する．複数の低速計算機の区間の計算が一致し，それが主計算と異なる場合は主計算の方が正しくないと判断できる．このような場合は主計算を依頼する計算機を変更するなどの処理が必要である．また依頼した計算機すべてから報告データを受信しても一致する報告データが得られなければ依頼する計算機を増やす処理も必要である．

3．実装

提案手法の実装には Globus Toolkit 2.4.3 を用いた．Globus Toolkit とはグリッドコンピューティングを実現する上で必要となる資源管理，情報サービス，データ転送などの機能を提供するミドルウェアである．この Globus Toolkit を 10 台のパーソナルコンピュータにインストールし，小規模なグリッドコンピューティング環境を構築した．システムサーバの主なプログラムは Java のマルチスレッドプログラムから Globus Toolkit のコマンド群を操作する形で作成し，各計算機に計算を依頼する際にはインタプリタ言語である Perl のファイルを使用した．

いくつかの問題について実際に処理を行い，正しく実行されることを確認した．

4．処理時間の評価

高速計算機が 1 区間を計算するのに必要な時間をT_hとする．低速計算機に関しては不正な計算を行う確率をp，1 区間を計算するのに必要な時間をT_lと一律に定める．mを 1 区間の検証を依頼する低速計算機の台数とすると，ある区間の検証がT_l時間後に正しく終了する確率は$1-p^m$となり，1 台に依頼する場合に帰着できる．このため確率pで不正な結果を返す計算機 1 台に依頼するとして評価を行った．よってある区間についてシステムサーバがi回目の依頼の結果で初めて正しい計算結果が得られる確率は$p^{i-1}(1-p)$となる．ゆえにある区間について最初の正しい解を得られるまでに計算を依頼する低速計算機数の期待値は$\sum_{i=1}^{\infty} ip^{i-1}(1-p)=1/(1-p)$で表すことができる．

提案手法の金銭コストは高速計算機の計算単価と計算時間，低速計算機の計算単価と全低速計算機の延べ計算時間から求めることができる．全低速計算機の延べ計算時間の期待値は$N/(1-p)T_l$である．

システムサーバにユーザが計算を依頼してから検証処理を含めたすべての処理が終了し，システムサーバから計算結果が返されるまでの時間を総合計算時間とよぶ．今回は簡単のため高速計算機が正しく計算を行った場合の評価を行う．検証計算はパイプライン方式で処理される．区間の検証は最初の区間ほど早く終了する傾向がある．総合計算時間については最終区間の検証が最後に完了する場合の計算時間が支配的であると考えられるため，$(N-1)T_h+1/(1-p)T_l$で近似されると考えられる．

前節で予想した総合計算時間の近似式をシミュレーションにより検証する．現実的な値として高速計算機の性能が低速計算機の 100 倍である状況を想定し，T_hを 1 秒，T_lを 100 秒と仮定した．区間数Nは 1000 とした．低速計算機 1 台が不正な計算を行う確率pは 0 から 0.2 までの間で 0.01 ずつ変化させて各確率において計算を 10 回行いその平均値を求めた．シミュレーションプログラムでは単一計算機上でタイマ関数をマルチスレッドで実行する手法で実装した．不正な計算を行う確率と総合計算時間の関係について，前節の近似式とシミュレーション結果を比較したものを図 2 に示す．

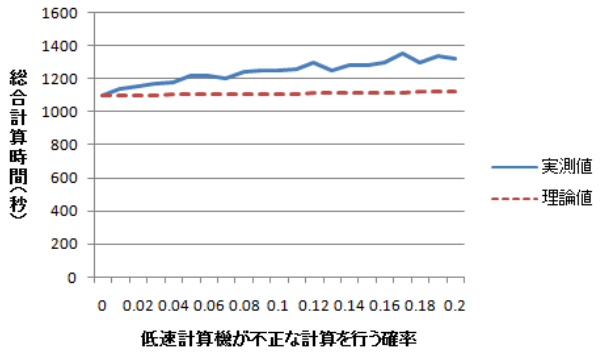

図 2: 測定結果

不正な計算を行う確率が大きいほど近似式よりもシミュレーションによる計算時間が大きくなっている．これは最終区間以外の区間の検証が最後になるケースが増えるためと考えられる．不正な計算を行う確率は実用的なケースでは比較的小さな値であると考えられ，前節の近似式は実用的であると言える．

5．まとめ

今回は検証を含めた計算時間について実験を行い実測値と近似値を比較した．今後は精度の高い乱数の使用や実験データのさらなる収集，より高い性能の計算機上での実験などを行う．

参考文献

[1] 茂木篤他，計算の検証の分散処理に関する一提案"，FIT2006 第 5 回情報科学技術フォーラム，L-022，2006.

[2] Hiroshi Toyosumi，Hiroshi Yamamoto，”A Study on Collusion Security of Distributed Verification”，2007 Hawaii and SITA Joint Conference on Information Theory，May 2007