1K-8

Webブラウザを用いた グリッドコンピューティングフレームワークの研究

川﨑　寛文†　　丸山　広†　　高嶋　章雄†　　中村　太一†

東京工科大学大学院†

1.はじめに

マルチエージェントシミュレーションや多次元データの類似度計算などを安価に実現するため，ネットワークに接続された多数のパソコンやサーバの資源を活用するグリッドコンピューティング(以下，グリッド)の適用が検討されている．しかし，多くのグリッドのアプリケーションは，グリッドの要素として稼働させるために特別な構成を資源側に求めるため，資源を提供する側の負担が大きく，所要のコンピュータ資源を確保することが難しい [1][2].

この問題に対処するため， Web ブラウザに搭載されている JavaScript エンジンを仮想のコンピュータとして利用することで，資源提供者の負担を回避する方法が提案されている．しかし，JavaScript エンジンの性能は Web ブラウザ毎に異なり[3]，そのブラウザを搭載するパソコンの性能も様々であるため,アプリケーションが処理するデータを複数に分割し，同時並行処理させた場合，全体の処理の完了時間を予測することが難しい.

これに対処するため,予め測定しておいた類似の処理に要する時間を予測値とする方法が考えられる.しかしながら，この方法は，Web ブラウザの動作環境が変わる度に，パソコンや JavaScript エンジンの性能の違いを補正しなければならない．また,処理が類似していても扱うデータ量の違いで予測が合わないことがある.

このような問題に対処するため，ジョブを完了するためにアプリケーションプログラムが実行する回数を予め計測し，その回数と実環境でジョブを実行する時間との関係からジョブ完了時間を予測する方法を提案する.

以下，２.で提案方法を述べる．本提案方法の有効性を文書の類似度を評価するコピーレポート判定処理を行いで評価した結果を３．に示す．４．で実験結果の考察を５．に今後の展開を述べる.

Framework for Grid Computing using Web Browser
† Hirofumi Kawasaki †Hiroshi MARUYAMA
† Akio TAKASHIMA †Taichi NAKAMURA; Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology

2. 提案方法

2.1 処理完了予測の方法

ジョブを処理するためにアプリケーションプログラムに予め設定したポイントを通過する回数を n と，ノード j においてそのポイントを通過した時刻の間隔 τ_j とすると，ジョブの処理完了時間 T_jは，式（1）予測できる.

$$T_j = \tau_j \times n \qquad (1)$$

ただし，n は予め計測しておいた値である.
n はパソコンや Web ブラウザには依存せず，ジョブの性質にのみ依存する，他方 τ_j は，ジョブ実行中に計測される値で，実際のグリッドコンピューティング環境に依存する変数である，

すなわち，予め n を求めておくことで，どのような環境でも，ジョブの完了時間を予測できると考える.

2.2 システム概要

図 1 に本提案方法を実装するシステム（GRid computing On the Web-browser：GROW）の構成を示す．資源管理は，ノードの接続や離脱を管理する．ジョブ管理は，ジョブの割り当てや消失したノードからのジョブの回収をする．タイムスタンプ管理は，ノードから送られてきたタイムスタンプの受付をする．処理完了予測は，式(1)を実行し，処理完了時間予測をする.

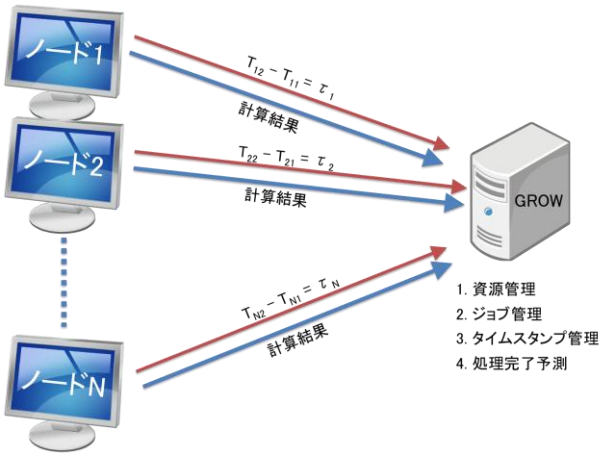

図１　GROW のシステム構成

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.

2.3. 提案方法の実装

ジョブ実行中に，実行時間を求める手法として，チェックポインティングのログから，ジョブが実行中のノードのチェックポイント間の平均実行時間を算出し，実行時間を求めることで，どのノードのジョブがいつ終了するのかを把握できるようになる．

3. 実験

3.1 実験環境

表 1 に GROW を構成するノードの仕様を示す．ノードは 4 台である．

表 1 ノードのパソコンの仕様

項番	環境	ノード
1	CPU	Core 2 Duo 2GHz
2	メモリ	1GB
3	OS	Windows XP 32bit
4	Web ブラウザ	Google Chrome 16

3.2 実験方法

GROW はフレームワークであるため，テストアプリケーションとしてコピーレポート判定のアプリケーションを作成し，実行時間予測を行った．

ノードを 4 台用意し，コピーレポート判定を，実行させ，処理完了時間を予測した．n を 67 として，各ノードの予測時間を求めた．1 ノードあたりの処理完了時間は，176 秒であった，

4. 結果

実行中予測を，図 2 に実行中予測の推移を示す． 縦軸は，$\sqrt{(T_j - T_{j-1})^2}$で求めた処理完了時間予測誤差，横軸は経過時間 T_j(秒)である．

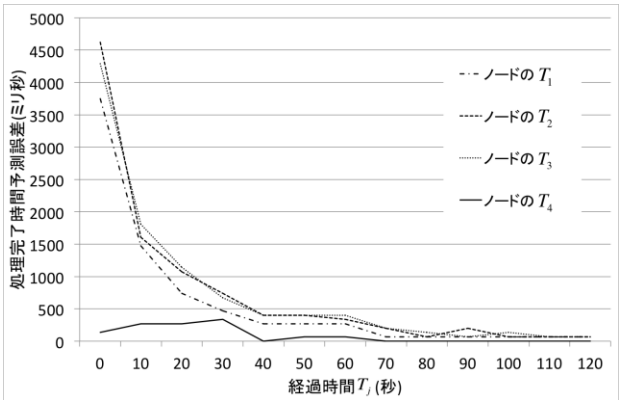

図 2 処理完了時間予測誤差の推移

図 2 から 100 秒程度で予測時間が収束していることがわかる．しかし，ジョブが新たに開始されるときには，予測時間が大きく伸びてしまっている．そのため，ジョブの再割り当てにこの予測手法を導入するためには，収束するまで，ジョブの再割り当てを待つ必要があると考えられる．

実行後予測は，ジョブが割り当てられ，ある程度時間が経てば，正常に予測ができることがわかった．チェックポインティングで収集するログがコピーレポート判定に近いものであれば，予測も近いものになっていたと考えられる．

5. おわりに

本稿では，ジョブ実行中に実行時間を予測する手法と実行前にジョブの実行時間を予測する手法を提案した．しかし，ジョブ実行中に実行時間を予測することは概ね達成されたが，実行前にジョブの実行時間を予測することは，達成できなかった．

実行中予測を使用して，計算の遅い Web ブラウザから計算の速い Web ブラウザへのジョブの移行ができるため，先に述べた計算全体の性能劣化を抑え，より早いグリッドの環境を構築できると考えられる．

今後は，各ノードの処理性能に比例してジョブサイズを動的に変更する手法を提案し，各ノードの処理時間の平準化を実現することで，各ノードの処理能力を使いきることができる．

参考文献

[1] Foster, I.; Yong Zhao; Raicu, I.; Lu, S.; "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop (GCE '08), 2008.

[2] 首藤一幸, “グローバルコンピューティング (9)「ボランティアコンピューティング」”, Computer Today 2001 年 9 月号, pp.65-72, サイエンス社, 2001 年 8 月

[3] Mozilla Foundation, “Dromaeo: JavaScript Performance Testing”, <http://dromaeo.com>, (最終アクセス日：2012/01/13)

Copyright ©2012 Information Processing Society of Japan.
All Rights Reserved.