2J-7

Grid Computing の性能見積もり手法の研究

坂巻雅実† 野々田 峰寛† 伊藤 剛† 中 健一† 中村 太一†
東京工科大学†

1. はじめに

Grid Computing (以下 GRID とする)は種々の性能のパソコン(以下 PC とする)で構成され, それらのパソコンの使用率は通常の業務の負荷によって変動する．このような GRID に実装した業務処理の応答時間やスループットを保証するために，性能見積もり方法を明確にする必要がある．本稿では，GRID システム上で性能評価用の分散アプリケーションの実行時間を測定し, 待ち行列モデルから記述した見積もり結果と比較することで，待ち行列モデルによる GRID システムの動作モデル記述方法の妥当性を示す．

2. 考慮する GRID システム

GRID システムはマスタとスレーブから構成される．マスタは業務処理をジョブに分割してスレーブに送り，スレーブからのジョブの処理結果を収集してまとめる．スレーブはマスタから送られた処理を実行し, 結果をマスタに返す．

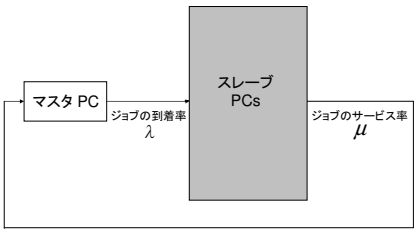

図 1　GRID システムの構成

待ち行列モデルを用いて GRID システムの振る舞いを記述するに当たり，マスタが単位時間当たりにスレーブに送るジョブ数をジョブの到着率 λ，スレーブが単位時間当たりに処理できるジョブ数をジョブのサービス率 μ とする．

GRID システムを用いた処理として，マスタがジョブをバッチ処理としてスレーブに投げる処理モデルと，スレーブに対してマスタがオンライントランザクション処理を行う処理モデルが挙げられる．

GRID システムの構築には, GRID ミドルウェアに Globus Toolkit を，分散処理実行環境ミドルウェアに MPICH-G2 を用いる．

3. 分散処理オーバーヘッドの把握

GRID システムのオーバーヘッドとして, MPI 通信関数の実行時間が挙げられる．このうち，初期化関数である MPI_Init 関数の実行時間と比べ，他の関数の実行時間は MPI_Init 関数の 10^{-5} %程度であった事から，MPI_Init 関数に注目する．通信先となる PC の CPU のクロック周波数と MPI_Init 関数の実行時間の関係を図 2 に示す．

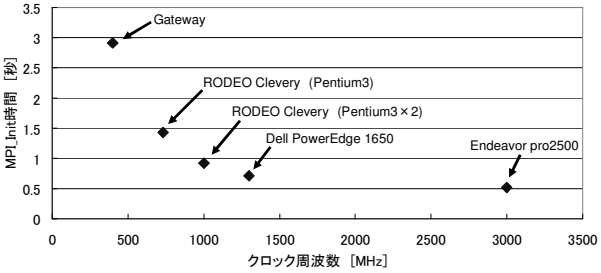

図 2 クロック周波数と MPI_Init 関数の時間

スレーブが n 台の PC で構成される時，MPI_Init 関数の実行時間 $I(n)$ を式(1)で定義する．n 台の PC の CPU のクロック周波数に対応した MPI_Init 関数の実行時間を $I_i(i=1,2,..,n)$ と表し，図 2 から値を得るものとしている．

$$I(n)=\sum_{i=1}^{n} I_i \quad \cdots(1)$$

式(1)を利用した MPI_Init 関数の実行時間の予測値と，実測値との比較を図 3 に示す．

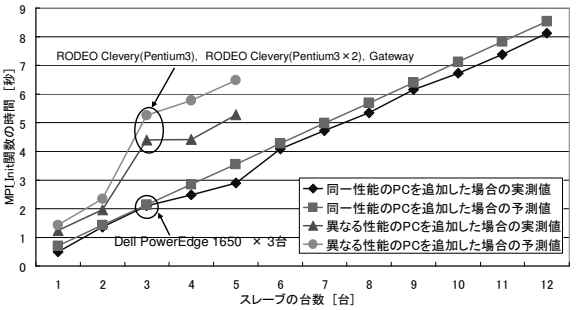

図 3　MPI_Init 時間の予測値と実測値の比較

Performance estimation for Grid Computing
†Masami SAKAMAKI †Takahiro NONODA †Tuyoshi ITO †Kenichi NAKA †Taichi NAKAMURA:Tokyo University of Technology

4. 分散処理時間短縮のための PC の台数制限

4. 1 バッチ処理

この処理モデルではスレーブの処理能力に関係なくジョブが一定間隔でスレーブに送信される(λは固定となる). スレーブを構成する各 PC の性能差に応じたジョブの振り分けを行うことが難しいため，各 PC へ割り振るジョブ数は同数とする. PC の台数を加算することでスレーブの μ を変化させた時，スレーブの待ち行列に溜まるジョブの数を測定した結果を図 4 に示す. 図 4 では，スレーブを構成する PC の台数が 4 台までの時に $\lambda > \mu$ の関係が成り立ち, 5 台以降は $\lambda < \mu$ の関係が成り立っている.

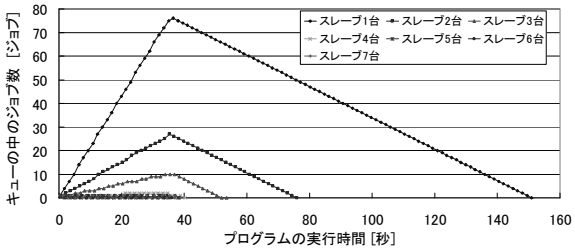

図 4 スレーブの待ち行列に溜まるジョブ数

スレーブを構成する PC 台数が 4 台までの時，アプリケーションの実行開始から待ち行列に溜まるジョブが一時的に増大する．マスタからのジョブ送信が終了すると待ち行列の中のジョブ数は減少傾向となり，やがて処理が完了する．5 台以降は，ジョブが待ち行列に溜まることがなくなり，実行時間のうち分散処理時間は，マスタが全てのジョブをスレーブに送信する時間となる．そのため台数を増やしても分散処理時間は飽和し，実行時間はオーバーヘッドの分だけ増加する(図 5)．また，この処理モデルでは PC の性能差に応じたジョブの振り分けが難しく，同数のジョブを各 PC に送信するために，異なる性能の PC をスレーブに加えていった時，PC の性能差が影響して分散処理時間が増減する.

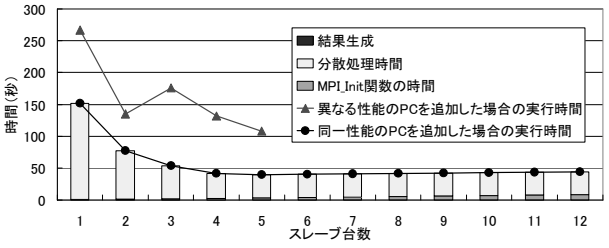

図 5 実行時間の増減と各処理時間

4. 2 オンライントランザクション処理

この処理モデルはマスタが処理の完了した PC に対して次のジョブを送信する処理モデルとなる．スレーブを構成する PC の台数と μ の実測結果を図 6 に示す.

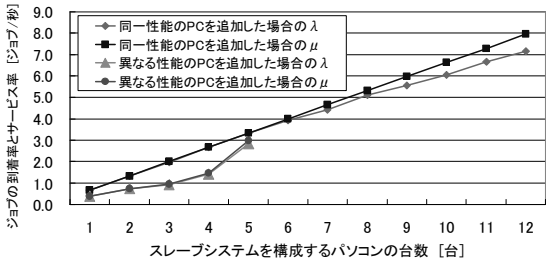

図 6 PC 台数が増加する時の λ と μ の比較

この処理モデルでは，スレーブに PC を加算していった時，分散処理時間の飽和は見られなかった．また，PC の性能差に応じたジョブの振り分けを自動的に行うことになるので，異なる性能の PC をスレーブに追加しても，PC の性能差が分散処理時間の増加原因にはならないことがわかる.

5. おわりに

本稿では, Globus Toolkit と MPICH-G2 ライブラリを利用して GRID を構築した際の主なオーバーヘッドとして，MPI_Init 関数の時間を把握した．また，処理モデルの違いによる，分散処理時間を短縮するための PC の台数の制限の違いと，マスタからスレーブを構成する各 PC へのジョブの振り分け方の違いを把握した.

今後はさらに PC の利用状況が動的に変化する場合のスレーブのジョブのサービス率 μ を導出することと，複数の異なる種類の業務処理を同時に GRID システムに投げた場合の処理モデルの表現方法を明確にすることが課題となる.

参考文献

[1] 日本アイ・ビー・エム　システムズ・エンジニアリング株式会社著：グリッドコンピューティングとは何か Globus Toolkit ではじめるグリッドの基礎，2004.4

[2] Ian Foster, Carl Kesselman : The Grid2:Blueprint for a New Computing Infrastructure，2003

[3] Daniel A. Menasce, Virgilio A.F. Almeida : Capacity Planning for WEB PERFORMANCE，1998