マルチスレッドに着目したクライアント／サーバシステムの性能評価モデルとその測定手法の検討

井上哲哉　勅使河原可海

創価大学大学院工学研究科

e-mail: tinoue@edu.t.soka.ac.jp

概要

近年の Web 技術の動向として，Web ブラウザをもつクライアントでアプレットなどを実行させるよりも，クライアントのサービス要求をサーバ側に任せるシステムが多くなってきている。このようなシステムのサーバ環境の多くは，Java の基盤技術をもち，複数のコンポーネントが存在する。

本稿では，サーバ側に複数のコンポーネントを置く Web 環境を視野に入れ，簡単なクライアント／サーバシステムを想定して，プロセスの挙動を考慮した分散システムとしての性能評価モデルを提案する。分散システムの性能評価モデルを構築するために，システムの構成要素と特性要因を明確にし，ネットワークの状態を含むシステム的性能と，特定した評価対象のプロセスの実行時間等をマルチスレッドを用いて測定する手法を提示する。

A study of a performance evaluation model and its measurement techniques focused on multithread in client-server systems

Tetsuya Inoue　Yoshimi Teshigawra

Graduate School of Engineering, Soka University

Abstract

Recently, as shown by the trend of Web technology, the number of systems which commits service requests from clients to server sides have increased rather than those which execute Applets in the clients with Web browsers. In the most of sever environment like these Java based technology is used and various components exist.

This paper takes into consideration of Web environment where multiple components exist in a server side, assumes a simple client-server system, and proposes a performance evaluation model focused on process behaviors for distributed systems. In order to build the model for distributed systems, it clarifies system configuration elements such as hardware, software and networks, and their characteristic factors. In addition, it indicates performance evaluation techniques using multithread to measure system performance including network factors and run time of processes specified as evaluation targets.

1．はじめに

システム全体の性能を評価したい場合，システムの各構成要素の性能値を合成して求めることが望まれる[1]。各構成要素には，ハードウェア的な構成要素のほかにシステムで実行されるソフトウェア（プロセス）や利用者も含まれる。Web 技術の基本であるクライアント／サーバシステムのプロセス間通信では，クライアントプロセスは

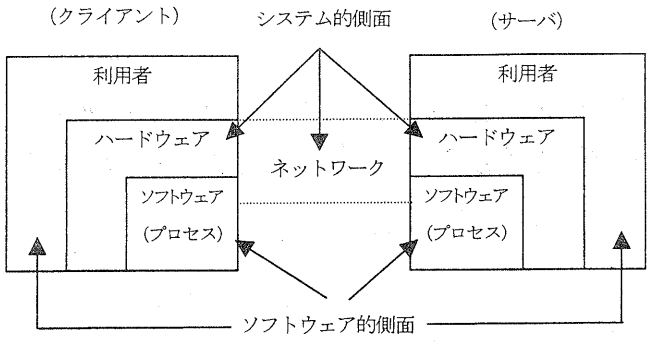

図1　簡単なクライアント／サーバシステムと構成要素

サーバからのサービスを要求するプロセスであり，サーバプロセスはクライアントにサービスを提供するプロセスである。通信ネットワークを介したプロセス間通信の性能を評価するには，ハードウェアやネットワークの各構成要素の性能測定値とともに，それらのリソースを使用するプロセスの性能測定値も重要となってくる。

本稿では，今日の Web 技術の動向から，サーバ側に複数のコンポーネントを置く Web 環境を視野に入れ，簡単なクライアント／サーバシステムを想定して，プロセスの挙動を考慮した分散システムとしての性能評価モデルを提案する。このモデルは，マルチスレッドを用いて実現し，３つのスレッドを使って，ハードウェアやネットワークの状態を含むシステム的側面からの測定値と，特定したプロセスの挙動を考慮したソフトウェア的側面からの測定値を抽出し，クライアントがサービスを要求するごとに分散システムとしての性能を測定する。

2．システムの構成要素と特性要因

2.1 システムの構成要素

図1に示すような簡単なクライアント／サーバシステムにおいて，システムの構成要素をハードウェア，ネットワーク，ソフトウェア（プロセス），そして利用者の４つの要素に分けることにする（**表1**参照）。

2.2 特性要因

表１に各構成要素に対する主な特性要因を示す。特性要因とは，構成要素の品質に影響を与える要因と定義する。図１に示すように，ハードウェアとネットワークは，システム的側面とし，ソフトウェア（プロセス）と利用者は，ソフトウェア的側面とする。次項では，システム的側面とソフトウェア的側面のそれぞれのアプローチから，システムの各構成要素の特性を決定づける要因について述べる。

表1　システムの構成要素と主な特性要因

構成要素	主な特性要因
ハードウェア	メモリ容量，CPU 利用率，ディスク利用状況（ファイルシステム），カーネルパラメータ値
ネットワーク	受信バッファサイズ，通信回数，帯域幅，遅延
ソフトウェア（プロセス）	実行時間，資源利用状況，故障数，フォールト数
利用者	データ入力時間（思考時間含む）

3．システム的側面のアプローチ

ハードウェアの特性要因は，システムの性能を決定づける要因として，メモリ容量と CPU 利用率，ディスクの利用状況，およびカーネルパラメータ値を特定する。これらの特性要因は，１台のハードウェアに装着されている物理的な要因として挙げている。ただし，実際のシステムの性能

評価は，ソフトウェア（プロセス）がハードウェアの資源を利用して動作することから，ソフトウェア的側面からプロセス単位に評価する。ディスクの利用状況は，例えば，UNIX の df コマンドで得られる情報と同等であり，NFS（Network File System）を含むデバイスファイル（ファイルシステム）ごとの資源の利用状況である。カーネルのパラメータ値は，ソフトウェアを利用者の期待通りに動作させるためにたいへん重要であり，カーネルパラメータ値のセマフォ数や，１つのプロセスが利用可能なスレッド数などは，プロセスの挙動に影響を与えることは言うまでもない。

次に，ネットワークの特性要因は，通信時間を決定づける要因として，受信バッファサイズ，通信回数，帯域幅，および遅延を特定する。受信バッファサイズは，サーバ側がクライアントからの送信バッファを受信バッファとして受け取ったサイズである。バッファとは，ここでは，データとメッセージの２つの転送量を含む。すなわち，データはメガバイト単位の転送量を意味し，メッセージは，１バイト単位の転送量を意味している。

帯域幅は，あるプロセスから別のあるプロセスへデータが移動する速度であり，次の式で表される。

$$\text{帯域幅} = \frac{\text{総転送量}}{\text{転送に要した時間} \times \text{転送回数}}$$

また，遅延は，メッセージが２つのプロセス間を往復するのに要する時間であり，クライアントがサーバに対して１バイト単位のメッセージを送り，サーバからの応答を受け取るまでの時間となる。

4．ソフトウェア的側面のアプローチ

ソフトウェアの特性要因は，その対象をプロセス単位とし，プロセス自身の挙動を決定づける要因として，プロセスの実行時間，プロセスが使っている資源の利用状況を特定する。プロセスの資源利用状況（表１の下線部）は，前述したハードウェアの特性要因に基づいて，プロセス単位にそれらの情報を得る。具体的には，プロセスが消費したメモリ容量や，プロセスの CPU 利用率を得ることである。

また，ソフトウェア工学の視点からソフトウェア（プロセス）の故障数とフォールト数もソフトウェアの特性要因として挙げている。ソフトウェア故障とは，ソフトウェアが利用者の期待通りに動作しないことであり[2]，ソフトウェアフォールトとは，ソフトウェア故障を引き起こしたプログラム内の欠陥や誤りである[2]。

最後に，利用者の特性要因として，データの入力時間を特定した。この入力時間には，利用者の思考時間も含まれている。これらの時間は，クライアントのプロセスが起動した時間（開始通知）から終了した時間（終了通知）までの時間差である（**図２**参照）。

5．特性要因の抽出方法

5.1 監視プロセス

これまで，システムの構成要素とその特性要因について述べてきた。本項では，これらの特性要因の抽出方法について述べる。また，その抽出方法を述べるにあたって，**図３**のようなシステム構成を考える。図３では，クライアントとサーバ間は，ポート番号 6500 番を使って TCP 接続している。クライアントのプロセスがサーバへサービスを要求したとき，サーバ側に置かれている監視

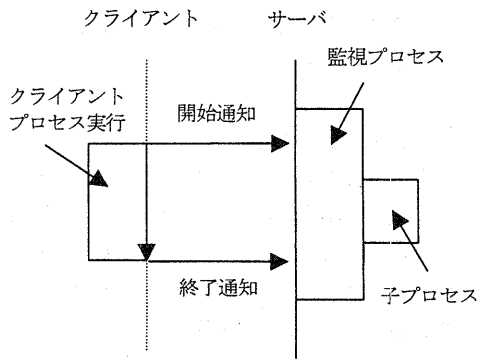

図２　クライアントの開始通知と終了通知

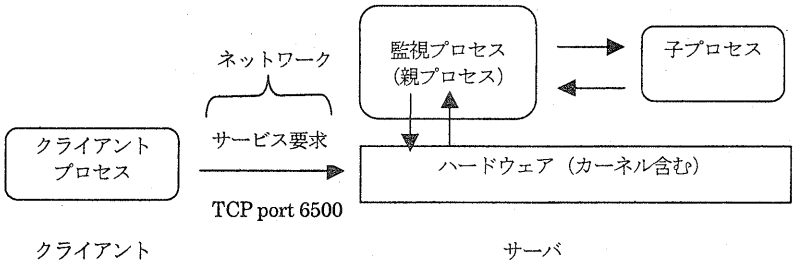

図3　TCP 接続のクライアント／サーバシステム

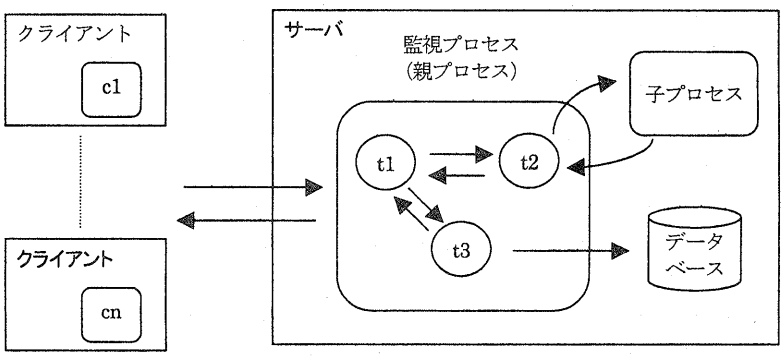

図4　マルチスレッドを用いた性能評価モデル

プロセスは，クライアント情報として，クライアントのホスト名や IP アドレス，クライアントが送信したバッファなどを取得する。

監視プロセスは，これらの情報を正常に処理すると，クライアントから要求されたサービスを子プロセスとして生成する。そして，次項で述べる３つのスレッドを用いて特性要因の抽出を行い，子プロセスに依存するシステムの性能評価をクライアントごとに行う。

5.2 マルチスレッドの適用

システムの性能を評価するのにその評価するプロセス（図３の場合は監視プロセス）がシステムに負荷をかけてはいけない。このことから，監視プロセスの実装には，自分自身はプロセスとして生成するが，特性要因の抽出には，プロセスよりも軽量であり，何よりもシステムにあまり負荷がかからないスレッドを使用する。

図４は，マルチスレッドを用いた監視プロセスの内部処理の様子を表している。図４の t1，t2，t3 がスレッドである。監視プロセスは C 言語で実装され，スレッドは POSIX スレッドを使っている。また，監視プロセスが複数のクライアントを処理できる台数は，コンパイル時のヘッダ情報に依存するが，その制限台数を変更することも可能である。

5.3 マルチスレッドプログラムの設計

プログラムのスレッド化に関して，代表的な３つのモデルがある。ボス／ワーカモデル，ピアモデル，パイプラインモデルである。これらのモデルは，マルチスレッドのアプリケーションの作業をどのようにしてスレッドに委ねるのか，またスレッドがどのようにして相互に通信を行うのかを定義している[3]。

ボス／ワーカモデルは，ひとつのボススレッドがプログラム全体の入力を受け取り，各ワーカスレッドに個々のタスクを渡す。このとき，ワー

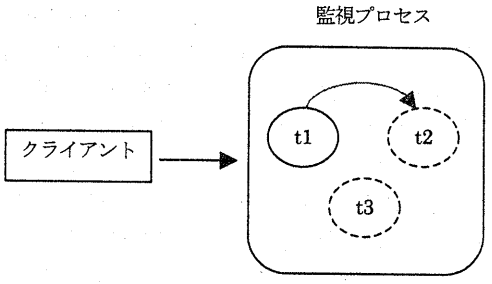

図5　スレッド1

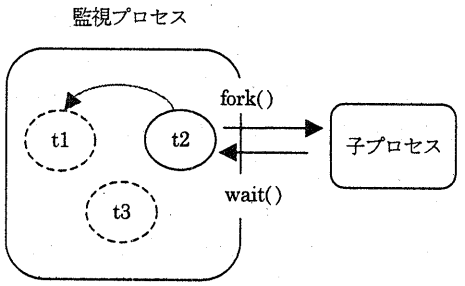

図6　スレッド2

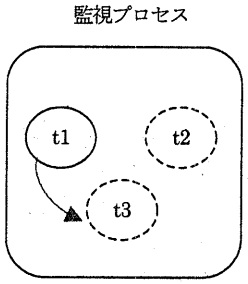

図7　スレッド1からスレッド3へ

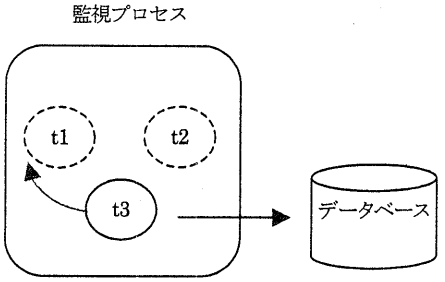

図8　スレッド3

カスレッドはそれぞれ独立して処理を行う。ピアモデルは，ボス／ワーカモデルとは異なり，ボススレッドは存在せず，すべてのスレッドが並行して処理を行う。パイプラインモデルは，各工程におけるスレッドの処理結果を次のスレッドに渡しながら処理を行う（流れ作業）。

図4の監視プロセスは，ボス／ワーカモデルとパイプラインモデルの設計概念を参考にした。

次項では，この監視プロセスの基本概念について触れ，各スレッドがどのような処理を行っているのか述べる。

5.4 基本概念

図4の監視プロセスのボススレッドは t1 である。t2 と t3 は，ワーカスレッドとしての作業を委ねられ，t1 からのパラメータ値をパイプラインとして受け取り，それぞれの役割に応じて処理を行う。

（1）スレッド1

監視プロセスは，クライアントからのサービス要求があると，クライアントからの情報を取得し，スレッド1を生成する（**図5**参照）。スレッド1では，システム的側面からネットワークの特性要因の抽出を行い，ソフトウェア的側面から利用者の特性要因としてクライアントプロセスの実行時間を測定する。

次のC言語のソースコードをみてわるように，このスレッドでは，client[i]によってクライアントごとにスレッドを生成する。そして，あるクライアントのスレッドが正常に終了すると client_count[i].count によって通信回数がカウントされる。buff[0]は，クライアントからの送信バッファ（サーバ側から言えば，受信バッファ）である。バッファサイズは，これにより求められる。スレッド1の処理が完了するとスレッド2に処理が移行する。このとき，スレッド1で測定された通信時間は，パラメータ値としてスレッド2に渡される。

```
* pthread 1 processing */
t1_p= (package_t1 *)malloc(sizeof(package_t1));
t1_p->t1_1 = i;
t1_p->t1_2 = client[i];
t1_p->t1_3 = client_count[i].count;
t1_p->t1_4 = &(buff[0]);

if((pthread_create(&tid_1, NULL, t1, (void *)t1_p)) != 0)
    perror("pthread_create t1");
pthread_join(tid_1, NULL);
client_count[i].count++;
```

（2）スレッド2

このスレッドでは，システム的側面であるハードウェアの特性要因とソフトウェア的側面であるソフトウエアの特性要因の抽出を行う。実際は，評価対象とするプロセスを fork 関数によって子プロセスを生成し，この子プロセスが使用するリソースとその子プロセス自身の実行時間を測定する（**図6**参照)。これらの性能値を測定後，スレッド1から受け取った通信時間のパラメータ値を使って分散システムとしての性能評価をクライアントごとに行う。そして，スレッド2の処理が完了すると，スレッド1へ処理が戻る。

（3）スレッド3

このスレッドでは，スレッド1とスレッド2によって測定された性能測定値をスレッド1からのパラメータ値として受け取り（**図7**参照)，データベースに格納する（**図8**参照)。スレッド3の処理が完了すると，スレッド1へ処理が戻り，スレッド1は，クライアントからの次のサービス要求待つことになる。

6．今後の課題

以上，簡単なクライアント／サーバシステムを想定して，システムの構成要素と特性要因を明確にし，ネットワークの状態を含むシステム的性能と，特定した評価対象のプロセスの実行時間等を監視プロセスの3つのスレッドを用いて測定する手法を述べた。

今後の課題として，監視プロセスそのものの評価と，実際に測定された値から各特性要因の関連性をみつけることである。しかしながら，問題とするところは，データの数が膨大になることであり，実際に測定された値から各特性要因の関連性を見つけることは困難である。この困難を避けるために，それぞれのスレッドから抽出された測定値に重みをつけ，クライアント／サーバシステムの性能評価値を確率・統計的な手法を用いることを検討している[1]。また，ソフトウェアの特性要因で述べたソフトウェア故障数とソフトウェアフォールト数であるが，，実際に稼動しているシステムにおいて，これらを測定することは容易でなく，今後どのようにして測定するかは検討課題の一つである。

参考文献

[1] 亀田壽夫，紀一誠，李頡：性能評価の基礎と応用，共立出版，pp. 6-12（1998）

[2] 山田茂：ソフトウェア信頼性モデル －基礎と応用，日科技連，p. 30（1994）

[3] Bradford Nichols, Dick Buttlar, Jacqueline Proulx Farrell　榊正憲 訳： P threads プログラミング，オライリー・ジャパン，pp. 33-41（1998）