九州大学大学院
システム情報科学研究科報告
第5巻 第1号 平成12年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.5, No.1, March 2000

WWWサーバの動作内容を考慮したスケジュール法における状態判定パラメータRWの自動設定法と評価

スカンヤ スラナワラッ* ・ 谷口 秀夫**

Evaluation of a Scheduling Policy for a WWW Server based on Its Contents – When Scheduling Parameter RW is Automatically Set –

Sukanya SURANAUWARAT and Hideo TANIGUCHI

(Received December 10, 1999)

Abstract: Since traditional process schedulers control the sharing of the processor resources among processes using a fixed scheduling policy based on the utilization of a computer system, not based on *contents* or *behavior of processes*, this can hinder an effective use of a processor or can extend the processing time of a process unnecessarily. We have already proposed a process scheduling policy, which responds to the behavior of multiple processes of a WWW server, in order to improve the response time of a WWW server. This policy gives any process of a WWW server that is predicted to be a WWW server process handling a text data request from a browser priority over all other processes by moving it to the head of the ready queue where processes waiting for the processor to become available are placed. In order to determine which processes are WWW server processes handling text data requests, we introduced the scheduling parameters SLP and RW. In this paper, we explain our method of how we predict and update scheduling parameter RW based on the execution behavior of a WWW server, and we evaluate the effectiveness of our process scheduling policy when the scheduling parameter RW is set according to the proposed method.

Keywords: Process scheduling, WWW server, Response time, Behavior, Contents, Predict

1. はじめに

従来のプロセススケジューラは, 実時間や時分割の処理といったプロセス実行の以前に得られる「利用形態」を基に, プロセスの実行を制御している. 実時間処理の場合は, コンピュータ外部で発生した事象に対して, ある定められた時間制約で処理を行わなければならない. また, 時分割処理の場合は, 各ユーザに対して, できるだけすみやかにコンピュータシステムが応答しなければならない. スケジュール法の研究として, 特に処理の時間制約が厳しい実時間処理についての研究[1)~8)]が多い. これらの方式は, いずれも, 「プロセスの動作内容」に合わせて実行制御を行っているわけではない. このため, プロセッサの有効利用を妨げ, また, プロセスの処理時間を無用に引き延ばしている場合がある. 具体的な例としては, 残り僅かでプロセッサ処理を終えようとするプロセスでも, タイムスライス機能によって, 強制的に再スケジュールが行われる場合が挙げられる. この例では, そのプロセスの処理が次のプロセッサ割当まで待たされることとなり, プロセスの処理時間の増加をもたらす. また, 強制的な再スケジュールは, プロセス切替処理の増加も招く. もし, 強制的な再スケジュールを行わなければ, プロセスの継続実行によりプロセスの処理時間を短縮し, プロセス切替処理も減少させることができる. つまり, 「残り僅かでプロセッサ処理を終える」というプロセスの動作を予測し, プロセスのプロセッサ利用に合わせたスケジュールを行えば, 上述のような無駄は発生しない.

「プロセスの動作内容」に合わせた実行制御は, 処理の性質やデータの関連に注目したマルチプロセッサ環境で行われている[9)~11)]. しかし, プログラムの過去の実行履歴から将来の動作を予測して実行を制御するものではない.

そこで, 我々は, プロセスの動作を記録し, その記録内容に基づき動作を予測して, プロセスの実行を制御するプログラム指向スケジュール[12)] (POS : Program Oriented Schedule) を提案した. このスケジュール法により, 上記の問題は回避でき, 効率的なプロセス動作を可能にできる.

POSの基本的な考え方を以下に示す.

(1) プロセスの動作を把握し, プログラム動作内容の記録として保存する.

(2) プログラム動作内容の記録を利用して, プロセスの実行制御法を変更する.

平成11年12月10日受付

* 情報工学専攻博士後期課程
** 情報工学専攻

POSと同様な考え方として,「応用プログラムの起動を高速化する」というWindows98の機能が挙げられる. これは, ユーザの使用状況に合わせてディスク上のデータ格納位置を最適化し, 応用プログラムの起動を高速化する. 具体的には, ユーザがどの応用プログラムをどの程度の頻度で利用しているか, 各応用プログラムが利用するファイルとその利用頻度をOSが自動的に記録する. それを基にデータ格納位置を最適化することで, 次にその応用プログラムを起動する際は, より高速に起動できる仕組みである[13]. しかし, この機能は, 我々のPOSの考え方と異なり, 記録した内容を利用してプロセスの実行制御を変更するものではない. したがって, この機能を用いることによって, オペレーティングシステムは, 応用プログラムを起動するまでの過程を効率的に制御することができるものの, 実行中の応用プログラムをより効率的に制御することはできない.

これまでに, POSの考え方を用いた研究を文献12）と14）に報告している. 文献12）ではプロセスの動作を記録し予測してプロセスの切替時期を変更する方式を提案している. また, 文献14）ではプロセスの動作を記録し予測して入力処理を効率化する方式を提案している. 前者の方式は, 演算スケジュールへ適用したものであり, サービスが1プロセスで構成されている場合に有効である. しかし, 複数のプロセスで構成されることが多い今日のサービスへの適用には不十分である. このため, 我々は, 現在よく使われているWWWサーバを, 複数のプロセスを構成するサービスの一例として取り上げる. 具体的には, サーバプロセスの動作を記録し, その記録内容を基に状態判定パラメータを予測し, その予測を利用してクライアント・サーバによるトランザクションの応答時間を短くするスケジュール法を提案している[15]. 状態判定パラメータには, SLPとRWの二つがある. SLPの自動設定法および評価については, 文献15）と16）に報告している.

本論文では, 状態判定パラメータRWの自動設定法および評価結果を報告する.

2. スケジュール機構の概要

スケジュール機構は, ログ機構とプロセス制御機構から成る. ログ機構は, WWW サーバプロセスの動作を把握し, WWWサーバプログラムの動作内容の記録として保存する. プロセス制御機構は, WWWサーバプログラムの動作内容の記録を利用して, プロセスの実行制御法を変更する. それぞれの機構について, 以下に説明する.

2.1 ログ機構

WWWサーバが走行しているときに, サーバの動作内容を把握するために, 必要な情報として各サーバプロセスのプロセスの状態とその時刻を記録する. そして, 各サーバプロセスのプロセス識別子とその状態列からなる情報であるPFS[12]（ Program Flow Sequence ）を作成する. 状態列の1要素は, プロセスの状態とその連続時間の情報である. プロセスの状態は, RUN状態かWAIT状態のどちらかである.

2.2 プロセス制御機構

2.2.1 スケジュール法

ブラウザを用いてWWWサーバにアクセスし, WWWページを要求するときのクライアント・サーバによるHTTPトランザクションについて説明する. なお, サーバが提供しているWWWページには, 一つの画像が含まれるとする.

まず, ブラウザはユーザから入力されたURLを読み込んで解釈し, 該当するサーバマシンへ接続してWWWページを要求する. 次に, サーバマシン上でクライアントからの接続を待機している, 複数のサーバプロセスの中の一つがこの要求を受け付け, URLに指定されたパスに対応するHTMLファイルを探して, 要求元のブラウザに返す. 最後に, ブラウザは, サーバから取得したHTMLファイルを解釈しながら必要な整形を行ない, 画面に表示する. ブラウザがこのHTMLファイルを解釈している間に, このWWWページの表示には画像ファイルが必要なことが判明したら, ブラウザはWWW サーバに画像ファイルの要求を送信する. その後, HTMLファイルの要求の場合と同様な処理を行う.

上述のWWWページの要求に関するトランザクションは, HTMLファイルの要求と画像ファイルの要求に関する二つのトランザクションから成る. ユーザにとっては, クリックしてからテキストを表示するまでの時間, すなわち, HTMLファイルの要求に関するトランザクションの応答時間が短いことが望ましいと考えられる. この応答時間を短くするには, HTMLファイルの要求を処理するサーバプロセスの処理時間を短縮することが必要である. これについて, 次に説明する. サーバプロセスは, RUN状態, WAIT状態およびREADY状態を繰り返しながら一つのトランザクションを処理する. この一連の処理を行う上で, 基本的に削減できない時間は, プロセッサ処理時間（プロセス状態はRUN状態）とDISK入出力やデータ通信のための実入出力を行う処理時間（プロセス状態はWAIT状態）である. しかし, 削減できる時間として, オペレーティングシステムのプロセス制御により発生する待ち時間（ハードウェアを占有できれば発生しない時間）がある. この時間は大きく三つある. READYキューでの待ち時間, DISK入出力待ちキューでの待ち時間, およびデータ通信待ちキューでの待ち時間である. したがって, HTMLファイルの要求を処理するサーバプロセスがそれぞれのキューに置かれているとき, キューに繋がれている

時間を短くすることにより, 処理時間を短縮することができる. したがって, 次のスケジュール法を示した[15]. プロセッサ（DISKまたはデータ通信）がボトルネックになった場合には, READYキュー（DISK入出力待ちキューまたはデータ通信待ちキュー）に繋がっているプロセスの中で, HTMLファイルの要求を処理するプロセスを優先してキューを繋ぎ換える.

2.2.2 実現時の課題と対処

プロセッサのボトルネックに着目した場合のスケジュール法（「READYキューでの優先制御法」と呼ぶ）を実現した[15]際, 以下の課題があった.

＜課題1＞ HTMLファイルの要求を処理するプロセスの検出方式

＜課題2＞ プロセスを優先制御するためのREADYキューでの処理方式

これらの課題を対処するために, WWWサーバの動作内容を分析する必要があった. 分析した結果により, HTMLファイルの要求を処理するプロセスは以下の特徴を持つことが分かった.

（特徴1）長いWAIT状態の後に走行する.

（特徴2）RUN状態とWAIT状態を数回繰り返す.

この二つの特徴を利用した課題への対処をし, 以下に示す.

＜課題1への対処＞ プロセスが上述の特徴を持つかどうかを判断するために, 以下の判断基準を導入した.

（1）長いWAIT状態を判断するための状態判定パラメータSLP

（2）RUN状態とWAIT状態が数回であることを判断するための状態判定パラメータRW

RUN状態の直前のWAIT状態の時間がSLPより長い時, そのプロセスは長いWAIT状態の後に走行（特徴1）したと判断する. 次に,（特徴1）を有するプロセスについて, RUN状態とWAIT状態の繰り返し回数がRWより小さい時, そのプロセスはRUN状態とWAIT状態を数回繰り返した（特徴2）と判断する. これにより, HTMLファイルの要求を処理するプロセスを検出する.

＜課題2への対処＞ プロセスを優先制御するためのREADYキューでの処理方式として,（特徴1）を有するプロセスはREADYキューの先頭に繋ぎ,（特徴2）を失った時点でREADYキューの末尾に繋ぎ換える.

なお, SLPとRWは, ログ機構によって作成したPFSを基に予測して自動的に設定する.

3. 状態判定パラメータ RW の設定法

3.1 基本方式

PFSを基に分析したWWWサーバの動作内容により, HTMLファイルや画像ファイルの要求を処理するサーバプロセスのファイルの入力処理によるRUN状態とWAIT状態の繰返し回数は, ファイルの大きさに比例することが分かった. 多くの場合, 画像ファイルがHTMLファイルより大きいので, 画像ファイルの要求を処理するサーバプロセスのRUN状態とWAIT状態の繰返し回数は, HTMLファイルの要求を処理するサーバプロセスのものより多い. このことを利用して, 単位時間の度に, PFSを基に各サーバプロセス毎にRUN状態とWAIT状態の繰返しの最小回数を求め, 次に, 全てのサーバプロセスの最小繰返し回数のうちの最大回数をRWと設定する. 具体的な処理流れを以下に示す.

（1） サーバプロセスが生成されたときに, サーバプロセスのPFSを作成するようにPFSを管理する表に登録する.

（2） 登録した全てのサーバプロセスのPFSの作成の時間間隔を指定する. この時間間隔は, RWの設定における単位時間である.

（3） 単位時間の度に, 各サーバプロセス毎のPFSを作成し, それに基づき, RUN状態とWAIT状態の繰返しの最小回数を求める. 次に, 全てのサーバプロセスの最小繰返し回数のうちの最大回数をRWと設定する.

（4） サーバプロセスが終了するときに, サーバプロセスのPFSの作成を止める.

3.2 設定アルゴリズム

PFSからRUN状態とWAIT状態の繰返しの最小回数を求めるアルゴリズムを以下に説明する. なお, 一連のRUN状態とWAIT状態の繰返し状態であることを判断するために, パラメータshortSLPを導入する. また, 最小RWの閾値minRWと過去のRWの候補を複数格納する領域RWbuffを用意する.

（1） 単位時間で生成されたPFSをWAIT状態がshortSLPより長い点で分割し, 複数のRUN状態とWAIT状態の繰返し部分に分離する.

（2） 分離されたRUN状態とWAIT状態の繰返し部分の各々について, RUN状態とWAIT 状態の繰返し回数を算出する.

（3） 算出した繰返し回数がminRWより大きければ, その値をRWの候補としてRWbuff に格納する.

（4） RWbuffに格納された値の中の最小値を, 当該単位時間におけるRUN状態とWAIT 状態の繰返しの最小回数とする. なお, 単位時間で生成されたPFSの両端において, 状態をshortSLPにより分割できないことが考えられる. このため, 単位時間で生成されたPFSの前後に跨ってRUN状態とWAIT状態の繰返し部分を考え, RUN状態とWAIT状態の繰返し回数を算出する.

上記の設定アルゴリズムは, 以下の点を考慮したものである.

（1） SLP設定アルゴリズムとの分離
RWは, SLPにより（特徴1）を有すると判断されたプロセスについて, （特徴2）を有するか否かを判断する状態判定パラメータである. この条件にしたがってRWを決定すると, RWはSLP設定アルゴリズムの良し悪しの影響を大きく受ける. これを避けるため, RW設定アルゴリズムはSLP設定アルゴリズムとの分離を行った. つまり, RW設定アルゴリズムは, SLP値に関係なく, PFSとshortSLPからRWを決定する. なお, 入出力処理などによる短いWAIT状態の時間は200ミリ秒以下なので, shortSLPを200ミリ秒とした.

（2） 無関係処理の除外
サーバプロセスは, HTMLファイルや画像ファイルの入出力以外に, 通信処理などのために短いRUN状態とWAIT状態の繰返しを行う. この短いRUN状態とWAIT状態の繰返し回数は1〜2回程度であり, HTMLファイルや画像ファイルの入出力処理によるRUN状態とWAIT状態の繰返し回数より少ない. そこで, minRWを2とし, 最小RWの閾値minRWの判定を行うことにより, 無関係な処理を除外した.

（3） 過去情報の利用
最新RWの候補一つではなく, 最近の複数のRWの候補を用いてRW値を決定する. これは, 予測のミスを最小限にすること, 単位時間おいて状態をshortSLPにより分割できない場合へ対処すること, およびサーバ状態の変化に柔軟追従することを図るためである. なお, 過去のRWの候補を複数格納する領域RWbuffは五つとした.

4. 評　　価

状態判定パラメータRWの自動設定のアルゴリズムを評価するために, 提案したアルゴリズムをBSD/OSに実現した. そして, RWの効果を明白にするために, READYキューでの優先制御が必ず行われる状態で実験を行った. 具体的には, プロセッサをボトルネックにするために, 無限ループのプロセッサ処理を行うプロセスをサーバプロセスと共存させ, クライアントからのWWWページ要求の間隔（30秒）をSLPの値（20秒固定）より大きく設定した.

4.1 処理内容

処理の内容は, 30秒毎にブラウザを用いて, サーバ・マシンへ接続しWWWページを要求する. WWWページは, 一つのHTMLファイルと一つの画像から成る.

測定は, RWを固定したとき（RW=1,3,5）と自動的に設定したときについて, HTML ファイルの大きさを変化させて, 以下の時間を測った.

（1） URLの入力からHTMLファイル読み込み開始までの時間（以降,「テキスト・データの応答時間」と略す）

（2） URLの入力から画像ファイル読み込み終了までの時間（以降,「画像データの応答時間」と略す）

ここで, HTMLファイル読み込み開始とは, ブラウザがHTMLファイルの先頭データを受信した時を意味し, 画像ファイル読み込み終了とは, ブラウザが画像ファイルの最後のデータを受信した時を意味する.

4.2 測定環境

実測に用いたサーバプログラムとブラウザプログラムは, それぞれApache Ver.1.2.5とNetscape Navigator Ver.3.04である. 使用した計算機は, Apacheがプロセッサ AMD-K6 233MHzの計算機, Netscape Navigatorがプロセッサ Pentium Pro 200MHzの計算機である. それぞれはBSD/OSを使用し, 10Mbpsのイーサネット型通信路で結ばれている. 測定は, シングルユーザ環境で, オペレーティングシステムが持つ入出力バッファのキャッシュ（131,072バイト）にヒットしないように, 同様のWWWページを持つ異なったURLをたくさん用意し, 各URL に対して上述のテキスト・データと画像データの応答時間を測った. また, Netscape Navigatorが持つバッファのキャッシュも無効にした.

4.3 結果と考察

以下の各図では, テキスト・データおよび画像データの応答時間について, 各RWにおけるREADYキューでの優先制御法を利用したときの時間を利用しないときの時間で割算した値を, HTMLファイルの大きさの変化と共に示す.

Fig. 1は, HTMLファイルの大きさを1倍, 3倍, 5倍,および7倍に変化させたときの応答時間を示す. なお, 1倍のときのHTMLファイルの大きさは1,772バイトである. また, 画像ファイルの大きさは固定43,770バイトである. 通常, オペレーティングシステムがファイルの読み込みの効率化を図るために, ファイルの先読みを行う. したがって, 実験結果を分析するには, 先読みの影響を考慮する必要がある. 先読みは大きく2種類に分けられる. 一つはファイルが連続して格納される場合による先読み（type1）であり, もう一つは, ファイルが連続して格納されない場合による先読み（type2）である. type1の場合は, 要求したファイルの大きさに関係なく, 一回当たりの入力処理は8KB単位で行われる. 一方, type2の場合は, 一回当たりの入力処理は, ファイルの格納の最小の単位であるブロック（4KB）単位で行われる. **Fig. 1(a)**において, オペレーティングシステムが先読みを行う場合のHTMLファイルの入力処理の回数を**Table–1**に示す.

Table 1 The number of disk accesses required to get an HTML file when the operating system performs the sequential read-ahead.

size of HTML file	number of disk accesses	
	type1 case	type2 case
1 time (1,772 bytes)	1 time	1 time
3 times (5,316 bytes)	1 time	2 times
5 times (8,860 bytes)	2 times	3 times
7 times (12,404 bytes)	2 times	4 times

Table–1により, 1倍から5倍までのHTMLファイルの入力処理の回数は, 高々3回である. したがって, RWが3のときとRWが5のときのテキスト・データの応答時間は, **Fig. 1(a)**に示したように, ほぼ同じ値である. また, RWが自動的に設定したときにも同様な値が得られた. そして, HTMLファイルを7倍にしたとき, **Fig. 1(a)**に示したようにRWが5のときのテキスト・データの応答時間は, RWが3のときのものより改善された. これは, おそらく7倍にしたHTMLファイルが連続して格納されていないため, 種類2の先読みが行われ, HTMLファイルの入力処理の回数が**Table–1** に示したように約4回になるからである. また, RWを自動的に設定したときの応答時間もRWが5のときと同様に改善された. したがって, RWをうまく予測し設定できたといえる.

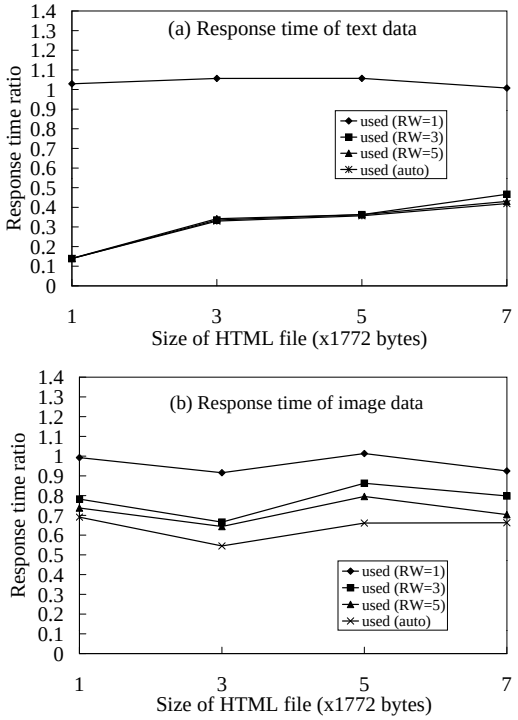

Fig.1 The effect of RW for various sizes of HTML files (1, 3, 5 and 7 times the original size).

Fig. 1(a)において, RWを5と設定したときと自動的に設定したときのテキスト・データの応答時間はほぼ同じである. 一方, 画像データの応答時間は, **Fig. 1(b)**に示したように, RWを自動的に設定したときの方が良い結果が得られた.

HTMLファイルを7倍したときに, RWが3のときの応答時間は, **Fig. 1(a)**に示したように, RWが5のときと自動的に設定したときのものと異なった程度で改善された. そこで, その程度を調べるため, HTMLファイルの大きさを大きくした場合の実験を行った. 実験結果を**Fig. 2**に示す.

Fig. 2は, HTMLファイルの大きさを10倍, 20倍, 30倍, 40倍, および50倍に変化させたときの応答時間を示す. **Fig. 2(a)**と**Fig. 2(b)**において, HTMLファイルを30倍以上にすると, RWが3のときのテキスト・データおよび画像データの応答時間は, 明らかにRWが5のときと自動設定のときのものと異なった程度で改善された. また, RWを自動的に設定したときに応答時間が最も良い結果が得られた.

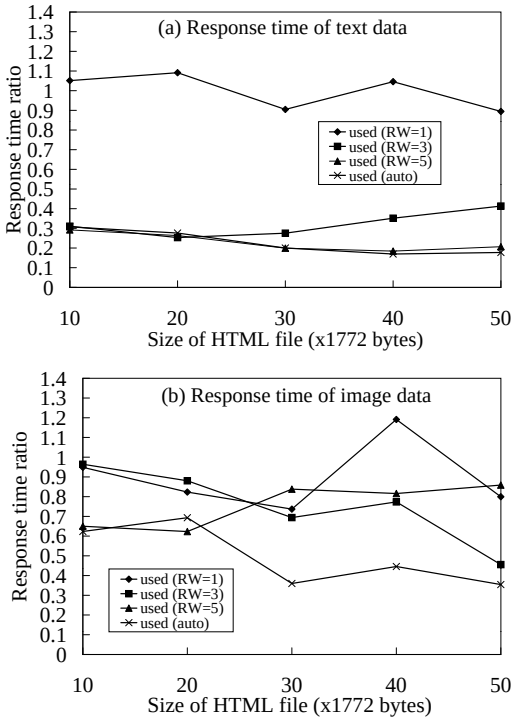

Fig.2 The effect of RW for various sizes of HTML files (10, 20, 30, 40 and 50 times the original size).

実験結果により, RWを自動的に設定したときに, テキスト・データおよび画像データの応答時間は固定のRWに比べ, 同様またはそれ以上に改善された. したがって, RWをうまく予測し設定できたといえる. 一方, 実際のHTMLファイルの大きさは, それほど大きくなく, また, オペレー

ティングシステムによる先読みの効果により, **Fig. 1**に示したように, RWを3〜5に固定しても十分かもしれない.

5. おわりに

WWWサーバの動作内容の記録に基づき状態判定パラメータRWを予測して自動的に設定するときについて, 提案したスケジュール法の有効性を評価した. 評価結果として, RWを自動的に設定したときのWWWサーバのテキスト・データの応答時間は, 固定のRW に比べ, 同様またはそれ以上に改善された. したがって, RWをうまく予測し設定できたといえる.

残された課題として, 画像ファイルの大きさを変化させたときの評価や, 状態判定パラメータSLPとRWの両方を自動的に設定したときについての評価がある.

参考文献

1) K.Ramamritham, J.A.Stankovic and P.Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems," IEEE Trans. Parallel and Dist. Systems, vol.1, no.2, pp.184–194, April 1990.
2) T.Shepard and J.A.M.Gagné, "A pre-run-time scheduling algorithm for hard real-time systems," IEEE Trans. Software Eng., vol.17, no.7, pp.669–677, July 1991.
3) K.Schwan and H.Zhou, "Dynamic scheduling of hard real-time tasks and real-time threads," IEEE Trans. Software Eng., vol.18, no.8, pp.736–748, Aug. 1992.
4) J.Xu and D.L.Parnas, "On satisfying timing constraints in hard-real-time systems," IEEE Trans. Software Eng., vol.19, no.1, pp.70–84, Jan. 1993.
5) W.K.Shih, J.W.S.Liu and C.L.Liu, "Modified rate-monotonic algorithm for scheduling periodic jobs with deferred deadlines," IEEE Trans. Software Eng., vol.19, no.12, pp.1171–1179, Dec. 1993.
6) M.G.Härbour, M.H.Klein and J.P.Lehoczky, "Timing analysis for fixed-priority scheduling of hard real-time systems," IEEE Trans. Software Eng., vol.20, no.1, pp.13–28, Jan. 1994.
7) A.Burns, K.Tindell and A.Wellings, "Effective analysis for engineering real-time fixed priority schedulers," IEEE Trans. Software Eng., vol.21, no.5, pp.475–479, May 1995.
8) W.Feng and J.W.S.Liu, "Algorithms for scheduling real-time tasks with input error and end-to-end deadlines," IEEE Trans. Software Eng., vol.23, no.2, pp.93–106, Feb. 1997.
9) Y.Kwok and I.Ahmad, "Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors," IEEE Trans. Parallel and Dist. Systems, vol.7, no.5, pp.506–521, May 1996.
10) H.Wang, A.Nicolau and K.S.Siu, "The strict time lower bound and optimal schedules for parallel prefix with resource constraints," IEEE Trans. Comput., vol.45, no.11, pp.1257–1271, Nov. 1996.
11) M.Wu, and W.Shu, "On parallelization of static scheduling algorithms," IEEE Trans. Software Eng., vol.23, no.8, pp.517–528, Aug. 1997.
12) 谷口秀夫, "POS:プログラム指向スケジュールの提案," 情報処理学会シンポジウム論文集, vol.96, no.7, pp.123–130, 1996.
13) http://biztech.nikkeibp.co.jp/image/cgi/bizfrm.cgi/general/bp980203307.html
14) 谷口秀夫, "入出力要求流れの学習による入出力処理の効率化," 情報処理学会シンポジウム論文集, vol.97, no.8, pp.141–148, 1997.
15) スカンヤ・スラナワラッ, 谷口秀夫, "WWW サーバにおけるサービスの処理内容を考慮したプロセススケジュール法," 情報処理学会論文誌, vol.40, no.6, pp.2510–2522, June 1999.
16) S.Suranauwarat and H.Taniguchi, "Evaluation of process scheduling policy for a WWW server based on its contents," IPS Japan Proc. of Computer System Symposium'99, vol.99, no.16, pp.105–112, Nov. 1999.