WWW サーバにおけるサービスの処理内容を考慮したディスクスケジュール法の実現と評価

スカンヤ・スラナワラッ† 谷口秀夫†

WWW（World Wide Web）の急速な普及にともない，利用者からのアクセス数が増加し，サーバの負荷が増大してサービス品質が悪化する問題が生じている．この対処策として，オペレーティングシステムが，WWW サーバが使用する資源を効率的に制御し管理することがあげられる．本論文では，資源として磁気ディスク装置に着目し，リアルタイム処理やタイムシェアリング処理といった計算機の利用形態を基にした従来のディスクスケジュール法と異なり，WWW サーバを構成する複数のプロセスの処理内容を記録し，その記録内容に基づき予測した HTML（Hyper-Text Markup Language）ファイルの読み出し要求を処理するプロセスの入出力要求を優先して処理するディスクスケジュール法を実現する．評価の観点として，WWW ブラウザの画面に最初に表示されるデータである HTML ファイルの表示が速くなることを確認する．

The Implementation and Evaluation of a Disk Scheduling Policy for a WWW Server Based on Its Contents

Sukanya Suranauwarat† and Hideo Taniguchi†

The explosive growth of the WWW (World Wide Web) is placing a heavy demand on WWW servers. As a result, the quality of service become worse. This has become a critical issue. A way to improve the quality of service is to improve operating systems support for servers, especially in the area of resources allocation. In this paper, we implement a disk scheduling policy that controls the allocation of a disk resource based on the behavior of server processes rather than based on a fixed policy using in traditional operating systems, in which a utilization of a computer system such as a real-time or a time-sharing system is a major concern. To be more precise, this policy gives preferential use of the disk resource to any process that is predicted to be a server process handling an HTML (Hyper-Text Markup Language) file request, by operating the disk I/O queue. We also evaluate this policy experimentally in order to verify that by using this policy user can view text and the general layout of a WWW page in a timely manner during periods of high demand.

1. はじめに

近年，インターネットの急速な普及により，WWW（World Wide Web）の利用者が爆発的に増加している．これは，世界中に分散した膨大な情報の中からサービス利用者が真に所望する情報を簡単かつリアルタイムに入手可能なことや，テキスト以外の静止画像やアニメーション画像などの情報が簡単に扱えることなど，理由に事欠かない．しかし，その爆発的な利用にともない，利用者からのアクセス数が増加し，サーバの負荷が増大してサービス品質が悪化する問題が生じている．ここで，サービス品質とは，WWW ページを要求してから表示が開始されるまでの時間もしくは表示が終了するまでの時間である[1)]．特に，サービス利用者は「より速くページが表示される」ということを要求するようになった．本論文では，特に断らない限り，サービス品質といえば，前者を意味することとする．

サービス品質の悪化という問題への対処策として，オペレーティングシステム（以降，OS と略す）が，WWW サーバが使用するプロセッサやディスク装置や通信路といった資源を効率的に制御し管理することがあげられる．これまで，OS が使用している資源の割当ての手法は，リアルタイム処理（real-time processing）やタイムシェアリング処理（time-sharing system）などの計算機の利用形態を背景に，数多く提案されている[2)~7)]．特に，サービス品質の保証については，リア

† 九州大学大学院システム情報科学研究科
Graduate School of Information Science and Electrical Engineering, Kyushu University

ルタイム処理を中心に研究が行われている．具体的には，利用者がサービスの処理を要求してから処理が終了するまでの時間について時間的な制約を保証する．しかし，利用者がサービスの処理を要求してから処理が開始されるまでの時間に関しては考慮されていない．このため，処理の開始を重点におく場合，つまり，WWW のサービス品質の向上には不十分である．

また，現在，WWW サーバの環境としてタイムシェアリング処理システムの場合は，WWW サーバが同時に複数の利用者からアクセスされると，次の問題がある．プロセスが同じ優先度を持つ場合，処理を均等化するため，プロセスが異なった処理内容を行うにもかかわらず，各プロセスには一定の時間プロセッサが割り当てられる．このため，WWW ブラウザの画面に最初に表示するデータを保存した HTML（Hyper-Text Markup Language）ファイルの読み出し要求を処理するサーバプロセスは，画像ファイルなどの読み出し要求を処理するサーバプロセスと同様にプロセッサが割り当てられる．そこで，WWW サーバが同時に複数の利用者からアクセスされると，それぞれの要求を処理するためのサーバプロセスの数が多くなり，HTML ファイルの要求を処理するサーバプロセスは長い時間にプロセッサの割当てを待たされる可能性がある．その結果として，WWW ページの表示が開始されるまでは長い時間がかかってしまい，結局利用者がページの表示を待ちきれず途中で転送を中止してしまうことになりかねない．また，ディスクへの入出力の場合，各プロセスからの入出力の要求は，読み書きのヘッドの動きが最適になるような順に入出力キューに入れられる．一方，通信路の場合は，単に要求の順番に送信キューに入れられる．これらにより，HTML ファイルの要求を処理するサーバプロセスからの要求がキューの後ろの方に入れられると，ディスクや通信路を利用できるまでの時間が長くなり，プロセッサの割当ての場合と同様な問題が起こりうる．さらに，プロセスが異なった実行優先度を持つ場合，WWW サーバプロセスのようにプロセスが高い実行優先度を持っているにもかかわらず，低い実行優先度を持つプロセスによるディスクや通信路の利用の終了を待たなければならない状況，つまり，優先度の逆転（priority inversion）が生じる可能性がある．

一方，我々は，サービスを構成するプロセスの処理内容を考慮してサービスの処理を効率化する資源の割当ての手法を提案[16]している．提案手法の特徴は，プロセスの処理内容を記録して，その記録内容をそのサービスの特徴に合わせて利用できることである．本手法は，プロセスの処理内容を予測することが可能であるサービス，つまり，トランザクション処理のように同じ処理を繰り返すようなサービスを対象している．対象サービスの例としての WWW サーバへ適用した場合の提案手法[8]（以降，提案スケジュール法と呼ぶ）は，サーバプロセスの処理内容を記録し，記録内容に基づき HTML ファイルの読み出し要求を処理するプロセスを予測し，そのプロセスの処理を優先させるために資源の割当てを待つキューをつなぎ換える．この手法により，WWW ページ表示の開始が速くなり，ページがブラウザに表示されるまで待ちきれない問題が解消できる．プロセッサに着目した場合について，提案スケジュール法を実現して評価し，ページ表示の開始が速くなるという期待している結果が得られた[8)~10)]．また，HTML ファイルの読み出し要求を処理するサーバプロセスを優先させても，画像ファイルの読み出し要求を処理するサーバプロセスに与える影響は少ないことを明らかにした[9),10)]．そこで，本論文では，磁気ディスク装置に着目し，提案スケジュール法を実現し評価結果を報告する．

なお，WWW サーバへの要求は，CGI などの技術によって要求ごとに内容が変わる動的なファイル（dynamic files）[17)]よりも，すべての要求に対して同一の内容である静的なファイル（static files）[17)]の方が多い．特に，HTML ファイルと画像ファイルは，90%以上の要求を占めている[18)~21)]．一方，CGI 技術の利用も年々増えているが，CGI 技術による WWW サーバへの負荷の増大やセキュリティの問題で，プロバイダによっては，CGI の使用を認めていなかったり，制限したりしているところがある．したがって，本論文では，静的なファイルを扱う WWW サーバを対象にした．

2. 関 連 研 究

提案されているディスクスケジュール法の中で，最も単純でインプリメントしやすいとよく知られている方法としては，入出力要求の順番に入出力を行う FCFS（First Come First Served）法である．しかし，この方法は，プロセスが要求する入出力の順序は，ディスクの回転位置や読み書きヘッドの位置をまったく考慮していないので，ディスクの回転待ちや，読み書きヘッドに無駄が生じる場合が多く，性能が悪いと指摘されている．これにより，回転位置や読み書きヘッドの位置を考慮する様々な手法が提案されている．代表的な例として，以下のようなスケジュール法があげられる．

SSTF（Shortest Seek Time First）法[11),12)]は，要

求されている入出力の中で，最も読み書きヘッドの動作が少なくてすむ入出力の方法である．この方法は，読み書きヘッドの動作が最小限であるため，高速な入出力が可能である．一方，入出力要求が長い時間にわたって待たされる可能性がある．たとえば，現在の読み書きヘッドの位置から遠くに離れた位置の入出力要求を行う場合，それより近い位置の処理がすべて終了しないと，この処理は実行されない．次から次に，近い位置の処理が要求されると，この処理は長い時間待たされてしまうことになる．

SCAN 法[11),12)]は，ディスクをスキャンするように入出力を行う方法である．具体的には，読み書きヘッドの位置に最も近い入出力要求の位置（軸方向または外側方向）をヘッドの進み方向とする．そして，その方向へ向かって動くときには動きに近い場所への入出力を行う．読み書きヘッドが軸または最も外側に着くと，反対の方向をヘッドの進み方向とし，同様に入出力を行う．この方法は，SSTF 法のように，特定の入出力要求が長い時間待たされる危険性はない．さらに，SCAN 法を基に次の手法が提案されている．C-SCAN（Circular SCAN）法[11),12)]は，読み書きヘッドが最も外側（または軸）に着くと，ヘッドが軸（または最も外側）へ移動する以外，SCAN 法と同様に入出力を行う．この方法は，中部のシリンダに位置する入出力要求を好む傾向がある SCAN 法を改善し，各シリンダに対してより平等に入出力を行う．LOOK 法[12),13)]は，SCAN 法と異なり，読み書きヘッドが軸または最も外側まで進むのではなく，進んでいる方向に入出力要求がなくなると，ヘッドの進み方向が変わる．さらに，C-SCAN 法と LOOK 法との組合せにより C-LOOK 法が提案されている．

VSCAN（R）法[14)]は，基本的に SSTF 法と同じであるが，読み書きヘッドの移動方向が変わる度に，パラメータ R や読み書きの位置に基づくペナルティが加えられる．

SLTF（Shortest Latency Time First）法[11)]は，回転待ち時間が少ない順に入出力を行う方法である．この方法も FCFS 法と同様な問題がある．

上述の方式は，いずれも，ディスク入出力のスループットを向上しようとするものである．このため，入出力を要求する処理の内容を考慮したものではなく，サービス品質を向上させることはできない．

3. 提案スケジュール法の概要

本章では，本論文と特に関連が深い提案スケジュール法[8)]の内容について簡単に述べる．

WWW において，サービス利用者は，ブラウザなどの WWW クライアントを用いて，WWW サーバの提供している WWW ページを入手する．具体的には，ブラウザは WWW サーバのディスク上に保存されている HTML☆で記述されたテキスト・ファイルの読み出しを要求し，WWW サーバはその要求に対して該当ファイルを読み出してブラウザへ転送する．そして，ブラウザは，取得した HTML ファイルの内容を解釈して画面に表示する．取得した HTML ファイルが画像などのデータとの関連づけがある場合☆☆，つまり，画像などのデータを保存する WWW サーバ上のファイルの場所の記述がある場合，ブラウザはその HTML ファイルの解釈・表示中に画像ファイルなども WWW サーバから読み出して，テキスト中に画像などが埋め込まれた複合文書をブラウザの画面に表示する．

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などの要求に関するトランザクションからなる．ユーザにとっては，WWW ページを要求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．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によって，実現できると考えられる．次に，この処理時間の短縮について簡単に述べる．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 1 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うえで，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）とディスク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，OS 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 3 つある．READY キューでの待ち時間，ディスク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（またはサーバプロセスの入出力要求）がそれぞれのキューに置かれているとき，キューにつながれている時間を短くする

☆ HTML はテキストの構造やレイアウト，表示方法（点滅など）を記述する．実際には「< 何とか >」という形で始まり，「</何とか >」という形で終わる「タグ」で表現する．

☆☆ HTML ファイル中には画像や音声などのテキストでないデータを置くことができない．

ことにより，処理時間を短縮することができる．これにより，プロセッサ（またはディスクや通信路）がボトルネックになった場合には，READY キュー（またはディスク入出力待ちキュー☆やデータ通信待ちキュー）につな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つなぎ換えるスケジュール法を提案した．この提案スケジュール法により，テキスト表示の開始および終了を速くすることができる．

4. 実　　現

本章では，磁気ディスク装置に着目した場合の提案スケジュール法（以降,「DK スケジュール法」と呼ぶ）について，実現時の課題と対処および実現方式を説明する．

4.1 実現時の課題と対処

DK スケジュール法を実現するには，以下の課題がある．

＜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ディスク入出力待ちキューでの処理方式

課題 1 への対処：この課題への対処は，すでに実現したプロセッサに着目した提案スケジュール法[8]の場合と同じなので，次に簡単に述べる．HTML ファイルの要求を処理するプロセスは,「長い WAIT 状態の後に走行する（特徴 1）」と「RUN 状態と WAIT 状態を数回繰り返す（特徴 2）」の 2 つの特徴を持つ．この 2 つの特徴を利用し，他のプロセスと区別し検出することができる．また，長い WAIT 状態を判断するために，しきい値 SLP を，RUN 状態と WAIT 状態が数回であることを判断するために，しきい値 RW を導入した．ここで，RUN 状態の直前の WAIT 状態の時間が SLP より長いとき，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．また,（特徴 1）を有するプロセスについて，RUN 状態と WAIT 状態の繰返し回数が RW より小さいとき，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．これらにより，HTML ファイルの要求を処理するプロセスを検出する．なお，SLP と RW は，サーバプロセスの処理内容の記録である PFS（Program Flow Sequence）を基に予測して設定する．PFS は，サーバプロセスの識別子とその状態の列からなる．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．PFS の情報作成は，一定周期（PFS 作成単位時間）で行い，このときに作成した情報を最新情報として PFS に追加する．ただし，PFS として保有する情報の時間長は，格納領域の大きさで制限される．SLP と RW の予測も一定の周期（SLP 予測単位時間もしくは RW 予測単位時間）で行い，この予測に利用する PFS の情報は予測時に PFS として保有されている情報の最新の単位時間（PFS 利用時間）分である．

SLP の予測法について，次に簡単に述べる．HTML ファイルの要求を処理するプロセスは，利用者からの接続を待機しているプロセスである．接続を待機する時間が比較的長い．したがって，SLP 予測単位時間において PFS を基に各サーバプロセスごとに最も長い WAIT 状態の時間を求め，次に，すべてのサーバプロセスの最長 WAIT 状態時間のうちの最小時間を SLP と設定する．また，RW の予測法について，次に簡単に述べる．HTML ファイルや画像ファイルの要求を処理するプロセスのファイルの読み出し処理による 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．通常，画像ファイルが HTML ファイルより大きい．このことを利用して，RW 予測単位時間において PFS を基に各サーバプロセスごとに最小の RUN 状態と WAIT 状態の繰返し回数を求め，次に，すべ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．

課題 2 への対処：プロセスを優先制御するためのディスク入出力待ちキューでの処理方式として，HTML ファイルの要求を処理するプロセスの入出力要求をディスク入出力待ちキューの先頭に入れること☆☆にする．具体的には,（特徴 1）を有するプロセスはディスク入出力の際にその入出力要求をキューの先頭に入れ,（特徴 2）を失った時点で以降そのプロセスからの入出力要求の順序づけが使用した OS に任される．なお,（特徴 1）を有するプロセスが複数ある場合，ディスク入出力の際にその入出力要求をキューの先頭に FCFS（First Come First Served）順に入れる．

4.2 実 現 方 式

以降，DK スケジュール法について述べる．

サーバプロセスを制御するため，**表 1** に示す管理表を用意した．表 1 における 1 要素は，プロセスの識別子（pid），要素が使用中か否かを示す要素の状態

☆ 文献 8) では，DISK 入出力待ちキューという呼び方をしているが，本論文では，ディスク入出力待ちキューと呼ぶことにした．

☆☆ なお，本スケジュール法を実現した BSD/OS 2.1 では，先頭の要求は現在の入出力の処理が行われているため，その直後に入れることにした．

表 1 対象プロセスの管理表
Table 1 A process management table for our disk scheduling policy.

pid	status	flag	rw_cnt	wait_cnt
⋮				
⋮				

（status），DK スケジュール法を使って入出力要求をキューに入れるかまたは OS の使用している従来スケジュール法を使って入出力要求をキューに入れるかを示すフラグ（flag），RUN 状態と WAIT 状態の繰返し回数を記録するカウンタ（rw_cnt），および WAIT 状態の連続時間を記録するカウンタ（wait_cnt）の 5 項目からなる．

サーバプロセスを生成するプロセスがサーバプロセスを生成したときに，DK スケジュール法の対象プロセスとするために，システムコールを用いてサーバプロセスを表 1 に示す管理表に登録する．このシステムコールは，指定されたプロセスに表 1 の要素を新たに割り当て，各フィールドを次のように初期化する．pid には指定されたプロセス（サーバプロセス）の識別子，status には使用中，flag には従来スケジュール法の使用，rw_cnt および wait_cnt にはそれぞれゼロと設定する．そして，プロセスが終了すると，当該要素を解放する．そして，プロセスが生成されてから終了するまでの間は，以下の手順を繰り返す．

（1） 対象プロセスが WAIT 状態になると，wait_cnt を 1 秒ごとにカウントアップする．

（2） 対象プロセスが WAIT 状態から READY 状態へ移行する際，

（A）その対象プロセスが WAIT 状態にいる時間が長い場合（wait_cnt >= SLP），そのプロセスは HTML ファイルの要求を処理するプロセスと判断し，以降そのプロセスからの入出力要求を DK スケジュール法を用いてキューに入れるように flag を ON にする．そして，rw_cnt を 1 とカウントアップし，wait_cnt をクリアする．

（B）その対象プロセスが WAIT 状態にいる時間が短い場合（wait_cnt < SLP），

- そのプロセスが前回 HTML ファイルの要求を処理するプロセスと判断され flag が ON であれば，wait_cnt をクリアする．そして RUN 状態と WAIT 状態の繰返し回数が小さい場合（rw_cnt < RW），flag を ON のままにし，rw_cnt をカウントアップする．そうでなければ，つまり，RUN 状態と WAIT 状態の繰返し回数が大きい場合（rw_cnt >= RW），そのプロセスは HTML ファイルの要求を処理するプロセスでないと判断し，そのプロセスからの入出力要求を優先する必要がないので，flag を OFF にし，rw_cnt をクリアする．
- そのプロセスが前回 HTML ファイルの要求を処理するプロセスでないと判断され flag が OFF であれば，そのプロセスからの入出力要求を優先する必要がないので，flag を OFF のままにし，wait_cnt をクリアする．

なお，本論文では，サーバの OS として，多くのサーバに採用されている UNIX（BSD/OS）を使用することにした．UNIX は，2 章で述べた C-LOOK 法を用いて要求をシリンダ番号の昇順に並べてキューに入れるための disksort ルーチンを用意している．disksort は，キュー内の最初の要求のシリンダ番号が，ヘッドの現在の位置を表していると仮定する．大きなシリンダ番号を持った新しい要求は，この要求の後ろに昇順にキューに入れられる．現在のシリンダ番号より小さな値を持つ要求は，最大のシリンダ番号を持つ要求の後ろに昇順にキューに入れられる．disksort を用いて入出力要求をキューに入れる例を**図 1** (a) に示す．この例は，キュー内の最初の要求のシリンダ番号を 100 とし，新たにシリンダ番号の 75，30，120 を順に要求される場合のキューの様子を示す．さらに，新たに HTML ファイルのシリンダ番号の 140 および 80 を順にファイルの読み出しが要求されたとき，disksort を用いた場合と DK スケジュール法を用いた場合についてキューの様子をぞれぞれ図 1 (b) と図 1 (c) に示す．この例からも分かるように，キューの長さが長くなると，DK スケジュール法を用いることによって，HTML ファイルの読み出し要求がより速く処理され，テキストの表示，つまり，WWW ページの表示が開始されるまでの時間が短くなる．

5. 評　　価

DK スケジュール法を評価するために，4.2 節で述べたアルゴリズムを BSD/OS 2.1 に実現した．なお，DK スケジュール法の効果を明白にするため，DK スケジュール法のみを適用し，プロセッサに着目したスケジュール法[8]は共存させていない．測定は，システム環境からの影響を抑制するために，シングルユーザ

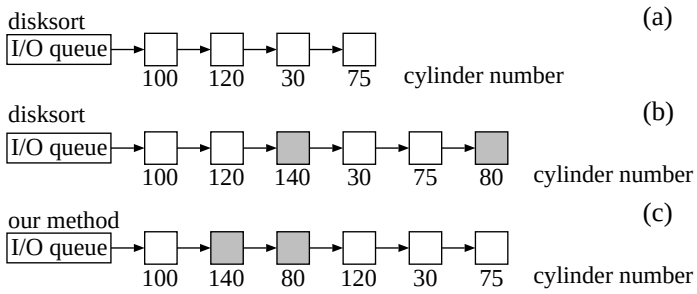


図 1 disksort を用いた場合と提案スケジュール法を用いた場合におけるディスク入出力キューの様子の例
Fig. 1 Examples showing the content of disk I/O queue when using disksort and when using our method.

環境で，OS が持つ入出力バッファのキャッシュがヒットしない状態で行った．評価実験は，2 つの観点で行った．1 つは，DK スケジュール法を用いることによって，ファイルの読み出し時間を短縮できることを確認する（実験 1）．もう 1 つは，同時に複数の利用者からアクセスされる場合の WWW サーバの性能を DK スケジュール法を用いた場合について評価する（実験 2 と実験 3）．それぞれについて以下に説明する．

5.1 実 験 1

評価プログラムの処理は，大きさ 8,000 バイトのテキスト・ファイルを読み出す．実験の内容は，入出力要求をキューに入れるときに disksort を用いた場合と DK スケジュール法を用いた場合について，テキスト・ファイルの読み出し時間を 20 回測定した．また，実験に使用した計算機は，AMD-K6 233 MHz である．実験の結果を**図 2** に示す．図 2 は，異なった 20 種類のファイルの読み出しを無限に繰り返す処理を行うプロセス（以降，共存プロセスと呼ぶ）の有無について，読み出し時間の 16 回☆の平均値を表す．なお，共存プロセスがある場合，その数を 1 から 3 まで変化させた．図 2 には次のことを示している．

（1） disksort を用いたとき，共存プロセスがない場合と共存プロセスが 1 つある場合の読み出し時間が同じあるので，共存プロセスが 1 つの場合には，ディスクのボトルネックが起こらないことが分かった．つまり，プロセスがディスクを利用できるまでキューで待たされることはない．このため，入出力待ちキューを操作する DK スケジュール法の効果が見られない．

（2） 2 つ以上の共存プロセス数があるときに disksort を用いた場合の読み出し時間は，1 プロセスが所要する読み出し時間を共存プロセス数の分に近い割合で増加する傾向が見られる．一方，DK スケジュールを用いた場合の読み出し時間は，あまり増加していない．

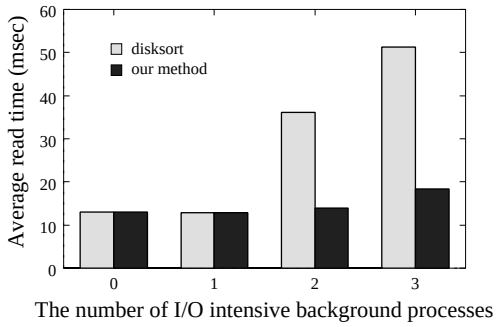


図 2 評価プログラムによる結果
Fig. 2 The effect of our scheduling policy when using a test program.

上記のことから，DK スケジュール法を用いて入出力の要求をキューに入れることにより，ファイルの読み出し時間が短くなることが確認できた．

5.2 実 験 2

実験に使用したハードウェアとソフトウェアを**表 2** に示す．実験システムの構成は，表 2 に示したスペックのクライアント・マシン 3 台とサーバ・マシン 1 台を 10 Mbps のイーサネット型通信路で結んだ．WWW サーバには Apache 1.2.5 [15] を用いた．Apache の処理概要を通常運用モードであるスタンドアロン・モードの場合について以下に簡単に説明する．Apache を稼働させると，マスタと呼ぶプロセスが生成される．このプロセスは，サービス利用者からの接続要求をいっさい受け付けないが，コンフィギュレーション・ファイルに指定された，起動時のサーバプロセスの数（StartServers と呼ぶ）で新たなプロセスを生成し，これらのプロセスの状態を監視する．具体的には，生成したプロセス

☆ 測定初期や終期では，値が大きく変動したため，測定回数 20 回の中で最大・最小各々2 回を除く 16 回の測定値の平均した．

表 2 実験の環境
Table 2 Experimental environment.

	Server	Client
CPU	AMD-K6 233 MHz	Pentium Pro 200 MHz
Memory	64 MB	64 MB
OS	BSD/OS 2.1	BSD/OS 2.1
Program	Apache 1.2.5	Netscape Navigator 3.04

の中から接続待ち状態のプロセスの数がコンフィギュレーション・ファイルに指定された，接続待ち状態のサーバプロセスの最小数（MinSpareServers と呼ぶ）よりも少なくなった場合は，マスタ・プロセスが新しいプロセスを 1 秒に 1 つ新たに生成する．一方，生成されたプロセスは，コンフィギュレーション・ファイルに指定された，接続待ち状態のサーバプロセスの最大数（MaxSpareServers と呼ぶ）や，各サーバプロセスが処理できる要求の最大数（MaxRequestsPerChild と呼ぶ）に達すると終了する．なお，本論文では，文献 8) と同様にコンフィギュレーション・ファイルを設定した．簡単にいえば，StartServers を 1，MinSpareServers を 1，MaxSpareServers を 3，MaxRequestsPerChild を 3 と設定した．

実験の内容は，各マシンから 3 つの Netscape プロセス（合計 9 プロセス）☆が，固定間隔（30 秒）で同時に WWW ページを要求する．このとき，DK スケジュール法の効果を明白にするため，SLP の値（20 秒）を WWW ページ要求の間隔（30 秒）より小さく設定した．さらに，より一般の場合として，WWW ページ要求の間隔がランダムのとき（10〜30 秒の範囲）についても行った．このとき，SLP については，PFS を基に予測して SLP 予測単位時間ごとに自動的に更新する．なお，PFS 作成単位時間や PFS 利用時間や SLP 予測単位時間は，それぞれ 500 ミリ秒とし，PFS の情報作成時に SLP の予測も行うことにした．また，各 Netscape プロセスが要求する WWW ページ☆☆は，異なった URL（Uniform Resource Locator）を持ち，1 つの HTML ファイル（1,772 バイト）と 1 つの画像（43,770 バイト）からなる．そして，各 Netscape プロセスが持つバッファ・キャッシュの効果の影響を受けないように，バッファ・キャッシュを無効にした状態で測定を行った．測定は，RW を 1 から 10 まで変化させて，以下の時間を異なった 18 種類の URL について 5 回測った．

☆ 9 つの Netscape プロセスは，文献 9) に示したように実験用のサーバ・マシンの CPU をボトルネックさせることができる負荷である．

☆☆ 著者が所属している九州大学のホームページ相当とした．

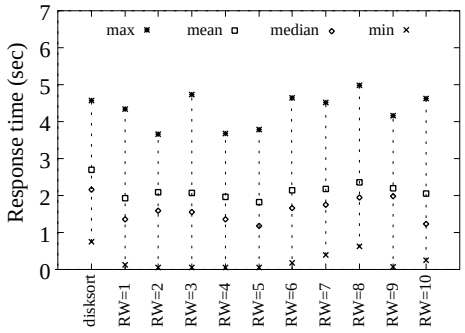
（1） URL の入力から HTML ファイル読み込み開始までの時間（T1）

（2） URL の入力から画像ファイル読み込み終了までの時間（T2）

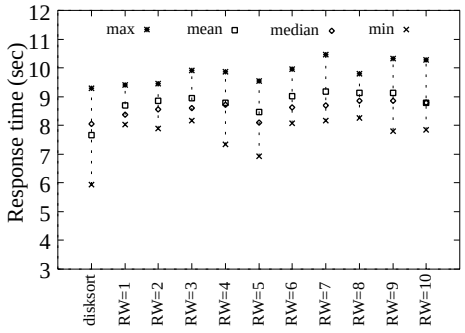
ここで，HTML ファイル読み込み開始とは，Netscape プロセスが HTML ファイルの先頭データを受信したときを意味し，画像ファイル読み込み終了とは，Netscape プロセスが画像ファイルの最後のデータを受信したときを意味する．なお，T1 と T2 の 5 回の平均値をそれぞれ「HTML ファイル読み込みの開始時間（t1）」と「画像ファイル読み込みの終了時間（t2）」と名付ける．

測定の結果を**図 3** と**図 4** に示す．このとき，ディスクの入出力の単位（ブロック）は 8,192 バイトであり，1,772 バイトの HTML ファイルは 1 ブロック，43,770 バイトの画像ファイルは 6 ブロックに相当する．ただし，OS の先読み機能により，同一のファイルのブロックがディスク上に連続して格納された場合には，それらをいくつかにまとめて入出力を行うことがある．また，ディスク入出力待ちキューの長さが長い場合には，1 ブロックの入出力において WAIT 状態と RUN 状態を複数回繰り返す．なお，比較するために disksort を用いたときの結果も表す．図 3 (a) と図 4 (a) において，各 RW について HTML ファイル読み込みの開始時間（t1）の最大値，最小値，平均値，中央値および分布範囲（点線で表す）を示す．ただし，中央値は，それぞれ T1 と T2 から求めた．また，結果を議論するために，最大値，最小値，平均値，中央値のそれぞれについて最善の場合，最悪の場合，および全体の改善された HTML ファイル読み込みの開始時間（t1）の割合をそれぞれ**表 3** と**表 4** に示す．なお，括弧の中の数字は，HTML ファイル読み込みの開始時間（t1）を disksort の場合と比べて，どの程度改善（プラスで表す）または悪化（マイナスで表す）されたかを示す．図 3 (a) と図 4 (a) から分かるように，最大値以外，DK スケジュール法を用いて入出力要求をキューに入れる場合，各 RW について HTML ファイル読み込みの開始時間（t1）が速くなっている．特に，最小値は，最も改善され，80%以上も短縮できた．また，代表値である平均値と中央値は，WWW ページ要求の間隔が 30 秒固定のときにそれぞれ最大 32%と 45%（最悪でもそれぞれ 12%と 8%），また，WWW ページ要求の間隔がランダムのときにそれぞれ最大 30%と 54%（最悪でもそれぞれ 11%と 14%）に改善された．

HTML ファイル読み込みの開始時間（t1）の最大値について，DK スケジュール法を使用したときに記録



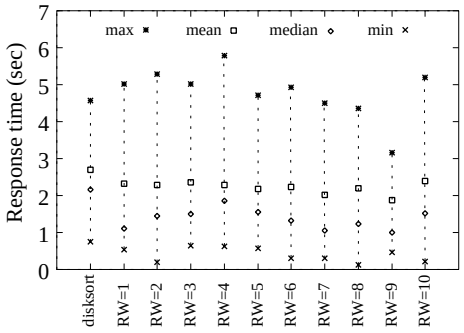
(a) Response time of text data



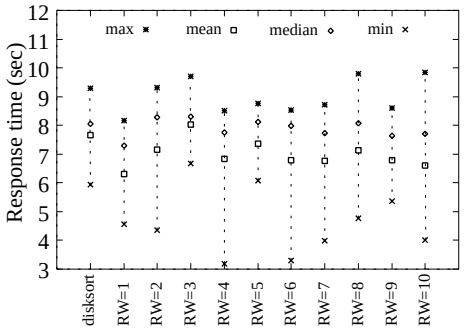
(b) Response time of image data

図 3 WWW ページ要求の間隔が固定かつ SLP を固定したときの実験結果

Fig. 3 The experimental results when browsers access the WWW server periodically.



(a) Response time of text data



(b) Response time of image data

図 4 WWW ページ要求の間隔がランダムかつ SLP を自動的に更新したときの実験結果

Fig. 4 The experimental results when browsers access the WWW server randomly.

表 3 WWW ページ要求の間隔が固定かつ SLP を固定したときの実験結果の分析

Table 3 Analysis of experimental results when browsers access the WWW server periodically at the same time and SLP is fixed.

	disksort	our method		
	(sec)	best (sec)	worst (sec)	better
max	4.57	3.66 (+20%)	4.98 (− 9%)	60%
mean	2.69	1.83 (+32%)	2.36 (+12%)	100%
median	2.17	1.19 (+45%)	1.99 (+ 8%)	100%
min	0.76	0.05 (+94%)	0.63 (+17%)	100%

表 4 WWW ページ要求の間隔がランダムかつ SLP を自動的に更新したときの実験結果の分析

Table 4 Analysis of experimental results when browsers access the WWW server randomly at the same time and SLP is set automatically.

	disksort	our method		
	(sec)	best (sec)	worst (sec)	better
max	4.57	3.15 (+31%)	5.79 (−27%)	30%
mean	2.69	1.87 (+30%)	2.39 (+11%)	100%
median	2.17	0.99 (+54%)	1.87 (+14%)	100%
min	0.76	0.13 (+82%)	0.64 (+15%)	100%

した入出力キューの内容のログを用いて分析する．ログの各要素は，入出力を要求したプロセスの識別子，要求のシリンダ番号および物理ブロック番号の情報である．また，ログ作成のオーバヘッドが小さいことも確認した．測定時に作成したログを分析した結果として，ディスクがあまりボトルネックにならなかったときは，最大値が disksort の場合と同様またはそれ以上大きい．これは，5.1 節で述べたように，ディスクがボトルネックにならない場合，入出力待ちキューを操作する DK スケジュール法の効果が見られないからである．

画像ファイル読み込みの終了時間（t2）について，測定の結果を図 3 (b) と図 4 (b) に示す．HTML ファイル読み込みの開始時間（t1）を速くすることにより，画像ファイル読み込みの終了時間（t2）が遅くなる可能性が高い．いい換えれば，WWW ページを要求してから表示が開始されるまでの時間を速くすることにより，WWW ページを要求してから表示が終了するまでの時間が遅くなる可能性が高い．画像の表示の遅れは，少ない程度であれば，大丈夫と考えられる．たとえば，利用者が速く表示されたテキストを読んでいる間に WWW ページの表示（つまり，画像の表示）が

終了すればよい．しかし，画像の表示が非常に遅くなると，利用者がページ全体の表示を待ちきれずページの閲覧を諦めてしまうことも考えられる．この実験では，画像ファイル読み込みの終了時間（t2）が最も悪い場合は，図3 (b) は，RW=7，図4 (b) は，RW=10のときである．特に悪い場合である図3 (b) の RW=7のときでも，画像ファイル読み込みの終了時間（t2）は 13%遅くなっている程度であり，大半は 10%以下の遅れで抑えられる．

以上の実験結果を次にまとめる．WWW ページ要求の間隔が固定である図3 (a) において，HTML ファイル読み込みの開始時間（t1）の平均値および中央値は DK スケジュール法を用いることによってそれぞれ最大 32%と 45%に改善された．また，WWW ページ要求の間隔がランダムである図4 (a) において，HTML ファイル読み込みの開始時間（t1）の平均値および中央値はそれぞれ最大 30%と 54%に改善された．さらに，図3 (b) と図4 (b) において，HTML ファイル読み込みの開始時間（t1）を速くしても，画像ファイル読み込みの終了時間（t2）に与える影響は，せいぜい 10%以下の画像表示の遅れで抑えられている．したがって，入出力待ちキューを操作する DK スケジュール法を用いることにより，WWW ページを要求してから表示が開始されるまでの時間を速くすることができるといえる．また，HTML ファイルの要求を処理するサーバプロセスを優先させても，つまり，WWW ページを要求してから表示が開始されるまでの時間を速くしても，WWW ページを要求してから表示が終了するまでの時間に与える影響は小さいといえる．

ところが，RW は RUN 状態と WAIT 状態の繰返し回数が多いか否かを判別するためのしきい値であるため，サーバプロセスの実行状況の変化とともに，その最適値は刻々と変化する．図3 と図4 では RW を固定値として扱ったが，刻々と変化する RW の最適値により近い値として，RW を自動的に更新することを考えた．このことは SLP についても同様である．そこで，SLP だけではなく RW も自動的に更新したときの WWW サーバの性能を評価する必要があり，実験3として行った．

なお，図4 において自動的に設定した SLP は，SLP の予測手法の実現アルゴリズムにより，次のように変化すると推測できる．HTML ファイルの要求を処理するプロセスは，利用者からの接続を待機しているプロセスなので，接続による待ち時間がディスク入出力などによる待ち時間（通常，ミリ秒単位）より長く，1秒以上と考えられる．このことを SLP の予測手法を実現したアルゴリズムに適用し，より正確な値の SLP を予測することとした．具体的には，SLP 予測単位時間において，SLP の予測手法によって算出した値が 1秒以上であれば，その値を新たな SLP として設定し利用する．そうでなければ，以前に予測した SLP を利用する．これにより，予測時に PFS として保有されている情報にディスク入出力による短い WAIT 状態の時間しかない場合でも，より正確な SLP の値を予測することができる．また，図4 においては，WWW ページの要求間隔が 10〜30 秒の間なので，自動的に設定された SLP は，10〜30 秒の範囲内を変動する．

表5 WWW ページ要求の間隔がランダムかつ SLP と RW を自動的に更新したときの実験結果の分析

Table 5 Analysis of experimental results when browsers access the WWW server randomly at the same time and SLP and RW are set automatically.

	disk sort (sec)	our method		
		RW = fix		RW = auto
		best (sec)	worst (sec)	(sec)
max	4.57	3.15 (+31%)	5.79 (−27%)	4.67 (− 2%)
mean	2.69	1.87 (+30%)	2.39 (+11%)	2.17 (+19%)
median	2.17	0.99 (+54%)	1.87 (+14%)	1.26 (+42%)
min	0.76	0.13 (+82%)	0.64 (+15%)	0.05 (+93%)

5.3 実 験 3

この実験は，SLP と RW をともに PFS を基に自動的に更新するようにしたものであり，その他の環境は実験2と同様である．なお，PFS 作成単位時間や PFS 利用時間や SLP および RW 予測単位時間はそれぞれ 500 ミリ秒とし，PFS の情報作成時に SLP と RW の予測も行うことにした．

HTML ファイル読み込みの開始時間（t1）の測定の結果を**表5** に示す．表5 には，HTML ファイル読み込みの開始時間（t1）の最大値，最小値，平均値，中央値のそれぞれについて最善と最悪の場合を示す．また，比較するために disksort を用いたときの結果と RW を固定したときの結果も表す．なお，RW を固定したときの結果を RW=fix を，RW を自動的に更新したときの結果を RW=auto と表す．また，括弧の中の数字は，実験2と同様に HTML ファイル読み込みの開始時間（t1）を disksort の場合と比べて，どの程度で改善（プラスで表す）または悪化（マイナスで表す）されたかを示す．測定の結果，RW を固定したときに 27%も悪化した最悪の場合の最大値に対して，RW を自動的に更新したときにわずか 2%しか悪化していなかった．さらに，最小値は，RW 固定と比べて最も小さい（93%も改善された）．また，平均値と中央値について，それぞれ 19%と 42%に改善されたと

いう良い結果が得られている．一方，画像ファイル読み込みの終了時間（t2）については，RW を自動的に更新したときにわずか 1%しか遅くならなかった．なお，この実験において自動的設定した RW は，1 つの Netscape プロセスが WWW サーバにアクセスする場合の PFS の分析により，次のように変化すると推測できる．HTML ファイルの要求を処理するサーバプロセスは，利用者からの接続待ちによる長い WAIT 状態の後に 3〜5 回の RUN 状態と短い WAIT 状態を繰り返す．したがって，自動的に設定された RW は，3〜5 の範囲内を変動する．

以上の結果により，利用者からのランダムアクセスで SLP を固定することが不可能であり，プロセスの処理内容の変化やサーバが処理する HTML ファイルの大きさによって変動する RW を固定することが不可能な一般の WWW システムの場合，SLP と RW を自動的に更新する提案スケジュール法を用いることによって WWW サーバの性能を改善できるといえる．

6. おわりに

提案ディスクスケジュール法について実現時の課題と対処を示した．スケジュール法の基本的な考え方は，ディスクがボトルネックになった場合，HTML ファイルの読み出し要求を処理するプロセスの入出力要求を優先してキューをつなぎ換えることである．このため，HTML ファイルの読み出し要求を処理するプロセスの検出手法が必要となり，プロセスが HTML ファイルの読み出し要求を処理するプロセスであるか否かについてプロセスの特徴を基に判断するためのしきい値 SLP と RW を利用した．さらに，複数の Netscape を用いて同時に WWW ページを要求したときについて，RW を 1 から 10 まで変化させた場合の評価を行った．このとき，提案したスケジュール法の効果を明白にするにため，利用者からの WWW ページ要求の間隔を固定した SLP の値より大きく設定した．さらに，より一般の場合として，WWW ページ要求の間隔がランダムのときについても行った．このとき，SLP については，PFS を基に予測して SLP 予測単位時間ごとに自動的に更新した．評価結果として，提案したプロセススケジュール法により，テキスト表示の開始が速くなるという期待した結果が得られた．具体的には，テキスト表示が開始されるまでの時間の平均値および中央値は，WWW ページ要求の間隔が固定のときにそれぞれ最大 32%と 45%，また，WWW ページ要求の間隔がランダムのときにそれぞれ最大 30%と 54%である．さらに，SLP と RW をともに自動的に更新した場合，つまり，利用者からのランダムアクセスで SLP を固定することが不可能であり，プロセスの処理内容の変化やサーバが処理する HTML ファイルの大きさによって変動する RW を固定することが不可能な一般の WWW システムの場合について評価した．評価により，テキスト表示が開始されるまでの時間の平均値および中央値は，それぞれ 19%と 42%という良い結果が得られた．

また，HTML ファイルの読み出し要求を処理するサーバプロセスを優先させても，つまり，WWW ページを要求してから表示が開始されるまでの時間を速くしても，WWW ページを要求してから表示が終了するまでの時間に与える影響は少ない．特に，SLP と RW をともに自動的に更新したときにわずか 1%しか遅くならなかった．

以上の評価は，OS が持つ入出力バッファのキャッシュがヒットしない状態で行ったが，ヒットする状態においても，同様に良い結果が得られると考えられる．なぜならば，キャッシュの単位は，ファイルではなく，ファイルを構成するブロックである．また，キャッシュの大きさの制限により，キャッシュヒット率は 100%ではない．このため，WWW サーバへ要求するファイルは，キャッシュ内にあったとしても，そのファイルを構成する全ブロックの一部だけである場合が多い．そこで，ファイルの大きさに比例したキャッシュヒット率と仮定すると，たとえば，20 キロバイトの HTML ファイルと 200 キロバイトの画像ファイルのキャッシュヒット率がともに 50%の場合の動作は，10 キロバイトの HTML ファイルと 100 キロバイトの画像ファイルのキャッシュヒット率がともにゼロの場合の動作と同様であると考えられる．

したがって，一般の WWW システムは，提案スケジュール法を用いることによって WWW サーバの性能を改善できるといえる．

残された課題として，ディスク入出力待ちキューの長さに影響を与える画像ファイルの大きさを変化させたときの WWW サーバの性能の評価，提案したプロセッサスケジュール法とディスクスケジュール法を同時に使用する場合の WWW サーバの性能の評価，および通信路に着目した場合の提案スケジュール法の実現・評価がある．

参考文献

1) 中村 豊，知念賢一，砂原秀樹，山口 英：ENMA：パケットモニタによる WWW サーバの性能計測システムの設計と実装，信学論（D-I），

Vol.J83-D-I, No.3, pp.329–338 (2000).
2) Xu, J. and Parnas, D.: On Satisfying Timing Constraints in Hard-real-time Systems, *IEEE Trans. Softw. Eng.*, Vol.19, No.1, pp.70–84 (1993).
3) Härbour, M., Klein, M. and Lehoczky, J.: Timing Analysis for Fixed-priority Scheduling of Hard Real-time Systems, *IEEE Trans. Softw. Eng.*, Vol.20, No.1, pp.13–28 (1994).
4) Burns, A., Tindell, K. and Wellings, A.: Effective Analysis for Engineering Real-time Fixed Priority Schedulers, *IEEE Trans. Softw. Eng.*, Vol.21, No.5, pp.475–479 (1995).
5) Feng, W. and Liu, J.: Algorithms for Scheduling Real-time Tasks with Input Error and End-to-end Deadlines, *IEEE Trans. Softw. Eng.*, Vol.23, No.2, pp.93–106 (1997).
6) Waldspurger, C. and Weihl, W.: Stride Scheduling: Deterministic Proportional-share Resource Management, Technical Report MIT-LCS-TM-528, MIT laboratory for computers science (June 1995).
7) Ford, B. and Susarla, S.: CPU Inheritance Scheduling, *Proc. 2nd USENIX Symposium on Operating Systems Design and Implementation*, pp.91–106 (Oct. 1996).
8) 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 (1999).
9) Suranauwarat, S. and Taniguchi, H.: Evaluation of a Process Scheduling Policy for a WWW server Based on Its Contents, *IEICE Trans. Inf. & Syst.*, Vol.E83-D, No.9, pp.1752–1761 (2000).
10) Suranauwarat, S. and Taniguchi, H.: Operating Systems Support for the Evolution of Software: An Evaluation Using WWW Server Software, *Proc. 2000 International Symposium on Principles of Software Evolution*, pp.289–298 (Sep. 2000).
11) Deitel, H.: *An Introduction to Operating Systems*, 2nd ed., Addison-Wesley (1990).
12) Silberschatz, A. and Galvin, P.: *Operating System Concepts*, 5th ed., John Wiley & Sons (1997).
13) Worthington, B.: Scheduling Algorithms for Modern Disk Drives, *Proc. 1994 Conference on Measurement and Modeling of Computer Systems*, pp.241–251 (1994).
14) Geist, R.: A Continuum of Disk Scheduling Algorithms, *ACM Trans. Comput. Syst.*, Vol.5, No.1, pp.77–92 (Feb. 1987).
15) Apache HTTP Server Project. http://www.apache.org/
16) 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).
17) Comer, D.: *Computer Networks and Internets*, 2nd ed., Prentice Hall(1999).
18) Arlitt, M. and Williamson, C.: Internet Web Servers: Workload Characterization and Performance Implications, *IEEE/ACM Trans. Networking*, Vol.5, No.5, pp.631–645 (Oct. 1997).
19) Woodruf, A., Aoki, P., Brewer, E. and Gautheir, L.: An Investigation of Documents from WWW, *Proc. 5th World Wide Web Conference* (May 1996).
20) Cunha, C., Bestavros, A. and Crovella, M.: Characteristics of WWW Client-based Traces, Technical Report BU-CS-95-010, Boston University, Computer Science Department (1995).
21) Sedayao, J.: Mosaic Will Kill My Network!, *Electronic Proc. 2nd World Wide Web Conference: Mosaic and Web* (Oct. 1994).

(平成 12 年 12 月 19 日受付)
(平成 13 年 4 月 6 日採録)

スカンヤ・スラナワラッ

平成 8 年九州大学工学部情報工学科卒業．平成 11 年同大学大学院修士課程修了．現在，同大大学院システム情報科学研究科博士後期課程に在学中．オペレーティングシステムに興味を持つ．

谷口　秀夫（正会員）

昭和 53 年九州大学工学部電子工学科卒業．昭和 55 年同大学大学院修士課程修了．同年日本電信電話公社電気通信研究所入所．昭和 62 年同所主任研究員．昭和 63 年 NTT データ通信（株）開発本部移籍．平成 4 年同本部主幹技師．平成 5 年九州大学工学部助教授．平成 8 年九州大学システム情報科学研究科助教授．博士（工学）．オペレーティングシステム，分散処理に興味を持つ．著書「オペレーティングシステム」（昭晃堂）．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員．