2007－OS－106 (13)
2007／8／3

ホームページの構成ファイルを優先的にバッファキャッシュに保持する制御法の評価

小峠 みゆき　田端 利宏　谷口 秀夫
岡山大学 大学院自然科学研究科

Webは，多くのコンテンツを保有している．Webサイトには，入り口となるページ(多くの場合，ホームページである)が存在し，多くの利用者は，そのホームページから様々なサービスを利用することが多い．このため，Webサービスにおいて，ホームページのアクセス応答時間を短くすることは重要である．我々は，特定ディレクトリ内のファイルを優先的にWebサーバのバッファキャッシュに残す制御法を提案している．Webサーバには，意味のあるファイル群毎にディレクトリに格納されることが多い．本論文では，提案したバッファキャッシュ制御法について述べ，岡山大学のWebサーバについて，各部局のホームページを構成するファイルのアクセス応答時間を評価し，有効性を示す．

Evaluation of a Buffer Cache Mechanism for Homepage Contents

Miyuki Kotoge　Toshihiro Tabata　Hideo Taniguchi
Graduate School of Natural Science and Technology, Okayama University

Web site contains various contents. It also has an entrance page of the web site. Many internet users tend to browse the various contents from the entrance page. Therefore, it is important to shorten the access response time of the entrance page for web service. To meet this requirement, we proposed a buffer cache control method that stores files of specific directories in buffer cache on a web server. We focused attention on that the web server stores similar types of files for a same directory. In this paper, we describe the buffer cache control method, and evaluate access response time at each entrance page of department of Okayama University with our method. From the result of this e valuation, we show effectiveness of our method.

1 はじめに

Webは，様々なサービスで利用されており，多くのコンテンツを保有している．サービス提供時に，頻繁にアクセスされると，対象のファイルをWebサーバから毎回送信することになり，コンテンツの応答時間が長くなってしまう．また，多くの利用者に集中的にアクセスされると，通信路の負荷が上昇してしまう．これらの問題を解決するため，主に3種類のキャッシュが存在する．1つ目は，利用者側のブラウザのキャッシュである．2つ目は，プロキシサーバのキャッシュである．3つ目は，Webサーバのディスクキャッシュである．

1つ目のブラウザのキャッシュとは，一度表示したページを利用者のハードディスクに蓄えておく機能である．2つ目のプロキシサーバとは，Web上で提供されているWebサイトなどのコンテンツの複製を蓄積し，利用者からの要求があったときに本来のサーバに代わって配信することにより，ネットワークのトラフィックやサーバの負荷の分散を図るサーバのことである．3つ目のWebサーバのディスクキャッシュとは，Web上のコンテンツを提供しているサーバが，リクエストされたコンテンツをメモリ上に保存しておく領域のことで

ある．

ここでは，ディスクキャッシュを対象とし，その効率化について議論する．以降では，リクエストされたファイルのデータを保存するキャッシュをバッファキャッシュと呼ぶ．バッファキャッシュを管理するブロック置き換えアルゴリズムとして，従来のOSでは，バッファキャッシュをLRU方式で管理することが一般的である．既存の多くのOSは，固定長のブロックを単位として，LRU方式を用いている．LRU方式は，頻繁にアクセスされるブロックをキャッシュに残すことができるため，キャッシュヒット率が高く，効率がよいことが知られている．

ブロック置き換えアルゴリズムについて，現在までに様々な研究[1]～[8]が行われてきた．ブロック置き換えアルゴリズムは，大きく3つに分類される．1つ目は，各ブロックの参照順序や参照頻度に基づいてブロック置き換えを行う方式である．この方式の代表的な例として，LRU，FIFO，LFU，FBR[1]，LRU-k[2]，IRG[3]が存在する．2つ目は，利用者によってあらかじめ提供されたブロック参照パターンに基づいてブロック置き換えを行う方式である．この方式の代表的な例として，ACFC[4]，UBM[5]が存在する．3つ目は，ブロック参照の定期性に基づいてブロック置き換えを行う方式である．この方式の代表例として，2Q[6]，SEQ[7]，EELRU[8]が存在する．これらに対し，我々は，特定のディレクトリに格納されたファイル群を優先的にキャッシュし，その処理を高速化するバッファキャッシュ制御法を提案した[9]．提案方式は，優先処理と非優先処理が混在する環境で，優先処理の性能低下を抑制すると期待できる．

Webサイトには，入り口となるページ(多くの場合，ホームページである)が存在し，多くの利用者は，そのホームページから様々なサービスを利用することが多い．このため，Webサービスにおいて，ホームページへのアクセス応答時間を短くすることは，Webサーバ管理者にも重要である．しかしながら，既存のバッファキャッシュ制御法は，ブロックを意識しており，特定のファイルやファイル群を優先的に扱うものではない．ファイルは，意味のあるデータの単位であり，意味のあるファイル群毎にディレクトリに分けて格納されることが多い．

そこで，本論文では，文献[9]の方式を利用し，ホームページに関連するファイルの格納ディレクトリを優先指定することで，自動的にそのページにアクセスする処理を優先処理として扱うこととし，その有効性を評価する．これにより，写真や動画などの他の大きなサイズのファイルを集中的に要求されたとしても，ホームページの応答時間にあまり影響を与えないことが期待できる．

2 ファイルの格納ディレクトリを考慮したバッファキャッシュ制御法

2.1 機能概要

文献[9]で提案したディレクトリ優先方式と名付けたバッファキャッシュ制御法は，ブロック単位ではなく，ファイルを単位として，優先処理のバッファキャッシュのブロックを優先してキャッシュする手法である．ファイルは，意味のある単位でディレクトリに格納されることが多い．また，個々のファイルを設定する方法では，設定に手間がかかる．このため，本方式では，ディレクトリ単位でファイルを指定する．具体的には，指定されたディレクトリの直下に存在するファイルの内容を優先的にバッファキャッシュに保存する．なお，指定したディレクトリの直下にあるファイルを優先ファイルと呼び，その他のファイルを非優先ファイルと呼ぶ．

ディレクトリ優先方式の概要を図1に示す．ディレクトリ優先方式では，バッファキャッシュを保護プールと通常プールに分割する．保護プールには，優先ファイルのブロックが格納される．一方，通常プールには，非優先ファイルのブロックが格納される．保護プールは，通常プールに比べ，優先的にバッファキャッシュに残されるため，優先ファイルを扱う処理は，ディスクI/Oに伴うオーバヘッドを抑制することができる．ディレクトリ優先方式は，優先ディレクトリ情報の管理機能，読み込みデータへのバッファの割当機能，及び不要なバッファの解放機能からなる．各機能の概要を述べる．

(1) 優先ディレクトリ情報の管理
指定したディレクトリ（以降，優先ディレクトリ）の情報を管理する管理表（以降，優先ディレクトリ情報管理表）を持ち，バッファを割り当てる際の検索に使用する．

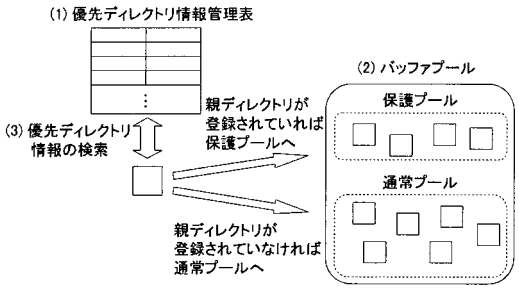

図 1 ディレクトリ優先方式の概要

(2) バッファの割当
優先ディレクトリ情報管理表を参照して，読み込んだブロックに対応するファイルの親ディレクトリ情報が登録されているかどうか検索する．その判定に基づいて，そのブロックが持つバッファを保護プールまたは通常プールへ移す．

(3) バッファの解放
新規のブロック読み込みが発生した時に，通常プール中の最も古くに参照されたバッファを解放する．その際に通常プール中にバッファが存在しない場合は，保護プール中の最も古くに参照されたバッファを解放する．

ブロック置き換え時には，通常プールのブロックから優先して置き換えられる．保護プールにあるブロックは，通常プールにあるブロックがすべてなくならない限り，置き換えられることはない．以降，保護プールと通常プールを合わせたものをバッファプールと呼ぶ．

2.2 処理の流れ

新たにブロックを読み込む時のディレクトリ優先方式のバッファ割当と解放処理の流れを述べる．

(1) 空きバッファを確保する．このために，バッファ解放処理を呼び出し，通常プールにバッファが存在すれば，通常バッファの最も古く参照されたバッファを解放する．そうでなければ，保護プールから最も古く参照されたバッファを解放する．

(2) 読み込んだブロックをバッファとして登録する．最初に読み込んだブロックに対応する親ディレクトリの情報を取得する．親ディレクトリが優先ディレクトリであれば保護プール，そうでなければ通常プールにバッファを登録する．

OS のバッファキャッシュを保護プールと通常プールに分割し，次の処理を行う．

(1) 優先ファイルは，保護プールに保存する．

(2) 非優先ファイルは，通常プールに保存する．

(3) 保護プールの大きさは，必要に応じて大きくなる．ただし，バッファキャッシュの大きさを超えることはない．

(4) 通常プール内の処理は，LRU に基づく．

(5) 保護プールの大きさが最大となり，通常プールの大きさが 0 の場合，操作対象ファイルが優先ファイルか非優先ファイルか関わらず，保護プールの中で最も古くに使用されたブロックを置き換える．

2.3 効果

2.3.1 利用方法

ディレクトリ優先方式の導入は，ファイル毎にバッファキャッシュの優先度を導入することを意味するため，設定においては，CPU 利用の優先度と矛盾が起きないように設定する必要がある．以降では，提案方式を有効に利用できる利用方法を示す．

(1) 特定の処理を優先して実行する場合
その処理がよく利用するファイルが格納されたディレクトリを優先ディレクトリとして指定することで，その処理を高速に実行できる．また，非優先処理のディスク I/O 処理による影響を抑制できる．優先したい処理の CPU 利用の優先度が高い場合は，ディレクトリ優先方式を併用することでさらに大きな効果を得ることができる．優先処理と非優先処理の CPU 利用の優先度に差がない場合でも，適切に優先ディレクトリを指定することで，優先実行の効果を得ることができる．優先制御の例として，make コマンドによるプログラムのコンパイル処理において，ヘッダファイル

が格納されているディレクトリを指定することがある．なお，スクリプト等を利用することで，優先処理の走行時のみ，当該ディレクトリを優先ディレクトリとして指定ことができる．

(2) 複数のプロセスから構成されるサービスで，リクエスト毎に処理の優先度を変更したい場合
Webサーバのように，リクエストがファイルに対応しており，その処理を別プロセスで処理する場合，リクエストの内容（ファイル）毎に処理の優先度を変更したい場合がある．このような優先処理をAPで実現するためには，AP自体を改版し，リクエストの内容を解析する必要がある．一方，ディレクトリ優先方式を利用すれば，リクエストされるファイル毎に優先度を決定することができる．

以上に述べたように，特定のプロセス群の処理をより速く実行したい場合，ファイルに対応した処理を高速に実行したい場合にディレクトリ優先方式は有効である．

また，ディレクトリ優先方式を有効に利用するための設定例を以下に述べる．

(1) 開発者が，その処理がアクセスするファイル群の情報を提示する，もしくは設定を指定しておくことが考えられる．例えば，プログラムの起動スクリプトに，優先ディレクトリの指定処理を加えることが考えられる．

(2) 計算機管理者または運用者が，プログラムの導入時に優先ディレクトリを指定することが考えられる．例えば，Webサービスの導入時に，リクエストを優先して処理したいページを選択し，その設定を行うことがある．

(3) プログラム利用者が優先ディレクトリを指定する方法がある．プログラムの知識がある人であれば，処理内容を予測してうまく設定することができる．例えば，プログラムのmake処理において，includeファイルが格納されたディレクトリを設定することがある．また，トレースにより，プログラムがよくアクセスするファイルを調査し，優先ディレクトリを指定する方法もある．

優先処理が非優先処理に与える影響を抑制するには，優先処理の実行時のみ優先ディレクトリの指定を行うことがある．また，異なるCPU利用の優先度を持つプロセス群が同時に走行する場合，優先度の逆転をなくすため，優先度の高いプロセス群にのみディレクトリ優先を適用することが必要である．

2.3.2 期待される効果

提案方式を有効に利用した際に期待される効果について，以下に述べる．

(1) 複数処理実行時の優先処理への影響抑制
優先処理以外に，非優先処理が実行されていた場合，非優先処理のI/O処理が優先処理に及ぼす影響を抑制できる．

(2) 利用者が明示的に優先処理を指定可能
優先ディレクトリに格納されたファイルを扱う処理が，優先的にバッファキャッシュを利用するように指定可能となる．また，優先的に処理したいファイルを利用者が一つのディレクトリに集めて，そのディレクトリを優先ディレクトリに指定することでも，優先処理を指定できる．

(3) APを修正することなくサービスの優先制御を実現可能
サービス処理の重要度をアクセスするファイル群を単位として，設定することが可能となる．例えば，Webサービスで待ち時間をできるだけ短くしたいページがあれば，提案方式を用いて，そのコンテンツの格納ディレクトリを優先指定することで，自動的にそのページにアクセスする処理を優先処理として扱うことができる．

3 評価

3.1 評価環境

評価は，以下の計算機を用いた．

(1) OS: FreeBSD 4.3-RELEASE

(2) CPU: Celeron 2.8GHz

(3) 搭載メモリ: 256MB

表 1 1ヶ月間の要求回数が 20 以上のファイルデータ

	全体	優先ディレクトリ	優先ページ
ディレクトリ数 (個)		14	14
ファイル数 (個)	16,657	920	117
合計要求回数 (回)	13,935,112	8,126,455	2,955,024
合計ファイルサイズ (MB)	1,349.3	13.2	0.7

表 2 測定で使用したファイルデータ (100,000 回要求)

	全体	優先ディレクトリ	優先ページ
ディレクトリ数 (個)		14	14
ファイル数 (個)	8,844	810	114
合計要求回数 (回)	100,000	58,159	21,346
合計ファイルサイズ (MB)	598.9	10.4	0.7

また，実験に用いた計算機は，100BASE-TX の Ethernet を搭載し，ハブを経由して接続されている．

3.2 評価目的

この評価の目的は，実運用されているサービスに，ディレクトリ優先方式を適用した場合の効果を明らかにすることである．評価では，実運用されている Web サーバとして，岡山大学の Web サーバ (www.okayama-u.ac.jp) を取り上げ，提案した制御法の有効性を確認する．Web サーバには，ホームページが存在し，多くの利用者は，そのホームページから様々なサービスを利用することが多い．このため，Web サービスにおいて，ホームページの応答時間を短くすることは重要である．平均応答時間を短くすることも重要であるが，特に最長の応答時間を短くすることは，利用者のストレスを軽減する上でも重要と考えられる．

そこで，本評価では，岡山大学の Web サーバに実際に構築されているディレクトリ構成と頻度を再現し，各部局のホームページを構成するファイルのアクセス応答時間を評価対象とした．具体的には，岡山大学，文学部，教育学部といった各部局のホームページを構成するファイルを直下に持つディレクトリを優先ディレクトリに指定することにより，各部局のホームページへのアクセスの応答時間が向上することを示す．

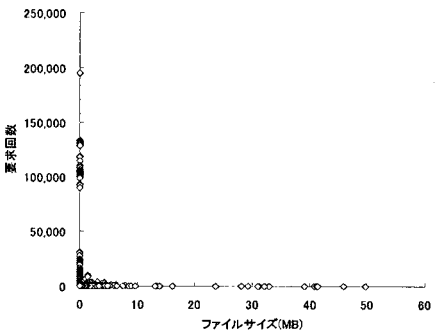

図 2 ファイルサイズと 1ヶ月の要求回数

3.3 評価方法と結果

岡山大学の Web サーバについて，1ヶ月間 (2006 年 7 月のデータを使用) に要求回数が 20 以上のファイルに関する情報を表 1 に示す．また，ファイルの大きさと要求回数の関係を図 2 に示す．表 1 からわかるように，総要求回数は 13 百万を超える．各部局のホームページを構成するファイルを直下に持つディレクトリ (優先ディレクトリ) は 14 個であり，その直下には 920 個のファイルがある．なお，各部局のホームページを構成するファイル (優先ページ) は 117 個である．つまり，約 800 個のファイルが，各部局のホームページを構成するファイルではないにもかかわらず，優先ディレクトリの直下に存在し，優先的にキャッシュされてしまう状況にある．

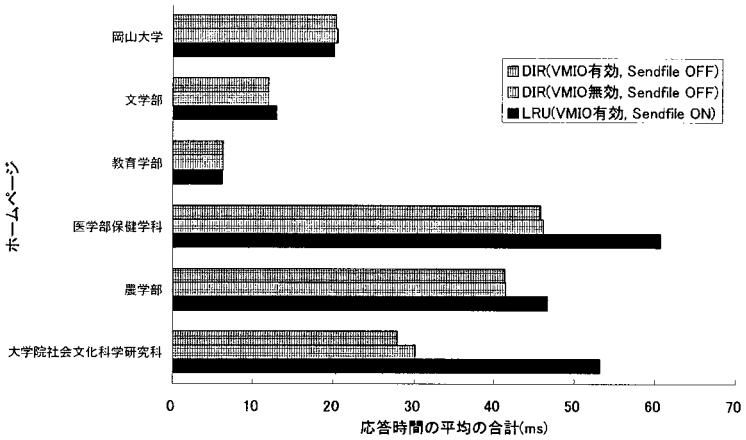

図 4 優先ページを構成するファイルの応答時間の平均の合計

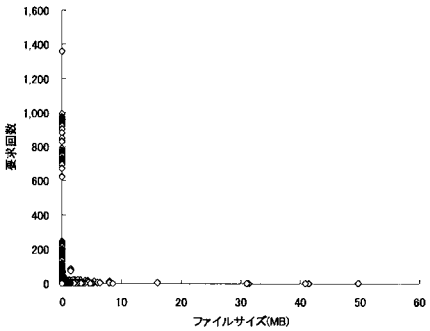

図 3 ファイルサイズと要求回数 (100,000 回)

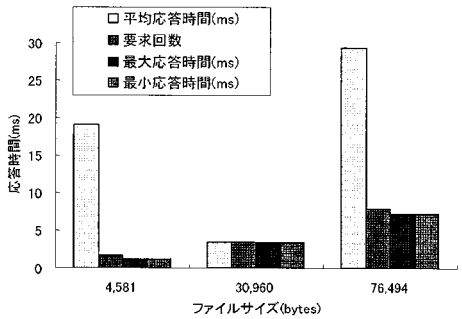

図 5 文学部のトップページを構成するファイルの応答時間（VMIO 有効，LRU 方式，sendfile ON）

なお，以降では，要求回数を抑制し実験の効率をあげるため，次の対処を行った．期間 1ヶ月の場合の要求回数の頻度により，ファイルを確率的に選択しアクセスする．このアクセスを繰り返し行い，期間 1ヶ月の場合のファイルの大きさと要求回数の関係（図 2）と同様な傾向を有する要求回数として，100,000 回を設定した．この要求回数 100,000 回の場合について，ファイルの大きさと要求回数の関係を図 3 に示す．また，この場合のファイルに関する情報を表 2 に示す．

実験環境を定常状態として評価を行うため，100,000 回の要求を行い，その後 100,000 回の要求を行ったときの応答時間を図 4 に示す．ここで，図 4 の応答時間とは，ホームページを構成する各ファイルの平均応答時間を合計したものである．また，このときの，各部局のホームページへのアクセス状況を表 3 に示す．図 4 と表 3 から，以下のことがわかる．

(1) 図 4 より，文学部，医学部保健学科，農学部，および大学院社会文化科学研究科においては，VMIO 機能の有無に関係なく，DIR 方式は LRU 方式（VMIO 機能と sendfile 機能が有効）より，応答時間が短い．これは，表 3 からわかるように，これらのホームページは，その大きさが比較的大きく，要求回数が少ないためである．

(2) また，文学部に比べ，医学部保健学科，農学部，および大学院社会文化科学研究科は，応答時間が大きく短縮されている．特に，大学院社会文化科学研究科では，約 43.2%短縮されている．この現象を分析するため，LRU 方式について，ホームページを構成する各ファ

表 3 各ホームページへのアクセスに関する情報

ページ	構成ファイルの合計サイズ (B)	ファイル数	各ファイルの平均応答時間の合計	各ファイルの要求数の合計	1 ファイル当たりの要求数
岡山大学	59,349	19	20.109	18,027	948.8
文学部	112,035	3	13.017	103	34.3
教育学部	54,348	2	6.211	129	64.5
医学部保健学科	126,709	44	60.743	1,430	32.5
農学部	252,739	24	46.602	799	33.3
大学院社会文化科学研究科	109,773	26	53.172	739	28.4

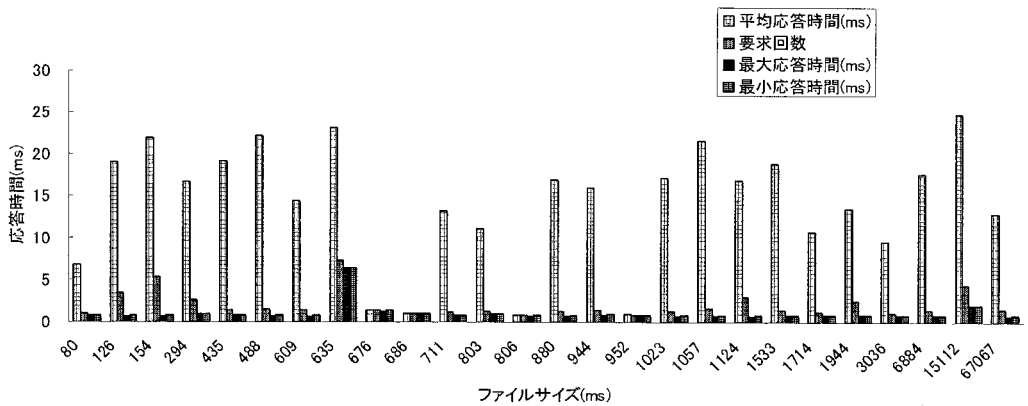

図 6 社会文化科学研究科のトップページを構成するファイルの応答時間（VMIO 有効，LRU 方式，sendfile ON）

イルの応答時間を図 5 と図 6 に示す．図 5 の文学部では，3 つのファイルの中で 2 つのファイル（66%）について，最大時間が大きくなっている．これに対し，図 6 の大学院社会文化科学研究科では，26 個のファイルの中で 21 個のファイル（80%）について，最大時間が大きくなっている．したがって，ホームページを構成するファイルの大きさの合計は同様であっても，構成するファイル数が多いと応答時間を短縮できる程度が大きいといえる．なお，DIR 方式の文学部の場合について，ホームページを構成する各ファイルの応答時間を図 7 に示す．図 7 より，DIR 方式では応答時間の変動が小さいことがわかる．

(3) 上記に対し，教育学部においては，DIR 方式と LRU 方式で応答時間に差がない．これは，表 3 からわかるように，このホームページは，その大きさが比較的小さいためである．

(4) これらに対し，岡山大学においては，DIR 方式は LRU 方式より，僅かに応答時間が長くなっている．これは，表 3 からわかるように，このホームページは，その大きさが比較的小さく，かつ要求回数が非常に多いためである．つまり，LRU 方式でもキャッシュされるようなファイルは，DIR 方式により，僅かであるが応答時間が長くなってしまう．

以上の評価により，DIR 方式は LRU 方式に比べ，各部局のホームページへのアクセスの応答時間を短縮することが期待できる．また，バッファキャッシュのミスヒットを減らすことで，応答時間のバラツキを小さくでき，最大応答時間が長くなる場合が少ないと推察できる．特に，大学院社会文化科学研究科では，最長の応答時間を大きく削減

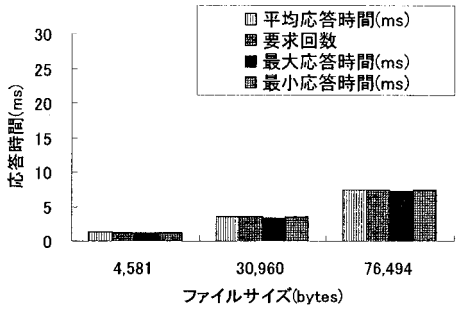

図 7 文学部のトップページを構成するファイルの応答時間（VMIO 有効，DIR 方式）

できていることから，平均応答時間を約 43.2%短縮できた．

4 おわりに

Web のホームページのアクセス応答時間を短縮させるため，ディレクトリ優先方式を Web 環境に適用し，評価結果について述べた．ディレクトリ優先方式は，特定のコンテンツを優先的にキャッシュし，その処理を高速化することができる．

岡山大学の Web サーバについて，各部局のホームページを構成するファイルのアクセス応答時間を評価した．ディレクトリ優先方式を用いることで，ホームページを構成するファイルを優先的にバッファキャッシュに保持することができる．これにより，平均応答時間を最大で約 43.2%短縮でき，最大応答時間を短くでき，応答時間のバラツキを小さくできることを示した．

謝辞 プログラムの実装にご協力いただいた齊藤 圭氏（岡山大学大学院自然科学研究科）に感謝します．

参考文献

[1] J. T. Robinson and M. V. Devarakonda, "Cache Management Using Frequenfcy-Based Repalacement," Proc. the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp.134–142 (1990)

[2] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-k Page Replacement Algorithm for Database Disk Buffering," Proc. the 1993 ACM SIGMOD Conference, pp.297–306 (1993)

[3] V. Phalke and B. Gopinath, "An Inter-Reference Gap Model for Temporal Locality in Program Behavior," Proc. the USENIX Summer 1994 Technical Conference, pp.291–300 (1995)

[4] P. Cao , E. W. Falten. and K. Li, "Application Controlled File Caching Policies," Proc. the USENIX Summer 1994 Technical Conference, pp.171–182 (1994)

[5] J. M. Kim, J. Choi, J. Kim, and Sam H. Noh, "A Low-Overhead, High-Performance Unified Buffer Management Scheme That Exploits Sequential and Looping References," Proc. 4th Symposium on Operating System Design and Implementation (OSDI 2000), pp.119–134 (2000)

[6] T. Johnson and D. Shasha, "A Low Overhead High Performance Buffer Management Replacement Algorithm," Proc. the 20th International Conference on VLDB, pp.297–306 (1993)

[7] G. Glass and P. Cao, "Adaptive Page Replacement Based on Memory Reference Behavior," Proc. the 1997 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp.115–226 (1997)

[8] Y. Smaragdakis, S. Kaplan, and P. Wilson, "Simple and Effective Adaptive Page Replacement," Proc. the 1999 ACM SIGMETRICS Conference on Mesurement and Modeling of Computer Systems, pp.122–133 (1999)

[9] 田端利宏, 小峠みゆき, 齊藤圭, 乃村能成, 谷口秀夫, "ファイルの格納ディレクトリを考慮したバッファキャッシュ制御法," 情報処理学会コンピュータシステムシンポジウム, 情報処理学会コンピュータシステムシンポジウム論文集, pp.53–62 (2006)