B_036

ディレクトリに着目したバッファキャッシュ制御法の評価
Evaluation of a Directory Oriented Buffer Cache Mechanism

小峠 みゆき Miyuki Kotoge　齊藤 圭 Kei Saitou　田端 利宏 Toshihiro Tabata　乃村 能成 Yoshinari Nomura　谷口 秀夫 Hideo Taniguchi

1. はじめに

従来のオペレーティングシステム（以降，OS）では，バッファキャッシュを入出力の単位であるブロック単位で LRU 管理することが一般的である．一方，利用者は，ファイルという抽象度の一段高いレベルでデータを扱うため，ファイル操作の観点からバッファキャッシュを管理制御する方式が考えられる．

そこで，我々は，ファイルとブロックの対応関係を考慮して，指定ディレクトリ下のファイルを優先的にキャッシュするバッファキャッシュ制御法（以降，ディレクトリ優先方式）を提案している[1]．

本稿では，ファイル入出力例として Web サーバを取り上げ，提案方式の評価結果を報告する．

2．ディレクトリ優先方式

ディレクトリ優先方式[1]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方式である．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．本方式の主な特徴を以下に示す．

(1) 利用者は，優先的に扱うディレクトリを指定できる．
(2) 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．
  (A) 優先ファイルは，保護プールに保存する．
  (B) 非優先ファイルは，通常プールに保存する．
  (C) 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．
  (D) 通常プール内の処理は，LRU に基づく．
  (E) 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．

3．評価と考察

3．1 環境

ディレクトリ優先方式を FreeBSD 4.3-RELEASE に実装した．評価の環境を図 1 に示し，以下に説明する．

サーバ：
(1) OS：FreeBSD 4.3-RELEASE
(2) CPU：Celeron（2.80GHz）
(3) LAN：100base-TX
(4) メモリ：32MB
(5) バッファキャッシュの大きさ：6.3MB

岡山大学 大学院自然科学研究科

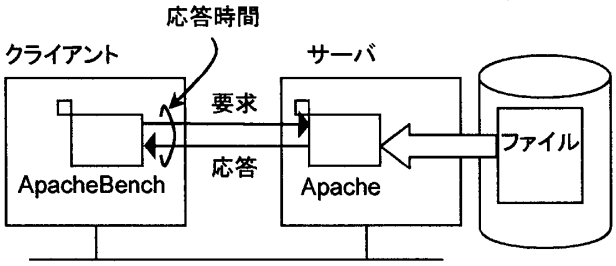

図 1　実験の概要

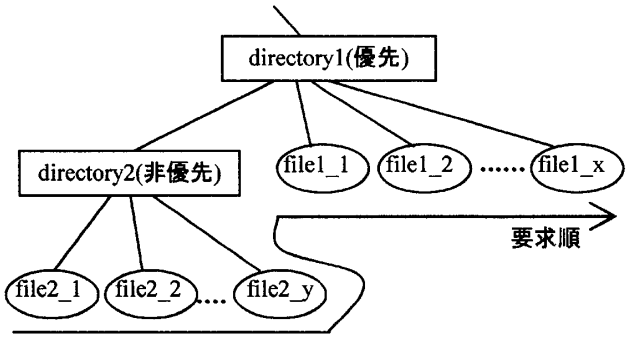

図 2　テストファイルの位置関係と要求順

システム維持のために常時確保されるスペースが存在するため，実際に使用できる大きさは約 5.7MB である．
(6) Web サーバソフトウェア：Apache 2.0.55

クライアント：
(1) OS：FreeBSD 4.3-RELEASE
(2) CPU：Celeron（2.80GHz）
(3) LAN：100base-TX

なお，キャッシュ内に該当するブロックが見つからなかった場合，実メモリ上に該当するページがあるかどうかを探索する機能を無効にして測定を行った．これは本方式の効果範囲を明確化するためである．

3．3 実験方法

クライアントからサーバ上のファイルを読み出す．サーバ上のファイルの様子を図 2 に示す．なお，ApacheBench を利用して応答時間を測定した．図 2 に示す通り，ディレクトリを 2 個用意し，一方にはファイルを x 個，もう一方にはファイルを y 個配置する．ファイルの大きさは，それぞれ 1MB とする．実験の手順を以下に述べる．

(手順 1) directory1 を優先的に扱うディレクトリに指定し，directory2 は特に指定しない．以降，directory1 を

優先ディレクトリと呼び，directory2 を非優先ディレクトリと呼ぶ．

(手順 2) 非優先ディレクトリ下のファイル y 個，次に優先ディレクトリ下のファイル x 個の順でファイルの読み出しを要求し，応答時間を測定する．

(手順 3) 上記の 2 つの手順を 100 回繰り返す．

なお，上記の実験を x, y の値を変化させて測定を行う．

3. 2 測定項目

Apache 2.0.55 には，バッファキャッシュを介さずにファイルのデータ転送を行う sendfile システムコールを利用する機能がある．以降，この機能を sendfile 方式と呼ぶ．sendfile 方式は，バッファキャッシュを使用せず，空きの実メモリを利用する．測定では，従来の LRU 方式を加え，以下の 3 方式を比較する．

(方式 1) LRU 方式

(方式 2) sendfile 方式

(方式 3) ディレクトリ優先方式

次に，測定に際し，バッファキャッシュの大きさと空きメモリの大きさ，および x や y の関係に着目し，測定ケースの場合分けを行う．場合分けした様子を図 3 に示す．直線はバッファキャッシュの大きさ，点線は空きメモリの大きさを示す．当然のことながら，空きメモリの大きさは，他プロセスの実行状態により増減する．

(場合 1), (場合 2), (場合 3)は，優先ファイルの大きさの合計(1 回当り)がバッファキャッシュの大きさよりも小さい場合である．(場合 4), (場合 5)は，優先ファイルの大きさの合計(1 回当り)がバッファキャッシュの大きさより大きい場合である．(場合 2), (場合 4)は，優先ファイルの大きさの合計(1 回当り)がバッファキャッシュの大きさより小さく，要求するファイルの大きさの合計(1 回当り)がバッファキャッシュの大きさ以上で空きメモリの大きさ以下の場合である．(場合 3), (場合 5)は，要求するファイルの大きさの合計(1 回当り)が空きメモリの大きさより大きい場合である．

3. 4 結果と考察

表 1 に，優先ファイルの平均応答時間を示す．括弧の中は，LRU 方式と比較して何%応答時間が短縮されたかを示す．

(場合 1), (場合 2), (場合 3)では，ディレクトリ優先方式は応答時間が変化していない．この理由は，優先ファイルのブロックはバッファキャッシュ上にすべて存在し，要求する非優先ファイルの大きさに関係なくディスク I/O が発生しないからである．

(場合 4), (場合 5)では，ディレクトリ優先方式は LRU 方式より約 2～4%遅い．ディレクトリ優先方式は，優先ファイルならば保護プールに移し，非優先ファイルならば通常プールへ移す処理を行う．この処理のオーバーヘッドと考えられる．

(場合 3), (場合 5)では，sendfile 方式は LRU 方式よりも 10%遅い．したがって，sendfile システムコールでディスク I/O が発生する場合は，read システムコールでディスク I/O が発生する場合よりも遅くなるといえる．このため，(場合 3)では，ディレクトリ優先方式よりも 37% (123/90*100 – 100)遅くなっている．

以上により，要求するファイルの大きさがバッファキャッシュの大きさよりも小さいファイル群を優先ディレクトリ下に配置すれば，そのファイル群はバッファキャッシュに残り続けるので，ディスク I/O は最初の一度のみでよく，応答性能は落ちないことを確認できた．つまり，他の大きな大きさのファイルを要求されても，優先ファイルの応答性能に影響はない．一方，sendfile 方式は，空きメモリの大きさが大きい場合は有効であるが，空きメモリの大きさが小さくなる(相対的にいえば，大きなファイルを操作する)と応答性能が大きく低下するといえる．

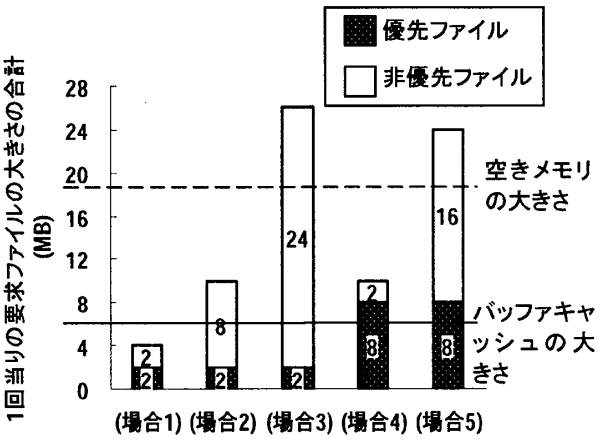

図 3 測定ケースの場合分け

表 1 測定結果

	平均優先ファイル応答時間(ms)		
通番	LRU	sendfile	ディレクトリ優先
(場合 1)	90	90 (-1%)	90 (±0%)
(場合 2)	108	90 (-21%)	90 (-19%)
(場合 3)	110	123 (+10%)	90 (-22%)
(場合 4)	111	90 (-24%)	116 (+4%)
(場合 5)	112	124 (+10%)	114 (+2%)

4. おわりに

指定したディレクトリ下のファイルを優先的にキャッシュすることに着目したバッファ制御法の評価を行った．Web サーバで測定を行った結果，提案方式は，従来方式に比べ，応答時間が最大 22%短縮できることを明らかにした．また，sendfile 方式よりも，最大 37%応答時間を短縮できることを明らかにした．

残された課題として，実環境を事例とした評価がある．

謝辞 本研究の一部は，科学研究費補助金 基盤研究(B)「適応性と頑健性を有する基盤ソフトウェアのカーネル開発」(課題番号：18300010)による．

参考文献

[1] 齊藤圭，乃村能成，谷口秀夫，“ディレクトリに着目したバッファキャッシュ制御法の実現”，情報処理学会研究報告 2006-OS-101，Vol.2006，No.15，pp.69-76，2006.