

ファイルの特性とアクセスパターンに適応可能な メモリ管理手法

林 佳寛[†] 瀧本 栄二^{††} 毛利 公一^{†††} 大久保 英嗣^{†††}
[†]立命館大学理工学部 ^{††}株式会社 ATR 適応コミュニケーション研究所
^{†††}立命館大学情報理工学部

1 はじめに

近年, 計算機は, 低価格化が進み, 家庭, 企業, 学校などでさまざまな用途に用いられている. また, 計算機で扱うデータの量と種類も増加し, 外部記憶装置を備える計算機が一般的である. 外部記憶装置は, 計算機で用いる CPU や主記憶に比べて低速である. このため, CPU は, 処理に必要なデータの外部記憶装置からの読出しを待つ. この問題を解決するために, オペレーティングシステム (以下, OS と記す) は, ディスクキャッシュを実装し, 効率的な外部記憶装置の入出力を行っている. しかし, Linux や Windows といった既存の OS では, さまざまな特徴をもつ外部記憶装置への入出力を 1 つの固定された機構で処理する. このため, これらの OS は, それぞれの入出力に適した機構を提供することができないという問題がある. この問題を解決するために, 我々は, おおのの入出力のもつ特徴に適応するディスクキャッシュ機構を開発している. 適応的ディスクキャッシュ機構は, 外部記憶装置から読み出すファイルのもつ情報とファイルを用いるユーザプロセスのアクセスパターンを管理することで入出力の特徴を判断し, それに応じて処理を変える. 本稿では, ファイルの特性とアクセスパターンに適応するディスクキャッシュ機構の概要と実装について述べる.

2 適応的ディスクキャッシュ機構の概要

適応的ディスクキャッシュ機構は, 図 1 に示すように, ファイルシステムとディスクキャッシュマネージャで構成される.

以下, 本章では, 各構成要素の必要性について述べた後, アクセスポリシーとそれを用いるディスクキャッシュの管理について述べる.

2.1 ファイルシステムとディスクキャッシュマネージャの必要性

適応的ディスクキャッシュ機構を用いる例として, ハードディスク上に記録されている動画ファイルを再生する場合を考える. 30 秒の動画を繰り返し再生する場合と,

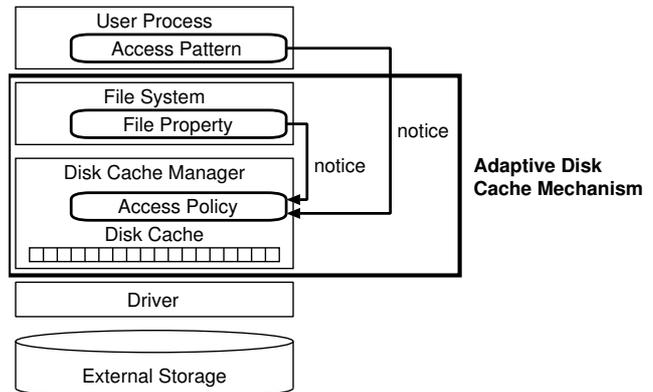


図 1 適応的ディスクキャッシュ機構

30 分の動画を繰り返し再生する場合を比較して考える. 30 秒の動画ファイルでは, 30 秒ごとに動画の同じ部分へのアクセスがあると考えられることができる. したがって, 2 回目以降の動画の再生時には, ハードディスクからデータを読み出さず, OS が保持しているディスクキャッシュを用いて再生することができる. 後者では, 30 分経たないとディスクキャッシュへのアクセスはないと考えられる. したがって, ディスクキャッシュにデータを保持せず, 他の用途へディスクキャッシュ用いるの方が良いと考えられる.

適応的なディスクキャッシュの管理を行うためには, ファイルが持つ情報を把握する必要がある. この例において, ディスクキャッシュマネージャは, ファイルの再生時間という情報を用いて, ディスクキャッシュの扱いを変える. このような情報をファイルの特性と呼ぶ. しかし, 必要な動画ファイルをすべてユーザ領域のメモリ上に読み出した場合, 2 度目以降の動画の再生時に, ディスク上に保存されたファイルの読出しは発生しない. このことから, ファイルを扱うユーザプロセスが, どのようにファイルを使用するかという情報についても把握する必要がある. この情報をアクセスパターンと呼ぶ. ファイルの特性とアクセスパターンをまとめてアクセスポリシーと呼ぶ.

ディスクキャッシュマネージャは, アクセスポリシーの内容に応じて, ディスクキャッシュの扱いを変える仕組みを備える. これにより, 適応的ディスクキャッシュ機構は, 外部記憶装置の入出力を特徴づけるアクセスポリシーに応じてディスクキャッシュの扱いを変えることが可能となる.

An adaptive memory management for file property and file access pattern

Yoshihiro Hayashi[†], Eiji Takimoto^{††}, Koichi Mouri^{†††}, and Eiji Okubo^{†††}

[†]Faculty of Science and Engineering, Ritsumeikan University

^{††}Advanced Telecommunications Research Institute International

^{†††}College of Information Science and Engineering, Ritsumeikan University

2.2 ファイルの特性

適応的ディスクキャッシュ機構で用いるファイルの特性には、以下のものがある。適応的ディスクキャッシュ機構は、これらの情報を用いてファイルの読出しに必要なキャッシュの量を計算する。

ファイルの種類 プログラム、テキスト、画像、動画などの種類

ファイルサイズ ファイルの内容をすべてメモリ上に展開したときに必要になるメモリの量

ビットレート 動画や音声などの時間的な連続性をもつデータを用いるために、単位時間あたりに読み出す必要があるデータの量

再生時間 動画や音声などの時間的な連続性をもつデータを再生するために必要な時間

2.3 アクセスパターン

適応的ディスクキャッシュ機構を用いることで、ディスクキャッシュの効率化を図ることが可能なアクセスパターンには、次にの 2 つがある。1 つは、ファイルを読み出すユーザプロセスが 2 度目の読み出しを行わないことがあらかじめ分かっている場合が考えられる。これは、プログラマがプログラム作成時に把握することができる。これをアクセスポリシとして把握することで、無駄なディスクキャッシュを減らすことが可能となる。もう 1 つは、一度読み出したデータを次に読み出すまでの時間も把握することが可能な場合がある。これもアクセスポリシとして用いることができる。

2.4 アクセスポリシに基づいた適応的ディスクキャッシュの管理

ディスクキャッシュマネージャは、与えられたファイルの特性とアクセスパターンをアクセスポリシとして管理する。また、アクセスポリシに基づき、ディスクキャッシュに重み付けを行う。ディスクキャッシュの管理には、LRU(Least Recently Used) が広く用いられている。ディスクキャッシュマネージャは、解放すべきディスクキャッシュの選択の際に、LRU にディスクキャッシュの重みを考慮した重み付き LRU を用いる。重み付き LRU において、ディスクキャッシュは、重みが大きいとき、LRU において選択されにくくなる。また、重みが小さい時、選択されやすくなる。特定のディスクキャッシュに関して、近い将来にアクセスが発生するとアクセスポリシから読みとれるとき、重みを大きくする。また、長い間アクセスが起こらないと読みとれるとき、重みを小さくする。

3 AG における適応的ディスクキャッシュ機構

現在、我々が開発しているエージェント指向 OS AG[1] に本機構を実装している。AG は、環境の動的な変化に適応可能とすることを目的とした OS である。AG は、マイクロカーネル方式に基づいて設計されている。カーネルとしての機能は、エージェントと呼ぶシステムサーバが提供する。エージェントは、環境の動的な変化に適応するために、熟考型スレッドと即応型スレッドをもつ。即応型スレッドは、OS としての機能を要求に応じて処

理する。熟考型スレッドは、環境を観測し、環境に適応するように即応型スレッドの機能の変更や特定のイベントに依存しない非同期の処理を行う。エージェントは、熟考型スレッドが、即応型スレッドの機能を変更することで、カーネルとしての機能を自律的に変更することが可能となる。これをリフレクションと呼ぶ。

適応的ディスクキャッシュ機構は、File System Agent (以下、FSA と記す) と Disk Cache Agent (以下、DCA と記す) で構成される。各構成要素をエージェントとして実装し、リフレクションを用いることで、環境に適応した処理を行う。

3.1 FSA

FSA は、ファイルの管理と監視を行う。FSA の熟考型スレッドは、読み出したファイルの特性を DCA へ通知する。FSA の即応型スレッドは、一般的なファイルシステムと同様に、open, read, write, close などの処理を行う。また、ファイルと外部記憶装置の LBA (Logical Block Address) との対応づけを行い、DCA に対して読出しの要求を発行する。

3.2 DCA

DCA は、ディスクキャッシュの管理とアクセスポリシの管理を行う。即応型スレッドは、FSA から読出し要求を受けたデータが、既にディスクキャッシュ内に存在するか否かを確認し、存在するならばディスクキャッシュ内のデータを FSA へ返す。ディスクキャッシュ内に必要なデータが存在しなければ、外部記憶装置へ読み出しの要求を発行し、データを受け取る。受け取ったデータを保存する領域を確保し、データをキャッシュする。また、FSA からファイルの特性、ユーザプロセスからアクセスパターンを受け取り、アクセスポリシとして管理する。DCA の熟考型スレッドは、CPU に余裕があるときに、アクセスポリシを用いてディスクキャッシュに重み付けを行った後、ディスクキャッシュの解放を行う。解放するディスクキャッシュを決定するアルゴリズムには、ディスクキャッシュの重みを考慮した LRU を用いる。

4 おわりに

本稿では、ファイルの特性とアクセスパターンを把握することで、これらに適応したディスクキャッシュの管理が可能となることについて述べた。現在、本機構の有効性を確認するために、Linux 上にプロトタイプを実装中である。また、プロトタイプの実装にあたり、アクセスポリシの記述法とアクセスポリシから入出力の特徴を抽出しディスクキャッシュに重みづけを行う手法を検討中である。

参考文献

- [1] 瀧本栄二, 滝沢泰久, 毛利公一, 大久保英嗣: “エージェント指向オペレーティングシステム AG におけるリフレクティブエージェントの実現手法,” 情報処理学会研究報告 2003-OS-92, Vol. 2003, No. 19, pp. 1-6 (2003).