入出力仮想化による計算機高速化機能の開発とその分析・評価†

櫻庭健年†† 吉澤康文††
新井利明†† 藤田不二男†††

計算機の高性能化に主記憶をファイルのキャッシュとする方法は有効である．本研究では，大形計算機におけるファイル入出力の分析を行い，キャッシュ容量と入出力削減効果の分析を行った．この結果，アクセス総量の5%の主記憶をキャッシュとして使用することで67%の入出力削減が可能であることがわかった．そこで，主記憶と拡張記憶の記憶階層をキャッシュとして構成する仮想ファイルを提案する．仮想ファイルを用いると，プログラムからの入出力を論理的な入出力とし，物理的な入出力と切り離すことになるので入出力仮想化と呼んでいる．仮想ファイルを既存の複数のファイルシステムに適用するには，個々のファイルシステムの要求や特性に合った機能拡張を必要とする．この要求や特性を満たすために，入出力仮想化の基本部分を抽出した汎用入出力仮想化機能 HAF (High performance Access Facility) を設計した．HAF の効果を確認するために，2種類のファイルシステムに適用した．その結果，TSS (Time Sharing System) では応答時間を20%短縮し，入出力を多用するバッチ処理では処理時間を約1/2に短縮する効果があった．

1. はじめに

計算機システムでは，入出力処理が性能上の隘路となっている場合がある．理由は，主記憶へのアクセス時間に比べ，機械的な動作を必要とする磁気ディスクへのアクセスは数万〜数十万倍と遅いことに起因する．たとえば，外部記憶への入出力を多用するようなバッチ処理では，実行時間の大半が入出力時間に占められるばかりでなく，入出力の起動や入出力完了割り込み処理などのオペレーティングシステム (OS) のオーバヘッドが大きい．一方，オンライン処理や TSS の場合には，入出力時間が応答時間に占める割合が大きい場合があり，高トラフィック状態では，入出力の待ち要因も重なり，入出力が応答時間の主たる劣化要因となることがある．そこで，OS がなんらかの手段によって物理的な入出力操作を大幅に削減することができれば，バッチ処理や対話処理の性能を向上させることが可能となる．

入出力を削減し，入出力に要していた時間を削減する研究は二つに分類できる．一つは主記憶を利用するものであり[1)~5)]，「仮想記憶常駐」と呼ばれるものであり他方は特殊な入出力装置を使用するものである[6)]．

「仮想記憶常駐」法は，外部記憶上のファイルをあらかじめ仮想記憶に読み込んでおき，入出力動作をページングで行う方法である．つまり，ファイル入出力要求時に該当のデータが主記憶上に存在すれば，主記憶間のデータ転送ですむ．主記憶に該当データがない場合には，ページング用のファイルへの入出力となる．仮想記憶常駐方式は広く利用されているが，(1)仮想記憶のページリプレースメント方式との関係，ならびに(2)仮想記憶上に展開されたファイル内容とファイルシステム内のデータ一致性の保証，などに問題がある．

上記(1)の問題点は，リプレースメントアルゴリズムとして多くの OS が採用している LRU 法やワーキングセット法が必ずしもファイルのデータ参照モデルに合致していない点にある．このため，大幅な入出力の削減を目標とするには OS の機能をそのまま使用するのではなくファイルへのデータ参照モデルに基づくアルゴリズムを開発する必要がある．

(2)の問題点は，システムダウン時に予想されるデータの不一致の問題である．ファイルを仮想記憶上に配置し，アプリケーションプログラムが内容を更新しただけでは外部記憶上のファイル内容は更新されていない．この状態でシステムダウンが発生すると，アプリケーションの処理結果とファイル内容との間で矛盾が生ずる．

入出力時間を短縮する他の方法は半導体ディスク，キャッシュ付き磁気ディスク，等の高速な記憶装置の

† Performance Improvement for Computer Systems by Virtualized Input/Output and Its Analysis by TAKETOSHI SAKURABA, YASUFUMI YOSHIZAWA, TOSHIAKI ARAI (Systems Development Laboratory, Hitachi, Ltd.) and FUJIO FUJITA (Software Development Center, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所
††† (株)日立製作所ソフトウェア開発本部

使用である．特に，キャッシュ付き磁気ディスクは入出力時間を短縮してシステムの性能向上を図る点では優れている．ここでの問題は，ファイル内のどの部分をキャッシュ上に保持しておくかを決定するアルゴリズムにある．計算機の運用上，高速な記憶媒体に配置するファイルはシステムの初期設定時に決定しておくほうが都合がよく，多くの場合固定的になってしまう．つまり，負荷の変動する実環境では最適なデータ配置を動的に行っていくことが困難になるという問題が生じる．また，高速な入出力装置を使用して入出力時間を短縮しても，入出力操作に伴う CPU オーバヘッドの削減はできない．

そこで，本研究の目的は，(a)ファイルに対する入出力処理時間削減のため，主記憶ならびに拡張記憶を利用して入出力回数および OS の CPU オーバヘッドを削減すること，(b)データアクセスのパターンに合わせた動的なページリプレースメントを OS のアルゴリズムと独立に行えるようにすること，(c)仮想記憶上のファイルとファイルシステムの内容との一致性を保証すること，(d)既存のアプリケーションとのインタフェースを変更することなく本論文で提案する仮想ファイルを利用できること，等にある．仮想ファイルを用いると，プログラムからの入出力を論理的な入出力とし，物理的な入出力と切り離すことになるので入出力仮想化と呼んでいる．効果として，対話システムの応答時間を改善しバッチ処理のスループットの向上を図ることが期待できる．

2. 入出力動作の分析と拡張入出力の効果予測

2.1 情報収集と解析方法

正確な入出力情報の収集は一般的に容易ではない．入出力仮想化の研究の効果を見極めかつ制御方式の検討を行うために基礎的な情報の収集と分析をするツール（FOCAS/M と呼んでいる）を開発した．FOCAS/M は，OS が磁気ディスクに入出力を実行する時点でチャネルへの I/O コマンドをすべて解析し，結果をトレース情報として蓄積するものである．収集する主な情報は，ファイルのディスク内配置情報，装置アドレスと入出力ブロックのディスク内アドレス，転送長，書込み読込みの区別などである．FOCAS/M の収集した情報は FOCAS/A と呼ぶ解析プログラムによってファイル単位にアクセス特性が分析される．

TSS とバッチ処理主体の計算センタにおける入出力情報を FOCAS によって情報収集し解析を行った．さらに，オンラインデータベースシステムでの入出力を分析するために，オンライン稼働時に取得するジャーナル情報を分析した．FOCAS と異なり，ここでは特定のデーターベースに対するアクセスを分析することとした．以下，その分析結果を述べる．

2.2 TSS，バッチ処理の入出力分析

測定の対象とした計算機システムはソフトウェアの研究開発に使用されているものであり，大形計算機 HITAC M 180 と汎用オペレーティングシステム VOS 3 で構成されている．TSS では主に FORTRAN プログラムの開発とデバッグが行われ，バッチは開発済みプログラムの実行である．測定時間は負荷の最も高い時間帯を選び，約 2 時間 40 分である．

ファイルは使用目的別に分類すると，OS がシステム管理の目的で使用しているシステムファイルと，エンドユーザの使用するユーザファイルになる．ここでは，主に OS の管理用ファイルの入出力削減の可能性を知るために，システムファイルを 5 種類に分類し分析した．**図 1** に入出力の実行回数と各ファイルに

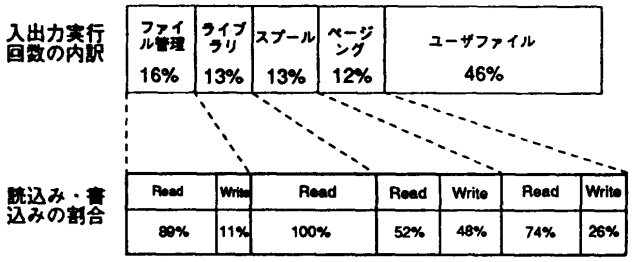


図 1 計算センタにおける入出力実行回数の分析
Fig. 1 An analysis of I/O executions in a laboratory site.

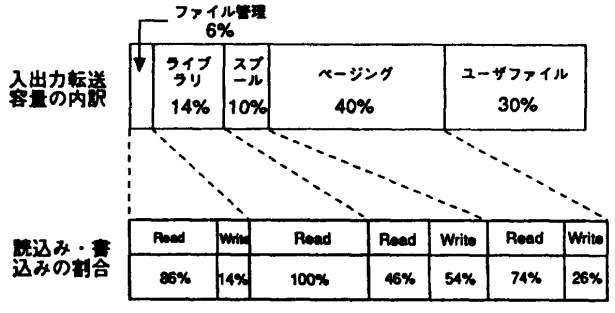


図 2 計算センタにおける入出力転送容量の分析
Fig. 2 An analysis of I/O transmission in a laboratory site.

対する読込みと書込みの割合を示した．また，図2には，データ転送容量の分析を行った結果を示す．

（1） ファイル管理用ファイル

ファイル管理用ファイル群はシステム内のすべてのファイルの位置，セキュリティ情報，等を管理するファイルである．総入出力回数の16%が集中しているが入出力一回あたりの転送量は 0.96 KB であり，分類したファイルの中で最も小さい．転送容量は全体の6%であった．入出力中の読込み比率（読込み率）は89%であり，ほとんどが読込みである．

（2） プログラムライブラリ

システムが提供するプログラムライブラリ群で言語ライブラリを含む．入出力はプログラムローディングが主であり，全体の13%の入出力回数を占める．入出力あたりの転送量は 3.06 KB/回で，ページングを除くすべての入出力の中で最大である．転送容量は全体の14%を占めている．すべてが読込み処理である．

（3） スプールファイル

ジョブ入出力情報の格納されているファイルである．入出力回数は全体の 13% を占めている．読込み率は 52% で，ほぼ半分である．転送容量は 10% であった．

（4） ページングファイル

仮想記憶を管理するためのファイルであり，主記憶の内容を保持する．入出力回数は全体の 12% を占めている．ページサイズは 4 KB であり，転送容量は 40% を占めている．ページングは主記憶を増設することで削減することができるため，この部分は入出力仮想化の対象外である．

（5） ユーザファイル

ソースプログラム，オブジェクトプログラム，一時的なデータを格納するファイルなどである．全入出力実行回数は46%を占めるが転送容量は30%である．

（6） システムファイルとユーザファイル

FOCAS/M ではファイルのブロックアドレスを区別しているのでディスクを参照したメモリ領域の大きさをファイル単位で計算することが可能である．この分析から，ユーザファイルへの全アクセス容量は全体の 90% 以上を占めていた．アクセス回数は 46% と多いが同一のブロックを何度もアクセスすることは少ない．逆に，システムファイルのアクセスは繰り返し行われている，という特性を得た．

2.3 TSS，バッチ処理における入出力仮想化の効果予測

FOCAS/M にて収集した情報を基に入出力仮想化機能の効果をシミュレーションにて予測する．つまり，アクセスの生じたレコードを主記憶に保持し，再参照の発生は入出力が省略できる（ヒットと呼ぶ）とする．主記憶は 4 KB 単位のページとし，レコードが 4 KB に満たない部分はパディングされて格納されるものとする．また，入出力仮想化に使用する主記憶（キャッシュ）容量とヒット率との関係を求めるが，この時のリプレースメントアルゴリズムは LRU とする．

図3に入出力仮想化機能をシステムファイルに適用した場合のシミュレーション結果を示す．この結果から，アクセス総量の 1% のキャッシュを用いることで約 50% の入出力動作が削減できること，5% のキャッシュ容量で 67% の入出力動作が削減できることが判明した．このことから，システムファイルは再参照レコードが多くあることが分かり，入出力仮想化機能は比較的小容量の主記憶を利用することで大幅な入出力削減を可能にできることが予測できた．

2.4 データベースにおける入出力仮想化の効果予測

データベースに対するアクセス特性を調べるために，生産ラインを制御する産業系のオンラインシステムで取得しているファイルジャーナルを分析した．ファイルジャーナルには，データベースに対するアクセスレコード，レコードの長さ，読込み，書込みの区別，などの情報が含まれている．分析の対象としたデータベースに対し，アクセスのあった相異なるレ

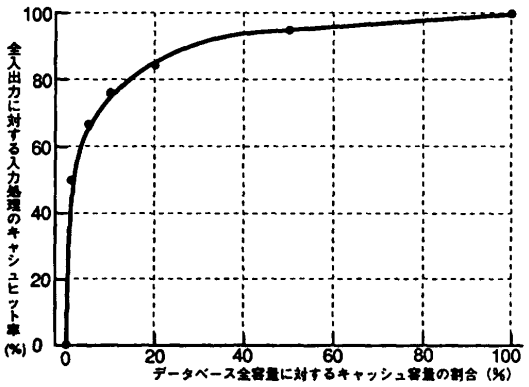


図3 TSS およびバッチ処理における入出力仮想化のシミュレーションによる効果予測

Fig. 3 Estimation of the effect of I/O virtualization in TSS and batch processing system.

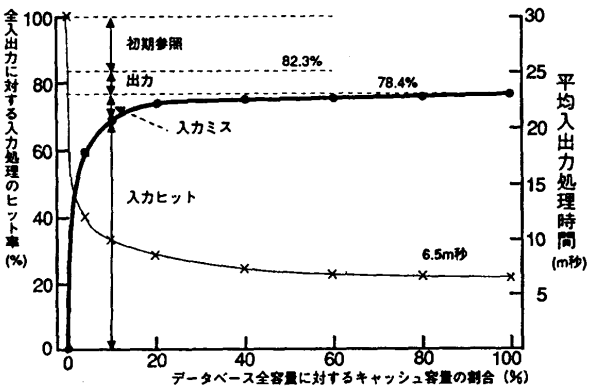


図 4 オンラインデータベースシステムにおける入出力仮想化の効果予測
Fig. 4 Estimation of the effect of I/O virtualization in an on-line database system.

コードの総容量は 190 MB である.

オンラインデータベースシステムは信頼性に対する要求が強いために，仮想入出力機能ではファイルの更新時に磁気ディスクに更新レコードを書き込む必要がある．そこで，シミュレーションでは書込み時にはたとえ主記憶上に目的のレコードが存在してもヒットせず，必ずディスクへの書込みが発生するものとする．また，初期読込みをミスと計算することにする．したがって，拡張入出力のキャッシュが無限にあった場合でもヒット率が100% とはならない．**図4**にシミュレーション結果を示す．

全アクセスに対してレコードの更新は 18% であった．また，初期入力は全アクセスの4%であった．したがって，このシミュレーションではキャッシュが無限にあってもヒット率は 78% が限界である．図4から，アクセス容量の5%のキャッシュによって，約60% の入出力操作が削減されることがわかった．

図4には，キャッシュへのヒット率の変化と入出力に要する時間の関係も示した．ここでは磁気ディスクへの平均アクセス時間が 30 ミリ秒とすると，読込みがすべてヒットした時の平均入出力時間は6.5ミリ秒になる．ヒット率は比較的小容量のキャッシュで急激に高くなるので，それに伴い平均ディスクアクセス時間は急激に短くなることが予測できる．

以上，TSS およびバッチ処理のシステム，およびオンラインデータベースシステムにおける仮想入出力機能の入出力削減効果を統計的ならびにシミュレーションによる分析から予測した．いずれの場合も，比較的少量の主メモリをキャッシュとして使用することで，大幅な入出力の削減が可能であることが判明し，入出力仮想化機能の有効性を示す結果である．そこで，入出力仮想化を実現するにあたり，汎用性，既存ユーザプログラムの移行などを考慮した設計について述べる．

3. 汎用入出力仮想化機能：HAF

3.1 HAF の位置づけ

入出力仮想化を実現する具体的な格納領域を仮想ファイルと呼び主記憶と拡張記憶で構成する．仮想ファイルは OS の基本機能として用意し，既存プログラムとのインタフェースを変えることなく実現する必要がある．このためには，ファイルアクセス法（ファイルシステムと呼ぶ）が仮想ファイルを利用する構造が望ましい．

一般に，OS は複数のファイルシステムを備えており，それぞれが独立したソフトウェアになっている．ファイルシステムはファイルへアクセスの目的別に作成されている．例えば，SAM (Sequential Access Method) ファイルではデータを連続的にアクセスする目的で開発され，DAM (Direct Access Mothod) ファイルでは任意のレコードをアクセスするように作られている．このため，いずれのファイルシステムにも適用可能な汎用的な仮想ファイル構築機能が必要となる．

以上の考えから，ファイルシステムが仮想ファイルを利用する際の基本機能を抽出した．また，ファイルシステム固有の機能をカスタマイズする機能を提供することとした．この共通機能を汎用入出力仮想化機能：High performance Access Facility (HAF) と名付けた[7]．

3.2 HAF のオブジェクトと操作

上記の機能を実現するため，HAF では四つの抽象化したデータの型を導入し，各々に対して操作を定義した．それらの型をオブジェクトと呼ぶ．HAF のオブジェクトおよびその操作を**図5**に示す．また，**表1**にオブジェクトへの操作の一覧を示す．

（1） データ格納エリア：AREA

HAFがデータを格納する仮想記憶上の領域である．領域を複数個作成することができ，独立に管理される各々をサブエリアと呼ぶ．アクセス性能を保証するために，領域を実記憶または拡張記憶にマッピングする機能を提供する．サブエリアの作成，削除，および容量の変更を可能とする．作成時に，エリア中のリプレースメントアルゴリズムを指定することができる．

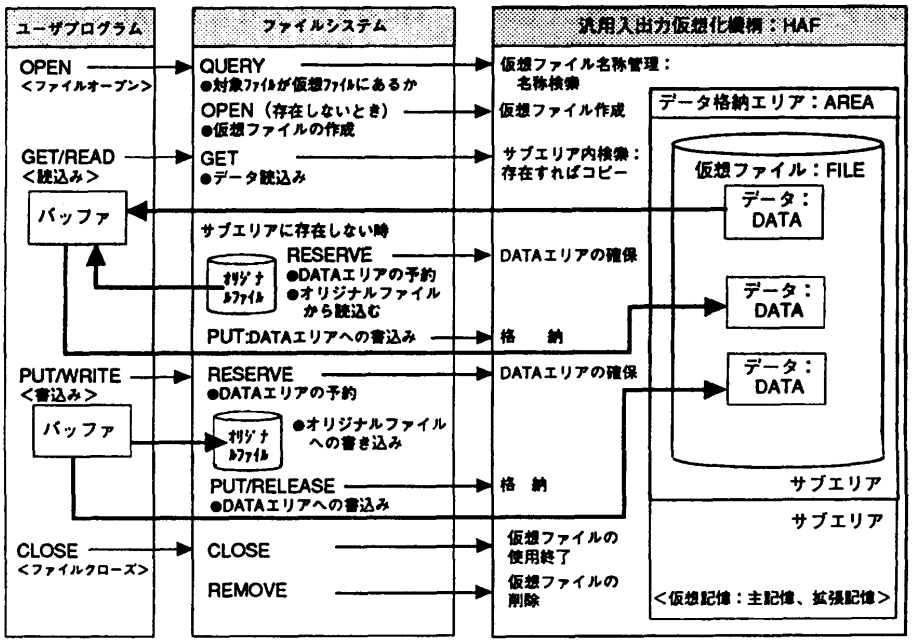


図 5 HAF をファイルシステムが利用する際の手順
Fig. 5 Procedure for using HAF in file system.

表 1 ファイルシステムに提供される汎用仮想入出力機能（HAF）
Table 1 I/O virtualizing primitives provided for file system.

項番	オブジェクト	操作名	機　　能
1	データ格納エリア：AREA	DEFINE	データ格納エリアの作成
2		DELETE	データ格納エリアの削除
3		EXPAND	データ格納エリアの領域拡張
4		REDUCE	データ格納エリアの領域縮小
5	仮想ファイル：FILE	OPEN	仮想ファイルの作成／既存仮想ファイルへの連結（再使用）
6		CLOSE	仮想ファイルの消去／使用終了
7		INACTIVATE	仮想ファイルの閉塞（消去／サービス停止）
8		ACTIVATE	仮想ファイルの閉塞解除（サービス再開）
9	データ：DATA	GET	データの入力（ミス時はステージング用書込み予約）
10		PUT	書込み予約済みデータのデータ格納領域への出力
11		RESERVE	出力データの書込み予約
12		RELEASE	データ書込み予約の解除
13		PURGE	データの閉塞（削除／アクセス禁止）
14	対象ファイル名：NAME	ADD	対象／非対象ファイル名称の登録
15		REMOVE	登録されている対象／非対象ファイル名称の削除
16		QUERY	対象ファイル名称の問い合わせ

（2） 仮想ファイル：FILE

仮想ファイルのデータは仮想ファイルに対して定めた一つのサブエリアに格納される．一つの仮想ファイル内ではデータ長は一定である．複数のタスク／ジョブから使用することができる．

一つのファイルを，複数のユーザが同時に使用している場合，HAF を適用しているタスクと使用していないタスクが同時に存在すると，該当ファイルの内容に矛盾が生じる．このような状況を回避するため，仮想ファイルのオープン機能では作成要求された仮想ファイルが既に作成済か否かにより，仮想ファイルを新たに作成する／再使用する／削除して新たに作成し直すのいずれかが選択できる．

図5に示すように，ユーザプログラムの中でファイルの OPEN がファイルシステムに要求されると，ファイルシステムは，まず，該当ファイルがすでに他のジョブで仮想ファイルとして OPEN されているかを調べる (QUERY)．もし，仮想ファイルとして存在しない時には，HAF に対して OPEN 要求を出し仮想ファイルを作成する．

（3） データ：DATA

仮想ファイル内のデータは識別子により区別される．識別子はファイル名称とファイル内のレコード番号の組み合わせである．データ長はファイル名称ごとに固定させる．データの取り出し要求 (GET) では，AREA 内の探索がなされる．データが存在した場合（ヒット）には指定のバッファにデータを転送し，存在しない場合にはミスとする．

AREA への書込み（ファイルの更新）は逐次処理を行わないと AREA 上のデータとオリジナルファイル（磁気ディスク上のファイル）との不一致が起きる．例えば，タスクAとタスクBが相前後して AREA 上のデータとオリジナルファイルを更新する場合，タスクAがオリジナルファイルをまず初めに更新し，AREA のデータはタスクBが先に更新した場合には，AREA 上のデータとオリジナルファイルとが不一致となる．

このために，更新を行う時は AREA 上のデータを更新することを必ず HAF に予約 (RESERVE) を行う規約としている．図5に示すように，RESERVE は上記の DATA オブジェクトへ書込みを行う時は必ず行う．

HAF を使用するプログラムは上記(1)によって格納領域のサイズを最初に定義するのでリプレースメントを行うこともできる．しかし，HAF への格納要求時に空きがなくなると HAF の標準的なリプレースメント機能によって空きが作成される．

（4） 対象ファイル名：NAME

HAF 適用対象ファイル名称の登録，削除，検索を行う．

図5に示すように，HAF はファイルアクセス法から利用され，データ格納エリアの作成 (DEFINE)，仮想ファイルの作成 (OPEN)，仮想ファイルへのデータの蓄積 (PUT)，格納エリアからの読込み (GET)，更新データの書込み (RESERVE, PUT, RELEASE)，仮想ファイルの使用完了 (CLOSE, DELETE) など一連の操作が行われる．

3.3 信頼性の確保の機能

HAF は揮発性メモリである主記憶および拡張記憶をキャッシュとして使用している．このため，システムダウンに備えてファイルの内容を保証する工夫が必要となる．内容の保証は，ファイルの更新時に要求される．つまり，HAF 領域にデータを書き込む際に，オリジナルファイルへも内容を書き出す．この機能は選択的であり，性能向上を主目的としたファイルでは不要であり，ある時点でまとめてオリジナルファイルに書き出す機能も備えている．それぞれ，ライトスルー型とライトアフタ型制御と呼ぶ．

（1） ライトスルー制御

システムの障害発生時にも最新のデータがオリジナルファイルに残るため，信頼性の点で優れるが，出力操作は削減されないので，更新が少ないファイルへの適用に有効．

（2） ライトアフタ方式

HAF 領域が飽和するとデータ格納の要求元に割り込みを起こし，HAF 領域の開放を要求する．この際，格納されているレコードの中で最も長時間未参照のレコードを提示し，LRU 法によるプレースメントを可能とする．更新されたデータがオリジナルファイルに書き込まれるまでに時間差があるため，障害発生時には最新のデータが失われる可能性がある．

3.4 高性能化保障とリプレースメント方式

HAF は主記憶と拡張記憶をキャッシュとして利用するが，計算機の運用上，HAF が無制限に主記憶を利用できない．そこで，限られた主記憶，拡張記憶を効率的に活用するために以下の機能を用意する．

（1） HAF エリアの主記憶常駐化

HAF エリアは仮想記憶上にあり，ページングの対

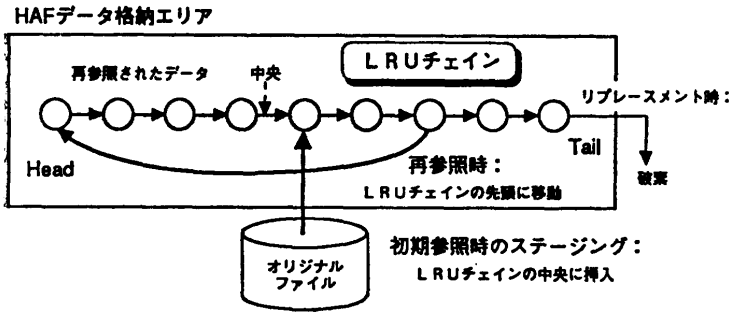


図 6 HAF で標準として採用する2階層 LRU 方式
Fig. 6 2-Layered LRU algorithm adopted by HAF.

象である．キャッシュ領域がページアウトされてしまうと，高速アクセスの保障がなくなる．そのため，HAF 領域をページアウトの対象外にする機能を提供する．

一般に，HAF エリアは大容量となる可能性があるので，拡張記憶が使用できるシステムでは，HAF エリアのページング先を拡張記憶と指定し，主記憶の負荷を緩和する．

（2） 2階層 LRU 管理

HAF エリアに空き領域が不足したときの管理は，ファイルシステムが独自のリプレースメントを表1に示した機能を用いて行う場合と HAF の標準機能とがある．HAF の標準リプレースメント法は，プログラムのメモリアクセスではなくファイルのアクセスに適用することを考慮し，その変形である2階層 LRU 管理を採用する．

2階層 LRU 管理のデータアクセスモデルは「再参照のあったデータはまだ再参照されていないデータに較べて将来の再参照可能性が高い」とし，両者を区別してそれぞれ LRU 管理する．具体的な実現方法を図6に示す．つまり，参照があったデータが HAF データ格納領域にない場合には，オリジナルファイルよりステージングを行うが，この時，該当データを LRU チェインの中央の位置に入れる．このようにすると，中央より先頭側は再参照されたデータが存在し，逆に，中央より後ろ側は再参照のないデータや参照されてから時間が経過したデータが多くなる．リプレースメントはこの LRU チェインの最後尾から行われる．このアルゴリズムを2階層 LRU 管理と呼ぶ．

本方式によると，一度だけアクセスされて再参照のないデータ（例えば，逐次ファイルなど）を早期に HAF エリアから追い出すことができる．

2階層 LRU 管理の効果を明らかにするため，2章で分析の対象としたオンラインデータベースシステムの実測データを基にシミュレーション評価を行った．その結果を表2に示す．

2階層 LRU 管理は，HAF エリアが少ない場合（メモリ量が総アクセス量の 20% 以内）に効果があり，多い場合には（30〜50%）逆にヒット率が若干低下している．このことは，ファイルが再参照される場合には2種類のパタンが存在することを示している．すなわち，短い参照間隔で頻繁に再参照されるパタンと，長い参照間隔で数回程度再参照されるパタンの2種類である．そのため，メモリ量が少なく LRU スタックが短い場合には，再参照の参照のあったデータを優先して残すことにより前者のパタンの再参照のヒット率が向上する．反対に，メモリ量が多い場合には，後者の再参照を優遇しすぎるため返ってヒット率が低下する．しかし，HAF エリアが多い場合のヒット率の低下の度合いは HAF エリアが少ない場合の向上の度合いに比べて小さいため，全体としては2階層 LRU 方式は効果があるといえる．

以上のことから，2階層 LRU 方式はメモリ量が少なく，全体のヒット率が低い場合に有効である．この評価は，ランダムアクセスのあるデータベースを対象としているが，連続的なデータ参照モデルではより効果が高いと期待できる．

表 2 2階層 LRU の効果
Table 2 Hit ratio improvement by 2-layered LRU.

リプレースメント ＼ *メモリ比	ヒット率 (%)								
	4	10	20	30	40	50	60	80	100
LRU	59.8	67.6	72.6	75.3	76.8	77.6	78.1	78.4	78.4
2階層 LRU	61.0	68.0	72.7	75.1	76.5	77.5	78.1	78.4	78.4

* メモリ比＝メモリ容量／総アクセス量 (%)

4. 入出力仮想化機能の実測評価

4.1 HAF 適用のファイルシステム

仮想ファイルの効果を確認するために，大形計算機用 OS のファイルシステムに HAF を開発し[8),9)]，性能評価を行った．このために，2種類のファイルシステムに HAF を適用した．一つはデータベースに使用される仮想記憶アクセス法 (VSAM : Virtual Storage Access Method) であり，もう一つのファイルシステムは，ソースライブラリやプログラムライブラリの管理・保守に利用される LIME (LIbrary Management and Editing system) である．

VSAM はデータベースに適用されるため信頼性が要求される．このため，HAF の管理方式としてライトスルー方式を適用した．出力処理の削減はできないが，第2章で説明したように，入力の削減が期待できる．適用対象となるファイルはシステム立ち上げ時のパラメータまたはジョブ制御文 (JCL) で指定することとした．

4.2 TSS での HAF の効果

TSS では，第2章で示した，エンドユーザが使用する「ユーザファイル」のほかに，ファイルの所在を管理したり，セキュリティ管理を行うためのファイル管理のための「システムファイル」にアクセスが集中する．システムファイルはファイルの作成，削除，使用，などで必ずアクセスされるため，これらのファイルの入出力を仮想化した場合には TSS コマンドの応答時間短縮に効果が期待できる．

システムファイルに HAF を適用した場合のコマンド応答時間短縮の効果を，TSS 端末シミュレータを用いて測定した結果を図7に示す．テキスト編集コマンド，ファイル削除コマンドなどコマンドの実行にシステムファイルへのアクセスが発生する処理では，応答時間が最大20%向上した．また，全体のスループットも4%向上した．

4.3 バッチ処理における HAF 適用効果

多くのバッチ処理，特に事務処理の分野では，ファイルの読込み，コピー，などが繰り返し行われる．そこで，事務処理などで頻繁に使用される，ファイル形式変換処理，およびファイルコピー処理の2種類のジョブを性能評価の対象とした．実測評価では，HAF 適用時と非適用時との処理経過時間ならびに入出力実行回数を比較項目とした．結果を表3に示す．この実測では，総容量が 17.6 MB のファイルコピーを行った．ファイル構成は区分データセット (PAM : Partitioned Access Method) であり，512 個のメンバからなる．

入出力回数の削減は，ファイル変換処理が 45% でありファイルコピーは 52% であった．経過時間は各々，49%，54% と短縮している．このように，バッチ処理では入出力の高速化が直接ジョブ性能向上に直結

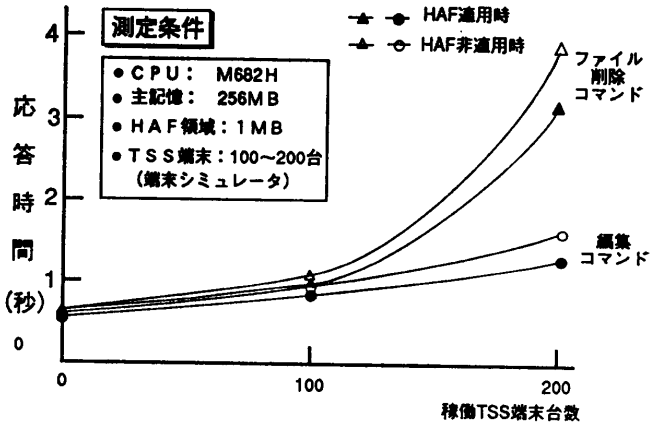


図 7 HAF 適用時の TSS 応答時間の改善効果
Fig. 7 TSS response time improvement by HAF.

表 3 バッチ処理へ HAF を適用したときの実測結果
Fig. 3 Performance improvement for batch processing by HAF.

		既存ファイルシステム	HAF をファイルシステムへ適用	削減効果
ファイル変換ジョブ	ジョブ処理時間 (秒)	236	121	49%
	入出力回数 (回)	16,289	8,896	45%
ファイルコピージョブ	ジョブ処理時間 (秒)	204	95	54%
	入出力回数 (回)	14,486	6,955	52%

注) 測定条件：512 メンバからなる区分データセットをコピー．総容量 17.6 MB

している．ここで，経過時間のほうが入出力の削減以上に短縮されているのは，入出力に伴うOSのCPUオーバヘッドが削減されたからである．各々のジョブで，CPUオーバヘッドが2〜4%削減された．

上記の結果から，TSS，バッチ処理のいずれにおいても入出力仮想化機能(HAF)を使用することにより，ユーザプログラムを修正することなく，計算機の性能向上が達成できることを確認できた．

5. おわりに

OSは，過去において，比較的小容量の主記憶を如何にして有効に利用し計算機の性能を向上させるかに腐心してきた．しかし，このような制約がなくなってくると，逆に，如何に大量の主記憶を利用してシステムとしての性能を上げるかが問題となってきた．ここに提案した入出力仮想化機能はその一つの方式である．

本研究では，現在の計算機システムがどのような入出力を実行しているのかを詳細に分析した．その目的は，主記憶や拡張記憶のような半導体記憶をキャッシュメモリとして利用し，どのぐらいの入出力が削減できるかの予測を行うことにあった．この結果，比較的小容量のキャッシュが入出力の削減に極めて有効であることが判明した．

現実の計算機システムでは，既存のユーザプログラムを継承することが前提になる．さらに，ファイルシステムは過去のOSの発展のなかで複数開発されており，各々に容易に適用できる汎用的な機能が要求されている．ファイルシステムはその用途によって，信頼性，データへのアクセスパターンなどまちまちである．この条件の下に，本研究では汎用性を持った入出力仮想化機能(HAF)を提案し，開発した．

最後に，HAFの効果を確認するために，2種類のアクセス法(VSAM, LIME)に適用してHAFの効果を実測した．TSSおよびバッチ環境における性能解析の結果，HAFの効果を確認した．

謝辞 本研究の推進にあたり(株)日立製作所システム開発研究所堂免信義所長ならびに久保隆重副所長のご指導に深謝する．また，ソフトウェア開発本部ならびに日本ソフトウェアエンジニアリングの方々からは，実用化における多くのご意見とご協力をいただいた．特に，ソフトウェア開発本部小平光彦部長には，本研究をVOS3/ASに適用する機会をいただき，多大のご支援をいただいた．最後に，入出力仮想化の基礎データ収集ツール(FOCAS)を開発していただいたシステム開発研究所吉岡正壱郎氏に感謝の意を表す．

参考文献

1) Stonebraker, M.: Virtual Memory Transaction Management, *ACM Operating Systems Review*, Vol. 18, No. 2, pp. 26-48 (1982).
2) Traiger, I. L.: Virtual Memory Management for Database Systems, *ACM Operating Systems Review*, Vol. 16, No. 4, pp. 26-48 (1982).
3) Robinson, J. T. and Devarakonda, M. V.: Data Cache Management Using Frequency-Based Replacement, *ACM Proceedings of SIGMETRICS*, pp. 134-142 (1990).
4) Effelsberg, W. and Haerder, T.: Principles of Database Buffer Management, *ACM Trans. Database Syst.*, Vol. 9, No. 4, pp. 560-595 (1984).
5) 平石：一般データセットの常駐化手法，第27回情報処理学会全国大会論文集，pp. 219-220 (1983).
6) Smith, A. J.: Disk Cache—Miss Ratio Analysis and Design Considerations, *ACM Trans. Computer Systems*, Vol. 3, No. 3, pp. 161-203 (1985).
7) 櫻庭，山本ほか：汎用入出力仮想化機能HAFの開発，情報処理学会OS研究会資料，90-OS-47 (1990).
8) 吉澤，米田ほか：大形システム用基本ソフトウェアの技術動向，日立評論，Vol. 73, No. 2, pp. 15-26 (1991).
9) 吉澤，新井ほか：OS VOS 3/ASの高性能・大容量化方式，日立評論，Vol. 73, No. 2, pp. 67-76 (1991).

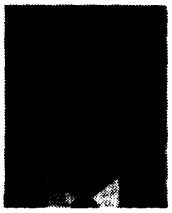
(平成4年1月23日受付)
(平成4年9月10日採録)

櫻庭 健年(正会員)

昭和33年生．昭和56年東北大学理学部数学科卒業．昭和58年同大学院修士課程修了．同年(株)日立製作所入社．以来，システム開発研究所にて，オペレーティングシステムの研究・開発に従事．平成2年度情報処理学会研究奨励賞．ソフトウェア科学会，日本数学会各会員．

吉澤 康文（正会員）

1944 年生．1967 年東京工業大学理工学部応用物理学科卒業．同年(株)日立製作所入社．中央研究所に勤務し，HITAC 5020/TSS の研究開発に従事．1973 年発足したシステム開発研究所に勤務．以後，仮想記憶，大規模TSS，オンラインシステム，など大形計算機のオペレーティングシステムの性能評価ならびに性能向上方式の研究開発に従事．これらの研究により，1981 年 3 月東京工業大学より工学博士を授与．また，オペレーティングシステムのテスト・デバッギングシステムの開発に従事．現在はワークステーション，サーバの高性能化ならびに高信頼化機能の研究をてがける．情報処理学会論文賞（昭和 47 年度）授与．IEEE Computer Society 会員．同研究所主管研究員．著書「オペレーティングシステムの実際」(昭晃堂)．



新井 利明（正会員）

昭和 29 年生．昭和 51 年早稲田大学理工学部電気工学科卒業．昭和 53 年同大学院理工学研究科修了．同年(株)日立製作所入社．システム開発研究所にて，大形計算機のオペレーティングシステムの性能評価および性能向上方式の研究・開発に従事．IEEE Computer Society, ACM 各会員．

藤田不二男（正会員）

1949 年生．1972 年岡山理科大学応用物理学科卒業．同年(株)日立製作所に入社．現在オペレーティングシステムの開発に従事．電子情報通信学会会員．