オペレーティング・システム　47－3
（１９９０．６．８）

汎用入出力仮想化機能ＨＡＦの開発

櫻庭健年[1]　山本伸夫[1]　野村雅光[3]　吉澤康文[1]　藤田不二男[2]　丹代美智夫[4]
[1]日立製作所 システム開発研究所　[2]同 ソフトウエア工場
[3]日立西部ソフトウエア　[4]日立ソフトウエアエンジニアリング

大型計算機の大容量メモリをシステムの高性能化に積極的に利用する技術が求められている。汎用入出力仮想化機能ＨＡＦをＯＳの基本機能と位置付けて開発し、大容量メモリを用いてシステムの多様な入出力を容易に仮想化可能とした。実稼働システムの入出力の分析によると、比較的少量のメモリを用いて大幅な入出力の削減が可能であることがわかった。汎用アクセス法ＶＳＡＭにＨＡＦを適用し、性能評価を行った。ＨＡＦの稼働環境はシステムパラメータおよびコマンドにより定義／変更が可能であり、稼働状態のモニタリング機能によりチューニングできる。メモリは拡張記憶と主記憶を用い、記憶階層制御は２階層ＬＲＵ方式を採用している。

General I/O Virtualizer HAF -Its Design and Application

Taketoshi SAKURABA[1],Nobuo YAMAMOTO[1],Masamitu NOMURA[3],
Yasufumi YOSHIZAWA[1],Fujio FUJITA[2],Michio TANDAI[4]
[1]Systems Development Laboratory,Hitachi,Ltd.
(1099 Ohzenji Asao-ku Kawasaki-shi,215 JAPAN)
[2]Software Works,Hitachi,Ltd.
[3]Hitachi Seibu Software,Ltd. [4]Hitachi Software Engineering,Ltd.
(5030 Totsuka-cho Totsuka-ku Yokohama-shi,244 JAPAN)

The strategy of applications of huge capacity memory is needed in mainframe computers. The general I/O virtualizer HAF has been developed as a primitive of mainframe operating system. According to an analysis of empirical data of dataset access, relatively small quantity of memory can reduce many I/Os. We adapted HAF to an access method widely used, and evaluated its performance. The operation environment of HAF are defined by system parameters, and modifiable by operator commands. HAF activity monitoring facility can be used for system tuning. HAF uses extended storage as well as main storage, and adopts the 2-level LRU algorithm.

1．はじめに

半導体技術の進歩により、安価、かつ大容量の主記憶が利用可能となってきた。一方、磁気ディスクのような２次記憶のアクセス時間には大きな改善はなく、主記憶とのアクセスギャップが計算機システムの性能上の問題点となっている。ディスク装置にキャッシュメモリを備え、アクセス時間を短縮することが行われているが（１）入出力に伴うＣＰＵオーバヘッドは解消されない（２）プログラムの動きに応じたキャッシュメモリのきめ細かい制御が難しい（３）各デバイスにキャッシュメモリを設けることによる分割損、などにより性能向上効果は限定される。そのため、それとは別に、ＣＰＵ側の制御による、大容量メモリを用いた入出力の仮想化、即ちディスク上のデータをあらかじめ主記憶に読み込んでおき、本来ならば必要であった物理的なディスクアクセスを論理的で高速なメモリアクセスに置き換えてアクセスギャップを解消する機能が有効である。

大型計算機では、主記憶と２次記憶の間の記憶階層として、拡張記憶と呼ばれる大容量で高速にＣＰＵから命令でアクセスできる新しい記憶媒体も利用可能となってきており、主記憶／拡張記憶の容量はＧＢ(ギガバイト：10^9byte) オーダに達している。大型計算機システムにおける既存の入出力仮想化機能には一時データセットを対象としたもの、データベース管理システムによるバッファリング機能などがあるが、適用できるデータセットが限られていること、機能／制御方式が大容量メモリに十分に対応していないことなどの点で上記のような大容量メモリを使いこなすには機能的に不十分である。

上記のような状況に鑑み、汎用大型計算機システムにおける大容量メモリを利用した入出力仮想化方式である拡張入出力制御機構（ＸＶＩＯ：EXtended Virtual Input/Output feature）を提案し、その有効性、及び制御方式について検討してきた。

研究のアプローチとして、始めに実稼働システムにおける入出力を分析し、入出力仮想化技術の適用分野の検討を行った。この結果、適用の有効性と適用容易性の観点から、プログラムローディングの仮想化方式をＸＶＩＯフェーズ１（ＸＰＬ：EXtended Program Loading）［１］として、また常時出力のある一般のデータセットアクセスの仮想化方式をフェーズ２として、分けて研究開発を行うことにした。また入出力履歴データを用いて適用効果の予測を行い、入出力仮想化機能の有効性：比較的小容量のメモリで大幅な入出力削減が可能であることを確認した。本稿では一般データセットに対する入出力の仮想化方式について述べる。

汎用性の観点から大型計算機システムでは、アクセス法／データセット編成が多様である。また信頼性や共用制御などに関して細かい制御が必要である。そのため一つの機能で全ての入出力の仮想化を図ることは難しい。一方、個々のコンポーネント／アプリケーション毎に入出力仮想化機能を開発するのは生産性の点で問題がある。

そこで我々は「汎用入出力仮想化機能ＨＡＦ（High performance Access Facility）をＯＳの基本機能」として位置付けて開発した。ＨＡＦでは、データアクセスの他、信頼性、及び実運用に関する要因も含めてモデル化し、基本機能として取り込んでいる。このため、入出力を行うＯＳのコンポーネントやアプリケーションは、ＨＡＦの提供する機能を利用することにより、入出力の仮想化を容易に実現することができる。さらにオンラインシステムなどで広範囲に使用されている汎用のアクセス法であるＶＳＡＭに適用し、その有効性を確認した。

大型計算機システムにおける入出力仮想化技術としては［２］がある。大容量のメモリに大量のデータを効率よく格納するため、データ格納専用の仮想空間を用いるなど、ワンレベルストレージを指向したアーキテクチャ上の革新が行われている。新しいデータセット編成とアクセスインタフェースを用意しており、新しいプログラムから利用できる。一方、ＨＡＦは汎用のインタフェースを直接利用できる他、既存のアクセス法であるＶＳＡＭに適用したことにより、ＶＳＡＭを利用しているアプリケーションプログラムを全く変更することなく、その入出力を仮想化することができる点に特徴がある。

以下では一般データセットのアクセス仮想化機能であるＨＡＦの設計思想、機能について説明する。またＨＡＦのＶＳＡＭ適用の効果について述べる。

2．実稼働システムにおける入出力の分析

入出力仮想化機能の研究開発の動機付けを明確にするために、実稼働システムにおける入出力の実態を分析した。対象とするシステムはＴＳＳ／バッチを主とするシステム、及びオンラインシステムのデータベースアクセスである。

2．1　ＴＳＳ／バッチシステムの分析

入出力仮想化制御のキーポイントは出力の取扱い方式である。出力の有無、出力データの保存の必要性の有無により制御方式が異なる。またＴＳＳ／バッチシステムではシステムデータセットの入出力仮想化は適用効果の波及範囲が広い。以上のような観点からデータセットを

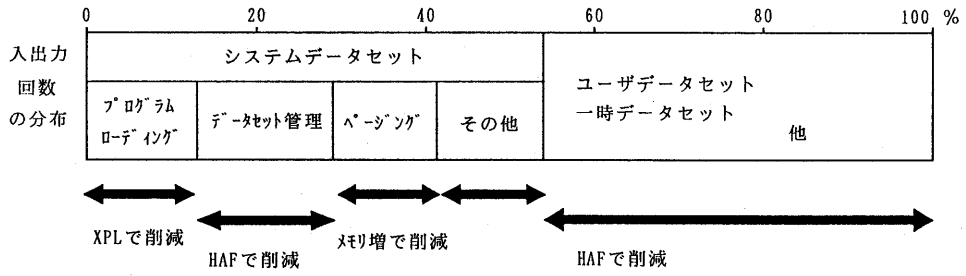

図１．ＴＳＳ／バッチシステムにおける入出力回数の分布の例

分類し、アクセス回数を調べたものが図１である。

（１）プログラムライブラリ

プログラムライブラリは読み出し専用であり、ページングを除くシステム全体の入出力の約１５％を占める。

（２）データセット管理

システムのデータセットの所在、及びデータセットアクセスのセキュリティの管理を行うカタログデータセット等は書き込みが常時発生し得るが、実際は読み出しが圧倒的に多く、ページングを除くシステム全体の入出力の２０％近くを占める。

（３）ページング

メモリを増設すれば解消できる。入出力仮想化機能の対象外である。

（４）ユーザデータセット

ユーザのソースプログラムを初めとする作業用データセットである。常時更新があり、内容を保存しておく必要がある。全体の約５０％を占める。

以上の分析から、プログラムライブラリ、カタログはアクセス回数が多く、全てのＴＳＳユーザが使用するため、その入出力仮想化は効果が高い事が期待される。

２．２　ＴＳＳ／バッチシステムにおける入出力仮想化の効果予測

実稼働システムの入出力履歴データを利用して入出力仮想化制御をシミュレートし、ヒット率等の適用効果を定量的に予想した。使用するデータには入力／出力の区別があることが必要である。

図２にＴＳＳ／バッチシステムにおけるシステムデータセット（カタログ、ライブラリ）の入出力仮想化効果を示す。図は全アクセス量に対するメモリ容量比を変えて、入力、ないし出力のヒット率がどのように変化するかを示している。アクセス量の５％のメモリで入出力の６０％以上がヒットしている。

２．３　データベースアクセスにおける入出力仮想化の効果予測

図３にオンラインシステムにおけるデータベースアクセスの入出力仮想化効果を示す。ライトスルー方式を適用した場合、実際に削減できる入出力は入力だけなので、ここでは入力時のヒットのみの値を示した。メモリが無限にあってもヒット率が１００％にならないのは、出力、及び初期読み込みがあるからである。アクセス量の５％のメモリで約６０％の入力が削減されている。

いずれの場合も、比較的少量のメモリで、大幅な入出力削減が可能であることがわかる。このことは一般データセットの入出力仮想化機能の有効性と必要性を強く支持するものである。

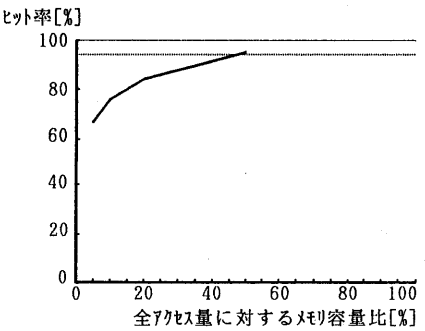

図２．ｼｽﾃﾑﾃﾞｰﾀｾｯﾄｱｸｾｽ仮想化の効果

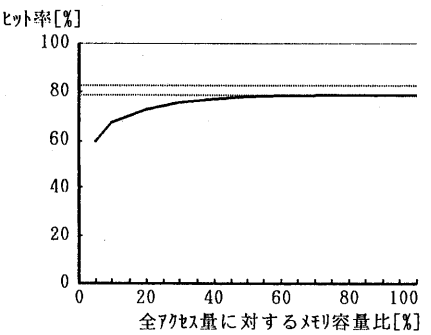

図３．データベースアクセス仮想化の効果

３．ＨＡＦの入出力仮想化方式

３．１　ＨＡＦの位置付け

ＨＡＦ（High Performance Access Facility）の目的は、汎用大型計算機に実装されるＧＢオーダの大容量メモリを積極的に利用して、恒久データセットに対する入出力を仮想化し、システムの高性能化を図ることである。

入出力仮想化機能の実現には次のような、信頼性、及びシステム運用上の問題点がある。

（１）出力データのオリジナルへの反映方式
（２）障害発生時の回復方式
（３）データ更新の排他方式
（４）システム間共用制御方式
（５）システム構成の変更時の管理方式

このような制御は入出力を仮想化しようとするＯＳコンポーネント／アプリケーションにより、最適な方式が異なる。一方、そのためにそれぞれが入出力の仮想化機能を開発するのは生産性の点で問題がある。

そこでＨＡＦでは「入出力仮想化機能をＯＳの備えるべき基本機能」として位置付け、メモリ上のデータの管理、及び上記のような制御を可能とする機能を汎用的なインタフェース（ＨＡＦマクロ）の形で提供し、ＯＳコンポーネント／アプリケーションから共通に利用して、容易に入出力仮想化を実現できるようにした。

ＨＡＦは入出力仮想化に必要なメモリを管理し、このメモリ上に展開される仮想ファイルを導入した。データセットに対する入出力を仮想ファイルに対するアクセスに置き換えることにより入出力の仮想化を実現させる。各コンポーネント／アプリケーションはそれぞれのデータセットアクセスを行い、ＨＡＦの提供する機能を利用して、最適な入出力仮想化方式を採用することができる。

一方、ＨＡＦ適用プログラムを入出力仮想化のために使用する記憶媒体と独立させたことにより、システムはＨＡＦの領域管理、データの管理、及び記憶階層の最適制御を通じてフレキシブルな資源運用が可能となる。例えば、将来、より有利な記憶階層が現れた場合、ＨＡＦがそれを利用するだけでシステム全体がその恩恵を享受できるようになる。

３．２　ＨＡＦの適用モデル

アクセスパターンにより、最適な入出力仮想化方式は異なる。ＨＡＦはカタログ、データベースなどを適用対象と想定しており、参照の局所性のあるランダムなアクセスの仮想化を目的とする。順アクセスの場合、先読みが有効であるが、実際の入出力を起動するプログラムが行うものとし、ＨＡＦでは対象としない。

入出力仮想化方式には出力データのオリジナルへの反映方式によってライトスルー方式、及びライトアフタ方式があり、どちらを採用するかで制御の骨格が定まる。ＨＡＦはこれらの２方式を適用モデルとして機能を定めており、いずれの方式でも適用できる。

（１）ライトスルー方式

出力要求の度に、オリジナルに反映する方式である。障害時にも最新のデータがディスク上に残り、信頼性の点で優れるが、出力は削減されないので、入力がある程度多いことが採用の条件である。

（２）ライトアフタ方式

出力要求のあったデータを後からまとめてオリジナルに反映する方式である。出力の削減をねらいとしているが、障害発生時にはデータが失われる。信頼性を重視しないアプリケーション：例えば繰り返し実行のできるプログラムなどで採用できる。

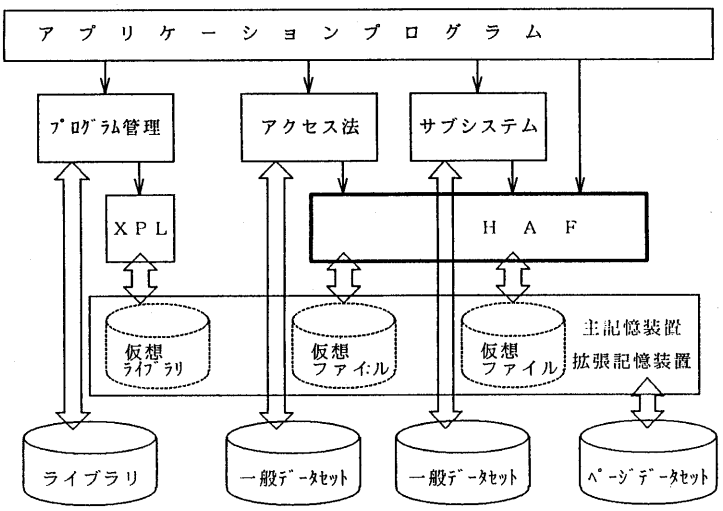

図４．ＨＡＦの位置付け

３．３　アクセス法への適用

ＨＡＦは各コンポーネント／アプリケーションにより使用可能であるが、単独利用だけに限ると効果の波及範囲が狭く、特に既存のプログラムの高性能化に結び付かない。しかし、ＨＡＦを既存のアクセス法に適用すると、アクセス法を使用するコンポーネント／アプリケーションの入出力を自動的

に仮想化することができ、プログラムを全く変更することなく、その高性能化が実現できる。ＶＳＡＭ(Virtual Storage Access Method) はカタログ、データベースなどに用いられている汎用のアクセス法であり、他にもＶＳＡＭを利用したアプリケーションも多く、波及効果が大きい。そこで我々はＨＡＦの適用対象としてＶＳＡＭを選択した。

ＶＳＡＭが取り扱うのはカタログやデータベースなどシステムの要となるデータセットであり、最高の信頼性が要求される。そのため、ＨＡＦを信頼性の高いライトスルー方式にて適用する。この場合、出力の削減はできないが、第２章にも見た通り、入力の削減だけでもかなりの効果が期待できる。出力がアクセスの大部分であったり、再参照が行われないなど、データセットのアクセスパターンによってはＨＡＦの適用が適当でない場合がある。また他システムと共用しているデータセットやＶＳＡＭ以外のプログラムが更新する可能性があるデータセットに不用意に適用するとデータの矛盾を生ずることがある。従って、データセットの特性を無視した無制限なＨＡＦの適用はできず、何らかの選択制御が必要である。そこで、ＨＡＦのＶＳＡＭ適用ではＨＡＦを適用するデータセットをシステムパラメータ、ないしジョブ制御文に指定できるようにした。ＨＡＦ適用が指定されたデータセットを対象データセットと呼ぶ。

３．４　アプリケーションによる利用

例えばＤＢＭＳ (DataBase Management System) 個有の情報に基づき、オペレーティングシステムの入出力仮想化機能をうまく制御することができれば、入出力を効果的に削減できる［３］。ＨＡＦでは実際の入出力、及びステージングの主導権は適用プログラム側に与えられており、ＨＡＦの記憶階層制御と組合せて、独自の最適制御を可能としている。

４．ＨＡＦの機能概要

４．１　ＨＡＦの制御対象

ＨＡＦ適用プログラムは以下のオブジェクトをＨＡＦマクロを用いて操作することにより、入出力の仮想化を図る。

（１）ＨＡＦエリア：ＡＲＥＡ

ＨＡＦがデータを格納する仮想記憶上の領域。独立に管理される領域を複数個作成することができ、各々をサブエリアと呼ぶ。アクセス性能を保証するため、実記憶または拡張記憶にマッピングされる。

（２）仮想ファイル：ＦＩＬＥ

ＨＡＦが取り扱う一連のデータの集まり。仮想ファイル名称（１２８バイト）により識別される。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。複数の仮想ファイルのデータが一つのサブエリアに格納されてもよい。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。

（３）データ：ＤＡＴＡ

データはいずれかの仮想ファイルに属す。仮想ファイル内では、各データはデータ番号（８バイト）により識別される。

（４）対象データセット名　　：ＮＡＭＥ

ＨＡＦ適用対象／非対象とするＶＳＡＭデータセットの名称。

４．２　ＨＡＦの制御対象の操作

上記オブジェクトに対し、表１のようなオペレーションが可能である。以下に主な機能について説明する。

（１）ＡＲＥＡの操作

サブエリアの作成、削除、及び容量の変更が可能である。容量はＭＢ（メガバイト：10^6 byte）単位で指定する。ＨＡＦ適用プログラムは仮想ファイルが使

表１．ＨＡＦ機能一覧

対　象	操　作	内　容
AREA	DEFINE	サブエリアの作成
	DELETE	サブエリアの削除
	EXPAND	サブエリアの拡張
	REDUCE	サブエリアの縮小
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）
	CLOSE	仮想ファイルの消去／使用終了
	INACTIVATE	仮想ファイルの閉塞（消去／サービス停止）
	ACTIVATE	仮想ファイルの閉塞解除（サービス再開）
DATA	GET	データの入力（ミス時はステージング用書き込み予約）
	PUT	書き込み予約済みデータの（ＨＡＦエリアへの）出力
	RESERVE	出力データの書き込み予約
	RELEASE	書き込み予約の解除
	PURGE	データの閉塞（削除／アクセス禁止）
NAME	ADD	対象／非対象データセット名称の登録
	REMOVE	対象／非対象データセット名称の登録解除
	QUERY	対象データセット名称の問い合わせ

用するサブエリアを作成しておかなければならない。

（２）ＦＩＬＥの操作

ＨＡＦ適用プログラムがデータを取り扱うためには、作成済みのサブエリアを指定してFILE-OPEN機能により、仮想ファイルを作成しなければならない。１つのデータセットに対し、ＨＡＦを適用しているタスクと適用していないタスクが同時に存在すると、それらのタスクの処理、及びそのデータセットの内容に矛盾が生じる。そのため、ＨＡＦ適用プログラムはこのような状況が生じないことを保証する必要がある。FILE-OPEN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、上記のような状況の発生を防ぐことができる。仮想ファイルにはOPENカウンタがあり、使用中のタスクの有無を管理している。

仮想ファイルの使用終了時はFILE-CLOSE機能により、ＨＡＦに使用終了を通知する。この時、仮想ファイルを消去せず、データとともに残しておくことができる。これにより、他のタスク／ジョブによる仮想ファイル／データの再利用が図られる。

一方、障害発生などの場合は、仮想ファイルを閉塞し、ＨＡＦのサービスを停止することができる。使用中のタスクがある場合はデータ入力を要求してもＨＡＦはデータを渡さない。更に一旦停止したサービスを再開することもできる。

（３）ＤＡＴＡの操作

DATA-GET機能でＨＡＦにデータを要求すると、ＨＡＦはＨＡＦエリアで要求されたデータを探索する。データがあれば（ヒットすれば）指定のユーザ領域にデータを転送し、なければ（ミスすれば）その旨をリターン情報として返す。

データアクセスの競合によるＨＡＦエリア上のデータとオリジナルデータとの不一致を防ぐため、データの書き込みでは、予めＤＡＴＡを書き込み予約しておく必要がある。書き込み予約はDATA-RESERVE機能で明示的に行う他、DATA-GET機能で指定したデータがＨＡＦエリアでミスした場合にも行われる。これはこの直後にＨＡＦエリアにオリジナルデータセットから入力したデータ書き込む（ステージングする）のがライトスルー方式における通常の適用方法だからであるある。

データの書き込み予約状態はDATA-PUT機能により、データが書き込まれた時に解除される。障害発生などによりデータ書き込みができなくなった場合は、DATA-RELEASE機能により書き込み予約状態を解除する必要がある。またＨＡＦエリア上のデータに矛盾が発生したことが検出された場合はＨＡＦ適用プログラムはDATA-PURGE機能により、そのデータをＨＡＦエリアから削除することができる。

データの書き込みは DATA-PUT機能で行うが、この時、上位の記憶階層にステージングすることを指定できる。ＨＡＦは２階層ＬＲＵ（後述）における再参照のあったデータと同じ扱いをする。

（４）ＮＡＭＥの操作

対象データセット名称の登録／削除ができる。また指定したデータセットが対象データセットか否かを問い合わせることができる。

４．３　ＶＳＡＭにおけるＨＡＦ適用方式

上記機能のライトスルー方式での標準的な適用例としてＶＳＡＭにおける適用方式を紹介する（図５）。

（１）ＶＳＡＭはＯＰＥＮ処理中のＶＳＡＭデータセットがＨＡＦ適用対象データセットであるか否かをNAME-QUERY機能により問い合わせ、対象データセットであった場合は、対象化処理（FILE-OPEN）を行う。

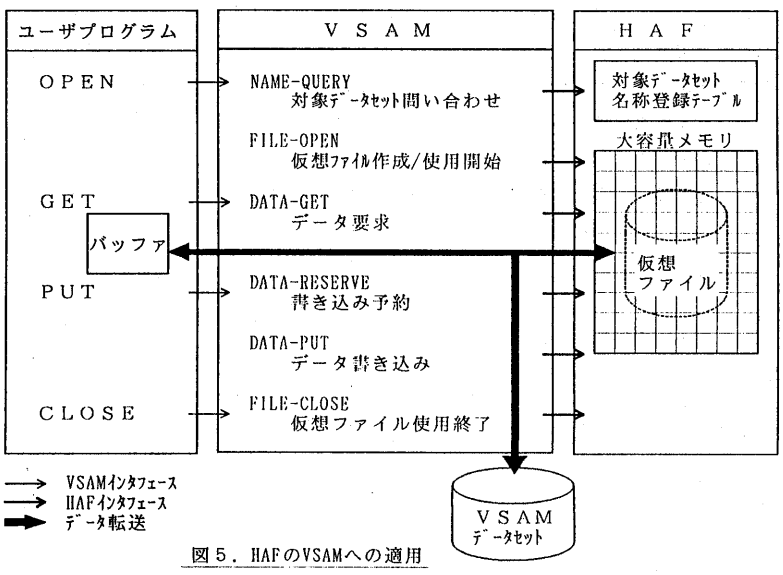

図５．HAFのVSAMへの適用

（２）対象データセットの入力要求処理では、ＨＡＦに対してデータの入力要求（DATA-GET）を出す。

（３）要求したデータがヒットした場合は、ＨＡＦはこのデータをＶＳＡＭの指定した領域へコピーし、入力処理は完了する。

（４）データがミスした場合は、ＶＳＡＭはオリジナルデータセットからデータを物理的に入力し、ユーザプログラムに渡すとともに、入力したデータをＨＡＦに対してステージング（DATA-PUT）しておく。

（５）データ出力要求処理では、該当データを書き込み予約（DATA-RESERVE）し、オリジナルデータセットに書き込んでから、ＨＡＦエリア上のデータを更新（DATA-PUT）する。

（６）ＨＡＦの記憶階層制御により、ＨＡＦエリア上のデータが自動的に失われる（デステージされる）ことがあり、この場合、以前にステージングしたデータについて再び入力要求はミスとなる。このような場合でも、ＶＳＡＭはオリジナルデータセットに最新データを格納しており、改めて入力／ステージングを行えばよい。

４．４　ＨＡＦの運用制御

ＨＡＦが用いるメモリ量、及び対象データセットはシステム全体の資源運用方針に基づき設定する必要がある。そのためにＨＡＦではこれらをシステムパラメータ、及びオペレータコマンドにより指定／変更することができるようにした。

（１）ＨＡＦパラメータ

システムパラメータとして以下が指定できる。

表２．ＨＡＦパラメータの概要

指定項目	内　容
サブエリア	容量、属性
対象ﾃﾞｰﾀｾｯﾄ	ﾎﾞﾘｭｰﾑ/ﾃﾞｰﾀｾｯﾄ名、使用ｻﾌﾞｴﾘｱ 総称名によるｸﾞﾙｰﾌﾟ指定可能
非対象ﾃﾞｰﾀｾｯﾄ	同上
その他	JCLによる対象化の可否

（２）ＨＡＦコマンド

ＨＡＦコマンドにより、以下の操作ができる。

表３．ＨＡＦコマンドの概要

指定項目	内　容
対象ﾃﾞｰﾀｾｯﾄ	追加、削除、使用サブエリア
非対象ﾃﾞｰﾀｾｯﾄ	同上
ＨＡＦ停止	
ＨＡＦ再開	ＨＡＦﾊﾟﾗﾒｰﾀが指定可能

これらのコマンドにより、ＨＡＦの柔軟な運用が可能になる。例えば、昼間と夜間でシステムの運用形態が変わり、ＨＡＦの適用の方法も変更したい場合、システムを止めずに、ＨＡＦ停止／再開コマンドにより、ＨＡＦパラメータを取り換えることができる。

４．５　ＨＡＦ稼働状態のモニタリング

ＨＡＦの効果を最大限に引き出すためには、ＨＡＦ稼働環境の最適化を図る必要がある。このチューニング作業を支援するために、ＨＡＦの稼働状態のモニタリングを可能とした。以下の情報をコンソール表示、ないしバッチジョブの出力として得ることができる。

表４．ＨＡＦモニタリング情報の概要

対　象	表示内容
サブエリア	ｱｸｾｽ回数、ﾋｯﾄ率、容量
仮想ファイル	ｱｸｾｽ回数、ﾋｯﾄ率、ﾃﾞｰﾀ量
対象ﾃﾞｰﾀｾｯﾄ	登録ﾃﾞｰﾀｾｯﾄ名/総称名

例えば、ヒット率の低い仮想ファイルがあった場合、容量を増やす、別のサブエリアを使用させる、適用をやめるなどの措置がとれる。

５．ＨＡＦの制御方式

５．１　初期設定

ＨＡＦの稼働環境はシステムの起動時、及びＨＡＦ再開コマンド処理時にＨＡＦパラメータに従って作成する。対象データセット名の登録（NAME-ADD）、使用するサブエリアの確保（AREA-DEFINE）を行う。これらは主にＶＳＡＭ適用のための環境である。それ以外の適用プログラムはそれぞれの初期設定時に、ＨＡＦ適用環境を作成する必要がある。

５．２　データの管理方式

ＨＡＦエリアは４ＫＢのページ単位に管理する。そのため、データ長に原理的な制限はないが、４ＫＢの場合が格納効率、処理効率ともに最良である。

データは一つのサブエリア内でＬＲＵ管理し、異なるサブエリア上の仮想ファイル同士は互いのアクセスの影響を受けないようにする。仮想ファイル毎のハッシングによりデータサーチの高速化を図っている。

５．３　記憶階層制御

（１）ＨＡＦエリアのメモリ常駐化

ＨＡＦエリアは記憶階層上にあり、ページングの対象となる。しかし、データが２次記憶までページアウトされてしまうと、メモリキャッシュとしての意味がない。そのため、ＨＡＦエリアはメモリに固定する。

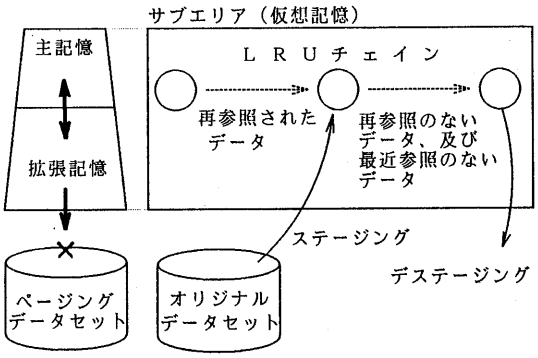

図６．ＨＡＦの記憶階層制御

但し、ＨＡＦエリアは大容量なので、拡張記憶が使用できる場合は、ＨＡＦエリアを主記憶ないし拡張記憶にマッピングし、システムのメモリ負荷を緩和する。

（２）２階層ＬＲＵ制御

ＨＡＦエリア上にどのデータを残すかといった選択制御はＬＲＵアルゴリズムを基本としているが、ＨＡＦでは、一般データセットのアクセスに適用することを考慮し、その変型である２階層ＬＲＵ方式を採用している（図６）。

本方式では「再参照の実績のあるデータとまだ再参照されていないデータのは今後の再参照可能性には差がある」と考え、両者を区別してＬＲＵ管理する。本方式によると、一過性の大量入力によりＨＡＦエリアに書き込まれたデータを早期にＨＡＦエリアから追い出す事ができる効果がある。

６．ＨＡＦの適用効果

６．１　適用容易性

ＶＳＡＭにＨＡＦを適用した際の開発量は、ＶＳＡＭ特有の制御／障害回復機能なども含めて、ＨＡＦの開発量の数％であった。これはＨＡＦの適用容易性を示している。

６．２　ＬＩＭＥデータセット適用効果

ＬＩＭＥ（Llibrary Management and Editing system）はＶＳＡＭデータセットを用いたソースプログラムやロードモジュールのライブラリ管理システムである。メンバ管理のためのディレクトリ、ビットマップなどの再参照性の高いデータを含み、ＶＳＡＭのＨＡＦ適用の効果が高い。

通常の区分データセットからＬＩＭＥデータセットへの変換処理、及びＬＩＭＥデータセットからＬＩＭＥデータセットへのコピー処理ではＨＡＦを適用すると、適用しない場合に較べて、実行時間が半減し、ＣＰＵオーバヘッドも４％以上削減される。

６．３　カタログ適用効果

カタログデータセットはシステム内のデータセットの存在情報、及びセキュリティ情報などが格納されているＶＳＡＭデータセットである。一般データセットの作成／削除／使用の際には必ずアクセスされるため、その入出力の仮想化は特にＴＳＳコマンドのレスポンス性能向上に効果がある。

ＴＳＳ端末シミュレータを用いてＨＡＦ適用時と非適用時のコマンド応答時間を測定した。

（１）ソースデータセットの編集開始／終了など、カタログアクセスのあるコマンドは平均応答時間が２０％向上した。

（２）テキストサーチなどのトリビアルコマンドの平均応答時間では数％劣化するものがある。これは物理的な入出力が削減されて入出力割り込みが少なくなり、タスクスイッチの機会が減少し、ＣＰＵ待ち時間が増えたためと考えられる。

（３）スループットは４％向上した。

７．おわりに

汎用大型計算機システムにおける入出力の仮想化を容易に実現する汎用入出力仮想化機能ＨＡＦを開発し、汎用のアクセス法であるＶＳＡＭに適用した。これにより、ＶＳＡＭデータセットユーザは既存のインタフェースのままで入出力を削減し、性能向上を図ることができる。ＨＡＦ対象データセットの選択指定ができ、ＨＡＦ稼働状態のモニタリング情報によって、そのチューニングが可能である。また、ＶＳＡＭ適用の効果として、ＬＩＭＥデータセット、及びカタログデータセットに適用した場合の評価を行った。

８．参考文献

［１］新井他：拡張入出力制御方式ＸＶＩＯフェーズ１の開発、ＯＳ研究会資料1984年12月

［２］Rubsam,K.:MVS data services、
IBM Systems J.,Vol 28,No 1,1989

［３］Stonebraker,M.:Operating System Support for Database Management、
CACM,Vol 24,No 7,pp412-418,July 1981