システムソフトウェアと
オペレーティング・システム 61－2
（１９９３．８．１８）

ストライプド高速UNIX ファイルシステムの開発
－バーチャルアレイ・ファイルシステム(VAFS)－

秋沢 充　山下洋史　加藤寛次　鬼頭 昭†　牧 敏行‡　山田秀則‡
（株）日立製作所 中央研究所，　†ソフトウェア開発本部
‡日立コンピュータエンジニアリング（株）

単一のSCSIバスに接続した複数のディスク装置にストライピングによりファイルを分割格納し，多重にアクセス要求を発行して並列に動作させることにより高速なアクセスを実現する高速UNIXファイルシステム，バーチャルアレイ・ファイルシステム(Virtual Array File System:VAFS)を開発した。ファイルの分割格納を意識せず従来同様のインタフェースでアクセスできるようにするAPI仮想化方式，同一プロセスから非同期にアクセス要求を発行できるようにする非同期I/O制御方式および複数のディスク装置の並列動作を可能にする多重アクセス制御方式により実現した。１本のSCSIバス(10MB/s)でシーケンシャルアクセス8MB/s，ランダムアクセス2MB/sを達成した。

SCSI:Small Computer System Interface, API:Application Programming Interface

Performance Improvement on UNIX File System with Striping
－Virtual Array File System (VAFS)－

Mitsuru Akizawa, Hirofumi Yamashita, Kanji Kato,
Akira Kito †, Toshiyuki Maki ‡, and Hidenori Yamada ‡
Central Research Laboratory and † Software Development Center, HITACHI, Ltd.
‡ HITACHI Computer Engineering Co., Ltd.

Performance improved UNIX File System with striping, has been developed. The Striped File System, called Virtual Array File System (VAFS), divides file into subfiles and stores them into disk drives connected in a single SCSI bus. In case of access, it issues multiple access requests to disk drives asynchronously to activate in parallel for fast data transferring. Virtual Application Programming Interface for regarding subfiles as one file, is also developed, together with asynchronous I/O control, and multiple disk access control. Its performance with a single SCSI-2 (10 MB/s) channel reaches up to 8 MB/s for sequential access and 2 MB/s for random access.

1．はじめに

近年，急速に進展してきたマルチベンダー分散コンピューティング環境においては，クライアント・サーバシステム(CSS)による分散協調システムが主流となってきている。ここでは，サーバが重要な位置を占め，特にファイルの共用と一元管理ならびに大容量の二次記憶空間の提供を目的としたファイルサーバに対するニーズが高まってきている。ファイルサーバにおいてはファイルアクセス性能がシステム性能を決定する大きな要因となるため，様々な高速化のアプローチが行われている。

ハードウェア的アプローチとしてはサーバ本体すなわちCPUや内部バス等の高速化の他に，I/Oバスやディスク装置の高速化が行われてきている。またディスク装置単体の性能限界を超える方法としてRAID(Redundant Array of Inexpensive Disks)技術を用いたアレイディスク装置の開発も試みられている[1]。

ソフトウェア的アプローチとしては，ディスク媒体上へのデータ配置を最適化することによりシーク時間や回転待ち時間を短縮して高速化を図るBSD(Berkeley Software Distribution)のFFS(Fast File System)における高速化方式や[2][3]，デバイスドライバにおいて複数のリクエストをマージすることによりコマンド発行回数を削減してプロトコル・オーバヘッドを短縮する方式が開発されてきている[4]。また複数のディスク装置にファイルを分割格納して並列アクセス制御することにより高速化を図るストライプド・ファイルシステムの開発も試みられている[5]。

ファイルシステムの高速化のためにはディスク装置の性能向上が大きな課題であるが，単体ディスク装置の性能を超えるためには並列動作に依らざるを得ない。しかしアレイディスク装置のように専用ハードを用いるものは高コストになるという問題点がある。またストライプド・ファイルシステムにおいてもUNIXが同期I/O制御を採用しているために性能向上に限界がある。

このような背景に鑑みて，報告者らはUNIXワークステーション/サーバに適用可能な汎用の高性能UNIXファイルシステムとしてバーチャルアレイ・ファイルシステム(Virtual Array File System:VAFS)を提案した[6]。VAFSはストライプド・ファイルシステムをベースとし，高コスト・パフォーマンスを狙ったUNIXカーネル内の汎用な高速ファイルシステムである。

本稿ではVAFSの実現方式，特にディスク装置の並列動作を実現する多重アクセス制御方式と性能評価について報告する。

2．VAFSの概要

バーチャルアレイ・ファイルシステム(VAFS)は，複数のディスク装置に分割格納したファイルを多重制御によって高速なアクセスを可能とするファイルシステムである。これを実現するために，ファイルシステムに非同期I/O制御方式を，デバイスドライバにディスク装置の多重アクセス制御方式を開発する。さらにファイルの分割格納を意識せずにアクセスできるようにするため，アプリケーション・プログラミング・インタフェース(API)仮想化方式を開発する。デバイスドライバを特にバーチャルアレイ・デバイスドライバ(VADD)と，ファイルを格納する複数のディスク装置をバーチャルアレイ・ディスク(VAHD)と呼ぶ。また，ファイルを分散格納するための分割処理は一般的にストライピングと呼ばれる。

VAFSのディレクトリ構造を図1に示す。VAFSはスペシャルファイルをルートファイルシステム(Unix File System:UFS)のディレクトリへマウント

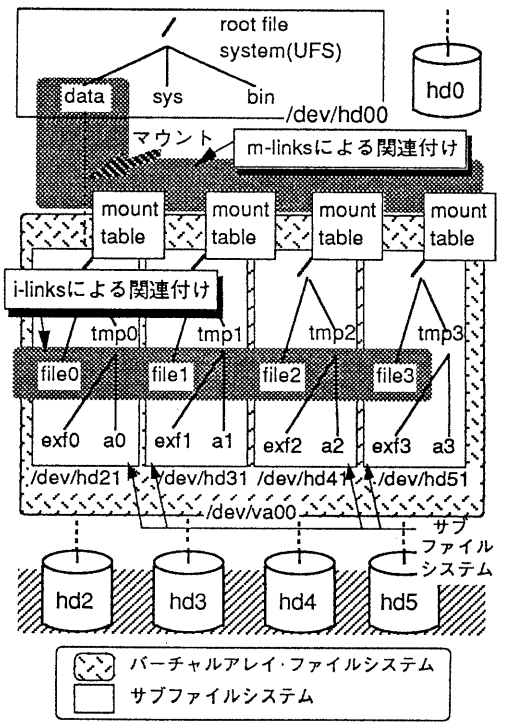


図1　VAFSのディレクトリ構造

するだけで，ファイルの分割格納を意識せずアクセスできる。VAHDを構成するディスク装置の台数を増やすことにより並列度を上げ性能を向上させることができる。また非同期I/Oシステムコールを設けることにより，アクセス制御をすべてカーネル内のソフトウェア処理で実現する。さらにVAFSを搭載したシステム上で，エンドユーザが過去に蓄積したアプリケーション・プログラム(AP)をそのアクセス・インタフェースを変えることなく実行できる。また，vnodeインタフェースにより他のファイルシステムと共存できる。

このように，VAFSは特殊なハードウェアを必要とせず，SCSIバスに標準接続された既存のディスク装置を高速・大容量のディスク装置として扱えるようにする高コスト・パフォーマンスなファイルシステム方式である。

VAFSを実現するためには以下の各方式の開発が必要となる。

(1) ファイル・ストライピング方式
(2) API仮想化方式
(3) 非同期I/O制御方式
(4) 多重アクセス制御方式

カーネルにおけるVAFSの構成を図2に示す。全体は(a)システムコール・インタフェース，(b)ファイルシステムおよび(c)デバイスドライバの各層から構成される。

(a) システムコールインタフェース層

非同期I/Oシステムコールのインタフェース機能および既存システムコールから非同期I/Oシステムコールへの自動変換機能を実現する。

(b) ファイルシステム層

ファイルシステムは上位層と下位層の2層からなる。上位層では，ファイル管理やread/writeのシーケンス制御，先読み制御，パス名とファイルおよびディレクトリとの対応づけを行なう。さらにこれらの機能を用いたファイルストライピングとAPI仮想化も本層で実現する。下位層では，論理ブロックを用いたブロック型インタフェースによるドライバ・コールおよびディスクI/Oのバッファとなるバッファキャッシュの管理を行なう。さらに，非同期I/O制御も本層で実現する。

(c) デバイスドライバ層

デバイスドライバも上位層と下位層の2層からなる。上位層ではファイルシステムとのインタフェースを取り，下位層ではディスク装置とのインタフェースを司るアダプタボード上のハードウェア・レジスタ群の制御を行う。両層で多重アクセス制御方式を実現する。

3．VAFSの技術課題

ファイルのストライピングは，ファイルシステムとデバイスドライバの間にストライピングモジュールを組み込むことにより実現できる。しかし，この方式ではファイルシステムからディスク装置が論理的にしか見えなくなるため，FFSのようなディスク上のデータブロックの配置を意識してデータ領域を割り当てるというきめ細かい高速化技術が適用できなくなるという問題点がある。またファイルシステムに非同期I/Oの発行機能を，デバイスドライバにディスクの多重アクセス機能を付加しなければ高速化を達成することが困難である。このため高速なファイルシステムを実現するためには，ストライピングのみならずこれらの課題も克服しなければならない。

すなわち，上述した機能を持つVAFSを実現するためには，次のような技術課題を解決する必要がある。

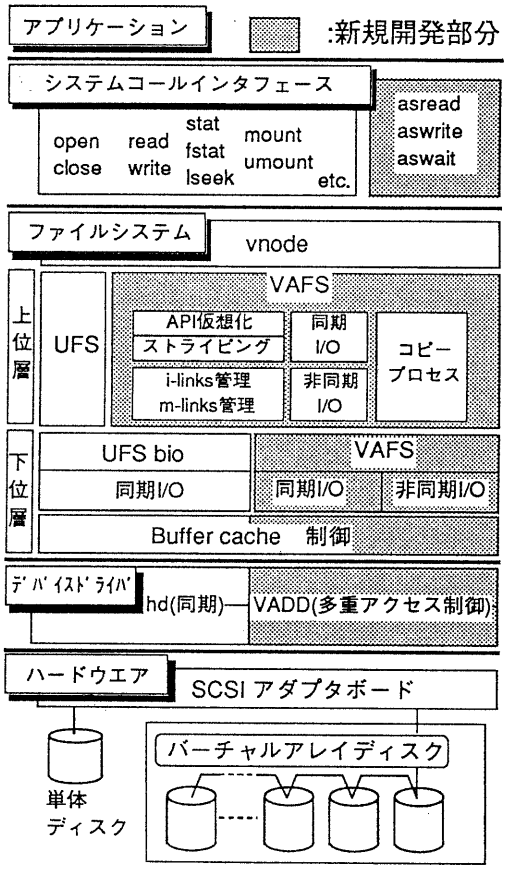


図2　カーネル内VAFS構成

3.1 ファイル・ストライピング方式の課題

ストライピング方式としては，カーネル内のファイルシステム層で行う方式とデバイスドライバ層で行う方式とが考えられる。どちらの層で行うかにより，ファイルシステムの制御方式やファイルアクセス・インタフェースへの影響が異なってくる。

したがって，ストライピングを行う層の選択が重要な課題となる。

3.2 API仮想化方式の課題

API仮想化方式の目的は，VAFSのファイルをアクセスする際に分割格納をAPIに意識させず，UFSと互換なインタフェースを提供することである。すなわちVAFSでは従来同様のディレクトリ構造のイメージをAPIに与え，VAFS内の格納ファイルをあたかも一つのファイルであるかのようにアクセスできるようにする。

すなわち，分割格納されたファイルの関連付けと，これらから構成される分割格納されたファイルシステムの関連付けが課題となる。

3.3 非同期I/O制御方式の課題

UNIXファイルシステムは本来同期I/O方式を採っているために，個々のプロセスからのファイルアクセスは前に発行されたアクセス要求の終了を待って発行される[7]。このため，分割格納したファイルに対して多重にアクセス要求を発行することができず，ファイルストライピングによる高速性を引き出せないという問題がある。

また，処理中のアクセス要求が同時に複数存在するため，I/O終了時にテーブルの解放を行うには，発行と終了の対応関係を取らなければならないという問題がある。

さらに非同期I/Oを実現したとしても，非同期リードの場合には，ディスク装置から読み込んだデータをバッファリングしておくバッファキャッシュからI/O終了後にアプリケーション空間へのデータコピーを行う際に問題が生じる。すなわち，非同期であるためにI/O終了時点ではアクセス要求を発行したプロセスがメモリ上に存在する保証がなく，コピー先の領域もメモリ上に存在する保証がないという問題である。このため発行プロセスがI/O終了割込みの延長でデータコピーを行なう既存の方式では，コピー先の領域が確保できないという問題が発生する。

したがって，従来の同期制御の非同期化，非同期I/O要求の発行履歴の管理，アプリケーション空間へのデータコピーが課題となる。

3.4 多重アクセス制御方式の課題

多重アクセス制御方式は，ディスク装置へのアクセス要求を多重に発行してディスク装置の並列動作を実現する。アクセス性能の向上を図るためにはSCSIバスの空き時間を最大限に利用して，並列動作するディスク装置から高速にデータ転送する制御方式の開発が必要となる。

そのために，READY状態のディスク装置に直ちにアクセス要求を発行できるキューイング方式，アクセス発行時のロックのオーバヘッドを最小化する排他制御方式，SCSIプロトコルのオーバヘッドを低減するリクエストのマージ方式および並列動作するディスク装置の並列度を高めるアクセス・スケジューリング方式の開発が課題となる。

4. VAFSの実現

VAFSを実現するために上記の技術課題に対して以下のような解決を図った。

4.1 ファイル・ストライピング方式の実現

デバイスドライバでのストライピングは開発規模が小さいという利点があり，ファイルシステムでのストライピングはディスク媒体上のデータブロックの配置を最適化することによりアクセスを高速化できるという利点がある。VAFSでは高速化を目的としてFFSのデータブロックを最適に配置する機能を活かすために，ファイルシステム層でのストライピングを行う方式とする。

4.2 API仮想化方式の実現

VAFSは図2に示すように複数のサブファイルシステムから構成され，各サブファイルシステムはディスク装置に割り当てられる。サブファイルシステム間の関連付けを行う構造体m-linksとサブファイル間の関連付けを行う構造体i-linksを導入し，VAFS内で対応付けを管理する。上位とのインタフェースをvnodeとして，従来のファイルシステムと同様のアクセス・インタフェースを実現する。

4.3 非同期I/O制御方式の実現

(1) 非同期I/Oシステムコール

VAFSでは非同期I/O制御方式を導入することにより高速なファイルアクセスを実現する。これを

APから利用できるようにするために，read()およびwrite()システムコールを非同期化したasread()およびaswrite()システムコールと，さらにこれらを同期化するaswait()システムコールを新たに設ける。

VAFSは非同期I/Oを使用して複数のディスク装置の並列動作を制御し，高速なファイルアクセスを実現する。このためread()あるいはwrite()を使用する既存のアプリケーションがVAFSをアクセスする場合には，これらをカーネル内部で非同期のシステムコールasread()およびaswrite()へ変換する方式とする。例えばVAFSに対して発行されたread()は，1回当り8KBのasread()を複数回発行するとともに，これらに引き続き1回のaswait()を発行する形に内部で展開し実行する。

これにより既存のアプリケーション・プログラムとの互換性を確保するとともに，VAFSとUFSどちらも同じインタフェースで利用することが可能となる。

(2) 非同期I/O制御方式による制御フロー

非同期I/O制御方式のフローの実現方式について以下に説明する。先ず，図3に非同期システムコールの制御フローを示す。

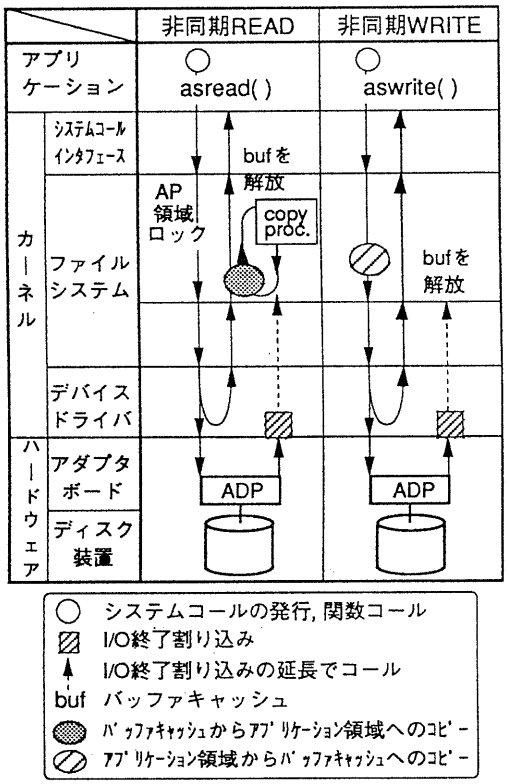


図3　非同期I/O制御のフロー

非同期システムコールasread()が発行されると，まずファイルを読み込むアプリケーション領域をメモリ上にロックする。次にディスク装置にreadアクセス要求を発行する。同期システムコールとは異なり，ここで終了待ち状態(wait)に入らずに次のステップに処理を進める。ディスク装置からバッファキャッシュへのデータの読み込み処理が終了すると，I/O終了割り込みが発生し，これによりコピープロセスが起動される。コピープロセスは，ディスク装置から読み込まれたデータが格納されたバッファキャッシュからアプリケーション領域へデータをコピーし，これが済んだ後バッファキャッシュの解放を行なってスリープする。これで非同期システムコールasread()の処理が終了する。この間該当システムコールを発行したプロセスは，この先の処理を行っている。

aswrite()でも同様に，まずアプリケーション領域からバッファキャッシュへデータをコピーし，次にディスク装置にwriteアクセス要求を発行する。ここでファイルシステム内で終了待ち状態(wait)に入らずに処理を次のステップに進める。バッファキャッシュからディスク装置へのデータの書き込みが終了すると，I/O終了割り込みが発生し，これによりバッファキャッシュの解放が行なわれる。この間該当システムコールを発行したプロセスはこの先の処理を行っている。

4.4 多重アクセス制御方式の実現

多重アクセス制御方式では，アクセス要求のキューイング方式，ディスク装置の排他制御方式，リクエストのマージ方式およびアクセス・スケジューリング方式の4点が課題となる。これらをバーチャルアレイ・デバイスドライバVADDの内部で解決する。

以下，各機能の実現方式について説明する。

(1) アクセス要求のキューイング方式

デバイスドライバ内でアクセス要求をキューイングするI/Oキューが1本しかないと，各ディスク装置へのアクセス要求が1本のキュー内に混在してシリアライズされてしまう。その結果，実行中のアクセス要求の処理が終了するまでは，他のディスク装置がREADY状態であっても新たにアクセス要求を発行することができなくなる[8]。

VAFSでは図4のようにI/Oキューおよびその管理に必要な変数や構造体をディスク装置ごとに持たせる。各I/Oキューには，そのディスク装置に対するアクセス要求だけをキューイングし，READY

状態にあるディスク装置へキュー内のアクセス要求を常に発行できるようにする。

上位層から発行されたアクセス要求は下位層に渡される。制御情報は1本のSCSIチャネルに対応したスロットごとに，ヘッダhead[slot]にリンクされたリクエスト構造体request[slot][n]を介して渡す。VADDはSCSIバスにバスフリーの時間ができると，直ちにこのリンクをたどり未発行のアクセス要求のリクエスト構造体を見つけて，該当するディスク装置へアクセス要求を発行する。

(2) ディスク装置の排他制御方式

READY状態にあるディスク装置へ常にアクセス要求を発行できるようにするためには，ディスク装置ごとにI/Oキューを用意するだけでなく，ディスク装置単位に排他制御を行う方式とする必要がある。VAFSでは排他制御をVAHDを構成するディスク装置単位に行う方式とする。これにより，アクセス要求処理中は同一のディスク装置にアクセス要求を発行せず，他のREADY状態にあるディスク装置へ全く独立にアクセス要求を発行することができるようになる。

(3) ブロックマージ方式

SCSIプロトコルのオーバヘッドを低減して高速化を図るために，複数のリクエストを一括して1回のリクエストにまとめて発行する，リクエスト・ブロックのマージ機能をVADDに導入する。

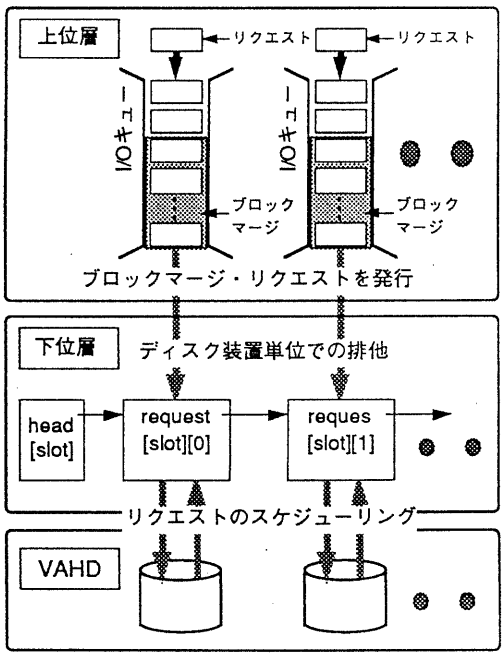


図4 ﾊﾞｰﾁｬﾙｱﾚｲ･ﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞの構成

ディスク装置へアクセス要求を発行する際に，I/Oキュー内にキューイングされている複数のリクエストがディスク媒体上で連続する場合に，これらを一括して一つのアクセス要求として発行する。マージするブロック数には上限値を設けて，上限値に至るまで一括してマージできるようにする。

(4) アクセス・スケジューリング方式

SCSIバス仕様として定義されているディスコネクト／リコネクト機能を用い，VADDの下位層で以下のようなアクセス・スケジューリングを行い，多重アクセス制御を実現する。

アクセス要求が渡されると，リクエスト構造体request[slot][i]を確保してヘッダhead[slot]で示されるリクエストチェインにつなぐ。リクエスト構造体には次の三つの状態のいずれかが表示される。

●BUSY：ディスク装置とホストがコネクト中でありSCSIバスを占有している状態。

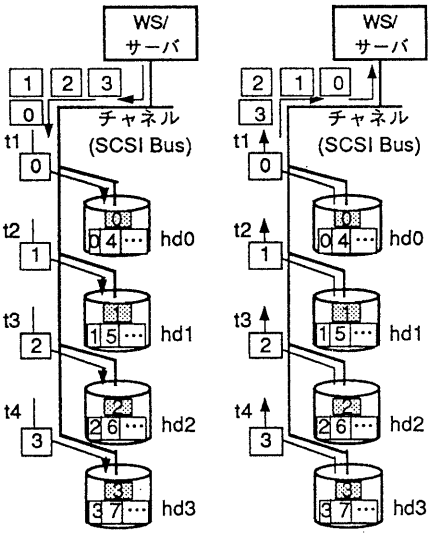
●DISCONNECT：ディスク装置は動作中であるが，ホストとは切り離されSCSIバスを空け渡している状態。

●WAIT：SCSIバスが他の装置に使用されているため，バスの空きを待っている状態。

ヘッダにリンクされているすべてのリクエスト構造体の状態をチェックし，BUSY状態のものがなければリクエストを発行する。BUSY状態のものがあればリクエストを発行せずにWAIT状態としてリストにつなぐ。I/O終了割込みの延長処理として他にWAIT状態のリクエストがあるかどうか探し，あればアクセス要求を発行して上位層へ制御を移す。

このようなディスコネクト／リコネクト制御を行なうことにより，ディスク装置内の媒体から先読みバッファへのデータの読み出しと，先読みバッファからバッファキャッシュへのデータ転送をディスク装置間で全く独立に行うことができる。これにより，ディスク装置の動作の並列度を高めることが可能になりI/Oスループットを向上できることになる。また，ディスク装置からバッファキャッシュへのデータ読み出しは，ディスク装置内の先読みバッファに読み込まれたデータに対して行うことができるようになるため，媒体読み出し以上の高速転送が実現できることになる。

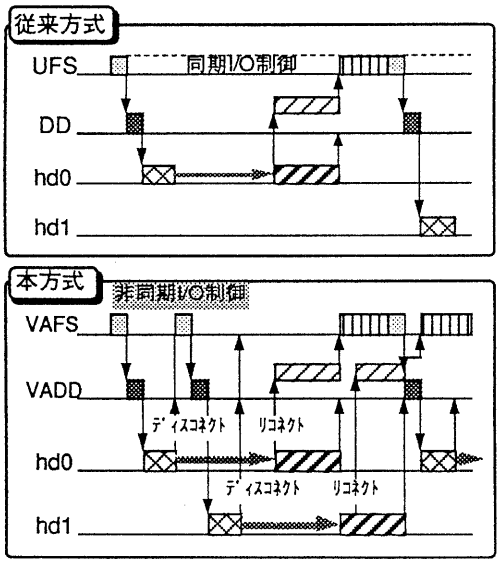
以上に説明した各機能の統合により実現される多重アクセス制御方式により，VAHDに格納されたVAFSファイルをアクセスする動作を図5と図6に示す。非同期I/O制御によりファイルシステム



(a) ライトの場合　　(b) リードの場合

<ディスク装置4台の場合>

図5　VAFSのファイルアクセス動作



<ディスク装置2台の場合>

図6　ファイルリード のタイムチャート

からアクセス要求が発行されると，デバイスドライバがディスコネクトおよびリコネクトによりアクセス・スケジューリングを行い，ディスク装置に多重にアクセス要求を発行する。従来方式に比較して，単位時間のデータ転送量が増加してアクセス速度が向上することが分かる。

上述のファイル・ストライピング方式，API仮想化方式，非同期I/O制御方式および多重アクセス制御方式を開発することによりVAFSを実現した。

5．性能評価

5.1　測定結果

開発したVAFSの性能測定を行った。日立ワークステーション3050RX/330(公称105MIPS)にディスク装置を最大4台まで接続し，シーケンシャル・アクセス性能およびランダム・アクセス性能の測定を行った。比較のためにUFSの測定も合わせて行った。なお，性能評価の条件および測定システムの機器構成は表1に示す通りである。

表1　VAFS性能測定の条件

		シーケンシャル		ランダム	
		read	write *	read	write *
VAFS	ｼｽﾃﾑｺｰﾙ	asread()	aswrite()	asread()	aswrite()
	ﾃﾞｨｽｸ	1～4台	←	←	←
UFS	ｼｽﾃﾑｺｰﾙ	read()	write()	read()	write()
	ﾃﾞｨｽｸ	1台	←	←	←
アクセスサイズ		8MBから8MBread	8MBをwrite	8MBから2MBread	2MBwrite
備　考				ﾌﾞﾛｯｸ境界を意識せずﾗﾝﾀﾞﾑにｱｸｾｽ	
機器構成	WS	3050RX/330 (105MIPS)			
	SCSIﾊﾞｽ	10MB/s (1本のみ使用)			
	ディスク	3.5inch,1.0GB,5400rpm,256KBﾊﾞｯﾌｧ			

＊：no delay書き込みのsynchronous write modeにて測定

測定結果を図7に示す。ディスク装置4台の時に以下のような性能と，対UFS加速率が得られた。

・シーケンシャルリード･･･
　性能：8.1MB/s，対UFS加速率：2.3倍
・シーケンシャルライト･･･
　性能：6.5MB/s，対UFS加速率：6.5倍
・ランダムリード･･･････
　性能：1.8MB/s，対UFS加速率：3.0倍
・ランダムライト･･･････
　性能：1.7MB/s，対UFS加速率：8.5倍

5.2 結果の考察

リードおよびライトともにシーケンシャル性能はディスク装置3台目から飽和傾向がみられる。SCSIバスの10MB/sという物理転送速度の限界が原因であると考えられる。一方ランダム性能についてはディスク装置4台目までは飽和傾向があまりみられない。SCSIバスの物理転送速度の限界まで至っていないため，ディスク装置の増設によりさらに性能向上が期待できる。

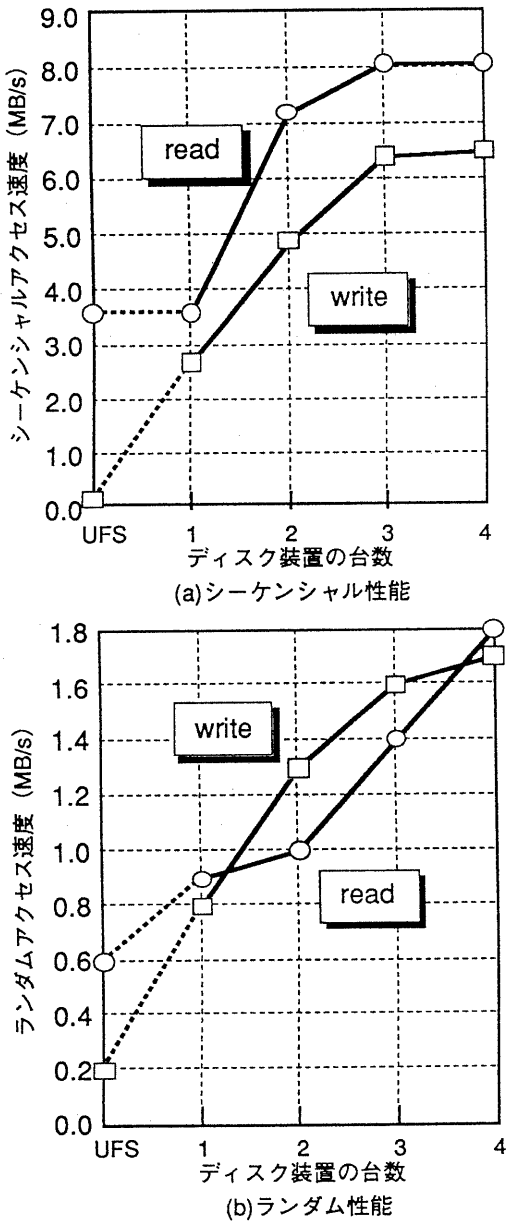


図7　VAFSの性能測定結果

6．おわりに

ファイルを分割格納したディスク装置を，多重アクセス制御方式により並列動作させることによりアクセスを高速化するストライプド高速UNIXファイルシステム，バーチャルアレイ・ファイルシステム(VAFS)の実現方式を開発し，高速化のための有効性を検証することができた。1本のSCSIバスにディスク装置をディジーチェイン接続したアクセス性能として，最大 8.1 MB/sのシーケンシャルリード性能と，最大 1.8 MB/sのランダムリード性能を達成できた。

今後は，詳細な評価・解析によるさらなる高速化の検討と，ネットワークを介した高速アクセスの実現が課題である。

参考文献

[1]D.A.Patterson, P.Chen, G.Gibson, and R.H.Katz, "Introduction to Redundant Arrays of Inexpensive Disks(RAID)" ,spring COMPCON '89, pp.112-117, Feb.1989

[2]M.K.McKusick, W.N.Joy, S.J.Leffler, and R.S.Fabry, "A Fast File System for UNIX" , ACM Trans. on Comp. System, Vol.2, No.3, pp.181-197, Aug.1984

[3]S.J.Leffler, M.K.McKusick, M.J.Karels, and J.S.Quarterman, "The Design and Implementation of the 4.3BSD UNIX Operating System" , Addison-Wesley, 1989

[4]手塚,「ワークステーション用Unixカーネルの高速化の試み」,情処研報, Vol.92, No.22(オペレーティングシステム54-4), pp.25-32, 1992.3.13

[5]Andy DeBaets, et. al, "High Performance PA-RISC Snakes Motherboard I/O" , Digest of Papers, COMPCON Spring '93, pp.433-440

[6]秋沢他5,「バーチャルアレイ・ファイルシステム(vafs)の基本構想」,情報処理学会第45回全国大会講演論文集,4-62,1992.10

[7]M.J.Bach, "The Design of the UNIX Operating System" ,Prentice-Hall,1986

[8]J.I.Egan and T.J.Teixeira(野中,大西 共訳),「UNIXデバイスドライバ」,アスキー, 1989