社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－OS－109 (8)
2008／8／7

AnT におけるファイル管理サーバの設計

野 村 裕 佑[†1] 谷 口 秀 夫[†1]

計算機の進歩に伴い，様々な場面で計算機が必要とされ，提供するサービス種別も飛躍的に増大している．このような背景から，我々は，様々な環境に適応でき，高い堅牢性を持つ ***AnT*** オペレーティングシステム (An operatings system with adaptability and toughness) を開発している．***AnT*** は，マイクロカーネル構造を採用し，ゼロコピー通信や高速なサーバ間通信制御といった機能を持つ．ここでは，***AnT*** におけるファイル管理サーバの設計について述べる．ファイル管理サーバは，ディスクドライバ等の他の OS サーバとの連携が多発するため，性能が低下すると考えられる．このため，***AnT*** のファイル管理サーバには，入出力の高速化が求められる．また，複数の論理ボリュームを統合することにより，ファイル管理の利便性の向上を目指す．

Design of File Management Server on *AnT*

Yusuke Nomura[†1] and Hideo Taniguchi[†1]

Computer systems are needed in various scenes, and the offered service type increases rapidly. We are developing the ***AnT*** operating system. ***AnT*** have both adaptability and solidity. This article explains the design of the file management server on ***AnT***. The file management server is to be decrease of the performance because of the frequent occurrence of cooperation other OS servers. Therefore, it is necessary to speed it up to the filesystem server on ***AnT***. Moreover, it aims to integrate two or more logical volumes.

1. はじめに

マイクロプロセッサや入出力ハードウェアの進歩には目覚しいものがある．また，様々な場面で計算機が必要となり，提供するサービス種別も飛躍的に増大している．このような背景から，これらのハードウェアの機能や性能を有効に利用でき，さらに多様なサービスの提供を支える基盤ソフトウェアが必要になっている．そこで，我々は，適応性と堅牢性をあわせ持つ ***AnT*** オペレーティングシステム[1](An operatings system with adaptability and toughness) を開発している．***AnT*** は，高い適応性と堅牢性を実現するために，マイクロカーネル構造を採用し，OS 機能の大半を OS から独立したプロセス（OS サーバ）として実現する．

ここでは，***AnT*** おけるファイル管理サーバの設計について述べる．***AnT*** におけるファイル管理サーバは，ファイル入出力処理の際に，ディスクドライバ等の他の OS サーバとの連携が多発するため，性能が低下すると考えられる．そこで，性能低下を抑制するようなサーバ間通信方式，およびキャッシュ機構の設計が必要である．また，従来のファイル管理では，論理ボリュームを超えるサイズのファイルを作成することが不可能であった．そこで，***AnT*** のファイル管理では，複数の論理ボリュームを利用することで，論理ボリュームを超えるサイズのファイルを作成可能にする．

†1 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

2. *AnT* オペレーティングシステム

2.1 適応性と堅牢性

AnT は，計算機システムの開発者と利用者の両者に「使いやすい」かつ「壊れにくい」という特徴を有する基盤ソフトウェアであることを目指している．

開発者にとって「使いやすい」ためには，新たなサービス（機能）の追加を容易にできる必要がある．っこのため，ドライバプログラムの新規作成の容易化だけではなく既存プログラムの流用ができれば好ましい．利用者にとって「使いやすい」ためには，高性能であるとともに，利用したいサービスを簡単に利用できる必要がある．したがって，両者の要求を満足するには，基盤ソフトウェアは高い適応性を有することが求められる．

そこで，高い適応性を有する基盤ソフトウェアの構成法が必要になる．具体的には，サービス（機能）の組み込みや取り外しが簡単なプログラム構成法，および利用者の動作履歴を考慮したプログラムの実行制御法が必要である．また，計算機システムの性能低下の大きな要因であるメモリ間データ複写をゼロにする高速なプログラム間データ通信機構も必要である．

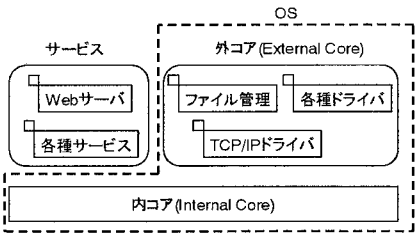

図 1 *AnT* の基本構造

開発者にとって「壊れにくい」ためには，開発中のプログラムの暴走によりシステム全体が壊れることを防ぐ必要がある．利用者にとって「壊れにくい」ためには，いろいろなサービス（機能）を使っても相互干渉によって不都合な動作にならないような高い信頼性とともに，高いセキュリティが必要である．したがって，両者の要求を満足するには，基盤ソフトウェアは高い堅牢性を有することが求められる．

そこで，高い堅牢性を有する基盤ソフトウェアの構成法が必要になる．具体的には，組み込んだプログラムに不具合が発生しても，他のプログラムやシステム全体に悪影響を与えない仕組み，およびネットワークを介したセキュリティ低下を防ぐ仕組みが必要である．

2.2 プログラム構造

プログラムは，OS とサービスからなる．OS は，内コアとプロセスとして動作する外コアからなる．サービスは，プロセスからなる．この様子を**図 1** に示す．

内コアは，最小のシステムの動作を保証するプログラム部分である．外コアは，適応したシステムに必須なプログラム部分であり，動的に再構成な構造を有する．サービスは，サービスを提供するプログラム部分である．

2.3 ゼロコピー通信

プロセスと内コアの間あるいはプロセス間の通信を高速化するため，ゼロコピーでのデータ授受機能[2] がある．授受を行うデータについては，ページ単位で管理される領域にデータを格納して通信を行う．この領域を，コア間通信データ域 (ICA: Inter-core Communication Area) と名づける．ICA は，内コアにより管理される．

プロセスと内コアの間あるいはプロセス間でのデータ授受において，データの複写を避けるため，ICA に格納されたデータは，仮想空間のマッピング表を書き換えることによりデータを授受する．これを ICA の貼り替えと名づける．ICA はページ単位であるため，このような貼り替えが可能である．

プロセス間通信を行う場合，送信プロセスの仮想空間にある ICA を受信プロセスの仮想空間に貼り替える．つまり，通信する 2 つのプログラム間でのデータ授受を 2 仮想空間の間での ICA の貼り替えにより実現し，ゼロコピー通信を実現する．

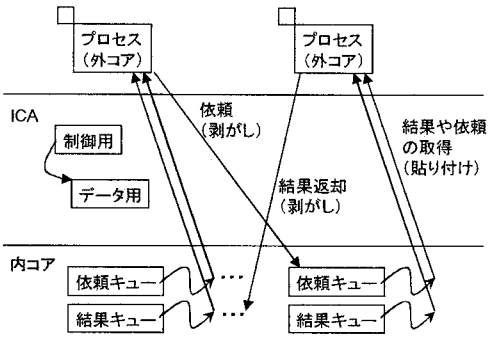

図 2 プログラム間通信の基本機構

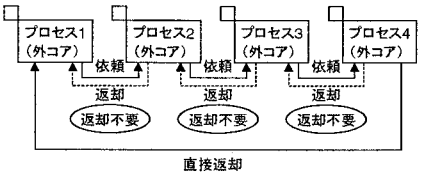

図 3 直接返却

2.4 サーバ間通信制御

AnT は，OS の機能の大半を独立した OS サーバとして実現する．このため，OS 処理の際に OS サーバ間の処理連携が多発し，性能が低下してしまう．そこで，この性能低下を抑えるための高速なプログラム間通信機構[3)4)] がある．

AnT のサーバ間呼び出し制御の基本機構を**図 2** に示す．内コアは，各 OS サーバに依頼キューと結果キューを用意している．呼び出しは，制御用 ICA とデータ用 ICA を授受することで行う．基本的な呼び出しの処理の流れを以下に説明する．

(1) 依頼元プロセスが依頼先プロセスの依頼キューに依頼を登録する．

(2) 登録された情報を依頼先プロセスが取得し処理を実行する．

(3) 依頼先プロセスが依頼元プロセスの結果キューに結果を登録して返却する．

また，***AnT*** のサーバ間呼び出し制御の特徴的な機能として，直接返却がある．この様子を**図 3** に示す．多段呼び出しされた処理は，必ずしも逐次返却が必要ではない．たとえば，AP プロセスから要求されたデータ入力は，ファイル管理部，ブロック管理部，ディスクドライバの 3 つの OS サーバを介して処理が進められる．しかし，入力データの返却結果は，ディスクドライバから直接 AP プロセスに返却できる場合もある．そこで，依頼時に結果返却の要否を指定することにより，直接返却を可能にする．

3. ファイル管理の設計

3.1 要　　求

ファイル管理の設計にあたり，以下の要求がある．

(1)　ファイル入出力の高速化

高速なファイル入出力を実現する．***AnT*** では，ファイル管理やディスクドライバといった OS のモジュールを，個別のプロセスとして実現している．このため，モノリシックカーネルと比較して，ファイル入出力時に通信のオーバヘッドが発生する．このオーバヘッドを抑制し，処理を高速化することが望まれる．

(2)　論理ボリュームを超えるサイズのファイルの実現

通常，1 つの論理ボリュームは，1 つのファイルシステムのみが利用できる．この場合，小さいファイルが大きくなった際に，1 つの論理ボリュームに割り当てているディスク容量が足りなくなり，拡張した領域をディスクへ格納できない場合が起こりうる．そこで，このような状況に対処し，大きくなったファイルをディスクへ格納可能にすることが望ましい．

3.2 対　　処

前節で述べた各要求を満足するための対処策を以下に述べる．

(1)　ファイル入出力の高速化

(A) サーバ間通信数削減

プロセスがファイル入出力を行うとき，複数のサーバが他段階に呼び出されるため，サーバ間通信によるオーバヘッドが発生する．このため，サーバ間通信数を削減することにより，ファイル入出力を高速化する．

(B) ゼロコピーでのデータ授受

複数サーバ間でのデータ授受の際に，データの複写が発生すると，性能が低下する．このため，データ複写を抑制することにより，ファイル入出力を高速化する．

(2)　論理ボリュームを超えるサイズのファイルの実現

(A) 複数論理ボリュームの統合

複数の論理ボリュームを統合し，一つの論理ボリュームとして扱うことができるようにする．

4. 実 現 方 式

4.1 構　　成

ファイル管理の構成を図 **4** に示し，以降でそれぞれのプロセスの役割について説明する．

(1)　ファイル管理部

open, close といったファイルのアクセス情報とファイル情報（i ノード相当）を管理する．i ノードキャッシュを持つ．実際のデータの入出力は，ブロック管理部に処理を依頼する．

(2)　ブロック管理部

ファイルのブロックごとの入出力を管理する．ブロックキャッシュを持つ．実際にディスクドライバとデータの授受を行う．

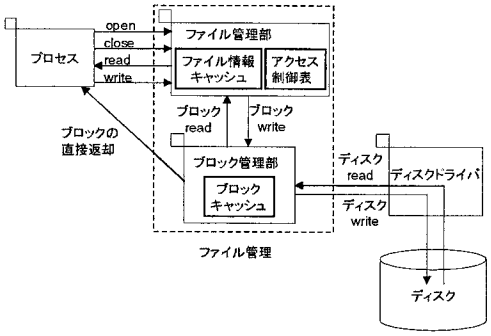

図 4　ファイル管理の構成

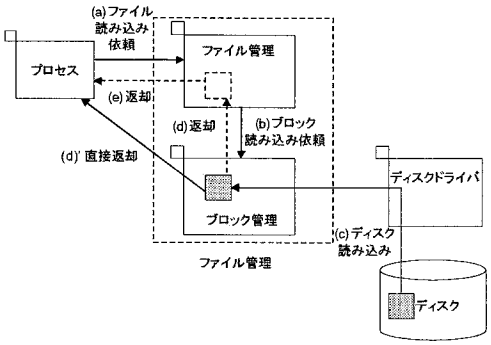

図 5　ファイル読み込み時の直接返却の流れ

4.2 具体的な対処

3.2 節で述べた対処について，以降に具体的に述べる．

(1)　サーバ間通信数削減

ファイル読み込み時の直接返却の様子を図 **5** に示し，以降で説明する．ファイル読み込みは，以下の手順で実現する．(a) プロセスがファイル管理部にファイル読み込み処理を依頼する．(b) ファイル管理部がブロック管理部にブロック読み込み処理を依頼する．(c) ブロック管理部がブロックキャッシュからデータを探索する．または，ディスクドライバにディスク読み込み処理を依頼し，ブロックデータを読み込む．(d) ブロック管理部がブロックデータをファイル管理部に返却する．(e) ファイル管理部がブロックデータをプロセスに返却する．

このとき，ファイル読み込みのデータ返却時は，ファイル管理部を経由せずに，(d)' ブロック管理部からユーザプロセスに直接返却することで，サーバ間通信数を削減する．

(2)　ゼロコピーでのデータ授受

ブロックサイズを 4KB 固定とし，ICA を用いてキャッシュ管理する．キャッシュ管理の様子を図 **6** に示す．ICA を用いてキャッシュ管理を行うことで，プロセスとファイル管理間のブロックの授受は，ICA の

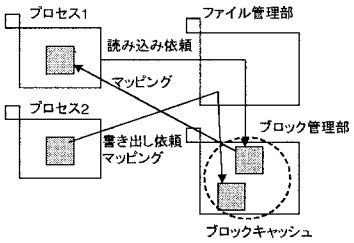

図 6　キャッシュ管理

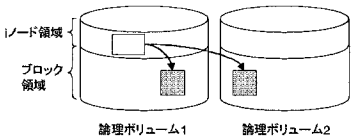

図 7　論理ボリュームの一括管理

貼り付けと剥がしで行うことができる．これにより，ゼロコピーでのデータ授受を実現する．読み込み時は，ブロックデータを格納した ICA を，ブロック管理部の仮想空間からプロセスの仮想空間に貼り付ける．書き出し時は，ブロックデータを格納した ICA を，プロセスの仮想空間からブロック管理部の仮想空間に貼り付ける．

(3)　複数論理ボリュームの統合

ブロック管理部が，複数の論理ボリュームを一括管理する．論理ボリューム 1 上の i ノードから，論理ボリューム 2 のブロックを参照できるようにする．概念を**図 7** に示す．

4.3　インタフェース

4.3.1　ファイル管理部

ファイル管理部のインタフェースを**表 1** に示し，以降で説明する．

(1)　オープン

ファイルをオープンする．ファイルとプロセスの対応をアクセス制御表に書き込み，ファイル情報番号を返す．

(2)　クローズ

アクセス制御表から，ファイルとプロセスの対応を削除する．

(3)　ファイル読み込み

ファイルの指定した位置のデータを読み込んだ ICA を，依頼元プロセスの仮想空間に貼り付ける．貼り付けられた ICA の保護は，引数の保護情報により設定する．ブロックデータの読み込みのために，ブロック管理部に処理を要求する．返却時は，ファイル管理部を経由せずに，ブロック管理部から依頼元プロセスへ直接返却を行う．

(4)　ファイル書き出し

ファイルの指定した位置に，指定した ICA の内容を書き出す．

表 1　ファイル管理部のインタフェース

(1) オープン	
機能	ファイルをオープンする
引数	fname: ファイル名 flag: アクセス情報
戻り値	ファイル情報番号
(2) クローズ	
機能	ファイルをクローズする
引数	fi: ファイル情報番号
戻り値	なし
(3) ファイル読み込み	
機能	ファイルの指定した位置のデータを読み込む
引数	fi: ファイル情報番号 offset: 読み込み開始位置 size: 読み込みサイズ prot: 返却される ICA の保護情報
戻り値	読み込んだブロックを格納した ICA の先頭アドレス
(4) ファイル書き出し	
機能	ファイルの指定した位置のデータを書き出す
引数	fi: ファイル情報番号 offset: 書き出し開始位置 size: 書き出しサイズ prot: 返却される ICA の保護情報
戻り値	prot が放棄の場合はなし 放棄以外の場合は書き出したブロックを 格納した ICA の先頭アドレス

表 2　ブロック管理部のインタフェース

(1) ブロック読み込み	
機能	指定したブロックのデータを読み込む
引数	blkaddr: 読み込むブロックアドレス size: 読み込みサイズ prot: 返却される ICA の保護情報 (読み込み専用/読み書き可能)
戻り値	読み込んだブロックを格納した ICA の先頭アドレス
(2) ブロック書き出し	
機能	指定したブロックにデータを書き出す
引数	blkaddr: 書き出し先のブロックアドレス buf: 書き出すデータを格納した ICA の先頭アドレス size: 書き出しサイズ prot: 返却される ICA の保護情報 (所有権放棄/読み込み専用/読み書き可能)
戻り値	書き出したブロックを格納した ICA の先頭アドレス
(3) キャッシュの強制書き出し	
機能	ブロック管理部が保持するキャッシュを 全てディスクに書き出す
引数	なし
戻り値	成功 or 失敗

ブロックデータの書き出しのために，ブロック管理部に処理を要求する．

4.3.2　ブロック管理部

ブロック管理部のインタフェースを**表 2** に示し，以降で説明する．

(1)　ブロック読み込み

指定したブロックを ICA に読み込む．このとき，指

定したブロックが，ブロック管理部内のキャッシュに存在する場合と存在しない場合の 2 つの場合がある．
(A) ブロックがキャッシュに存在しない場合は，ディスクからブロックを読み込み，対応する ICA を返却する．
(B) ブロックがキャッシュに存在する場合は，そのまま対応する ICA を返却する．
(2)　ブロック書き出し
指定したブロックを書き出す．このとき，指定したブロックが，ブロック管理部のキャッシュに存在する場合と存在しない場合の 2 つの場合がある．
(A) 指定したブロックのキャッシュが既に存在する場合は，データを上書きする．
(B) 指定したブロックがキャッシュに存在しない場合は，指定したブロックをキャッシュに格納する．
(3)　キャッシュの強制書き出し
キャッシュに格納されているデータをディスクに書き出す．

4.4　データの保持と保護

4.4.1　データの状態

ブロック管理部で管理されるブロックのキャッシュの状態には，以下の 4 つがある．それぞれの状態について，以降で説明する．
(1)　データなし
ブロック管理部がブロックのキャッシュを保有していない状態である．ブロックのデータはディスク上にのみ存在する．
(2)　ブロック管理部が保有
ブロック管理部がブロックのキャッシュを保有しており，キャッシュを他プロセスが共有していない場合である．
(3)　ブロック管理部と他 AP が共有 (R)
ブロック管理部がブロックのキャッシュを保有しており，キャッシュを他プロセスの仮想空間に読み取り専用で貼り付けている場合である．
(4)　ブロック管理部と他 AP が共有 (RW)
ブロック管理部がブロックのキャッシュを保有しており，キャッシュを他プロセスの仮想空間に読み書き可能で貼り付けている場合である．

4.4.2　データ操作の種類

ファイルのデータを読み込む際には，データが格納された ICA を自プロセスの仮想空間に貼り付ける．このとき，自プロセスに貼り付ける ICA の保護状態を設定することができる．読み込みの場合に 2 種類，書き出しの場合に 3 種類の保護属性がある．読み込みの場合の保護を**図 8** に，書き出しの場合の保護を**図 9** に示し，以降で説明する．
(1)　読み込み
(a)　読み取り専用 (R)
ブロックデータを格納した ICA を，読み取り専用で依頼元プロセスの仮想空間に貼り付ける．

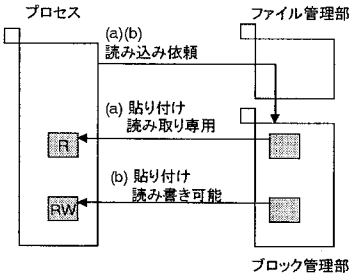

図 8　読み込み操作時の保護

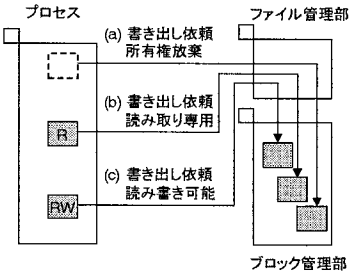

図 9　書き出し操作時の保護

(b)　読み書き可能 (RW)
ブロックデータを格納した ICA を，読み書き可能で依頼元プロセスの仮想空間に貼り付ける．
(2)　書き出し
(a)　所有権放棄 (放棄)
書き込みデータを格納した ICA をブロック管理部に渡す．書き込みデータを格納した ICA は，依頼元プロセスの仮想空間から剥がす．
(b)　読み取り専用 (R)
書き込みデータを格納した ICA をブロック管理部に渡す．書き込みデータを格納した ICA は，読み取り専用で依頼元の仮想空間に貼り付いたままとなる．
(c)　読み書き可能 (RW)
書き込みデータを格納した ICA をブロック管理部に渡す．書き込みデータを格納した ICA は，読み書き可能で依頼元の仮想空間に貼り付いたままとなる．

4.4.3　データ操作時の処理内容

複数のプロセスが同一のブロックに対して読み書きを行う場合のブロック管理の動作を**表 3** に示し，以降で説明する．
(A-1) ブロックデータがキャッシュに無いので，ディスクドライバに依頼し，ICA にブロックデータを読み込み，データをブロックキャッシュに登録する．読み込んだデータを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．読み込んだデータを格納した ICA の先頭アドレスを返却する．
(A-2) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA を

表 3　ブロックへのアクセス時の動作

操作		データ状態			
内容	保護	(1) ディスク上	(2) ブロック管理部が占有	ブロック管理部と他 AP が共有	
				(3)AP は読み取り専用	(4)AP は読み書き可能
読み込み	(A) R	ディスクから読み込み キャッシュに登録 読み取り専用で貼り付け	既存キャッシュを取得 読み取り専用で貼り付け	既存キャッシュを取得 読み取り専用で貼り付け	既存キャッシュを取得 読み取り専用で貼り付け
	(B) RW	ディスクから読み込み キャッシュに登録 読み取り専用で貼り付け	既存キャッシュを取得 読み書き可能で貼り付け	既存キャッシュを取得 読み書き可能で貼り付け	既存キャッシュを取得 読み書き可能で貼り付け
書き出し	(C) 放棄	データ ICA を剥がし キャッシュに登録	既存キャッシュを削除 データ ICA を剥がし キャッシュに格納	データ ICA を剥がし 書き込みデータを 既存キャッシュに上書き	データ ICA を剥がし 書き込みデータを 既存キャッシュに上書き
	(D) R	データ ICA を キャッシュに登録 読み取り専用で貼り付け	既存キャッシュを削除 書き込みデータを キャッシュに格納 読み取り専用で貼り付け	書き込みデータを 既存キャッシュに上書き 上書きされたデータを 読み取り専用で貼り付け	書き込みデータを 既存キャッシュに上書き 上書きされたデータを 読み取り専用で貼り付け
	(E) RW	データ ICA を キャッシュに登録 読み書き可能で貼り付け	既存キャッシュを削除 書き込みデータを キャッシュに格納 読み書き可能で貼り付け	書き込みデータを 既存キャッシュに上書き 上書きされたデータを 読み書き可能で貼り付け	書き込みデータを 既存キャッシュに上書き 上書きされたデータを 読み書き可能で貼り付け

ブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(A-3) ((A-2) と同様) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA をブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(A-4) ((A-2) と同様) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA をブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(B-1) ブロックデータがキャッシュに無いので，ディスクドライバに依頼し，ICA にブロックデータを読み込み，データをブロックキャッシュに登録する．読み込んだデータを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．読み込んだデータを格納した ICA の先頭アドレスを返却する．

(B-2) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA をブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(B-3) ((B-2) と同様) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA をブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(B-4) ((B-2) と同様) ブロックデータがブロックキャッシュに保存されているので，要求されたブロックを格納した ICA をブロックキャッシュから取得する．要求されたブロックを格納した ICA を，ブロック管理部の仮想空間からは剥がさずに，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．要求されたブロックを格納した ICA の先頭アドレスを返却する．

(C-1) 書き込みデータが格納された ICA を，依頼元プロセスの仮想空間から剥がし，ブロックキャッシュに登録する．返却値はなし．

(C-2) 書き込みデータが格納された ICA を，依頼元プロセスの仮想空間から剥がす．書き込みデータと同一のブロックがブロックキャッシュに保存されているので，書き込みデータと同一のブロックに関連するキャッシュデータを削除し，書き込みデータが格納された ICA をブロックキャッシュに登録する．返却値はなし．

(C-3) 書き込みデータが格納された ICA を，依頼元プロセスの仮想空間から剥がす．書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．返却値はなし．

(C-4) ((C-3) と同様) 書き込みデータが格納された ICA を，依頼元プロセスの仮想空間から剥がす．書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．返却値はなし．

(D-1) 書き込みデータが格納された ICA をブロックキャッシュに登録する．データ ICA を依頼元プロセスの仮想空間から剥がし，読み取り専用で貼りなおす．書き込みデータが格納された ICA の先頭アドレスを返却する．

(D-2) 書き込みデータと同一のブロックがブロックキャッシュに保存されているので，書き込みデータと同一のブロックに関連するキャッシュデータを削除し，書き込みデータが格納された ICA をブロックキャッシュに登録する．データ ICA を依頼元プロセスの仮想空間から剥がし，読み取り専用で貼りなおす．書き込みデータが格納された ICA の先頭アドレスを返却する．

(D-3) 書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．上書きされたデータを格納した ICA を，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．上書きされた ICA の先頭アドレスを返却する．

(D-4) ((C-3) と同様) 書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．上書きされたデータを格納した ICA を，依頼元プロセスの仮想空間に読み取り専用で貼り付ける．上書きされた ICA の先頭アドレスを返却する．

(E-1) 書き込みデータが格納された ICA をブロックキャッシュに登録する．データ ICA を依頼元プロセスの仮想空間から剥がし，読み書き可能で貼りなおす．書き込みデータが格納された ICA の先頭アドレスを返却する．

(E-2) 書き込みデータと同一のブロックがブロックキャッシュに保存されているので，書き込みデータと同一のブロックに関連するキャッシュデータを削除し，書き込みデータが格納された ICA をブロックキャッシュに登録する．データ ICA を依頼元プロセスの仮想空間から剥がし，読み書き可能で貼りなおす．書き込みデータが格納された ICA の先頭アドレスを返却する．

(E-3) 書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．上書きされたデータを格納した ICA を，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．上書きされた ICA の先頭アドレスを返却する．

(E-4) ((E-3) と同様) 書き込みデータと同一のブロックがブロックキャッシュに保存されており，該当ブロックデータが他プロセスと共有されているので，新たに受け取った書き込みデータ ICA の内容を，既存のキャッシュデータに上書きする．上書き後，受け取った書き込みデータ ICA は削除する．上書きされたデータを格納した ICA を，依頼元プロセスの仮想空間に読み書き可能で貼り付ける．上書きされた ICA の先頭アドレスを返却する．

AnT のファイル管理では，キャッシュデータをプロセスの仮想空間に直接貼り付けることによりファイルデータの授受を行う．これにより，ファイルデータの授受を高速化している．ただし，複数のプロセスから同一のブロックに対して読み書きを行う場合，同一のブロックに読み書きを行う全てのプロセスが同一のメモリ領域を参照することになるため，データの不整合がおきる場合がある．このため，排他制御を行う必要があると考えられる．

5. 関連研究

FreeBSD[5] の mmap() では，ファイルをメモリにマッピングして，ファイルの内容を複数プロセス間で共有することができる，mmap() では，マッピングしたプロセスがそれぞれファイルデータのキャッシュのコピーを持つ．このため，プロセスによるファイル内容の変更は，即座に他プロセスが持つデータに反映されるわけではない．ファイルデータの共有のためには，msync() などによりファイルとメモリの同期をとり，ファイルを介して別のプロセスに変更を反映する必要がある．ファイル内容の同期の契機は，msync() を呼び出すことにより明示的に指定することもできるが，多くは OS により自動的に同期される．***AnT*** では，複数のプロセスから同じファイルに対して入出力を行う場合，ファイルのキャッシュデータが格納されたメモリを共有する．このため，一方のプロセスで行った編集が，即座に他方のプロセスが持つデータに反映される．ここで，***AnT*** では，データの不整合を防ぐために排他制御が必要になると考えられる．

柔軟なデータの管理を行うことを可能とするファイルシステムとして，KFS[6] がある．KFS のブロック

データの格納方式には，以下の3つがある．(1) 1つのファイルを2つのディスクに複製して記録する．(2) 2つのディスクに対してラウンドロビンでブロックの内容を分散して記録する．(3) ブロックデータを圧縮して記録する．KFS では，これらの機能を組み合わせることにより，利便性を向上させている．***AnT*** では，(1)(2)(3) のような機能を持つブロック管理部を実現し，既存のブロック管理部と入れ替えることで，容易にこれらの機能を実現することが可能である．

6. おわりに

AnT におけるファイル管理サーバの設計について述べた．***AnT*** のファイル管理サーバは，ファイル入出力の高速化と論理ボリュームを超えるサイズのファイルの実現を目指す．ファイル入出力の高速化のために，サーバ間通信数削減とゼロコピーでのデータ授受を行う．サーバ間通信数削減は，***AnT*** のサーバ間通信の直接返却機能を利用し，ブロック管理部からのデータ返却を，ファイル管理部を経由せずに直接依頼元プロセスに返却することにより実現する．ゼロコピーでのデータ授受は，***AnT*** の ICA を用いたゼロコピー通信機能を利用し，ブロックデータのキャッシュを ICA を用いて管理し，ブロックデータの授受を ICA の貼り付け剥がしで行うことで実現する．また，論理ボリュームを超えるサイズのファイルは，ブロック管理部が複数の論理ボリュームを一括管理することにより実現する．i ノード情報から，他の論理ボリュームのブロックを参照することにより実現する．

残された課題として，ファイル管理サーバの実現と評価がある．

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) 谷口秀夫，乃村能成，田端利宏，安達俊光，野村裕佑，梅本昌典，仁科匡人：適応性と堅牢性をあわせもつ ***AnT*** オペレーティングシステム，情報処理学会研究会報告，Vol.2006-OS-103, pp.71–78 (2006).
2) 梅本昌典，田端利宏，乃村能成，谷口秀夫：AnT オペレーティングシステムにおけるメモリ領域管理の設計と実現，情報処理学会研究報告 2007-OS-104，第 104 回システムソフトウェアとオペレーティング・システム (OS) 研究会，pp.33–40 (2007).
3) 岡本幸大，谷口秀夫：***AnT*** におけるサーバ間の高速なプログラム間通信機構，マルチメディア通信と分散処理ワークショップ論文集，pp.61–66 (2007).
4) 岡本幸大，谷口秀夫：AnT オペレーティングシステムにおけるサーバプログラム間通信機構の評価，電子情報通信学会技術研究報告，Vol.107, No.558, 組み込み技術とネットワークに関するワークショップ，pp.49–54 (2008).
5) : The FreeBSD Project, http://www.freebsd.org/.
6) Dilma M. da Silva, Livio B. Soares and Orran Krieger: KFS: Exploring Flexibility in File System Design, Brazilian Workshop on Operating System WSO'2004 (2004).