Vol.2015-OS-132 No.7
2015/2/26

ファイル操作に着目したOS処理分散法

江原 寛人[1]　河上 裕太[1]　山内 利宏[1]　谷口 秀夫[1]

概要：ファイル操作処理は，OS 処理として実現される．また，マイクロカーネル構造 OS は，ファイル管理処理やディスクドライバ処理といった OS 処理を OS サーバとして実現する．したがって，マルチコアプロセッサ環境において，OS サーバをコア毎に分散することで，OS 処理をコア毎に分散できる．本稿では，マルチコア向け ***AnT*** において，ファイル操作処理に関する OS サーバを複数同時起動し，これら OS サーバをコア毎に分散できることを述べる．また，複数の外部記憶装置からの独立したファイル操作処理について，評価結果を報告する．

1. はじめに

サービス処理の実行において，ファイル操作処理は入出力処理を伴うため，処理時間に大きな影響を与える．このため，入出力処理，特に実 I/O 時間を短縮する工夫が多くなされており，代表的なものとして，入出力装置を複数用意して，並列実行により実 I/O 時間を短縮する技術がある [1].

また，マルチコアプロセッサが登場し，サービスを実現する処理を複数のコア（実行ユニット）へ分散させることにより，つまりプロセッサ負荷分散により，スループットの向上が可能になっている．多くの場合，応用プログラム（AP）の処理を分散する事例が多い．しかし，サービスを実現する処理は，AP 処理だけではなく OS 処理も含んでいる．このため，OS 処理も分散できれば，さらなるスループットの向上が可能である [2]-[6].

一方，マイクロカーネル構造 OS[7]-[9] は，OS 機能をプロセス（以降，OS サーバと呼ぶ）として実現する．つまり，ファイル操作処理を行う OS 機能（ファイル管理機能）についても，OS サーバとして実現する．

したがって，マイクロカーネル構造 OS をマルチコアプロセッサ環境で走行させ，ファイル管理機能を提供する OS サーバを分散することにより，サービス処理のスループットを向上できる．さらに，コア毎に入出力装置を用意できれば，実 I/O 処理を並列に行うことも可能である．

OS 処理の分散を可能にするマルチコア向けマイクロカーネル構造 OS としてマルチコア向け ***AnT*** オペレーティングシステム [10]（An operating system with adaptability and toughness）（以降，マルチコア ***AnT*** と呼ぶ）がある．マイクロカーネル構造 OS は，割り込み処理や例外処理といった最小限の OS 処理をカーネルとして実現し，その他のファイル管理処理やディスクドライバ処理といった OS 処理を OS サーバとして実現する．このため，OS サーバをコア毎に分散することで，OS 処理を分散できる．さらに，マルチコア ***AnT*** では，同等の OS 処理を実行する OS サーバを複数同時起動させることで高スループットを実現する OS 処理分散法 [11]-[12] を有する．しかし，OS 処理分散法は，ファイル操作処理に関する OS サーバであるファイル管理サーバやディスクドライバサーバでは実現されていない．

そこで，本稿では，OS 処理の分散を可能とするマルチコア ***AnT*** において，ファイル操作処理に関する OS サーバの OS 処理分散法を提案する．具体的には，ファイル操作処理に関する OS サーバを一組とし，これら OS サーバを組単位で複数同時起動し，組単位でコア毎に分散することで，複数の外部記憶装置からの独立したファイル操作処理を実現し，ファイル操作処理の負荷分散を可能にする．

2. マルチコア向け *AnT* オペレーティングシステム

2.1 基本構造

マルチコア ***AnT*** の基本構造を**図 1** に示す．OS は，カーネルと OS 機能を有するプロセス（以降，OS サーバと呼ぶ）からなる．カーネルは，BSP（Boot Strap Processor）上で動作し最初に起動するカーネルである m-カーネルと AP（Application Processor）上で動作し m-カーネルにより起動されるカーネルである p-カーネルの 2 種類から構成される．m-カーネルは，カーネルに必要な全機能を有す

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

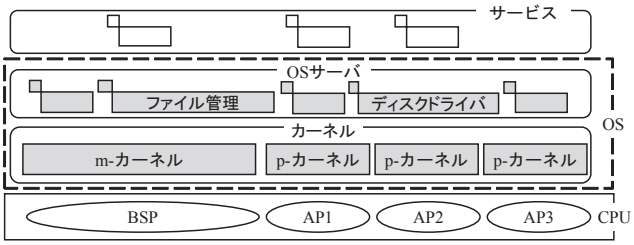


図 1 マルチコア ***AnT*** の基本構造

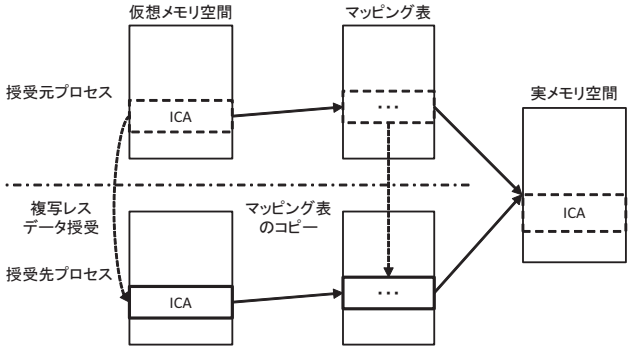


図 2 複写レスデータ授受の様子

る．一方，p-カーネルは，機能を例外・割り込み機能，サーバプログラム間通信機能，スケジューラ機能，およびコア間通信機能に絞り，カーネルの軽量化を図っている．OS サーバは，システム利用形態に適応するために必須なプログラム部分であり，ファイル管理機能やディスクドライバ機能をプロセスとして提供する．また，OS サーバは，実行する処理に対応した一意の識別子（以降，コア ID と呼ぶ）を有している．このコア ID を用いて，処理依頼先の OS サーバを識別する．サービスは，サービス提供するプログラム部分である．

マルチコア ***AnT*** は，サービスとして動作する応用プログラム（以降，AP と呼ぶ）のプロセスと OS サーバ間での通信に利用するデータ領域としてコア間通信データ域（ICA : Inter-core Communication Area）をもつ．ICA の特徴として，以下の三つがある．

(1) ページ（4 KBytes）を単位とし，n ページ分の領域の確保と開放
(2) 確保した領域（n ページ）の実メモリ連続の保証
(3) 2 仮想空間の間での領域の貼り替え

ICA は，カーネルによりページを最小単位として管理される領域であり，ICA へのアクセスは，プロセス毎の仮想空間のマッピング表を通して行われる．ここで，マッピング表への登録を貼り付けと呼び，マッピング表からの削除を剥がしと呼ぶ．プロセス間の複写レスでのデータ授受の様子を**図 2** に示す．ICA を利用したプロセス間でのデータ授受は，授受するデータを格納した ICA をデータ授受故プロセスの仮想空間から剥がし，データ授受先プロセスの仮想空間へ ICA を貼り付けることで行われる．これらの操作をまとめて ICA の貼り替えと呼ぶ．

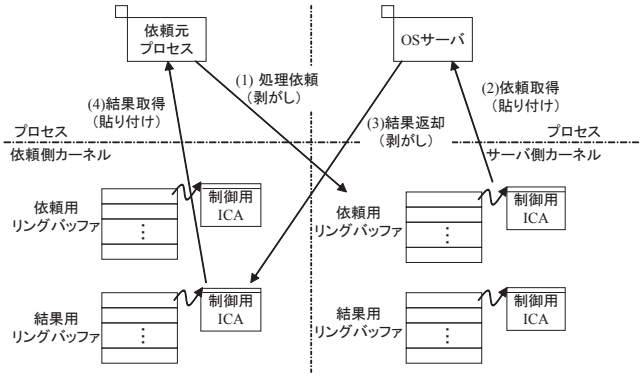


図 3 マルチコア ***AnT*** のサーバプログラム間通信機構の処理流れ

2.2 サーバプログラム間通信機構

AnT の有するサーバプログラム間通信機構 [13] は，ICA の貼り替えによる複写レスデータ授受機能を利用した高速な通信機構である．この機構は，制御用情報とデータ情報を ICA に格納し，サーバプログラム間でのデータ授受を複写レスで実現している．なお，制御用情報を格納する ICA を制御用 ICA，データ情報を格納する ICA をデータ用 ICA と名付けている．さらに，マルチコア ***AnT*** では，コア間でのサーバプログラム間通信機構を高速化している．具体的には，リングバッファを用いた制御機構により，排他制御を行わない制御用 ICA の貼り替えを実現している．また，ICA の授受方式を改善し，各プロセスの動作するコア上のカーネルがサーバプログラム間通信時における ICA の貼り替えを行っている．さらに，コア間でのサーバプログラム間通信機構を利用する場合，依頼元プロセスは，コア間での通信（以降，コア間通信）が発生する．マルチコア ***AnT*** でのサーバプログラム間通信機構の処理流れを**図 3** に示し，以下に説明する．

(1) 依頼元プロセスが処理依頼を行うと，依頼元プロセスの動作するコア上のカーネル（依頼側カーネル）は OS サーバの依頼用リングバッファに依頼情報を格納した制御用 ICA を登録し，依頼元プロセスから制御用 ICA を剥がす．また，WAIT 状態の OS サーバに依頼情報を格納した制御用 ICA を登録した場合は OS サーバを起床させる．
(2) OS サーバの動作するコア上のカーネル（サーバ側カーネル）は，OS サーバに制御用 ICA を貼り付ける．OS サーバは，依頼用リングバッファから依頼情報を格納した制御用 ICA を取得し，処理を実行する．
(3) OS サーバが結果返却を行うと，サーバ側カーネルは依頼元プロセスの結果用リングバッファに結果情報を格納した制御用 ICA を登録し，自身から結果情報を格納した制御用 ICA を剥がす．また，WAIT 状態の依頼元プロセスに結果情報を格納した制御用 ICA を登録した場合，依頼元プロセスを起床させる．

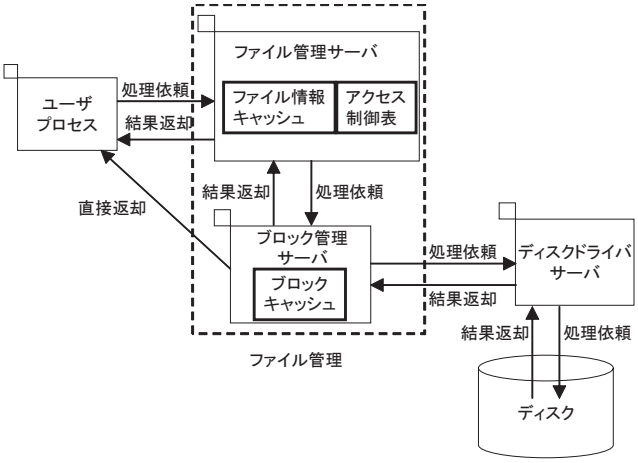


図 4 ファイル管理機能の構成

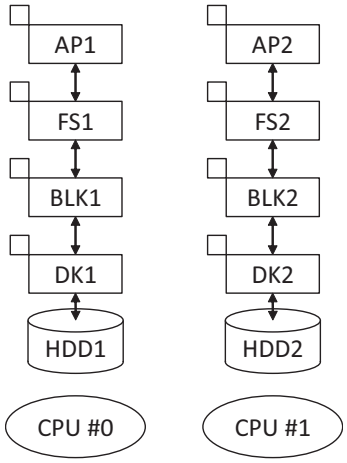


図 5 ファイル操作処理に関する OS サーバにおける OS 処理分散法

(4) 依頼側カーネルは，依頼元プロセスに制御用 ICA を貼り付ける．依頼元プロセスは，結果用リングバッファから結果情報を格納した制御用 ICA を取得し処理を終了する．

2.3 ファイル管理機能の構成

AnT のファイル管理機能 [14] の構成を**図 4** に示す．***AnT*** のファイル管理機能はファイル管理サーバ，ブロック管理サーバ，およびディスクドライバサーバの三種類の OS サーバで構成される．ファイル管理サーバは，外部記憶装置に構築されたファイルシステムの管理情報，およびファイルのアクセス情報と管理情報を管理する．ブロック管理サーバは，ファイルデータのキャッシュ（ファイルキャッシュ）をブロック単位で管理する．ディスクドライバサーバは，外部記憶装置への入出力処理を制御する．

AnT のファイル管理機能の特徴として，ブロック管理サーバの直接返却がある．例として，ファイルの読み込み処理の処理流れを示す．まず，AP プロセスは，ファイルの読み込み処理をファイル管理サーバへ依頼する．依頼を受け取ったファイル管理サーバは，ファイルデータの読み込みをブロック管理サーバへ依頼する．ブロック管理サーバは，ファイルデータがファイルキャッシュ上に存在する場合，ファイルデータを AP プロセスへ直接返却する．存在しない場合，ディスクドライバサーバへファイルデータの読み込みを依頼し，読み込んだファイルデータをファイルキャッシュに登録する．登録後，ファイルデータを AP プロセスへ直接返却する．これにより，サーバプログラム間通信の回数を一回削減している．

3. ファイル操作処理における OS 処理分散法

3.1 考え方

計算機に搭載された複数の外部記憶装置毎にファイル操作処理に関する OS サーバを起動することで，外部記憶装置を利用するサービスのファイル操作処理の負荷を分散する．ファイル操作処理に関する OS サーバにおける OS 処理分散法を**図 5** に示す．図 5 では，二つの AP プロセス（AP1，AP2）について，それぞれファイル管理サーバ（FS1，FS2），ブロック管理サーバ（BLK1，BLK2），およびディスクドライバサーバ（DK1，DK2）の三種類の OS サーバと外部記憶装置（HDD1，HDD2）を個別に割り当て，かつ CPU コア（CPU#0，CPU#1）毎に AP プロセスと OS サーバを分散させた構成である．図 5 の構成では，二つの AP プロセスは，個別に外部記憶装置を割り当てられており，かつそれぞれ別の CPU コア上で走行しているため，それぞれ独立して外部記憶装置へのファイル操作処理を実行できる．

3.2 課題と対処

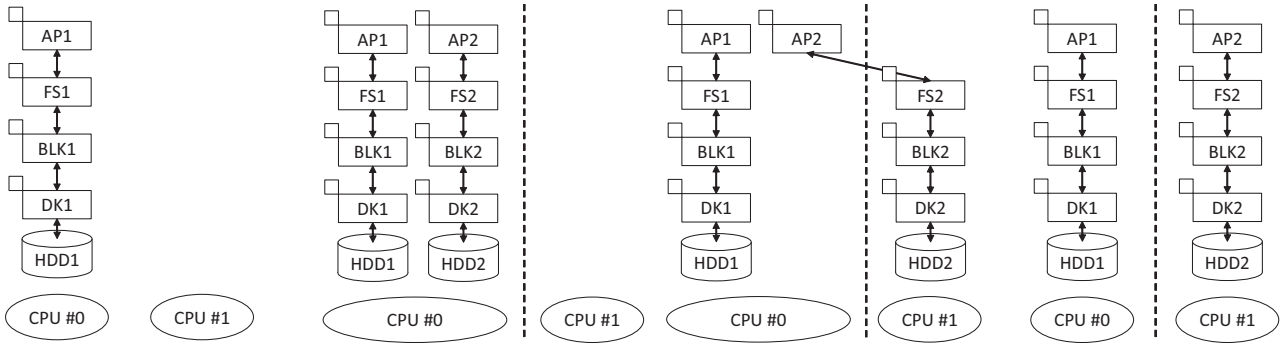
3.1 節で示したファイル操作処理における OS 処理分散法を実現するために，以下の課題がある．

(課題 1)ファイル操作処理に関する OS サーバを複数起動させる方式

(課題 2)ファイル操作処理に関する OS サーバ間を対応付ける方式

(課題 1) は，同等の OS 処理を実行する OS サーバを識別するために必要となる．これは，同等の OS 処理を実行する OS サーバは，同一のコア ID を有することに起因する．対処として，コア ID に，同等の OS 処理を実行する OS サーバが複数起動した場合，通番を格納する領域を用意する．これにより，ファイル操作処理に関する OS サーバを複数起動した場合，それぞれの OS サーバを個別に識別可能になる．

(課題 2) は，ファイル操作処理に関する OS サーバを複数起動した場合，ファイル操作処理に関する OS サーバ間で連携を実現するために，OS サーバ間で対応付けを行う必要がある．これは，***AnT*** の OS サーバは，静的に決定されたコア ID を用いて連携する OS サーバ間で対応付けられることに起因する．対処として，OS サーバの起動時に，連携する OS サーバの情報を与えることで，動的な対応付



(A) 分散形態 #0　(B) 分散形態 #1　(C) 分散形態 #2　(D) 分散形態 #3

図 6　測定プロセスと OS サーバの分散形態

けを実現する．具体的には，OS サーバの起動時に，コマンドライン引数に，連携する OS サーバのコア ID を指定する．これにより，OS サーバは指定されたコア ID を用いることで，連携する OS サーバを決定することが可能になる．

3.3　期待される効果

ファイル操作処理における OS 処理分散法を実現することで，計算機に搭載された複数の外部記憶装置毎にファイル操作処理に関する OS サーバを起動し，外部記憶装置の入出力処理を AP プロセス毎に割り当てることが可能となる．これにより，複数の AP プロセスに専用の外部記憶装置を割り当てることで，AP プロセスの提供するサービス処理の負荷を分散できる．また，外部記憶装置と AP プロセスをそれぞれコア毎に分散することで，各 AP プロセスのサービスを並列して実行可能となり，システム全体の負荷を分散できる．例えば，二台の外部記憶装置を計算機上に搭載し，一台の外部記憶装置をファイル転送のサービスに割り当て，もう一台の外部記憶装置をシステムのログを記録するサービスに割り当てる．これにより，二つのサービスは，独立してファイル操作処理を実行可能となり，各サービスの処理性能の低下を抑制できる．

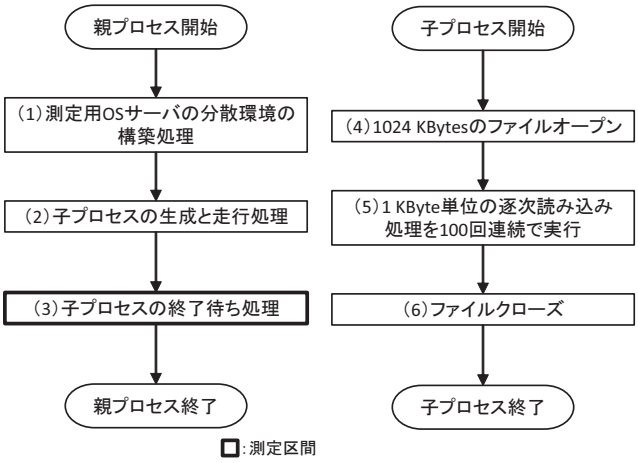
4.　評価

4.1　観点と評価環境

本評価では，3 章で示したファイル操作処理における OS 処理分散法について，ファイルの読み込み処理の分散効果を評価する．具体的には，二台の外部記憶装置に存在する別々のファイルを，二つの AP プロセスが同時に読み込み処理を実行した場合の処理時間を測定する．

本評価の測定で用いる測定プロセス（AP1，AP2），ファイル操作処理に関する OS サーバ（FS1，FS2，BLK1，BLK2，DK1，DK2），および外部記憶装置（HDD1，HDD2）の分散形態を**図 6** に示す．

(A) 分散形態 #0 は，一つの測定プロセス（AP1），ファイル操作処理に関する OS サーバ（FS1，BLK1，DK1）の各種一つずつ，および外部記憶装置（HDD1）一台の構成である．これは，他の分散形態を比較する際の基準として用いるための分散形態である．



(1) 親プロセスの処理流れ　(2) 子プロセスの処理流れ

図 7　測定プロセスの処理流れと測定区間

(B) 分散形態 #1 は，二台の外部記憶装置を用いて入出力処理のみ分散し，コア毎の分散は行わない分散形態である．

(C) 分散形態 #2 は，分散形態 #1 からファイル操作処理に関する OS サーバをコア毎に分散し，測定プロセスはコア毎に分散しない分散形態である．

(D) 分散形態 #3 は，ファイル操作処理に関する OS サーバと測定プロセスの両方をコア毎に分散する分散形態である．

これら四種類の分散形態について，それぞれの測定結果を比較することで，OS 処理の分散効果を明らかにする．

測定は，測定用の親プロセス（以降，親プロセスと呼ぶ）と親プロセスによって起動される測定用 OS サーバと測定用の子プロセス（以降，子プロセスと呼ぶ）によって行う．プロセスの優先度は，親プロセスの優先度を最も高い値に設定し，次いでディスクドライバサーバ，ブロック管理サーバ，ファイル管理サーバ，および子プロセスの順で優先度の値を設定する．親プロセスは m-カーネルによって起動され，CPU#0 上で走行を開始する．親プロセスと子プロセスが走行を開始してから終了するまでの処理流れを

図 7 に示す．以下に，親プロセスの処理流れを説明する．
(1) 測定用 OS サーバの分散環境の構築処理
親プロセスは，図 6 で示した四種類の分散形態の走行環境を構築するため，ファイル操作処理に関する OS サーバであるファイル管理サーバ（FS1，FS2），ブロック管理サーバ（BLK1，BLK2），およびディスクドライバサーバ（DK1，DK2）を CPU#0 上で最大二つずつ生成し，それぞれ CPU#0 上で走行させる．ファイル操作処理に関する OS サーバの走行後，図 6 の分散形態 #2 と分散形態 #3 の測定の場合，親プロセスは，ファイル管理サーバ（FS2），ブロック管理サーバ（BLK2），およびディスクドライバサーバ（DK2）を CPU#1 に移譲する．
(2) 子プロセスの生成と走行処理
ファイルの読み込み処理を実行する子プロセスを CPU#0 上で最大二つ連続に生成し，生成したすべての子プロセスの pid を記録する．この pid は，子プロセスの終了待ちの処理で使用する．子プロセスの生成後，生成した子プロセスを CPU#0 上走行させる．子プロセスの走行後，図 6 の分散形態 #3 の測定の場合，親プロセスは，子プロセス（AP2）を CPU#1 に移譲する．その後，(3) の子プロセスの終了待ちに移行する．
(3) 子プロセスの終了待ち処理
親プロセスは，(2) の処理で走行したすべての子プロセスの終了を待つ処理を実行する．具体的には，(2) の子プロセスの生成時に取得した pid を用いて，プロセスの終了待ちシステムコールを生成した子プロセス数だけ連続して発行する．すべての子プロセスの終了待ちを完了後，親プロセスの処理を終了する．
次に，子プロセスの処理流れを説明する．
(4) 1024 KBytes のファイルオープン
子プロセスは，外部記憶装置に格納された 1024KBytes のファイルオープンを実行する．具体的には，子プロセス（AP1）はファイル管理サーバ（FS1）に，子プロセス（AP2）はファイル管理サーバ（FS2）にそれぞれファイルオープンの処理依頼を実行する．
(5) 1 KByte 単位の逐次読み込み処理を 100 回連続で実行
子プロセスは，(3) でオープンしたファイルの先頭から 1 KByte 単位で逐次読み込み処理を 100 回連続で実行する．具体的には，子プロセス（AP1）はファイル管理サーバ（FS1）に，子プロセス（AP2）はファイル管理サーバ（FS2）にそれぞれファイル読み込み処理の処理依頼を 100 回連続で実行する．
(6) ファイルクローズ
子プロセスは，(4) の処理の終了後，(3) の処理でオープンしたファイルをクローズする．具体的には，子プロセス（AP1）はファイル管理サーバ（FS1）に，子プロセス（AP2）はファイル管理サーバ（FS2）にそれぞれファイルクローズの処理依頼を実行する．ファイルクローズの処理依頼実行後，プロセスの処理を終了する．

表 1 評価環境

CPU	Intel (R) Core (TM) i7 2600 (3.4GHz) 4 コア
メモリ	8 GBytes
HDD	250 GBytes 7200 rpm Seagate ST3250312AS 二台

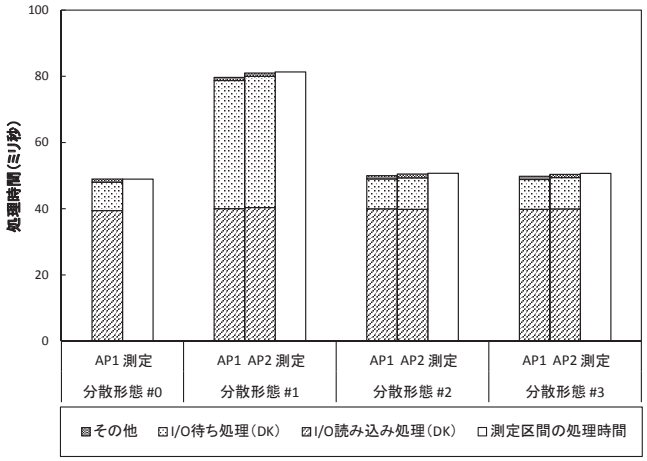


図 8 子プロセス毎の処理時間と測定区間の処理時間の比較

上記の測定プロセスの処理流れのうち，測定区間は，(3) の親プロセスのすべての子プロセス終了待ち処理の開始直前からすべての子プロセスの終了待ちの終了直後までの区間である．
評価環境を**表 1** に示す．マルチコア ***AnT*** を物理コア数が 4 コアである Intel Core i7-2600（3.4 GHz）のプロセッサ，および回転数 7200 rpm，バッファキャッシュ 8 MBytes の外部記憶装置（HDD）を二台搭載した計算機上で動作させ評価した．

4.2 結果と考察

4.1 節の図 7 で示した測定を図 6 の四種類の分散形態について測定した結果を**図 8** と**表 2** に示す．図 8 と表 2 では，図 6 の四種類の分散形態について，子プロセス（AP1，AP2）の処理内容毎の処理時間の合計と図 7 の測定区間の処理時間を示す．子プロセスの処理内容を，以下の九種類の処理に分類する．
(1) PU 処理 (AP)
(2) サーバ間通信処理（AP）
(3) PU 処理（FS）
(4) サーバ間通信処理（FS）
(5) PU 処理（BLK）
(6) サーバ間通信処理（BLK）
(7) PU 処理（DK）
(8) I/O 待ち処理（DK）
(9) I/O 読み込み処理（DK）
これらの処理内容のうち，PU 処理は，子プロセス（AP1，AP2）と OS サーバの固有の処理である．例えば，PU 処

表 2　子プロセス毎の処理内容の処理時間と測定区間の処理時間（ミリ秒）

通番	処理内容	分散形態 #0	分散形態 #1		分散形態 #2		分散形態 #3	
		AP1	AP1	AP2	AP1	AP2	AP1	AP2
(a)	PU 処理 (AP)	0.02	0.02	0.02	0.02	0.02	0.02	0.02
(b)	サーバ間通信処理（AP）	0.25	0.25	0.25	0.25	0.43	0.25	0.25
(c)	PU 処理（FS）	0.04	0.04	0.04	0.04	0.04	0.04	0.04
(d)	サーバ間通信処理（FS）	0.04	0.04	0.04	0.04	0.04	0.04	0.04
(e)	PU 処理（BLK）	0.03	0.03	0.03	0.03	0.03	0.03	0.03
(f)	サーバ間通信処理（BLK）	0.07	0.07	0.07	0.07	0.07	0.07	0.07
(g)	PU 処理（DK）	0.51	0.51	0.52	0.53	0.52	0.51	0.52
(h)	I/O 待ち処理（DK）	8.58	38.71	39.66	9.03	9.50	8.98	9.48
(i)	I/O 読み込み処理（DK）	39.45	40.02	40.36	39.98	39.84	39.85	39.94
子プロセス毎の処理内容合計		48.99	79.69	80.99	49.99	50.49	49.79	50.39
測定区間の処理時間		49.10	81.33		50.71		50.68	

理（FS）は，ファイルオープン処理，ファイル読み込み処理，およびファイルクローズ処理である．サーバ間通信処理は，子プロセスと OS サーバ間とのサーバプログラム間通信機構の処理である．I/O 待ち処理（DK）は，ディスクドライバサーバが外部記憶装置へデータの読み込み処理の I/O 命令を発行した直後から，外部記憶装置からの割り込みを受け取り，割り込み処理を実行開始するまでの処理である．I/O 読み込み処理（DK）は，I/O 待ち処理（DK）後，I/O 命令を使用し，外部記憶装置からデータを読み込み処理である．***AnT*** では，外部記憶装置からのデータの読み込み処理をディスクドライバサーバの割り込みハンドラ内で実行する．また，外部記憶装置からの一回の割り込みで読み込むデータの単位は 1 セクタ（512 Bytes）である．

図 8 と表 2 より，分散形態 #1 について以下のことが分かる．

(1) 測定区間の処理時間について，四種類の分散形態のうち分散形態 #1 の処理時間は最も大きい．具体的には，分散形態 #1 の処理時間は，約 81.33 ミリ秒であり，他の分散形態と比較して，最大約 32.23 ミリ秒大きい．これは，二つの子プロセスとすべての OS サーバが同一コア上で走行しているため，二台の外部記憶装置からの I/O 読み込み処理（DK）がすべて逐次実行されるためである．分散形態 #1 の処理内容毎の測定結果より，子プロセス（AP1, AP2）の I/O 読み込み処理（DK）の処理時間は，それぞれ約 40.02 ミリ秒と約 40.36 ミリ秒であり，合わせて約 80.38 ミリ秒である．よって，分散形態 #1 の測定区間の処理時間のうち，I/O 読み込み処理（DK）は約 98.8% であるため，分散形態 #1 の測定区間の処理時間は妥当であるといえる．

(2) 子プロセス毎の処理内容の I/O 待ち処理（DK）の処理時間について，四種類の分散形態のうち分散形態 #1 の処理時間は最も大きい．具体的には，分散形態 #1 の処理時間は，約 38.71 ミリ秒と約 39.66 ミリ秒であり，他の分散形態と比較して，最大約 31.08 ミリ秒大きい．これは，***AnT*** では，I/O 読み込み処理（DK）をディスクドライバサーバの割り込みハンドラ内で実行するためである．分散形態 #1 では，二台の外部記憶装置毎にディスクドライバサーバを割り当てているため，それぞれ個別に割り込み処理を実行できる．しかし，I/O 読み込み処理（DK）の処理時間が I/O 待ち処理（DK）の処理時間より大きい場合，外部記憶装置から割り込み通知を受け取っても割り込み処理を開始できない．すなわち，I/O 待ち処理（DK）が終了できず，I/O 読み込み処理（DK）だけ I/O 待ち処理（DK）の処理時間が増加することになる．

また，図 8 と表 2 より，分散形態 #2 と分散形態 #3 について以下のことが分かる．

(1) 測定区間の処理時間について，分散形態 #2 と分散形態 #3 は分散形態 #1 と比較して処理時間が小さい．具体的には，分散形態 #2 の処理時間は約 50.71 ミリ秒，分散形態 #3 の処理時間は約 50.68 ミリ秒であり，分散形態 #1 の処理時間である約 81.33 ミリ秒よりそれぞれ約 30.62 ミリ秒，約 30.65 ミリ秒だけ短い．これは，分散形態 #2 と分散形態 #3 の構成において，ファイル操作処理に関する OS サーバをコア毎に分散することで，I/O 待ち処理（DK）と I/O 読み込み処理（DK）をコア毎で独立して実行できるためである．

(2) 測定区間の処理時間について，分散形態 #2 と分散形態 #3 の処理時間は，子プロセスの数が一つである分散形態 #0 の処理時間に近い．具体的には，分散形態 #0 の処理時間は約 49.1 ミリ秒であり，分散形態 #2 と分散形態 #3 は，それぞれ約 1.61 ミリ秒，約 1.58 ミリ秒であり，差はわずかであるといえる．

上記の (1)(2) より，分散形態 #2 と分散形態 #3 は，それぞれファイル読み込み処理の並列化を実現しているといえる．

5. おわりに

マルチコア向け ***AnT*** オペレーティングシステムにおい

て，ファイル操作処理に関する OS サーバを分散し，評価結果を述べた．

ファイル操作処理に関する OS サーバを複数起動する方式として，ファイル操作処理に関する OS サーバが複数機能した場合，OS サーバを識別するコア ID の通番を格納する領域に OS サーバの起動順に通番を格納するように変更し，同等の OS 処理を実行する OS サーバの識別を実現した．また，ファイル操作処理に関する OS サーバ間を対応付ける方式として，OS サーバの起動時にコマンドライン引数に，連携する OS サーバのコア ID を指定することで，動的な OS サーバ間の対応付けを実現した．

評価では，二台の外部記憶装置を搭載した計算機において，外部記憶装置毎にファイル操作処理に関する OS サーバ起動し，コア毎に分散し走行することで，すべての OS サーバを同一コア上で走行する場合より，ファイル読み込み処理の時間が短くなることを示した．具体的には，ファイル操作処理に関する OS サーバをコア毎に分散した場合の処理時間は，すべての OS サーバを同一コア上で走行する場合の処理時間と比較して約 62.3% まで短縮できる．

残された課題として，ファイルの書き込み処理の分散効果の評価，および実サービスにおけるファイル操作処理の負荷分散効果の評価がある．

謝辞 本研究の一部は，科学研究費補助金基盤研究（B）（課題番号：24300008）による．

参考文献

[1] Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H., Patterson, D.A.: RAID: high-performance, reliable secondary storage, *ACM Computing Surveys*, vol.26, no.2, pp.145-185, (1994).

[2] Wickizer, S.B., Chen, H., Chen, R., Mao, Y., Kaashoek, F., Morris, R., Pesterev, A., Wu, L.S.M., Dai, Y., Zhang, Y., and Zhang, Z.: Corey: An Operating System for Many Cores, *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation*, pp.43-57, (2008).

[3] Ballesteros, F.J., Evans, N., Forsyth, C., Guardiola, G., McKie, J., Minnich, R., and Salvador, E.S.: NIX: A Case for a Manycore System for Cloud Computing, *Bell Labs Technical Journal*, vol.17, no.2, pp.41-54, (2012).

[4] Matarneh, R.: Multi Microkernel Operating Systems for Multi-Core Processors, *Journal of Computer Science*, vol.5,no.7, pp.493-500, (2009).

[5] Wentzlaff, D., and Agarwal, A.: Factored Operating Systems (fos): The Case for a Scalable Operating System for Multicores, *ACM SIGOPS Operating Systems Review*, vol.43, no.2, pp.76-85, (2009).

[6] Nightingale, E.B., Hodson, O., McIlroy, R., Hawblitzel, C., and Hunt, G.: Helios: Heterogeneous Multiprocessing with Satellite Kernels, *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, pp.221-234, (2009).

[7] Liedtke, J.: Toward real microkernels, *Communications of the ACM*, Vol.39, No.9, pp.70-77 (1996).

[8] Tanenbaum, A.S., Herder, J.N., and Bos, H.: Can we make operating systems reliable and secure?, *IEEE Computer Magazine*, Vol.39, No.5, pp.44-51 (2006).

[9] Black, D.L., Golub, D.B., Julin, D.P., Rashid, R.F., Draves, R.P., Dean, R.W., Forin, A., Barrera, J., Tokuda,H., Malan, G., and Bohman, D.: Microkernel operating system architecture and mach, *Journal of Information Processing*, Vol.14, No.4, pp.442-453 (1992).

[10] 井上 喜弘, 佐古田 健志, 谷口 秀夫 : マルチコアプロセッサ上での負荷分散を可能にする ***AnT*** オペレーティングシステムの開発，情報処理学会研究報告，vol.2012-DPS-150, no.37，電子媒体 (2012.03).

[11] 佐古田 健志，山内 利宏，谷口 秀夫 : 高スループットを実現する OS 処理分散法の実現，マルチメディア，分散，協調とモバイル (DICOMO2013) シンポジウム論文集，vol.2013，no.2，pp.1663-1670 (2013).

[12] 佐古田 健志，山内 利宏，谷口 秀夫 : OS 処理の分散を可能にするマルチコア向けマイクロカーネル構造 OS の評価，電子通信学会技術研究報告，vol.IEICE-113，no.497，pp.283-288 (2014).

[13] 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高速なサーバプログラム間通信機構の実現と評価，電子情報通信学会論文誌 (D)，Vol.J93-D，No.10，pp.1977-1989 (2010).

[14] 野村 裕佑, 谷口 秀夫 : ***AnT*** におけるファイル管理サーバの設計，情報処理学会研究報告，Vol.2008-OS-109，no.77，pp.53-60 (2008).