Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

AnT オペレーティングシステムにおける マルチコア環境での負荷分散評価

佐古田 健志[1]　枡田 圭祐[1]　蓮岡 宜明[1]　山内 利宏[1]　谷口 秀夫[1]

概要：マイクロカーネル構造 OS は，高い適応性と堅牢性をあわせ持つ．また，LSI 技術の進歩によりマルチコアプロセッサが登場した．そこで，このマルチコアプロセッサ上で，マイクロカーネル構造 OS を走行させ，OS サーバを各コアに分散できれば，OS 処理の負荷分散が可能になる．これにより，OS 処理の負荷が大きいトランザクション処理をマルチコア環境で有効に提供できる．本稿では，マルチコア環境で動作するマイクロカーネル構造の ***AnT*** オペレーティングシステムにおいて，OS 処理を分散させたときの評価結果を報告する．具体的には，基本性能の評価として，ブロック単位でのデータ参照時の分散形態による性能の違いを報告する．また，PostMark と Bonnie を用いて，分散形態による分散効果を報告する．

Evaluation of Load Balancing in Multi-core for *AnT*

Takeshi SAKODA[1]　Keisuke MASUDA[1]　Noriaki HASUOKA[1]　Toshihiro YAMAUCHI[1]
Hideo TANIGUCHI[1]

Abstract: Operating system based on microkernel architecture has high adaptability and toughness. Otherwise, multi-core processor appeared by progress of LSI technology. If it is made to run OS based on microkernel architecture and OS server can be distributed to each core on the multi-core processor, load balancing of OS processing will become possible. For the above reasons, the load balancing of OS processing can offer large transaction processing effectively on the multi-core environment. This paper shows the result of evaluations for performance of distributing OS processing in the ***AnT*** operating system based on microkernel architecture works on the multi-core environment. To put it concretely, this paper shows the difference in the performance by distributed forms at the time of data reference in a block unit. Moreover, this paper shows the distribution effect by distributed forms using PostMark and Bonnie.

1. はじめに

マイクロカーネル構造 [1]~[4] は，スケジューラなどのオペレーティングシステム（OS）の基本機能をカーネルとして実現し，ファイル管理機能や種々のドライバ機能をプロセス（以降，OS サーバと呼ぶ）として実現する OS 構造である．このため，サービスシステムに適した OS サーバを構成しやすく，また不具合が起こった OS サーバを停止や交換することができる．つまり，高い適応性と堅牢性をあわせ持つ OS 構造である．さらに，OS サーバを分散して動作させることにより，OS 処理を分散できる分散処理システムの構築が可能である．

一方，LSI 技術の進歩により，一つのプロセッサ内に複数の命令実行部を有するマルチコアプロセッサ [5] が登場した．そこで，このマルチコア環境において，マイクロカーネル構造 OS を走行させ，OS サーバを各コアに分散できれば，OS 処理の負荷分散が可能になる．これにより，例えば，OS 処理の負荷が大きいトランザクション処理をマルチコア環境で実行する場合，複数のプロセスがトランザクション処理を利用しても高スループットを実現できる．

ここでは，マルチコア環境で動作するマイクロカーネル構造の ***AnT*** オペレーティングシステム [6] について，OS 処理を分散させたときの評価結果を報告する．具体的には，基本性能の評価としてブロック単位でのデータ参照時の分

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

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

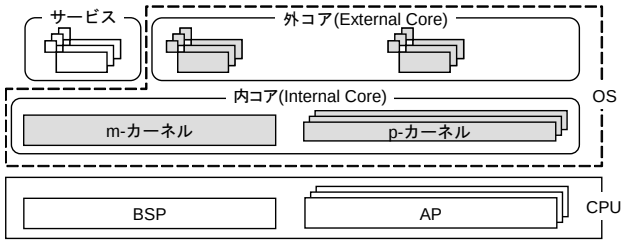


図 1 マルチコア ***AnT*** の基本構造

散形態による性能の違いを報告する．また，ベンチマークプログラムである PostMark と Bonnie を用いて，分散形態による分散効果を報告する．

2. マルチコア環境用 *AnT* オペレーティングシステム

2.1 基本構造

マルチコア環境用 ***AnT*** オペレーティングシステム（以降，マルチコア ***AnT*** と略す) の基本構造を**図 1** に示す．OS は，内コアとプロセス (OS サーバ) として動作する外コアからなる．内コアには，m-カーネル（BSP（Boot Strap Processor）上で動作し最初に起動するカーネル）と p-カーネル（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）がある．m-カーネルは，カーネルに必要な全機能を有する．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，およびスケジューラ機能に絞り，軽量化を図っている．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，例えば，ファイル管理機能やディスクドライバを OS サーバとして提供する．

AnT の仮想空間の構成は，多重仮想記憶である．さらに，特徴的な領域として，コア間通信データ域 (以降, ICA : Inter-core Communication Area）がある．この領域は，内コア，外コア，およびサービスが相互のデータ通信に利用する空間である．

2.2 サーバプログラム間通信機構

AnT は，高速なサーバプログラム間通信機構 [7] を有している．さらに，マルチコア ***AnT*** では，マルチコアでのプログラム間通信機構を高速化している．この様子を**図 2** に示す．具体的には，OS サーバ間の通信相手を制限し，通信時のキュー操作を 2 排他制御にしている．さらに，リングバッファを用いた制御機構により，排他制御を行なわない制御用 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 の貼り替えを行なっている．

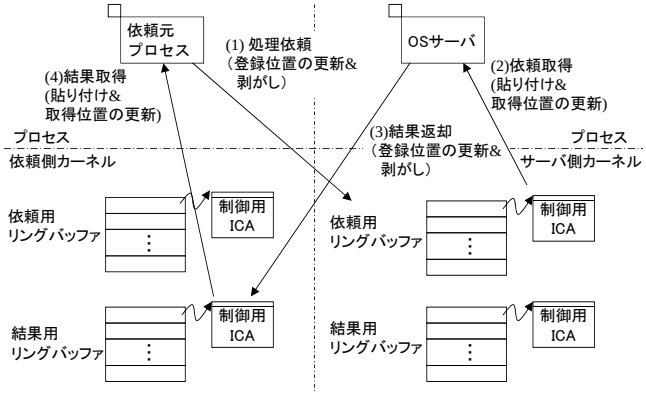


図 2 マルチコア ***AnT*** のサーバプログラム間通信機構の処理流れ

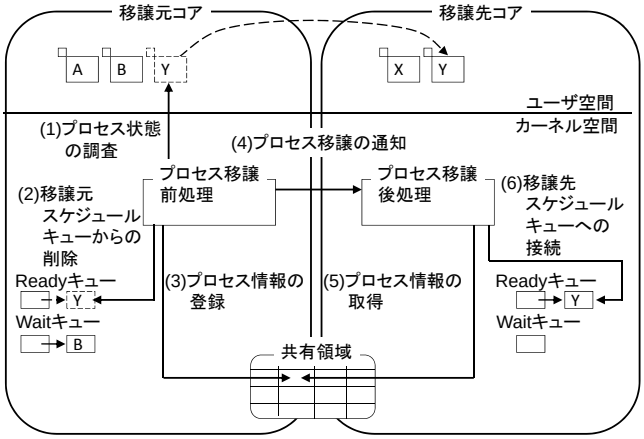


図 3 プロセス移譲機能の処理流れ

2.3 処理分散機能

マルチコア ***AnT*** では，マルチコア環境に適したプロセス移動（プロセスマイグレーション）機能を実現し，プロセス移譲機能と名づけている [8]．プロセス移譲機能は，移譲を実行するコア（以降，移譲元コアと略す）上で走行しているプロセスを任意のコア（以降，移譲先コアと略す）に移譲する機能である．プロセス移譲機能の処理流れを**図 3** に示し，以下に説明する．

（1）移譲可能なプロセス状態の判定を行なう．

（2）移譲対象プロセスが RUN 状態の場合，プロセスの状態を READY 状態に変更し，プロセスのコンテキストを保存する．また，移譲対象プロセスが READY 状態，または WAIT 状態の場合，それぞれのスケジュールキューからプロセスを削除する．

（3）移譲対象プロセスのプロセス情報を共有領域へ書き込む．

（4）IPI を用いて移譲先コアへプロセス移譲を通知する．

（5）移譲対象プロセスのプロセス情報を共有領域から読み込む．

（6）(2) の処理により，すべてのプロセスの状態は READY

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

表 1 プロセス分散形態の一覧

番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BSP	A,F,B,D	A,F,B	A,F,D	A,B,D	A,F	A,B	A,D	A	A,F	A,B	A,D	A	A	A	A
AP1		D	B	F	B,D	F,D	F,B	F,B,D	B	F	F	F,B	F,D	B,D	F
AP2									D	D	B	D	B	F	B
AP3															D

※ A：ベンチマークプロセス，F：ファイル管理サーバ，B：ブロック管理サーバ，D：ディスクドライバサーバ

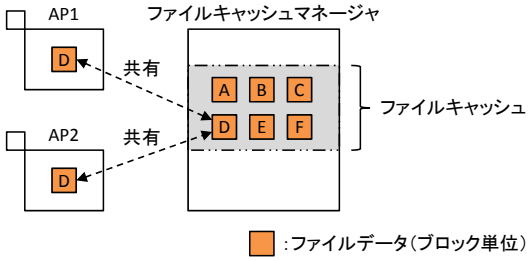


図 4 オンメモリファイル機能

状態か WAIT 状態のどちらかに変更されているため，該当するスケジュールキューへプロセスを接続する．

また，分散したプロセスが別コアの OS 処理を利用する方式として，m-カーネルへの処理依頼と m-カーネル専用機能の共有化を実現し，要求処理速度に応じて適用している．

m-カーネルへの処理依頼は，p-カーネル上のプロセスが発行したカーネルコール処理を m-カーネルに処理依頼するものである．

m-カーネル専用機能の共有化は，p-カーネルからも m-カーネル専用機能を利用可能にするものであり，共有化するにあたり，m-カーネル専用機能に排他制御を用いる．

2.4 オンメモリファイル機能

オンメモリファイル（OMF）機能 [9] は，以下の考え方に基づいている．

(1) プロセスとファイルキャッシュを共有する．
(2) ファイルサイズを拡縮できる．
(3) ファイルの参照日時や更新日時を更新できる．

OMF 機能の基本方式を**図 4** に示す．OMF 機能は，プロセスとファイルキャッシュを共有する．つまり，プロセスのファイルデータの参照や更新は，ファイルデータを格納する実メモリをプロセスの仮想メモリ空間へマッピングして行なう．したがって，通常入出力機能と異なり，ファイルキャッシュとプロセスの仮想メモリ空間の間でデータの複写は発生しない．また，OMF 機能は，ファイルデータをブロック（4KBytes）単位で管理する．

また，OMF 機能は，メモリマップドファイル機能 [10] と異なり，ファイルサイズを拡縮できる．具体的には，ファイルサイズを拡縮するために，ファイル情報を管理する i ノード内のファイルサイズやファイルデータの外部記憶装置上の格納位置を更新する．さらに，OMF 機能は，ファイルの参照日時や更新日時を更新できる．参照日時や更新日時を更新するために，i ノード内の参照日時や更新日時を更新する．これらの，i ノードの更新は，ファイルデータを外部記憶装置へ書き出す際に行なう．

3. 評価

3.1 観点と評価環境

トランザクション処理は，その処理の大半を OS 処理が占めるため，高スループットを得るには OS の入出力処理の負荷分散が効果的である．ここでは，ファイル入出力処理に着目し，その負荷分散の効果について明らかにする．

マイクロカーネル構造 OS では，多くの OS サーバが同じコアで動作する．このため，OS サーバに加えベンチマークプログラムのプロセス（以降，ベンチマークプロセスと略す）も含めたプロセス優先度の違いが性能に影響を与える．この点を明らかにするため，ファイル入出力について基本性能を評価する．

次に，OS 処理を分散させた場合，コア間通信を含む OS サーバ間通信のオーバヘッドが問題になる．そこで，このオーバヘッドを明らかにするため，PostMark[11] ベンチマークプログラムを用いて評価する．PostMark は，1 つのプロセスが複数のファイルに対してファイルのオープン，シーケンシャルなデータ参照，およびクローズを指定した回数実行するベンチマークプログラムである．また，OS 処理の負荷分散の有効性を明らかにするため，Bonnie[12] ベンチマークプログラムを用いて評価する．Bonnie は，複数のプロセスが 1 つのファイルデータのランダムな位置を参照し，任意の確率で更新するベンチマークプログラムである．

さらに，上記二つのベンチマークプログラム ***AnT*** と FreeBSD で実行した結果を比較し，OS 構造の違いによる影響を明らかにする．

OS サーバとベンチマークプロセスも含めたプロセス分散形態を**表 1** に示す．ファイル入出力に関連する OS サーバは，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 3 つである．ファイル管理サーバは，i ノードを管理している．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）を管理しており，実入出力回数を削減する．ディスクドライバ

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

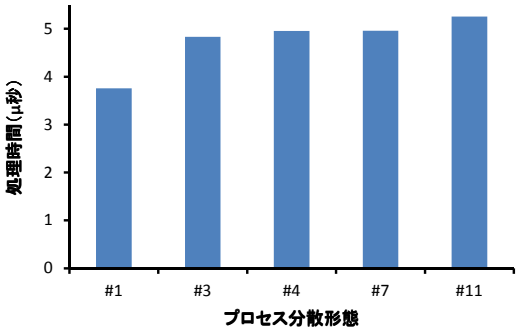


図 5 基本性能（1 ブロックのデータ参照）

サーバは，外部記憶装置を制御しており，ファイルデータを入出力する．表 1 において，#1 はプロセスを分散しない場合，#2〜#8 は 2 コアへプロセスを分散した場合，#9〜#14 は 3 コアへプロセスを分散した場合，#15 は 4 コアへプロセスを分散した場合である．

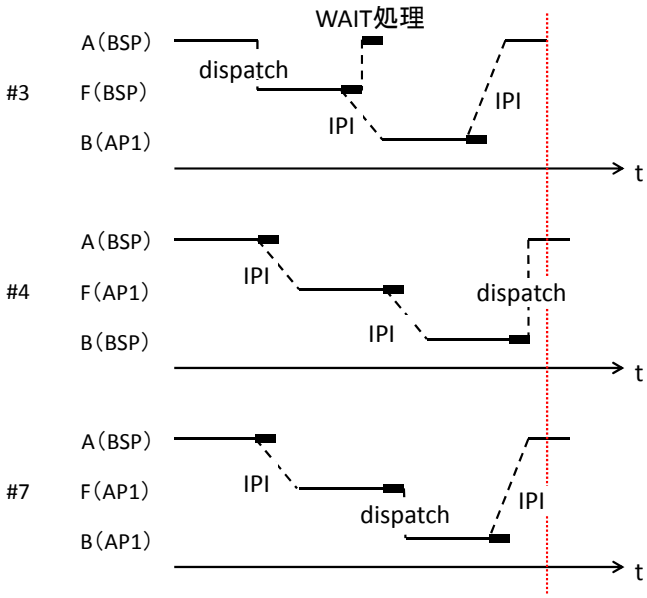
AnT を Core i7-2600（3.4GHz）の計算機で動作させ，表 1 の各場合について評価した．なお，プロセス優先度は，ディスクドライバサーバ，ファイル管理サーバとブロック管理サーバ，およびベンチマークプロセスの順で高優先度に設定した．ファイル管理サーバとブロック管理サーバは同じ優先度である．

3.2 基本性能

プロセスが 1 ブロック（4KBytes）のファイルデータを参照する処理時間を**図 5** に示す．この処理時間は，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の時間である．このため，ファイルキャッシュからの読み込み処理として，ベンチマークプロセスがファイル管理サーバへ処理依頼し，ファイル管理サーバがブロック管理サーバへ処理依頼し，ブロック管理サーバがファイルキャッシュからデータを取得し，ベンチマークプロセスへ結果を返却する．つまり，ディスクドライバサーバは動作しない．したがって，図 5 は，ファイル管理サーバとブロック管理サーバの分散形態に着目したものである．図 5 より，以下のことがわかる．

（1）OS 処理を 1 コア（#1），2 コア（#3，#4，および#7），3 コア（#11）と分散するに伴い，処理時間が増加する．これは，OS 処理の分散により，コア間通信回数が増加するためである．

（2）OS 処理を 2 コアへ分散した分散形態（#3，#4，および#7）は，コア間通信が同じ回数（2 回）であるため，同程度の処理時間である．しかし，ブロック管理サーバのみを分散した場合（#3），他の分散形態（#4，#7）に比べ，処理時間がわずかに短い．これは，サーバ間通信時のプロセス WAIT 処理の影響によるものである．#3，#4，および#7 におけるプロセスの動作状況を**図 6** に示す．#3



A: ベンチマークプロセス（優先度：低）
F: ファイル管理サーバ（優先度：高）
B: ブロック管理サーバ（優先度：高）

図 6 プロセスの動作状況

では，依頼元プロセス（ベンチマークプロセス）が依頼先プロセス（ファイル管理サーバ）よりも低優先度であるため，プロセス WAIT 処理時間がコア間通信時間よりも短い場合，依頼元プロセスのプロセス WAIT 処理は処理時間に影響しない．一方，#4 と#7 では，依頼元プロセスが依頼先プロセスと同等以上の優先度であるため，依頼元プロセスのプロセス WAIT 処理が処理時間に加算される．したがって，#3 は，#4 と#7 に比べ，処理時間がプロセス WAIT 処理時間（0.1 μ 秒）だけ短くなる．

3.3 PostMark

PostMark のパラメータとして，ファイル数を 25，実行回数を 25 回，ファイルサイズを 500Bytes〜10000Bytes の間とし，測定した．結果を**図 7** に示す．図 7 より，以下のことがわかる．

（1）ディスクドライバサーバのみを分散した場合（#2）の処理時間は，分散しない場合（#1）と同等である．これは，ファイル数とファイルサイズの総数の和が小さいため，ファイル作成時にファイルデータはすべてファイルキャッシュ上に存在し，ディスクドライバサーバへの処理依頼が発生しないことに起因する．したがって，ファイル管理サーバとブロック管理サーバの分散形態が同じ場合，処理時間はディスクドライバサーバの分散形態に関係なく同等である．例えば，4 コア分散の#15 の処理時間は，3 コア分散の#11 や#13 や#14 と同等である．

（2）ベンチマークプロセスとファイル管理サーバを同じコアへ分散した場合（#3，#5，および#9）の処理時間は，

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

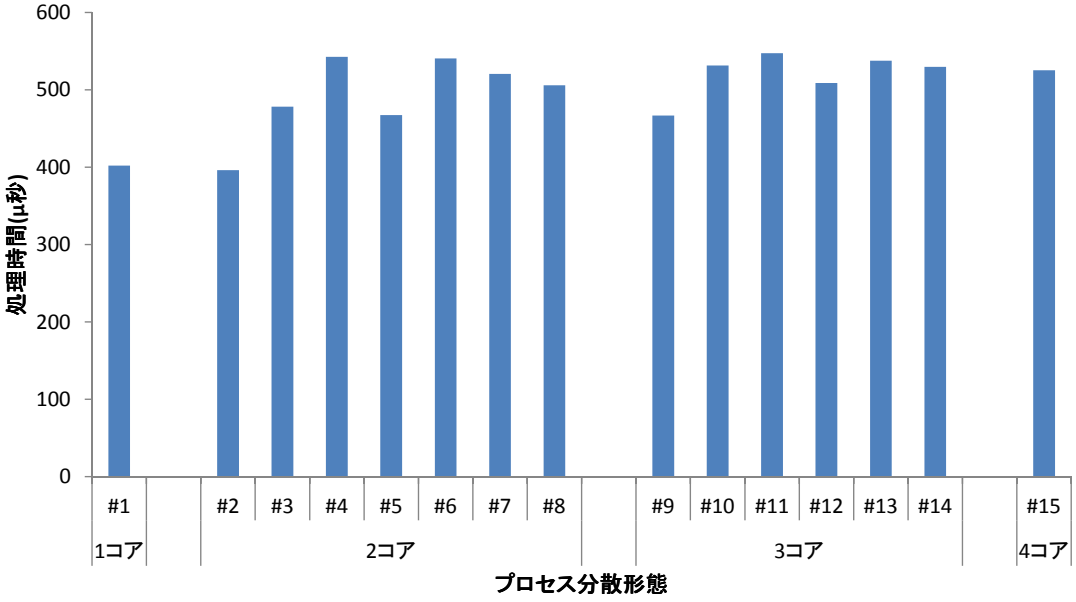


図 7 PostMark の測定結果

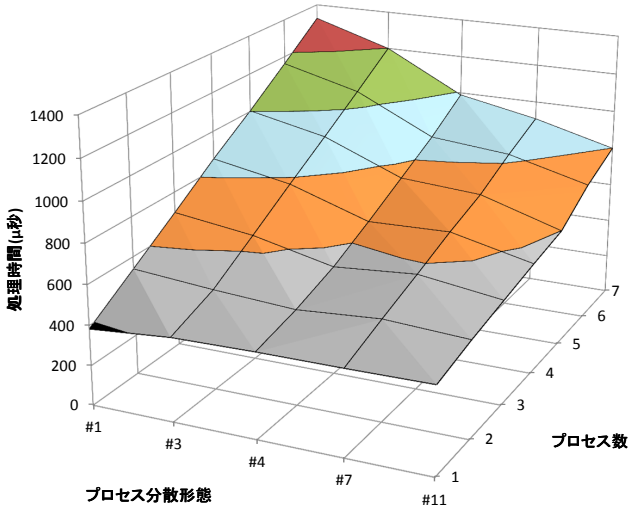


図 8 Bonnie の測定結果

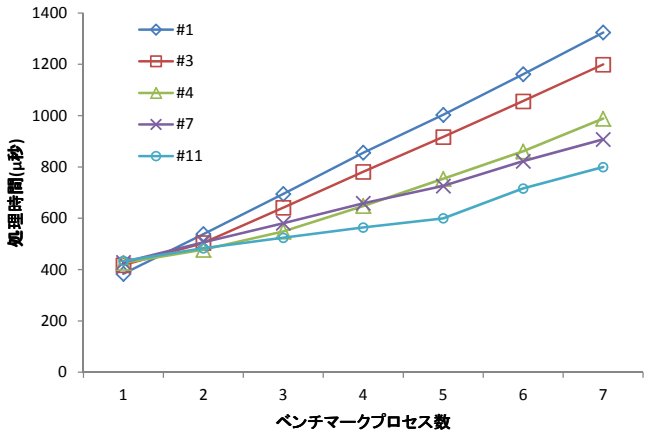


図 9 ベンチマークプロセス数と処理時間の関係

両プロセスを別々のコアへ分散した場合に比べ短い．これは，ファイルのオープン時とクローズ時に発生するコア間通信回数が少ないためである．具体的には，同じコアへ分散した#3，#5，および#9 ではオープン時に 2 回，クローズ時に 0 回のコア間通信が発生するのに対し，別々のコアへ分散した#4，#6，および#10 ではオープン時に 4 回，クローズ時に 2 回，#7，#8，および#12 ではオープン時に 2 回，クローズ時に 2 回のコア間通信が発生する．

3.4 Bonnie

Bonnie のパラメータとして，プロセス数を 1〜7，ファイルサイズを 128KBytes，参照回数を 4000 回，更新する確率を 10%とし，測定した．3.3 節の評価と同様に，ファイルサイズが小さいため，ファイルデータはすべてファイルキャッシュ上に存在するようになり，ディスクドライバサーバへの処理依頼は最初に発生するのみである．そこで，本測定では，ディスクドライバサーバの分散については考慮しないこととした．具体的には，表 1 の#1，#3，#4，#7，および#11 について測定した．結果を**図 8** に示す．図 8 に基づく分析のために**図 9** と**図 10** を示し，考察する．

ベンチマークプロセス数と処理時間の関係を図 9 に示し，以下に考察する．

（1）OS 処理の負荷分散効果は，ベンチマークプロセス数が多いほど大きくなる．いずれの場合も，ベンチマークプロセス数が増加すると処理時間が増加してしまう．しかし，その増加率は，使用するコア数が多いほど小さい．このことは，#1（1 コア）と#11（3 コア）を比較すると明

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14

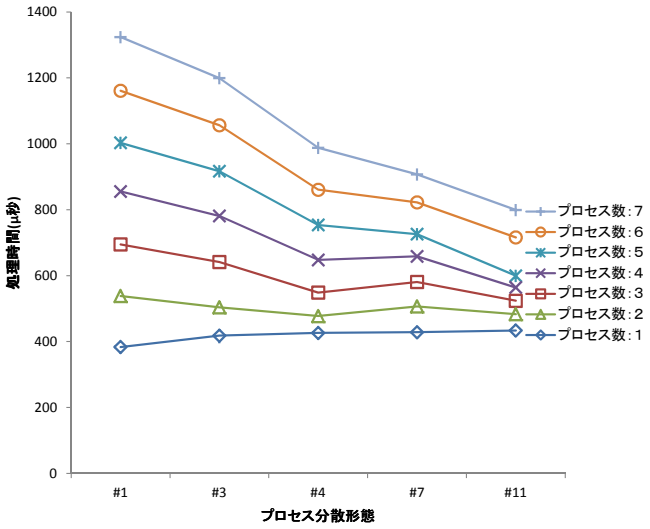


図 10 プロセス分散形態と処理時間の関係

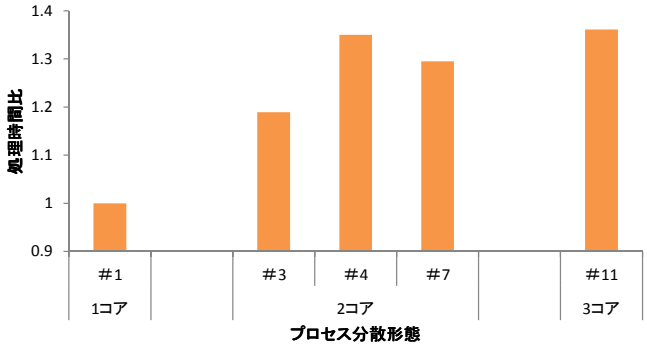
確である．具体的には，#1 の場合，ベンチマークプロセス数が 1 個増加するにつれて約 155 μ 秒（ベンチマークプロセス数が 1〜7 の平均値）だけ処理時間が増加する．一方，#11 の場合、約 50 μ 秒しか増加しない。ただし，ベンチマークプロセス数が 6 以上では，OS サーバの処理が多く（具体的には，ファイル管理サーバの処理であり，詳細は後述する）なり，やや増加率が大きくなっている．

（2）ファイル管理サーバを別コアに分散させることは，有効である．#3 と#4 に着目する．いずれの場合も 2 コアを利用しており，ブロック管理サーバを別コアとするか，ファイル管理サーバを別コアとするかの違いである．#4 の処理時間が#3 より短いことから，ファイル管理サーバを別コアに分散させることが有効といえる．これは，ファイル管理サーバの処理量がブロック管理サーバの処理量より多いことに起因する．具体的には，#3 の場合，ベンチマークプロセス数が 1 個増加するにつれて約 127 μ 秒だけ処理時間が増加するが，#4 の場合，約 86 μ 秒しか増加しない．

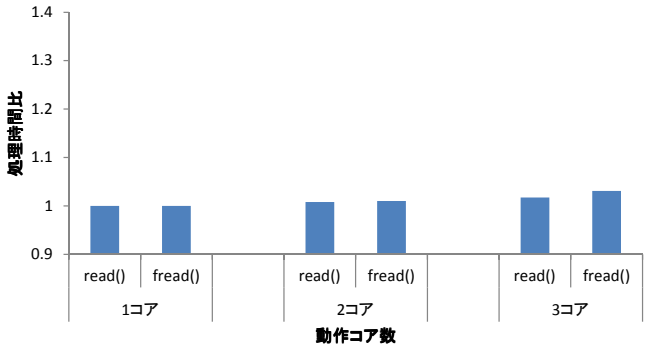
（3）OS 処理を別コアに分散させることは，有効である．#4 と#7 に着目する．いずれの場合も 2 コアを利用しているものの，#7 はファイル管理サーバだけではなくブロック管理サーバも別コアに分散させた場合である．この有効性は，ベンチマークプロセス数が多くなると現れている．つまり，ベンチマークプロセスの要求を多数行なえる環境が有効である．

次に，プロセス分散形態と処理時間の関係を図 10 に示し，考察する．

（4）プロセス分散形態，つまり使用コア数（1 コア（#1），2 コア（#3，#4，および#7），3 コア（#11））により，効果に大きな違いがある．ベンチマークプロセス数が 1 の場合は，当然のことながら，分散処理の効果が出ないため，



(a) ***AnT***



(b) FreeBSD

図 11 PostMark による処理時間比

使用コア数を増加させると，コア間通信のオーバヘッドにより処理時間は増加してしまう．これに対し，ベンチマークプロセス数が多い，つまりベンチマークプロセスの要求を多数行なえる環境であるほど、OS 処理の分散の効果は大きいといえる．ただし、#4 と#7 の比較でわかるように，OS 処理を実現するサーバの処理量に依存して効果に違いが生じる．

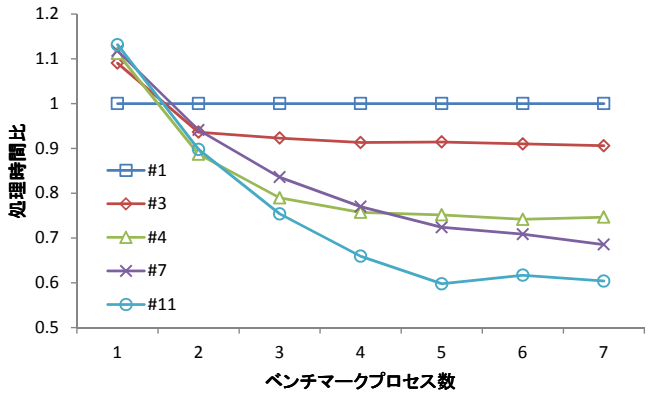
3.5 *AnT* と FreeBSD の比較

PostMark と Bonnie を FreeBSD6.3-RELEASE 上で測定し，***AnT*** と比較する．***AnT*** は，#1，#3，#4，#7，および#11 の分散形態とし，FreeBSD は，動作コア数を 1〜3 とした．1 コアを利用する場合の処理時間に対する比を**図 11** と**図 12** に示す．各図より，以下のことがわかる．

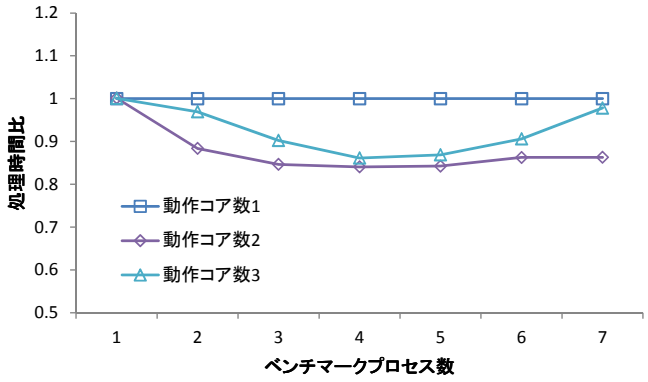
（1）PostMark の場合，***AnT*** における OS 処理の分散効果はない．PostMark はベンチマークプロセス数が 1 個であるため，***AnT*** では，OS 処理を分散するに伴い，コア間通信オーバヘッドが増加し，処理時間が増加する．一方，モノリシックカーネル構造の FreeBSD では，OS 処理を分散できないため，動作コア数によらず，処理時間は一定になる．

（2）Bonnie の場合，***AnT*** における OS 処理の分散効果は高い．Bonnie はベンチマークプロセス数が多い場合，***AnT*** では，OS 処理を分散するに伴い，処理時間が大きく減少する．一方，FreeBSD では，動作コア数を増加しても，処理時間が大きく減少しない．例えば，ベンチマーク

Vol.2012-DPS-152 No.31
Vol.2012-GN-85 No.31
Vol.2012-EIP-57 No.31
2012/9/14



(a) ***AnT***



(b) FreeBSD

図 12 Bonnie による処理時間比

プロセス数が 5 個の場合，***AnT***（#11：3 コア分散）では処理時間を 60% に短縮できるのに対し，FreeBSD（動作コア数 3）では，87% までしか短縮できない．

4. まとめ

マルチコア環境用 ***AnT*** オペレーティングシステムについて，OS のファイル入出力処理の負荷分散効果を評価した．

基本性能の評価では，OS 処理の分散により，コア間通信回数が増加し，処理時間が増加することを明らかにした．なお，同一コアに共存走行するプロセス優先度の違いが処理時間に影響を与える．PostMark ベンチマークプログラム（ベンチマークプロセス数は 1）の評価では，ファイルキャッシュ上にすべてのファイルデータが存在する場合，ディスクドライバサーバの分散効果は無いことを明らかにした．また，ベンチマークプロセスとファイル管理サーバを同じコアで走行させることにより，ファイルのオープンとクローズ時のコア間通信回数が減少して処理時間が短くなることを明らかにした．Bonnie ベンチマークプログラム（ベンチマークプロセス数は 1〜7）の評価では，ベンチマークプロセス数が多く，ベンチマークプロセスの要求を多数行なえる環境であるほど分散の効果が大きいことを明らかにした．最後に，PostMark と Bonnie を FreeBSD 上で測定し，***AnT*** と比較した．PostMark の場合，***AnT*** では，OS 処理を分散するに伴い，コア間通信オーバヘッドが増加し，処理時間が増加するのに対し，FreeBSD では，OS 処理を分散できないため，処理時間が一定になることを明らかにした．Bonnie の場合，***AnT*** では，OS 処理を分散できるため，処理時間を 60%に短縮できるのに対し，FreeBSD では，OS 処理を分散できないため，処理時間を 87%までしか短縮できないことを明らかにした．

残された課題として，通信の入出力処理に関する評価がある．

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] Liedtke, J.: Toward real microkernels, Communications of the ACM, Vol.39, No.9, pp.70-77 (1996).
[2] 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).
[3] 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).
[4] 島崎泰, 山内利宏, 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 Linux の LKM 形式ドライバのプロセス化手法, 電子情報通信学会論文誌 (D), Vol.J93-D, No.10, pp.1990-2000 (2010).
[5] Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A., and Weissmann, E.: Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge, IEEE Micro, Vol.32, No.2, pp.20-27 (2012).
[6] 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, 情報処理学会研究報告, Vol.2012-DPS-150, No.37, pp.1-8 (2012).
[7] 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).
[8] 佐古田健志, 枡田圭祐, 井上喜弘, 谷口秀夫: マルチコア ***AnT*** における処理分散機能, 情報処理学会研究報告, Vol.2012-OS-122, No.5, pp.1-8 (2012).
[9] 枡田圭祐, 谷口秀夫: ファイルサイズの拡縮が可能なメモリ上ファイル操作機能の提案, 情報処理学会研究報告, Vol.2012-OS-121, No.7, pp.1-8 (2012).
[10] Gallmeister, B.O.: POSIX.4, O'Reilly, pp. 128-129 and 389-391. (1995).
[11] Katcher, J.: PostMark: A New File System Benchmark, Technical Report TR3022, Network Appliance (1997).
[12] Bray, T.: The Bonnie home page(online), 入手先 ⟨http://www.textuality.com/bonnie/⟩ (1996).