Vol.2015-ARC-215 No.14
Vol.2015-OS-133 No.14
2015/5/27

AnT のOSサーバ入替え機能における入替え時間と応答時間の分析

澤田 淳[1]　山内 利宏[1]　谷口 秀夫[1]

概要：計算機が提供するサービスの高度化に伴い，OS の動作中に OS のプログラムを修正する機能が必要になっている．また，OS に対し，計算機の多様な利用を支える高い適応性，および不具合の発生に対応する高い堅牢性が要求されている．マイクロカーネル構造を有する ***AnT*** オペレーティングシステムでは，OS サーバの入替え機能が提案されており，OS 機能を更新することにより，高い適応性と堅牢性の実現を目指している．本稿では，OS サーバの処理内容と OS サーバ利用 AP プロセス数に着目し，これらが OS サーバの入替え時間と応答時間に与える影響を分析し，明らかにする．

1. はじめに

計算機システムの高性能化と低価格化により，様々なサービスの提供に計算機システムが導入されている．また，常時接続ネットワークも普及し，計算機システムの 24 時間無停止が求められている．これに伴い，基盤ソフトウェアであるオペレーティングシステム（以降，OS）に対し，計算機の多様な利用を支える高い適応性，および不具合の発生に対応する高い堅牢性が要求されている．また，システムの停止を抑制し，サービスを継続することが要求されている [1]．

高い適応性と堅牢性を実現するための OS の構成法として，マイクロカーネル構造 [2]~[4] がある．マイクロカーネル構造は，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，ファイル管理や通信制御などの処理をプロセスとして実現するプログラム構造である．つまり，多くの OS 機能は，カーネル外にプロセス（以降，OS サーバ）として実現する．これにより，OS サーバの機能拡張時や不具合発生時には，対象の OS サーバを新たな OS サーバに入替える [5]~[7] ことで，システムを停止することなくサービスの継続が可能となる．

マイクロカーネル構造を有する ***AnT*** オペレーティングシステム [8]（An operating system with adaptability and toughness）（以降，***AnT***）において，OS サーバ入替え機能 [7] が提案されている．本機能の特徴は，OS サーバの保有情報を最小化し，入替え前後の OS サーバ間における処理の継続を可能にしている点である．***AnT*** は，OS サーバ入替え機能により，OS 機能を更新することで，高い適応性と堅牢性の実現を目指している．しかし，複数の応用プログラム（以降，AP）プロセスが同時に OS サーバを利用する場合について，OS サーバの処理内容と OS サーバ利用 AP プロセス数が OS サーバの入替え時間と応答時間に与える影響は明らかでない．

そこで，本稿では，OS サーバの処理内容と OS サーバ利用 AP プロセス数に着目し，これらが OS サーバの入替え時間と応答時間に与える影響を分析し，明らかにする．

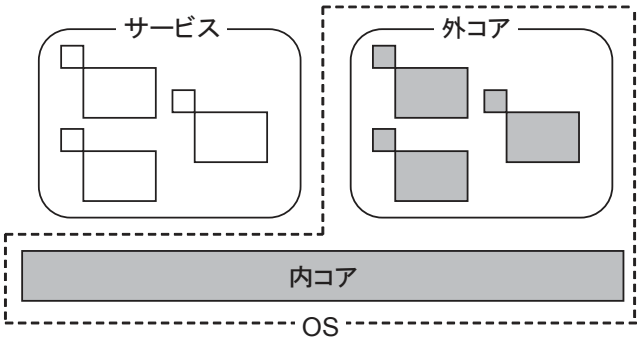

図 1　***AnT*** の基本構造

2. *AnT* オペレーティングシステム

2.1 基本構造

AnT の基本構造を**図 1** に示す．***AnT*** はマイクロカーネル構造 OS であり，プログラムは，OS とサービスからなる．OS は，カーネル（内コア）とプロセスとして動作する OS サーバ（外コア）からなる．サービスは，AP プロセスからなる．

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

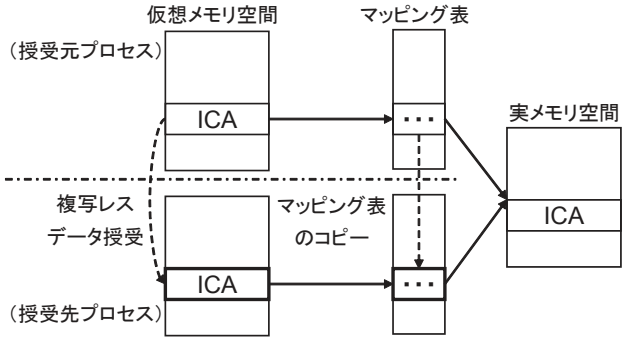

図 2 複写レスデータ授受の様子

内コアは，システムの動作を保証する最小のプログラム部分である．外コアは，計算機の多様な利用に適応したシステムに必須なプログラム部分であり，動的に再構築可能な構造を有する．サービスは，サービスを提供するプログラム部分である．

2.2 複写レスデータ授受

プロセス間の通信を高速化するため，コア間通信データ域（ICA: Inter-core Communication Area）を利用した複写レスデータ授受機能がある．ICA の特徴として，以下の 3 つがある．

（1）ページ（4 KB）を単位とする n ページ分の領域の確保と解放

（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，内コア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，プロセスごとの仮想空間のマッピング表を通して行われる．ここで，マッピング表への書き込みを貼り付けと呼び，マッピング表からの削除を剥がしと呼ぶ．プロセス間での複写レス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 をデータ授受元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．

2.3 サーバプログラム間通信機構

サーバプログラム間通信の基本機構を**図 3** に示す．ICA を利用することにより，プロセス間でデータ複写レスでの通信を実現している．具体的には，OS サーバへ渡す引数や通信制御の情報（以降，依頼情報）を制御用の ICA（以降，制御用 ICA）に格納し，扱うデータをデータ用の ICA（以降，データ用 ICA）に格納し，これらの ICA の貼り替えにより，データを授受する．カーネルは，各プロセスごとに通信のための依頼キューと結果キューをもつ．なお，依頼元プロセスの呼び出し先の特定は，OS サーバの提供する OS 機能に対して一意に与えられた識別子（以降，機

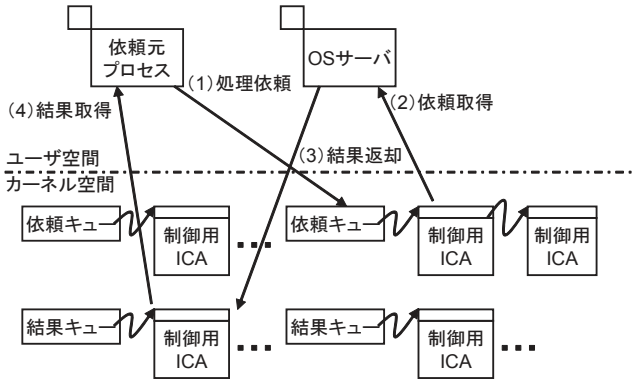

図 3 サーバプログラム間通信の基本機構

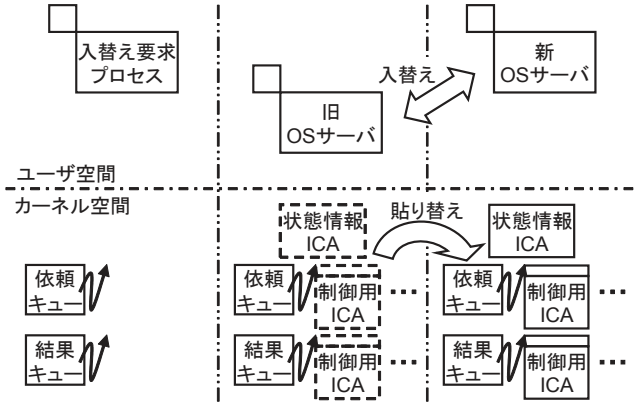

図 4 OS サーバ入替えの基本機構

能識別子）を利用して行う．サーバプログラム間通信の流れを以下に述べる．

（1）依頼元プロセスが処理依頼を行うと，内コアは OS サーバの依頼キューに依頼情報を格納した制御用 ICA を登録し，OS サーバへ制御用 ICA を貼り替える．

（2）OS サーバは，依頼キュー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，内コアは依頼元プロセスの結果キューに処理の結果（以降，結果情報）を格納した制御用 ICA を登録し，依頼元プロセスへ制御用 ICA を貼り替える．

（4）依頼元プロセスは，結果キュー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．

2.4 OS サーバ入替え機能

OS サーバ入替え機能は，動作中の OS サーバを同等のサービスを提供する新たな OS サーバへ入替える機能である．OS サーバ入替えの基本機構を**図 4** に示す．OS サーバ入替え機能を呼び出すプロセス（以降，入替え要求プロセス）は，システムコールにより内コアに入替え要求を発行する．これにより，内コアは，入替え対象の OS サーバ（以降，旧 OS サーバ）と同等のサービスを提供する新たな OS サーバ（以降，新 OS サーバ）を生成し，旧 OS サーバの保持する他プロセスからの依頼と結果の情報（以降，

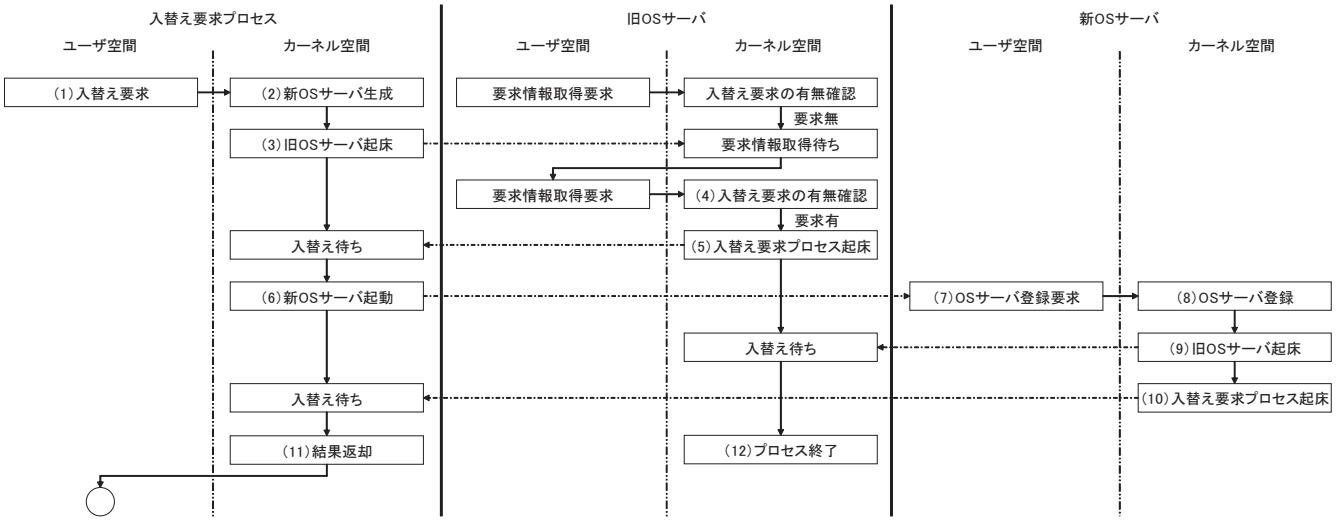

図 5 OS サーバ入替えの処理流れ

要求情報）を新 OS サーバに移譲する．具体的には，内コアは旧 OS サーバの依頼キューや結果キューに繋がれているすべての ICA を新 OS サーバへ貼り替える．また，OS サーバの内部状態に関する情報（以降，状態情報）についても，状態情報を ICA（以降，状態情報 ICA）に格納することで，複写レスデータ授受を行っている．

入替え要求プロセスが旧 OS サーバを新 OS サーバに入替える際の処理流れを**図 5** に示し，以下で説明する．

（1）入替え要求プロセスは，システムコールにより内コアに入替え要求を発行する．

（2）上記（1）の要求後，内コアは，新 OS サーバを生成する．

（3）新 OS サーバの生成後，内コアは，旧 OS サーバを起床させ，入替え要求を通知する．

（4）起床した旧 OS サーバは，要求情報の取得に優先して入替え要求の有無を確認する．

（5）上記（4）の確認後，旧 OS サーバは，入替え要求プロセスを起床させる．

（6）内コアは，新 OS サーバを起動する．

（7）起動した新 OS サーバは，システムコールにより内コアに OS サーバ登録要求を発行する．

（8）上記（7）の要求後，内コアは，新 OS サーバを OS サーバとして登録する．具体的には，旧 OS サーバの要求情報，状態情報，および機能識別子を新 OS サーバに引き継ぐ．

（9）上記（8）の引き継ぎ後，内コアは，旧 OS サーバを起床させる．

（10）旧 OS サーバの起床後，内コアは，入替え要求プロセスを起床させる．

（11）内コアは，入替えを終了し，入替え要求プロセスに結果返却を行う．

（12）旧 OS サーバは，プロセスを終了する．

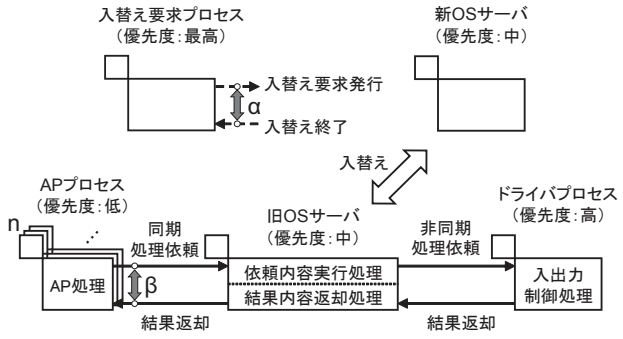

図 6 OS サーバ入替えのモデル
（α：入替え時間，β：応答時間，n：AP プロセス数）

なお，上記は，入替え要求プロセスがシステムコールにより入替え要求を発行し，即座に入替えを開始できる場合である．

3. 入替え時間と応答時間

3.1 OS サーバ入替えのモデル

OS サーバ入替えのモデルを**図 6** に示す．AP プロセスは，OS サーバに同期処理を依頼し，OS サーバは，ドライバプロセスに非同期処理を依頼する．この場合の処理流れを以下で説明する．なお，各プロセスの優先度は，入替え要求プロセス，ドライバプロセス，旧 OS サーバ，AP プロセスの順（入替え要求プロセス > ドライバプロセス > 旧 OS サーバ > AP プロセス）で高いものとし，新 OS サーバの優先度は，旧 OS サーバの優先度と同等とする．

（1）AP プロセスは，AP 処理を行う．

（2）AP 処理後，AP プロセスは，OS サーバに同期処理を依頼する．

（3）OS サーバは，入替え要求の有無を確認し，なければ依頼情報を依頼キューから取得し，依頼情報に基づいた処理（以降，依頼内容実行処理）を行う．

（4）依頼内容実行処理後，OS サーバは，ドライバプロセスに非同期処理を依頼する．
（5）依頼情報を依頼キューから取得したドライバプロセスは，依頼情報に基づいた入出力制御処理（プロセッサ（PU）処理と実 I/O 等による待ち（WAIT）処理からなる）を行う．
（6）入出力制御処理後，ドライバプロセスは，OS サーバに結果返却を行う．
（7）OS サーバは，入替え要求の有無を確認し，なければ結果情報を結果キューから取得し，この情報に基づいた処理（以降，結果内容返却処理とする）を行う．
（8）結果内容返却処理後，OS サーバは，AP プロセスに結果返却を行う．

上記のモデルにおいて，入替えは任意の契機で行うものと仮定する．

3.2 入替え時間

入替え要求プロセスがシステムコールにより入替え要求を発行してから，終了するまでの時間（以降，入替え時間）は，入替え処理が OS サーバやドライバプロセスの PU 処理の影響を受けない場合に最も短くなる．

これに対し，入替え時間が長くなる要因には 2 つある．1 つは，入替え可能となるまでに待ちの発生する場合である．具体的には，以下の場合である．
（1）旧 OS サーバが依頼内容実行処理中である場合，入替え時間は，残存する依頼内容実行処理と，依頼内容実行処理に続いて実行される入出力制御処理の PU 処理の影響を受けて長くなる．
（2）ドライバプロセスが入出力制御処理の PU 処理中である場合，入替え時間は，残存する入出力制御処理の PU 処理の影響を受けて長くなる．
（3）旧 OS サーバが結果内容返却処理中である場合，入替え時間は，残存する結果内容返却処理の影響を受けて長くなる．

もう 1 つは，入替え処理中にドライバプロセスが PU 処理を開始する場合である．ドライバプロセスは，入替え処理を行う新旧 OS サーバに比べて高優先度である．このため，ドライバプロセスが実 I/O 等による WAIT から解除されると READY 状態になり，入替え処理より優先して実行される．

3.3 応答時間

AP プロセスによる処理依頼の発行から，結果返却の完了までの時間（以降，応答時間）は，OS サーバやドライバプロセスの PU 処理が入替え処理の影響を受けない場合に最も短くなる．

これに対し，OS サーバやドライバプロセスの処理中に入替えの開始を指示した場合，応答時間は長くなる．具体的には，以下の場合である．
（1）旧 OS サーバが依頼内容実行処理中である場合．
（2）ドライバプロセスが入出力制御処理の PU 処理中である場合．
（3）旧 OS サーバが結果内容返却処理中である場合．

いずれの場合も，各処理の後に実行する処理が遅延することに起因する．

4. 評価

4.1 評価内容

評価モデルには図 6 を用いた．Intel® Corei7 搭載の計算機を利用して測定し，OS サーバの処理内容と OS サーバ利用 AP プロセス数が入替え時間と応答時間に与える影響を明らかにする．具体的には，各 AP プロセスにより処理依頼を 100 回発行し，入替えを 50 回行い，入替え時間と応答時間を測定した．

入替え要求の発行は，入替え処理無しの場合の応答時間を t_0 とすると，t_0 以上 $2t_0$ 以下のランダムな間隔とした．また，AP 処理は 10ms の WAIT 処理，入出力制御処理は 5ms の PU 処理と 15ms の WAIT 処理の組み合わせとし，PU 処理である依頼内容実行処理と結果内容返却処理の処理時間を変化させた．なお，PU 処理は特定領域のインクリメントを繰り返すプロセッサ処理とし，WAIT 処理は指定時間だけ実行権を放棄するシステムコールを用いた待ち処理とした．

4.2 入替え時間

4.2.1 依頼内容実行処理時間と入替え時間の関係

依頼内容実行処理時間と入替え時間の関係を明らかにするため，結果内容返却処理時間を 1ms と短い時間に固定し，依頼内容実行処理時間を 10ms，15ms，および 20ms とした場合の入替え時間を**図 7** に示す．図 7 より，以下のことがわかる．
（1）入替え時間の最小値は，AP プロセス数や依頼内容実行処理時間に関係なく 7.4ms である．当然のことながら，この値は，AP プロセスが処理を依頼しない場合の入替え時間（以降，基本入替え時間）7.4ms と同じである．
（2）入替え時間の最大値は，AP プロセス数に関係なく一定である．いずれの場合も，最大値は，基本入替え時間に依頼内容実行処理時間と入出力制御処理の PU 処理時間を加えた値である．例えば（A）の場合，22.4ms（$= 7.4 + 10 + 5$）である．これは，依頼内容実行処理の開始直後に入替え要求が発行され，入替え時間が依頼内容実行処理と入出力制御処理の PU 処理に受ける影響の最大化したためである．
（3）AP プロセス数が多くなると，入替え時間の平均値と中央値は大きくなる．これは，AP プロセス数が多くなると，入替え時間が依頼内容実行処理や結果内容返却処理の影響を受けて長くなる確率の増加したためである．

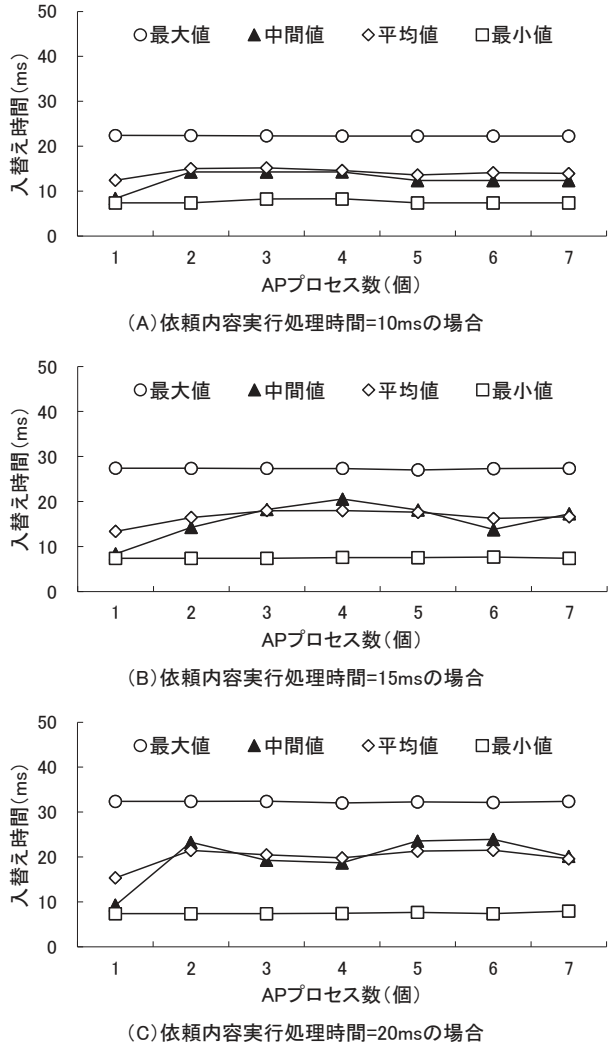

図 7 入替え時間（結果内容返却処理時間=1ms）

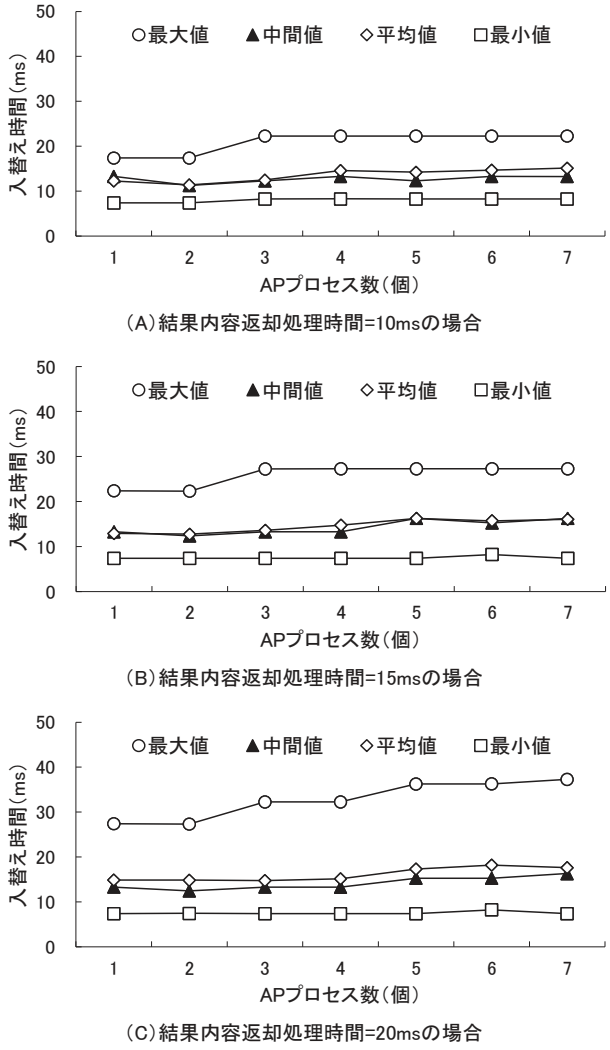

図 8 入替え時間（依頼内容実行処理時間=1ms）

4.2.2 結果内容返却処理時間と入替え時間の関係

結果内容返却処理時間と入替え時間の関係を明らかにするため，依頼内容実行処理時間を 1ms と短い時間に固定し，結果内容返却処理時間を 10ms，15ms，および 20ms とした場合の入替え時間を**図 8** に示す．図 8 より，以下のことがわかる．

（1）入替え時間の最小値は，4.2.1 項（1）で述べた理由と同様で，7.4ms（基本入替え時間）である．

（2）AP プロセス数が 2 個以下の時，入替え時間の最大値は，いずれの場合も，基本入替え時間に結果内容返却処理時間を加えた値である．例えば（A）の場合，17.4ms（$= 7.4 + 10$）である．これは，結果内容返却処理の開始直後に入替え要求が発行され，入替え時間が結果内容返却処理に受ける影響の最大化したためである．なお，依頼内容実行処理時間は 1ms と短いため，依頼内容実行処理の開始直後に入替え要求を発行した場合には，入替え時間は 6ms（$= 1 + 5$）しか長くならず，入替え時間は最大とならない．

（3）入替え時間の最大値は，AP プロセス数が 5 個以上の時，一定である．いずれの場合も，最大値は，基本入替え時間に結果内容返却処理時間と，入出力制御処理の PU 処理時間（5ms）の倍数を加えた値である．例えば（C）の場合，37.4ms（$= 7.4 + 20 + 5 \times 2$）である．これは，結果内容返却処理の開始直後に入替え要求が発行され，入替え時間が結果内容返却処理と入出力制御処理の PU 処理に受ける影響の最大化したためである．この場合，結果内容返却処理中にドライバプロセスに制御移行する回数は，結果内容返却処理時間（20ms）と入出力制御処理の WAIT 処理時間（15ms）から多くとも 2 回（$= \lceil 20/15 \rceil$）となり，入出力制御処理の PU 処理の影響による長大化は 10ms（$= 5 \times 2$）となっている．

4.3 応答時間

依頼内容実行処理時間を 10ms，結果内容返却処理時間を 1ms とした場合の応答時間を**図 9** に示す．図 9 より，以下のことがわかる．

（1）入替え処理無しの場合（A）において，AP プロセス数が 1 個の時，応答時間は 31ms である．この値は，依頼内容実行処理時間（10ms），入出力制御処理の PU 処理時

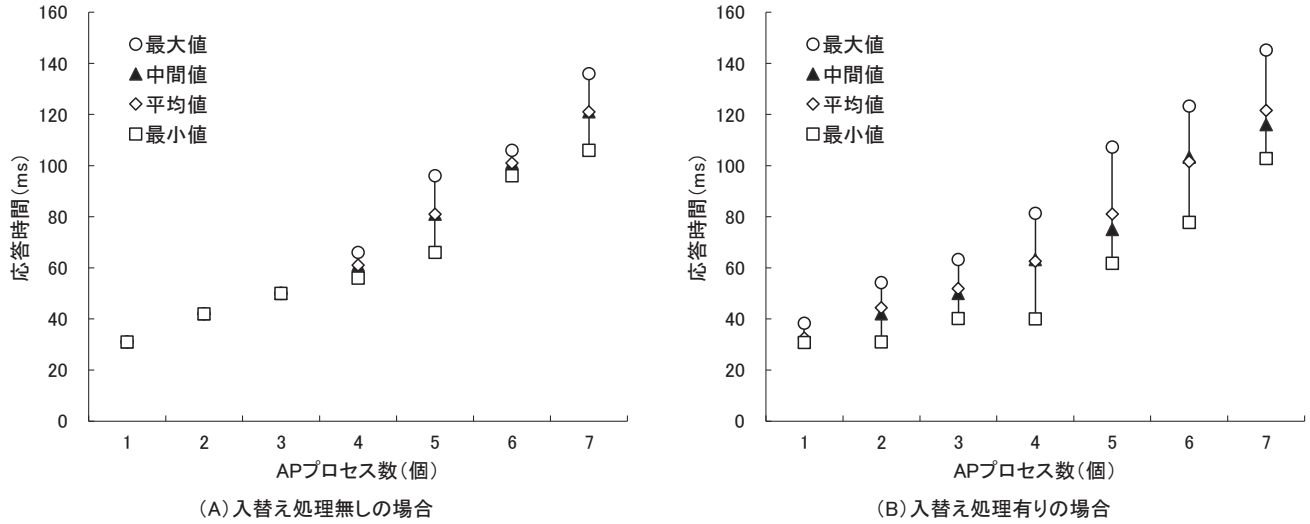

図 9 応答時間（依頼内容実行処理時間=10ms，結果内容返却処理時間=1ms）

間（5ms），入出力制御処理の WAIT 処理時間（15ms），および結果内容返却処理時間（1ms）の和（以降，基本応答時間）である．これに対し，AP プロセス数が複数の時，応答時間は，基本応答時間より長くなる．これは，複数の依頼に対する処理の相互作用に起因する．

（2）入替え処理有りの場合（B）において，AP プロセス数が 1 個の時，応答時間の最小値は 31ms となっている．この値は，基本応答時間に等しい．したがって，入替え処理が応答時間に影響を与えない契機において処理依頼の発行されたことがわかる．また，この時の最大値は，最小値に比べて 7.4ms 大きい．この値は，基本応答時間と基本入替え時間の和であり，OS サーバやドライバプロセスの PU 処理中に入替え要求が発行され，入替え処理の影響を受けたことがわかる．

次に，AP プロセス数が複数の場合について，入替え処理が応答時間に与える影響を考察する．例えば，AP プロセス数が 6 個の時，応答時間の最大値は，入替え処理無しの場合 106ms，入替え処理有りの場合 123ms である．この場合，これらの最大値の差 17ms（= 123 - 106）は，基本入替え時間より大きい．この現象を説明するために，依頼内容実行処理時間と結果内容返却処理時間を変化させた場合について，入替え処理無しの場合の応答時間の最大値と，入替え処理有りの場合の応答時間の最大値の差を**図 10** に示す．図 10 より，以下のことがわかる．

（1）依頼内容実行処理時間と結果内容返却処理時間が入出力制御処理の WAIT 処理時間（15ms）より短く，AP プロセス数が複数の場合，入替え処理無しの場合の応答時間の最大値と入替え処理有りの場合の応答時間の最大値の差は，7.4ms（基本入替え時間）より大きい．例えば，依頼内容実行処理時間が 10ms，結果内容返却処理時間が 1ms，AP プロセス数が 6 個の場合，先に述べた通り 17ms である．これは，AP プロセスの優先度が最低であり，OS サーバやドライバプロセスが PU 処理を行っていない時間にしか結果取得を行えないにも関わらず，この時間が入替え時間に隠蔽されるためである．この結果，入替え処理無しの場合の応答時間の最大値と入替え処理有りの場合の応答時間の最大値の差は，入替え処理に加え，入替え処理に続いて実行される他の依頼に対する処理の影響を受け，基本入替え時間より長くなる．

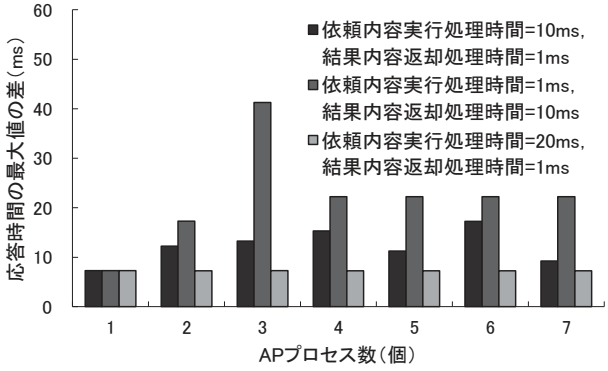

図 10 入替え処理無しの場合の応答時間の最大値と入替え処理有りの場合の応答時間の最大値の差

（2）依頼内容実行処理時間または結果内容返却処理時間が入出力制御処理の WAIT 処理時間（15ms）以上であり，AP プロセス数が複数の場合，入替え処理無しの場合の応答時間の最大値と入替え処理有りの場合の応答時間の最大値の差は，AP プロセス数に関係なく 7.4ms（基本入替え時間）で一定となっている．例えば，依頼内容実行処理時間が 20ms，結果内容返却処理時間が 1ms，AP プロセス数が 7 個の時，7.4ms である．これは，OS サーバの優先度が AP プロセスより高く，入替え処理の有無に関係なく，依頼内容実行処理により AP プロセスの結果取得が待たされるためである．

5. おわりに

AnT オペレーティングシステムにおいて，OS サーバ入替えのモデルに基づき，OS サーバの処理内容と OS サーバ利用 AP プロセス数が入替え時間と応答時間に与える影響を分析した．

入替え時間は，依頼内容実行処理中，結果内容返却処理中，および入出力制御処理の PU 処理中に入替え要求を発行した場合に長くなる．また，応答時間は，依頼内容実行処理，結果内容返却処理，および入出力制御処理の PU 処理中に入替え要求を発行した場合に長くなる．

また，上記モデルを用いた実測により，入出力制御処理の WAIT 処理時間が 15ms，結果内容返却処理時間が 20ms の時，結果内容返却処理中にドライバプロセスへの制御移行の発生する回数は多くとも 2 回（$= \lceil 20/15 \rceil$）であることから，入替え時間は，ドライバプロセスへの制御移行により長大化する場合にも，AP プロセス数に比例して無限に長大化することのないことを明らかにした．また，応答時間は，基本入替え時間（7.4ms）に加え，他の AP プロセスによる処理依頼の影響を受けてさらに長大化する場合のあることを明らかにした．

残された課題として，実サービス環境での評価がある．

参考文献

[1] J. Arnold, M.F. Kaashoek, “KSplice: Automatic Rebootless Kernel Updates,” EuroSys ’09 Proceedings of the 4th ACM European conference on Computer systems, pp.187–198，2009.
[2] J. Liedtke, “Toward Real Microkernels,” Commnuications of The ACM，Vol.39，Issue 9，pp.70–77，1996.
[3] A.S. Tanenbaum, J.N. Herder, Herbert Bos, “Can we make operating systems reliable and secure?,” IEEE Computer Magazine，Vol.39，No.5，pp.44–51，2006.
[4] D.L. Black, D.B. Golub, D.P. Julin, R.F. Rashid, R.P. Draves, R.W. Dean, A. Forin, J. Barrera, H. Tokuda, G. Malan, D. Bohman, “Microkernel Operating System Architecture and Mach,” Journal of Information Processing, Vol.14, No.4(19920315)，pp.442–453，1992.
[5] C. Giuffrida，A.S. Tanenbaum，“Cooperative Update: A New Model for Dependable Live Update,” Proceedings of the Second International Workshop on Hot Topics in Software Upgrades，pp.1–6，2009.
[6] C. Giuffrida，A.S. Tanenbaum，“Safe and Automated State Transfer for Secure and Reliable Live Update,” Proceedings of the Fourth International Workshop on Hot Topics in Software Upgrades，pp.16–20，2012.
[7] 藤原康行，井上喜弘，後藤佑介，山内利宏，乃村能成，谷口秀夫，“***AnT*** における通信制御サーバ入れ替え処理の評価，” マルチメディア通信と分散処理ワークショップ論文集, Vol.2010, No.11, pp.101–106, 2010.
[8] 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.