AnT オペレーティングシステムにおける ドライバプロセス入替え機能

蓮岡 宜明[1]　鶴谷 昌弘[1]　山内 利宏[1]　谷口 秀夫[1]

概要：マイクロカーネル構造を持つ ***AnT*** オペレーティングシステムについて，OS サーバ入替え機能が提案され，基本評価と実サービス評価がされている．しかし，この OS サーバ入替え機能は，割り込み処理を行うドライバプロセスの入替えを考慮していない．ここでは，既存の OS サーバ入替え処理の流れを示し，ドライバプロセス入替えを実現するための課題と対処について述べ，ドライバプロセス入替え機能の実現方式を示す．また，ドライバプロセス入替え機能のオーバヘッド，およびドライバプロセス入替え処理が通信処理とデータ入出力処理に与える影響を示す．

Dynamic Driver Process Replacement Mechanism for *AnT*

Noriaki HASUOKA[1]　Masahiro TSURUYA[1]　Toshihiro YAMAUCHI[1]　Hideo TANIGUCHI[1]

Abstract: We have proposed a dynamic OS server replacement mechanism in ***AnT*** operating system based on microkernel architecture, and evaluated basic performance and the influence for services. However, the previously proposed mechanism can not replace a driver process that includes an interrupt processing. In this paper, we discuss the design and the implementation of a dynamic driver replacement mechanism for ***AnT***. In particular, we show the existing dynamic OS server replacement processing flow, the problems and the solutions to realize the dynamic driver process replacement, and the processing flow of it. In addition, we evaluate the overhead and the influence for communication processing and I/O processing of the dynamic driver process replacement mechanism.

1. はじめに

計算機システムの重要性の増大とともに，計算機システムに対して高い適応性と堅牢性が要求されている．特に，システムの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 と略す）において，高い適応性と堅牢性を実現することは非常に重要である．

OS の適応性と堅牢性を高める手法として，マイクロカーネル構造 OS[1]–[3]がある．マイクロカーネル構造 OS は，最小限の OS 機能をカーネル（以降，内コアと略す）として構築し，他の大半の OS 機能をプロセス（以降，OS サーバと略す）として実現する．OS 機能の追加や削除は OS サーバの追加や削除として実現できるため，システムが必要とする OS 機能を柔軟に構成でき，高い適応性を実現できる．また，OS 機能の不具合発生時の対処として，プログラム入替え機能がある．この機能は，モノリシックカーネル構造 OS において，不具合が生じた OS のプログラムモジュールの入替えで用いられる[4],[5]．多くの場合，プログラムモジュール相互の関係に比べてプロセス相互の関係は疎である．このため，マイクロカーネル構造 OS では，不具合が生じた OS サーバの入替え，つまりプロセスの入替えを簡易な入替え条件で実現でき，モノリシックカーネル構造 OS に比べて高い堅牢性を実現できる．

我々は，マイクロカーネル構造を持つ ***AnT*** オペレーティングシステム[6]（An operating system with adaptability and toughness）（以降，***AnT*** と略す）について，OS サーバの入替え機能を提案し，評価した[7]．しかし，この OS サーバ入替え機能は，割り込み処理を行うドライバプロセ

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

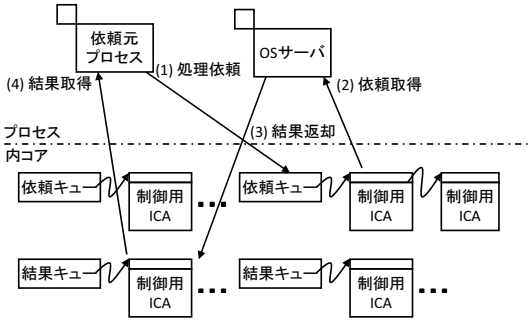

図 1 サーバプログラム間通信機構の基本構造

スの入替えを考慮していない．文献 [8] によると Linux の不具合の 70% 以上はデバイスドライバに存在するという調査結果が報告されている．したがって，システムを停止させずにドライバプロセスの入替えを実現できれば，高い堅牢性を提供できる．

ここでは，***AnT*** におけるドライバプロセス入替え処理を実現するための課題と対処について報告する．また，ドライバプロセス入替え機能のオーバヘッドとドライバプロセス入替え処理が通信処理とデータ入出力処理に与える影響を示す．

2. OS サーバ入替え機能[7]

2.1 サーバプログラム間通信機構

AnT は，コア間通信データ域（ICA : Inter-core Communication Area）を利用して複写レスでのデータ授受機能を実現している．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．

サーバプログラム間通信機構の基本構造を**図 1** に示し，基本的な通信の流れを以下に説明する．

(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．

2.2 実現方式

OS サーバ入替え機能を実現するために次の対処を行う．通信先 OS サーバの特定法として，OS サーバが提供する OS 機能に識別子（以降，機能識別子と略す）を設け，機能識別子をもとに通信相手を特定する通信制御機構を設ける．また，新旧 OS サーバ間の情報移譲法として，OS サーバ入替えにより移譲する情報を OS サーバ外部の ICA に格納する．さらに，OS サーバ入替え処理では移譲する ICA の情報を新 OS サーバに通知する．

OS サーバ入替え処理の流れを**図 2** に示し，以下に説明する．OS サーバ入替え処理は，OS サーバ入替えシステムコール（changeserver システムコール）を発行する入替え制御サーバ，入替え対象であり入替え処理終了後にプロセスを終了する旧 OS サーバ，および入替え処理により新たに生成される新 OS サーバの 3 つのプロセスにより行われる．

（1）入替え制御プロセスは，changeserver システムコールを発行して，入替えを行う．changeserver システムコールは，入替え要求フラグを設定し，旧 OS サーバを起床させ，入替えの終了を待って，復帰する．

（2）旧 OS サーバは，依頼/結果取得システムコール（get システムコール）の処理において，入替え処理を行う．get システムコールは，入替え要求フラグが設定されていると，次の処理を行う．

（A）新 OS サーバを生成し，引継起動フラグを設定し，新 OS サーバを起動し，新 OS サーバの OS サーバ登録システムコール（registserver システムコール）からの同期を待つ．

（B）起床されると，依頼キューと結果キューの情報（要求情報），および ICA に格納している内部状態に関する情報（状態情報）を新 OS サーバに移譲し，入替え要求フラグをリセットする．その後，新 OS サーバと入替え制御プロセスを起床し，プロセスを終了する．

（3）新 OS サーバは，自プロセスを OS サーバとして内コアに登録するために registserver システムコールを発行する．入替え処理は，このシステムコール処理において行われる．registserver システムコールは，引継起動フラグが設定されていると，引継起動フラグをリセットし，旧 OS サーバを起床し，旧 OS サーバの入替え処理終了を待つ．その後，起床されると，要求情報と状態情報を引き継ぎ，get システムコールを発行して，サービスの提供を開始する．

このように，本入替えは，OS サーバが発行するシステムコール（registserver システムコールと get システムコール）内の処理として実現されており，OS サーバのプログラムへの影響が非常に少ない特徴を持つ．

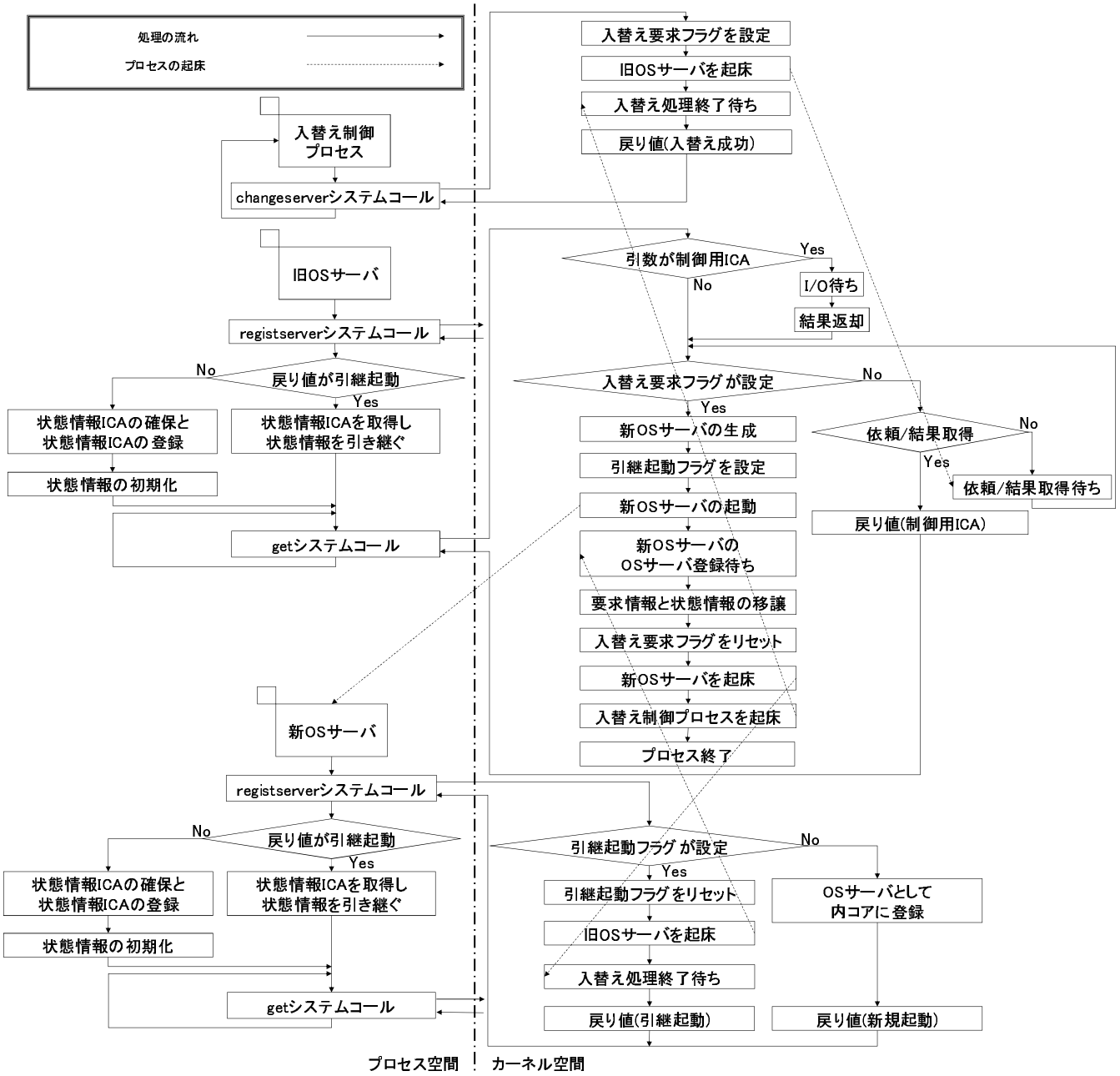

図 2 OS サーバ入替え処理の流れ

3. ドライバプロセス入替え

3.1 ドライバプロセスの処理流れ

ドライバプロセスの処理流れを**図 3** に示す．ドライバプロセスは，registserver システムコールを発行し，自プロセスを OS サーバとして内コアに登録する．次に，状態情報 ICA を確保し，内コアに登録し，状態情報を初期化する．その後，割り込み登録システムコール（setintr システムコール）を発行し，プロセス識別子，仮想空間識別子，機能識別子，および割り込み処理ルーチンのアドレス（以降，割り込み情報と略す）を登録する．最後に，get システムコールを発行して，サービスの提供を開始する．

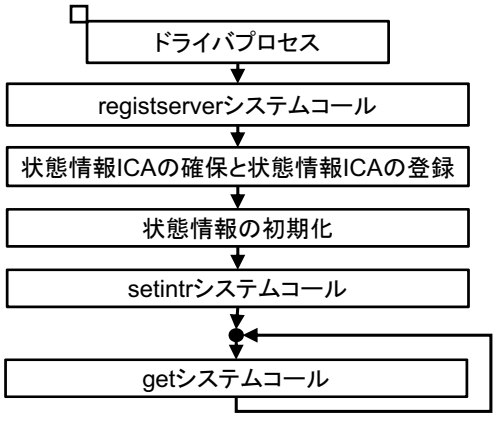

図 3 ドライバプロセスの処理流れ

3.2 課題と対処

OS サーバ入替えの機能をそのままでドライバプロセスに適用しようとすると以下の対処が必要である．

（1）状態情報の引き継ぎの実現

ドライバプロセスで registserver システムコールを発行し，この戻り値を利用して，新規起動か引継起動かを識別する．引継起動の場合，状態情報 ICA を取得し，状態情報を引き継ぐ．

さらに，ドライバプロセス特有の対処として以下の課題に対処が必要である．

（2）入替え中に発生した割り込みの扱い方

入替え処理中の旧ドライバプロセスは割り込み処理を実行できない．このため，入替え処理中に旧ドライバプロセスへの割り込みが発生すると問題となる．この対処として，割り込みの発生を予測可能なドライバプロセスの入替えは，入替え処理の開始を割り込みの発生する期間を終えるまで保留する．また，割り込みの発生を予測不可能なドライバプロセスの入替えは，入替え処理中の割り込みを禁止する．これにより，入替え処理中の旧ドライバプロセスへの割り込みを防止する．

（3）新ドライバプロセス生成に必要なドライバプロセスの入替え機構の確立

入替え処理はプロセス生成処理を伴い，生成するプロセスのデータを取得するため，ディスクドライバプロセスへデータ取得処理依頼を行う．また，既存の OS サーバ入替え機能において，新 OS サーバ生成処理は，旧 OS サーバにより行われる．このため，ディスクドライバプロセスの入替えでは，旧ディスクドライバプロセスが自プロセスに対してプロセスのデータ取得依頼を発行し，データ取得依頼を保持したまま結果返却待ちとなり問題となる．この対処として，新ドライバプロセスを生成するプロセスを旧ドライバプロセスから入替え制御プロセスに変更する．これに伴い，入替え中か否かを示す入替え要求フラグを変更し，プロセス生成中，情報移譲中，または入替え処理中でないことを示す入替え状態にする．

3.3 対処内容

ドライバプロセス入替え処理の流れを**図 4** に示す．以下に対処の内容を説明する．

（1）入替え制御プロセスは，changeserver システムコールを発行して，入替えを行う．ドライバプロセスの入替えのため，新たに入替え状態を導入する．入替え状態は，入替え処理においてプロセス生成中，情報移譲中，または入替え処理中でないことを示す状態である．changeserver システムコールは，入替え状態をプロセス生成中に設定する．その後，新ドライバプロセスを生成し，入替え状態を情報移譲中に設定し，旧ドライバプロセスを起床させ，引継起動フラグ設定を待つ．起床されると，新ドライバプロセスを起動し，入替えの完了を待って，復帰する．ここで，新ドライバプロセスの生成と起動は，既存の入替え処理では旧ドライバプロセスが行っていた．

（2）旧ドライバプロセスは，get システムコールの処理において，入替え処理を行う．get システムコールは，入替え状態が情報移譲中に設定されていると，次の処理を行う．

（A）引継起動フラグを設定し，入替え制御プロセスを起床し，新ドライバプロセスの registserver システムコールからの同期を待つ．

（B）起床されると，要求情報と状態情報を新ドライバプロセスに移譲し，新ドライバプロセスを起床させ，新たに割り込み情報の登録を待つ．

（C）起床されると，入替え状態をリセットする．その後，新ドライバプロセスと入替え制御プロセスを起床し，プロセスを終了する．

（3）新ドライバプロセスは，自プロセスを OS サーバとして内コアに登録するために registserver システムコールを発行する．入替え処理は，このシステムコール処理において行われる．registserver システムコールは，引継起動フラグが設定されていると，引継起動フラグをリセットし，旧ドライバプロセスを起床し，旧ドライバプロセスの情報移譲終了を待つ．その後，起床されると，要求情報と状態情報を引き継ぎ，新たに setintr システムコールを発行する．setintr システムコールは，割り込みテーブルに割り込み情報を登録し，旧ドライバプロセスを起床させ，入替え処理終了を待つ．その後，起床されると，get システムコールを発行して，サービスの提供を開始する．

図 4 に示す処理は，図 2 に示す OS サーバ入替え処理を拡張したものであり，ドライバプロセスの入替えだけではなく OS サーバの入替えにも適用できる．

4. 評価

4.1 評価内容と評価環境

ドライバプロセス入替え機能の評価として，以下の 3 つを評価する．

（評価 1）ドライバプロセス入替えに対処した OS サーバ入替え機能のオーバヘッド

（評価 2）NIC ドライバプロセスの入替えが通信に与える影響

（評価 3）ディスクドライバプロセスの入替えがデータ入

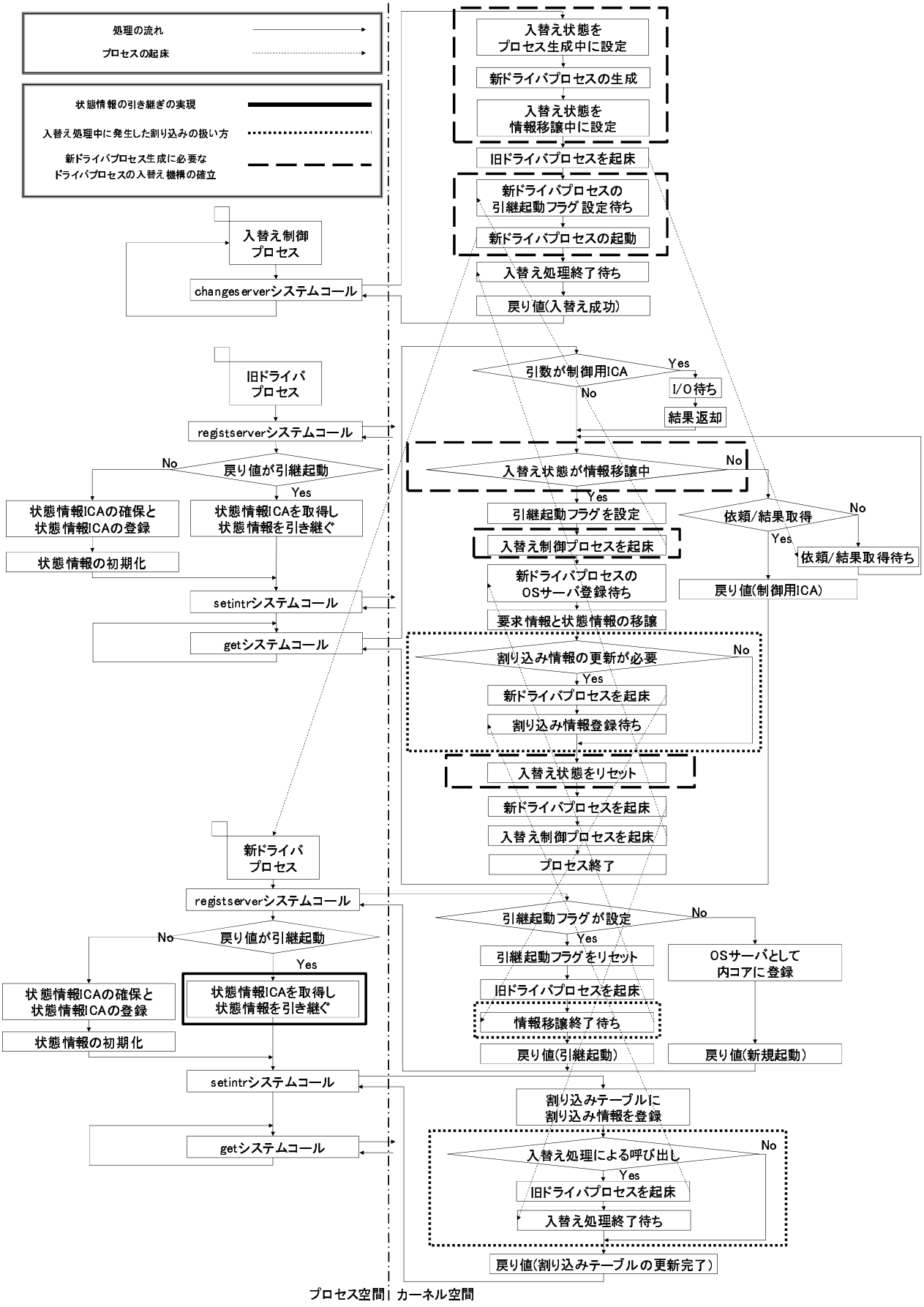

図 4　ドライバプロセス入替え処理の流れ

出力に与える影響

（評価 1）により，OS サーバ入替え機能のドライバプロセス入替えへの対処が入替え処理時間に与える影響を示す．（評価 2）により，実サービスにおいて，NIC ドライバプロセスの入替えが送信処理時間に与える影響を評価し，この原因を考察する．（評価 3）により，実サービスにおいて，ディスクドライバプロセスの入替えがデータ入出力処理時間に与える影響を評価し，この原因を考察する．

なお，評価は，Celeron プロセッサ (2.0GHz) を搭載した計算機を利用し，NIC は Realtek 8139 PCI Ethernet Card，Disk は Maxtor 6Y080L0 を使用した．

4.2 ドライバプロセス入替えに対処した OS サーバ入替え機能のオーバヘッド

ドライバプロセス入替えへの対処前と対処後の OS サーバ入替え処理時間を比較する．なお評価には，各プロトコルヘッダの設定やデータを生成するプロトコル制御部である OS サーバ（通信制御サーバ）を利用する．

測定を 10 回行ったときの 1 回当たりの入替え処理時間の平均は，対処前が 41.66ms，対処後が 42.72ms であり，通信制御サーバの入替え処理時間は 1.06(42.72 - 41.66)ms 増加した．これは，入替え要求フラグから入替え状態への変化に伴う管理する状態数の増加，割り込み処理を行うドライバプロセスか否かの識別，および新 OS サーバを生成するプロセスの変更に伴うプロセス切り替え数の増加によるものである．

入替え処理時間の増加量は，対処前の入替え処理時間の 2.5%であり，オーバヘッドは小さい．これより，OS サーバ入替え機能のドライバプロセスへの対処が既存の OS サーバ入替え処理時間に与える影響は小さいとわかる．

4.3 NIC ドライバプロセス入替え処理が通信に与える影響

4.3.1 測定方法

NIC ドライバプロセスの入替え処理が送信処理時間に与える影響を評価するため，送信処理中に入替え処理を行わない場合と行う場合の送信処理時間を測定する．NIC ドライバプロセス入替え処理の様子を**図 5** に示す．なお，本評価ではデータ送信依頼を発行する AP プロセスは 1 つとし，FreeBSD（4.3-RELEASE）を搭載した計算機と通信を行う．また，各プロセスの優先度は高いプロセスの順に，入替え制御プロセス，NIC ドライバプロセス，通信制御サーバ，AP プロセスである．データの送信処理の流れを以下に述べる．

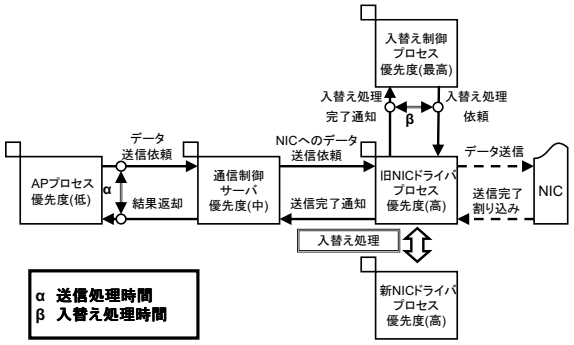

図 5 NIC ドライバプロセス入替え処理の様子

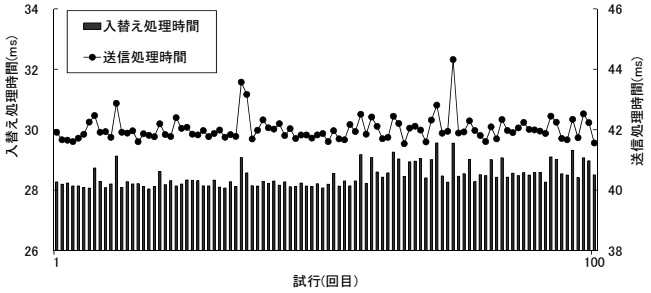

図 6 入替え処理が送信処理時間に与える影響

（1）通信プロトコルは UDP/IP を利用し，送信するデータの大きさは 1024 バイトとする．AP プロセスは通信制御サーバへデータ送信依頼を行い，これを受け，通信制御サーバは NIC ドライバプロセスに NIC へのデータ送信依頼を行う．

（2）NIC ドライバプロセスは，NIC へデータ送信を行い，送信完了割り込み待ちとなる．NIC からの送信完了割り込みを受け，通信制御サーバに送信完了通知を行う．

（3）通信制御サーバは，AP プロセスに結果返却を行う．これを受け，AP プロセスは送信処理を完了とする．

入替え処理を行わない場合，送信処理を 100 回行ったときにかかる送信処理時間（α）を 1 回の試行とし，この試行を 100 回行う．

入替え処理を行う場合，AP プロセスと入替え制御プロセスは同期を取り，送信処理を 100 回行う間に NIC ドライバプロセスを 1 回だけ入替える．これを 1 回の試行とする．なお，入替え開始のタイミングは擬似乱数を利用してランダムにした．この試行を 100 回行い，送信処理時間（α）と入替え処理時間（β）を測定する．

4.3.2 考察

送信処理中に入替え処理を行わない場合の送信処理時間は最小で 15.30ms，平均で 15.46ms，最大で 16.40ms であった．また，AP プロセスが通信制御サーバへデータ送信依頼を行わない状態，つまり通信制御サーバや NIC ドライバプロセスが動作しない状態で NIC ドライバプロセスを 10 回入替えたところ，1 回当たりの入替え処理時間は

最小で 27.57ms，平均で 27.75ms，最大で 28.52ms であった．入替え処理時間の変動は，新 NIC ドライバプロセスを生成したときの磁気ディスク装置の回転待ちやシーク待ちが原因と考えられる．

送信処理中に入替え処理を行う場合の入替え処理時間と送信処理時間を**図 6** に示す．図 6 より，以下のことがわかる．

（1）入替え処理時間は同様

（2）送信処理時間の増加量は大きい

入替え処理時間は最小で 28.04ms，平均で 28.46ms，最大で 29.56ms であった．これは，AP プロセスが通信制御サーバへデータ送信依頼を行わない状態で NIC ドライバプロセスを入替えた場合の入替え処理時間と同様である．これは，入替え処理は，NIC ドライバプロセスの依頼キューに保持されている送信処理依頼に優先して開始されるためである．これより，送信処理が入替え処理時間に与える影響は小さいことがわかる．

送信処理時間は最小で 41.53ms，平均で 42.02ms，最大で 44.32ms であった．NIC ドライバプロセスを入替えない場合の送信処理時間は平均 15.46ms であるため，送信処理時間は最小で 26.07(41.53 - 15.46)ms，最大で 28.86(44.32 - 15.46)ms だけ増加している．送信処理時間の増加量は，入替え処理時間に相当しており，送信処理時間は入替え処理の影響を受けているといえる．送信処理時間が入替え処理の影響を受けるタイミングとして，以下の場合がある．

（1）NIC ドライバプロセスの入替え処理中に，通信制御サーバが NIC ドライバプロセスに対して NIC へのデータ送信依頼を発行した場合

（2）NIC ドライバプロセスの入替え処理中に，NIC からの送信完了割り込みが発生した場合

4.4 ディスクドライバプロセス入替え処理がデータ入出力に与える影響

4.4.1 測定方法

ディスクドライバプロセスの入替え処理がデータ入出力処理時間に与える影響を評価するため，データ入出力処理中に入替え処理を行わない場合と行う場合のデータ入出力処理時間を測定する．ディスクドライバプロセス入替え処理の様子を**図 7** に示す．なお，本評価ではデータ入出力依頼を発行する AP プロセスは 1 つとする．また，各プロセスの優先度は高いプロセスの順に，入替え制御プロセス，ディスクドライバプロセス，ブロック管理部，ファイル管理部，AP プロセスである．データ入出力の処理の流れを以下に述べる．

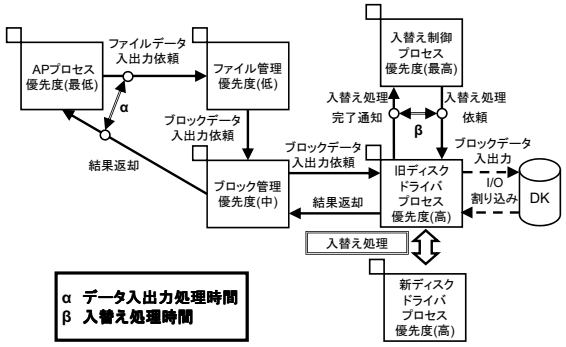

図 7 ディスクドライバプロセス入替え処理の様子

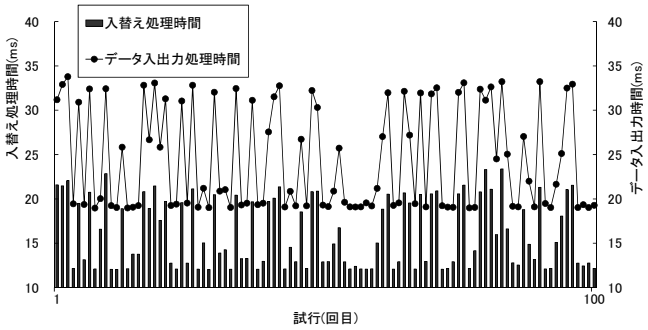

図 8 入替え処理がデータ入出力処理時間に与える影響

（1）入出力ファイルデータの大きさは 4096 バイトである．AP プロセスは，ファイル管理部へファイルデータ入出力依頼を行う．これを受け，ファイル管理部はブロック管理部へブロックデータ入出力依頼を行い，ブロック管理部はこれをディスクドライバプロセスへ転送する．

（2）ディスクドライバプロセスは，ディスクへブロックデータ入出力を行い，I/O 割り込み待ちとなる．ディスクからの割り込みを受け，ブロック管理部へ結果返却を行う．

（3）ブロック管理部は，AP プロセスへ結果返却を行う．これを受け，AP プロセスはデータ入出力処理を完了とする．

入替え処理を行わない場合，データ入出力処理を 10 回行ったときにかかるデータ入出力処理時間（α）を 1 回の試行とし，この試行を 100 回行う．

入替え処理を行う場合，AP プロセスと入替え制御プロセスは同期を取り，データ入出力処理を 10 回行う間にディスクドライバプロセスを 1 回だけ入替える．これを 1 回の試行とする．なお，入替え開始のタイミングは擬似乱数を利用してランダムにした．この試行を 100 回行い，データ入出力処理時間（α）と入替え処理時間（β）を測定する．

4.4.2 考察

データ入出力処理中に入替え処理を行わない場合のデータ入出力処理時間は最小で 18.96ms，平均で 19.15ms，最大で 19.27ms であった．また，AP プロセスがファイル管理サーバへデータ入出力依頼を行わない状態，つまりブ

ロック管理部やディスクドライバプロセスが動作しない状態でディスクドライバプロセスを 10 回入替えたところ，入替え処理時間は最小で 12.03ms，平均で 12.50ms，最大で 13.09ms であった．データ入出力処理時間と入替え処理時間の変動は，磁気ディスク装置の回転待ちやシーク待ちが原因と考えられる．

データ入出力処理中に入替え処理を行う場合の入替え処理時間とデータ入出力処理時間を**図 8** に示す．図 8 より，以下のことがわかる．

（1）入替え処理時間の増加量は大きい

（2）データ入出力処理時間の増加量は大きい

入替え処理時間は最小で 12.03ms，平均で 16.14ms，最大で 23.37ms であった．入替え処理時間が 12ms 程度となるのは，ディスクドライバプロセスが結果返却を行ってから次のデータ入出力処理を行うまでの間に入替え処理依頼を受けた場合である．このとき，入替え処理は即座に開始されるため，入替え処理時間はデータ入出力処理の影響を受けない．入替え処理時間が 16ms 程度となるのは，ディスクドライバプロセスがデータ入出力処理の I/O 割り込み待ち中に入替え処理依頼を受けた場合である．このとき，入替え処理依頼はデータ入出力処理を完了するまで保留され，入替え処理時間は I/O 割り込み待ち時間と結果返却時間だけ増加する．入替え処理時間が 20ms 程度となるのは，ディスクドライバプロセスがデータ入出力処理を開始した直後に入替え処理依頼を受けた場合である．このとき，入替え処理依頼はデータ入出力処理を完了するまで保留され，入替え処理時間はデータ入出力処理時間だけ増加する．さらに，入替え処理において発行されるプロセス生成依頼は，データ入出力処理と競合が起き，入替え処理時間は増加する．また，データ入出力処理中の I/O 割り込み待ち時間が非常に短く，I/O 割り込み待ち中に入替え処理を受ける確率は小さいため，入替え処理時間が 12ms 程度の場合と 20ms 程度の場合で二極化してしまう．

データ入出力処理時間は最小で 18.96ms，平均で 24.37ms，最大で 33.78ms であった．ディスクドライバプロセスを入替えない場合のデータ入出力処理時間は平均 19.15ms であるため，データ入出力処理時間は最大で 14.63(33.78 - 19.15)ms だけ増加している．データ入出力処理時間の増加量は，入替え処理時間に相当しており，データ入出力処理時間は入替え処理の影響を受けているといえる．データ入出力処理時間が入替え処理の影響を受けるタイミングとして，ディスクドライバプロセスの入替え処理中に，ブロック管理部がディスクドライバプロセスに対してブロックデータ入出力依頼を発行した場合がある．

5. おわりに

AnT における OS サーバ入替え機能について，ドライバプロセス入替え処理を実現するための課題と対処，および評価結果を述べた．課題として，状態情報の引き継ぎの実現，入替え中に発生した割り込みの扱い方，および新ドライバプロセス生成に必要なドライバプロセスの入替え機構の確立を挙げ，これらの対処について述べた．

評価では，対処前と対処後の OS サーバ入替え機能の入替え処理時間を比較し，ドライバプロセス入替えへの対処は，入替え処理時間にほとんど影響を与えないことを明らかにした．また，UDP/IP 通信における送信処理時間は，入替え処理時間に相当する時間だけ増加し，入替え処理の影響を受けることを明らかにした．さらに，データ入出力処理中にディスクドライバプロセスを入替えたときのデータ入出力処理時間は，入替え処理時間に相当する時間だけ増加し，入替え処理の影響を受けることを明らかにした．

残された課題として，入替え処理時間の短縮による割り込み禁止時間の短縮がある．

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] D. L. Black, D. B. Golub, D. P. Julin, R. F. Rashid, P. P. Draves, R. W. Dean, A. Forin, J. Barrera, H. Tokuda, G. Malan, and D. Bohman, “Microkernel Operating System Architecture and Mach,” Journal of Information Processing, Vol.14, No.4, pp.442-453, 1992.

[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, and H. Bos, “Can we make operating systems reliable and secure?,” IEEE Computer Magazine, Vol.39, No.5, pp.44-51, 2006.

[4] 谷口 秀夫, 伊藤 健一, 牛島 和夫, “プロセス走行時におけるプログラムの部分入替え法,” 電子情報通信学会論文誌 (D-I), Vol.J78-D-I, No.5, pp.492-499, 1995.

[5] Linux Journal Staff, “Kernel Korner: Dynamic Kernels - Modularized Device Drivers,” Linux Journal, Vol.1996, Issue 23, No.7, 1996.

[6] 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, Vol.J93-D, No.10, pp.1977-1989, 2010.

[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,” 第 18 回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] A. Chou, J. Yang, B. Chelf, S. Hallem and D. Engler “An Empirical Study of Operating Systems Errors,” Proc. of 18th ACM Symposium on Operating Systems Principles, pp.73-88, 2001.