Vol.2010-OS-115 No.6
2010/8/3

セグメント機構を利用した走行モード変更機構の実現

公 文 宏 樹[†1]　谷 口 秀 夫[†1]　横 山 和 俊[†2]

システムコール発行は，プロセスの走行モード変更の処理を伴うため，オーバヘッドが大きい．そこで，オーバヘッドを削減する走行モード変更機構と OS 空間を保護するカーネル保護法を提案した．しかし，仮想空間切り替えが多発するためオーバヘッドが大きい．そこで，オーバヘッドの小さい Intel 社の SYSENTER 命令とセグメント機構を利用し，システムコール処理，I/O 処理，例外処理，および割り込み処理の 4 つの場合についてオーバヘッドを定式化し，実測により定量化する．また，定式化と定量化をもとに有効な走行モードの分岐点を求める．さらに，ドライバを用いて実測し，評価する．

Implementation of Dynamic Running Mode Switch of Application Program by Using Segment Mechanism

Hiroki Kumon,[†1] Hideo Taniguchi[†1]
and Kazutoshi Yokoyama[†2]

System-call has a large overhead with switching the running mode. We have proposed the Dynamic running Mode Switch Mechanism (DMSM) to decrease the overhead and the mechanism of kernel area protection to protect OS area. However, the mechanism of kernel area protection has a large overhead due to change of virtual space many times. In this paper, we use a SYSENTER instruction and a segment mechanism with a small overhead of Intel processors. Next, we formulate each overhead of four processes of system-call, I/O, exception, and interruption, and quantify it by the evaluation. Also, we calculate an effective running mode by the formulation and the quantification, and evaluate this running mode by using drivers.

†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 (株)NTT データ技術開発本部　Research and Development Headquarters, NTT Data

1. はじめに

オペレーティングシステム (以降，OS と略す) において，システムコール発行は，走行モード変更の処理を伴うため，オーバヘッドが大きい．そこで，走行モード変更機構[1)] を提案した．走行モード変更機構は，プロセス走行中に自由に走行モードを変更する機構である．例えば，プロセスの走行モードをスーパバイザモードにすることにより，システムコール発行の際に走行モード変更の処理を行う必要がなく，システムコール処理呼び出しを関数呼び出しの形で行うことができる．このため，オーバヘッドを削減できる．しかし，走行モード変更機構はプロセスをスーパバイザモードで走行させるため，OS 空間の保護機能が低下する．そこで，プロセスのスーパバイザモード走行時に OS 空間を保護するため，カーネル保護法[2)] を提案した．カーネル保護法は，プロセスがユーザモード走行時に利用する仮想空間 (以降，通常仮想空間と略す) と，スーパバイザモード走行時に利用する仮想空間 (以降，特殊仮想空間と略す) の 2 つの仮想空間を 1 つのプロセスに保有させる．これにより，OS 空間の保護機能は向上した．

一方，上記の提案により，スーパバイザモードで走行するプロセスにおいては，システムコール処理，I/O 処理，例外処理，および割り込み処理の際に，通常仮想空間と特殊仮想空間の仮想空間切り替えが発生し，新たなオーバヘッドとなる．これらの走行モード変更機構とカーネル保護法におけるオーバヘッドを明らかにするための定式化と定量化について文献 3) で報告した．これにより，仮想空間切り替えによる保護機構とソフトウェア割り込みによるシステムコール発行のオーバヘッドが大きいことが明らかになった．

ここでは，処理オーバヘッドの小さい Intel 社の SYSENTER 命令によるシステムコール発行とセグメント機構を利用した保護機構を実現し，ソフトウェア割り込みによるシステムコール発行と仮想空間切り替えによる保護機構を含めて 4 つの組み合わせにおける各処理時のオーバヘッドを明らかにする．また，このオーバヘッドを利用して，各走行モードの有効域の分岐点を明らかにする．さらに，実際のドライバを利用した評価を述べる．

2. 走行モード変更機構とカーネル保護法

2.1 走行モード変更機構

AP をユーザモードで実行するプロセスを，以降では，Umode プロセスと呼ぶ．また，AP をスーパバイザモードで実行するプロセスを，以降，Smode プロセスと呼ぶ．Umode プロセスでは，AP を実行する時の走行モードはユーザモードであり，OS を実行する時の走

行モードは，スーパバイザモードである．AP を実行している場合は，ユーザ領域のみが利用可能であり，OS 領域のシステムコール処理や資源を直接に呼び出しあるいは操作することができない．AP から OS への処理依頼は，AP からシステムコールを発行することで実現される．例えば，システムコール xyz() が発行されると AP にリンクされるシステムコールライブラリがソフトウェア割り込み命令を発行し，OS のシステムコール関数 sys_xyz() 処理に移行する．OS からの復帰は，OS がプロセッサに割り込みからの復帰命令を発行することで実現される．以降では，このシステムコールの形態を割り込み型システムコールと呼ぶ．割り込み型システムコールでは，走行モード変更の処理を伴うため，オーバヘッドが大きい．

Smode プロセスでは，AP と OS の両方がスーパバイザモードで実行される．Smode プロセスにおけるシステムコールでは，AP が OS のシステムコール処理である sys_xyz() を，直接に呼び出す．以降では，このシステムコールの形態を関数型システムコールと呼ぶ．関数型システムコールでは，割り込み型システムコールで発生した走行モード変更が不要なため，オーバヘッドの削減が期待できる．

動的走行モード変更機構 DMSM(Dynamic running Mode Switch Mechanism) を用いたプロセスは，モード変更システムコールにより，任意の時点で走行モードが変更される．switch_supervisor システムコールは，Umode プロセスを Smode プロセスに変更する．逆に，switch_user システムコールは，Smode プロセスを Umode プロセスに変更する．つまり，プロセスは，AP の実行に関して，ユーザモードとスーパバイザモードの両方の走行モードを使い分けることができる．言い換えれば，プロセスは，割り込み型システムコールと関数型システムコールを使い分けることができる．このシステムコールの形態を共存型システムコール形態と呼ぶ．

Smode プロセスは，システムコールを関数型システムコールで使用するため，システムコールのオーバヘッドを削減できる．更に，Smode プロセスのユーザ領域は，OS 空間と同様に，ページアウトの対象外としてメモリに常駐する．これにより，Umode プロセスでは発生したユーザ領域のページインとページアウトが Smode プロセスでは発生しない．

2.2 カーネル保護法

OS 空間を保護するために，1 つのプロセスに通常仮想空間と特殊仮想空間の 2 つの仮想空間を保有させる．通常仮想空間と特殊仮想空間では，OS 空間のデータ部分へのアクセス権限が異なる．通常仮想空間ではスーパバイザモードのみで読み書き可能とし，特殊仮想空間ではスーパバイザモードでの読み込み専用としている．

表 1 セグメントレジスタの設定

	UmodeプロセスとOS		Smodeプロセス	
仮想空間	コードセグメント	データセグメント	コードセグメント	データセグメント
ユーザ空間	読み込み専用	読み書き可能	読み込み専用	読み書き可能
OS空間(手続き部分)				
OS空間(データ部分)				

Umode プロセスが走行する際は，通常仮想空間を利用する．これにより，Umode プロセスは，既存の仮想空間アクセス制御を可能にする．Smode プロセスの場合，AP を実行する際には特殊仮想空間を利用し，システムコールにより OS を実行する際には通常仮想空間を利用する．これにより，Smode において AP を実行する際に OS 空間を保護できる．なお，2 つの仮想空間の切り替え契機は以下のようになる．

(1) Umode プロセスの場合，切り替えは発生しない．

(2) Smode プロセスの場合，以下の場合において，特殊仮想空間から通常仮想空間，または通常仮想空間から特殊仮想空間への切り替えが発生する．

- (A) システムコール処理時
- (B) AP 実行中における割り込みや例外の発生

(3) 走行モード切り替えの際，切り替えが発生する．

3. セグメント機構を用いた OS 空間の保護

2.2 節で示したカーネル保護法は仮想空間を切り替えることにより OS 空間の保護を実現している．しかし，仮想空間の切り替えの処理は大きなオーバヘッドとなる．このため，Smode プロセスの有効性は低下する．

ここで，処理オーバヘッドを小さくするために，Intel 社の CPU アーキテクチャに実装されているセグメント機構を用いた OS 空間の保護を実現した．OS 空間の保護を実現するために必要なセグメントレジスタの設定の詳細を**表 1** に示す．

Umode プロセスと OS の処理は，保護を行う必要がないため，コードセグメントは仮想空間の全範囲を読み込み専用とした．また，データセグメントは仮想空間の全範囲を読み書き可能とする．Smode プロセスは，システムコール処理などの受付処理を行えるようにするため，コードセグメントはユーザ空間と OS 空間の手続き部分を読み込み専用とする．また，Smode プロセスはユーザ空間走行中に OS 空間のデータを扱う必要がないため，デー

タセグメントはユーザ空間のみ読み書き可能とする．

Smode プロセスがシステムコールを発行する場合や割り込みの発生により OS 空間の処理を行う場合は，それぞれの処理の受付処理においてセグメントを切り替えることにより OS 空間の処理を行うことが可能となる．

以上のセグメントの設定により，Smode プロセスがユーザ空間走行中の OS 空間の保護を可能とする．

4. 適 用 域

4.1 走行モード変更を伴う処理

プロセスをスーパバイザモードで走行させる方式は，OS の機能を利用して，高速な処理を行うプログラムに対して有効である．特に，マイクロカーネル構造[4)5)6)] の OS において有効と考えられる．ここでは，ドライバ処理をプロセスとして実現するドライバプロセス機能[7)] を取り上げ，スーパバイザモード走行させることが有効な適用域を明確化する．

走行モードの変更は，システムコール処理，I/O 処理，例外処理，および割り込み処理で発生する．そこで，これらについて，Umode プロセス，OS 空間を保護しない Smode(Smode-nP) プロセス，および OS 空間を保護する Smode(Smode-P) プロセスの処理移行のオーバヘッドを明らかにする．

4.2 定 式 化

処理移行のオーバヘッドは，OS 空間の保護方式とシステムコールの発行方式で異なる．そこで，OS 空間の保護方式として仮想空間切り替えとセグメント切り替え，システムコールの発行方式としてソフトウェア割り込みと SYSENTER 命令を取り上げ，4 つの組み合わせでオーバヘッドを定式化する．以降では，仮想空間切り替えと SYSENTER 命令利用を (仮－S) 法と略す．同様に，仮想空間切り替えとソフトウェア割り込みの組み合わせを (仮－ソ) 法，セグメント切り替えと SYSENTER 命令利用を (セ－S) 法，セグメント切り替えとソフトウェア割り込み利用を (セ－ソ) 法と略す．

各オーバヘッドを以下のように定義する．

- O_t: ソフトウェア割り込みと復帰のオーバヘッド
- O_s: SYSENTER 命令と復帰のオーバヘッド
- O_{vm}: 仮想空間切り替えのオーバヘッド
- O_{seg}: セグメント切り替えのオーバヘッド
- O_{sys}: I/O 処理におけるシステムコール処理のオーバヘッド

表 2 システムコール処理 1 回あたりのオーバヘッド

	システムコール処理			
走行中プロセス	(仮－S)法	(仮－ソ)法	(セ－S)法	(セ－ソ)法
Umode	Os	Ot	Os	Ot
Smode-nP	0	0	0	0
Smode-P	Ovm×2	Ovm×2	Oseg×2	Oseg×2

Ot：ソフトウェア割り込みと復帰のオーバヘッド　Ovm：仮想空間切り替えのオーバヘッド
Os：SYSENTER命令と復帰のオーバヘッド　Oseg：セグメント切り替えのオーバヘッド

表 3 I/O 処理 1 回あたりのオーバヘッド

	I/O処理			
走行中プロセス	(仮－S)法	(仮－ソ)法	(セ－S)法	(セ－ソ)法
Umode	Os + Osys	Ot + Osys	Os + Osys	Ot + Osys
Smode-nP	0	0	0	0
Smode-P	0	0	0	0

Ot：ソフトウェア割り込みと復帰のオーバヘッド
Os：SYSENTER命令と復帰のオーバヘッド
Osys：I/O処理におけるシステムコール処理のオーバヘッド

このとき，システムコール処理，I/O 処理，および例外処理の時のオーバヘッドを**表 2**，**表 3**，および**表 4** に示す．システムコール処理の時は，Umode プロセスの場合，システムコールの発行方式に対応するオーバヘッドが発生する．また，Smode-nP プロセスの場合，OS 空間の保護をせず，システムコールは関数呼び出しの形で呼び出されるため，オーバヘッドは発生しない．Smode-P プロセスの場合，OS 空間の保護方式に対応するオーバヘッドが発生する．I/O 処理の時は，Umode プロセスの場合，システムコールの発行方式に対応するオーバヘッドに加えて，全ての方式で I/O 処理におけるシステムコール処理のオーバヘッドが発生する．Smode-nP プロセスと Smode-P プロセスは直接 I/O を発行できるためオーバヘッドは発生しない．例外処理の時は，Umode プロセスと Smode-nP プロセスの場合，オーバヘッドは発生しない．Smode-P プロセスの場合，OS 空間の保護方式に対応するオーバヘッドが発生する．

次に，割り込み処理の時のオーバヘッドについて述べる．割り込み時の処理の流れは，走行中プロセス，割り込み受付処理，割り込み処理，割り込み復帰処理，および走行中プロセ

表 4 例外処理 1 回あたりのオーバヘッド

	例外処理			
走行中プロセス	(仮－S)法	(仮－ソ)法	(セ－S)法	(セ－ソ)法
Umode	0	0	0	0
Smode-nP	0	0	0	0
Smode-P	Ovm×2	Ovm×2	Oseg×2	Oseg×2

Ovm：仮想空間切り替えのオーバヘッド
Oseg：セグメント切り替えのオーバヘッド

スへの復帰という順番である．これらの処理において，走行中プロセスは，Umode プロセスまたは Smode プロセスであり，いずれの場合も，割り込み発生時に実行しているプログラムの位置は，OS 空間またはユーザ空間である．また，割り込み受付処理と割り込み復帰処理は，OS 空間における処理である．さらに，割り込み処理は，OS 空間またはユーザ空間 (ドライバプロセスが保有する場合) で実行される．ここで，ユーザ空間で実行される際は，Umode プロセスの場合と Smode プロセスの場合がある．以上のことから，走行中プロセスと割り込み処理を実行するプロセスが異なるか否かによって，仮想空間切り替えの要否が決定される．また，割り込み処理を実行するプロセスが Smode-P プロセスか否かによって，セグメント切り替えの要否が決定される． つまり，割り込み処理時のオーバヘッドは，システムコール処理の発行方式に関連なく，OS 空間の保護方式にのみ依存する．OS 空間の保護方式が仮想空間切り替えの場合，オーバヘッドは以下の状況で発生する．

(1) 特殊仮想空間から通常仮想空間への切り替えは,「走行中プロセスが Smode-P プロセスでかつ実行プログラムの位置がユーザ空間の場合において，制御が走行中プロセスから割り込み受付処理に移行する時」，および「割り込み処理を実行するプロセスがユーザ空間 (特殊) の場合において，制御が割り込み処理から割り込み復帰処理に移行する時」である．

(2) 通常仮想空間から特殊仮想空間への切り替えは,「割り込み処理をユーザ空間 (特殊) で実行する場合において，制御が割り込み受付処理から割り込み処理に移行する時」および,「走行中プロセスが Smode-P プロセスでかつ実行プログラムの位置がユーザ空間の場合において，制御が割り込み復帰処理から走行中プロセスへ復帰する時」である．

(3) 通常仮想空間から通常仮想空間への切り替えは,「割り込み処理を実行するプロセスが走行中プロセスと異プロセスでユーザ空間 (通常) の場合において，制御が割り込み受付処理から割り込み処理または割り込み処理から割り込み復帰処理へ移行する時」である．また，OS 空間の保護方式がセグメント切り替えの場合，セグメント切り替えは以下の状況で発生する．

表 5 割り込み処理 1 回あたりの仮想空間切り替えとセグメント切り替えの回数

	走行中プロセス	実行プログラムの位置	同異	仮想空間切り替え方式		セグメント切り替え方式		
				割り込み処理を実行するプロセスの空間	仮想空間切り替え	割り込み処理を実行するプロセスの空間	仮想空間切り替え	セグメント切り替え
(1)	Umode Smode·nP	OS空間	同プロセス	ユーザ空間(通常)	0回	ユーザ空間 (Umode，Smode·nP)	0回	0回
(2)				OS空間	0回	OS空間	0回	0回
(3)		ユーザ空間		ユーザ空間(通常)	0回	ユーザ空間 (Umode，Smode·nP)	0回	0回
(4)				OS空間	0回	OS空間	0回	0回
(5)		OS空間	異プロセス	ユーザ空間(通常)	2回	ユーザ空間 (Umode，Smode·nP)	2回	0回
(6)				ユーザ空間(特殊)	2回	ユーザ空間 (Smode·P)	2回	2回
(7)				OS空間	0回	OS空間	0回	0回
(8)		ユーザ空間		ユーザ空間(通常)	2回	ユーザ空間 (Umode，Smode·nP)	2回	0回
(9)				ユーザ空間(特殊)	2回	ユーザ空間 (Smode·P)	2回	2回
(10)				OS空間	0回	OS空間	0回	0回
(11)	Smode·P	OS空間	同プロセス	ユーザ空間(特殊)	2回	ユーザ空間 (Smode·P)	0回	2回
(12)				OS空間	0回	OS空間	0回	0回
(13)		ユーザ空間		ユーザ空間(特殊)	4回	ユーザ空間 (Smode·P)	0回	2回
(14)				OS空間	2回	OS空間	0回	0回
(15)		OS空間	異プロセス	ユーザ空間(通常)	2回	ユーザ空間 (Umode，Smode·nP)	2回	0回
(16)				ユーザ空間(特殊)	2回	ユーザ空間 (Smode·P)	2回	2回
(17)				OS空間	0回	OS空間	0回	0回
(18)		ユーザ空間		ユーザ空間(通常)	4回	ユーザ空間 (Umode，Smode·nP)	2回	0回
(19)				ユーザ空間(特殊)	4回	ユーザ空間 (Smode·P)	2回	2回
(20)				OS空間	2回	OS空間	0回	0回

(1) OS 用のセグメントから Smode-P プロセス用のセグメントへの切り替えは,「割り込み処理をユーザ空間 (Smode-P) で実行する場合において，制御が割り込み受付処理から割り込み処理に移行する時」である．

(2) Smode-P プロセス用のセグメントから OS 用のセグメントへの切り替えは,「割り込み処理をユーザ空間 (Smode-P) で実行する場合において，制御が割り込み処理から割り

表 6 割り込み処理 1 回あたりのオーバヘッド

	割り込み処理			
走行中プロセス	(仮－S)法	(仮ーソ)法	(セーS)法	(セーソ)法
Umode	Ovm×Puintr×2×Pdif	Ovm×Puintr×2×Pdif	(Ovm×Puintr＋Oseg×Pusintr)×2×Pdif	(Ovm×Puintr＋Oseg×Pusintr)×2×Pdif
Smode-nP				
Smode-P	Ovm×(Purun＋Puintr)×2	Ovm×(Purun＋Puintr)×2	(Ovm×Puintr×Pdif ＋ Oseg×Pusintr)×2	(Ovm×Puintr×Pdif ＋ Oseg×Pusintr)×2

Ovm：仮想空間切り替えオーバヘッド
Oseg：セグメント切り替えオーバヘッド
Pdif：実行プログラムと割り込み処理を実行するプロセスが異なる確率
Purun：実行プログラムをユーザ空間で実行する確率
Puintr：割り込み処理をユーザ空間で実行する確率
Pusintr：割り込み処理をユーザ空間(Smode-P)で実行する確率

込み復帰処理に移行する時」である．

ここで，割り込み処理を実行するプロセスが走行中プロセスと異プロセスでユーザ空間の場合，仮想空間切り替えが発生する．なお，Smode-nP プロセスは Umode プロセスと同様になる．

以上をふまえ，割り込み処理 1 回あたりの仮想空間切り替えとセグメント切り替えの回数を**表 5** に示す．なお，ここでのセグメント切り替えの回数とは，走行モード変更機構の処理に伴って発生するセグメント切り替えの回数である．つまり，割り込み処理発生時にハードウェアで自動的に行われるセグメントの切り替えは含まれていない．ここで，各確率を以下のように定義する．

- P_{dif}: 実行プログラムと割り込み処理を実行するプロセスが異なる確率
- P_{urun}: 実行プログラムをユーザ空間で実行する確率
- P_{uintr}: 割り込み処理をユーザ空間で実行する確率
- P_{usintr}: 割り込み処理をユーザ空間 (Smode-P) で実行する確率

このとき，割り込み処理 1 回あたりのオーバヘッドは**表 6** になる．

さらに，各処理の回数を以下のように定義する．

- N_{sys}: システムコールの発行回数
- N_{io}: I/O 命令の発行回数
- N_{intr}: 割り込みの回数

これにより，走行モード変更機構のオーバヘッドは**表 7** になる．なお，ドライバプロセスの動作は正常と仮定し，例外は発生しないものとする．

4.3 オーバヘッドの定量化による走行モード変更機構の有効性

走行モード変更機構，SYSENTER 命令，仮想空間切り替え，およびセグメント切り替えの機能を実現した ***AnT*** オペレーティングシステム[8] を利用し，Pentium4(2.4GHz) を搭載した計算機でオーバヘッドを定量化した．測定には RDTSC 命令を使用し，測定値は 10 回の処理時間の平均である．

表 8 定量化したオーバヘッド

定数	説明	処理時間[クロック]
Ot	ソフトウェア割り込みと復帰のオーバヘッド	795
Os	SYSENTER命令と復帰のオーバヘッド	18
Ovm	空間切り替えのオーバヘッド	1217
Oseg	セグメント切り替えのオーバヘッド	125
Osys	I/O処理におけるシステムコール処理のオーバヘッド	439

この環境で，ソフトウェア割り込みと復帰のオーバヘッド (O_t) は約 795 クロック，仮想空間切り替えのオーバヘッド (O_{vm}) は約 1217 クロック，I/O 処理におけるシステムコール処理のオーバヘッド (O_{sys}) は約 439 クロックである．[3] また，同環境で処理が非常に短い getpid() システムコールを (セーS) 法で測定を行い，SYSENTER 命令と復帰のオーバヘッド (O_s) とセグメント切り替えのオーバヘッド (O_{seg}) を求めた．測定より，Umode プロセスは約 455 クロック，Smode-nP プロセスは約 437 クロック，Smode-P プロセスは約 686 クロックとなった．4.2 節の表 2 で示した (セーS) 法のシステムコール処理の定式化より，SYSENTER 命令と復帰のオーバヘッド (O_s) は Umode プロセスと Smode-nP プロセスのクロック数の差分であり，以下のようになる．

$$O_s = 455 - 437 = 18 \quad (1)$$

同様に，セグメント切り替えのオーバヘッド (O_{seg}) は 4.2 節の表 2 で示した (セーS) 法のシステムコール処理の定式化より，Smode-nP プロセスと Smode-P プロセスのクロック数の差分であり，以下のようになる．

$$O_{seg} = \frac{686 - 437}{2} = 124.5 \quad (2)$$

以上より，定量化したオーバヘッドを**表 8** に示す．

次に，走行モード変更機構の有効性について論じる．表 7 より，各方式において，オーバヘッドが等しい場合が各走行モードの優劣の分岐点である．また，このことに加え，オーバヘッドを定量化した表 8 を利用して有効域を整理する．

Umode プロセスと Smode-nP プロセスの分岐点を**表 9** に示す．表 9 より，傾きは負で

表 7 走行モード変更機構のオーバヘッド

方式	オーバヘッド		
	Umode	Smode-nP	Smode-P
(仮－S)法	Os×Nsys+(Os+Osys)×Nio+Ovm×Puintr ×2×Pdif×Nintr	Ovm×Puintr×2×Pdif×Nintr	Ovm×2×Nsys+Ovm×(Purun+Puintr) ×2×Nintr
(仮－ソ)法	Ot×Nsys+(Ot+Osys)×Nio+Ovm×Puintr ×2×Pdif×Nintr	Ovm×Puintr×2×Pdif×Nintr	Ovm×2×Nsys+Ovm×(Purun+Puintr) ×2×Nintr
(セ－S)法	Os×Nsys+(Os+Osys)×Nio+(Ovm×Puintr +Oseg×Pusintr)×2×Pdif×Nintr	(Ovm×Puintr+Oseg×Pusintr) ×2×Pdif×Nintr	Oseg×2×Nsys+(Ovm×Puintr×Pdif+ Oseg×Pusintr)×2×Nintr
(セ－ソ)法	Ot×Nsys+(Ot+Osys)×Nio+(Ovm×Puintr +Oseg×Pusintr)×2×Pdif×Nintr	(Ovm×Puintr+Oseg×Pusintr) ×2×Pdif×Nintr	Oseg×2×Nsys+(Ovm×Puintr×Pdif+ Oseg×Pusintr)×2×Nintr

Nsys：システムコールの発行回数
Nio：I/O命令の発行回数
Nintr：割り込みの回数
Ot：ソフトウェア割り込みと復帰のオーバヘッド
Os：SYSENTER命令と復帰のオーバヘッド
Ovm：仮想空間切り替えのオーバヘッド
Oseg：セグメント切り替えのオーバヘッド
Osys：I/O処理におけるシステムコール処理のオーバヘッド
Pdif：実行プログラムと割り込み処理を実行するプロセスが異なる確率
Purun：実行プログラムをユーザ空間で実行する確率
Puintr：割り込み処理をユーザ空間で実行する確率
Pusintr：割り込み処理をユーザ空間(Smode-P)で実行する確率

表 9 Umode プロセスと Smode-nP プロセスにおける各方式の有効域の分岐点

方式	有効域の分岐点
(仮－S)法	$N_{io} = -\frac{18}{457} \times N_{sys}$
(仮－ソ)法	$N_{io} = -\frac{795}{1234} \times N_{sys}$
(セ－S)法	$N_{io} = -\frac{18}{457} \times N_{sys}$
(セ－ソ)法	$N_{io} = -\frac{795}{1234} \times N_{sys}$

Nsys：システムコールの発行回数
Nio：I/O命令の発行回数

あるため，Smode-nP プロセスは Umode プロセスに対し，常に有効であるといえる．

Umode プロセスと Smode-P プロセスの分岐点を**表 10** に示す．表 10 は以下の一般式で表せる．

$$N_{io} = a \times N_{sys} + b \times N_{intr} \quad (3)$$

この式をグラフに表したものを**図 1** に示す．図 1 において，分岐点の上側の場合は Smode-P プロセスで走行させることが有効，分岐点の下側の場合は Umode プロセスで走行させることが有効である．

表 10 Umode プロセスと Smode-P プロセスにおける各方式の有効域の分岐点

方式	有効域の分岐点
(仮－S)法	$N_{io} = \frac{2416}{457} \times N_{sys} + \frac{2434 \times (P_{urun} + P_{uintr} \times (1 - P_{dif}))}{457} \times N_{intr}$
(仮－ソ)法	$N_{io} = \frac{1634}{1234} \times N_{sys} + \frac{2434 \times (P_{urun} + P_{uintr} \times (1 - P_{dif}))}{1234} \times N_{intr}$
(セ－S)法	$N_{io} = \frac{232}{457} \times N_{sys} + \frac{250 \times P_{usintr} \times (1 - P_{dif})}{457} \times N_{intr}$
(セ－ソ)法	$N_{io} = -\frac{545}{1234} \times N_{sys} + \frac{250 \times P_{usintr} \times (1 - P_{dif})}{1234} \times N_{intr}$

Nsys：システムコールの発行回数
Nio：I/O命令の発行回数
Nintr：割り込みの回数
Pdif：実行プログラムの位置と割り込み処理の位置が異プロセスである確率
Purun：実行プログラムの位置がユーザ空間である確率
Puintr：割り込み処理の位置がユーザ空間である確率
Pusintr：割り込み処理の位置がユーザ空間(Smode-P)である確率

Vol.2010-OS-115 No.6
2010/8/3

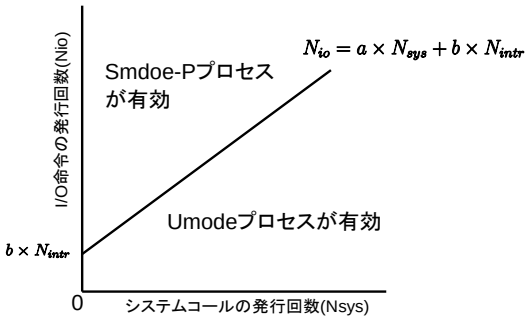

図 1 Umode プロセスと Smode-P プロセスの分岐点

表 11 ドライバプロセスの各処理の回数

		システムコール	I/O命令	割り込み
ディスクドライバプロセス (1セクタ)	Read処理	3回	7回	1回
NICドライバプロセス (パケット分割なし)	送信処理	2回	2回	1回

5. 事例評価

5.1 予　　想

4.3 節では定式化と定量化を用いて各方式の有効な走行モードの分岐点を求めた．ここでは，この有効域の分岐点を評価する．具体的な事例で評価するために，***AnT*** オペレーティングシステムにプロセスとして実現されているディスクドライバプロセスと NIC ドライバプロセスを利用する．各ドライバプロセスのシステムコール回数，I/O 回数，および割り込み回数を**表 11** に示す．

この際の各方式の分岐点は，表 10 で $P_{dif} = 0$，$P_{urun} = 0$，$P_{uintr} = 1$，および $P_{usintr} = 1$ の場合，つまり走行中プロセスが実行しているプログラムが OS 空間，割り込み処理が同プロセスのユーザ空間に存在する場合について，図 1 に示した様な分岐点を**図 2** に示す．なお，表 11 より割り込み 1 回の場合を示している．図中の●と▲は表 11 に基づいたドライバプロセスの位置を表している．図 2 より，以下のことが予想できる．

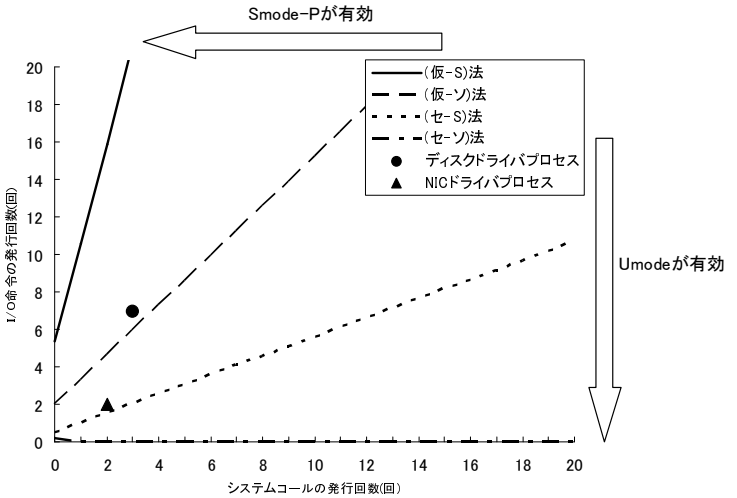

図 2 割り込み 1 回の場合の各ドライバプロセスの有効性 (走行中プロセスの実行しているプログラムが OS 空間，割り込み処理が同プロセスのユーザ空間)

表 12 各方式における有効な走行モード

	(仮－S)法	(仮－ソ)法	(セ－S)法	(セ－ソ)法
ディスクドライバプロセス	0.995(Smode-P)	1.011(Umode)	1.008(Umode)	1.016(Umode)
NICドライバプロセス	0.948(Smode-P)	0.956(Smode-P)	1.002(Umode)	1.016(Umode)

数字は「Umodプロセスにおける処理時間/Smode-Pプロセスにおける処理時間」を表している．
値が1.000以上ならSmode-Pが有効，1.000未満ならUmodeが有効である．

(1)　ディスクドライバプロセスは，(仮－S) 法では Umode プロセスが有効であるが，それ以外の方式では Smode-P プロセスが有効

(2)　NIC ドライバプロセスは (仮－S) 法と (仮－ソ) 法では Umode プロセスが有効であるが，それ以外の方式では Smode-P プロセスが有効

5.2 実 測 結 果

測定は，4.3 節の定量化と同様の環境で行った．ディスクドライバプロセスでは 1 セクタ (512 バイト) の Read 処理，NIC ドライバプロセスでは UDP/IP 通信の送信処理 (512 バ

イト) についてそれぞれの処理時間を測定した．

測定した処理時間を用いて各ドライバプロセスに対して，それぞれの方式で「Umode プロセスにおける処理時間/Smode-P プロセスにおける処理時間」を計算した結果を**表 12** に示す．この値が 1.000 以上なら Smode-P が有効，1.000 未満なら Umode が有効となる．

表 12 より，以下のことがわかる．

(1) ディスクドライバプロセスは，(仮－S) 法では Umode プロセスが有効であるが，それ以外の方式では Smode-P プロセスが有効

(2) NIC ドライバプロセスは (仮－S) 法と (仮－ソ) 法では Umode プロセスが有効であるが，それ以外の方式では Smode-P プロセスが有効

これは，5.1 節で述べた予想と一致しているため，それぞれの方式の各走行モードの有効域の分岐点は正しいことがわかる．

また，この結果より，OS 空間の保護にセグメント切り替えを利用すると必ず Smode-P プロセスが有効となっている．このため，ディスクドライバプロセスと NIC ドライバプロセスのような割り込み 1 回のみのドライバは，セグメント切り替えを用いると Smode-P プロセスが有効になる可能性が高いと考えられる．

6. おわりに

システムコール発行に SYSENTER 命令，OS 空間の保護にセグメント切り替えを実現し，ソフトウェア割り込みと仮想空間切り替えを含めて，4 つの組み合わせにおける走行モード変更機構と OS 空間の保護のオーバヘッドを定式化することにより処理に与える影響を明らかにした．

また，走行モード変更機構，SYSENTER 命令，仮想空間切り替え，およびセグメント切り替えの機能を実現した ***AnT*** オペレーティングシステムを利用し，Pentium4(2.4GHz) を搭載した計算機で各オーバヘッド要因を定量化した．この結果，SYSENTER 命令と復帰のオーバヘッドは約 18 クロック，セグメント切り替えのオーバヘッドは約 125 クロックであった．

定量化したオーバヘッドをもとに各方式における有効な走行モードの有効域の分岐点を明らかにした．これにより，Umode プロセスと Smode-nP プロセスを比較した場合，Smode-nP プロセスは Umode プロセスに対し常に有効であることを明らかにした．

また，Umode プロセスと Smode-P プロセスの有効域を評価するため，***AnT*** オペレーティングシステムに実現されているディスクドライバプロセスと NIC ドライバプロセスを利用して評価を行った．これにより Umode プロセスと Smode-P プロセスを比較した場合，OS 空間の保護にセグメント切り替えを利用すると Smode-P プロセスが有効になる可能性があると考えられる．

参 考 文 献

1) 横山和俊，乃村能成，谷口秀夫，丸山勝巳，“応用プログラムの走行モード変更を可能にするプロセス制御機構，” 電子情報通信学会論文誌 (D)，vol.J91-D，no.3，pp.696-708，2008.
2) 仁科匡人，梅本昌典，谷口秀夫，横山和俊，“***AnT*** における走行モード変更機構でのカーネル保護法，” 電子情報通信学会技術研究報告 (CPSY2007-34)，pp.57-62，2007.
3) 公文宏樹，谷口秀夫，横山和俊，“走行モード変更機構のオーバヘッド分析，” 情報処理学会研究報告，vol.2009-OS-112，2009.
4) J. Liedtke，“Toward Real Microkernels，” Commnuications of The ACM，vol.39，Issue9，pp.70-77，1996.
5) S. Tanenbaum，N. Herder，and H. Bos，“Can we make operating systems reliable and se-cure?，” IEEE Computer Magazine，vol.39，no.5，pp.44-51，2006.
6) 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，and D. Bohman，“MicrokernelOperating System Architecture and Mach，” Journal of Information Processing，vol.14，no.4，pp.442-453，1992.
7) A.S. Tanenbaum，J.N. Herder，H. Bos，B.Gras，and P.Homburg，“Modular System Programming in MINIX 3，” The USENIX Magazine，vol.31，no.2，pp.19-28，2006.
8) 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，” 情報処理学会研究報告，vol.2006-OS-103，pp.71-78，2006.