Vol.2009-OS-112 No.15
2009/8/6

走行モード変更機構のオーバヘッド分析

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，および 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，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する．また，実測により定量化し，オーバヘッドを分析する．

Overhead Analysis of Dynamic Running Mode Switch of Application Program

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 a mechanism of kernel area protection to protect OS area. However, the mechanism of kernel area protection has a large overhead to change virtual space many times. In this paper, we formulate the overhead of each process, such as system-call, I/O, exception and interruption. Also, we quantify and analyze these overheads by the evaluation.

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．また，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．

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，特殊仮想空間と通常仮想空間の切り替えが発生し，新たなオーバヘッドになる．

ここでは，走行モード変更機構について，各処理時のオーバヘッドを明らかにする．

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の基本動作

2.1.1 ユーザモードプロセスとスーパバイザモードプロセス

応用プログラム (以降，AP と略す) 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 プロセスと Smode プロセスの構造を**図 1** に示す．図 1(a) は，Umode プロセスの構造を示している．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出来ない．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．

図 1(b) は，Smode プロセスの様子を示している．Smode プロセスでは，AP と OS の両

†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

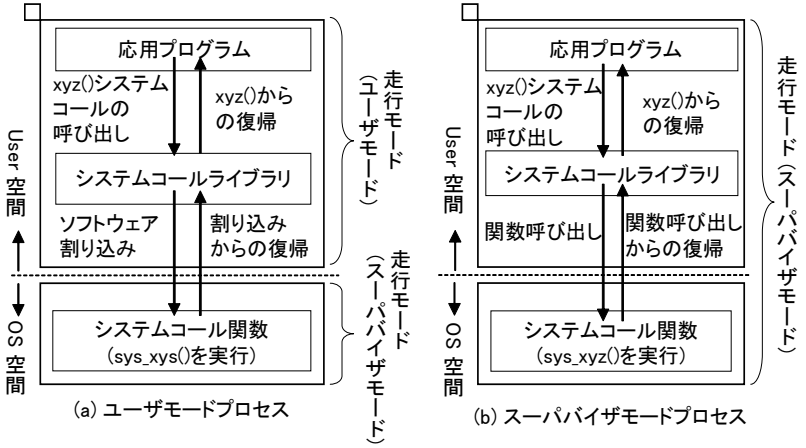

図 1 ユーザモードプロセスとスーパバイザモードプロセス

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．

2.1.2 動的走行モード変更機構

動的走行モード変更機構 DMSM(Dynamic running Mode Switch Mechanism) を用いたプロセス実行の概念を**図 2** に示し，以降で説明する．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

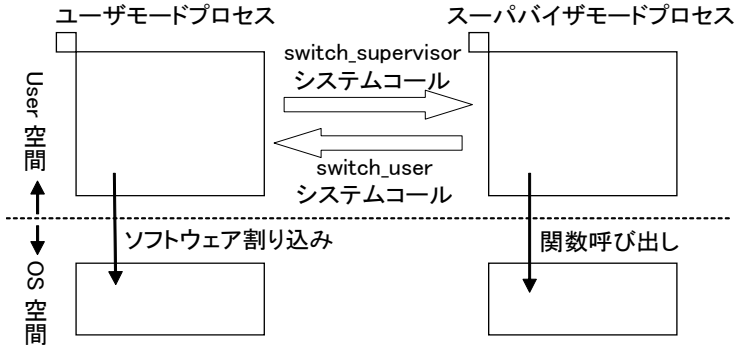

図 2 動的走行モード変更機構の概要

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

カーネルを保護するために，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．この様子を**図 3** に示す．

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．

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

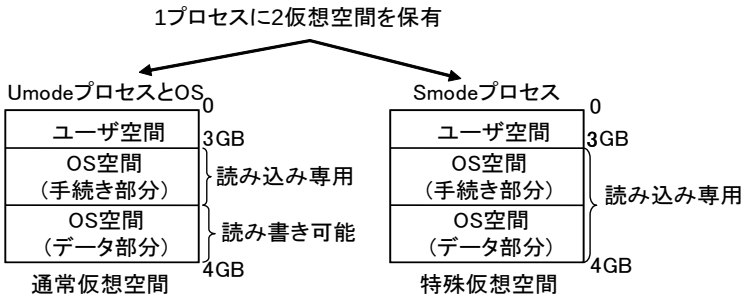

図 3 通常仮想空間と特殊仮想空間

(3) 走行モード切り替えの際，切り替えが発生する．

3. オーバヘッドの分析

3.1 定 式 化

3.1.1 走行モード変更の伴う処理

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．そこで，マイクロカーネル構造[3)4)5)] の OS において，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[6)] について，走行モード変更機構とカーネル保護法が処理に与える影響を考察する．

OS は，以下の 4 つの場合に走行モードを変更する場合がある．

(1) システムコール処理
(2) I/O 処理
(3) 例外処理
(4) 割り込み処理

これらは，走行モード変更機構を利用するか否かにより，処理のオーバヘッドが変化する．そこで，これらの 4 つの場合に対し，Umode プロセス，Smode プロセスにおいてカーネル保護を利用していない場合，および Smode プロセスにおいてカーネル保護を利用している場合の 3 つの場合を考え，オーバヘッドの定式化を行う．

なお，以降では，Smode プロセスにおいてカーネル保護を利用していない場合を Smode プロセス (保護無) と記述し，Smode プロセスにおいてカーネル保護を利用している場合を Smode プロセス (保護有) と記述する．

表 1 システムコール処理 1 回あたりのオーバヘッド

走行中プロセス	システムコール処理
Umodeプロセス	SWintr
Smodeプロセス (保護無)	0
Smodeプロセス (保護有)	VMchg×2

SWintr：ソフトウェア割り込みと復帰のオーバヘッド
VMchg：空間切り替えのオーバヘッド

3.1.2 システムコール処理

Umode プロセスは，1 回のシステムコール発行に対し，1 回のソフトウェア割り込みと復帰のオーバヘッドが発生する．Smode プロセス (保護無) は，システムコール発行を関数呼び出しで行うため，オーバヘッドは発生しない．Smode プロセス (保護有) は，1 回のシステムコール発行に対し，2 回の空間切り替えのオーバヘッドが発生する．

以上のことから，システムコール処理 1 回あたりのオーバヘッドを**表 1** に示す．表 1 において，SWintr はソフトウェア割り込みと復帰のオーバヘッド，VMchg は空間切り替えのオーバヘッドである．

3.1.3 I/O 処 理

Umode プロセスでは，入出力機器に対して直接 I/O を発行できない．このため，1 回の I/O 処理は 1 回のシステムコール処理を伴うことになり，1 回のソフトウェア割り込みと復帰のオーバヘッドおよびシステムコール処理のオーバヘッドが発生する．しかし，Smode プロセス (保護無) や Smode プロセス (保護有) は，入出力機器に対して直接 I/O を発行できるため，オーバヘッドはない．

以上のことから，I/O 処理 1 回あたりのオーバヘッドを**表 2** に示す．表 2 において，sysOH はシステムコール処理のオーバヘッドである．

3.1.4 例 外 処 理

Umode プロセスや Smode プロセス (保護無) は，空間切り替えが発生しないため，オーバヘッドはない．しかし，Smode プロセス (保護有) は，1 回の例外処理に対し，2 回の空間切り替えのオーバヘッドが発生する．

表 2 I/O 処理 1 回あたりのオーバヘッド

走行中プロセス	I/O処理
Umodeプロセス	SWintr+sysOH
Smodeプロセス (保護無)	0
Smodeプロセス (保護有)	0

SWintr：ソフトウェア割り込みと復帰のオーバヘッド
sysOH：システムコール処理のオーバヘッド

表 3 例外処理 1 回あたりのオーバヘッド

走行中プロセス	例外処理
Umodeプロセス	0
Smodeプロセス (保護無)	0
Smodeプロセス (保護有)	VMchg×2

VMchg：空間切り替えのオーバヘッド

以上のことから，例外処理 1 回あたりのオーバヘッドを**表 3** に示す.

3.1.5 割り込み処理

割り込み処理の流れは，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．このため，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，空間切り替えの要否が決定される．

上記の流れにもとづき，空間切り替え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 プロセス (保護有) 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「割り込み処理の位置がユーザ空間 (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,「割り込み処理の位置がユーザ空間 (特殊) 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に移行する時」および,「走行中プロセスが Smode プロセス (保護有) 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,「割り込み処理の位置が異プロセスのユーザ空間 (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

表 4 割り込み処理 1 回あたりの空間切り替え回数

	走行中プロセス	実行プログラムの位置	同異	割り込み処理の位置	空間切り替え
(1)	Umode Smode(保護無)	OS空間	同プロセス	ユーザ空間(通常)	0回
(2)				OS空間	0回
(3)		ユーザ空間		ユーザ空間(通常)	0回
(4)				OS空間	0回
(5)		OS空間	異プロセス	ユーザ空間(通常)	2回
(6)				ユーザ空間(特殊)	2回
(7)				OS空間	0回
(8)		ユーザ空間		ユーザ空間(通常)	2回
(9)				ユーザ空間(特殊)	2回
(10)				OS空間	0回
(11)	Smode(保護有)	OS空間	同プロセス	ユーザ空間(特殊)	2回
(12)				OS空間	0回
(13)		ユーザ空間		ユーザ空間(特殊)	4回
(14)				OS空間	2回
(15)		OS空間	異プロセス	ユーザ空間(通常)	2回
(16)				ユーザ空間(特殊)	2回
(17)				OS空間	0回
(18)		ユーザ空間		ユーザ空間(通常)	4回
(19)				ユーザ空間(特殊)	4回
(20)				OS空間	2回

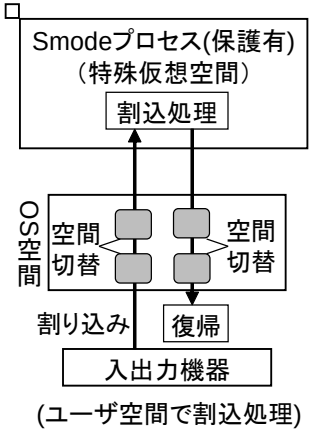

図 4 走行中プロセスが Smode プロセス (保護有) で走行中プロセスと割り込み処理を実行するプロセスが同プロセスの割り込み処理

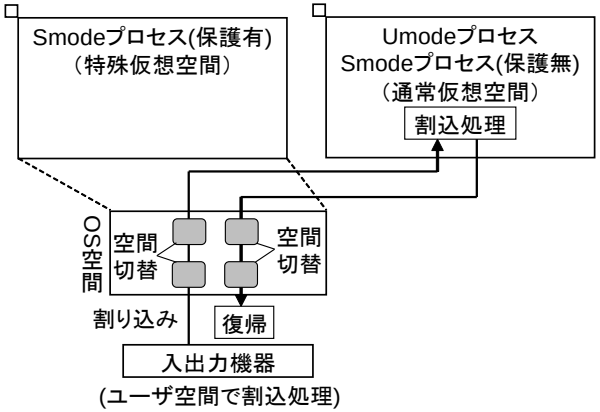

図 5 走行中プロセスが Smode プロセス (保護有) で走行中プロセスと割り込み処理を実行するプロセスが異プロセスの割り込み処理

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．

なお，Smode プロセス (保護無) は Umode プロセスと同様になる．ここで，割り込み処理 1 回あたりの空間切り替え回数を**表 4** に示し，以下に説明する．

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着目する．まず，同プロセスの場合を考える．走行中プロセスが Smode プロセス (保護有) で，実行プログラムの位置と割り込み処理の位置がユーザ空間の場合 (表 4(13)) を**図 4** に示す．走行中プロセスは Smode プロセス (保護有) であるため，特殊仮想空間で走行する．このため，OS 空間は読み込み専用の状態である．割り込み発生後，割り込み受付処理を行うために OS 空間へ移行する．ここで，OS 空間を読み書き可能にするために通常仮想空間へ切り替える．次に，割り込み処理の位置がユーザ空間であるため，ユーザ空間へ移行する．ここで，OS 空間を読み込み専用にして保護するために特殊仮想空間へ切り替える．割り込み処理の終了後，割り込み復帰処理を行うために OS 空間へ移行する．ここで，OS 空間を読み書き可能にするために通常仮想空間へ切り替える．そして，走行中プロセスである Smode プロセス (保護有) へ復帰する．ここで，OS 空間を読み込み専用にして保護するために特殊仮想空間へ切り替える．以上より，4 回の空間切り替えが発生する．

表 4(1)〜(4) では，走行中プロセスは Umode プロセスまたは Smode プロセス (保護無) である．この場合，実行プログラムの位置や割り込み処理の位置に関係なく，全て同プロセスの通常仮想空間で処理を行うため，空間切り替えは発生しない．

また，表 4(11)〜(14) では，走行中プロセスは Smode プロセス (保護有) である．(11) の実行プログラムの位置が OS 空間の場合，割り込み受付処理へ移行する際および走行中プロセスへ復帰する際の空間切り替えは発生しない．このため，空間切り替えは 2 回となる．(12) は全て OS 空間の処理であるため，空間切り替えは発生しない．(13) は図 4 の場合であるため，空間切り替えは 4 回となる．(14) の割り込み処理の位置が OS 空間の場合，割り込み処理へ移行する際および割り込み復帰処理へ移行する際の空間切り替えは発生しない．このため，空間切り替えは 2 回となる．

次に，異プロセスの場合を考える．走行中プロセスが Smode プロセス (保護有) で，実行プログラムの位置がユーザ空間，割り込み処理の位置が異プロセスのユーザ空間 (通常) の場合 (表 4(18)) を**図 5** に示す．走行中プロセスが Smode プロセス (保護有) であるため，特殊仮想空間で走行する．このため，OS 空間は読み込み専用の状態である．割り込み発生後，割り込み受付処理を行うために OS 空間へ移行する．ここで，OS 空間を読み書き可能にするために通常仮想空間へ切り替える．次に，割り込み処理の位置が異プロセスのユーザ

表 5 割り込み処理 1 回あたりのオーバヘッド

走行中プロセス	同異	
	同プロセス	異プロセス
Umodeプロセス	0	VMchg×Uintr×2
Smodeプロセス(保護無)		
Smodeプロセス(保護有)	VMchg×(Urun＋Uintr)×2	VMchg×(Urun＋Uintr)×2

VMchg : 空間切り替えオーバヘッド
Urun : 実行プログラムの位置がユーザ空間である確率
Uintr : 割り込み処理の位置がユーザ空間である確率

空間であるため，割り込み処理を行うプロセスのユーザ空間へ移行する．ここで，通常仮想空間から通常仮想空間へ空間切り替えが発生する．割り込み処理の終了後，割り込み復帰処理を行うために OS 空間へ移行する．ここで，通常仮想空間から通常仮想空間へ空間切り替えが発生する．そして，走行中プロセスである Smode プロセス (保護有) へ復帰する．ここで，OS 空間を読み込み専用にして保護するために特殊仮想空間へ切り替える．以上より，空間切り替えは 4 回である．

表 4(5)〜(10) では，走行中プロセスは Umode プロセスまたは Smode プロセス (保護無) である．(5)(6)(8)(9) の割り込み処理の位置がユーザ空間の場合，割り込み受付処理から割り込み処理を行うプロセスのユーザ空間および割り込み処理を行うプロセスのユーザ空間から割り込み復帰処理への空間切り替えが発生する．このため，空間切り替えは 2 回となる．(7)(10) の割り込み処理の位置が OS 空間の場合，全て OS 空間で処理を行うため，空間切り替えは 0 回となる．

また，表 4(15)〜(20) では，走行中プロセスは Smode プロセス (保護有) である．(15)(16) の実行プログラムの位置が OS 空間の場合，割り込み受付処理へ移行する際および走行中プロセスへ復帰する際の空間切り替えは発生しない．このため，空間切り替えは 2 回となる．(17) は全て OS 空間で処理を行うため，空間切り替えは 0 回となる．(18) は図 5 の場合であるため，空間切り替えは 4 回となる．(19) も同様に，空間切り替えは 4 回となる．(20) の割り込み処理の位置が OS 空間の場合，割り込み処理を行うプロセスへの切り替えは必要ない．このため，空間切り替えは 2 回となる．

以上の内容にもとづき，オーバヘッドと確率を

- VMchg: 空間切り替えのオーバヘッド
- Urun: 実行プログラムの位置がユーザ空間である確率
- Uintr: 割り込み処理の位置がユーザ空間である確率

とし，割り込み処理 1 回あたりのオーバヘッドの定式化を**表 5** に示す．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着目し，表 5 について説明する．まず，走行中プロセスは Umode プロセスまたは Smode プロセス (保護無) で，走行中プロセスと割り込み処理を実行するプロセスが同プロセスの場合を考える．表 4(1)〜(4) の空間切り替えは全て 0 回であるため，この時のオーバヘッドは 0 である．

次に，走行中プロセスは Umode プロセスまたは Smode プロセス (保護無) で，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プロセスの場合を考える．表 4(5)〜(7) の実行プログラムの位置が OS 空間の場合，割り込み処理の位置がユーザ空間のときのみ空間切り替えが 2 回となるため，この時のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 \quad (1)$$

表 4(8)〜(10) の実行プログラムの位置がユーザ空間の場合も同様に，割り込み処理の位置がユーザ空間のときのみ空間切り替えが 2 回となるため，この時のオーバヘッドは次式になる．

$$VMchg \times Urun \times Uintr \times 2 \quad (2)$$

上記の式 (1)(2) より，走行中プロセスが Umode プロセスまたは Smode プロセス (保護無) で，走行中プロセスと割り込み処理を実行するプロセスが同プロセスの場合のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 + VMchg \times Urun \times Uintr \times 2$$
$$= VMchg \times Uintr \times 2 \quad (3)$$

また，走行中プロセスは Smode(保護有) で，走行中プロセスと割り込み処理を実行するプロセスが同プロセスの場合を考える．表 4(11)(12) の実行プログラムの位置が OS 空間の場合，割り込み処理の位置がユーザ空間のときに空間切り替えは 2 回，割り込み処理の位置が OS 空間のときに空間切り替えは 0 回であるため，この時のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 \quad (4)$$

表 4(13)(14) の実行プログラムの位置がユーザ空間の場合，割り込み処理の位置がユーザ空間のときに空間切り替えは 4 回，割り込み処理の位置が OS 空間のときに空間切り替えは 2 回であるため，この時のオーバヘッドは次式になる．

$$VMchg \times Urun \times Uintr \times 4 + VMchg \times Urun \times (1 - Uintr) \times 2 \quad (5)$$

上記の式 (4)(5) より，走行中プロセスが Smode プロセス (保護有) で，走行中プロセス

と割り込み処理を実行するプロセスが同プロセスの場合のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 + VMchg \times Urun \times Uintr \times 4 + VMchg \times Urun \times (1 - Uintr) \times 2 = VMchg \times (Uintr + Urun) \times 2 \quad (6)$$

さらに，走行中プロセスは Smode プロセス (保護有) で，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プロセスの場合を考える．表 4(15)〜(17) の実行プログラムの位置が OS 空間の場合，割り込み処理の位置がユーザ空間のときに空間切り替えは 2 回，割り込み処理の位置が OS 空間のときに空間切り替えは 0 回であるため，この時のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 \quad (7)$$

表 4(18)〜(20) の実行プログラムの位置がユーザ空間の場合，割り込み処理の位置がユーザ空間のときに空間切り替えは 4 回，割り込み処理の位置が OS 空間のときに空間切り替えは 2 回であるため，この時のオーバヘッドは次式になる．

$$VMchg \times Urun \times Uintr \times 4 + VMchg \times Urun \times (1 - Uintr) \times 2 \quad (8)$$

上記の式 (7)(8) より，走行中プロセスが Smode プロセス (保護有) で，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プロセスの場合のオーバヘッドは次式になる．

$$VMchg \times (1 - Urun) \times Uintr \times 2 + VMchg \times Urun \times Uintr \times 4 + VMchg \times Urun \times (1 - Uintr) \times 2 = VMchg \times (Uintr + Urun) \times 2 \quad (9)$$

3.2 定 量 化

システムコール処理と I/O 処理について実測し，定式化の内容にもとづいて，ソフトウェア割り込みと復帰のオーバヘッド，空間切り替えのオーバヘッド，およびシステムコール処理のオーバヘッドを算出した．

測定は，走行モード変更機構とカーネル保護の機能を実現した ***AnT*** オペレーティングシステム[7] を利用し，Pentium4(2.4GHz) を搭載した計算機で行った．なお，測定には RDTSC 命令を使用した．測定結果は，10 回の処理時間の平均である．

処理が非常に短い getpid() システムコールの測定結果を**図 6** に示す．図 6 と 3.1.2 項で示したシステムコール処理の定式化より，ソフトウェア割り込みと復帰のオーバヘッドは Umode プロセスのクロック数と Smode プロセス (保護無) のクロック数の差であり，以下のようになる．

$$SWintr = 1276 - 481 = 795 \quad (10)$$

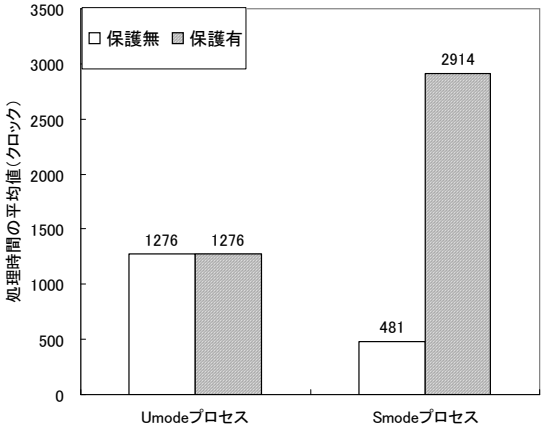

図 6 getpid() の測定結果

つまり，ソフトウェア割り込みと復帰のオーバヘッドは約 795 クロックとなる．

図 6 と 3.1.2 項で示したシステムコール処理の定式化より，空間切り替えのオーバヘッドは Smode プロセス (保護有) のクロック数と Smode プロセス (保護有) のクロック数の差であり，以下のようになる．ただし，1 回のシステムコールで 2 回の空間切り替えが発生するため，差の半分の値が空間切り替えのオーバヘッドとなる．

$$VMchg = \frac{2914 - 481}{2} = 1216.5 \quad (11)$$

つまり，空間切り替えのオーバヘッドは約 1217 クロックとなる．

I/O 処理 (ポート番号 0x2F8 に対する inb()，outb() 命令) の測定結果を**図 7** に示す．図 7，ソフトウェア割り込みと復帰のオーバヘッド，および 3.1.3 項で示した I/O 処理の定式化より，システムコール処理のオーバヘッドは，Umode プロセスのクロック数から Smode プロセスのクロック数とソフトウェア割り込みと復帰のオーバヘッドのクロック数の減算により以下のようになる．ここでは，図 7 の inb() 命令において OS 保護有の場合の Umode プロセスのクロック数と Smode プロセス (保護有) のクロック数を使用した．

$$sysOH = 4232 - 2998 - 795 = 439 \quad (12)$$

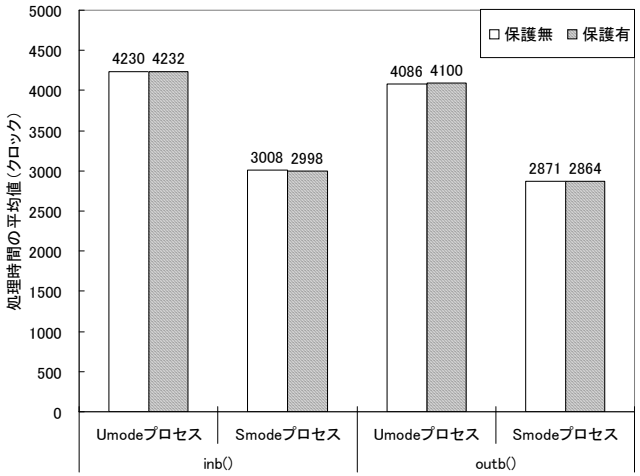

図 7 I/O 処理 (ポート: 0x2F8) の測定結果

つまり，システムコール処理のオーバヘッドは約 439 クロックとなる．

4. おわりに

マイクロカーネル構造の OS において，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について，走行モード変更機構とカーネル保護法のオーバヘッドが処理に与える影響を示した．

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について，オーバヘッドの要因 (ソフトウェア割り込みと復帰，空間切り替え，システムコール処理) を示し，各オーバヘッド量を定式化した．

また，走行モード変更機構とカーネル保護を実現した ***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．ソフトウェア割り込みと復帰のオーバヘッドは約 795 クロック，空間切り替えのオーバヘッドは約 1217 クロック，システムコール処理のオーバヘッドは約 439 クロックである．

残された課題として，走行モード変更機構の適用性の明確化がある．

謝辞 論文執筆の際にご指導して下さった岡山大学大学院自然科学研究科の後藤佑介助教に感謝します．

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) J. Liedtke，“Toward Real Microkernels，” Commnuications of The ACM，vol.39，Issue9，pp.70-77，1996.
4) 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.
5) 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.
6) 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.
7) 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.