システムソフトウェアと
オペレーティング・システム 75－5
（1997. 6. 5）

Tender におけるヘテロ仮想記憶の設計

長嶋 直希　谷口 秀夫　牛島 和夫

九州大学大学院システム情報科学研究科

E-mail: nagasima,tani,ushijima@csce.kyushu-u.ac.jp

本稿では、我々が開発しているオペレーティングシステム ***Tender*** におけるヘテロ仮想記憶の設計について報告する。ヘテロ仮想記憶とは、仮想メモリ空間上に一つのプロセスしか存在しない単一プロセス仮想メモリ空間と複数のプロセスが存在する多重プロセス仮想メモリ空間が融合した仮想記憶モデルである。ヘテロ仮想記憶の実現には、仮想メモリ空間をプロセスが移動する際のアドレス衝突の回避、仮想メモリ空間の外部断片化の回避という課題がある。これらの課題への対処を示す。次に、***Tender*** のメモリ管理の特徴を説明し、ヘテロ仮想記憶の実現法を述べる。

キーワード
オペレーティングシステム、メモリ管理、ヘテロ仮想記憶

Design of Heterogeneous Virtual Storage on *Tender*

Naoki NAGASHIMA, Hideo TANIGUCHI and Kazuo USHIJIMA

Graduate School of Information Science and Electrical Engineering,
Kyushu University

E-mail: nagasima,tani,ushijima@csce.kyushu-u.ac.jp

On this paper, we report a design of heterogeneous virtual storage on ***Tender***. Heterogeneous virtual storage is a virtual storage model containing both single and multiple process virtual memory spaces. There are two problems for implementing heterogeneous virtual storage. One is virtual address collision when a process migrates between virtual spaces. The other is fragmentation on virtual space. We describe solutions to these problems. Next, we explain the feature of virtual memory management on ***Tender***. Finally, we describe implementation of heterogeneous virtual storage on ***Tender***.

key words
operating system, memory management, heterogeneous virtual storage

1　はじめに

我々が開発しているオペレーティングシステム (OS) ***Tender***[1] では、プロセスと仮想メモリ空間を独立して管理している。これにより、一つの仮想メモリ空間上に複数のプロセスが存在する複数プロセス仮想メモリ空間 (MPVMS: Multiple Process Virtual Memory Space) と、一つの仮想メモリ空間上に一つだけプロセスが存在する単一プロセス仮想メモリ空間 (SPVMS: Single Process Virtual Memory Space) を共

存させることができる [2]。MPVMS と SPVMS を共存させた仮想記憶をヘテロ仮想記憶 (HVS: Heterogeneous Virtual Storage) と呼ぶ。本稿では、HVS の特徴を示し、実現時の問題と対処を述べる。さらに、***Tender*** への実装方式を説明する。

2 ヘテロ仮想記憶

2.1 従来の仮想記憶の特徴

仮想記憶のモデルとして、多重仮想記憶 (MVS: Multiple Virtual Storage) と単一仮想記憶 (SVS: Single Virtual Storage)[3] がある。MVS は、SPVMS が複数存在する仮想記憶モデルである。これに対し、SVS は、MPVMS が一つだけ存在する仮想記憶モデルである。図 1に MVS と SVS の様子を示す。

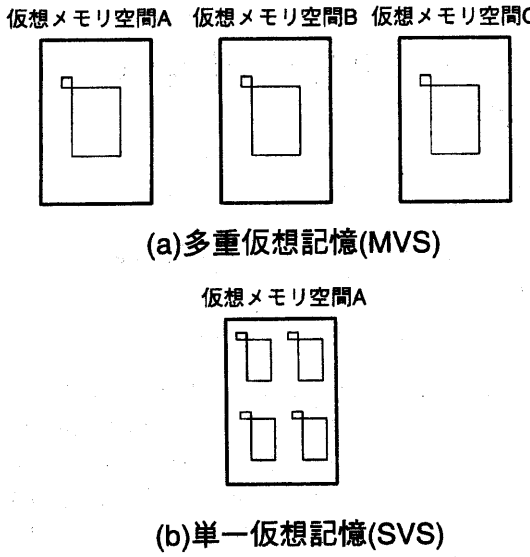

図 1: 多重仮想記憶と単一仮想記憶

以下に MVS と SVS の特徴を示す。

MVS では、各プロセスは固有の仮想メモリ空間を持ち、それぞれの仮想メモリ空間でアクセス保護がなされている。このため、各プロセスのメモリ操作に関する保護は強い。しかし、プロセス間通信やプロセスディスパッチには、カーネルを介した処理が必要となり、その分、オーバヘッドが大きい。また、プロセスディスパッチには、仮想メモリ空間の切替を伴うため、それまでの論理キャッシュや TLB (Translation Lookaside Buffer) の内容が無効になり、メモリアクセス速度が遅くなるという欠点もある。

SVS では、全てのプロセスは一つの仮想メモリ空間上で走行する。このため、プロセス間通信やプロセスディスパッチの処理が MVS に比べ、簡単である。さらに、プロセスディスパッチ時に仮想メモリ空間の切替が必要ないため、切替前の論理キャッシュや TLB を有効に用いることができる。しかし、プロセスのメモリ操作に関する保護が弱い。また、アドレス空間の大きさは有限であるため、プロセス一つあたりの大きさが、MVS の場合と比較して小さくなるという欠点もある。

2.2 HVS の特徴と利用法

HVS とは、SPVMS と MPVMS が融合した仮想記憶モデルである。SPVMS は MPVMS 上にプロセスが一つしか存在していない仮想メモリ空間と考えられるので、HVS は MPVMS が複数存在する仮想記憶モデルとみなすことができる。

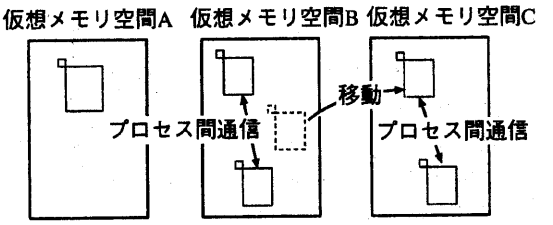

図 2: ヘテロ仮想記憶

HVS の様子を図 2に示す。HVS の特徴は、プロセスの処理内容に合わせて、適切な仮想メモリ空間を提供することにより、SVS と MVS の長所を生かすことができる点である。具体的には、プロセス間通信を頻繁に行なうプロセスやディスパッチを頻繁に行なうプロセスは同一の仮想メモリ空間上に置くことにより、SVS

の長所を得ることができる。図2 の仮想メモリ空間 B 上のプロセスがこれにあたる。逆に、強い保護を要求するプロセスや暴走を起こして他のプロセスを破壊する危険のあるプロセスは別の仮想メモリ空間上に配置することにより、プロセス間の保護を強くすることができる。図2 の仮想メモリ空間 A 上のプロセスがこれにあたる。

さらに、HVS の機能として、プロセスが現在動作している仮想メモリ空間上から、別の仮想メモリ空間に移動できれば、以下のような利点を得ることができる。

(1) 他の仮想メモリ空間上に存在するプロセスとプロセス間通信を頻繁に行なう場合、プロセス間通信を行なうプロセスを同じ仮想メモリ空間上へ移動して、プロセス間通信にかかるオーバーヘッドを削減することが可能となる。図2では、仮想メモリ空間 B 上のプロセスが仮想メモリ空間 C 上のプロセスと通信を行なうために仮想メモリ空間 C 上に移動する様子を示す。
(2) MPVMS では、プロセスの大きさが同じ仮想メモリ空間上に存在する他のプロセスによって制限されるため、プロセス実行途中で広いメモリ空間を必要とした場合、確保できないことがある。この場合、メモリ空間の確保が可能な他の仮想メモリ空間にプロセスを移動させることにより、この問題を解決できる。

2.3 実現時の課題と対処

HVS を実現するためには、以下の課題がある。

(1) プロセス移動時のアドレス衝突の回避
(2) 仮想メモリ空間の外部断片化の回避

それぞれの課題と対処について、以降に説明する。

2.3.1 プロセス移動時のアドレス衝突の回避

プロセスが別の仮想メモリ空間に移動した時に、移動先の仮想メモリ空間に既に存在しているプロセスとアドレスの衝突を起こす可能性がある。この問題への対処法を以下に示す。

(対処 A) プロセス生成時にアドレス衝突が生じないように、アドレスをずらす。
(対処 B) プロセス移動時に、アドレスを変更する。

(対処 A) は実現が容易であるが、プロセスの大きさが制限されるとともに、内部断片化が生じる欠点がある。(対処 B) はプロセスのアドレス配置が最適状態に保たれるという利点があるが、実行中のプロセスのアドレスを変更する必要があり実現は難しい。

2.3.2 仮想メモリ空間の外部断片化の回避

一つの仮想メモリ空間上で、プロセスの作成・削除やプロセスの仮想メモリ空間移動を繰り返すと、仮想メモリ空間の空き領域が断片化する。これにより、プロセスの大きさが空き領域の合計よりも小さいにも関わらず、プロセスの大きさよりも大きい連続領域が存在せず、プロセスを配置できないという問題が生じる。この問題の対処法として、上記に示した (対処 B) が考えられる。

3 *Tender* のメモリ管理

3.1 プロセスと仮想メモリの独立管理

Tender のメモリ管理の特徴は、プロセスと仮想メモリ空間を独立に管理していることである。

従来の OS では、仮想メモリ空間の管理情報はプロセス管理表に格納されていて、仮想メモリ空間の生成・消滅はプロセスの生成・消滅に同期している。そのため、仮想メモリ空間は、プロセス無しには存在し得ない。

それに対して、***Tender*** では、プロセス管理表から仮想メモリ空間の管理に必要な情報を分離して、その情報を仮想メモリ管理により管理している。仮想メモリ管理は、プロセス管理とは独立に機能しており、プロセス管理に対する操作に影響を受けない。このように、プロセスと仮想メモリ空間を独立に管理することにより、仮想メモリ空間は、プロセスの有無に関係なく存在することが出来る。

3.2 仮想空間と仮想領域

Tender のメモリ管理のもう一つの特徴として、仮想空間と仮想領域という概念がある。

仮想領域とは、メモリイメージを仮想化した領域で、その実体は実メモリ、または、外部記憶装置上に存在する。仮想空間とは、特定のアドレス範囲を持つ仮想的な空間である。具体的には、仮想アドレスを実アドレスに対応付けるための管理表のみからなる。仮想空間の任意の仮想アドレスに仮想領域を「貼り付ける」ことにより、仮想メモリ空間が作成される。ここで「貼り付ける」とは、仮想アドレスを実アドレスに対応付ける管理表に、対応付けのための情報を格納することである。仮想領域と仮想空間はそれぞれ単独ではプロセッサがアクセスすることはできない。仮想領域を仮想空間上の仮想アドレスに「貼り付ける」ことにより、仮想ユーザ空間、または、仮想カーネル空間が作成され、その上のデータへのアクセスが可能となる (図3)。

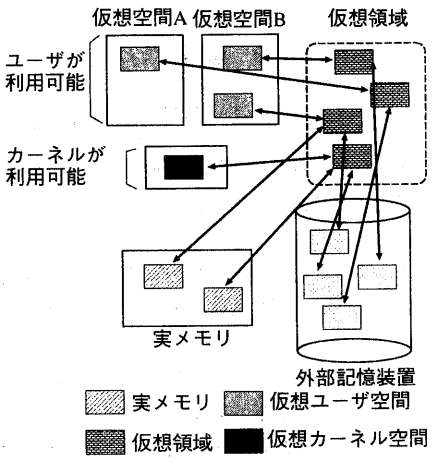

図 3: 仮想領域と仮想空間

仮想空間と仮想領域もまた、互いに独立に管理されている。従って、仮想空間の有無に関係なく、仮想領域は存在することが出来る。

3.3 メモリ管理の機能

Tender におけるメモリ管理の機能は、4つの資源「実メモリ」、「仮想領域」、「仮想カーネル空間」、「仮想ユーザ空間」を管理する各資源管理によって実現される。各資源管理の関係を図4に示し、以下に説明する。

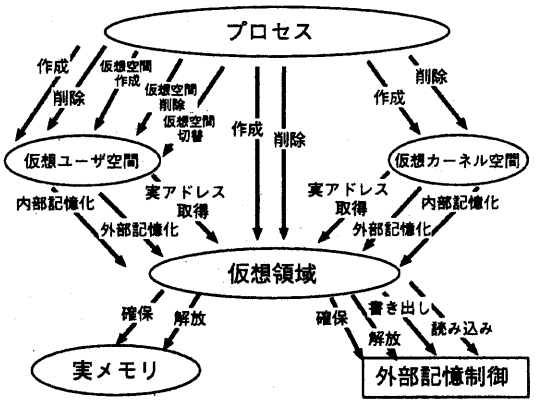

図 4: メモリ管理の機能を実現する資源の関係

3.3.1 実メモリ管理

資源「実メモリ」は、実メモリ上の領域である。

実メモリ管理は、資源「実メモリ」の管理を行なう。実メモリ管理の機能を以下に示し、説明する。

(1) 実メモリの確保、解放

資源「実メモリ」の確保、解放を行なう。

3.3.2 仮想領域管理

仮想領域管理は、資源「仮想領域」の管理を行なう。想領域管理が実現する機能を以下に示し、説明する。

(1) 仮想領域の作成、削除

資源「仮想領域」の作成、削除を行なう。

(2) 仮想領域の内部記憶化

資源「仮想領域」に資源「実メモリ」を割り当てる。資源「仮想領域」が既に外部記憶装置上の領域にデータを書き出していた場合、そのデータを実メモリ上に読み込むよう外部記憶制御に要求する。

(3) 仮想領域の外部記憶化

資源「仮想領域」が確保している資源「実メモリ」を解放する。実メモリの内容が更新されていた場合、実メモリの内容を外部記憶装置上の領域に退避するように外部記憶制御に要求する。

(4) 仮想領域の実アドレスの取得

資源「仮想領域」の実アドレスを返す。実メモリが割り当てられていない場合、その旨を知らせる。

3.3.3 仮想カーネル空間管理

資源「仮想カーネル空間」とは、資源「仮想領域」を仮想アドレスに対応付けることにより、カーネルモードでのみアクセス可能となった仮想アドレス範囲である。

仮想カーネル空間管理は、資源「仮想カーネル空間」を管理する。

仮想カーネル空間管理の機能を以下に示し、説明する。

(1) 仮想カーネル空間の作成

指定された資源「仮想領域」を指定された仮想アドレスに対応づけることにより、仮想カーネル空間を作成する。

(2) 仮想カーネル空間の削除

資源「仮想領域」と仮想アドレスの対応を断つことにより、仮想カーネル空間を削除する。

3.3.4 仮想ユーザ空間管理

資源「仮想ユーザ空間」とは、資源「仮想領域」を仮想アドレスに対応付けることにより、カーネルとユーザの両モードでアクセス可能となった仮想アドレス範囲である。

仮想ユーザ空間管理は、資源「仮想ユーザ空間」と仮想空間を管理する。

仮想ユーザ空間管理の機能を以下に示し、説明する。

(1) 仮想ユーザ空間の作成、削除

仮想ユーザ空間の作成、削除を行なう。

(2) 仮想空間の作成、削除

仮想空間の作成、削除を行なう。

(3) 仮想空間の切替

メモリ管理ユニット (MMU) に指定された仮想空間のマッピング表の先頭アドレスを与えることにより、指定された仮想空間にプロセッサがアクセスできるようにする機能である。

(4) 現在、プロセッサがアクセスできる仮想空間の確認

現在, プロセッサがアクセスできる仮想空間の通番を調べる機能である。これは、仮想空間管理表を参照することによりわかる。

4 HVS の実装方式

従来の OS では、仮想メモリ空間はプロセス管理により管理されているので、仮想メモリ空間とプロセスは独立に存在できない。そのため、HVS の実現は難しい。

一方、***Tender*** では、プロセスと仮想メモリ空間は独立しているので、HVS の実現は容易である。HVS を実現する方法を以下に説明する。

4.1 多重の MPVMS の実現

メモリ管理は、4 つの資源管理によって実現され、プロセス管理は、これらの資源管理によって管理される資源を用いてプロセスを生成する [4]。***Tender*** ではプロセスと仮想メモリ空間の管理は独立しているため、仮想空間は、プロセスの存在に関係なく、複数存在することができる。一つの仮想空間に複数の「仮想領域」を貼り付けて、複数の「仮想ユーザ空間」を生成できる。一つの「仮想ユーザ空間」をプロセスが利用するメモリ空間として対応づける。これにより、多重の MPVMS を実現できる。

プロセスの生成手順を図 5に示し、説明する。プロセス管理は、まず「仮想領域」を生成する。プロセスを新たな仮想空間上に生成する場合は、新たに仮想空間を作成し、その仮想空間に「仮想領域」を貼り付けて「仮想ユーザ空間」を生成する。プロセスを既存のプロセスと同じ仮想空間に生成する場合は、そのプロセスが存在する仮想空間に「仮想領域」を貼り付けて「仮想ユーザ空間」を生成する。「仮想ユーザ空間」

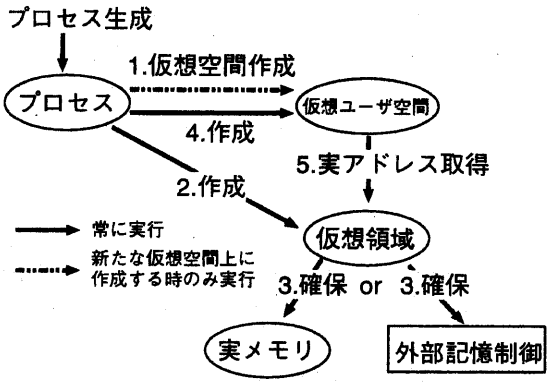

図 5: プロセス生成処理の流れ

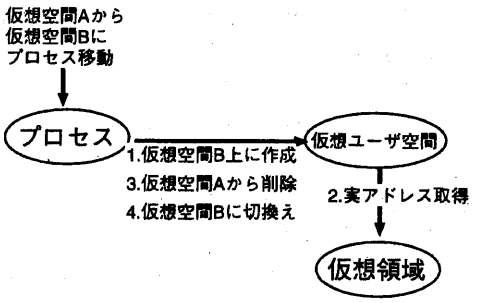

図 6: プロセス移動処理の流れ

上にプログラムやデータを読み込み、プロセスが生成される。

4.2 プロセスの仮想メモリ空間移動の実現

Tender では、「仮想領域」と仮想空間は独立に管理されている。そのため、異なる仮想空間上の「仮想ユーザ空間」によって、一つの「仮想領域」を共有することが可能である。これにより、プロセスの仮想メモリ空間移動を実現できる。

プロセスの仮想メモリ空間移動の手順を図 6に示し、説明する。プロセスの持つ「仮想ユーザ空間」に対応する「仮想領域」を移動先の仮想空間に貼り付けて、新たな「仮想ユーザ空間」を作成する。そして、それまでプロセスが走行していた仮想空間上の「仮想ユーザ空間」を削除する。プロセスの走行する仮想空間を、新たに作成した「仮想ユーザ空間」が存在する仮想空間に切り替えることにより、プロセスの仮想メモリ空間移動が実現できる。

5 おわりに

ヘテロ仮想記憶の特徴と利点、そして、実現時の課題と対処について述べた。ヘテロ仮想記憶の実現する時の課題として、プロセスの仮想メモリ空間移動時のアドレス衝突の回避、仮想メモリ空間の外部断片化の回避があるが、いずれも、プロセス生成時のアドレス調整によって対処できる。

Tender のメモリ管理を述べ、***Tender*** 上でのヘテロ仮想記憶の実現法について説明した。***Tender*** のメモリ管理の特徴として、プロセスと仮想メモリ空間の独立管理と、仮想領域と仮想空間の概念がある。この 2 つの特徴により、多重の複数プロセス仮想メモリ空間とプロセスの仮想メモリ空間移動が実現できる。

今後は、プロセスの仮想メモリ空間移動の機能を実現し、評価する予定である。

参考文献

[1] 谷口秀夫 : “分散指向永続オペレーティングシステム ***Tender***”, 情報処理学会シンポジウム論文集 Vol.95, No.7, pp.47-54 (1995).

[2] 村上大介, 青木義則, 谷口秀夫, 牛島和夫 : “Tender における資源「演算」の扱い”, 情報処理学会第 51 回全国大会予稿集, 5L-8 (1995).

[3] 岡本利夫 : “単一仮想記憶空間を特徴とするオペレーティングシステムについて”, 情報処理学会シンポジウム論文集 Vol.95, No 7, pp.39-46(1995).

[4] 谷口秀夫, 田中徳穂 : “***Tender*** のプロセス管理構造”, 情報処理信報, Vol.96, No.79, pp.109-114(1996).