

Tender オペレーティングシステムにおける プロセス間通信機能

3 F-5

田端 利宏 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

1 はじめに

我々は、プログラム構造を重視して機能を実現する *Tender*^[1](The ENduring operating system for Distributed EnviRonment) を開発している。*Tender* では、単一仮想記憶と多重仮想記憶を融合させたヘテロ仮想記憶^[2]を実装した。本稿では、ヘテロ仮想記憶の特徴を生かしたプロセス間通信機能の実現方式と性能評価について述べる。

2 プロセス間通信機能

2.1 プロセスとメモリ関連資源の関係

ヘテロ仮想記憶では、仮想空間上に0個以上のプロセスが存在でき、プロセスは、他の仮想空間への高速な移動が可能である。このため、効率的なプロセス間の協調処理が可能となっている。そこで、ヘテロ仮想記憶の特徴を生かしたプロセス間通信機能を実現する。

プロセスとメモリ関連の資源を図1に示す。仮想領域は、外部記憶装置あるいは実メモリのデータ格納域を仮想化した資源である。仮想空間とは、仮想アドレスの空間であり、仮想アドレスを実アドレスに変換する変換表に相当する。さらに、仮想領域を仮想空間に「貼り付ける」ことにより、プロセッサが仮想アドレスでアクセス可能な仮想カーネル空間や仮想ユーザ空間を作成できる。仮想カーネル空間と仮想ユーザ空間は、プロセッサが仮想アドレスでアクセスできる空間であり、各々、カーネルモードのみ、カーネルモードとユーザモードでアクセスできる。プロセスは、仮想空間上の仮想ユーザ空間を用いて作成される。

2.2 通信方式

プロセス間通信機能を実現するために、資源「コンテナ」と資源「コンテナボックス」を導入した。

資源「コンテナ」は、仮想空間に存在する仮想ユーザ空間を用いて生成される。資源「コンテナ」は、ユーザモードまたはカーネルモードでアクセス可能である。プロセスは、この資源「コンテナ」をやりとりすることで、プロセス間通信を実現することができる。また、資源「コンテナ」は、仮想領域の貼り付けと剥しにより、簡単に仮想空間間での移動、複写、共有が実現できる。コンテナの移動は、仮想領域を剥し、目的の仮想空間に

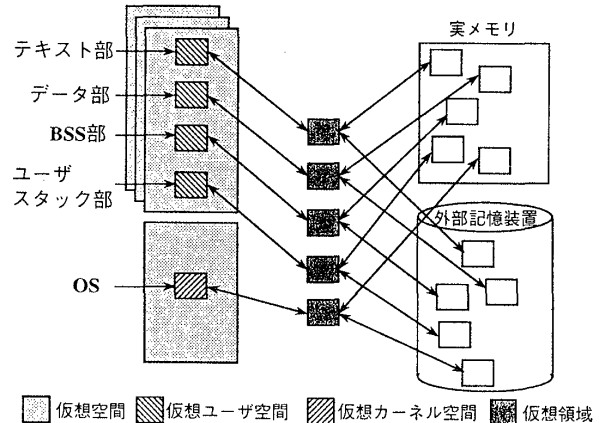


図1 プロセスとメモリ関連の資源

貼り付けることで実現でき、コンテナの共有は、仮想領域を別の仮想空間に貼り付けるだけで実現できる。また、コンテナの複写は、元のコンテナの内容を新しいコンテナに複写し、目的の仮想空間に貼り付けるだけで実現できる。

資源「コンテナボックス」は、プロセス間での資源「コンテナ」の受渡しの仲介をする。コンテナボックス管理処理部は、コンテナの送信要求を受けると、送信時に指定されたコンテナボックスのキューの最後に、送信要求のあったコンテナを格納する。コンテナの受信要求があった場合には、コンテナの送信時の貼り付けアドレス要求と受信側の貼り付けアドレス要求を比較し、一致する場合にコンテナを受信プロセスに渡す。

2.3 通信インタフェース

プロセス間通信インタフェースを表1に示す。コンテナの送信モードには、移動、複写、共有の3種類がある。移動は、送信元のコンテナを受信プロセスの空間に移動させる機能である。複写は、送信元のコンテナと同じ内容のコンテナを作り、そのコンテナを受信プロセスの空間に移動させる機能である。共有は、送信元のコンテナを、受信プロセスの空間で共有する機能である。受信プロセスは、コンテナボックスから送信されたコンテナを受けとることができる。この時、コンテナボックスにコンテナがなかった場合には、指定された時間だけ、コンテナが送信されるのを待つ。

また、プロセスは、コンテナボックスを介さずに直接指定したコンテナを受信ことができる。この場合にも、移動、複写、共有のモードを指定することができる。

コンテナボックスを介したプロセス間通信により、メッセージの送信、メモリ空間の共有が可能となり、プロセス間での協調作業が可能となる。また、コンテナボック

表 1 プロセス間通信インタフェース

通番	形式	機能
1	contbox_create(name)	資源名 name を持つコンテナボックスを生成し、コンテナボックス識別子 contbxid を返す。
2	contbox_delete(contbxid)	コンテナボックス contbxid を削除する。
3	contbox_check(contbxid, contid)	コンテナボックス contbxid の状態を確認する。
4	container_create(name, reqaddr, size, *paddr)	資源名 name を持つコンテナを、アドレス reqaddr に、大きさ size で生成し、コンテナ識別子 contid を返す。
5	container_delete(contid)	コンテナ contid を削除する。
6	container_send(contid, contbxid, reqaddr, op)	コンテナ contid をコンテナボックス contbxid に、アドレス reqaddr に貼り付けるように、モード op で送出する。
7	container_get(contid, contbxid, reqaddr, waittime, *paddr, *size)	コンテナボックス contbxid から、アドレス reqaddr にコンテナを受けとる。
8	container_sack(contid, reqaddr, op, *paddr, *size)	コンテナ contid をモード op で、受信プロセスの存在する仮想空間のアドレス reqaddr に貼り付ける。

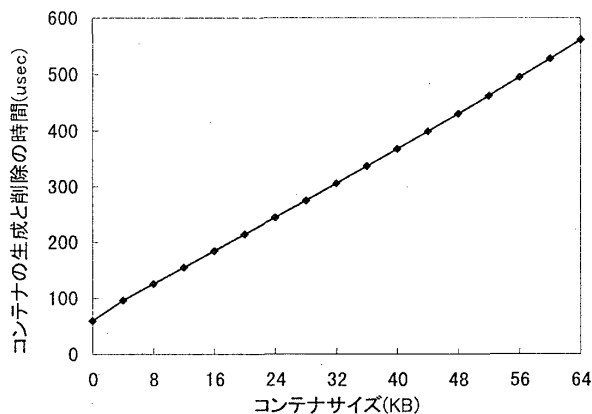


図 2 コンテナの生成と削除の時間

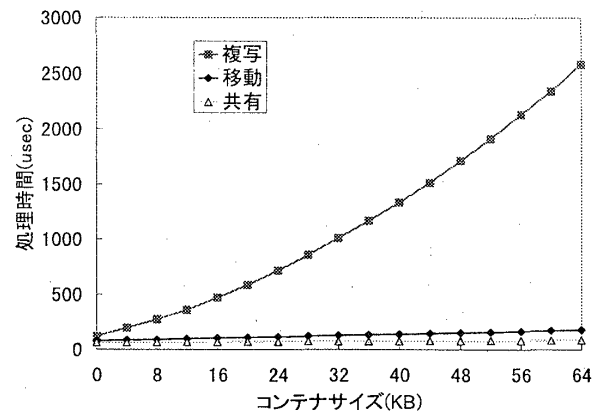


図 3 コンテナ送受信時間

スを介しないプロセス間通信では、コンテナを利用しているプロセスの送信を待つことなくコンテナを受けとることが可能となる。

3 性能評価

測定は、プロセッサ PentiumII 450MHz の計算機を使用した。

資源「コンテナボックス」を生成し削除する操作を 1000 回繰り返し、一回あたりの平均時間を算出した。コンテナボックスの生成と削除の一回当たりの平均時間は、約 5usec と高速である。

資源「コンテナ」の生成と削除を 1000 回繰り返し処理を、コンテナサイズを 0KB から、64KB まで 4KB ごとに大きさを変えて、測定し、一回当たりの平均時間を算出した。図 2 に測定結果を示す。図 2 より、コンテナの生成と削除時間は、コンテナサイズに比例して増加することがわかる。図 2 のグラフを一次近似した式を以下に示す。コンテナの生成と削除時間 (usec) を y 、コンテナサイズ (KB) を x とする。

$$y = 7.7376 * x + 60.516$$

コンテナ送受信時間の測定結果を図 3 に示す。図より、共有、移動、複製の順番で処理が速いことがわかる。複

写は、コンテナの内容を複製するため、共有、移動に比べかなり遅くなっている。共有と移動は、コンテナサイズに対して処理時間はほとんど一定である。共有は、移動に比べ高速である。これは、移動が送信元の空間から、メモリ空間を剥して、別の空間に貼り付けるのに対して、共有は、別の空間に貼り付けるだけで済むからである。

4 まとめ

Tender のプロセス間通信機能について述べた。評価した結果、コンテナボックスの生成は、5usec と高速であり、コンテナの生成は、60usec + コンテナサイズに比例した時間がかかる。コンテナ送受信速度は、移動と共有の処理時間はほとんどコンテナサイズに対して一定であり、複製はコンテナサイズに比例して処理時間がかかる。今後は、より詳細な評価を行なう予定である。

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**”, 情報処理学会コンピュータシステムシンポジウム, シンポジウム論文集 Vol.95, No.7, pp.47-54(1995).
- [2] 谷口秀夫, 長嶋直希, 田端利宏: “単一仮想記憶と多重仮想記憶を共存させたヘテロ仮想記憶の実現”, 情報研報, Vol.98, No.33, pp.87-94 (1998).