Vol.2013-OS-124 No.13
2013/3/1

Mintオペレーティングシステムにおける実メモリの分配と移譲

宮﨑 清人[1]　乃村 能成[1]　谷口 秀夫[1]

概要：計算機資源の効率的な利用のため，1 台の計算機上で複数 OS を走行させる方式が活発に研究されている．我々は，1 台の計算機上で複数 Linux の走行を実現する Mint オペレーティングシステムを研究開発している．仮想計算機方式とは異なり，Mint では，計算機資源の分割により，各 Linux は互いに処理負荷の影響を与えることなく独立に走行できる．Mint では各 Linux に分配する実メモリ量を静的に決定しているが，さらに Mint の Linux 間で実メモリを移譲できれば，各 Linux が必要とする実メモリ使用量の変動に対応できる．そこで，本稿では，Mint において各 Linux に実メモリを分配し，移譲するための課題について述べ，実現方式を明らかにする．

1. はじめに

計算機のプロセッサコア数や実メモリ量の増加により，計算機は高性能化している．これらの計算機資源を効率よく利用するためには，オペレーティングシステム (以降，OS と略す) と応用プログラム (以降，AP) を改修する必要がある [1]．そこで，既存のソフトウェア資源を生かしながら計算機資源を効率よく利用できる方式として，1 台の計算機上で複数 OS を走行させる方式が研究されている．この方式として，Xen[2] や VMware[3] といった仮想計算機方式がある．しかし，仮想計算機方式では，仮想化によるオーバヘッドのため，実計算機に比べて性能が低下するほか，OS は互いに処理負荷の影響を与えるという問題がある．

そこで，我々は，Mint オペレーティングシステム (以下，Mint と略す) を Linux ベースで研究開発している [4]．Mint では，計算機資源を分割し，各 OS に分配することで，1 台の計算機上で複数 OS を走行させる．これにより，Mint は，各 OS を実計算に近い性能で独立に走行できる特徴をもつ．

OS の負荷は，AP プロセスの実行状態によって変化する．このため，Mint において，各 OS の負荷に応じて資源を柔軟に分配する機能を実現することで，資源の利用効率をさらに高めることができる．本稿では，Mint において OS 間で実メモリを移譲可能にするための課題について述べ，実現方式を明らかにする．

[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. Mint の実メモリ分配

2.1 Linux における実メモリ利用の条件

Mint は x86_64 アーキテクチャ用の Linux(以降，64bit Linux と呼ぶ) をベースとしている．64bit Linux では，実メモリの利用に関して，以下 3 つの条件がある．

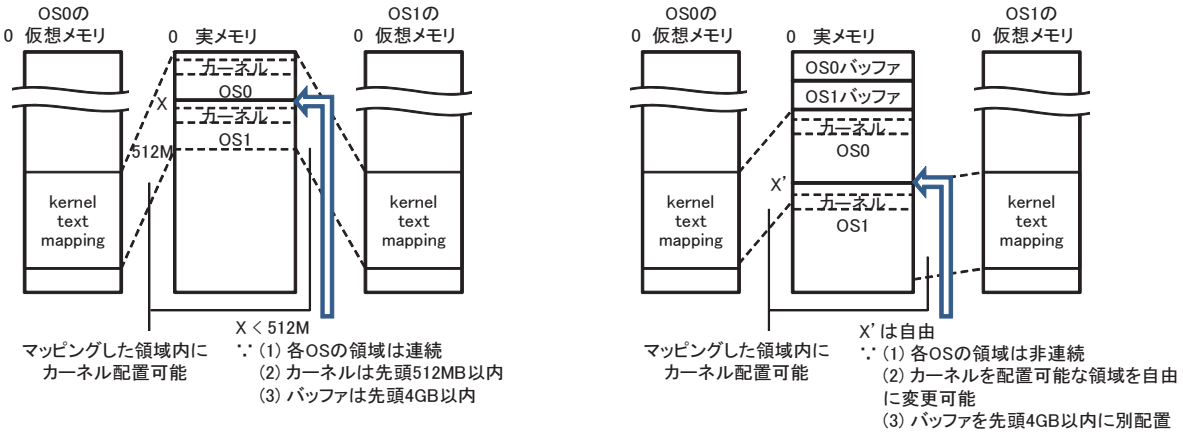
(条件 1) OS は連続な実メモリ領域を利用する：64bit Linux では，OS は実メモリ全体，つまり連続な領域を使用する．

(条件 2) 実メモリの先頭 512MB 以内にカーネルを配置しなければならない：これは，64bit Linux では，カーネルにアクセスするための仮想メモリ領域に，実メモリの先頭 512MB を固定的にマッピングするためである．この仮想メモリ領域を kernel text mapping と呼ぶ．

(条件 3) 実メモリの先頭 4GB 以内にバッファを配置しなければならない：これは，32bit PCI デバイスは，DMA により入出力する際，実メモリの先頭 4GB 以内しか利用できないためである．必要な DMA 用バッファの大きさは，64MB+56KB である．

2.2 課題と対処

前節の Linux における実メモリ利用の条件に基づき，Mint において実メモリ分配を実現する様子を**図 1**(A) に示し，そこから生じる課題について以下で説明する．前節の 3 条件により，各 OS への分配領域の境界 X は，メモリの先頭 512MB 以内に制限される．この結果，実メモリの終



(A) 64bit Linuxの実メモリ利用法に従った実メモリ分配　　(B) Mintの実メモリ分配

図 1　Mint のメモリ分配例

端側を分配する OS 以外は，512MB の領域を分割した少量の実メモリしか利用できない．そこで，各 OS に自由な割合で実メモリ分配を分配可能にする課題がある．

そこで，Mint では，Linux における実メモリ利用法を変更することで，各 OS に自由な割合で実メモリを分配可能にしている [5]．実メモリ分配の様子を図 1 の (B) に示し，以下で説明する．各 OS に 512MB 以上の実メモリを分配可能にするため，(条件 2) を撤廃した．具体的には，kernel text mapping にマッピングする実メモリ領域の先頭を自由に変更可能にし，カーネルを自由なアドレスに配置可能にした．(条件 2) の撤廃のみでは，(条件 3) により，各 OS に 4GB 以上の実メモリを割り当てできない問題が残る．そこで，バッファ領域を実メモリの先頭から別途配置することで，(条件 3) を撤廃した．これによって，各 OS の分配領域は非連続となるため，(条件 1) も撤廃した．以上により，各 OS に自由な割合で実メモリを分配できる．

前節の対処に基づいた実メモリ分配の実現方式を**図 2** に示し，以下で説明する．Mint では，実メモリを以下の 7 つの領域に分け，管理している．

(A) 0 から 0xfffff(1M-1) 番地までは，リアルモード領域であり，このうち使用できる領域は先頭の 640KB である．この領域は，OS が起動の初期段階に一時的に使用するため，各 OS に分配しなければならない．OS1 つにつき，必要な大きさは 128KB である．

(B) 0x100000(1M) から 0xffffff(16M-1) 番地までは，ZONE_DMA である．この領域は，ISA デバイス用のフレームバッファを配置するための領域である．ISA デバイスは，古いデバイスであり，現在では利用頻度は低い．このため，ZONE_DMA を OS に分配しなくとも，OS は走行できる．ここでは，ZONE_DMA は，使用せず予約しておく．

(C) 0x1000000(16M) から 0x1fffff(32M-1) 番地までは，共有メモリとして使用する．Mint において，コア占有

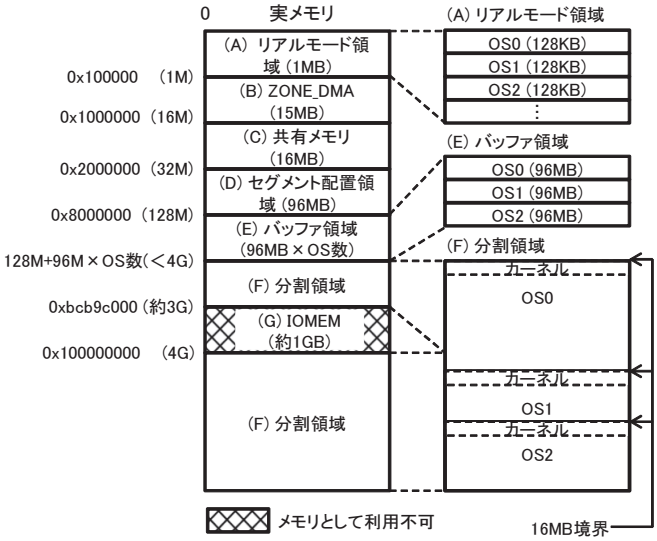


図 2　Mint における実メモリ分配の様子

状態管理や実メモリ占有状態管理のための OS 間通信に使用する．

(D) 0x2000000(32M) から 0x7fffff(128M-1) 番地までは，セグメント配置領域とする．これは，Mint における OS 起動の際，カーネル本体と初期 RAM ディスクといったデータを一時的に配置するための領域である．

(E) 0x8000000(128M) 番地から始まる領域は，バッファ領域とする．2.1 節の (条件 3) を満たすための領域として，OS1 つにつき 96MB ずつ分配する．ただし，(条件 3) のため，バッファ領域は実メモリの先頭 4GB 以内とする．各 OS は，この領域内にバッファを配置した残りの部分を自由に使用する．

(F) バッファ領域 (E) に後続する領域から IOMEM を除く領域は，分割領域とする．分割領域は，16MB 境界に従って，各 OS に分配する．各 OS は，この領域内にカーネルを配置し，領域を自由に使用する．

(G) 0xbcb9c000(約 3G) から 0xffffffff(4G-1) 番地までは，

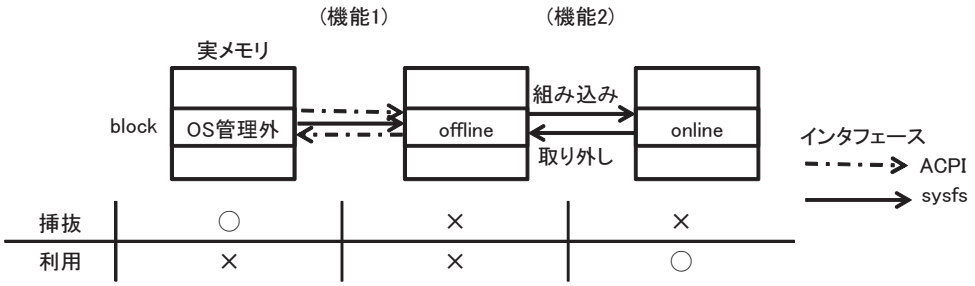


図 3 メモリホットプラグ機能と block 状態の関係

IOMEM である．この領域の先頭アドレスは，ハードウェアによって異なる．

上記の方式では，起動可能な OS 数の最大値は，バッファ領域を分配可能な数によって決定され，最大で 30 となる．リアルモード領域を分配可能な数 5 は，起動可能な OS の最大数とはならない．なぜなら，リアルモード部分 (A) は OS 起動の初期段階でのみ使用するため，OS の起動時期をずらすことで，複数 OS で共用できるためである．

分配は，OS がハードウェアから取得する実メモリマップを OS 毎に書き換え，OS の使用しない領域を予約領域として見せることで実現している．

3. メモリホットプラグ機能

3.1 Linux のメモリホットプラグ機能

3.1.1 機能

Linux のメモリホットプラグ機能 [6] には以下の 2 つの機能がある．

(機能 1) ページ管理データ構造の追加と削除

実メモリ領域の使用のため，例えばページテーブルや page 構造体といったページ管理データ構造が必要である．通常では，これらを OS 起動時に作成し，OS 起動後の追加や削除はできない．メモリホットプラグ機能では，これらを OS 起動後に追加や削除可能にする．

(機能 2) 実メモリ領域の使用可否の切り替え

ページ管理データ構造を作成済みの領域については，使用可否の切り替え操作ができる．実メモリ領域を使用する状態を online と呼び，使用しない状態を offline と呼ぶ．ただし，offline 状態に切り替えできる実メモリ領域は，未使用ページまたはユーザページからなる領域のみである．

これら 2 機能を適用する実メモリの最小単位は，block 単位である．Linux では，実メモリを先頭から固定長の block に分割して管理する．block の大きさはアーキテクチャによって異なり，例えば x86_64 では 1block は 128MB である．各 block には，実メモリの先頭側から順に 0 番から始まる番号が付けられる．この番号は，実メモリの内容によらず付けられ，例えば IOMEM のみからなる block にも付けられる．

上記の機能と block の状態との関係を**図 3** に示し，以下で説明する．図の左の block は，ページ管理データ構造が作成されておらず，OS 管理外の状態である．この状態では，実メモリの挿抜ができる．図の中央の block は，OS の管理下にあるが，利用はできない offline 状態である．さらに，図の右の block は，利用できる online 状態である．上記の (機能 1) によって OS 管理外の状態と offline 状態を切り替えでき，(機能 2) によって offline 状態と online 状態を切り替えできる．

上記の (機能 2) のため，メモリホットプラグ機能ではページ確保の分離機構とページマイグレーション機構を実現している．ページ確保の分離機構では，未使用ページまたはユーザページのみを配置する実メモリ領域を設定できる．これにより，設定した block を必ず offline 状態に変更できることを保証できる．

ページマイグレーション機構では，block の状態を online から offline にする処理において，使用ページを他の block に追い出す処理を実現している．

3.1.2 処理の流れ

(機能 2) について，block の状態を online にする処理 (以降，組み込みと呼ぶ) と offline にする処理 (以降，取り外しと呼ぶ) の流れをそれぞれ**図 4** と**図 5** に示し，以下で説明する．

実メモリのページを管理するデータ構造として page 構造体がページ毎にある．page 構造体の reserved フラグをセットすることで，当該ページの確保を停止できる．これを利用し，組み込みと取り外しを実現している．

組み込みでは，当該 block の各ページについて，page 構造体の reserved フラグをクリアする．reserved フラグのクリアに失敗したページがあれば，組み込みを異常終了する．block 中の前ページで reserved フラグのクリアに成功すると，各ページを free リストに登録し，組み込みを正常終了する．

取り外しでは，当該 block からのメモリ確保を停止する．当該 block に使用ページがある場合，ページマイグレーション機構により，ページを他の block に追い出す．ページの追い出し後，ページの reserved フラグをセットする．ページの追い出しまたはページの reserved フラグセット

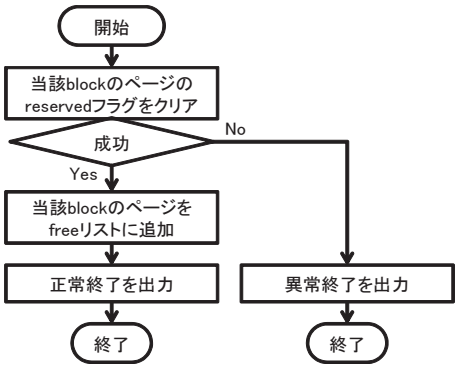


図 4　block 組み込みの処理流れ

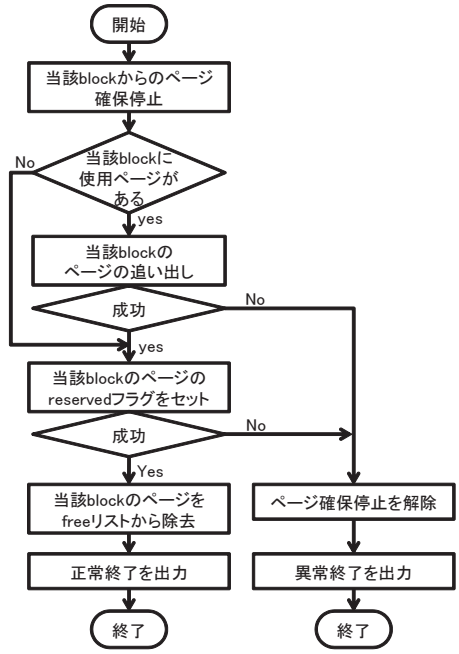


図 5　block 取り外しの処理流れ

に失敗すると，ページ確保停止を解除し，異常終了を出力し，取り外しを終了する．上記の両方に成功すると，当該 block のページを free リストから除去し，正常終了を出力して取り外しを終了する．

3.1.3　インタフェース

メモリホットプラグ機能は，sysfs への操作として実行する．この際，対象の block を選択する．

3.2　Mint での Linux メモリホットプラグ機能利用時の問題点

3.2.1　利用

分割領域 (F) をメモリホットプラグ機能によって分配する手法について，Mint で Linux のメモリホットプラグ機能を利用する様子を**図 6** に示し，以下で説明する．リアルモード領域 (A) からバッファ領域 (E) までは，現状の Mint と同様の方法で分配し，利用する．分割領域 (F) については，現状の Mint と同様の方法で分配する．ただし，以下の 3 条件全てを満たす部分のみを実メモリ移譲の対象とする．

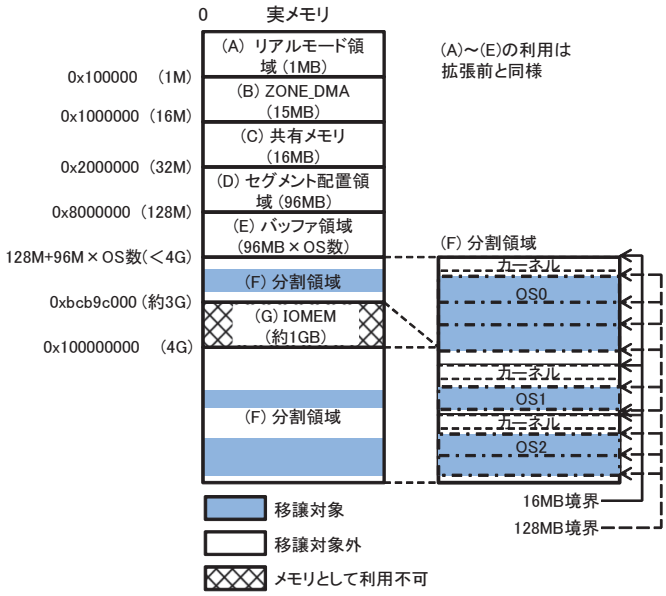


図 6　Mint でメモリホットプラグ機能を利用する様子

(1) block の境界に沿っている
　block の組み込みと取り外しの単位は block 単位であるため，block の境界に沿った領域だけを移譲できる．
(2) 単独の OS が使用している
　複数 OS で分割利用している block は，組み込みできない．
(3) 移動不可能なページを含まない
　空きページまたはユーザページのみを含む block は取り外しでき，移譲対象となる．一方，例えば，カーネルやページテーブルを含む block は取り外しできず，移譲対象とならない．

図 6 では，リアルモード領域 (A) からバッファ領域 (E) は移譲対象外である．また，分割領域 (F) については，上記の 3 条件を満たす block 部分のみが移譲対象となり，これ以外は移譲対象外となる．具体的には，各 OS のもつ分割領域のうち，先頭付近にはカーネルを配置し，終端付近には例えばページテーブルを配置する．このため，先頭付近と終端付近を除く block 部分が移譲対象となる．

3.2.2　問題点

図 6 の方法で実メモリを利用すると，以下の問題点がある．

(問題点 1)　OS 単独リブート時にデータを破壊：OS 単独リブートにおける問題について**図 7** に示し，以下で説明する．Mint におけるリブート処理として，計算機全体のリブートと OS 単独リブートがある．単独リブートする OS は，まず，占有 block を全て取り外して OS 終了し，その後，取り外した block を再び組み込んで OS を起動開始する．OS 終了から OS 起動開始までの期間において，取り外した block を走行中の他 OS が組み込むことが起こりうる．この結果，リブートする

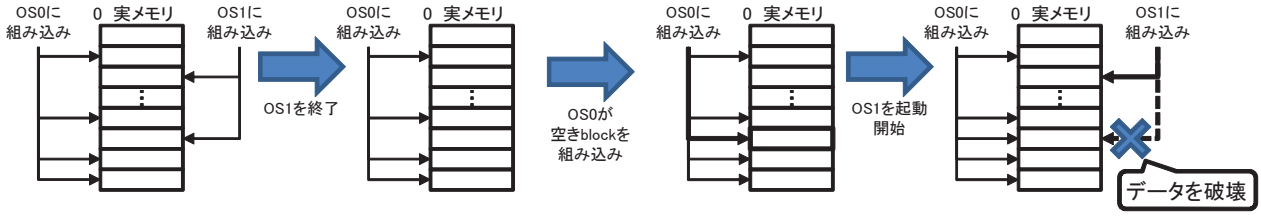


図 7　OS 単独リブートにおける問題

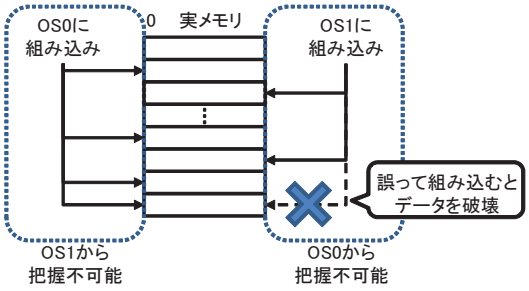


図 8　実メモリ領域の組み込みにおける問題

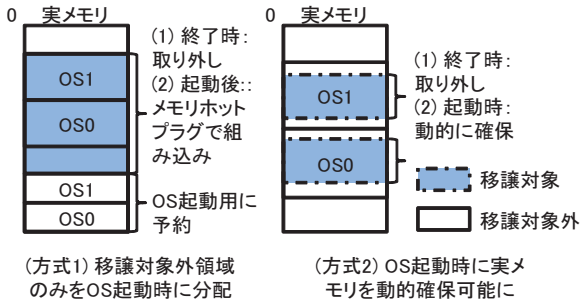


図 9　OS 単独リブート対処の方式

OS が起動開始すると，同一 block を複数 OS に組み込むことになり，互いのデータを破壊する．

(問題点 2)　他 OS の実メモリ組み込み状態を把握不可能：実メモリ領域の組み込みにおける問題について**図 8** に示し，以下で説明する．Mint で Linux のメモリホットプラグ機能を利用すると，各 OS は自由に block の組み込みと取り外しを行える．この際，Mint の各 OS は互いの実メモリ組み込み状態を把握できない．このため，1 つの実メモリ領域を誤って複数 OS に組み込み，互いのデータを破壊する可能性がある．

4. 実メモリの分配と移譲

4.1 OS 単独リブートへの対処

4.1.1 対処

(問題点 1) に対する 2 つの対処方式を**図 9** に示し，以下で説明する．

(方式 1)　移譲対象外領域のみを OS 起動時に分配する：実メモリ分配方式を変更し，分割領域 (F) を移譲対象領域と移譲対象外領域に分ける．移譲対象外領域は，OS 起動に十分な大きさとし，OS 起動用に予約しておく．OS は，リブートの際，終了段階で移譲対象領域を取り外す．また，起動段階では，移譲対象外領域のみを用いて OS を起動する．リブート完了後，実メモリ移譲によって移譲対象領域の block を組み込み，必要な実メモリ量を分配する．

(方式 2)　OS 起動時に実メモリを動的確保可能にする：OS 起動時に移譲対象領域の未使用 block を探索し，動的確保可能にする．実メモリ分配方式は，変更しない．OS は，リブートの際，終了段階で移譲対象領域を取り外す．また，起動段階では，必要な block を移譲対象領域から確保する．これにより，移譲対象外領域と確保した移譲対象領域の両方を使用して OS を起動する．

表 1　OS 単独リブート対処方式の比較

方式	分配方法の変更	OS 起動に利用する領域	実現工数
(方式 1)	あり	固定	少
(方式 2)	なし	動的	多

2 方式の比較を**表 1** に示し，以下で説明する．(方式 1) では，OS 起動段階で分配する実メモリ領域を静的に決定しているため，OS 起動段階での block 確保は必要ない．一方，(方式 2) では，OS 起動段階で block を動的に確保するため，block の排他機構を操作する必要がある．block の確保は，OS 起動後には容易に行えるものの，OS 起動段階で行うことは難しい．また，(方式 2) では block の確保失敗時への対処が必要となる．以上から，(方式 1) は (方式 2) よりも少ない工数で実現できる．

(方式 2) では，OS 起動段階での実メモリ量を動的に調整できる利点はあるものの，利用場面は少ない．

以上のことから，(方式 1) は少ない工数で対処を実現でき，かつ，利点において (方式 2) と遜色がない．このため，(方式 1) に基いて対処する．

4.1.2 実現方式

上記の対処に基づいた実メモリ分配の様子を**図 10** に示し，以下で説明する．リアルモード領域 (A) からバッファ領域 (E) の分割方法については，3.2.1 項と同様である．これらの領域は，移譲対象外である．

分割領域 (F) は，あらたに移譲対象領域 (F-a) と移譲対象外領域 (F-b) の 2 つに分ける．これらは，どちらも block の境界に従っている．

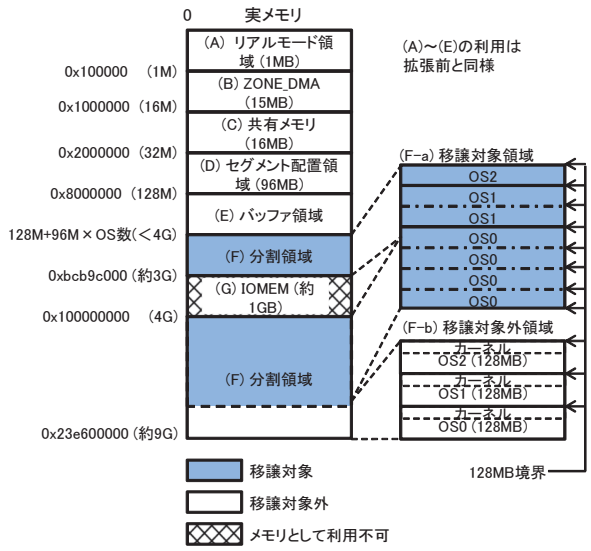


図 10 OS 単独リブートに対処した実メモリ分配

移譲対象領域 (F-a) は，リブートの際には，リブート完了後に実メモリ移譲機能で組み込む．また，リブート完了後には，自由に移譲できる．一方，移譲対象外領域 (F-b) は，リブートの際，OS の起動段階から OS に分配しておく．また，OS の起動から終了まで，同じ OS に分配する．

移譲対象領域 (F-a) を移譲できることを保証するため，ページ種類によるページ確保の分離機構を利用する．具体的には，移譲対象領域 (F-a) には移動可能なユーザページのみを配置するよう設定する．

移譲対象外領域 (F-b) については，あらかじめ分配を静的に決めておく必要がある．具体的には，Mint で起動する OS 数だけの block を実メモリ終端側から予約し，各 OS は 1block を移譲対象外領域 (F-b) とする．したがって，各 OS は，96MB のバッファ領域 (E) と 128MB の移譲対象外領域 (F-b) の合計 224MB の領域を静的に占有して起動する．この量は，64bit の Vanilla Linux の起動に十分である．

4.2 実メモリ領域の排他

各 OS への実メモリ分配状態を一望可能にし，かつ同一の実メモリ領域を複数 OS に組み込むことを防ぐため，共有管理表を用いて実メモリ領域を排他する．共有管理表には，各 block に対応するエントリを設け，各 block の分配状態を格納する．

5. まとめ

Mint において，OS 間で実メモリの移譲を実現する方式について述べ，課題と対処を明らかにした．

実メモリ移譲には，Linux のメモリホットプラグ機能を利用する．課題として，OS 単独リブートへの対処と実メモリ領域の排他がある．OS 単独リブートへの対処のためには，実メモリ分配法を変更し，OS 起動時に OS 相互のデータ破壊を防ぐ．また，実メモリ領域の排他のためには，共有管理表で実メモリ占有状態を管理する．

残された課題として，評価がある．

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] Boyd-Wickizer, S., Clements, A. T, Mao, Y., Pesterev, A., Kaashoek, M. F, Morris R. and Zeldovich N.: An analysis of Linux scalability to many cores, Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI'10), pp. 1-8, (2010).

[2] Paul, B., Boris, D., Keir, F., Steven, H., Tim, H., Alex, H., Rolf, N., Ian, P. and Andrew, W.: Xen and the Art of Virtualization, Proceedings of the 19th ACM Symposium on Operating Systems Principles, pp.164-177, (2003).

[3] Jeremy, S., Ganesh, V. and Beng-Hong, L.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, Proceedings of the General Track: 2002 USENIX Annual Technical Conference, pp.1-14, (2001).

[4] 千崎良太, 中原大貴, 牛尾裕, 片岡哲也, 栗田祐一, 乃村能成, 谷口秀夫: マルチコアにおいて複数の Linux カーネルを走行させる Mint オペレーティングシステムの設計と評価, 電子情報通信学会技術研究報告, vol.110, no.278, pp.29-34, (2010).

[5] 宮﨑清人，乃村能成，谷口秀夫: Mint オペレーティングシステムにおける実メモリ分配法, 情報処理学会研究報告, 2012-OS-122, No. 16, pp. 1-6, (2012).

[6] Joel, S., Dave, H., Mike, K., Hirokazu, T., Iwamoto, T., Yasunori, G., Kamezawa, H., Matt, T. and Bob, P.: Hotplug Memory Redux, Proceedings of the 2005 Linux Symposium, Vol. 2, pp. 151-174, (2005).