複数Linux走行方式における実メモリ移譲機能

宮﨑　清人
岡山大学　大学院自然科学研究科
2014年2月13日

目次

はじめに

(1) 1台の計算機上で複数OSを走行させる方式が広く利用

(2) 多数のサービス実行により実メモリ容量の需要が増大

⇒ 複数OS走行方式において実メモリ使用量を抑制できれば有用

＜仮想計算機方式(VM方式)＞

利点：実メモリを使用量に応じてOS間で動的再分配可能

⇒ 計算機システム全体の実メモリ使用量を抑制

欠点：仮想化により実メモリアクセスの性能が低下

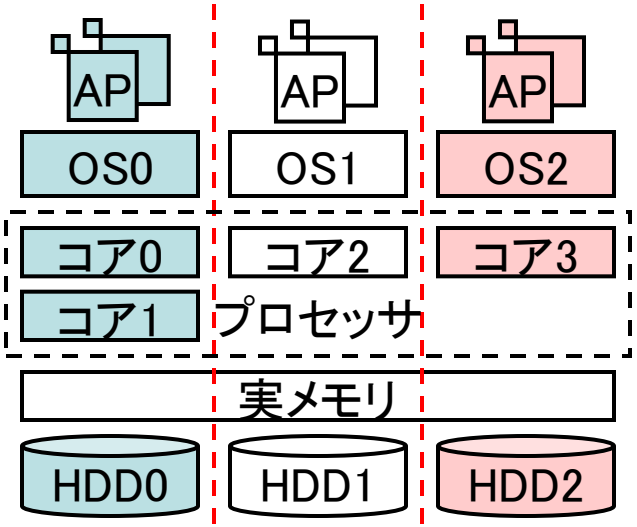
⇒ APの実メモリ使用時間が長大化し，実メモリ使用量の抑制効果が低下

↓

高い実メモリアクセス性能と実メモリの動的再分配を実現する実メモリ移譲機能を提案し，Mintにおいて実現

高いOS間の独立性を有する複数Linux走行方式

複数Linux走行方式：Mint



＜各OSの使用する資源の決定方法＞

(1) プロセッサ　コア単位で分割し，各OSは1つ以上のコアを占有

(2) 実メモリ　空間分割し，各OSに分割領域を分配

(3) 入出力機器　デバイス単位で分割し，各OSが仮想化によらず直接占有制御

＜各OSの実現方法＞

Linuxをベースとし，占有資源を制限するための改修を加えて実現

∴各OSの機能はフルセットのLinuxと同等

目次

目的と基本機能

＜目的＞

複数Linux走行方式において実メモリを効率よく利用

(1) 高速な実メモリアクセス

(2) 実メモリの動的再分配

＜基本機能＞

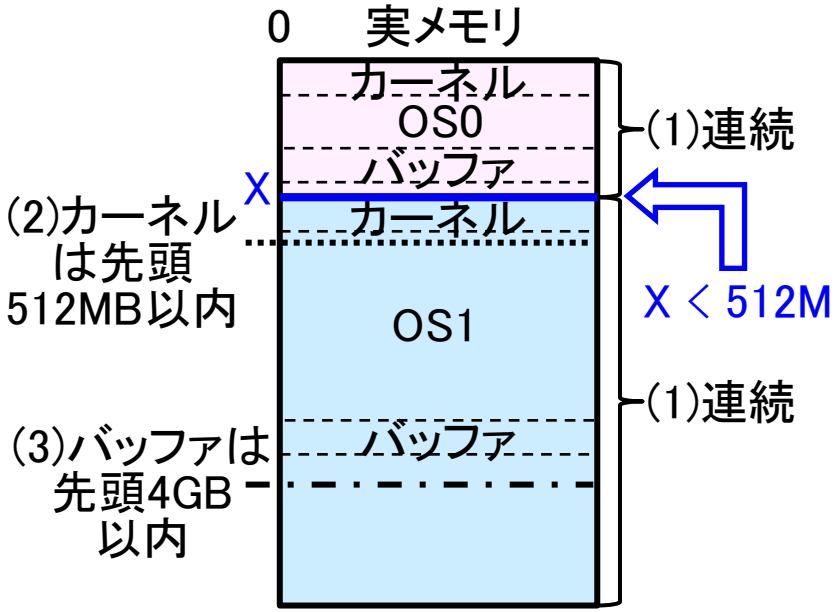
(1) 実メモリ分配

実メモリを仮想化によらず複数OSに分配し，高速な実メモリアクセスを実現

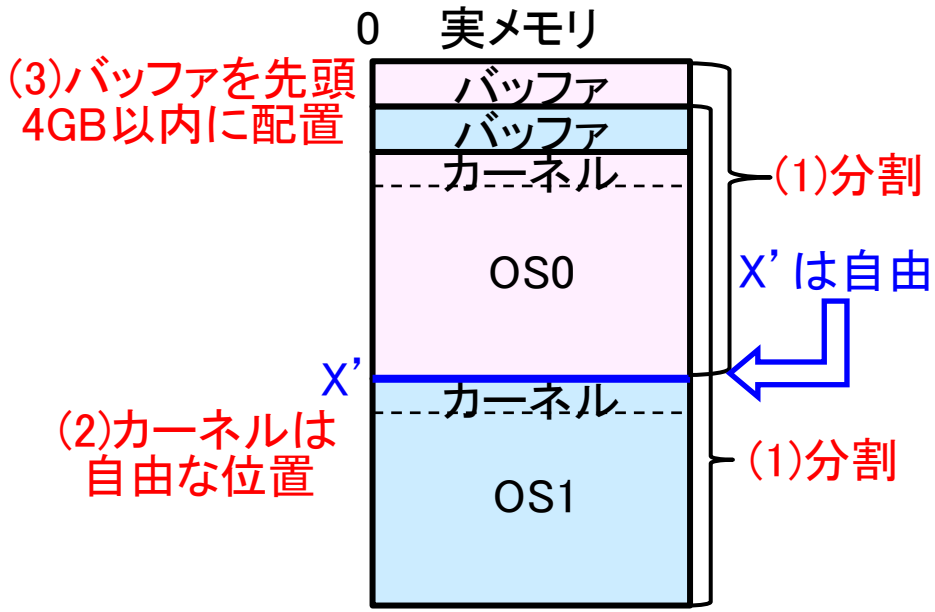
(2) 実メモリ移譲

Linuxのメモリホットプラグ機能を利用し，使用量の変動に応じて実メモリの動的再分配を実現

実メモリ分配における課題と対処



64bit Linuxの実メモリ利用法に従った実メモリ分配



Mintの実メモリ分配

(課題) 各OSに自由な割合で実メモリ領域を分配可能にする

(対処) カーネル配置領域の先頭アドレスを各OSで変更可能にする

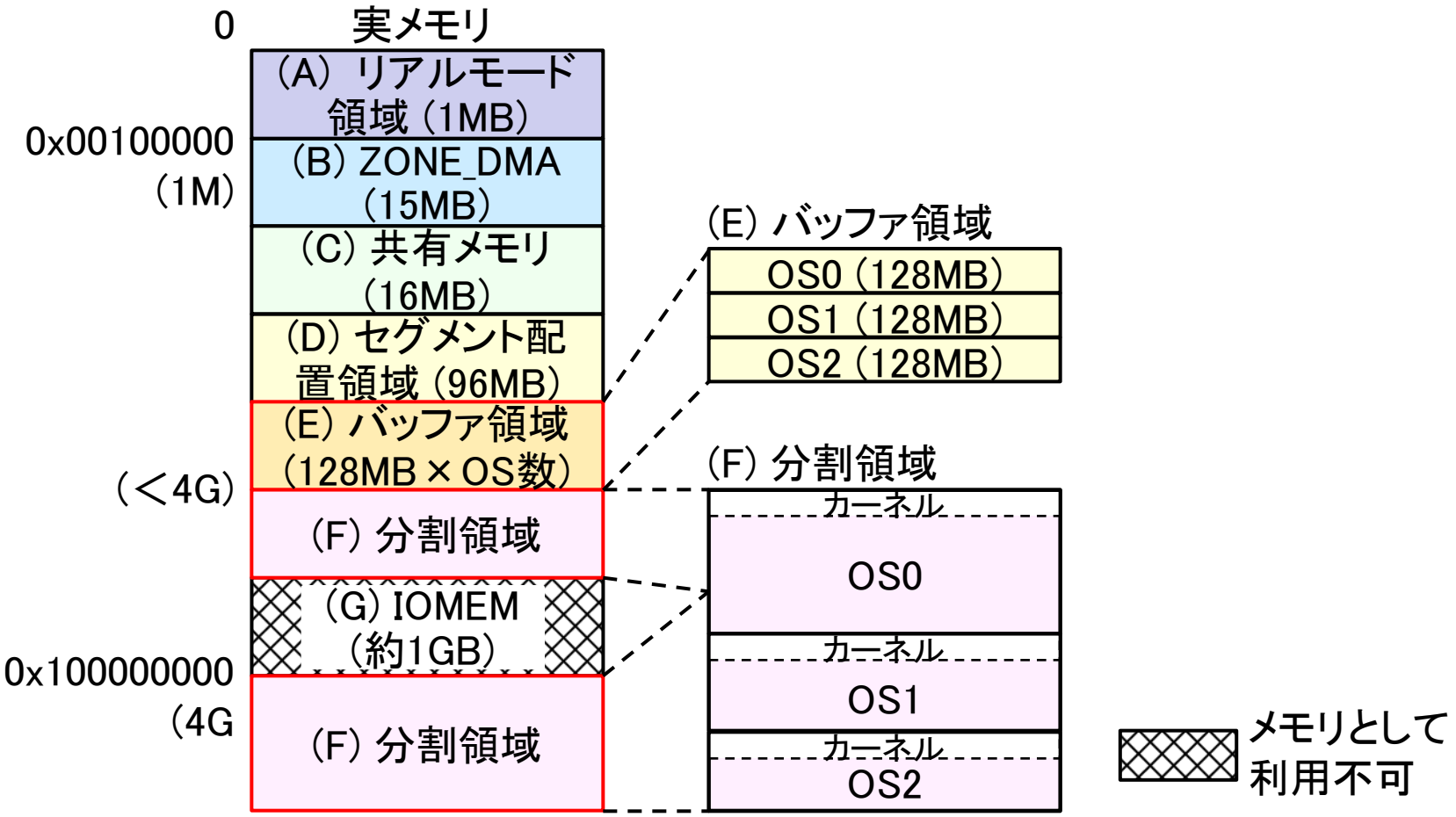
(1) 各OSの領域を分割

(2) カーネルを配置可能な領域を自由に変更可能

(3) バッファを先頭4GB以内に分離して配置

⇒ 改修箇所を局所化し，かつ動作において問題なし

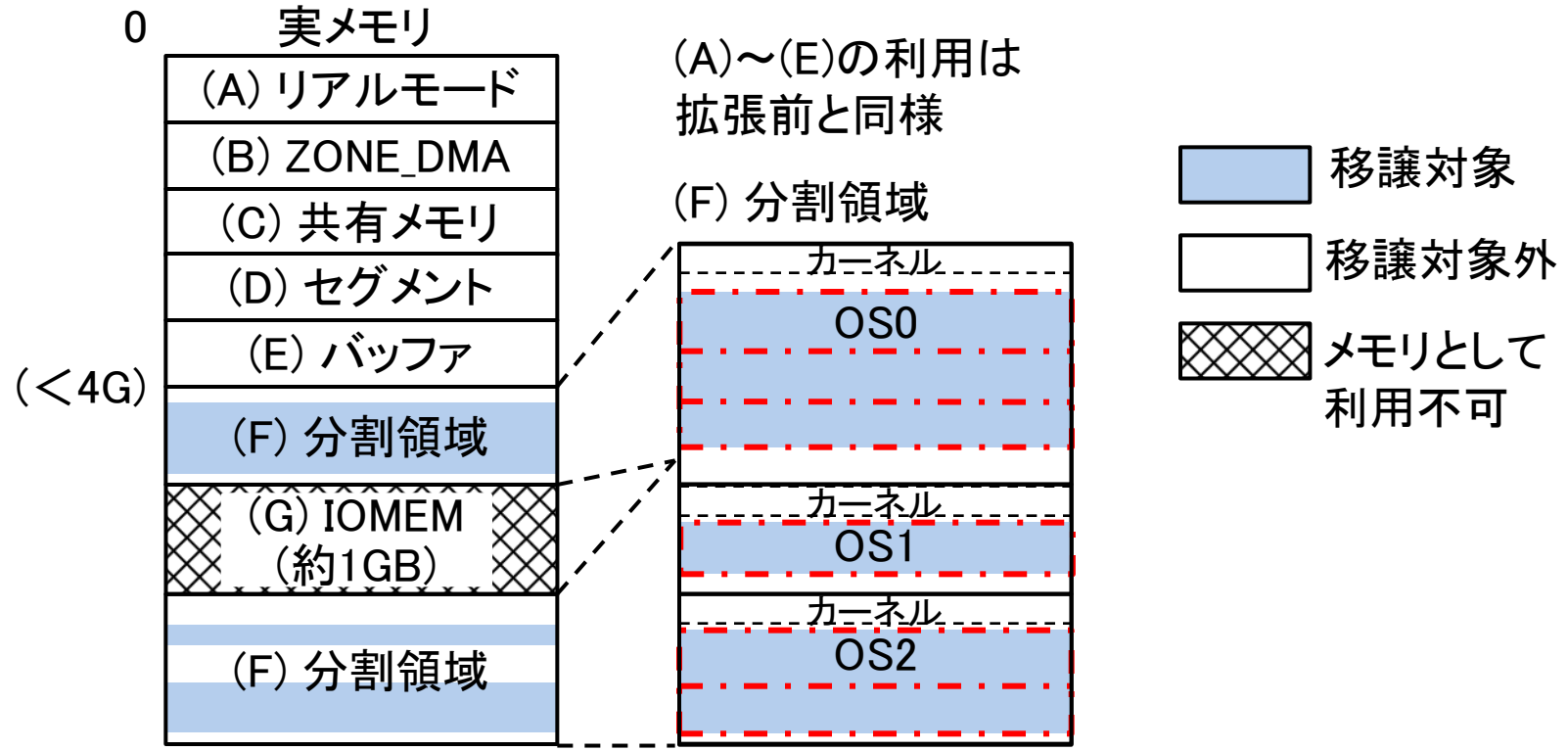
実メモリ分配の実現方式



(E) バッファ領域　　DMA用のバッファを配置

(F) 分割領域　　各OSのカーネルを配置し，自由に使用

実メモリ移譲の基本方式と課題



Linuxのメモリホットプラグ機能を利用して実現

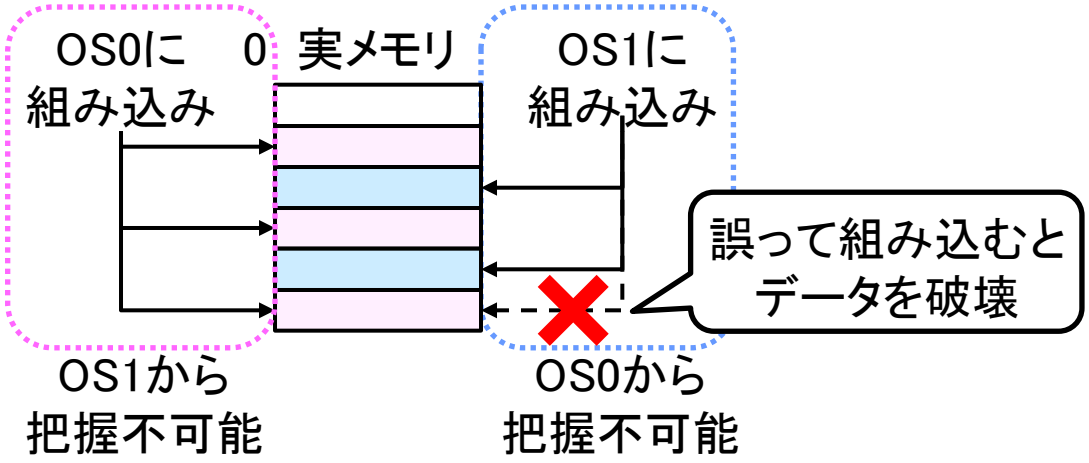
block(128MB)を単位として実メモリ領域の使用可否を切り替え可能

➡ 分割領域(F)中のうち，blockに沿った領域を自由に移譲可能

(課題1) 実メモリ領域の排他

(課題2) OS単独リブートへの対応

実メモリ領域の排他



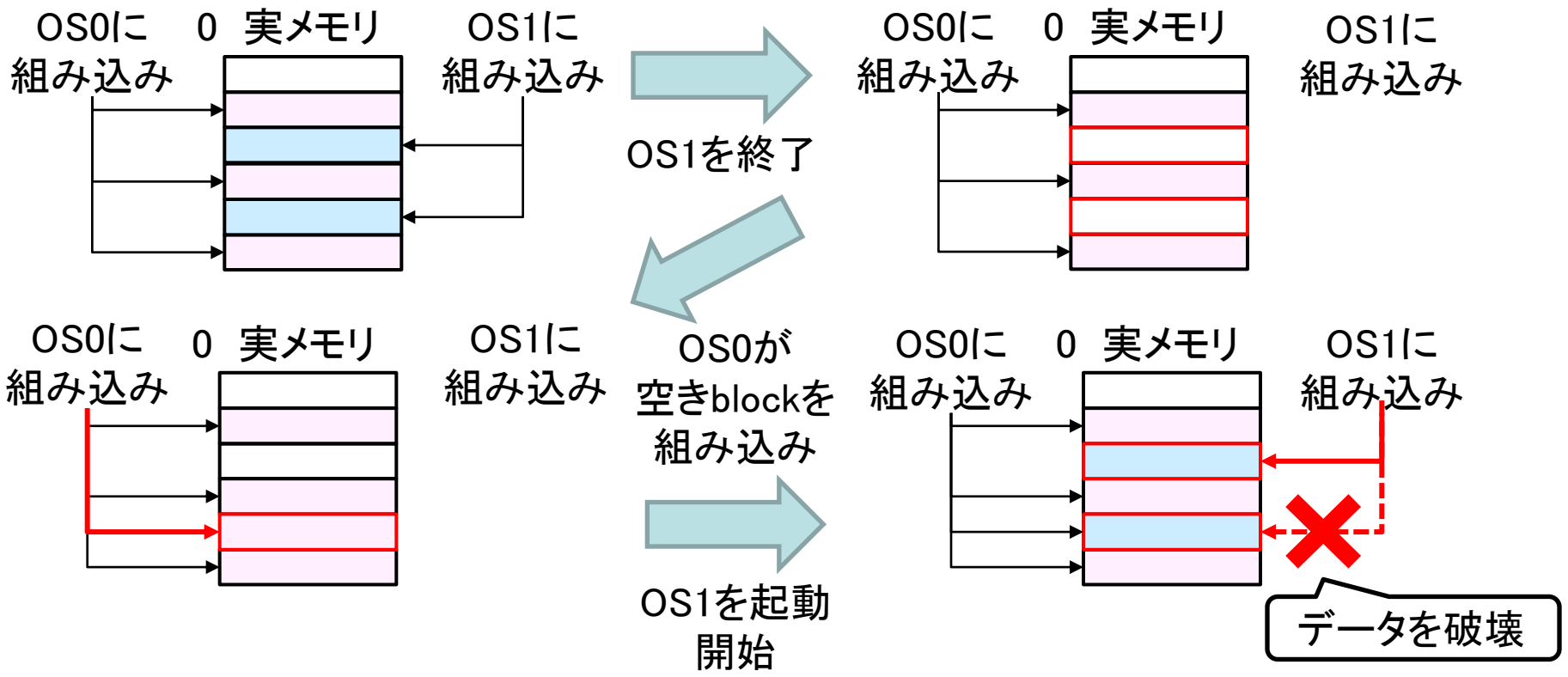
各OSはblockの組み込み状態を個別に管理

(A) 各OSは互いのblock組み込み状態を把握不可能

(B) 同一のblockを誤って複数OSに組み込むとデータを破壊

(対処1) 共有管理表を用いて実メモリ領域を排他

OS単独リブートへの対処



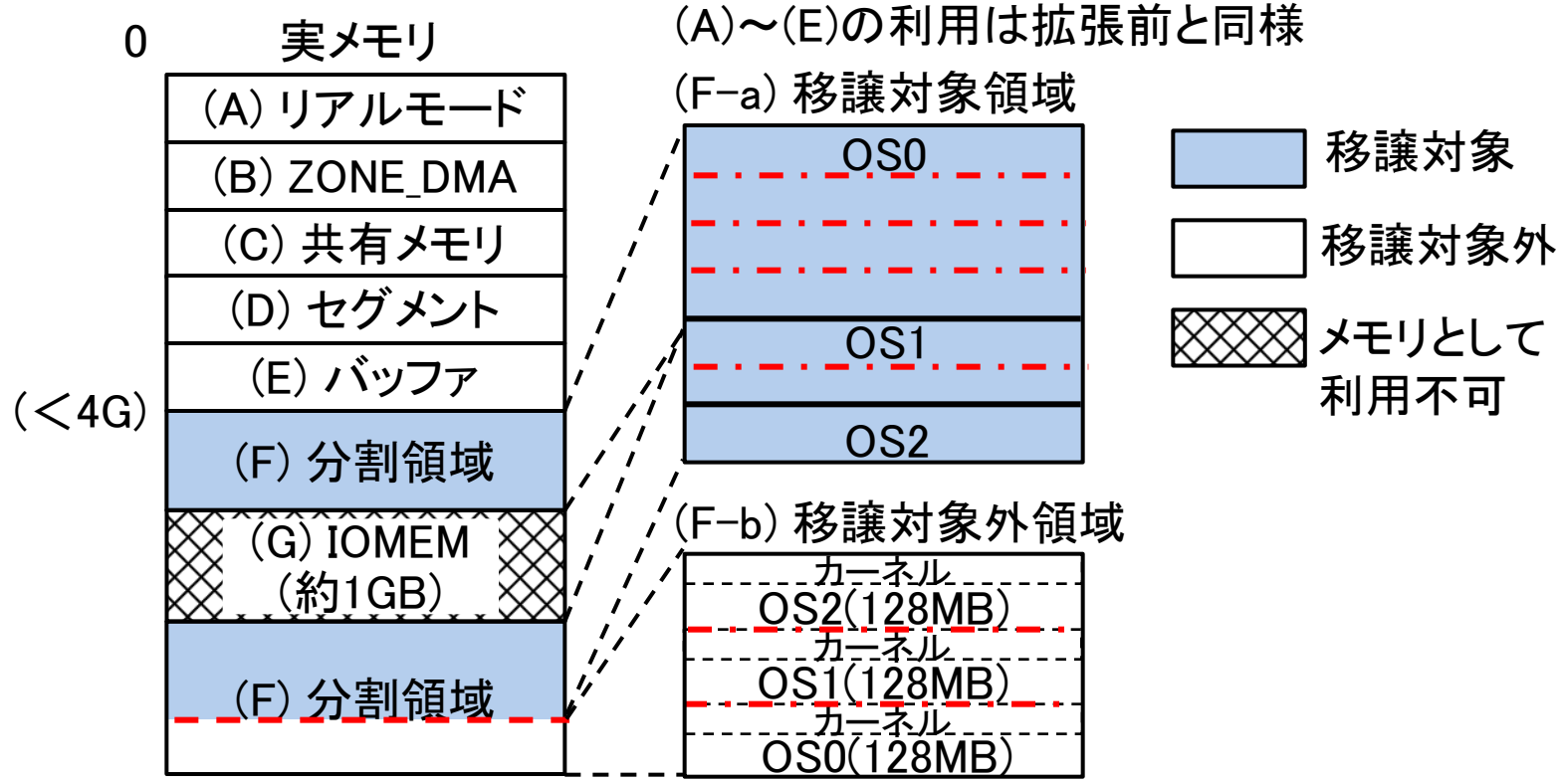
OS単独リブート時におけるblockの排他が不完全

(対処2) OS起動に最低限必要なblockを静的に分配

(A) 起動処理の複雑化を防ぎ，カーネル改修量を抑制

(B) OS起動後に実メモリを移譲し，実メモリ量を調整

OS単独リブートに対応した実現方式



分割領域(F)を移譲対象領域(F-a)と移譲対象外領域(F-b)に分離

(1) 移譲対象領域(F-a):OS起動後に移譲可能

(2) 移譲対象領域(F-b):OS起動時に静的に分配

目次

評価の観点

以下をKVMのバルーンドライバと比較

(1) 基本性能：実メモリの動的再分配の処理時間

(2) 実メモリ負荷分散効果：ベンチマークプログラムの走行に必要な実メモリ量の累積

(A) PARSEC-3.0のcannealとfreqmineを使用

(B) ベンチマークプログラムの使用メモリ量の推移から，必要な実メモリ量を推計

＜評価環境＞

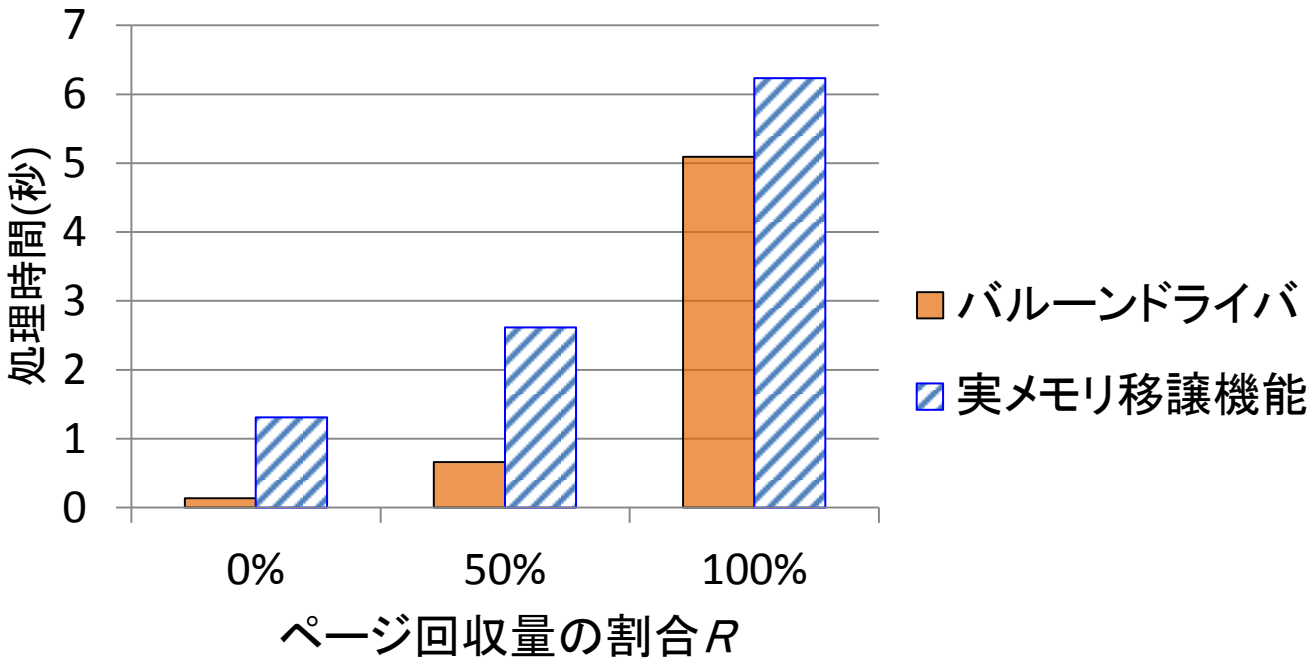
(1) プロセッサ： Intel Core i7-870 (2.93GHz, 4コア)

(2) メモリ： DDR3-1333 8GB

(3) HDD： 500GB × 3

(4) OS： すべてFedora14 x86_64 (Linux Kernel 3.0.8)

基本性能

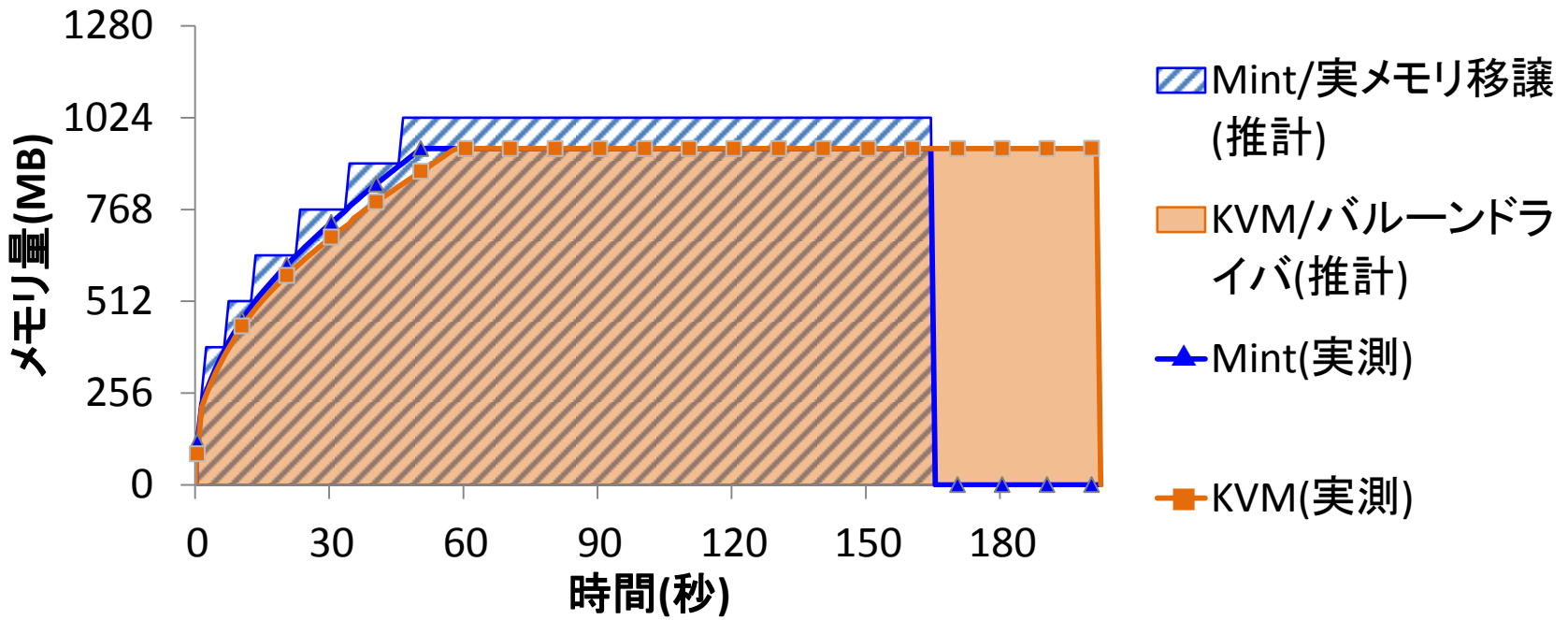


実メモリ移譲機能の処理時間はバルーンドライバよりも長大

∵ (1) 実メモリ動的再分配に要する処理量の差

(2) ページ追い出し発生量の差

実メモリ負荷分散効果(canneal)



実メモリ使用量の累積[GB・秒]

実メモリ移譲	149.3 (88.7%)
バルーンドライバ	168.4 (100%)

実メモリ移譲機能は実メモリ使用量の累積が小さく，実メモリ負荷分散効果が高い

∵ KVMではTLBミス時のアドレス変換のオーバヘッド大

⇒ プログラムの実行時間が長大化

まとめ

実メモリ移譲機能を実現する上での課題と対処について述べ，評価により性能を明らかにした

＜実メモリ分配＞

(課題) 各OSに自由な割合で実メモリ領域を分配

(対処)カーネル配置領域の先頭アドレスを各OS で変更可能にする

＜実メモリ移譲機能＞

(課題1) 実メモリ領域の排他

(課題2) OS単独リブートへの対応

(対処1) 共有管理表を用いて実メモリ領域を排他

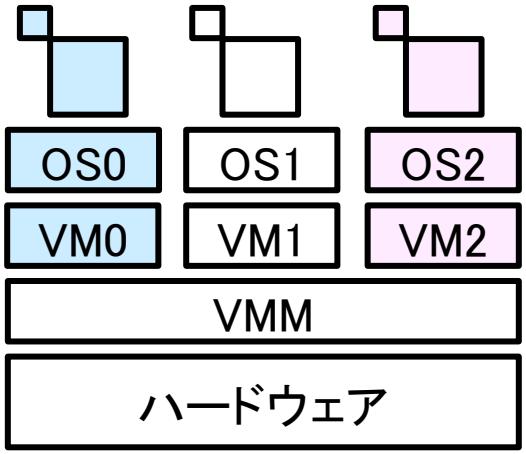
(対処2) OS起動に最低限必要なblockを静的に分配

＜評価＞

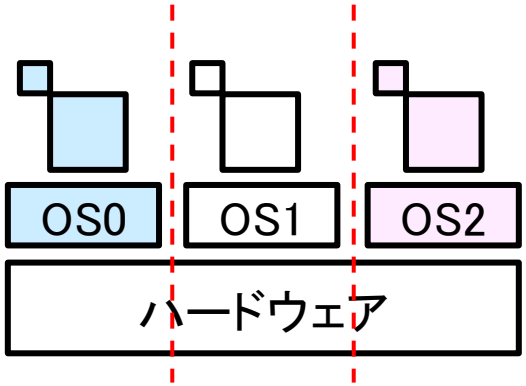
(1) 実メモリ移譲機能の処理時間はバルーンドライバに比べて長大

(2) 実メモリ移譲機能の実メモリ負荷分散効果はバルーンドライバに比べて高い

VM方式とMintの比較



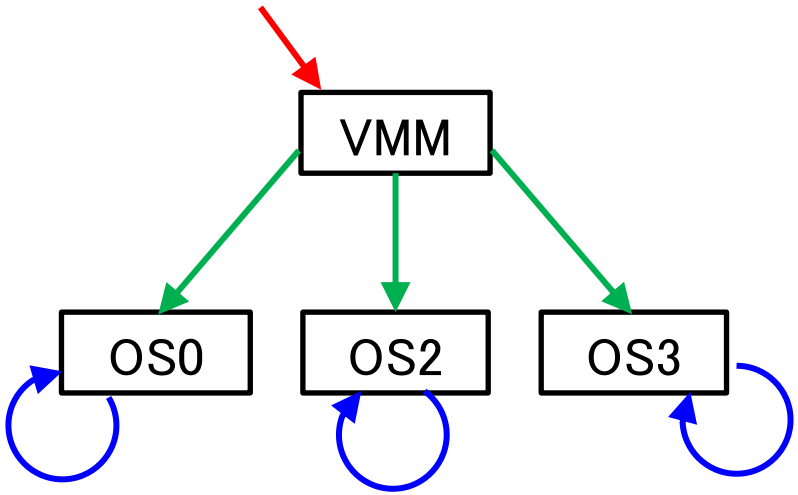
VM方式



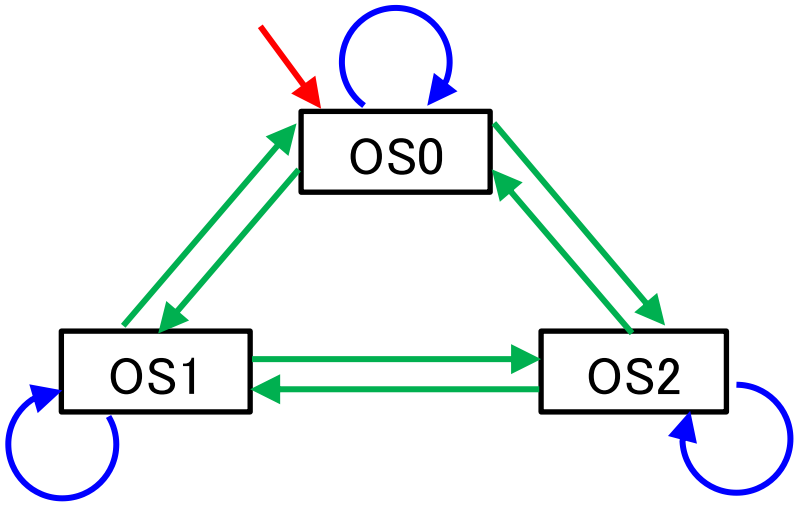
Mint

	VM方式	Mint
利点	カーネルを改修不要 資源のオーバコミットが可能	実計算機と同等の性能でOSを走行可能 OS間の処理負荷の影響を抑制可能
欠点	実計算機に比べて性能が低下 OS間の処理負荷の影響が発生	カーネルの改修が必要 走行可能なOS数はコア数により制限

自由な起動と終了



VM方式におけるOSの起動と終了



Mintにおける OSの起動と終了

計算機電源投入時のOS起動
他OSの起動
OSの単独リブート

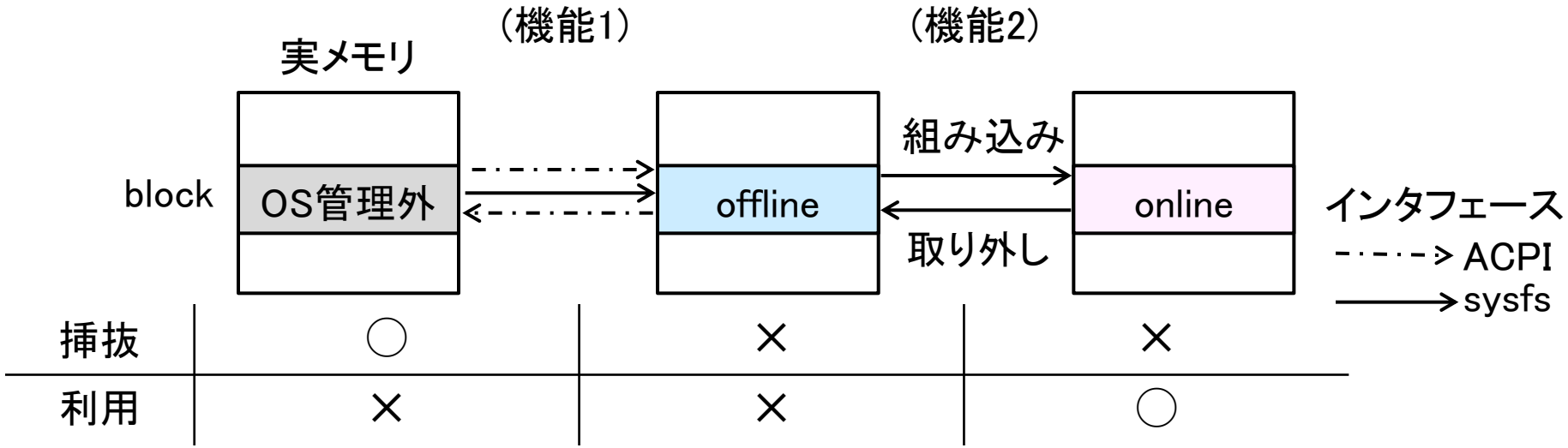
(1) VM方式におけるOSの起動と終了

VMMのみが他OSを起動可能 ⇒ VMMが単一障害点に

(2) Mintにおける OSの起動と終了

全てのOSが他OSを起動可能 ⇒ 単一障害点を排除

Linuxのメモリホットプラグ機能



＜機能＞

blockを最小単位として適用可能

実メモリを先頭から固定長に分割した単位
x86_64では1blockは128MB

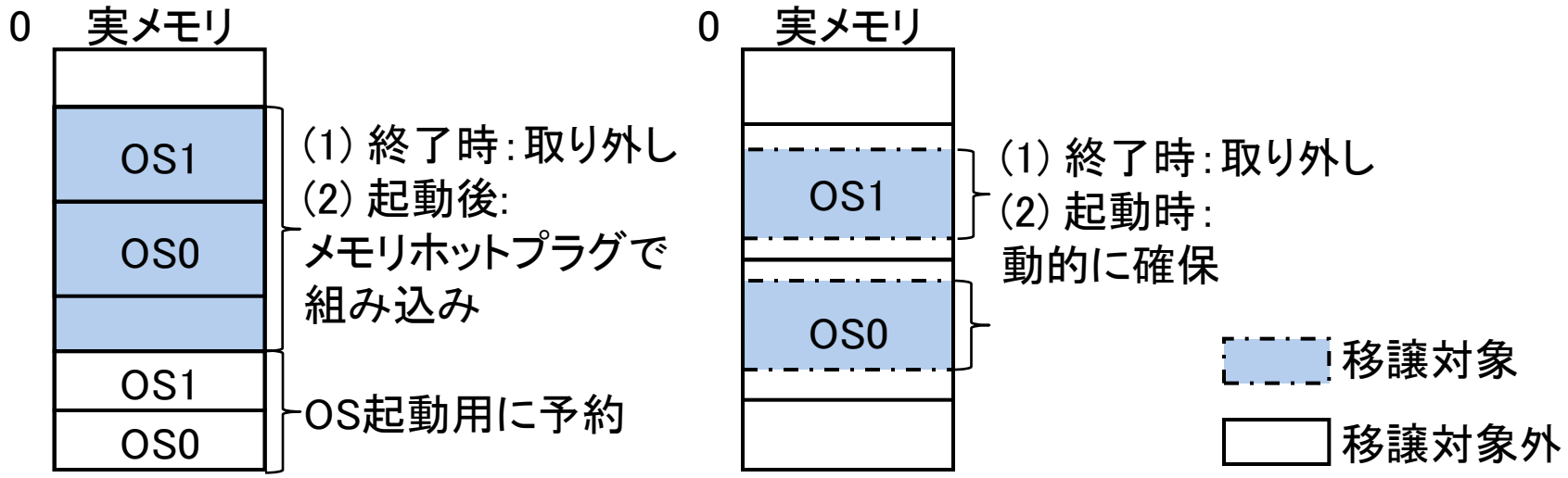
(機能1) ページ管理データ構造の追加と削除

(機能2) 実メモリ領域の使用可否の切り替え

＜インタフェース＞

対象blockを指定し，sysfsへの操作として実行可能

OS単独リブートへの対処



＜対処方式＞

(方式1) 移譲対象外領域のみをOSを起動時に分配する

(A) 分配法を変更し，移譲対象領域と移譲対象外領域を分離

(B) 移譲対象外領域：OS起動用に予約

(C) 移譲対象領域：OS起動後に実メモリ移譲で組み込み

(方式2) OS起動時に実メモリを動的確保可能にする

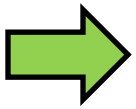
(A) 分配法を維持

(B) 必要な実メモリをOS起動時に未使用blockから動的確保

OS単独リブート対処方式の比較

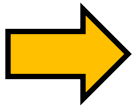
	分配方法の変更	OS起動に利用する領域	実現工数
(方式1)	あり	固定	少
(方式2)	なし	動的	多

(1) (方式1)ではOS起動段階での実メモリ確保操作が不要



工数を削減

(2) (方式2)ではOS起動段階での実メモリ量を調整可能



利点は限定

実現工数の少ない(方式1)で実現

実メモリ領域の排他

各OSへの実メモリ分配の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必要

共有管理表を用いて実メモリ領域を排他

(1) 構造

各blockに対応する1バイトエントリを実メモリ連続に配置

(2) 格納データ

(A) 使用中のblock：使用OSのBSPコアIDを格納

(B) 未使用のblock：未使用を示すフラグを格納

(C) 移譲対象外のblock：移譲対象外を示すフラグを格納

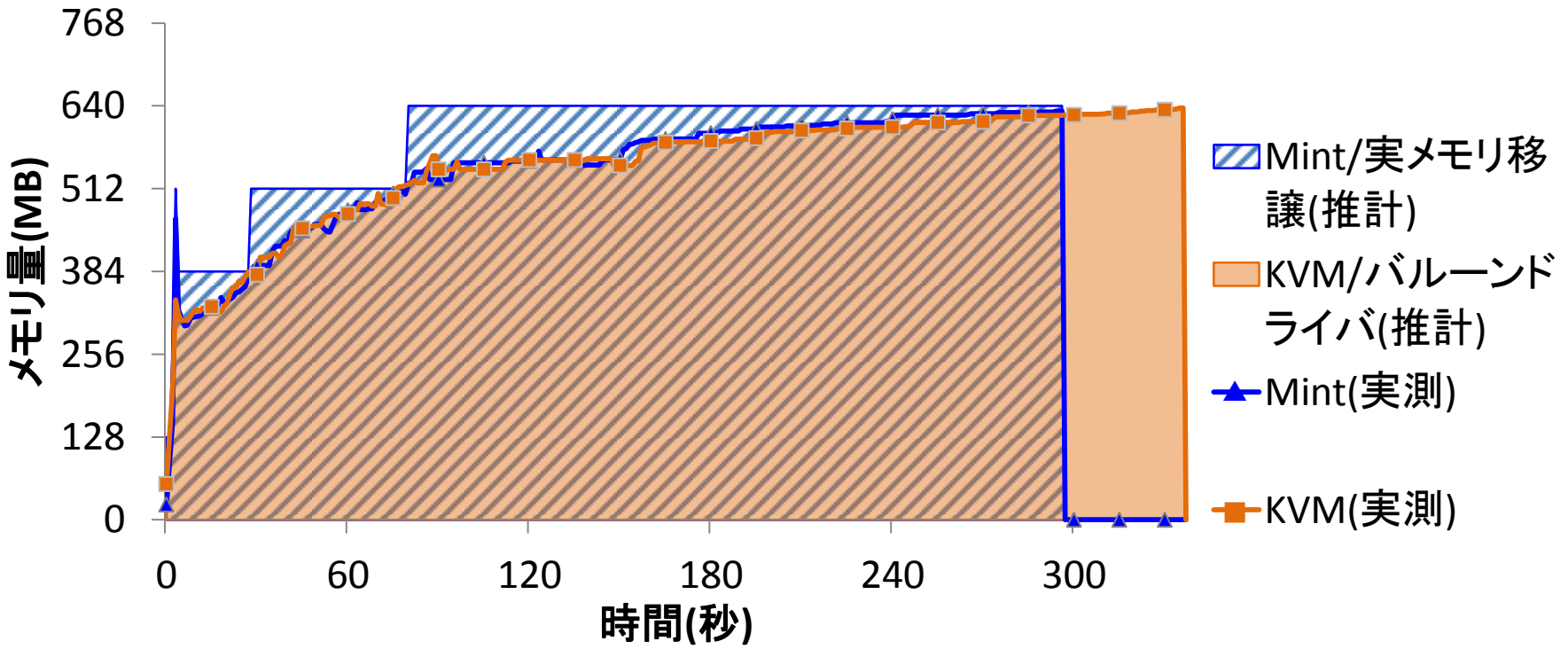
(3) エントリの排他

compare and exchange (CMPXCHG) 命令を使用

(4) 操作の契機

sysfsの操作に連動

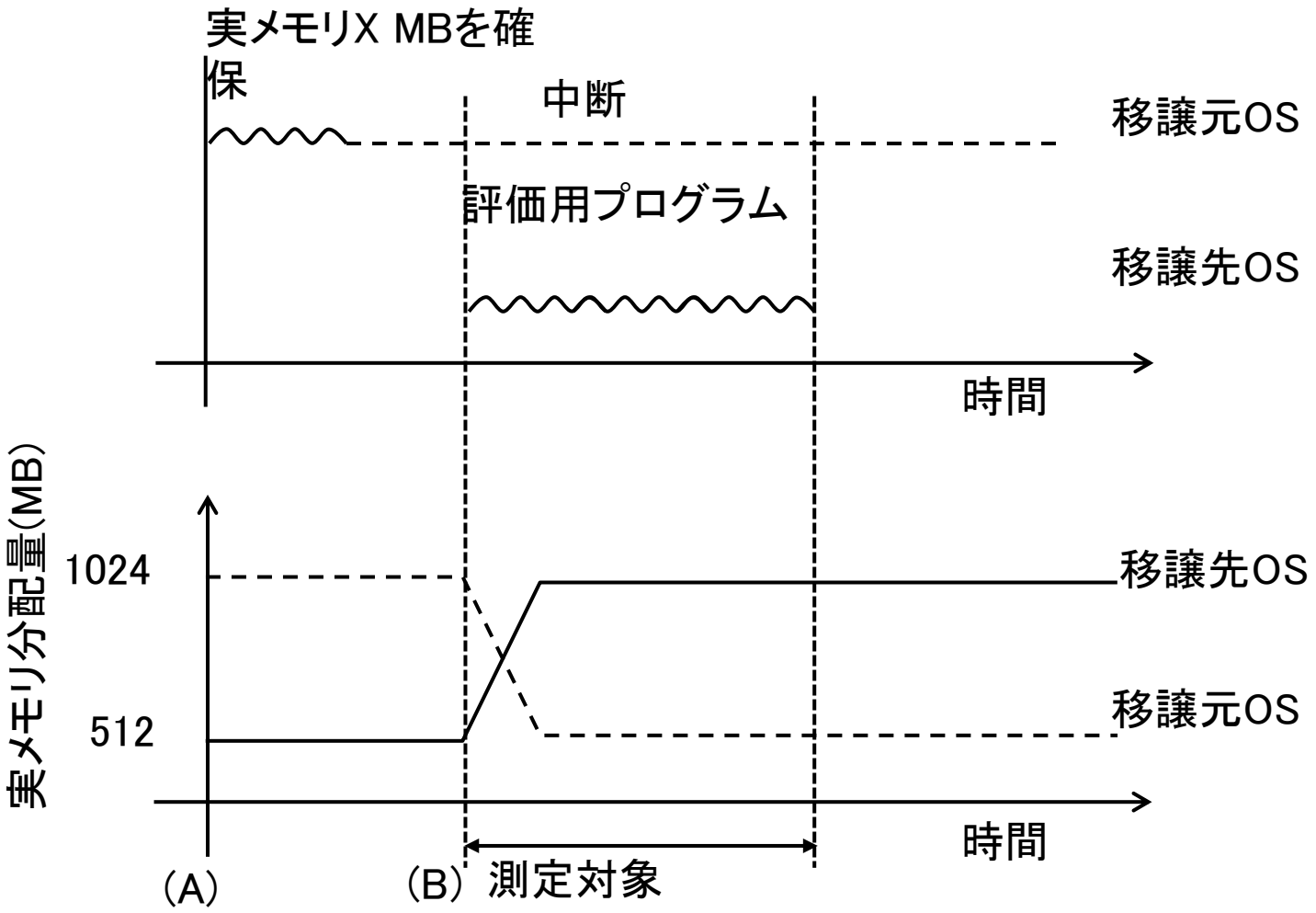
実メモリ負荷分散効果(freqmine)



実メモリ使用量の累積[GB・秒]

実メモリ移譲	171.6 (95.0%)
バルーンドライバ	180.6 (100%)

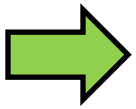
測定方法(基本性能)


実メモリX MBを確保
中断
移譲元OS
評価用プログラム
移譲先OS
時間
実メモリ分配量(MB)
1024
512
移譲先OS
移譲元OS
時間
(A)
(B)
測定対象

OS単独リブート対処方式の比較

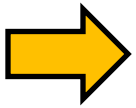
	分配方法の変更	OS起動に利用する領域	実現工数
(方式1)	あり	固定	少
(方式2)	なし	動的	多

(1) (方式1)ではOS起動段階での実メモリ確保操作が不要



工数を削減

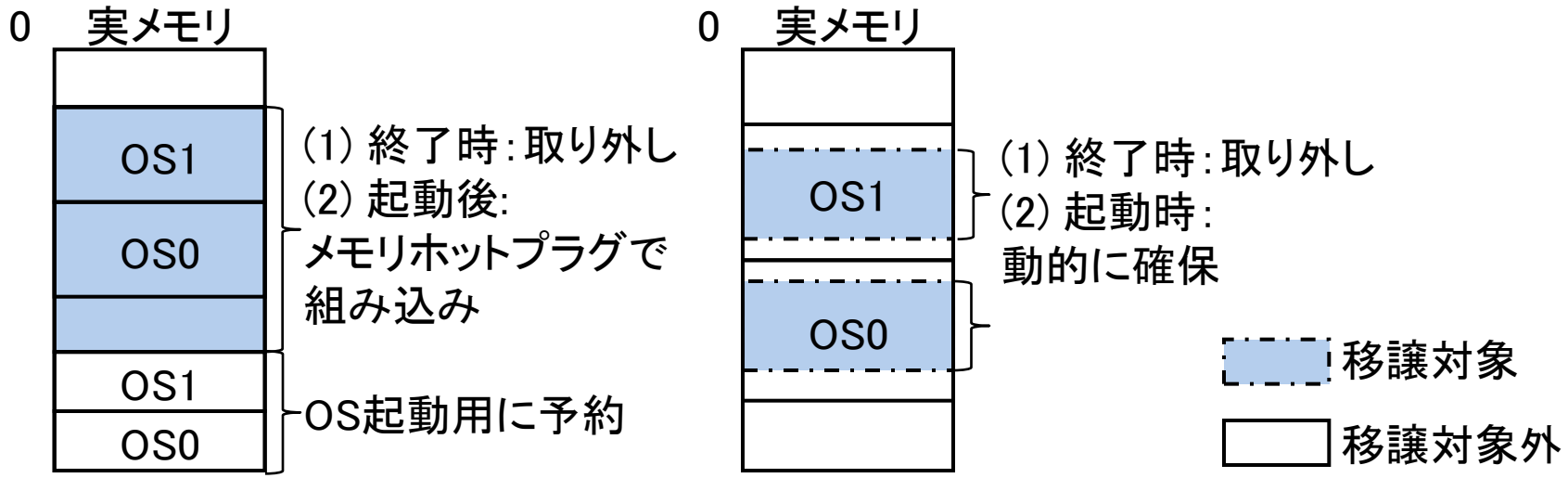
(2) (方式2)ではOS起動段階での実メモリ量を調整可能



利点は限定

実現工数の少ない(方式1)で実現

OS単独リブートへの対処



＜対処方式＞

(方式1) 移譲対象外領域のみをOSを起動時に分配する

(A) 分配法を変更し，移譲対象領域と移譲対象外領域を分離

(B) 移譲対象外領域：OS起動用に予約

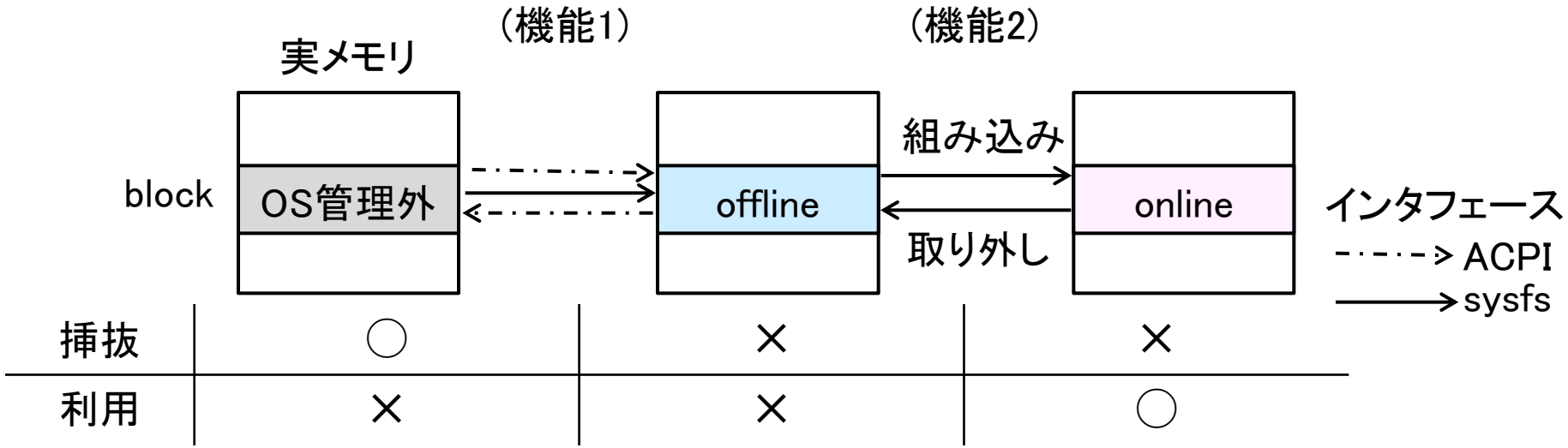
(C) 移譲対象領域：OS起動後に実メモリ移譲で組み込み

(方式2) OS起動時に実メモリを動的確保可能にする

(A) 分配法を維持

(B) 必要な実メモリをOS起動時に未使用blockから動的確保

実メモリ移譲の考え方



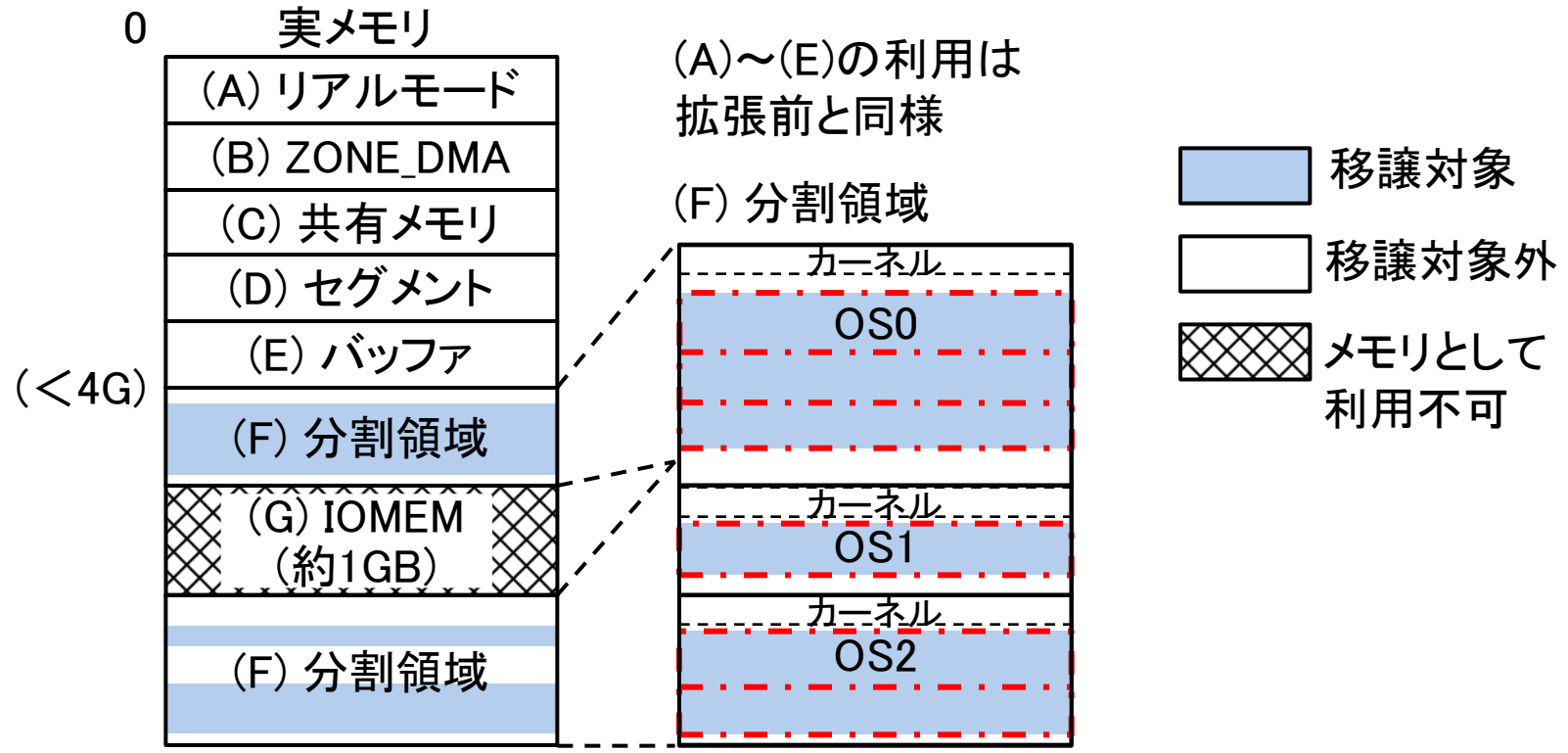
Linuxのメモリホットプラグ機能を利用して実現

(1) システム走行中に実メモリモジュールの交換を実現
(2) block(128MB)を最小単位として以下を適用可能
(機能1) ページ管理データ構造の追加と削除
(機能2) 実メモリ領域の使用可否を切り替え

＜手順＞

blockを移譲元OSから取り外し，移譲先OSに組み込み

実メモリ移譲の基本方式と課題

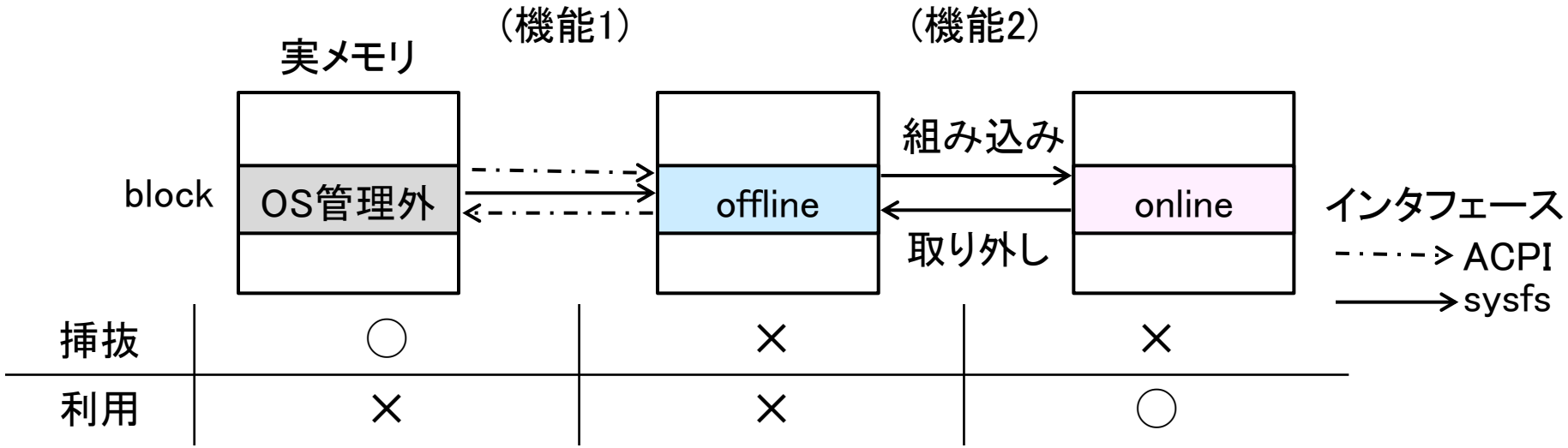


分割領域(F)中の各OSの分配領域のうち，先頭と終端を除きblockに沿った領域を自由に移譲可能

(課題1) 実メモリ領域の排他

(課題2) OS単独リブートへの対応

実メモリ移譲の考え方



Linuxのメモリホットプラグ機能を利用して実現

(1) システム走行中に実メモリモジュールの交換を実現
(2) block(128MB)を最小単位として以下を適用可能
(機能1) ページ管理データ構造の追加と削除
(機能2) 実メモリ領域の使用可否を切り替え

＜手順＞

blockを移譲元OSから取り外し，移譲先OSに組み込み