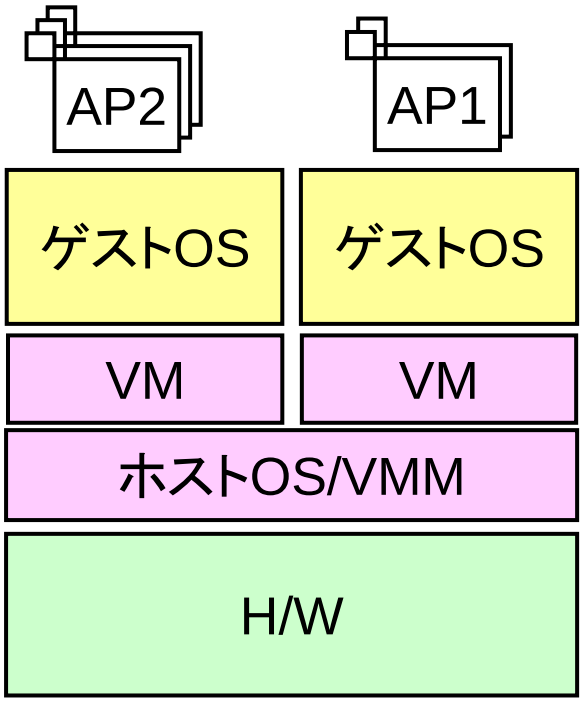
複数OS走行環境におけるマルチコアCPUの計算資源分割方式の検討

牛尾 裕
岡山大学 大学院 自然科学研究科
乃村研究室

背景

1台の計算機上で複数のOSを走行させる技術の研究が活発



＜仮想計算機(VM)方式＞

(1) 仮想マシンモニタ(VMM)により，計算機を複数の仮想的な計算機(VM)に分割

(2) 各VM上でOS(ゲストOS)を走行

計算機資源を効率的に利用可能

＜VM方式の問題点＞

各OSは実計算機と同等の性能で動作できない

実計算機に近い環境で，効率的にH/W資源を利用できる方式として，TwinOS方式とMint方式が提案されている

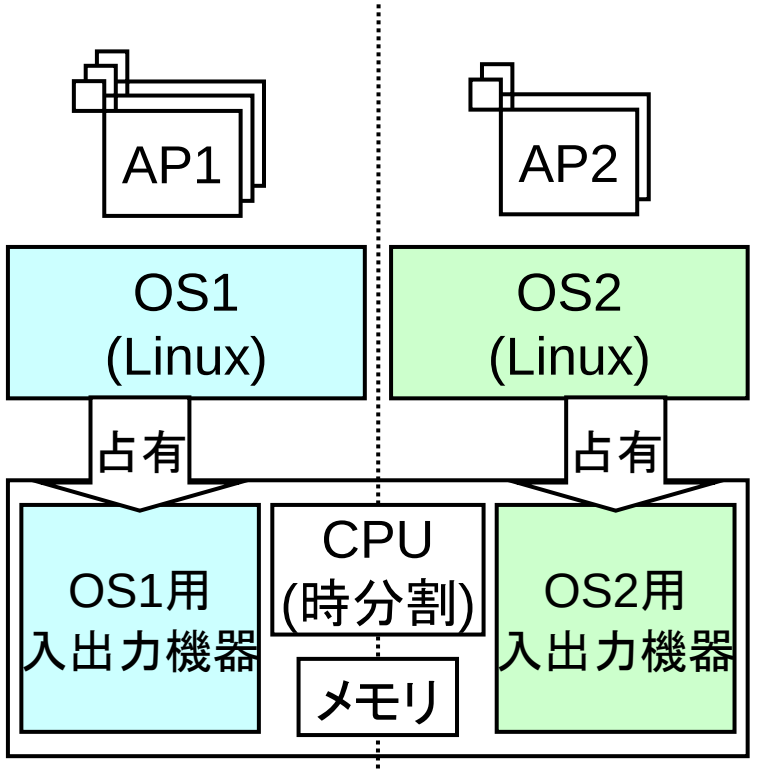
TwinOS方式

＜特徴＞

(1) 1台の計算機上に複数のOSが走行

(2) 各OSは実計算機上で直接走行

(3) ハードウェアを分割して占有



＜ハードウェアの分割方法＞

(1) 実メモリ: 空間分割して各OSで占有

(2) 入出力機器: 排他的に各OSに占有

(3) CPU:シングルコアCPUを2つのOSで時分割して占有

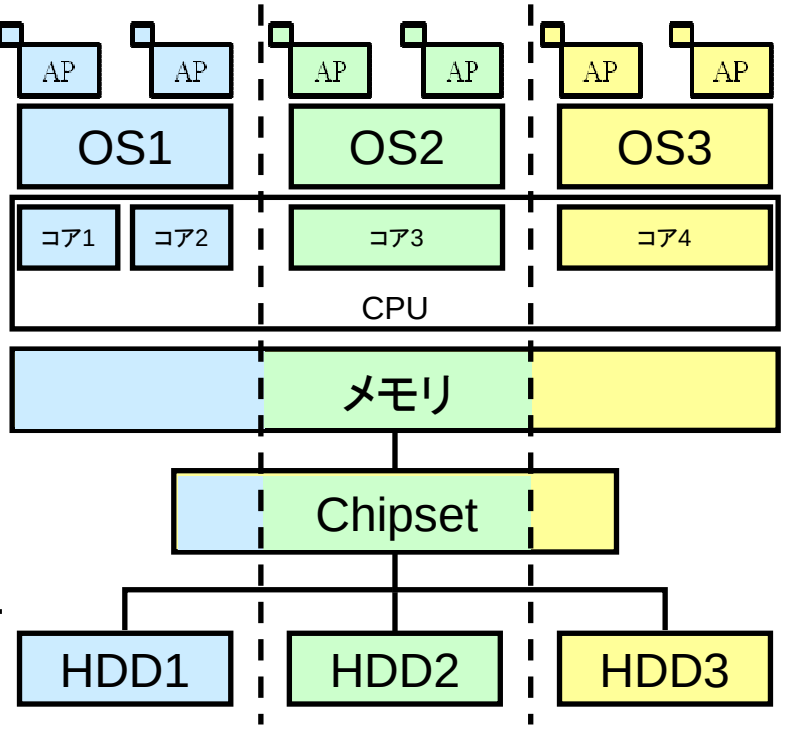
Mint方式

＜特徴＞

(1) 1台の計算機上に複数のOSが走行

(2) 各OSは実計算機上で直接走行

(3) ハードウェアを分割して占有



＜ハードウェアの分割方法＞

(1) 実メモリ: 空間分割して各OSで占有

(2) 入出力機器: 排他的に各OSで占有

(3) CPU:マルチコアCPUを複数のOSでコア分割して占有

CPU分割方式

＜TwinOS方式＞

シングルコアCPUを時分割

(利点) 並行に走行可能なOS数がコア数に依存しない

(欠点1) マルチコアCPUに対応していない

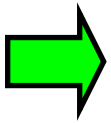
(欠点2) 計算資源の共有により独立性が低い

＜Mint方式＞

マルチコアCPUをコア分割

(利点) 計算資源を共有しないため各OSの独立性が高い

(欠点) コア数より多くのOSを並行に走行できない

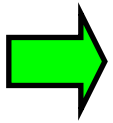


資源の分割粒度はコア数に依存するため粗い

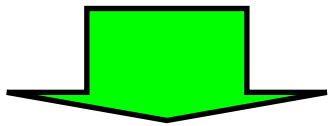
効率的な計算資源の利用法

＜現在の計算機＞

CPU性能に比べ入出力性能は低い



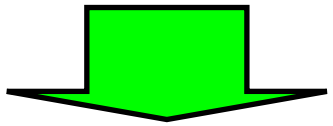
入出力ネックのためCPU性能を最大限に利用できない



CPUの時分割方式は計算資源の効率的な利用法として妥当

＜マルチコアCPUの時分割＞

(要求) Mint方式にTwinOS方式の時分割方式を取り入れたい



Mint方式のコア分割方式とTwinOS方式の時分割方式を組み合わせ計算資源を効率的な利用法を検討

以下資料

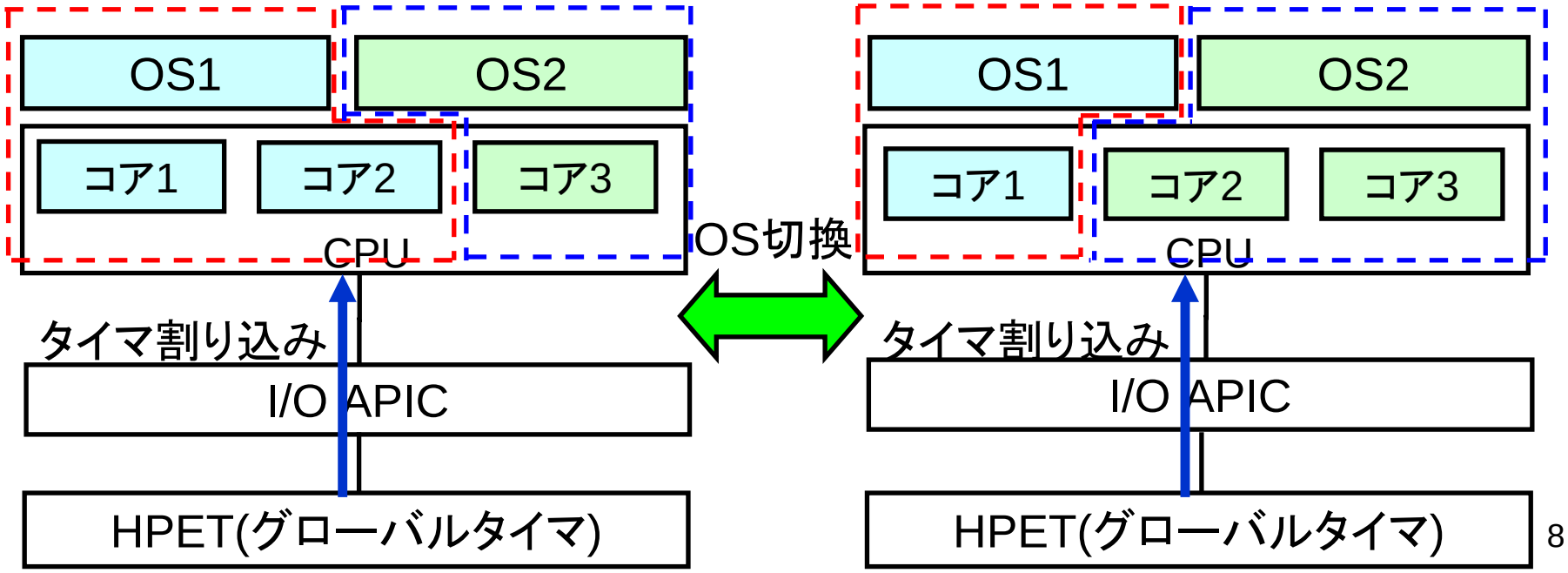
併用分割方式

コア分割方式と時分割方式を併用

＜特徴＞

(1) 細かい計算資源の分割が可能である

(2) 計算機性能の低下が少ない



マルチコア分割方式のまとめ

	コア分割方式	併用分割方式	全コア時分割方式
コア分割方式の種類	コア分割	コア分割と時分割	時分割
並行走行可能なOS数	コア数により制限	制限なし	制限なし
OS切換回数	なし	少ない	多い
各OSの独立性	高い	中程度	低い
ソースコード改修行数	少ない	多い	少ない
計算資源の分割粒度	粗い	細かい	細かい

＜併用分割方式＞

OS切換回数と各OSの独立性で優れる

＜全コア時分割方式＞

ソースコードの改修行数で優れる

細かい計算資源の分割が可能であり，計算機性能の低下が少ない併用分割方式について検討

併用分割方式の課題

(課題1) 走行環境の保存と復元

時分割時の各コアにおける走行環境の保存と復元方法

(課題2) 時分割コアの計算資源の割り当て

時分割コアの計算資源の割り当て方法

(課題3) 後続OSの起動

先行OSの環境を破壊しない後続OSの起動法

(課題4) 後続OS起動時の先行の環境保護

後続OS起動時における先行OSの環境保護方法

(課題5) OS終了時の他のOSの環境保護

OS終了時の他のOSの環境保護方法

(課題6) OS終了時のOS切換の変更

OS終了時のOS切換の変更方法

(課題1) 走行環境の保存と復元

OS切換時には切換元OSの走行環境を保存し，切換先OSの環境を復元

＜保存と復元を行う情報＞

(1) OSの仮想空間情報：各OSが独立にもつ仮想空間を構成するセグメントテーブルとページテーブル情報

(2) 割り込み管理情報：割り込み発生時にコアが参照する割り込み管理テーブル情報

(3) レジスタ情報：OSやAP処理で用いられるコアのレジスタ値情報

＜復元時のコアの識別＞

自コアと同じコアでの走行環境を復元する必要

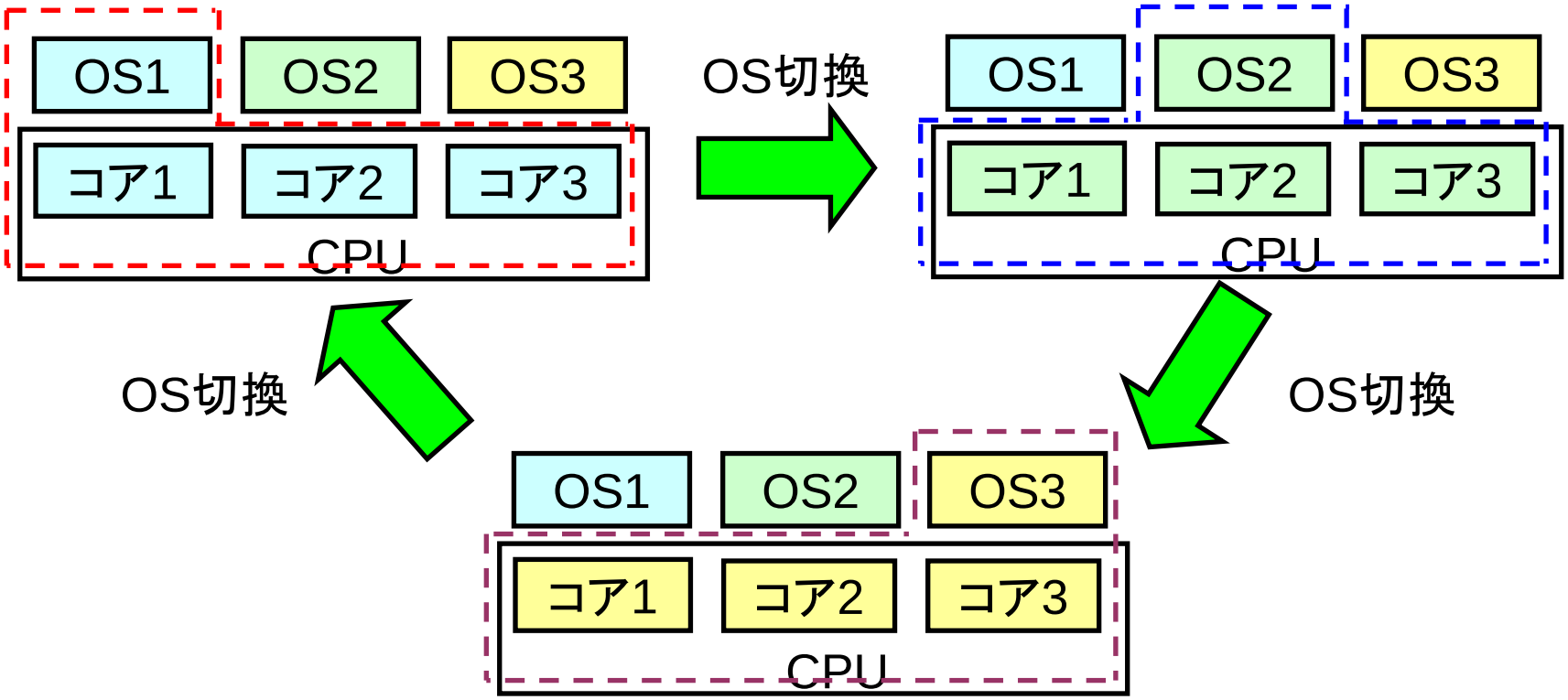
⇒ 走行コアのコア識別番号を参照

Linuxがコアを識別するために用いる値

(課題2) 時分割コアの計算資源の割り当て

コアを時分割で各OSに占有させている場合，各OSに計算資源を割り当てる必要

⇒ 順番に偏りなく切換先OSを決定



(課題3) 後続OSの起動方法

後続OSの占有するコアにより対処が異なる

(1) 先行OSが占有するコアを用いて，後続OSを時分割で起動

先行OSの占有するコアの1つを用いて後続OSを時分割で起動
TwinOS方式の後続OS起動処理と同様の処理で対処

(a) 更にコア分割で計算資源を占有

Linuxのコアのホットプラグ機能を利用してコアを占有

(b) 更に他のOSが占有するコアを時分割で占有

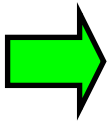
時分割するコアに対して時分割用のIDTを設定して占有

(2) 先行OSが占有していないコアを起動して，後続OSを起動

Mint方式のコア分割の起動処理と同様の処理で対処

(課題4) 後続OS起動時における他のOSの環境保護

後続OS起動時，走行中のOSの環境を破壊してはならない



既に起動しているOSの走行環境を保護する必要

(1) 割り込み登録処理

(a) 共有する資源の割り込みは同じベクタ番号を登録する

(b) 占有する資源の既に登録されている割り込みに重複しないように登録する

(2) ドライバ登録処理

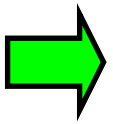
自OSが占有するデバイスのドライバのみ登録する

(3) デバイス初期化処理

全てのOSで共有するデバイスは，最初に起動するOSのみ初期化処理を行う

(課題5) OS終了時の他OSの環境保護

OSの正常終了時，他のOSは走行を継続出来なければならない



終了時，他のOSの環境を破壊してはならない

(1) CPU

(a) 時分割で走行しているコア

コアの終了処理を行わず，IDTを時分割を行う前の状態に復元

(b) コア割で走行しているコア

コアをHALT状態に変更

(2) I/O APIC

終了処理を行うOSに通知されている割り込みのエントリを除外

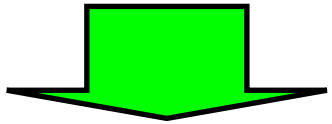
(3) グローバルタイマ

グローバルタイマの無効化処理を除外

(課題6) OS終了時のOS切換の変更

OSの正常終了時，他のOSは走行を継続出来なければならない

⇒ 終了したOSに対してOS切換を行ってはいけない



OS切換の変更により対処

＜OS切換の変更＞

(1) OS切換時に他OSの走行状態を確認

(a) OSの走行状態を識別するエントリ表を作成

(b) 各OSはOS起動時と終了時にエントリ表を修正

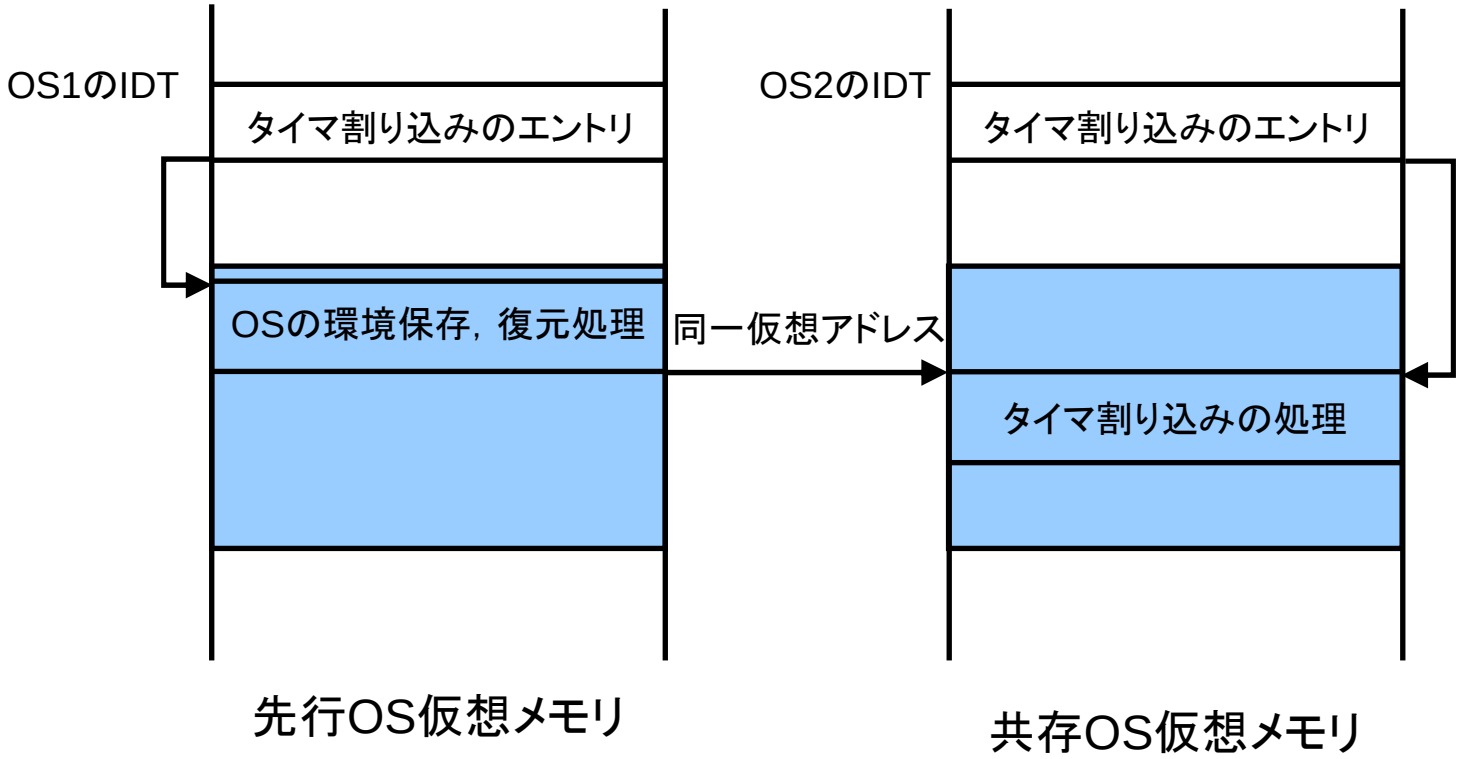
(2) 走行しているOSの中から切換先OSを決定

OS切換処理(1/2)

(1) 共存OSのIDTよりタイマ割り込みの開始アドレスを取得

(2) OSの環境保存，復元処理の終了アドレスの1ページ内相対位置がタイマ割り込み開始アドレスの先頭になるように実ページ作成

＜タイマ割り込み発生時の仮想空間＞



OS切換処理(1/2)

(3) 先行OSのページテーブルを変更し，作成した実ページに変更

(2) 先行OSのIDTの該当エントリをOS環境保存，復元処理開始アドレスに変更

＜タイマ割り込み発生時の仮想空間＞