Mintオペレーティングシステムにおけるコア移譲の管理機能について

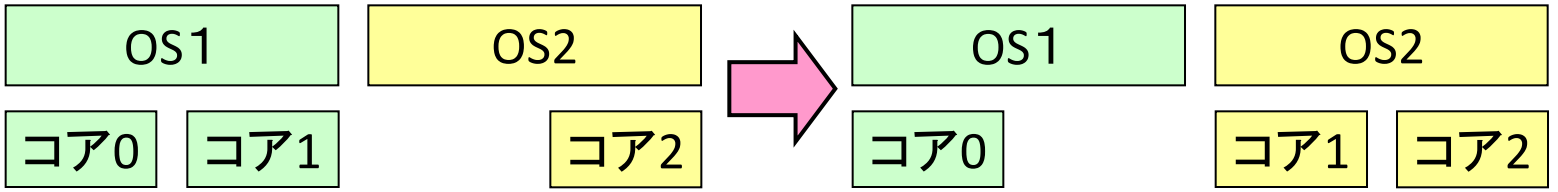
池田 騰

岡山大学 工学部 情報工学科

平成24年 2月17日

研究背景

「Mint」
(1) 1台の計算機上で複数のLinuxを独立に走行させる方式
(2) コアの動的割り当て機構によりOS間でのコア移譲を実現



(3) OS間での計算機資源が未管理

⇒ コアの移譲をユーザが手動で行う必要

(A) 各OSのコアの占有状況を個別に確認
(B) 移譲元のOSでコアを指定してコアの解放
(C) 移譲先のOSでコアを指定してコアの占有

⇓

コア移譲の管理機能を実現し，コアの移譲を自動化

Mintにおけるコアの識別の問題点

複数のOS間でコアIDの整合性を保持不可能

「コアID」

Linuxが自身の占有するコアの識別に使用するID

「Boot Strap Processor(BSP)」

(1) OSが最初に起動したコア

(2) コアID 0が割り当てられる

	OS1		OS2	OS3
	コア0(BSP)	コア1	コア2(BSP)	コア3(BSP)
LAPIC ID	LAPIC ID: 0	LAPIC ID: 2	LAPIC ID: 4	LAPIC ID: 6
OS1のコアID	コアID:0	コアID:1	コアID:2	コアID:3
OS2のコアID	コアID:1	コアID:2	コアID:0	コアID:3
OS3のコアID	コアID:1	コアID:2	コアID:3	コアID:0

コア移譲の管理機能における要求と課題

＜コア移譲の管理機能における要求＞

(要求1) 全てのコアの占有状況を把握

(要求2) OS間でコアの識別に関する整合性を保持

(要求3) 手動で行っている処理の自動化

＜コア移譲の管理機能を実現するための課題＞

(課題1) コアの占有状況を管理する機能の導入

(課題2) コア識別子変換機能の導入

(課題3) コアの自動指定機能の導入

コアの占有状況を管理する機能

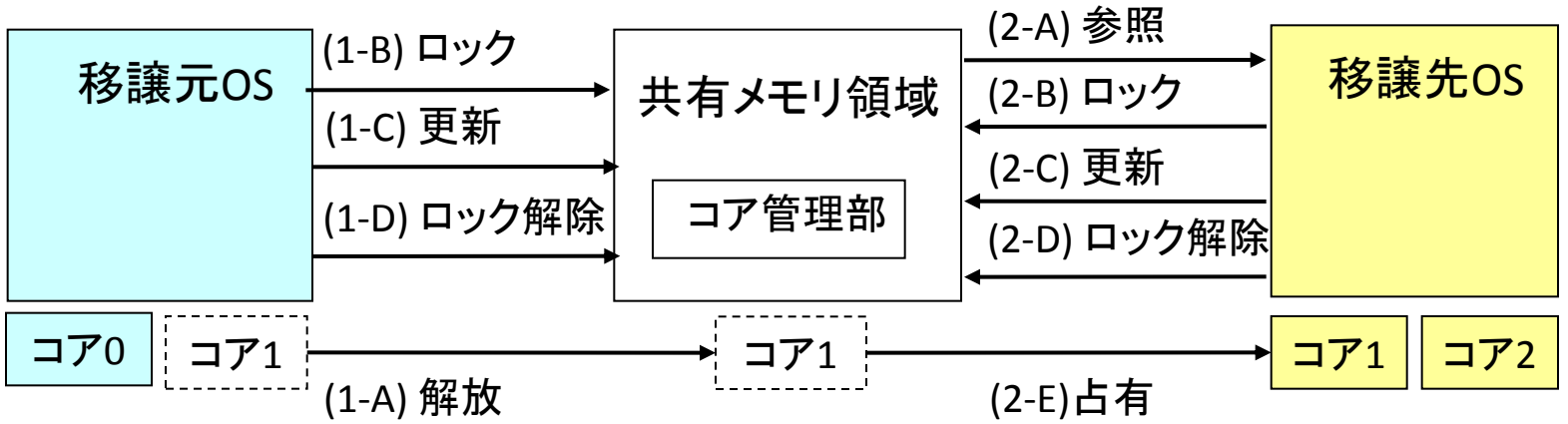
(対処案1) 特定のOSによるコアの集中管理

(対処案2) 各OSによるコアの分散管理

(1) 単一障害点が存在しない

⇒ OS間で依存関係が存在しない

(2) OS間で排他制御が必要



コア識別子変換機能

(対処案1) 全てのOSで同じコアIDを用いるように改変

BSPにコアID 0以外を割り当てる必要

⇒ カーネルに対する改変量が多く，実装が困難

(対処案2) 全てのOSで一意に設定されるコア識別子を用意し，
コアIDの代わりにコア移譲の際のみ使用

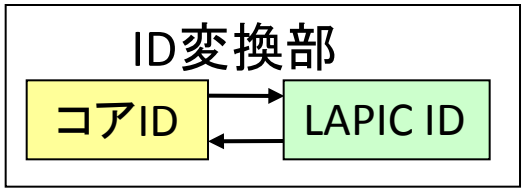
コアIDの代わりにLocal APIC IDを使用

⇒ 改変箇所を局所化でき，改変量を抑制可能

「Local APIC ID」

(1)電源投入時にハードウェアによって各コアに割り当てられるID

(2) Mintにおいて，OS間で一意の値を持つ



コアの自動指定機能

コア移譲の際にユーザがコアIDを指定する手順を自動化

(1) コア移譲APを作成

(2) コア移譲インタフェース部を作成

コア移譲APにコア管理部とID変換部の機能を提供

＜コア移譲APの機能＞

(1) 占有コア自動設定機能

(2) 解放コア自動設定機能

(3) OS間CPUホットプラグ機能

(4) コア管理部の更新機能

本発表のまとめ

＜実績＞

(1) コアの占有状況を管理する機能の導入

(2) コア識別子変換機能の導入

(3) コアの自動指定機能の導入

＜今後の課題＞

(1) コア移譲の際の割込み通知先の変更処理の実現

(2) CPUの使用率により自動でコアの解放と占有を行う方式の検討