

# Mintオペレーティングシステムにおける OS間の資源分配の研究

池田 騰

岡山大学 大学院 自然科学研究科

電子情報システム工学専攻

平成26年2月13日

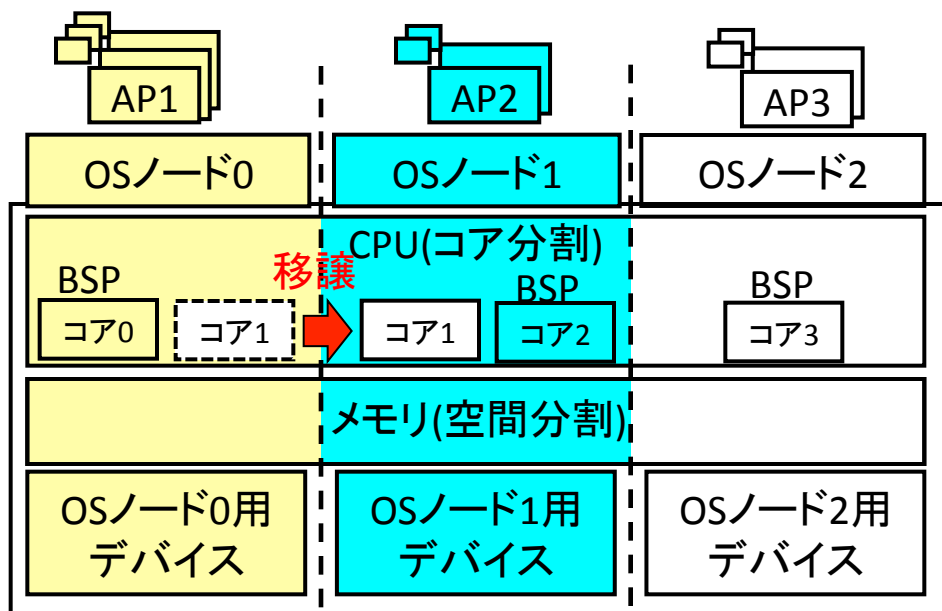
# はじめに

<複数のOSを同時に走行させる方式>

**VM** 問題点: 仮想化によるオーバヘッド, OS間の依存性

**Mint**

- (1) 仮想化によらず複数のLinuxを独立に走行
- (2) 各OSに資源を分配, 占有管理(コア, メモリ, 入出力機器)
- (3) OS間の資源移譲が可能



OSノード数増加 → 管理コスト増大

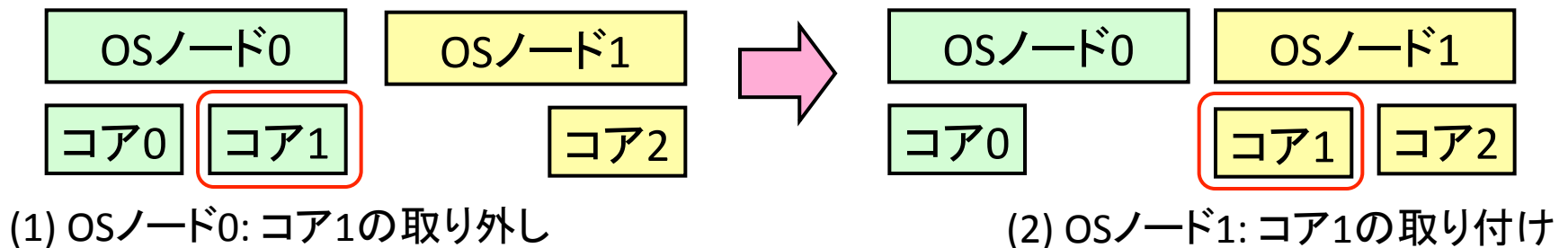
**資源管理機構が必要**

# Mintにおける資源移譲

Linuxの既存機能をそのまま提供するのみ:

- (1) メモリホットプラグ
- (2) Loadable Kernel Module(LKM)
- (3) CPUホットプラグ

## 手動によるCPUホットプラグの例



システム管理者の手動

➡ OSノード数の増加によるオペレーションミスの増大が問題

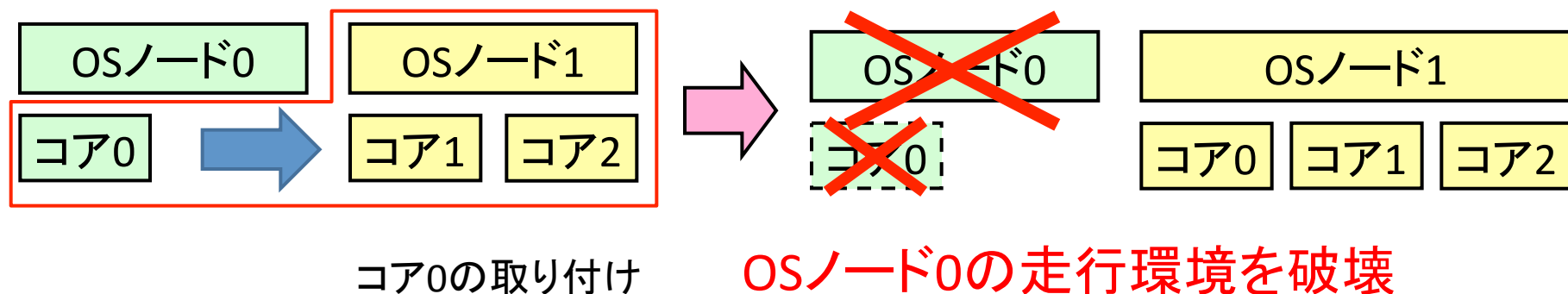
# Mintにおける資源移譲の問題点

(問題1) 移譲資源の誤選択により, 走行中のOSノード環境が破壊

∴ 他OSノードの資源を奪うことが可能

(問題2) 資源移譲の対象OSノードと対象資源をシステム管理者が選定

∴ 各OSノードは自身の資源のみ独立に管理



# 課題

(課題1) 他占有資源の保護機構の実現

資源移譲において他OSノード占有資源を保護

➡ 資源移譲の誤操作によるOSノード走行環境の破壊を防ぐ

(課題2) Mintにおける資源管理機構の提案

OSノードをまたがった資源管理を行う機構を提案

➡ システム管理者の手動管理による資源管理コストを削減

# (課題1)他占有資源の保護機構 (デバイス)

## <保護方式>

(方式1) 他占有資源の占有のみを防ぐ方式

実現方法: 資源移譲に用いる機能(LKM)を改変

(方式2) 自占有資源以外をOSノードから完全に隠蔽する方式

実現方法: OSによるデバイスの操作部分を改変

利点: OSノード間の独立性が非常に高い

∵ 資源として認識させないため, 占有しないデバイスは全てのソフトウェアから操作不可

OSノード間の独立性を高めるため, (方式2)を採用

➡ デバイスの操作部分に改変が必要

# PCIデバイスの操作

Linuxによるデバイス操作: PCIコンフィグレーション空間の読み書き

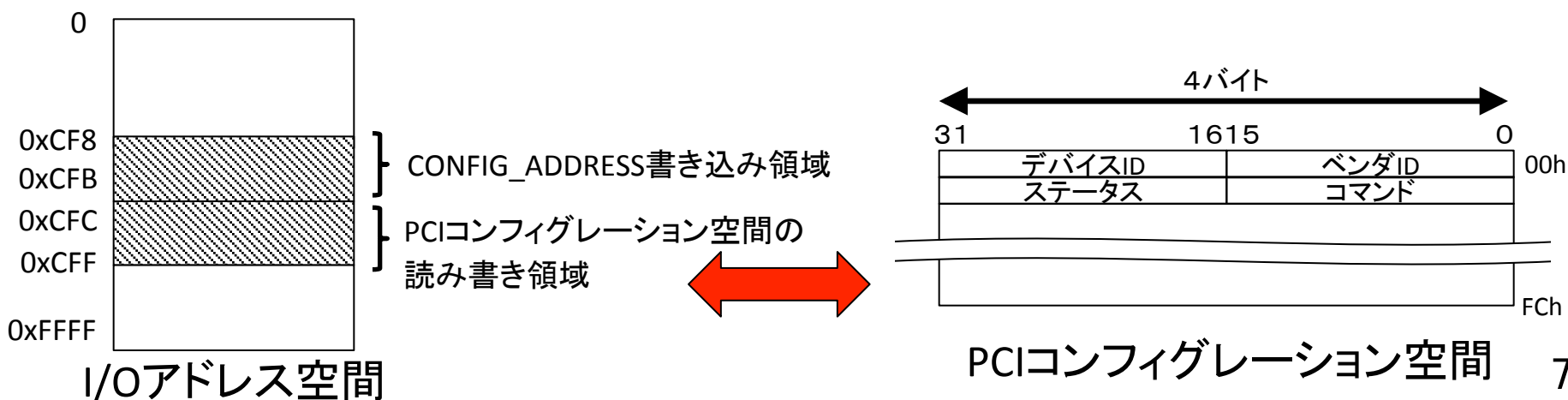
## <PCIコンフィグレーション空間>

- (1) PCIデバイスを操作するレジスタにアクセスするための空間
- (2) I/Oアドレス空間に専用の操作領域が存在

0xCF8~0xCFB: CONFIG ADDRESS書き込み領域

操作デバイスの指定

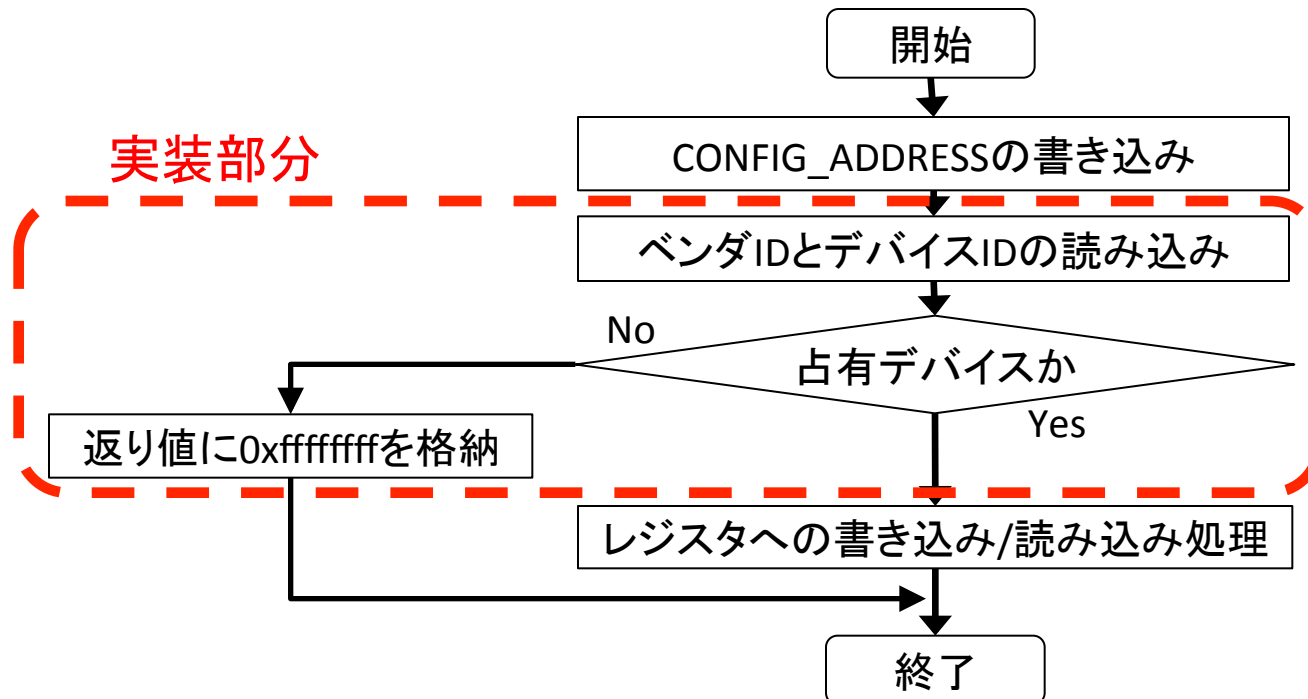
0xCFC~0xCFF: PCIコンフィグレーション空間の読み書き領域



# (方式2) Mintにおけるデバイス隠蔽 によるPCIデバイス保護

PCIコンフィグレーション空間を操作する関数を改変

- (1) レジスタへの書き込み/読み込み処理の直前でデバイス確認
- (2) 占有しないデバイスの場合, 空スロットへのアクセスに偽装





# (課題2)Mintにおける資源管理機構

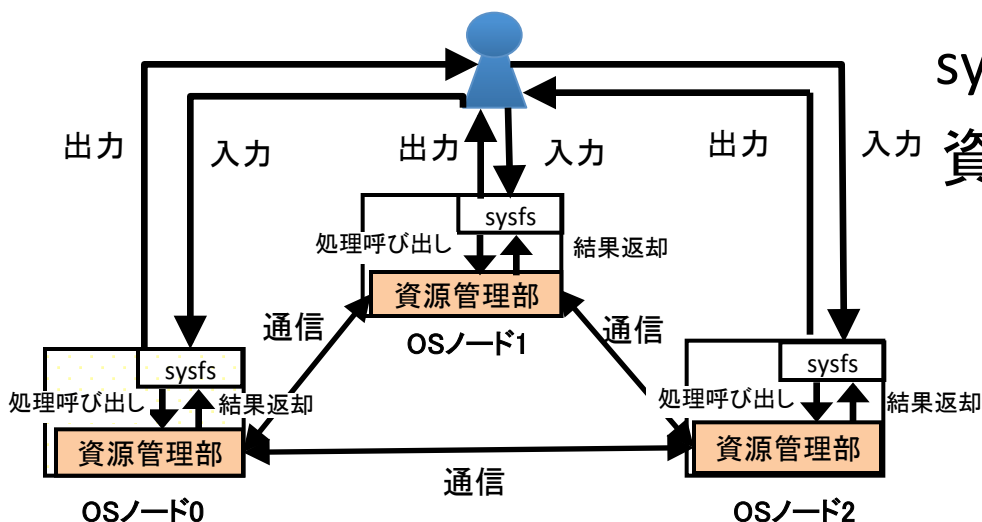
## <設計方針>

- (方針1) 分散管理によるOSノード間の独立性の維持
- (方針2) 既存のLinuxの資源管理インタフェース上に実装
- (方針3) OSノード数と資源量増加への拡張性

## <基本構成>

- (1) sysfsによるインタフェースの提供 (方針2)
- (2) 資源管理部による資源情報の分散管理 (方針1)(方針3)

システム管理者



sysfs: 個々の資源情報を管理

資源管理部: **新規作成部分**

- (1) 他OSノードと資源情報を通信
- (2) sysfsへ情報を提供


# 資源管理部における要求とBonjour

## <資源管理における要求>

(要求1) 特別な機能を持ったOSノードを要さない

(要求2) OSノードを資源で検索可能

(要求3) OSノードの構成や用途が分かるOSノード識別子の設定

 要求を満たす既存システムとしてBonjourの利用を検討

## <Bonjour>

(1) 集中管理部分不要のネットワーク機器同士での通信が可能  
(例. DHCPサーバ, DNSサーバ)

(2) 提供するサービスをもとにしたネットワーク上の機器の検索

(3) 各機器によるサービス情報が含まれたホスト名の自動設定

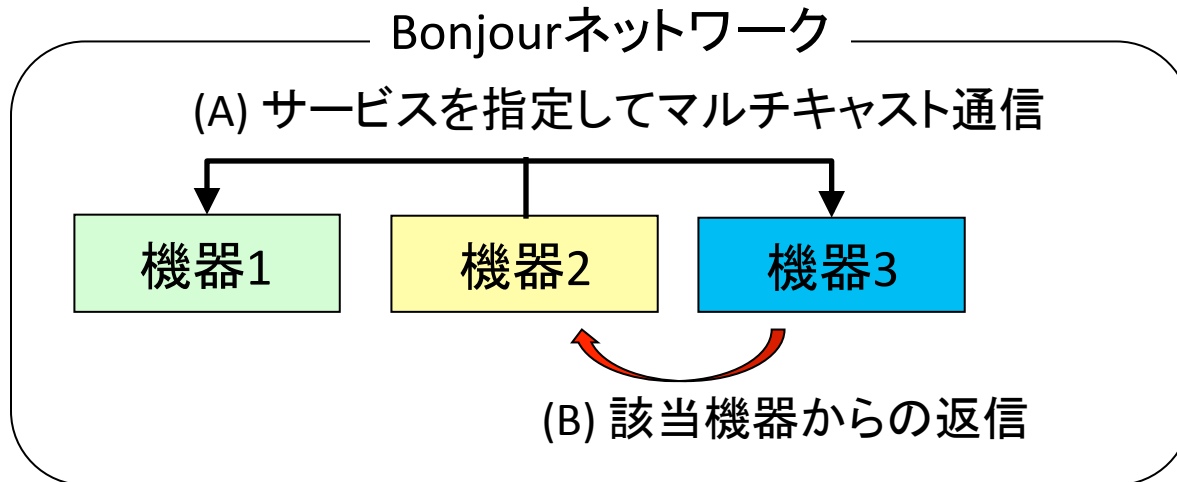
# Bonjourの機能

(1) Naming: 各機器が自身の情報を元にホスト名を設定する機能

マシン名 . サービス名 . プロトコル名 . ドメイン名

例: example.service.\_tcp.local

(2) Browsing: 目的のサービスを提供する機器を検索する機能



# 資源管理機構とBonjourの要求の違い

Bonjourの要求: サービス(資源)の有無による検索

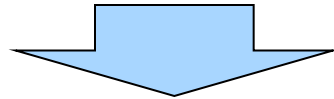
➡ サービス(資源)の量による検索が不可

○ コア, メモリを占有するOSノードの検索

✕ コアを2つ占有するOSノードの検索

資源管理機構の要求: 資源の有無と資源の量による検索

検索要求例: コア移譲のため, コアを3つ占有するOSノードを検索



資源の有無と資源の量の両方に対応する検索が必要

OSノード識別子とOSノードの検索に独自の設計が必要

# OSノード識別子

OSノード識別子: Bonjourにおけるホスト名相当  
OSノードの識別と検索に利用

コア数, メモリ量, および占有資源の種類を識別子に含有

(1) カーネル名, コア数, およびメモリ数を固定の領域として設定

(2) 4つ目以降の項目についてはデバイス等の有無を可変数で表記

➡ OSノード識別子のみで資源量と資源の有無を表現可能

カーネル名. コア数. メモリブロック数. デバイス . . .

固定順

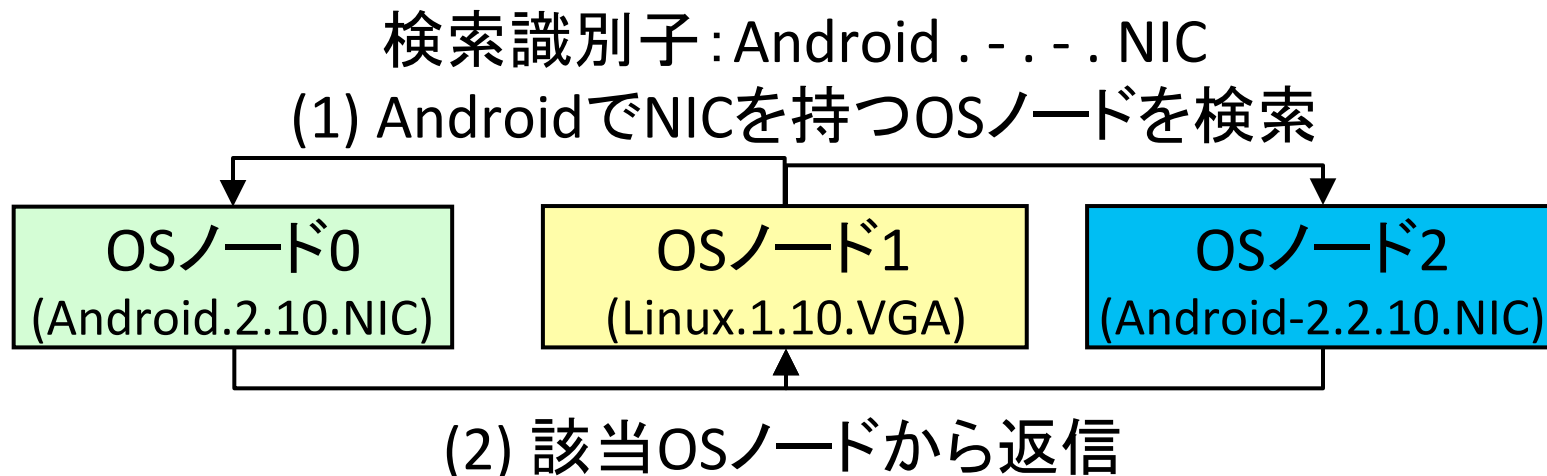
順不同の可変数の項目

例. Andoroid.2.10.NIC.VGA

# OSノードの検索

資源の有無と資源量によるOSノードの検索を提供

- (1) 検索したいOSノードの識別子を指定し、ブロードキャスト通信  
この際、検索条件にしない項目には“-”を指定
- (2) 受け取ったOSノードは自身の識別子が該当する場合は返信



# おわりに

Mintにおける資源移譲の問題点を明確化

➡ (課題1)と(課題2)を実現

(課題1) 他占有資源の保護機構の実現

(1) デバイス隠蔽による他占有デバイスの保護を実現

(課題2) Mintにおける資源管理機構を提案

(1) sysfsを利用した資源管理インタフェースの提案

(2) Bonjourを参考にした資源管理部の提案

＜残された課題＞

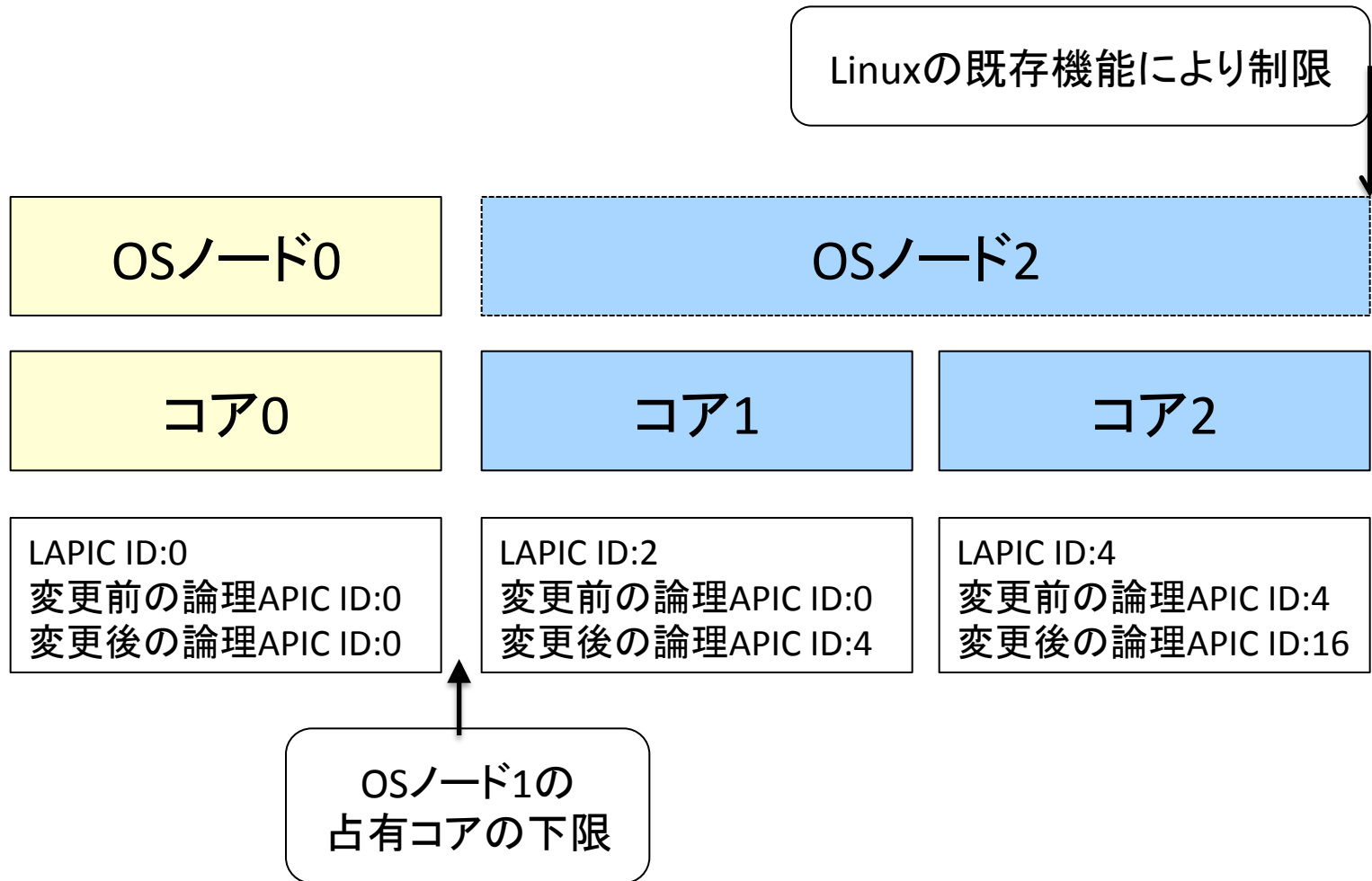
Mintにおける資源管理機構の詳細な設計, 実装, および評価



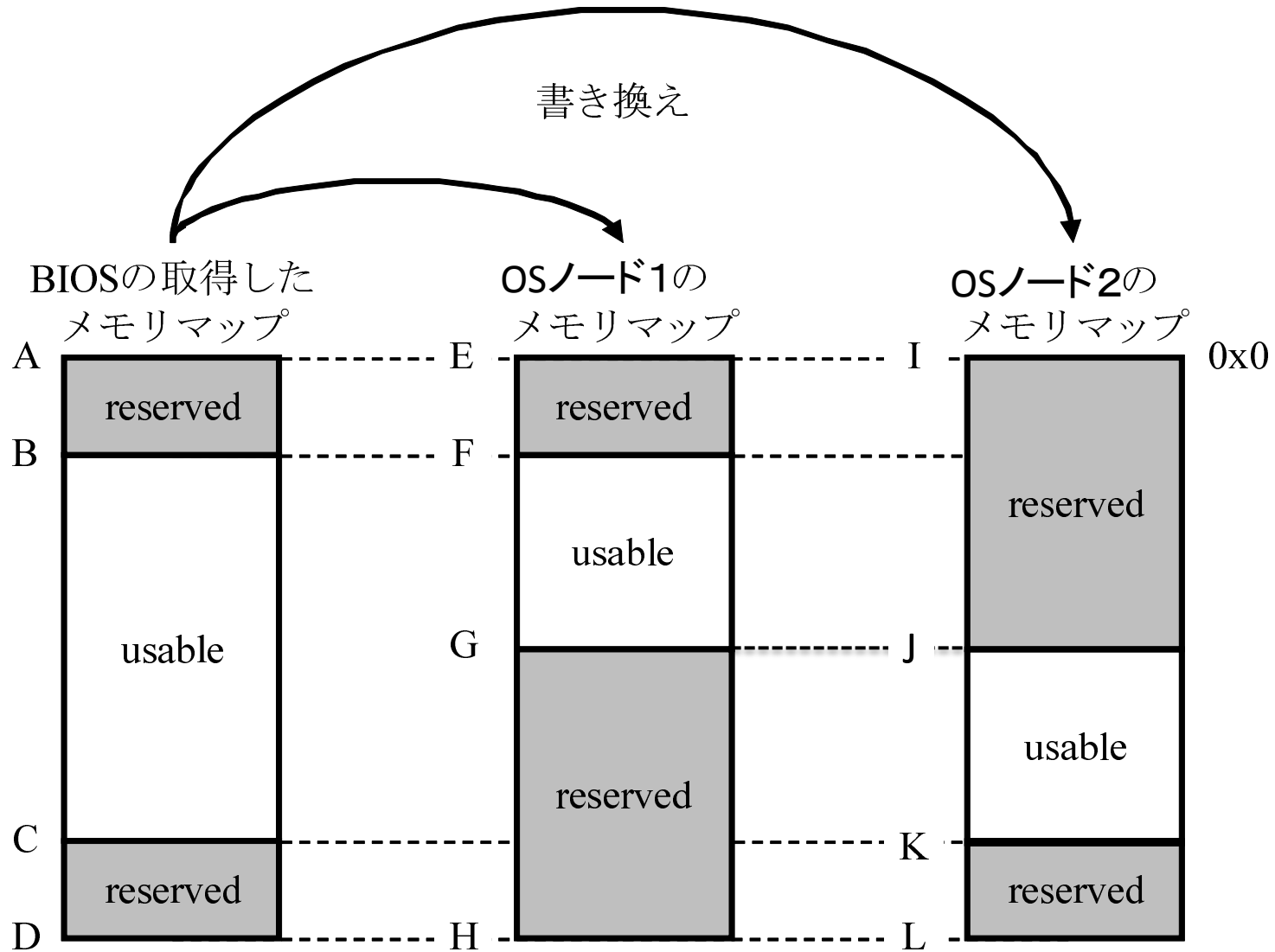


予備スライド

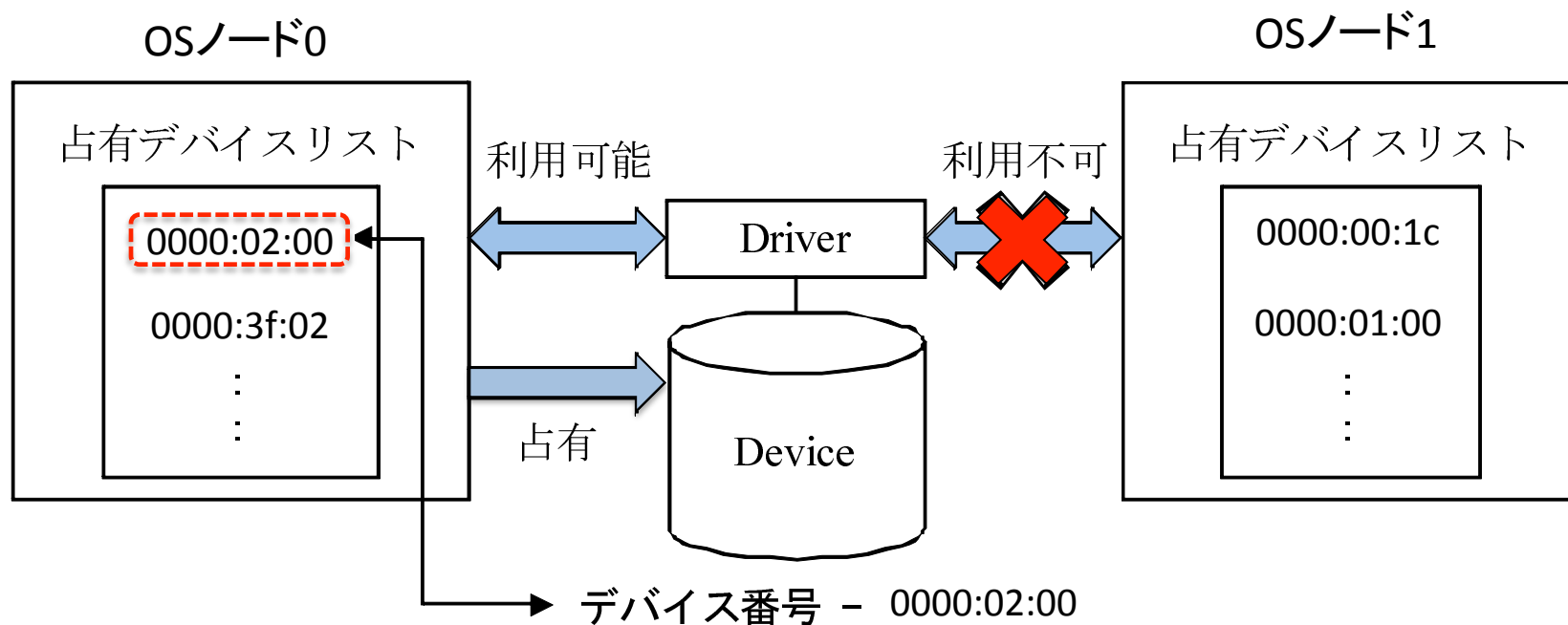
# コア分割



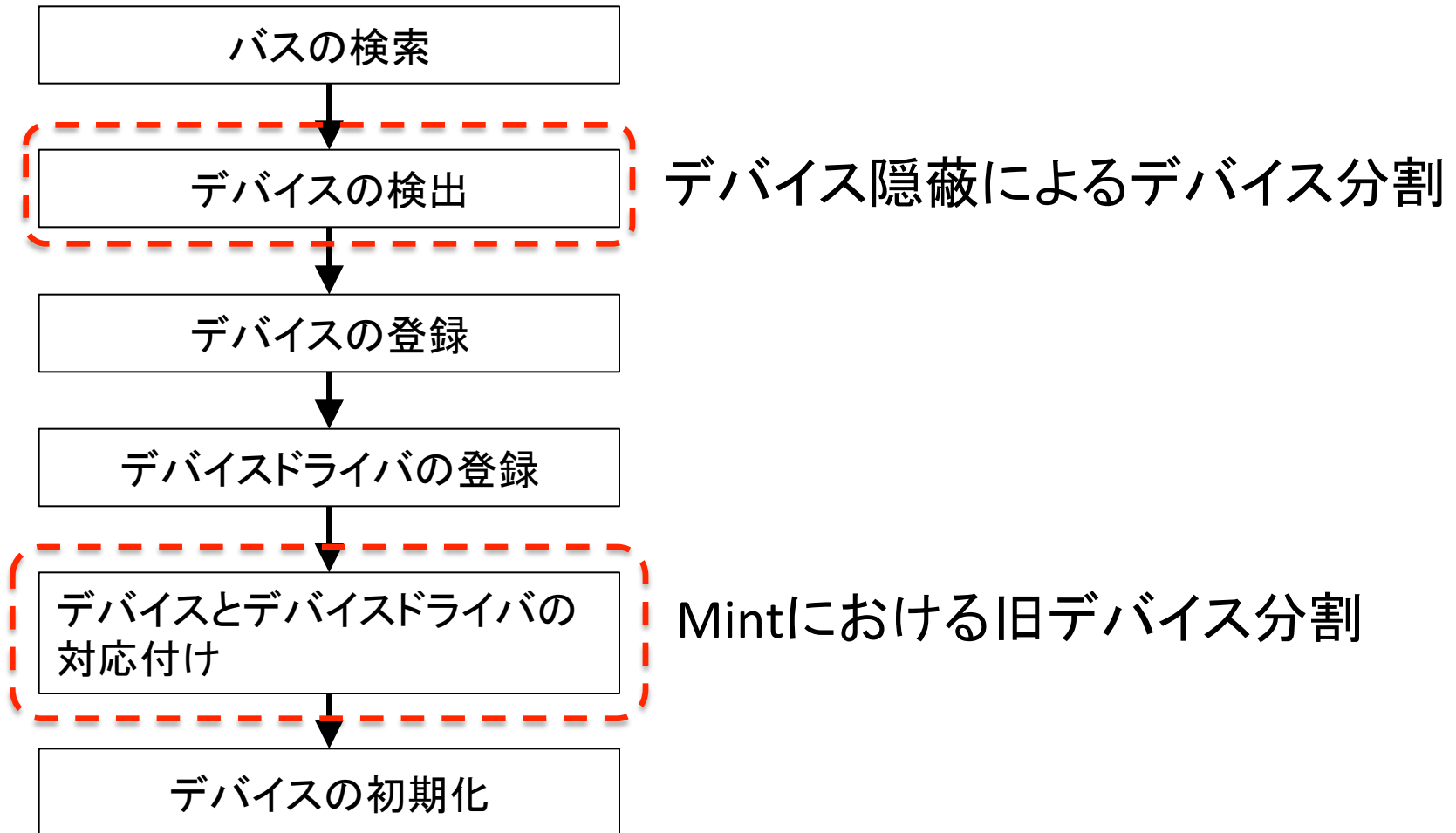
# メモリ分割



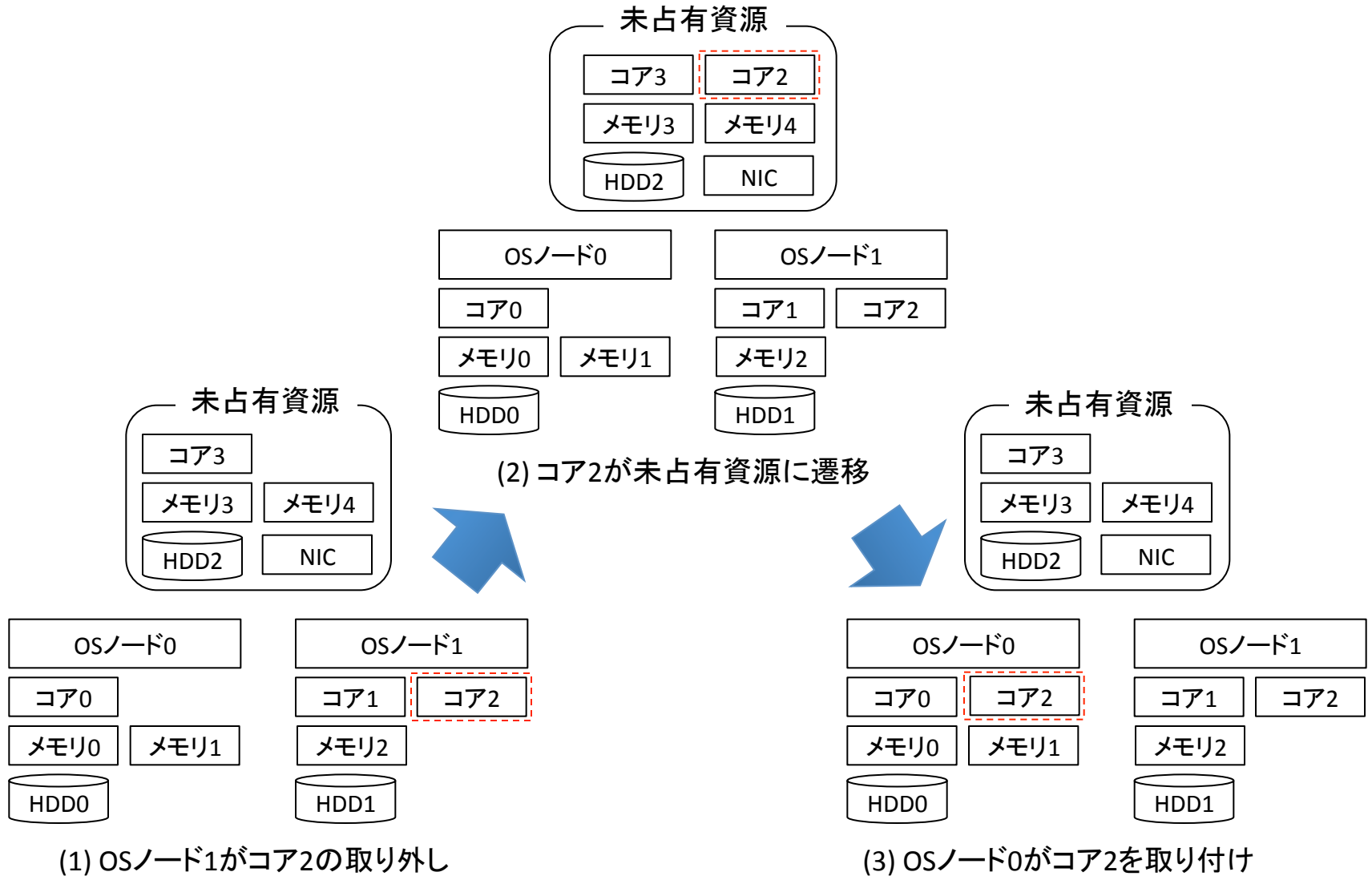
# デバイス分割



# Linuxによるデバイス利用の流れ



# 資源移譲モデル



# 各資源の保護方式

(1) コア, メモリ: 方式1を採用

(理由) 物理的なホットプラグ機能が未実装

➡ 動的な資源の隠蔽と再検出には大幅な改変が必要

(2) デバイス: 方式2を採用

(理由1) 既存の資源管理インタフェースの利用頻度が高い

(理由2) 物理的なホットプラグに対応(USBデバイス等)

➡ 資源の隠蔽と再検出が比較的少ない改変量で可能

(方式1) 各OSノードは全ての資源を認識し, 他占有資源の占有のみを防ぐ方式

(方式2) OSノードから自占有資源以外を完全に隠蔽する方式

# PCIデバイスの検出

(1) バス上の全てのスロットのPCIコンフィグレーション空間を確認し、ベンダIDとデバイスIDの読み込み

(2) ベンダIDとデバイスIDをもとにデバイスの検出と登録  
何も刺さっていないスロットはどちらも0xffff

(A) デバイスを検出: デバイスの登録

(B) ブリッジを検出: ブリッジを登録

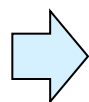
(3) ブリッジを発見した場合, ブリッジ以下のバスに対して(1)の処理

(2) の時点でデバイスを隠蔽することでデバイスの登録を防げる



# 動的なデバイス隠蔽と再検出

資源移譲にはデバイスの認識が必要



隠蔽したデバイスを動的に認識する機能が必要

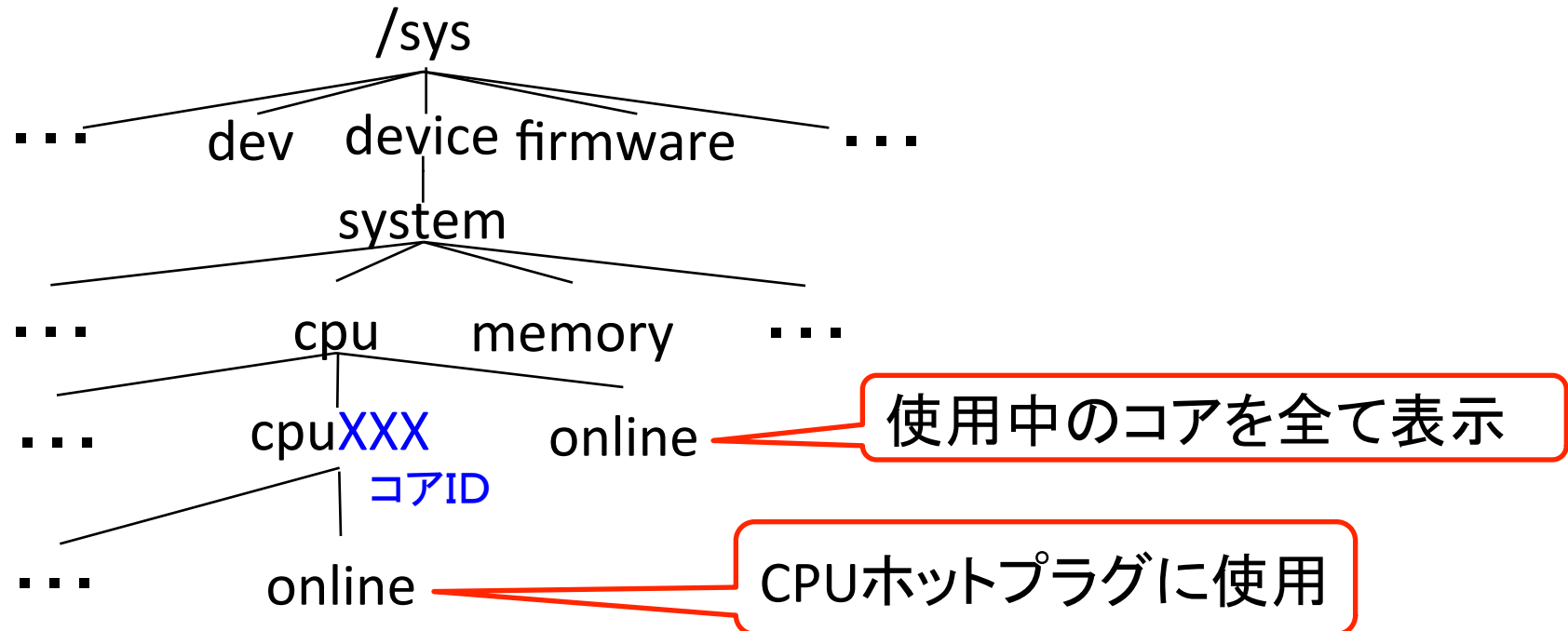
(対処) 隠蔽するデバイスを動的に変更し, Linux 既存機能により  
PCIバスを再スキャン

PCIバスの再スキャン機能:

- (1) PCIデバイスのホットプラグに対応
- (2) sysfsにより使用可能
- (3) PCIバスを再スキャン, デバイスの検出, デバイスの登録, および対応するデバイスドライバのロードを実行

# sysfs

- (1) Linux既存の仮想ファイルシステム
- (2) ディレクトリ構造で資源情報を管理
- (3) ファイルごとにアクセスされた際の処理を設定可能



# sysfsの処理

- (1) ファイルにアクセス(例 : cat, echo)
- (2) Readアクセスかwriteアクセスかでファイルごとに処理が分岐
- (3) 対応する処理を行う関数がカーネル内で実行

例 : CPUホットプラグ

/sys/device/system/cpu/cpuXXX/online  
コアID

Read

使用中 : 1  
未使用 : 0  
を表示

Write

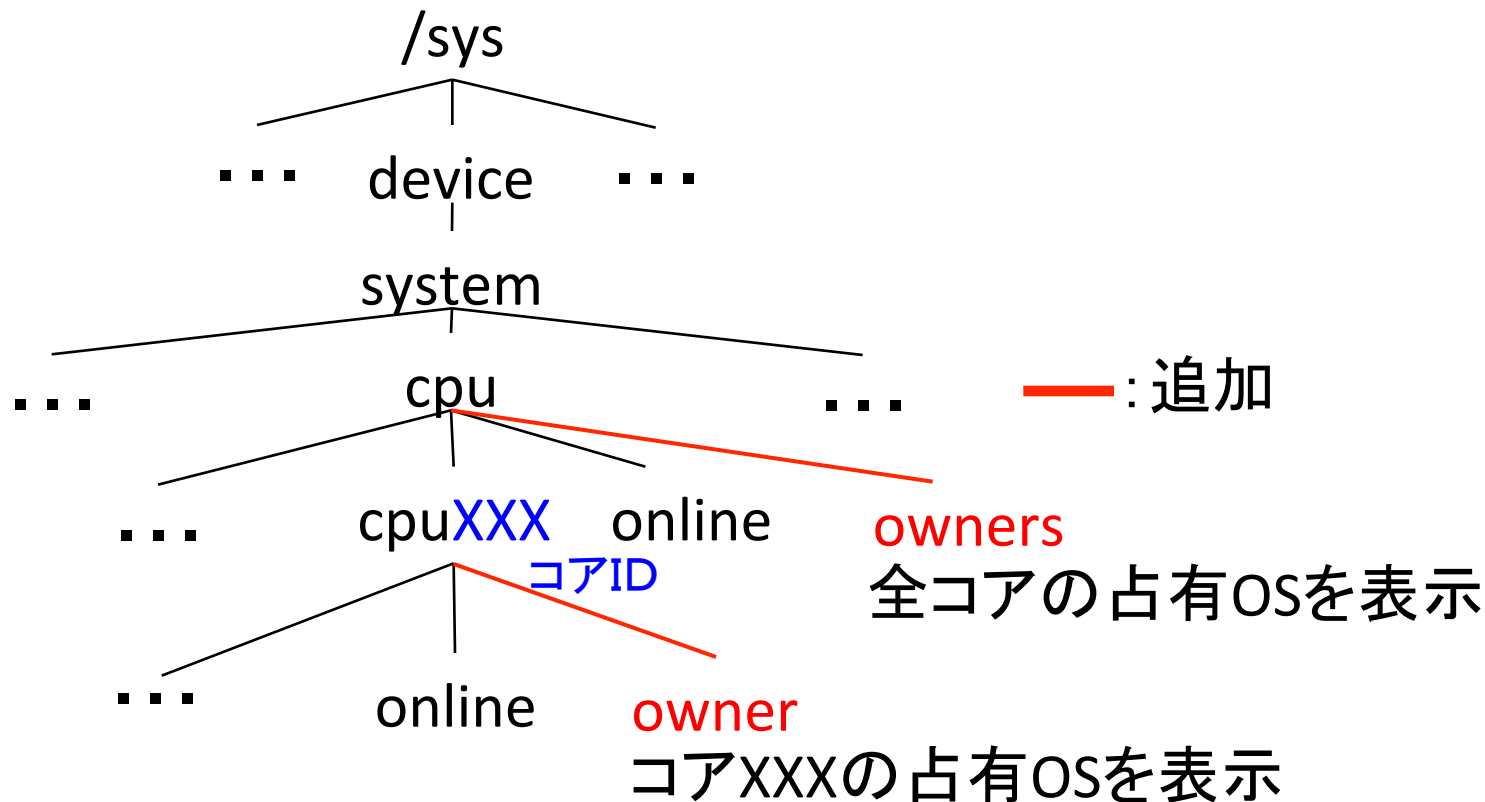
CPUホットプラグ用の関数に分岐  
0を書込 : コアの取り外し  
1を書込 : コアの取り付け

# sysfsを用いた資源管理インタフェース

sysfsにファイルを追加し，資源管理用の処理を割り当て

➡ Linux既存のインタフェースにより資源管理が可能

例：CPUの階層に占有OS表示用のファイルを設定



# Bonjour: Naming

各機器にホスト名を割り当てる機能

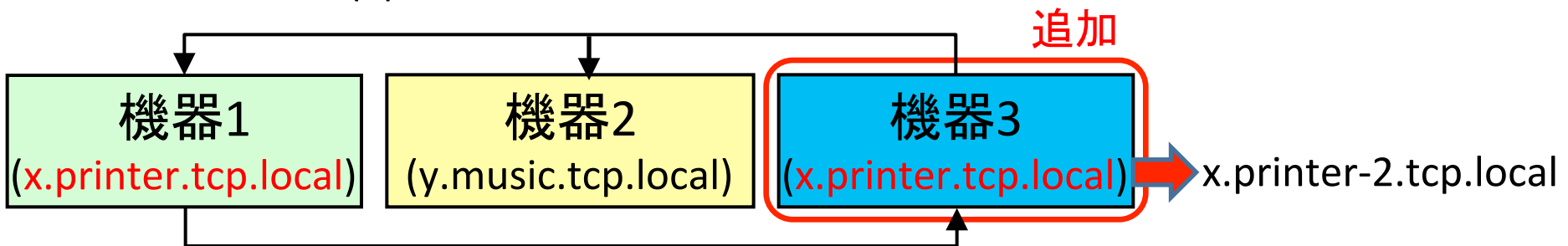
(1) 各機器は自身の情報を元にホスト名を設定

マシン名 + サービス名 + プロトコル名 + ドメイン名

例: x.printer.tcp.local

(2) 識別子の重複を防ぐために他の機器に通知

(1) 設定したホスト名を通知



(2) 同一ホスト名の機器が返信

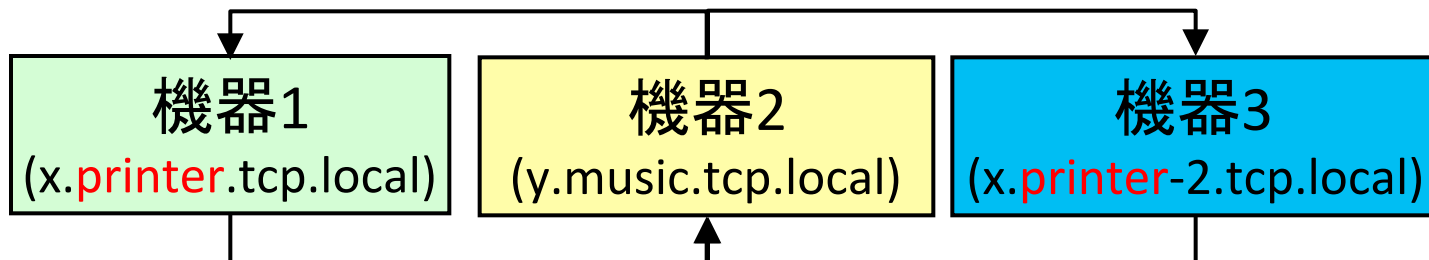
(3) ホスト名の変更と再通知

# Bonjour: Browsing

目的のサービスを提供する機器を検索する機能

- (1) サービスを指定してマルチキャスト通信
- (2) 自身に対応するサービスを持つ機器が応答

(1) サービス **printer** の検索通知



(2) 該当機器からの返信

# Bonjourのオーバヘッド削減技術

## キャッシング:

一度入手した機器情報をキャッシュし, 次回以降の検出に使用

## 重複応答の抑制:

- (1) サービス検索の際に, 既に知っているサービスの一覧も送信
- (2) 登録済みのサービスは検索要求を無視

## 指数バックオフとサービス通知:

- (1) 各機器は一定時間ごとにマルチキャストで提供可能なサービスの一覧を要求
- (2) マルチキャスト間隔を2のN乗(Nは回数)秒後に設定  
∴ ネットワーク機器は短時間接続と長時間接続に大別