B-037

マルチコアプロセッサにおいて複数の Linux カーネルを走行させる方式の設計

A Method for Running Multiple Linux Kernels on Multicore Processor

千崎 良太† Ryota Senzaki　乃村 能成† Yoshinari Nomura　谷口 秀夫† Hideo Taniguchi

1．はじめに

マルチコアプロセッサを持つ 1 台の計算機上で複数 OS を動作させる研究が活発である．先行研究として，SHIMOS[1]，[2]がある．

ここでは，同様に，1 台の計算機上で複数の Linux を独立に走行させる方式について述べる．本方式の特徴は，各 OS が独自に終了と再起動を行えることである．このため，各 OS は実計算機に近い走行環境を構築できる．

2．設計目標

設計目標を以下に述べる．

(1) 1台の計算機上で2つ以上のLinuxを動作させる

(2) 各OSは，複数あるコアのうち1つ以上を占有し，他OSから独立して動作する

(3) 入出力機器は，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する

提案方式は，TwinOS[3]と同様に，各OS間の影響が最小になるようにそれぞれが独立し，各OSが単独で動作する場合に近い環境と性能で走行できることを目指す．

図1は，3つのOSを走行させた場合の一例である．プロセッサについては，マルチコアプロセッサを用い，コアを分割し各OSに分配する．この例では，OS1にコア1とコア2を，OS2にコア3を，OS3にコア4を占有させている．実メモリについては，起動時に領域を分割し，共有部分を有さない．入出力機器については，OS毎に指定された入出力機器のみを占有制御させる．この例では，OS1にHDD1を，OS2にHDD2を，OS3にHDD3を占有させている．

3．課題

提案方式の実現における課題を以下に示す．

(課題1) メモリの分割と占有

走行させるOSの数だけ実メモリを空間分割する．メモリを分割占有するために，各カーネルが使用する物理メモリ範囲を制限できなければならない．占有するメモリの終端アドレスは，Linuxの既存機能によりブートパラメータで指定可能である．このため，本課題は，占有するメモリの先頭アドレスを指定可能にすることである．

(課題2) 各OSの起動

1番目のOS起動後に，残りのOSを順次起動することとする．この際，1番目のOSは，通常のLinuxに近い初期化処理で起動可能であるのに対して，2番目以降のOSは，ハードウェア制御を制限された状態で起動しなければならない．本課題は，2番目以降のOSが受ける制限を明らかにし，この制限に合わせてカーネル起動処理を変更することである．

†岡山大学 大学院 自然科学研究科 Graduate School of Natural Science and Technology, Okayama University

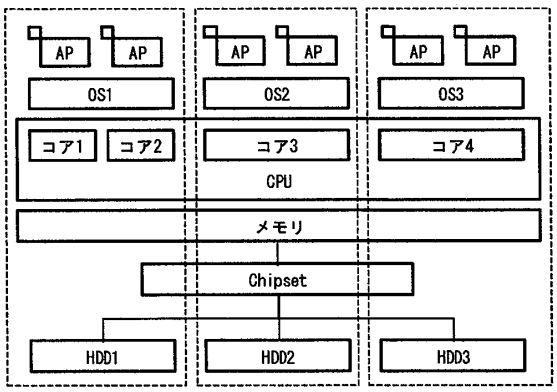

図1　提案方式の概要

(課題3) コアの分割と占有

複数あるコアを分配して，各OSに占有させる．利用コア数は，Linuxの既存機能により制限できるものの，常に先頭から何コアという指定しかできない．一方，各カーネルに異なるコアを割当てるには，何番目のコアから何個占有させるかを指定できる必要がある．

また，Linuxは，IPI(Inter-Processor Interrupt)によるコア間通信機能を利用する．しかし，各OSにコアを分割占有させるため，そのままIPIを利用させると，他カーネルに対して不要なIPIを送信してしまう恐れがある．このため，IPIに対する制限が必要である．

(課題4) 入出力機器の分割と占有

入出力機器をデバイス単位で分割し，各OSに占有させる．通常，Linuxカーネルは利用可能なデバイスをすべて占有しようとするため，各カーネルにおいて，占有するデバイスと占有しないデバイスを仕分ける必要がある．

(課題5) 割り込みの制御

外部割り込みを制御するI/O APICと一部の割り込みは，各OS間で共有する必要がある．Linuxは起動時にI/O APICの初期化と割り込みの設定を行う．この初期化処理により，自身が所有するコアに対して割り込みが通知されるよう設定され，利用しない割り込みはマスクされる．このため，最後に起動したカーネルが割り込みを設定し，先に起動しているカーネルの設定を上書きしてしまう．したがって，この初期化と設定の処理を各OS間で調整する必要がある．

(課題6) 終了処理と再起動処理

複数のOSが走行している場合，他のOSに影響を与えないように終了処理を行わなければならない．また，再起動処理については，終了処理を行った後に，独自にカーネルを展開し起動処理を行わなければならない．

4. 設計

4.1 メモリの分割と占有

BIOSが取得したメモリマップをOSが占有するメモリ領域のみが利用可能な領域であるよう書き換え，メモリを分割する．

4.2 各OSの起動

1番目に起動させるOSは通常の起動プロセスと同様にブートローダから起動させる．2番目以降に起動させるOSについては，1番目のOSから他のOSを1つずつ起動させる．

まず，1番目のOSにより，次に起動するOSのカーネルイメージをメモリ上に展開する．メモリ上に展開したカーネルイメージは，1番目のOSと違い，起動直後のコアの状態を初期化するコードを持たない．このため，次に，1番目のOSから起動するOSのBSP(Boot Strap Processor)にコア初期化処理を行わせる．この後，カーネルのセットアップコードを実行させる．

4.3 コアの分割と占有

x86 SMPの各コアは，Local APICのID(LAPIC ID)により識別できる．LAPIC IDは，BIOSにより，計算機の起動時に各コアのLAPICに設定され，計算機内のコアを一意に示す．このため，未使用のコアの内，LAPIC IDが最小のものをBSPとし，LAPIC IDの小さいコアから順番に利用する．そして，自身のBSPよりも小さいLAPIC IDを持つコアを利用しないことにより，各OSが占有するコアを指定可能である．

また，各コアのLAPICは，論理APIC IDを持つ．これは，IPIやI/O APICからの割り込み発生時，割り込み先のコアを示す値として用いられる．したがって，論理APIC IDは，各カーネル間で重複してはならない．このため，カーネルの論理APIC IDの決定規則を操作し，重複しないように変更する．具体的には，論理APIC IDを決定する処理に対して，各カーネルのBSPの持つLAPIC IDの値をオフセットとして追加する．

IPIについて，IPIは，割り込み送信先の全称表現としてall(全コア)やall without self(自分以外の全コア)を持つ．カーネルがこれを利用してしまうと，他のカーネルに不要なIPIを送信してしまう．この問題は，Linuxのブートパラメータno_ipi_broadcastにより全称表現の使用を抑制することで解決する．

4.4 入出力機器の分割と占有

PCI 等のデバイスは，デバイスドライバを利用させないことで制限できる．Linuxカーネルでは，デバイスに一意のデバイス番号を与えて管理しているため，この番号を使用して占有するデバイスを選択する．このため，デバイスドライバを組み込むカーネルコードを変更し，デバイス番号によってデバイスドライバの利用の可否を決定する．

また，PS/2キーボードやマウス，VGA，シリアルポートなどのレガシーデバイスは，デバイスドライバの有無にかかわらず，認識される．このため，カーネルコードを変更し，認識の可否を操作することで，分割占有することができる．

4.5 割り込みの制御

割り込みの種類によっては，各カーネル間で割り込みを共有する必要がある．例えば，計算機内に1つしかなく，すべてのカーネルに必要となるグローバルタイマからの割り込みや1つの割り込み線を複数のデバイスで共有するPCIデバイスからの割り込みが考えられる．これらは，デバイス分割を行っている場合でも，割り込みを共有しなければならない．

I/O APIC：I/O APICに制御される割り込みは，I/O APICを操作することで，同じ割り込みを複数のコアに通知できる．割り込みの通知先について，Linuxカーネルは起動時にI/O APICの設定を一旦クリアし，自身の占有するデバイスに対して，自身のコアに割り込みを通知するよう設定する．このとき，先に起動したカーネルがI/O APICに設定した内容は，後に起動したカーネルに上書きされてしまう．このため，各カーネルについて，他カーネルのコアに関する設定内容を操作しないようにする．

さらに，I/O APICの各割り込みには，対応するベクタ番号がある．ベクタ番号は，各割り込みに対して1つだけ設定できるため，割り込みを共有するカーネル間で同じ値を用いる必要がある．このため，提案方式の各カーネルは，すでにベクタ番号が設定されている割り込みに対しては，I/O APICからベクタ番号の設定を読み出し，同じベクタ番号を利用するように変更する．

MSI： Message Signaled Interruptを用いるデバイスは，デバイス単位で割り込みを設定できる．このため，特別な処理は必要ない．

4.6 終了処理と再起動処理

Linuxカーネルは，終了処理時にデバイスとLAPICを終了させ，I/O APICとタイマを無効化させる．I/O APICを無効化すると，他の走行するOSの割り込みの設定がクリアされるため，I/O APICの無効化処理を行わないようにする．代わりに，終了処理を行うOSに通知されている割り込みをI/O APICの設定から除外する．また，タイマの無効化を行うと，グローバルタイマと先に起動しているOSのローカルタイマの設定が破壊される．このため，タイマの無効化処理も行わないようにする．また，終了処理後に計算機をシャットダウンさせずに，占有しているコアをHALT状態にする．

再起動処理については，上記の終了処理後にカーネルイメージとinitrd，およびブートパラメータを指定のメモリ領域に配置し，配置したカーネルイメージの先頭から処理を始めることで，OS独自に再起動を可能にできる．

5. おわりに

1台の計算機上で複数のLinuxを独立に走行させる方式を設計した．残された課題として，提案方式の実装と性能の評価がある．

文　献

[1] T. Shimosawa, H. Matsuba, Y. Ishikawa, “Logical Partitioning without Architectural Supports,” Proc. of the 2008 Annual IEEE International Computer Software and Applications Conference, pp. 355-364, 2008.

[2] 下沢拓，藤田肇，石川裕，“マルチコアSHにおける複数カーネル実行機構の設計と実装，”情報処理学会研究報告 2008-OS-109, pp. 25-32, 2008.

[3] 田渕正樹，伊藤健一，乃村能成，谷口秀夫，“二つのLinuxを共存走行させる機能の設計と評価，”電子情報通信学会論文誌，vol. J88-D-I, no. 2, pp. 251-262, 2005.