B-036

Kexecによるコア別Linuxカーネルの起動方式
A Method for Booting Multiple Linux Kernels for Each Core Using Kexec

中原 大貴† 乃村 能成† 谷口 秀夫†
Daiki Nakahara Yoshinari Nomura Hideo Taniguchi

1. はじめに

我々は，プロセッサのコアごとに Linux カーネルを同時走行させる方式を研究開発している．実現における課題の1つに各カーネルの起動方式がある．我々は，Kexec を応用することで，複数のカーネルを順次別々のコアで起動させる方式を開発した．Kexec は，Linux に標準搭載されている機能であるため，本起動方式によって今後の Linux のバージョンアップへの追従が容易になる．

2. コアごとにカーネルを同時走行させる方式

コアごとにカーネルを同時走行させる方式[1]において，各コアは，事前に分割して与えられた物理メモリ領域を利用し，独立してカーネルを走行させる．これにより，仮想化によらずに複数の OS が同時に走行する．本方式におけるカーネル起動方式の流れについて以下で説明する．

(1) 利用する物理メモリ領域を互いに素になるように限定したカーネルイメージを必要数作成する．
(2) ブートローダを用いて1番目のカーネル(OS1)を起動する．その際，利用するコア数は制限しておく．
(3) 2番目以降のカーネル(OSi, i≧2)について
(A)リアルモードでアクセス可能な物理メモリ，つまり0x100000(1MB)番地以内の領域にコア初期化処理部，カーネルのリアルモードにおける初期化ルーチン(以降，セットアップルーチン)を配置する．
(B) OSi 用のカーネル本体と initrd を OSi に割り当てた物理メモリ領域に配置する．
(C) 起動対象(未使用)のコアに IPI(プロセッサ間割り込み)を送信し，(A)の領域から実行を開始させる．
(D) IPI を送信した OS1 は処理を完了する．IPI を受信したコアは，コア初期化処理部，セットアップルーチン，およびカーネル本体初期化処理を実行し，カーネルを起動する．

上記の起動方式において，問題点が2つある．1つ目は，今後の Linux のバージョンアップへの追従が困難であること(問題1)である．各カーネルが起動順序や自他の配置アドレスを意識した構造を内部に持ってしまうため，Linux 自身が持つ起動や再起動機能との整合性を維持することが難しいからである．2つ目は，カーネル構築時に利用メモリ領域を固定するため，カーネル間の物理メモリ配分をカーネルの再構築なしに変更できないこと(問題2)である．

3. Kexec を用いた後続カーネルの起動方式

3.1 目的

Kexec[2]を後続のカーネルの起動に利用することを考える．Kexec とは，Linux カーネルの高速な再起動方式である．通常の再起動では，BIOS，ブートローダ，セットアップルーチンを経てカーネル本体の初期化に至る．これに対して，Kexec は，これらの処理を自身で代替し，カーネル本体の初期化から再起動を開始する．また，Kexec は，指定した物理メモリ上に再起動用のカーネルイメージを配置する機能や，本来 BIOS やセットアップルーチンで行う初期化の結果をあらかじめ用意して再現する機能を持つ．これらの機能は，我々のカーネル起動方式と多くの類似点を持ち，(問題1)(問題2)を解決できる要件を持っているといえる．2章で述べた問題点に対処するため，Kexec を改変し，再起動処理の代わりに後続のカーネルの起動処理を行う．これにより，起動順序や自他の配置アドレスを意識した独自の起動機能を Linux カーネルに元々存在する機能で置き換えられるため，(問題1)に対処できる．また，各カーネルの利用メモリ領域を起動時に任意に指定できるため，(問題2)に対処できる．

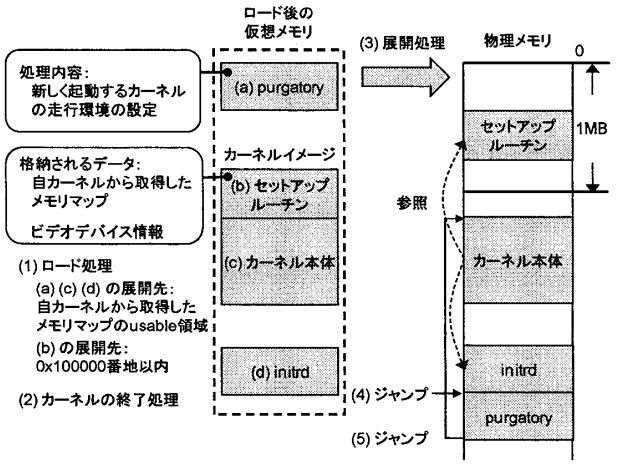

図1 Kexec の処理流れ

3.2 Kexec の処理流れ

まず，Kexec について，処理流れを**図1**に示し，説明する．
(1) **ロード処理**：Kexec は，まず，現在走行中のカーネルが保持するメモリマップ(物理メモリの利用可能領域情報)を取得して再起動用カーネルの展開先物理アドレスを決定する．次に，仮想空間上にメモリを予約し，再起動に必要な4つのデータを配置する: (a) purgatory (Kexec 固有の前処理), (b) セットアップルーチン，(c) カーネル本体，(d) initrd. これらは，(a) を除いて，Kexec によらない通常のカーネル起動(コールドブート)に利用されるデータと同一である．

Kexec は，高速化のため(b)を実行しない．その代わり，CPU の初期化等の省略できない処理を(a)に代替させる．また，(b)の実行で収集されるべきメモリマップやビデオデバイス情報などのハードウェア固有情報は，走行中のカーネル内部より取得し，(b)のデータ領域にあらかじめ再現して

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

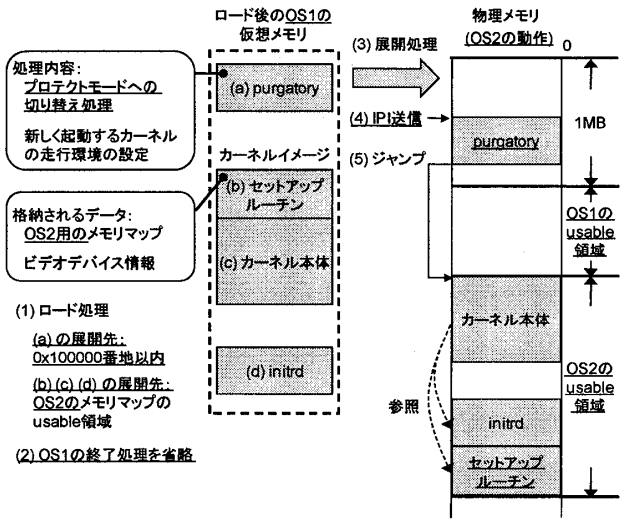

図2 Kexec を用いた後続カーネル起動方式の処理流れ

おく．この情報は(c)によって利用されるため，実行コードとして(b)は不要なものの，(c)へのデータ受渡し領域として必要とされる．

(2) **カーネルの終了処理**：再起動に備えて，割り込みの無効化や CPU の状態の初期化を行う．

(3) **物理メモリへの展開処理**：(1)で作成した仮想空間上に用意したデータを物理メモリ上に展開する．

(4) **purgatory へのジャンプ**：purgatory では，スタック領域，GDT，各種セグメントレジスタ，および各種汎用レジスタの設定を行う．つまり，(b)の部分的な実行といえる．ただし，(b)がリアルモードで走行開始するのに対して，purgatory は，プロテクトモードで動作する．

(5) **カーネル本体へのジャンプ**：カーネル本体が起動する．

3.3 課題

Kexec は，カーネルを再起動する機能であるため，後続のカーネルの起動用に改変する必要がある．改変には，以下の4つの課題がある．

(課題1) コアの起動

カーネルを再起動するのではなく，別のコアを起動し，起動したコアにカーネルを起動させる必要がある．

(課題2) プロテクトモードへの切り替え

起動直後のコアはリアルモードで走行する．一方，Kexec は，プロテクトモードのままで再起動を実現するため，リアルモードからプロテクトモードに切り替える処理を加える必要がある．

(課題3) 後続カーネル用のメモリマップの用意

Kexec は，図1-(1)のロード処理において，再起動のために必要なメモリマップを起動中のカーネルから取得する．しかし，後続カーネル起動のためには，メモリマップを起動中のカーネルから取得するのではなく，後続カーネルに合わせて外部から与えられるようにする必要がある．

(課題4) 先行するカーネルの走行環境の保護

図1-(2)のカーネルの終了処理は，そのまま実行すると先行するカーネルの走行環境を破壊する．これらの処理を省略する必要がある．

3.4 対処

3.3節で示した(課題1)から(課題4)を解決するため，それぞれ(対処1)から(対処4)を示す．Kexec を用いた後続カーネル起動方式の処理流れを**図2**に示す．課題への対処のために Kexec に対して行った改変を図2中下線部に示す．

(対処1) IPI の送信によるコアの起動

カーネルを物理メモリ上に展開した後，purgatory の先頭アドレスにジャンプするのではなく，起動対象のコアに IPI を送信するように改変した．IPI を用いて purgatory の先頭アドレスの情報を送信することで，IPI を受け取ったコアは，purgatory の先頭アドレスから処理を実行する．起動直後のコアはリアルモードで走行するので，purgatory は，物理メモリの0x100000番地以内に存在しなければならない．このため，purgatory の配置場所を0x100000番地以内に固定した．また，IPI を送信したコアは，Kexec の処理を終了させるように改変した．

(対処2) プロテクトモードへの切り替え処理を追加

リアルモードからプロテクトモードに切り替えを行う処理を purgatory の先頭に追加した．

(対処3) メモリマップの改変

メモリマップの取得に関して，自カーネルの情報から取得する代わりに，外部から与えられた値を使用するように改変した．また，purgatory 以外のデータをここで作成したメモリマップの usable 領域に展開するように改変した．元来，セットアップルーチンはリアルモードで走行するため，物理メモリの0x100000番地以内に配置しなければならない．しかし，セットアップルーチンはデータの受け渡し以外の目的では使われないため，0x100000番地以上の usable 領域に配置するように改変した．

(対処4) 終了処理の省略

先行するカーネルの走行環境を保護するため，終了処理を行う関数の呼び出しを省略した．また，Local IRQ の無効化，各種セグメントレジスタの初期化，GDT の初期化，IDT の初期化，および各種汎用レジスタの初期化を行わないように省略した．

以上の対処により，Kexec を用いて後続カーネルを起動することができる．

また，(対処3)により，各カーネルの利用メモリ領域を起動時に任意に指定できる．このため，単一のカーネルイメージを用いて異なるパラメータを指定することで，複数のカーネルを起動し，それぞれ独立に走行させることができる．この際，先行するカーネルはブートローダから起動し，後続のカーネルは Kexec を用いて起動する．

4. おわりに

Kexec を用いて，コアごとに Linux カーネルを同時走行させる方式の後続カーネルを起動する方式について，課題と対処を述べた．

文　献

[1] 千崎良太, 乃村能成, 谷口秀夫, “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,” FIT2010掲載予定, 2010.

[2] Hariprasad Nellitheertha, “Reboot Linux faster using kexec,” IBM, 〈https://www.ibm.com/developerworks/linux/library/l-kexec.html〉