1C-5

デュアル OS「NINJA」における 2 つの OS の統合

安達俊光† 田渕正樹†† 伊藤健一†† 乃村能成† 谷口秀夫†

†岡山大学工学部 ††株式会社 NTT データ

1. はじめに

計算機の普及は目覚しく，さまざまなサービスに計算機を利用している．一方，計算機ハードウェアの性能向上により，1 台の計算機上にさまざまなサービスを集約したいという要求がある．そこで，我々は，1 台の計算機上で 2 つの OS を独立に走行させるデュアル OS 方式を提案している[1]．具体的には，さまざまな機能に特化した開発が進んでいる Linux を取上げ，2 つの Linux にデュアル OS 方式を適用した「NINJA」を開発している．

現在，NINJA は，Linux をベースに改造を施し，最初に起動する OS（以降，先行 OS と呼ぶ）と 2 番目に起動する OS（以降，共存 OS と呼ぶ）の 2 つの Linux カーネルを作成することで実現している．このため，改造効率が悪く，メンテナンス工数も少なくない．

ここでは，作業の効率化のため，NINJA における 2 つの OS の統合について報告する．

2. デュアル OS「NINJA」

デュアル OS 方式では，1 台の計算機上で 2 つの OS を独立に走行させるため，各ハードウェア資源を分割し，各 OS ごとに占有させている．本方式の構成を図 1 に示す．プロセッサについては，起動時には処理を順次行い，起動後は時分割する．メモリについては，老番と若番に分割し，各 OS に占有させる．入出力機器については，起動時に各 OS ごとに指定された入出力機器のみを占有制御させる．

OS 切替えは，タイマ割込みと走行中でない OS が占有している入出力機器からの割込みを契機として走行している OS の環境を保存し，切替え後に走行する OS の環境を復元することで実現している．

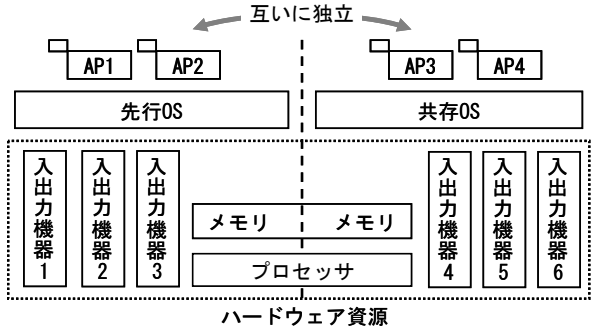

図 1 本方式の構成図

Unification of two OSs on a DualOS「NINJA」
†Toshimitsu Adachi, Yoshinari Nomura, Hideo Taniguchi
†Faculty of Engineering, Okayama University
††Masaki Tabuchi, Ken-ichi Itoh
††NTT DATA Co.

3. 統合

3.1 現状と問題点

現在，NINJA は， Linux をベースに改造を施し，先行 OS と共存 OS の 2 つの Linux カーネルを作成することで実現している．このため，改造効率が悪く，メンテナンス工数も少なくない．

そこで，2 つの環境を統合することを目指す．

NINJA における HDD の様子を図 2 に示す．BIOS が最初に読みに行く HDD は先行 OS 占有の HDD であるので，先行 OS 占有の HDD には，先行 OS のファイルシステムの他に，先行 OS と先行 OS 用ブートプラグラム，共存 OS と共存 OS 用ブートプラグラムがある．ここで，ブートプログラムとは，BIOS からのハードウェア情報の取得，CPU のプロテクトモードへの移行，そして圧縮されたカーネルの展開を行うプログラムである．また，先行 OS 占有の HDD には，共存 OS を起動させるために再起動処理を呼出す OS1load と先行 OS の監視下で共存 OS の起動処理を実行させるための処理を呼出す OS1boot の 2 つのコマンドもある．共存 OS 占有の HDD には，共存 OS のファイルシステムしかない．

NINJA における各 OS の起動の流れを図 3 に示し，以下に説明する．

(1)先行 OS 用ブートプログラムを読込み，実行
(2)先行 OS に制御が移り，起動処理を実行

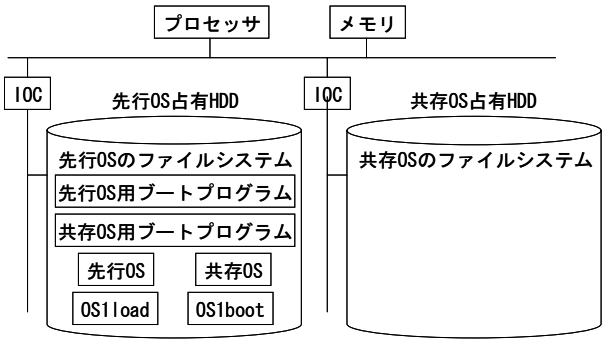

図 2 現状の HDD の様子

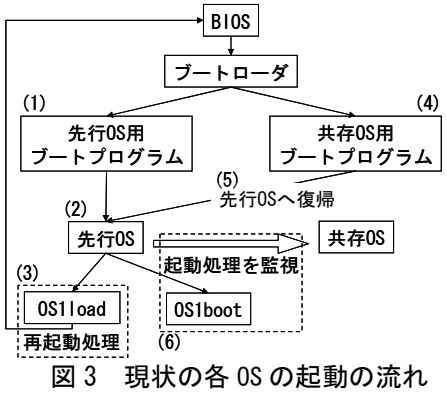

図 3 現状の各 OS の起動の流れ

(3)先行 OS の起動終了を契機に OS1load を実行し，再起動処理を実行
(4)共存 OS 用ブートプラグラムを読込み，実行
(5)先行 OS へ復帰
(6)OS1boot を実行し，先行 OS の監視下で共存 OS の起動処理を実行
上記の(6)において，先行 OS の監視下で共存 OS の起動処理を実行させるため，2 つの OS は仮想空間上でも分かれている．そのため，2 つの OS の統合は難しい．そこで，統合の第 1 段階として，OS1load と OS1boot の統合と各 OS 用のブートプログラムの統合を目指す．

3.2 共存 OS の起動コマンドの統合

共存 OS の起動コマンドである OS1load，OS1boot は，各機能の動作を確実に行うことを確認できるように別プログラム化していた．そこで，この 2 つのコマンドを統合する．統合は 2 つのコマンドがそれぞれ呼び出していた処理を順に呼び出すようにする．また，統合後のコマンドを OS1start とする．

3.3 ブートプログラムの統合

各 OS 用のブートプログラムは同一アドレスに読込まれ，実行中は他 OS に制御が移ることはない．したがって，統合は，先行 OS と共存 OS のどちらの OS として起動しているかを判断し，各 OS の処理に分岐させるようにする．

どちらの OS として起動しているかを判断する方法として，以下の 3 つが挙げられる．
(1)メモリの老番に先行 OS のコードがあるかどうかで判断する．
(2)OS1start でメモリに先行 OS が存在する値，あるいは共存 OS の起動を示す値を書込み，その値を用いて判断する．
(3)ブートローダにおいて起動する OS を選択時に，判断するためのパラメータを与え，そのパラメータを用いて判断する．
以下，それぞれについて，判断可能かどうか述べる．
(1)現在，NINJA が動作している計算機は常にウォームブートしているため，先行 OS を 1 度起動させると，メモリの初期化を行わない限り，メモリ上に先行 OS のコードが残っている状態となる．このため，以降の起動では必ず共存 OS が起動するため，この方法では判断できない．
(2)値を書込むアドレスは x86 プロセッサ特有のリアルモードでも使用できるように 1MB 以下のアドレスに限られる．しかし，1MB 以下のメモリは，共存 OS を読込む際に再起動を実行後，BIOS により初期化されるため，この方法では判断できない．
(3)通常，ブートローダは起動する OS を選択後，その OS に渡すパラメータをメモリに書込む．そこで，判断するためのパラメータを追加し，追加したパラメータの値の違いで判断することができる．
上記より，(3)の判断方法を 2 つの OS で処理が異なる箇所に実装する．2 つの OS で処理が異なる箇所は次のとおり．まず，先行 OS はメモリの老番を使用するようにし，共存 OS はメモリの若番を使用するようにしている．また，各 OS で占有制御する入出力機器の情報のみ BIOS から取得するようにしている．さらに，共存 OS はカーネルを展開した後，先行 OS に制御を戻している．

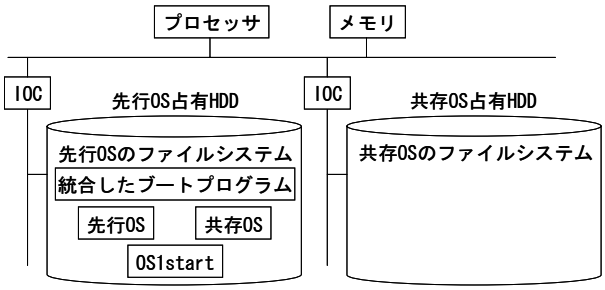

図 2　統合後の HDD の様子

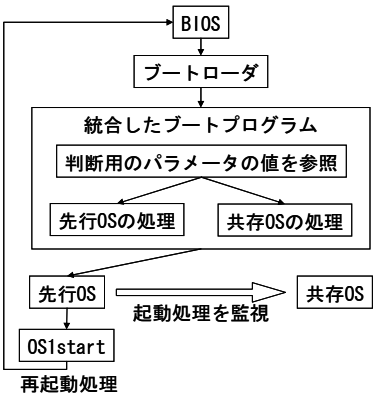

図 5　統合後の各 OS の起動の流れ

3.4 統合後の状態

統合後の NINJA における HDD の様子を図 4 に示す．OS1load と OS1boot を統合して，OS1start とした．また，各 OS 用のブートプログラムを統合した．

統合後の NINJA における各 OS の起動の流れを図 5 に示す．ブートプログラムは統合したため，2 つの OS で処理が異なる箇所で，ブートローダによって書込まれた判断するためのパラメータの値を参照して，どちらの OS として起動しているかを判断し，各 OS の処理に分岐させる．また，起動コマンドも統合したので，1 つのコマンドで共存 OS の起動を完了する．

4. おわりに

本稿では，デュアル OS「NINJA」における 2 つの OS の統合について述べた．ブートプログラムの統合において，どちらの OS として起動しているかを判断し，各 OS の処理に分岐させるようにした．

参考文献

[1]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 D(I)，掲載予定