システムソフトウェアと
オペレーティング・システム 91－7
（2002．8．21）

ハードウェアを非共有する複数オペレーティングシステムの構成法

谷口 秀夫† 乃村 能成† 田中 一男††
大塚 作一†† 井上 友二†††

パーソナルコンピュータ（以降，PC）に代表される計算機の普及は目覚しい．PC 上では，USB や IEEE1394 といった新しい入出力インタフェース (以降，I/F) が利用可能で，動作する OS にも UNIX や Windows といった様々な選択肢が存在する．一般に，利用可能な I/F やアプリケーション (以降，AP) は OS 毎に異なるので，ユーザは利用したい I/F や AP に対応した OS を選択する．このため，ユーザの利用形態によっては，複数の OS を必要とする状況が生じる．そこで，ひとつの計算機上に複数の OS を走行させる構成法として，各入出力機器をひとつの OS に占有制御させて各 OS の独自性を確保する方法を提案し，また，その適用システム例を述べる．提案法は，既存の複数 OS の構成法が抱える問題点を解消し，OS の変更は不要，ハードウェア入出力機器の機能と性能を十分に利用可能，および各 OS は独立に動作可能，の特徴を持つ．

MRM: A New Approach to the Construction of Multiple Real Machines in a Single System

Hideo TANIGUCHI,† Yoshinari NOMURA,† Kazuo TANAKA,††
Sakuichi OHTSUKA†† and Yuji INOUE†††

Personal computer (PC) is one of the fastest growing specialty in industry. Various kinds of I/O devices, application software packages (AP) and OSs are available under PC. Therefore, sometimes, we have a problem to find a good combination of devices, APs and an OS. To solve the problem, we need to run multiple OSs at the same time. In this paper, we describe a new technique to run multiple OSs in one box. We also illustrate some applications of this new technique. Our technique has three strong points – (1) no need to modify original OSs, (2) good I/O performance as bare OS, (3) no parent-child relationship among all OSs.

1. はじめに

パーソナルコンピュータ（以降，PC と略す）に代表される計算機の普及は目覚しく，様々なサービスで利用されている．また，プロセッサの性能向上だけではなく，USB や IEEE1394 のように新しい入出力インタフェース，さらに ATM および 100Mbps や 1Gbps のイーサネット型 LAN，無線 LAN といった通信インタフェースも登場している．

これらに伴い，サービスを提供する応用プログラム（以降，AP と略す）が走行する基盤ソフトウェア，特にオペレーティングシステム (以降，OS と略す) として，いくつかの OS が登場している．例えば，文書処理や電子メールのように個人が利用するサービスでは Windows が多く利用

† 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University
†† 株式会社 NTT データ技術開発本部
Research and Development Headquarters, NTT DATA CORPORATION
††† 株式会社 NTT データ技術開発本部
(現在，日本電信電話株式会社)
Research and Development Headquarters, NTT DATA CORPORATION
(Presently with NTT CORPORATION)

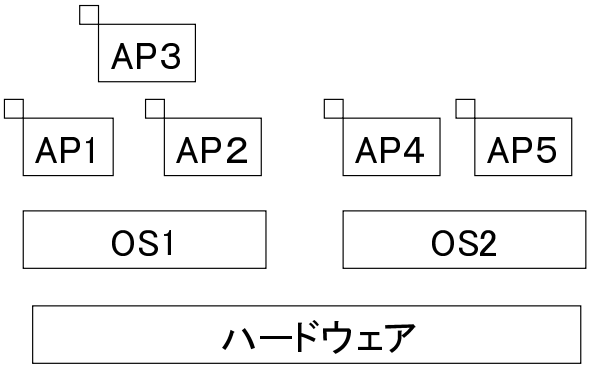

図 1　複数 OS 利用の様子

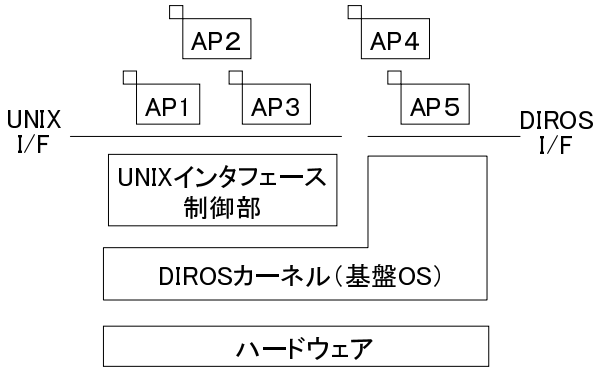

図 2　複数 OS インタフェースの実現例 (DIROS)

されており，Web サーバのようにトランザクション処理を行うサービスでは UNIX 系 OS が利用されている．つまり，サービスの特性に合わせた OS が利用されている．また，新たな入出力インタフェースや通信インタフェースを利用可能にするにはドライバの開発が必要なため，新しいインタフェースを利用できる OS は制限される．時には，OS1 は新インタフェース A を利用可能であるが新インタフェース B を利用できない，一方 OS2 は逆である，ことが起こる．さらに，新しいサービス機能を実現する場合，多くの OS で実現することは難しく，限られた OS になりやすい．したがって，複数の OS を利用できれば利便性が向上する．

以上のように，各 OS の特徴を生かした利用が望まれており，ひとつの計算機上で複数の OS を利用できれば，利便性の向上だけではなくサービス開発工数の削減も可能である．1 台の計算機で複数の OS を利用する様子を**図 1** に示す．OS1 上では AP1，AP2，および AP3 が走行し特徴的なサービスを提供し，OS2 上では AP4 と AP5 が走行し別の特徴的なサービスを提供している．

ここでは，上記の要求を満足することを目指し，ひとつの計算機上に複数の OS を走行させる構成法として，各入出力機器をひとつの OS に占有制御させて各 OS の独自性を確保する方法を提案する．また，その適用システム例を述べる．

2. 既存の複数 OS の構成法

ひとつの計算機上に複数の OS インタフェースあるいは複数の OS を実現する構成法について，既存の方法を説明する．

複数の OS インタフェースを実現した例として，DIROS[1] や Tender[2] がある．いずれの場合も基盤とする OS の中に別の OS インタフェースを実現している，DIROS の構成を**図 2** に示し，その特徴を以下に説明する．

（１）ハードウェアは，DIROS カーネル（基盤 OS）が全て制御している．

（２）UNIX インタフェースを実現する UNIX インタフェース制御部は，DIROS カーネルの機能を利用している．

（３）UNIX インタフェース制御部は，カーネル内に実現されることで，UNIX 系プロセス（AP1, AP2, AP3）にバイナリ互換の UNIX インタフェースを提供している．

この構成の場合，基盤 OS は独自性が高く，提供される別の OS は普及した OS のインタフェースであることが多く，以下の目的を持つ．

（A）ファイルの複写やバックアップなどの一般的なユーティリティプログラムについて，開発工数を削減する．つまり，既存のユーティリティプログラムを利用できるようにする．

（B）他の OS とのデータ互換を提供する．

したがって，上記の三つの特徴に加え，以下の特徴がある．

（４）UNIX インタフェース制御部が提供する機能は，UNIX 機能の一部，つまり上記の目的を達成する程度である．

次に，複数の OS を実現する方法として，仮想計算機を説明する．この様子を**図 3** に示し，その特徴を以下に説明する．

（１）ハードウェアは，VM モニタ（基盤 OS）が全て制御している．

（２）他の OS（OS1,OS2）は，バイナリ互換で走行できる．なお，バイナリ互換ではなく，

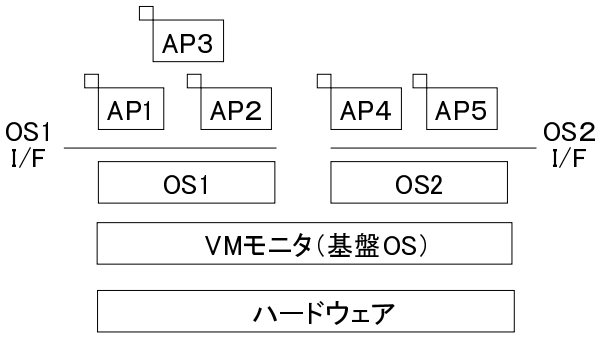

図 3　仮想計算機 (VM) による実現の様子

何らかの改変を必要とする場合もある．

（３）OS1 系プロセス（AP1,AP2,AP3）および OS2 系プロセス（AP4,AP5）は，バイナリ互換で走行できる．

この構成の場合，上記の特徴（２）を生かし，以下の目的を持つ．

（A）OS のデバグを支援できる．つまり，デバグしたい OS を VM モニタの上で動作させ，その OS の動作をトレースするなどして，デバグを効率的に進めることが可能になる．また，計算機ハードウェアが高価格な場合には，デバグ費用の抑制も図れる．

（B）各 OS の特徴を生かしたサービスを提供できる．

なお，VM モニタの部分としては，一般の OS を利用した VMware[3] や，性能の向上を図るために小さなカーネルを利用した Mach[4] や Lavender[5] や DARMA[6] がある．

このように，上記に示した既存の複数 OS の構成法は，ひとつのハードウェアを共有する形で構成されており，それぞれの目的を目指して構成されたものである．しかし，いずれの場合も，基盤 OS（主 OS）の上に他の OS（従 OS）が走行する，つまり複数の OS 間に主従関係があるため，以下のような問題がある．

（１）主 OS の停止は，計算機全体のシステムダウンとなる．

（２）従 OS の機能は，主 OS が提供する機能に制約される．

（３）従 OS の性能は，低い．

各問題について，以降に説明する．

ハードウェア全体を主 OS のみが制御しているため，主 OS の停止が計算機全体のシステムダウンとなることは避けられない問題である．

従 OS は主 OS 上で走行しているため，従 OS の機能は，主 OS が提供するハードウェア機能に制約されてしまう．したがって，様々なハードウェア機能，例えば種々の入出力インタフェース機能を持ったハードウェアであっても，主 OS がその機能を従 OS に提供できなければ，従 OS はそのハードウェアを利用できない．したがって，従 OS が計算機を占有して単独で走行する場合には利用できる入出力機能が，主 OS の上で従 OS が走行する場合には利用できないことが起こる．これを防ぐには，主 OS の機能強化が常に必要であり，その工数は大きい．

従 OS は主 OS 上で走行しているため，従 OS の性能は，従 OS が計算機を占有して単独に走行する場合に比べ低くなることは避けられない．これを回避するために，従 OS の走行を支援する処理を主 OS に組み込むことが VMware で実現されているが，ハードウェアを占有した場合と同等の性能を得ることは難しい．

3. ハードウェア非共有な複数 OS の構成法

3.1 方　　針

ひとつの計算機上において，ハードウェアを共有しないで複数の OS を走行させる方法は，以下の方針に基づいて構成する．

（１）OS の変更は一切伴わない．

（２）ハードウェアの入出力機能を十分に利用できる．

（３）各 OS はハードウェア性能を生かした性能を AP に提供できる．

（４）各 OS は独立に動作する．

上記の各方針について，以下に説明する．

各 OS の変更は一切行う必要はなく，再コンパイルつまり再システム生成することなく実行形式のままでの走行を可能にする．これにより，ソフトウェアライセンスに関連する価格や利便性の課題を排除できる．また，各 OS が有する独自性を大きく生かすことができる．例えば，利用者インタフェースに優れた OS と高速な通信機能を有する OS を同時走行させることにより，利用者は使いやすいインタフェースを使って高速な通信手段を得ることができる．また，高セキュリティな OS を利用することも容易である．さらに，機能拡充のための OS の変更によるソフトウェアの不

具合といった信頼性低下も防ぐことができる．したがって，世の中に存在する多種の OS を，その目的に合わせて，独自性を生かし，自由に組み合わせて利用することが可能になる．

ハードウェアの入出力機能について，古いものから新しいものまで十分に利用できる必要がある．ハードウェアの変遷は目覚しく，例えば，SCSI, USB，IEEE1394 といったように次から次に新しい接続インタフェースが登場している．このような環境では，ひとつの OS で全てのハードウェア入出力機能をサポートすることは難しい．また，古いながらも使いつづけたいハードウェア入出力機能も存在し，サポートしなくなってしまう OS ばかりでは困ってしまう．したがって，利用者が使いたいハードウェア入出力機能を自由に使えるためには，いろいろなハードウェア入出力機能を利用できることが望ましい．

ひとつの計算機上に複数 OS が走行することにより，各 OS の性能が低下することを防ぐ必要がある．入出力装置の性能はプロセッサ性能に比べ非常に低いため，AP は入出力装置の性能低下による影響を受けやすい．したがって，AP の利便性を確保するためには，入出力装置の性能低下を防ぐ必要がある．なお，ひとつの計算機上で複数の OS が走行する以上，複数の OS 間で共有する資源としてプロセッサがあるが，プロセッサは高性能化が著しく，共有による性能低下の影響は小さいと推察する．

各 OS の独自性を最大限に引き出すため，その動作を独立させる必要がある．つまり，ひとつの OS の開始から終了までは，他の OS の影響を受けないことが大切である．これにより，例えば，ひとつの OS がダウンしても他の OS は全く影響を受けず動作を継続することが可能になる．

3.2 複数実計算機

四つの方針に基づき，ひとつの計算機上に複数の OS を走行させ，各入出力機器をひとつの OS に占有制御させて各 OS の独自性を確保する方法を述べる．この方法は，ひとつの計算機を分割し，複数の実際の計算機として各 OS に見せる技術である．このため，各 OS に見せる計算機を複数実計算機（MRM: Multiple Real Machines）と名付け，仮想計算機（VM: Virtual Machine）と対応付ける．

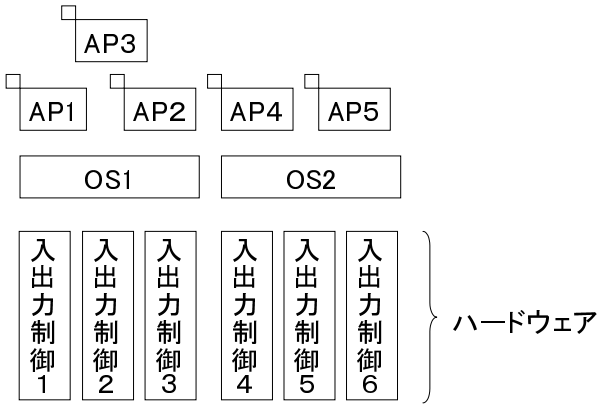

図 4　複数実計算機

複数実計算機（MRM）の様子を**図 4** に示す．入出力制御 1〜3 は OS1 により占有制御され，入出力制御 4〜6 は OS2 により占有制御されている．このように，ひとつの計算機のハードウェアを複数の OS で共有せず，ハードウェアを非共有する．これにより，先に述べた既存の複数 OS 構成法の問題点を解決し，ハードウェア非共有な複数 OS の構成法の方針を満足させる．すなわち，各 OS の変更は一切伴わず，ハードウェアの入出力機能を十分に利用でき，各 OS はハードウェア性能を生かした性能を AP に提供し，各 OS は独立に動作する．

3.3 課　　題

複数実計算機（MRM）を実現する際の課題を以下に述べる．

（１）共有必須ハードウェアの扱い
（２）立ち上げ方式
（３）終了方式

各課題について，以降に説明する．

ひとつの計算機を複数の OS で利用するため，プログラムの走行に必要なプロセッサとメモリは各 OS で共有する必要がある．メモリは，各 OS の利用範囲を限定することで，あたかも非共有のようにできる．ただし，ひとつの OS を除く他の OS では，メモリ開始位置が 0 番地でないような対処が求められる．一方，プロセッサを共有し分割して利用することを各 OS が意識しないで行うには，大きく二つの課題がある．ひとつは制御権の獲得と分配である．そもそも，ソフトウェアの実行はハードウェアからの割り込みを契機に始まる．つまり，制御権の獲得もハードウェアからの割り込みを利用する．例えば，周期的な割り込み

を起こすことができるタイマを利用することが考えられる．もうひとつは，例外や割り込みの扱いである．ここで，MRM では各入出力機器をひとつの OS に占有制御させるため，割り込みは当該の OS の割り込み処理で行うようにできる．一方，例外は走行しているプログラムの異常を通知するものであるから，各 OS への振り分けが必要である．

立ち上げを各 OS 自由に行わせるには，立ち上がる OS の処理化内容を抽出し，複数の OS が走行するための的確な対処が必要である．このため，大きく以下の対処が必要と考えられる．

（A）OS の初期化処理をトレースモードで実行し，特権命令は制御内容をシミュレートする．

（B）OS に対し，BIOS からの情報や実メモリ量として擬似情報を提供する．

（C）OS がカーネルモードからユーザモードに移行時に，OS の走行環境を構築する．

具体的には，例外や割り込み時の処理内容を変更する．

各 OS の終了は，ソフトウェアによる電源断処理を回避する必要がある．

以上のことから，MRM を実現するには，以下の対処が必要である．

（１）MRM を実現する部分（ここでは，MRM 制御部（MRMC: Multiple Real Machines Controller）と名付ける）が必要である．

（２）ひとつの OS を除く他の OS では，メモリ開始位置が 0 番地でないような対処が必要である．

3.4 実 現 方 式

課題から示したように，ひとつの計算機上に存在する OS を全て何らかの変更もなしに走行させるには，大きく二つの方法しかない．ひとつは新たに MRM 制御部を用意する方式であり，もうひとつは存在する OS のひとつに MRM 制御部的な機能を持たせる方式である．MRM 制御部を用意する方式は，多少なりとも，割り込み処理における処理オーバヘッドを生む．また，メモリ開始位置が 0 番地でないような対処は，多少なりとも OS の改造が必要である．そこで，我々は，存在する OS のひとつに MRM 制御部的な機能を持たせる方式を採用した．これにより，他の OS に対しては，

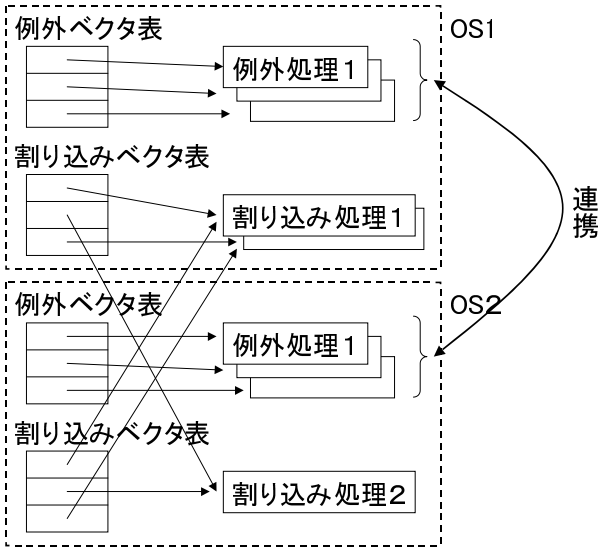

図 5　制御の流れ

（１）OS の変更は一切伴わない．

（２）ハードウェアの入出力機能を十分に利用できる．

（３）各 OS はハードウェア性能を生かした性能を AP に提供できる．

（４）各 OS は独立に動作する．

の方針を満足できる．二つの OS がひとつの計算機上で走行する場合について，例外や 割り込みの制御の流れを**図 5** に示す．各 OS は，例外ベクタと割り込みベクタを持つ．割り込みは，各 OS の割り込み処理がハードウェアから直接呼び出される．例外は，両 OS の連携により処理される．

4．適用システム例

本方式の適用システムとして，以下にいくつかの例を示す．

4.1 高セキュリティゲートウェイ

一方の OS については，ユーザが通常使用したい OS（OS1）を搭載し，もう一方の OS（OS2）でネットワークからの通信を監視し，外部のネットワークとの通信を監視する．例えば，外部からユーザの環境（OS1）に対してセキュリティの攻撃を行った場合，アタックを受けた OS1 の通信ログなどを改竄することが考えられる．しかし，本方式により OS1 と OS2 の通信ログが別々に管理されるようにできるため，アタックの痕跡をトレースすることができる．これにより，当該マシンへのアタックを抑止する効果が期待できる．

このシステムの様子を**図 6** に示す．AP2 は通

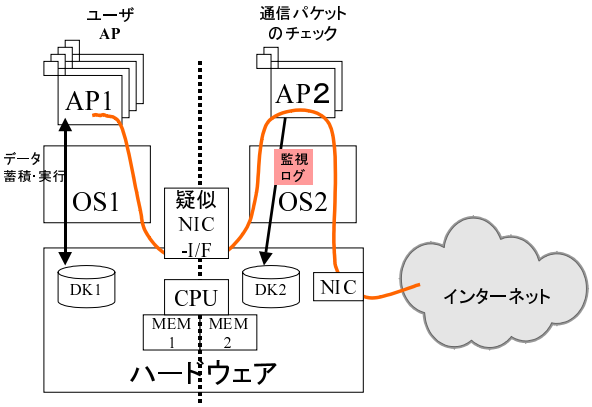

図 6　高セキュリティゲートウェイ

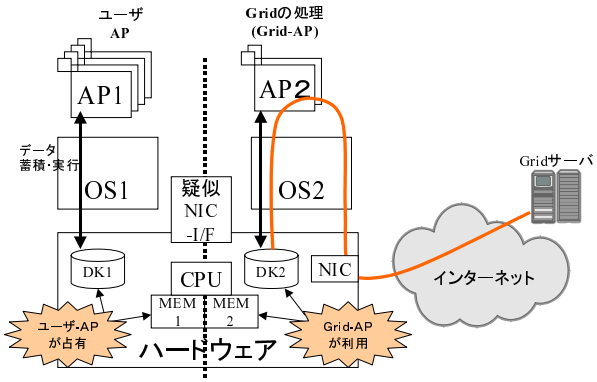

図 7　デスクトップ Grid 端末

信のパケットを外から見るだけで，内容に含まれる実行コードを処理することはないため，ウィルスに感染しない．また，DK2 に保管される監視ログは，悪意の第三者による改竄がなされない．したがって，アタックの痕跡をきちんと残すことにより，攻撃者の追跡が容易になる．

なお，OS2 は，ネットワークインタフェースコントローラ（NIC）を直接制御し，通信を行う．このため，OS2 は，OS1 の通信を支援するため，NIC の対ソフトウェア提供インタフェース（I/F）を疑似した疑似 NIC-I/F を実現し，OS1 に提供する．したがって，OS2 の通信は，OS1 を介して行うものとなる．

4.2　デスクトップ Grid 端末

ユーザが計算機を利用しない時，つまりプロセッサが遊休状態にある時に，大きな計算の一部を割り当てて計算を実現しようとするデスクトップ Grid 計算の環境において，ユーザ側から見ると，内容が不明な演算処理が自分の計算機環境に悪影響を及ぼす可能性に対する不安がある．そこで，本方式を利用することにより，自分の環境は OS1 の下で管理され，Grid 計算の処理環境は OS2 で実現することにより，Grid の演算処理がユーザ個人の環境に直接的な影響を及ぼすことを排除できる．

この様子を**図 7** に示す．Grid 計算で利用されるデータやプログラムは，全て OS2 の制御下のみで管理かつ実行され，OS1 の制御下にある DK1 や MEM1 という資源にはアクセスが許されない．したがって，Grid 計算のプログラムからの悪影響を排除できる．

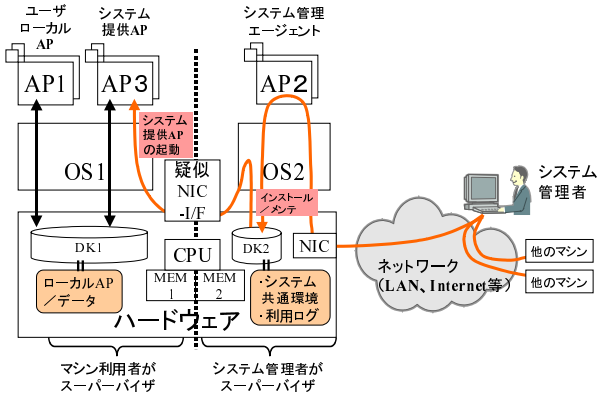

図 8　遠隔管理端末

4.3　遠隔管理端末

ユーザが管理できる範疇を OS1 に制限し，OS2 の管理はネットワークで接続されたシステム管理者が行うように設定する．これにより，オフィス環境などで利用する PC のデフォルト環境の構築はシステム管理者が OS2 の管理下に構築し，ユーザ個人の好みや状況に応じて利用する環境の構築は各ユーザが OS1 の管理下で構築できる．したがって，ユーザ個人の環境設定により，システム側で用意した環境が動作不具合になることを防止できる．

この様子を**図 8** に示す．OS2 はシステム管理者が管理し，システム側で設計した環境を用意する．ユーザは，システム提供環境を利用する場合には，OS2 のファイル情報を自分の環境（OS1）に読み込んで起動する．あるいは，OS2 に起動を依頼してクライアントサーバの形態で実行する．

4.4　高効率ネットサービス提供端末

遠隔管理端末と同様に，一方（OS1）の OS をユーザの管理下，もう一方（OS2）をネットワークサービスの管理下とし，例えば，サーバは封切

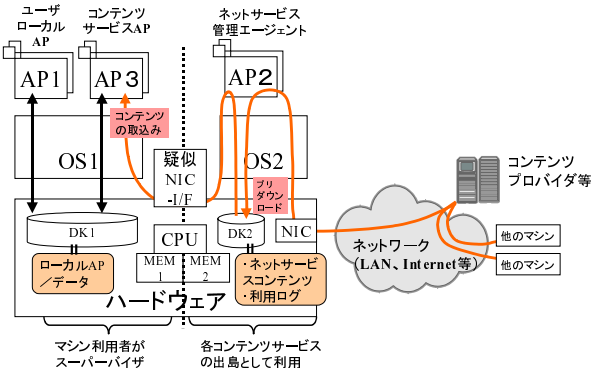

図 9　高効率ネットサービス提供端末

り前のコンテンツなどを事前に OS2 の管理下に配信し，封切り時刻に，OS1 側への提供を開始するサービスが考えられる．これにより，サーバは封切り時刻以前にスケジューリングした配信が可能になり，サーバへの集中が抑止できる．

この様子を**図 9** に示す．ネットサービスで利用するデータやプログラムは，OS2 の管理下に一旦配信され，そこからユーザの希望にしたがって提供される．これにより，コンテンツの事前配布，アクセスログの管理が行いやすくなる．また，プロバイダ側では，人気コンテンツへのダウンロードアクセスの集中を事前配布により平準化することができる．

4.5　高セキュリティWeb サービス提供サーバ

個人の環境のうち，外部からアクセスできるものと自分だけがアクセスできるものを区別し，それぞれの OS の環境下に構築する．例えば，ネットで一般に公開する Web コンテンツは OS2 に置き，作成途中のコンテンツや自分専用のコンテンツは OS1 に置く．これにより，公開したくない情報を誤って Web 上で公開してしまうことが起こりづらくなる．

この様子を**図 10** に示す．インターネットで公開するコンテンツは OS2 の管理下に保管され，作成途中のコンテンツやマシン利用者が個人的に使用するファイルは OS1 の管理下に保管する．これにより，Web サーバの設定ミスがあっても OS1 管理下の個人使用ファイルが公開されることが防げる．また，外部からの悪意の第三者による DoS 攻撃を受けても，OS1 側の環境はネットワーク機能を除いて動作することが可能である．

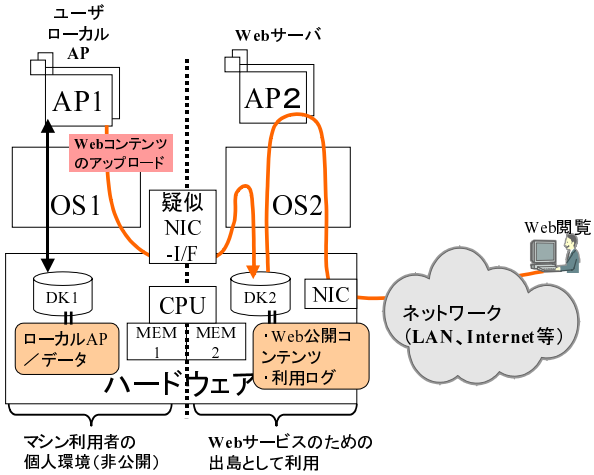

図 10　高セキュリティWeb サービス提供サーバ

5. ま と め

ひとつの計算機上に複数の OS を走行させる構成法として，各入出力機器をひとつの OS に占有制御させて各 OS の独自性を確保する方法を提案した．

提案法は，既存の複数 OS の構成法が抱える問題点を解消し，OS の変更は不要，ハードウェア入出力機器の機能と性能を十分に利用可能，および各 OS は独立に動作可能，の特徴を持つ．この方法は，ひとつの計算機を分割し，複数の実際の計算機として各 OS に見せる技術であり，各 OS に見せる計算機を複数実計算機（MRM: Multiple Real Machines）と名付けた．また，複数実計算機（MRM）を実現する際の課題として，共有必須ハードウェアの扱い，立ち上げ方式，および終了方式を述べた．さらに，その適用システム例として，高セキュリティゲートウェイ，デスクトップ Grid 端末，遠隔管理端末，高効率ネットサービス提供端末，および高セキュリティWeb サービス提供サーバを説明した．

今後は，課題に対する検討を進め，具体的なシステム開発を行う予定である．

最後に，本構成法の検討にご協力頂いた（株）NTT データ技術開発本部の伊藤健一氏に感謝します．

参 考 文 献

1) 箱守聰，谷口秀夫，“分散型リアルタイム OS: DIROS,” 情報処理，Vol.36，No.8，pp.751-

754, 1995.
2) 田端利宏，野口直樹，中島耕太，谷口秀夫, “Tender における異種 OS インタフェースの共存制御法,” 情処第 64 回全大論文集，pp.1-5,6, 2002.
3) J. Sugerman, G. Venkitachalam and B. Lim, “Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor,” Proc. of the 2001 USENIX Annual Technical Conference, 2001.
4) 徳田英幸, “Real-Time Mach: 実時間マイクロカーネル,” 情報処理，Vol.37，No.12, pp.1117-1124, 1996.
5) 外山正勝，毛利公一，大久保英嗣, “マイクロカーネル Lavender 上への UNIX サーバの構築,” 情処研報，Vol.2000, No.21, pp.25-30, 2000.
6) 齋藤雅彦，加藤直，大野洋，井上太郎，上脇正，中村智明, “2 つの OS 機能を共存させた組込みシステムにおけるタスク優先順位制御法,” 情処学論，Vol.43, No.6, pp.1940-1948, 2002.