2004－EVA－8　(12)
2004／3／19

デュアル OS「NINJA」の基本性能評価

田渕正樹†　　桝本圭†　　伊藤健一†
乃村能成‡　　谷口秀夫‡

計算機ハードウェアの性能向上により，複数の計算機環境で実現されてきた多様なサービスを，1 台の計算機上に集約したいという要求がある．しかしながら，1 台の計算機上に 1 つの OS が走行する従来の形態では，AP ソフトウェア移植や AP に合わせた OS の改良が必要となる．この問題を回避する方式の一つとして，1 台の計算機上に複数の OS が走行する環境を構築する仮想計算機方式がある．しかし，仮想計算機方式では，個々の OS の処理性能を十分に利用できていないという問題がある．そこで我々は，従来の仮想計算機方式の問題を解決するため，1 台の計算機上で 2 つの OS を走行させるデュアル OS 方式を提案し，2 つの Linux を対象にデュアル OS 方式を適用した「NINJA」を開発している．本稿では，NINJA の基本性能を評価し，特徴を明らかにする．

Evaluation of a DualOS「NINJA」

Masaki TABUCHI†　　Kei MASUMOTO†　　Ken-ichi ITOH†
Yoshinari NOMURA‡　　Hideo TANIGUCHI‡

In recent PC hardware's develop leads supplying of various services. These various services sometimes need to be performed in a PC, then OS has be customized in the existing way. Existing method like virtual machine, enables multiple OS's runs on a single PC and solves the problem. But, these method diminish I/O performance of each OS. We propose a DualOS method that two different types of OSes can run on a single PC for providing two different types of service or function. And we apply our method on two linuxes and name it「NINJA」.We evaluate NINJA's performance and mention its features.

1. はじめに

アプリケーション（以降,AP）とオペレーティングシステム（以降,OS）の組み合わせは多岐にわたり,様々なサービスが実現されている．一方，計算機ハードウェアの性能向上により，複数の計算機環境で実現されてきた多様なサービスを，1 台の計算機上に集約したいという要求がある．しかしながら，1 台の計算機上に 1 つの OS が走行する従来の環境でこの要求を満足するには，従来走行していた OS とは別の OS での走行を可能にするための AP ソフトウェア移植，あるいは AP が必要とする機能や性能に合わせた OS の改良が必要となる．この問題を回避する方式の一つとして，1 台の計算機上に複数の OS が走行する環境を構築する仮想計算機方式[1],[2],[3]がある．しかし，仮想計算機方式では，個別の OS 環境を実現するゲスト OS が全体を制御するホスト OS の上で動作するため，一方の OS の処理負荷が他方の OS の処理性能に影響を与えてしまう．また，ゲスト OS ではハードウェアの提供する入出力性能を十分に利用できないという問題がある．

そこで我々は，従来の仮想計算機方式の問題を解決するため，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，2 つの Linux を対象にデュアル OS 方式を適用した「NINJA」を開発している[4]．オープンソースである Linux は，改造が容易で，様々な機能に特

†株式会社 NTT データ
†Research and Development Headquarters, NTT DATA Co.
‡岡山大学工学部
‡Faculty of Engineering, Okayama University

化した OS として開発されている．このため，一口に Linux といっても様々な OS 環境が想定され，デュアル OS 環境における異なる OS といえる．

ここでは，NINJA の基本性能について評価し，特徴を明らかにする．

2. デュアル OS「NINJA」

2.1 開発方針

デュアル OS 方式は，仮想計算機方式が従来持つ問題を解決するために，以下の 2 点を設計の目標とする．

(1)各 OS は他 OS の処理負荷の影響を受けない

(2)両 OS とも入出力性能を十分利用できる

そのために本方式では，1 台の計算機の各ハードウェア資源を空間分割し，各 OS に占有させる．本方式の構成を図 1 に示す．メモリや入出力機器はどちらかの OS に占有される．ただし，ハードウェア機構上，空間分割が難しいプロセッサは各 OS で時分割する．したがって，デュアル OS の実現のためには，以下の方法を明らかにする必要がある．

(課題 1)各ハードウェア資源の分割占有方法

(課題 2)走行する OS の切り替え方法

また，各対処方法の実現においては，各 OS の改修を最小化するとともに，改修量が同等な場合は一つの OS に改修を集中させ局所化することとした．これにより，開発工数の削減だけではなく，Linux 以外の OS への適用工数の削減が可能になる．具体的には，2 つの OS において先に起動する OS を先行 OS と名付け，OS 共存のための課題への対処を中心的に行うこととし，2 番目に起動する OS を共存 OS と名付け，改修は最小限とした．以降，各課題に対する対処方法の概要について説明する．

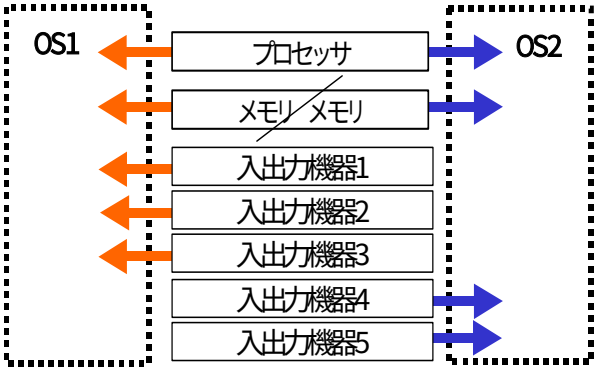

図 1 デュアル OS 方式の構成図

2.2 各ハードウェア資源の分割占有方法

各ハードウェアの資源の分割占有は，各 OS の起動時と入出力機器からの割込みに対して以下の対処を行うことで実現する．

(1)プロセッサ

起動時には起動処理を順次行うこととし，起動後はタイマ割込み，および(3)で述べる入出力機器からの割込みを契機としてプロセッサを占有する OS を切り替え，時分割する．

(2)実メモリ

共有部分を持たないよう起動時に上位と下位に 2 分割し，各 OS に占有させる．各 OS はそれぞれの物理メモリに仮想メモリ空間をマッピングする．

(3)入出力機器

起動時に各 OS 毎に指定された入出力機器のみを占有制御するように環境設定を行う．このため，当該割込みを発生した入出力機器を占有制御している OS の割込み処理が呼び出されるように制御する．したがって，割込み発生時に走行している OS（AP が走行している場合はその AP が利用している OS）が当該割込みを発生した入出力機器を占有制御している OS と異なる場合には，OS を切り替える．

なお，後から起動される共存 OS の起動時における OS 初期化処理により，先に起動している先行 OS の走行環境が壊されないように保護する必要がある．また，共存 OS の改修量を最小限とするために，できるだけ共存 OS は他の OS と共存していることを意識しなくても共存を可能とする工夫が必要となる．そこで，共存 OS の起動処理を先行 OS が監視の下で実行させ，先行 OS の環境を破壊しないよう保護する．具体的には，先行 OS はプロセッサをトレースモードに移行し，共存 OS のカーネル初期化処理を実行する．そして，1 命令毎に発生するトレース例外処理において，共存 OS の初期化処理内容を取得し，実行の許可/不許可を判断し，適切な処理を行う．トレースモードは，カーネル初期化の終了とみなせる init プロセスを生成する時点で解除する．

2.3 走行する OS の切り替え方法

走行する OS の切り替えは，割込みを契機にそれまで走行していた OS の環境を保存し，切り替え後に走行する OS の環境を復元（以降，OS 環境保存/復元処理）することで行う．保存/復元す

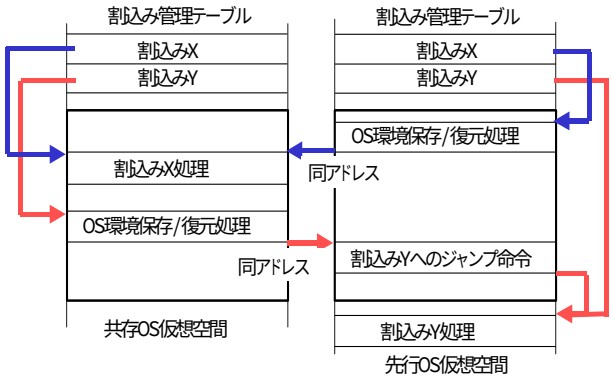

図 2 OS 環境保存/復元処理の仮想空間の様子

る情報を以下に述べる．

(1)各 OS が独立に持つ仮想空間情報

(2)割込み発生時にプロセッサが参照する割込み管理情報

(3)OS や AP の処理で用いられるプロセッサ上のレジスタ値情報

また，切り替え後に当該割込み処理を実行するために．図 2 に示すように各 OS の仮想空間上において，OS 環境保存/復元処理の終了位置と，切り替え先の OS の，当該割込み処理の開始位置を一致させる．これにより，他方の OS が占有するデバイスからの割込みが発生すると OS 環境保存/復元処理を実行し，他方の OS へ制御が移行した後，当該割込み処理を実行する．

3. 評価

3.1 評価の観点と測定環境

NINJA の基本性能を，共存 OS と同等の機能を持つ Linux(以下，単独 OS)の場合と比較することでデュアル OS 方式の特徴を明らかにする．具体的には，以下の項目を測定する．

(1)OS 初期化処理時間

(2)OS 環境保存/復元処理時間

(3)AP 処理内容と相互影響の関係

OS 初期化処理時間の測定により，各ハードウェア資源の分割占有処理の影響，およびトレースモードを利用した共存 OS の起動の遅延を明らかにする．また，OS 環境保存/復元処理の測定により，走行中の OS が占有しないハードウェア資源からの割込みへの応答時間を明らかにする．さらに，実 I/O 処理とプロセッサ処理の量を可変に混在させた評価用プログラムを用いて，AP の処理内容により他方の OS 上のプログラムに与える相互影響の関係を明らかにする．

測定環境を表 1 に示す．実装されている物理メモリ 256MB の 128MB を境として，上位を先行 OS が使用し，下位を共存 OS が使用する．時間測定はプロセッサのタイムスタンプ値を出力する rdtsc 命令を用いて行った．

表 1 測定環境

項目	内容
プロセッサ	Intel Pentium4 1.7GHz
実装メモリ	単独 OS 128MB NINJA 256MB
OS	Linux Kernel-2.4.7

表 2 OS 初期化処理時間

測定対象	処理時間（秒）
単独 OS(入出力機器を全て占有)	5.36
先行 OS(入出力機器を全て占有)	5.39
先行 OS(入出力機器を分割占有)	4.98
共存 OS(入出力機器を分割占有)	19.85

3.2 OS 初期化処理時間

OS 初期化処理時間として，ブートローダなどによりメモリ上に展開された OS イメージに制御が渡される直前から，カーネルモードからユーザモードへ移行する/sbin/init の処理開始までの時間を計測した．また，単独 OS は全入出力機器を占有するため，先行 OS についても全入出力機器を占有した場合の処理時間を計測した．共存 OS の処理時間は，先行 OS によるトレースモード移行後，メモリ上に展開された共存 OS に制御が渡される直前からの時間である．計測結果を表 2 に示す．

表 2 から，全入出力機器を占有する場合の先行 OS の処理時間は，同様に全入出力機器を占有する単独 OS の場合と同等である．また，入出力機器を分割占有する場合の先行 OS の処理時間は，単独 OS の場合よりも少し短くなる．先行 OS は初期化を行う必要のある占有入出力機器が少ないため，単独 OS よりも OS 初期化処理時間が短いといえる．一方，共存 OS の処理時間は非常に長い．これは，先行 OS がトレースモードで共存 OS のカーネル初期化処理を 1 命令ずつ監視するためである．しかし，この遅延はカーネル初期化処理時にのみ発生するものであり，起動後のシステムの性能や動作に影響を与えるものではない．

3.3 OS 環境保存/復元処理時間

NINJA において，OS の切り替えが不要な場合の割込み処理の流れは，単独 OS の場合と同様である．つまり，割込み発生により，プロセッサハードによるスタックへのレジスタ退避に続いて，OS が行うレジスタ退避などの割込み共通処理が実行され，その後，当該割込み処理が実行される．これに対し，OS の切り替えが必要な場合は，割込み発生により，ハードによるレジスタ操作の直後に OS 環境保存/復元処理が実行され，その後に割込み共通処理と当該割込み処理が実行される．

OS 環境保存/復元処理の開始から終了までの処理時間を表 3 に示す．表 3 より OS の切り替え処理時間は 2.33 マイクロ秒であることがわかる．また，単独 OS において，割込み発生から割込み共通処理開始までの処理時間は 0.5 マイクロ秒であった．したがって，OS の切り替えが不要な場合は，0.5 マイクロ秒で当該割込み処理が開始されるのに対し，OS の切り替えが必要な場合は，2.83 マイクロ秒で当該割込み処理が開始される．したがって，OS の切り替えが必要な場合は，2.33 マイクロ秒だけ割込み処理の開始が遅れる．

表 3 OS 環境保存/復元処理時間

処理	処理時間(マイクロ秒)
仮想空間情報切り替え	0.55
割込み管理情報切り替え	0.45
レジスタ値情報切り替え	1.33
計	2.33

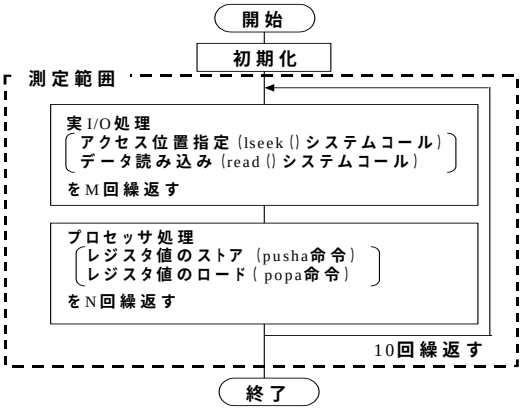

図 3 評価用プログラムの処理流れ

3.4 AP 処理内容と相互影響の関係

3.4.1 測定方法

測定は実 I/O 処理とプロセッサ処理を指定した比率で混在させる評価用プログラムを用いて以下の時間を計測した．

(1)単独 OS での評価用プログラム処理時間
(2)共存 OS での評価用プログラム処理時間(先行 OS での走行 AP なし)
(3)共存 OS での評価用プログラム処理時間(先行 OS でも評価用プログラムを実行)

上記(1)と(2)の比較により，NINJA の基本性能を明らかにする．また，(2)と(3)の比較により，NINJA における OS 間の相互影響を明らかにする．測定は共存 OS，先行 OS ともにシングルユーザモードで走行させた．各 OS が実 I/O 処理を行う磁気ディスク装置はそれぞれを別コントローラに接続し，プロセッサによるデータ転送(PIO)を行っている．また，OS の切り替えはタイマ割込み（周期:10ms）発生時と，他方の OS が走行中に自 OS が占有する磁気ディスク装置からの割込み発生時に行うよう制御した．

測定に用いた評価用プログラムの処理流れを図 3 に示す．評価用プログラムはシステムコール lseek()と read()を用いた磁気ディスク装置への実 I/O 処理（繰り返し回数：M 回）と，スタックへレジスタ値を出し入れするプロセッサ処理（繰り返し回数：N 回）を繰返し実行する．なお，ランダムな値を用いて lseek()を発行することにより，ランダムな位置からのデータ入力を行う．繰り返し回数ＭとＮを外部より与えることにより，実 I/O 処理とプロセッサ処理の比率を変えることが可能である．

3.4.2 基本性能

評価用プログラムの処理内容を，以下のそれぞれについて，「単独 OS」と「共存 OS(先行 OS での走行 AP なし)」での処理時間を比較する．

(1) プロセッサ処理のみの場合
(2) 実 I/O 処理のみの場合
(3) プロセッサ処理と実 I/O 処理が混在する場合

以下に、それぞれについて結果を考察する．

(1) プロセッサ処理のみ (M:N＝0:1)の場合

結果を表 4 に示す．共存 OS での処理時間は単独 OS の場合と比較して 2 倍の処理時間となる．これは，タイマ割込みによるプロセッサの時分割を行っているため，タイマ割込みの発生 2 回に 1 度の割合で先行 OS に制御が移り，次のタイ

マ割込みまで先行 OS にプロセッサが占有されることに起因する．

(2) 実 I/O 処理のみ (M:N＝1:0)の場合

実 I/O 処理については，1 回当たりの I/O のデータサイズを 512,1024,4096,8192 バイトについて測定した．処理時間を表 5 に示す．

共存 OS での処理時間は単独 OS の場合と比較して，I/O データサイズが小さい場合，遅延は少なく抑えられている．これは，I/O データサイズが大きくなるにつれ，実 I/O 処理内部におけるプロセッサ処理の割合が増大し，先行 OS から共存 OS に制御を切り戻す契機の多くが I/O 割込みではなくタイマ割込みにより発生することに起因する．

実 I/O 処理は，I/O を発行し磁気ディスク装置からデータを読込むデータ読込み処理と，読み込んだデータをメモリ上に転送するための転送処理にわけられる．データ読込み処理が終わった段階で I/O 割込みが発生する．今回の測定環境において，転送処理はプロセッサ処理である．

ディスク I/O における実 I/O 待ち時間は，多くにおいてタイマ割込みの周期である 10ms よりも小さいため，共存 OS 側でデータ読込み処理を行っていた際にタイマ割込みにより先行 OS に制御が移った場合，次のタイマ割込みを待たずしてデータ読込み完了による I/O 割込みが発生し，制御が共存 OS に戻る．このとき，共存 OS 側の処理時間の遅延は OS 切り替えのオーバーヘッド分のみとなる．

一方，共存 OS 側でデータ読込み処理以外のプロセッサ処理をしていて先行 OS に制御が移った場合には I/O 割込みが発生せず，次のタイマ割込みが発生するまでの間，先行 OS 側から制御が戻らない．つまり，共存 OS 側の処理はタイマ周期 1 回分と OS 切り替え 1 回分の間中断する．

したがって，データ読込み処理中に OS の切り替えが発生する頻度が多い場合，処理時間の遅延は少なく抑えられる．

また，今回の測定環境では，I/O データの転送をプロセッサが介在して行っているため，I/O データサイズが大きいほどプロセッサ処理の割合は多くなり，データ読込み処理中に OS の切り替えが発生する確率が小さくなる．表 6 に先行 OS に切り替わる際の共存 OS 側の処理がデータ読込み処理の割合を示す．

表 4 プロセッサ処理のみの場合の処理時間

測定対象	処理時間（秒）
単独 OS	8.01
共存 OS	16.06

表 5 実 I/O 処理のみの場合の処理時間

I/O データサイズ(バイト)	512	1024	4096	8192
単独 OS 処理時間(秒)	8.24	8.46	9.61	10.94
共存 OS 処理時間(秒)	8.46	8.91	11.07	13.50

表 6 先行 OS への切り替え時の処理状況

I/O データサイズ(バイト)	512	1024	4096	8192
データ読込み処理中(回)	775	786	822	888
転送処理中(回)	57	62	177	297
データ読込み処理の割合(%)	93.1	92.7	82.3	74.9

表 7 処理を混在させた場合の処理時間

実 I/O 処理:ﾌﾟﾛｾｯｻ処理	0:1	1:3	1:1	3:1	1:0
単独 OS 処理時間(秒)	8.01	7.99	8.35	8.26	8.24
共存 OS 処理時間(秒)	16.06	14.09	12.65	10.58	8.46

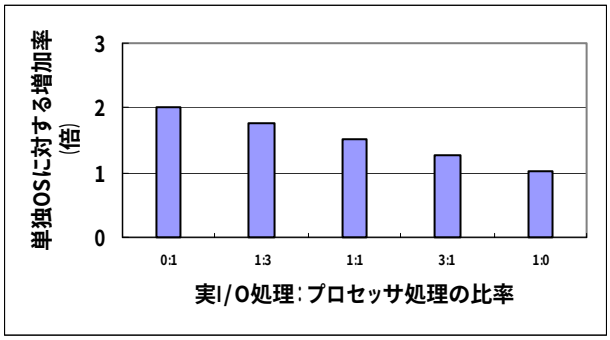

図 4 処理遅延の割合

(3) 混在処理 (M:N＝1:3，1:1，3:1)の場合

実 I/O 処理の I/O データサイズを 512 バイトにして実 I/O 処理とプロセッサ処理の割合を変化させて測定した結果を表 7 に示す．

プロセッサ処理の割合が大きくなるにしたがって単独 OS の場合に対する増加比は大きくなる．これは，単独 OS の処理時間と比較してプロセッサ処理の処理時間は 2 倍となり，実 I/O 処理の処理時間はほとんど変わらないことに起因する．

図 4 に、単独 OS に対する共存 OS の処理遅延の割合をグラフ化したものを示す．プロセッサ処理の割合が多くなると処理時間は 2 倍に近づき，実 I/O 処理の割合が大きいと処理時間は単独 OS の場合の処理時間に近づく．

3.4.3 各 OS の相互影響

ここでは、先行 OS の側にもアプリケーションを走行させ、NINJA の両 OS で評価用プログラムを実行した場合の共存 OS の処理時間を計測し，AP 処理内容と相互影響の関係を明らかにする．実 I/O 処理の I/O データサイズを 512 バイトにして，先行 OS，共存 OS ともに実 I/O 処理とプロセッサ処理の割合を、0：1，1：3，1：1，3：1，1：0 と変化させて測定を行った．測定結果を表 8 に示し，先行 OS での走行 AP なしの場合の共存 OS 処理時間に対する，処理遅延の割合をグラフ化したものを図 5 に示す．図 5 より以下のことがわかる．

(1)先行 OS の実 I/O 処理の割合が大きくなると共存 OS 側での処理の遅延が大きくなる．(表 8 の下にいくほど遅延が大)

(2)上記は，共存 OS 側の処理における I/O 処理の割合が大きいほど遅延が大きくなりにくい．

これは，両 OS とも I/O 処理の割合が大きくなると，他方の OS から自 OS に制御を切り戻す契機の多くがタイマ割込みでなく，I/O 割込みにより発生し，プロセッサ処理に比べ優先的に制御を取り戻すことが可能になることに起因する．

I/O 割込みが発生するタイミングは不定なため，共存 OS の実行時に先行 OS 側での I/O 割込みにより OS が切り戻される割合が大きくなると，共存 OS 側での処理の測定値における処理時間のばらつきが大きくなる．表 9 に共存 OS でプロセッサ処理のみを行った場合(M:N=0:1)に測定した処理時間の分散を示す．先行 OS の I/O 処理の割合が大きいほど共存 OS 側の処理時間のばらつきが多くなっていることがわかる．

表 8 両 OS で評価用プログラムを実行した際の共存 OS 側処理時間(秒)

先行 OS 処理比率 ＼ 共存 OS 処理比率	0：1	1：3	1：1	3：1	1：0
0：1	16.06	14.08	12.67	10.60	8.47
1：3	16.38	14.24	12.81	10.70	8.53
1：1	16.75	14.55	13.09	10.82	8.55
3：1	17.48	15.00	13.35	11.00	8.60
1：0	18.74	16.07	14.23	11.37	8.67

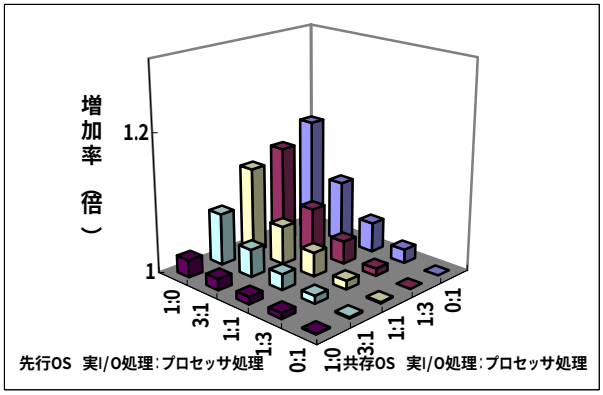

図 5 先行 OS 処理内容による処理遅延の割合

表 9 プロセッサ処理のみ行った場合の測定値の分散

先行 OS 処理内容 I/O 処理:ﾌﾟﾛｾｯｻ処理	0:1	1:3	1:1	3:1	1:0
共存 OS の分散	0.000	0.002	0.019	0.037	0.039

4. おわりに

デュアル OS 方式を Linux に適用した NINJA を実装し，特徴を明らかにするため評価を行った．

評価として，OS 初期化処理時間を測定し，先行 OS は単独 OS より早く起動し，共存 OS はトレースモードを利用した起動のため単独 OS より遅延することを示した．

OS 環境保存/復元処理時間を測定し，走行中の OS が占有しないハードウェア資源からの割込みへの応答時間が 2.33 マイクロ秒と少ない時間で行えることを示した．

実 I/O 処理とプロセッサ処理を混在させたプログラムを実行し，プロセッサ処理の割合が多くなると処理時間は単独 OS の 2 倍に近づき，実 I/O 処理の割合が大きいと処理時間は単独 OS の場合の処理時間に近づくことを示した．

今後は，I/O 割込みによる OS の切り替えのみではなく，各 OS 上を走行する AP の負荷状態に応じたプロセッサの有効利用方式の検討を行う予定である．

参考文献

[1]G.J.Popek,R.P.Goldberg:"Formal Requirementfor Virtualizable Third Generation Architectures,"Commun.ACM,17(7),pp.412-421,1974

[2]C.A.Waldspurger:"Memory Resource Management in Vmware ESX Server",OSDI2002

[3]J.Dike:"A user mode port of the Linux Kernel“ ,ALS2000,pp.63

[4]田渕正樹，桝本圭，伊藤健一，乃村能成，谷口秀夫:"2 つの Linux を同時走行させる機能の設計と評価",コンピュータシステム(CS2003)シンポジウム論文集,pp.71-78