Vol.2012-OS-122 No.16
2012/8/2

Mintオペレーティングシステムにおける実メモリ分配法

宮﨑 清人[1]　乃村 能成[1]　谷口 秀夫[1]

概要：我々は，1 台の計算機上で複数の Linux を同時走行させる方式の OS として，Mint オペレーティングシステムを開発している．Mint で広大なメモリ空間を利用可能にするため，64bit Linux をベースとした Mint を実現する．これを実現する方法として，64bit Linux の実メモリ利用法を変更せずに適用する方法が考えられる．しかし，64bit Linux の実メモリ利用法を変更せずに適用する場合，Mint の各 OS に任意の割合で実メモリを分配できない．そこで，本稿では，Mint において各 OS に任意の割合で実メモリ分配を可能にする課題について述べ，対処を明らかにする．また，この対処について，変更するコード量と OS の起動時間を評価する．

A Method for Segmenting Memory in Mint Operating System

Kiyohito Miyazaki[1]　Yoshinari Nomura[1]　Hideo Taniguchi[1]

Abstract: We have developed the Mint Operating System, which runs multiple Linux instances simultaneously on a single computer. In order to make the Mint utilize a huge amount of memory, we are developing another Mint based on 64-bit Linux. To achieve this, the most possible approach is to employ the memory-utilization method of the original 64-bit Linux directly. However, with this method, we cannot segment memory for each Linux in Mint in an arbitrary manner. In this paper, we describe the assignment of memory partitioning and show a solution for that. In addition, we evaluate code modification counts and times for booting OSes about this solution.

1. はじめに

計算機に搭載されるプロセッサのコア数や実メモリ量が増加し，計算機の性能が向上している．これらの資源を効率的に利用するため，1 台の計算機上で複数のオペレーティングシステム (以降，OS と略す) を同時走行させる方式の研究が活発に行われている．この方式として，仮想計算機 (VM : Virtual Machine) を用いる方式 (VM 方式) が広く用いられている．VM 方式を用いたソフトウェアの例として，Xen[1] や VMware[2] がある．しかし，VM 方式では仮想化により実計算機に比べて性能が低下する問題がある．

そこで，我々は，実計算機に近い性能で複数の OS を走行可能な Mint(Multiple Independent operating systems with New Technology) オペレーティングシステム [3] を開発している．Mint は Linux をベースに開発されており，1 台の計算機上で複数の Linux を同時走行させる方式の OS である．Mint では，仮想化によらず各 OS を実計算機上で直接走行させることで，実計算機に近い性能での OS の走行を実現している．また，OS 毎の独立性を実現しており，各 OS は互いに処理負荷の影響を与えない．

64bit Linux をベースとして Mint を実現する場合，64bit Linux の実メモリ利用法の影響を受け，各 OS に分配可能な実メモリ量の割合が制限されてしまう．

本稿では，Mint において，各 OS に任意の割合で実メモリ分配を可能にする課題について述べ，対処を明らかにする．また，対処の実現のために変更したコード量と各 OS の起動時間の観点から，評価する．

2. Mint の基本構成と問題点

2.1 基本構成

Mint は Linux をベースに開発されており，1 台の計算機

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

上で仮想化によらず複数の OS を走行させる方式の OS である．Mint では，計算機資源を分割し，各 OS を実計算機上で直接走行させることで，実計算機に近い性能を実現している．Mint では，各 OS が使用する計算機資源を以下のように決定している．

(1) プロセッサ：コア単位で分割し，各 OS は 1 つ以上のコアを占有する．
(2) 実メモリ：空間分割し，各 OS に分割領域を分配する．
(3) 入出力機器：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．

以上のように計算機資源を分割し，各 OS が占有する結果，Mint は各 OS の独立性という特徴を実現している．Mint により同時に走行する各 OS は，互いに処理負荷の影響を与えない．

2.2 64bit Linux の実メモリ利用法

64bit Linux をベースとした Mint を実現することにより，Mint において広大なメモリ空間を利用可能になる [4]．しかし，64bit Linux の実メモリ利用法を Mint に適用する場合，Linux が持ついくつかの制約が問題になる．これらの制約は単一カーネルで動作する Linux では問題とならないものの，Mint における複数 OS でのメモリ有効利用を考えた場合問題になる．この様子を**図 1** に示し，以下で説明する．

(制約 1) kernel text mapping に実メモリの先頭 512MB の領域をマッピングしなければならない．64bit Linux では，カーネルにアクセスするための領域として，仮想メモリ空間に kernel text mapping という領域がある．この領域には，実メモリの先頭から 512MB の領域をマッピングしなければならない．

(制約 2) kernel text mapping にマッピングされた実メモリ領域内にカーネルを配置しなければならない．(制約 1) で述べた実メモリ領域内，つまり，実メモリの先頭から 512MB 以内の領域にカーネルを配置しなくてはならない．

(制約 3) OS はカーネルを含む連続な実メモリ領域しか利用できない．

以上の制約に従い，64bit Linux は実メモリの先頭 512MB の領域内にカーネルを配置し，カーネルを配置した残りの実メモリ領域全体を利用している．

なお，以降では，kernel text mapping にマッピングされる実メモリ領域をカーネル配置領域と呼ぶ．

2.3 Mint での実メモリ分配の問題点

Mint の各 OS が前節の制約に従って実メモリ分配を実現する例を**図 2** に示し，そこから生じる問題点を以下で説明する．図 2 では，OS1，OS2，OS3，…，OSn の n 個 ($n \geq 2$) の OS を同時に走行させる．また，若番の OS ほど実メモリの先頭に近い分割領域を占有すると仮定する．Mint の各 OS が 2.2 節に示した Linux の実メモリ利用法の制約に従う結果，各 OS のカーネル配置領域は，全て実メモリの先頭 512MB の領域になる．また，各 OS の占有する実メモリ領域は連続でなければならないため，実メモリの先頭から 512MB の含む連続な n 個の領域に実メモリを分割し，これらを各 OS に分配しなければならない．この結果，OSn は実メモリの大半を占有できる，一方，他の OS は，カーネル配置領域を分割した少量の実メモリしか占有できない．

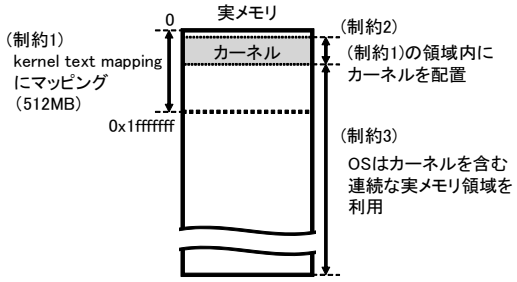

図 1 実メモリ利用の制約

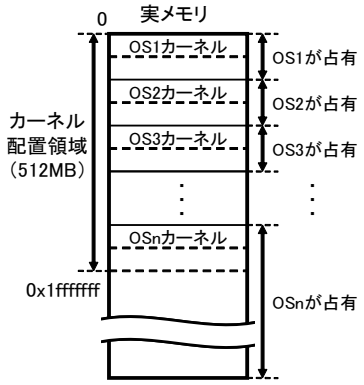

図 2 64bit Linux ベース Mint の実メモリ分配の様子

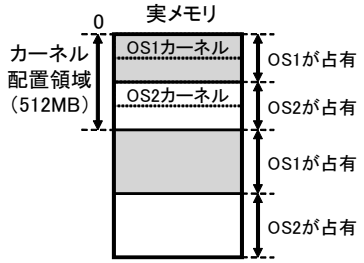

図 3 実メモリ分配法 (方式 1)

3. 対処

各 OS に任意の割合で実メモリを分配する方式として以下の案がある．

(方式 1) 各 OS が非連続な実メモリ領域を占有可能にする．この様子を**図 3** に示し，以下で説明する．本方式では (制約 3) を撤廃することで対処する．カーネル配

置領域を対処前と同様の方法で分割して各 OS に 1 つずつ分配し，各 OS は，この分割領域をカーネル用の領域に使用する．また，カーネル配置領域以外の領域も任意の割合で分割し，各 OS に 1 つずつ分配する．つまり，各 OS は非連続な実メモリ領域を占有する．本方式では，カーネル配置領域以外の領域を各 OS に任意の割合で分配できるため，課題に対処できる．

(方式 2) カーネル配置領域を拡大する．この様子を**図 4** に示し，以下で説明する．(制約 1) では，仮想メモリ空間の kernel text mapping の大きさは 512MB である．本方式では，この大きさを拡大する．Mint で同時に走行させる各 OS のカーネルを含む大きさまで，カーネル配置領域を拡大する．これにより，各 OS のカーネル配置位置は，実メモリの先頭から 512MB 未満の領域に限定されなくなる．この結果，各 OS の占有する分割領域の連続性を保ったまま，各分割領域の先頭と終端を自由に設定できるため，課題に対処できる．

(方式 3) カーネル配置領域の先頭アドレスを各 OS で変更可能にする．この様子を**図 5** に示し，以下で説明する．(制約 1) では，カーネル配置領域の先頭アドレスは常に実メモリの先頭である．本方式では，この先頭アドレスを OS 個別に変更可能にする．各 OS に分配する実メモリ領域の先頭アドレスに合わせ，各 OS のカーネル配置領域の先頭アドレスを変更する．これにより，カーネル配置領域を OS 個別に設ける．この結果，(方式 2) と同様に，各 OS のカーネル配置位置は実メモリの先頭から 512MB 未満の領域に限定されなくなり，課題に対処できる．

上記の方式の比較を**表 1** に示し，以下で説明する．

(方式 1) は，OS がハードウェアから取得するメモリマップを書き換え，各 OS に非連続な実メモリ領域を占有させることで実現できる．変更箇所は上記のみであり，変更量は小規模である．問題点は，配置可能なカーネルの大きさは合計で 512MB 未満に制限されることである．例えば，100MB を超える高機能なカーネルを使用する場合，同時に走行可能な OS の数は数個にとどまる．また，1 つのチップに 100 個以上のコアを搭載したメニーコアプロセッサが登場しており [5]，今後は同時に数十個の OS を走行させたいという要求が発生することが考えられる．(方式 1) では，この要求に対応できなくなる可能性がある．

(方式 2) は，仮想メモリ空間の構造を変更し，kernel text mapping を拡大することで実現できる．また，ページテーブルを作成する処理を変更し，kernel text mapping にマッピングする実メモリ領域の大きさを変更する．問題点は，実現が困難であることである．Linux では，カーネル配置領域の大きさは 512MB であることを前提としたコードが多く存在すると予想できる．これらの当該箇所はカーネルの幅広い箇所にわたっており，特定することが困難であると予想できる．

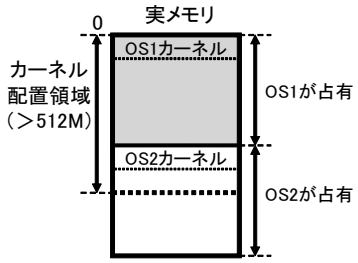

図 4 実メモリ分配法 (方式 2)

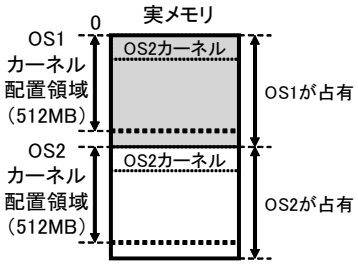

図 5 実メモリ分配法 (方式 3)

(方式 3) は，ページテーブルを作成する処理を変更し，実メモリ領域のマッピングを変更することで実現できる．このために変更するコードは，OS の起動処理の初期段階に集中している．問題点は，変更量が (方式 1) に比べると多いことである．ただし，(方式 1) にある制約は存在しない．

以上のことから，コード変更量は最少ではないものの，変更箇所が局所化されているため工数が見積もりやすく，かつ動作において制約の小さい (方式 3) を採用する．

4. 実現方式

以下で実メモリ分配の規則を説明する．OSi は以下の規則に従って実メモリ領域を使用する．ただし，OSi は Xi 番地から Yi 番地までの連続な実メモリ領域を占有し，カーネルの大きさは ZiB であると仮定する．この際，若番の OS ほど実メモリの先頭に近い分割領域を使用すると仮定する．

(規則 1) Xi は 16MB の境界に従い，Yi は 4KB の境界に従う．64bit Linux では，カーネルを配置する先頭アドレスは 16MB の境界に従っている．また，カーネルを配置する先頭アドレスと実メモリの先頭，つまり OS の占有する実メモリ領域の先頭アドレスとの差は 0x1000000(16M) である．このため，OS の占有する実メモリ領域の先頭アドレス Xi は 16MB の境界に従い，0x1000000(16M) の倍数とする．また，64bit Linux で扱う最小ページサイズは 4KB であるため，4KB の端数の領域を利用できない．このため，OS の占有する実メモリ領域の終端アドレス Yi は 4KB の境界に従い，0x1000(4K) の倍数とする．

(規則 2) kernel text mapping には，占有する実メモリ領

表 1 方式の比較

	実現の方法	変更量	問題点
(方式 1)	メモリマップを書き換え	小	カーネルの大きさは合計 512MB 以下
(方式 2)	仮想アドレス空間の構造を変更	大	実現が困難
(方式 3)	実メモリのマップピング処理を変更	中	変更量は最少ではない

域の先頭アドレス (Xi 番地) から 512MB の領域をマッピングする．kernel text mapping にマッピングする実メモリ領域の先頭アドレスを変更し，大きさは変更しない．

(規則 3) カーネルを Xi+0x1000000(16M) 番地から ZiB の領域に配置する．標準設定の 64bit Linux では，カーネル配置領域の先頭アドレスに 0x1000000 (16M) を加えた実アドレスを先頭として，カーネルを配置する．この関係を維持することで，変更するコード量を削減する．

(規則 4) アドレスは以下の条件を満たす．

(A) Xi+0x1000000+Zi < Yi．つまり，OSi の占有する実メモリ領域の終端は，OSi のカーネルの終端よりも老番側とする．

(B) Yi < Xj ($i < j$)．つまり，各 OS の占有する実メモリ領域は互いに重複しない．

(規則 5) 4G 番地以内の実メモリ領域から少なくとも 64MB+56KB の領域を占有する．OS の使用する実メモリ領域のうち，以下の 3 つは，32bit の実アドレスでアクセス可能な領域に配置する必要がある．

(A) kernel direct mapping tables
実メモリの先頭から 4GB の領域をマッピングするためのページテーブル (最大 24KB)

(B) software IO TLB
32bit デバイスが用いる I/O に用いるバッファ領域 (64MB)

(C) software IO TLB overflow buffer
(B) の予備領域 (32KB)

このため，4G 番地未満の実メモリ領域から少なくとも 64MB+56KB の領域を占有する．この際，OS の占有する実メモリ領域は非連続としなければならない．

以上の規則に従った対処例を**図 6** に示し，以下で説明する．図 6 は，Mint により OS1 と OS2 の 2 つの OS が同時に走行させる場合の例である．OS1 は X1 番地から Y1 番地までの連続な実メモリ領域を占有し，OS2 は X2 番地から Y2 番地までを占有する．これらのアドレスは (規則 1) を満たしているものとする．各 OS の占有する実メモリ領域は互いに重複せず，アドレスは (規則 4-B) を満足する．また，(規則 2) に従い，OS1 と OS2 はそれぞれ自身の占有領域の先頭から 512MB の大きさの実メモリ領域を kernel text mapping にマッピングする．さらに，(規則 3) に従い，OS1 と OS2 はそれぞれ自身の占有領域の先頭から 16MB

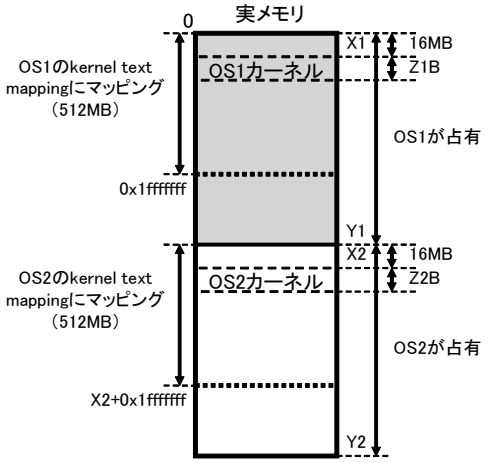

図 6 対処後の実メモリ分配例 1

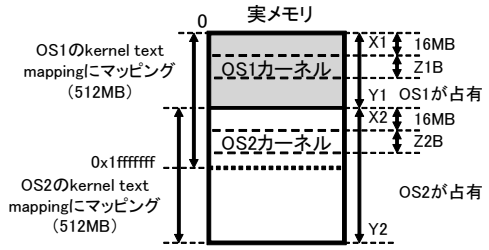

図 7 対処後の実メモリ分配例 2

の位置を先頭としてカーネルを配置する．OS1 と OS2 のカーネルの大きさはそれぞれ Z1B と Z2B とする．このとき，アドレスの関係は (規則 4-A) を満たしているものとする．

なお，OS は常に 512MB のカーネル配置領域を設定するため，カーネル配置領域の終端が他 OS の占有する実メモリ領域の終端を越える場合がある．この場合であっても，OS の占有する実メモリ領域の終端を越えて実メモリを使用することはない．この場合の例を**図 7** に示し，以下で説明する．図 7 では，X2 番地から 0x1fffffff(512M-1) 番地までの実メモリ領域は OS2 の占有する領域であると同時に，OS1 のカーネル配置領域である．この領域は，OS1 のメモリマップでは使用しない領域として設定されているため，OS1 によって使用されることはない．したがって，OS2 の占有領域が OS1 によって書き換えられることはない．このため，ある OS のカーネル配置領域の終端が OS の占有する実メモリ領域の終端を越える場合であっても，上記の規則によって (対処) を実現できる．

上記の実現方式を実装し，動作を確認した．

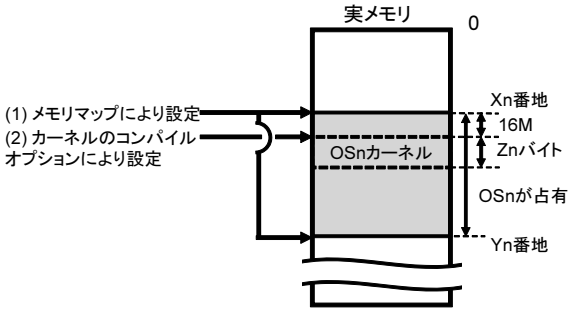

図 8 OS の占有する実メモリ領域の設定例

5. 実装

4 章の実現方式に基づき，対処を実装した．この際，各 OS の占有する実メモリ領域をコンパイル時に静的に決定している．ユーザが各 OS の占有する実メモリ領域をどのように設定するかについて**図 8** に示し，以下で説明する．OSn の占有する実メモリ領域の先頭アドレス Xn と終端アドレス Yn を設定するには，ユーザはメモリマップを書き換える．この設定については，最初に起動する先行 OS ではカーネルのセットアップルーチンを変更し，Mint において先行 OS から起動される後続 OS では，後続 OS の起動に用いる kexec[6] を変更する．また，カーネルの配置アドレス Xn+16M については，カーネルのコンパイルオプションを設定する config ファイルにより設定する．ユーザの行う設定は以上である．

カーネル側には，カーネルを上記の配置アドレスを先頭として展開し，走行可能にするための変更を加えている．具体的には，起動処理において，圧縮カーネルを展開する処理とカーネルの先頭付近の処理の 2 箇所において，ページテーブルの作成方法を変更している．また，カーネルの各シンボルに仮想アドレスを設定するために使用するオフセットの値を変更している．さらに，標準の 64bit Linux では，カーネルの配置先を実メモリの先頭 512MB 以内に限定するためのエラーチェックを行っている．このため，カーネルの配置先アドレスを変更可能にするにあたり，これらが問題となるため，変更した．以上により，対処を実装できた．

6. 評価

6.1 評価項目

3 章で示した (対処) について，以下の項目で評価する．
(評価 1)変更したコード量
(評価 2)起動処理に要する時間

(評価 1) として，対処を Mint に適用するために変更したコード量を評価する．Mint は Linux をベースに開発されており，Linux からの変更ステップ数を抑えることでメンテナンス性を向上させたいという要求がある．

表 2 変更したコード量

工程	ステップ数	ファイル数
(1) カーネル配置領域の変更	30 行	3
(2) その他	4 行	3
合計	34 行	6

表 3 実装環境

OS	Fedora 64bit(Linux Kernel 2.6.39)
CPU	Intel Core i7-2600 (3.4GHz)
各 OS の占有コア数	1
実メモリ	4GB×2
HDD	500GB×2

(評価 2) として，実現した Mint の起動時間が通常の Linux に比べて長くないことを示すため，Mint と通常の Linux で OS の起動処理に要する時間を比較し，評価する．また，Mint から実メモリ分配の対処を除外した実装についても起動時間を評価し，対処によって OS の起動時間が大きく変化しないことを示す．

6.2 変更したコード量

変更したコード量を**表 2** に示し，以下で説明する．

変更箇所を 2 つの工程に分けて説明する．(1) の工程では，カーネル配置領域の先頭アドレスを変更し，変更ステップ数は 28 行だった．具体的な変更箇所として，カーネルの先頭付近と圧縮カーネルの展開ルーチンにおいてページテーブルを変更し，kernel text mapping への実メモリのマッピングを変更した．(2) の工程では，(1) に伴って変更が必要になる箇所を変更し，変更ステップ数は 4 行だった．具体的には，カーネル配置領域は実メモリの先頭から 512MB であることを前提としたエラーチェックを変更した．

以上の通り，メモリ分配法の課題への対処のために変更ステップ数は 34 行である．また，変更したファイル数は 6 つであり，これらは全て OS の起動の初期段階で使用されるファイルだった．以上のことから，対処は少ない変更で実現でき，変更箇所を局所化できていることが分かる．

6.3 起動処理に要する時間

以下の 3 つをそれぞれシングルコア上で起動させ，起動時間を評価する．Linux Kernel のバージョンはすべて 2.6.39 64bit である．

(1) 通常の Linux
(2) Mint から実メモリ分配に関する対処を除外した実装
(3) Mint

(2) と (3) については，最初に起動する先行 OS の起動時間を測定する．計算機の電源を投入してからカーネルスレッドの実行の直前までの時間を TSC レジスタをの値により求める．実装環境を**表 3** に示す．

表 4　起動処理に要する時間 (25 回の平均値)

	起動時間 (秒)
(1) 通常の Linux	16.36
(2)Mint から実メモリ分配の対処を除外	16.28
(3)Mint	16.30

起動処理に要する時間をそれぞれ**表 4** に示し，以下で考察する．表には，それぞれ 25 回の測定の平均値を示している．これらより，以下のことが分かる．

Mint の起動時間は，ベースとなっている 64bit Linux に比べて約 0.06 秒短い．これは，Mint では OS の占有するハードウェアのみを登録することにより，64bit Linux に比べて登録するハードウェアが少ないためである．

また，Mint の起動時間は，各 OS に任意の割合で実メモリを分配するための対処によって約 0.02 秒長くなっていることが分かる．この原因として，主に圧縮カーネルの展開ルーチンにおいて，ページテーブルのエントリを作成する処理が増加したことが考えられる．

以上のことから，Mint の起動時間は 64bit Linux と遜色なく，また，対処による起動時間の増加はわずかであることが分かる．

7. おわりに

Mint において，各 OS に任意の割合で実メモリ分配を可能にする課題について述べ，対処を明らかにした．また，対処の実現のために変更したコード量と OS の起動時間の観点から，評価した．

対処として，3 つの案を比較し，これらのうち，カーネル配置領域の先頭アドレスを各 OS で変更可能にする方式を採用した．また，対処の具体的な実現方式を示し，実装について述べた．

評価により，対処のために変更したコード量は少なく，変更箇所を局所化できていることを示した．また，Mint の起動時間は 64bit Linux に比べてわずかに短いのみであり，また，課題への対処による起動時間の増加はわずかであることを示した．

残された課題として，OS 間での実メモリ領域の移譲の実現がある．Mint では，各 OS への実メモリ分配をカーネルのコンパイル時に静的に決定しており，OS の負荷に応じて動的に変更できない．そこで，OS 間で実メモリ領域の移譲を実現し，実メモリの利用効率をさらに向上させる．

謝辞　本研究の一部は, 科学研究費補助金基盤研究 (B)(課題番号：24300008) による．

参考文献

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, “Xen and the Art of Virtualization, ” Proc. of the 19th ACM Symposium on Operating Systems Principles, pp.164-177, 2003.

[2] Jeremy Sugerman, Ganesh Venkitachalam, and Beng-Hong Lim, Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor, ” Proceedings of the General Track: 2002 USENIX Annual Technical Conference, pp.1-14, 2001.

[3] 千崎 良太, 中原 大貴, 牛尾 裕, 片岡 哲也, 栗田 祐一, 乃村 能成, 谷口 秀夫, “ マルチコアにおいて複数の Linux カーネルを走行させる Mint オペレーティングシステムの設計と評価, ” 電子情報通信学会技術研究報告, vol.110, no.278, pp.29-34, 2010.

[4] 中原 大貴, 乃村 能成, 谷口 秀夫, “ 32/64bit カーネル混載方式の実現, ” 電子情報通信学会技術研究報告,vol.111, no.255, pp.25-30, 2011.

[5] Seiler, Larry and Carmean, Doug and Sprangle, Eric and Forsyth, Tom and Abrash, Michael and Dubey, Pradeep and Junkins, Stephen and Lake, Adam and Sugerman, Jeremy and Cavin, Robert and Espasa, Roger and Grochowski, Ed and Juan, Toni and Hanrahan, Pat, “Larrabee: a many-core x86 architecture for visual computing, ” ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008 TOG Homepage Volume 27 Issue 3 Article No. 18 , pp.1-15, 2008.

[6] 中原 大貴, 千崎 良太, 牛尾 裕, 片岡 哲也, 乃村 能成, 谷口 秀夫, “ Kexec を利用した Mint オペレーティングシステムの起動方式 ,” 電子情報通信学会技術研究報告, vol.110, no.278, pp.35-40, 2010.