様式 C-19

科学研究費補助金研究成果報告書

平成 21 年 6 月 29 日現在

研究種目：若手研究（B）
研究期間：2007～2008
課題番号：１９７００３４
研究課題名（和文）　コアスケジューラを用いた適応的計算機資源割当てによる仮想計算機システムの高性能化
研究課題名（英文）　High performance virtual machine system by core-scheduler which allocates computation resources adaptively

研究代表者
毛利　公一（MOURI KOICHI）
立命館大学・情報理工学部・准教授
研究者番号：９０３１３２９６

研究成果の概要：
本研究課題は,仮想計算機モニタとその上で動作するゲスト OS とが協調することによる効率的な計算機資源管理手法を提案するものである．研究成果は大きく二つから成る．(1)マルチコア対応のゲスト OS として Lavender の構築を行ったこと，(2)仮想計算機モニタとゲスト OS が協調する適応的計算機資源割当て方式を開発したことである．(1)では，ネイティブの計算機上で動作する Lavender と，仮想計算機モニタ Xen の準仮想化環境で動作する Lavender を実装した．(2)では，協調に必要なメカニズムの検討と設計が完了し，その知見が得られた．

交付額

（金額単位：円）

	直接経費	間接経費	合　計
２００７年度	1,200,000	0	1,200,000
２００８年度	1,000,000	300,000	1,300,000
総　計	2,200,000	300,000	2,500,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウェア
キーワード：オペレーティングシステム，仮想化技術

１．研究開始当初の背景

近年，実機を複製することを目的とした仮想計算機技術が注目されている．仮想計算機を実現するソフトウェアである仮想計算機モニタとしては,VMware，Virtual PC，Xen などがあり，プロセッサを１台しか搭載していない計算機でも，仮想的に複数の計算機を生成し，そこで OS（仮想計算機上で動作する OS を特にゲスト OS と呼ぶ）を動作可能としている．その用途は，1 台の計算機で多数のインターネットサーバを構築・提供するホスティングサービス，逆に多数の企業内のサーバを１台にまとめてコストダウンを図るシステム統合，OS の構築を支援するテスト・デバッグ環境をはじめ，既に種々の用途に用いられている．しかし，今後さらなる展開を予想することができる．本研究課題は，以下で述べる展開をにらみ，かつその先にある新たな課題へ取り組むものである．

【展開 1】今後のユビキタスコンピューティ

ングの広がりや携帯機器の高性能化により，小さな機器の中に複数の OS が搭載されることが予想できる．例えば，携帯電話に通信制御のためのリアルタイム OS とインタフェースやアプリケーションのための OS の両方を搭載することが考えられる．研究代表者が留学していたイリノイ大学では，携帯電話メーカの Motorola とともに，仮想計算機技術を用いてそれを実現するようなリアルタイム仮想計算機モニタの研究が進められている．

【展開 2】仮想計算機技術が注目される中，Intel や AMD などのプロセッサメーカは，ソフトウェアである仮想計算機モニタで実現されてきた機能の一部をハードウェアとして実現し，メモリを中心とした仮想計算機間の保護機能の向上，仮想計算機で用いられる計算機資源の管理のためのデータ構造の標準化など，仮想化技術をプロセッサへ搭載した．これによって，仮想計算機モニタをより容易で効率的に構築可能とした．

【展開 3】その一方で，動作周波数の向上に依存していたプロセッサの高速化は限界が近いとされ，一つのプロセッサパッケージの中に，複数のプロセッサコア（マルチコア）を搭載することで高速化を図る手法が主流となっている．現在は，2 つのコアを有するデュアルコアプロセッサが市販されており，2006 年後半以降に 4 つのコアを有するクアッドコアプロセッサが市販される予定となっていた．

２．研究の目的

以上で述べたように，仮想計算機技術が今後もより重要な技術の一つであり，プロセッサもそれを支援するための機能を付加し始めた．しかし，既存の仮想計算機技術は，既存の OS を仮想計算機上で効率的に動作させることを目的としている．本研究課題では，これをさらに発展させ，マルチコアかつ仮想化技術を有するプロセッサにおいて，複数のゲスト OS が動作しているときに，各ゲスト OS の処理特性や負荷を自動的に観測・検出し，仮想計算機モニタとゲスト OS が協調して適

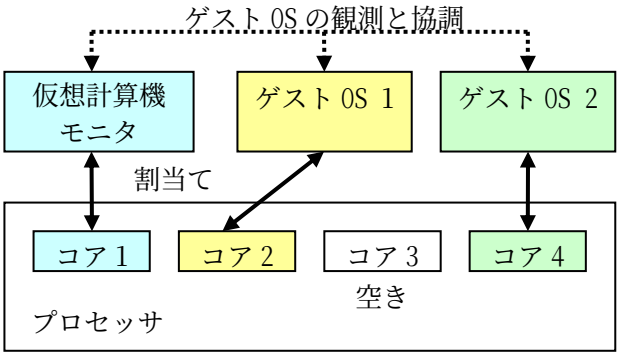

図 1 低負荷時のコア割付け

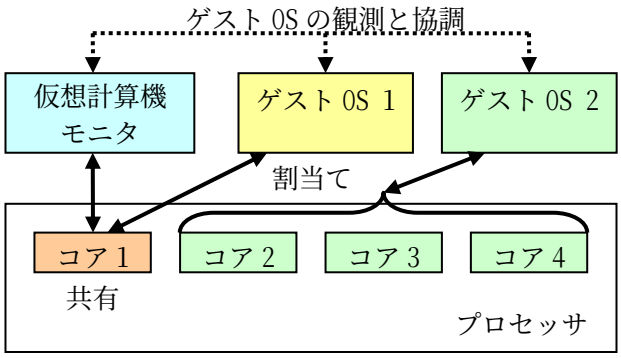

図 2 ゲスト OS 2 が高負荷時のコア割付け

応的にプロセッサコアやメモリなどの計算機資源を割り当てる，適応的計算機資源割当て手法を確立することを目的とするものである(図 1，図 2 参照)．これによって，複数の OS を動作させることによって得られるメリットに加え，見かけ上，プロセッサコアの数よりも大きな性能向上を得ることができる．例えば図 2 では，各ゲスト OS へ十分な処理速度を提供しながらも，コア数が 5 個であるのと同等に見せかけることができる．

３．研究の方法

OS を新規に開発することは相当の時間が必要であり，それだけで数年を費やしてしまう場合がある．一方で，Linux などのオープンソース OS を改造する手法では，初期の開発は不要であるが，逆にソースコードの規模が大きいために研究の本質以外の箇所での作業量が増加してしまう場合がある．これらのデメリットを解決し，かつ研究の本質を追及することへ時間を配分可能とするために，研究代表者が過去に開発したマイクロカーネル方式の OS である Lavender をベースに研究を遂行することとする．Lavender のソースコードは約 2 万行で，Linux の 300 分の 1 程度である．一方で，仮想記憶，プロセス・スレッド，シンプルな同期・通信機構など OS として必要な機能を有している．すなわち，本応募研究課題を遂行するためのベースとすることが適切であると考えている．

以下，平成 19 年度は①～③を，平成 20 年度は④～⑥を目標として研究を遂行する．

① デュアルコアプロセッサで Lavender を動作可能とする．
② 資源の変動を検出可能なメカニズムを構築する．
③ 仮想計算機モニタがゲスト OS の動作を観測可能とするメカニズムを構築する．
④ クアッドコアプロセッサで Lavender を動作可能とする．
⑤ コアスケジューラを構築する．
⑥ 仮想計算機モニタとゲスト OS 間での協調処理方式を実現する．

４．研究成果

(1) マルチコア対応ゲスト OS Lavender

Lavender の構築に向けて，本研究課題では 2 種類のアプローチを採った．一つはネイティブの計算機上で動作するもの，もう一つは Xen の準仮想化環境で動作するものとした．

ネイティブのものについては，計算機の起動時から動作する中心的なプロセッサであるブートストラッププロセッサ(BSP)が，それ以外のアプリケーションプロセッサ(AP)群を起動する手法を調査し，それを実装した．この流れは次のとおりである．

(a) AP 毎に存在する Local APIC(Advanced Programmable Interrupt Controller)の ID を取得する．
(b) AP に対してプロセッサ間割込み(IPI)を送信する．
(c) AP を初期化する．

これ以降は，CPU スケジューラがプロセスへプロセッサを割り当てればよい．

一方で，Xen の準仮想化環境への対応については，Xen に付属するサンプル OS である mini-OS を調査した．Xen によるドメインの生成が行われると，mini-OS が起動処理を開始する．

(a) スタック，共有メモリ，ハイパーバイザコール用のメモリ領域の確保を行い，スタックの初期化を行う．
(b) Xen から，esi レジスタを通じて VM の情報が格納されたアドレスが渡されるため，それをスタックへプッシュする．
(c) (b)でプッシュした esi レジスタの値を利用し，それが指す領域の内容を，起動時に確保した共有メモリ領域に設定する．
(d) メモリや割り込み，タイマの初期化などが行われ，最後にアイドルスレッドを実行する．

(a)の共有メモリ領域は，Xen から与えられるパラメータや，Xen との通信に用いられる領域であり，Xen とゲスト OS の両方が参照可能な領域である．ハイパーバイザコール用の領域には Xen が用意しているハイパーバイザコールの処理を行う関数へのポインタが格納される．

(b)では，VM の情報（start_info_t 型）が格納されたメモリ領域のアドレスが渡される．それを，直後に呼び出す，C 言語で記述された start_kernel 関数の引数として渡すために，スタックへプッシュする．これによって，mini-OS が VM に割当てられているメモリの情報や VCPU に関する情報，共有メモリ等を参照することができる．

(c)の処理によって，これ以降 I/O 要求などの Xen との通信が可能になる．そして，(d)で CPU スケジューラへと処理が移行する．

以上の方法で，ネイティブの計算機上で動作する Lavender と，準仮想化環境において動作する Lavender を実現した．

また，動的な計算機資源の変動に適応可能とするためのプロセッサ間同期機構についても実現した．特に，ロックされた資源があるときにプロセッサが新たに割り当てられた場合の処理方法，逆にプロセッサが減少する場合の処理方法を提案した．

プロセッサの増加には，プロセッサ数が 1 つから 2 つへと増加する場合と，プロセッサ数が 2 つ以上から増加する場合が考えられる．また，それぞれの場合でさらに，クリティカルセクションへのアクセス中に増加する場合と，クリティカルセクションへアクセスしていない際に増加する場合とが考えられる．このようなプロセッサ増加のタイミングのなかで問題となるのは，プロセッサ数が 1 つから 2 つへと増加しクリティカルセクションへのアクセス中に増加した場合である．通常の OS では，起動時にプロセッサ数が 1 つの場合プロセッサ間同期機構を利用せず，プロセスは割り込み禁止を行いクリティカルセクションへのアクセスを行う．このタイミングでのプロセッサの増加があり，このプロセッサが，他方のプロセッサがアクセスしているクリティカルセクションへのアクセスを行った場合，データの整合性を保つことが不可能になる．この問題点を解決するためには，本来シングルプロセッサでは利用しない，プロセッサ間同期手法を常に利用する．これによって，プロセッサ増加時におけるデータの整合性を維持することが可能となる．

プロセッサを開放する際のシナリオとして，OS が自発的に開放する場合と VMM からの要求によって開放する場合の 2 つが考えられる．前者の例は，OS が自ら解放するため問題とはならない．後者の例では，VMM はどの OS に対してプロセッサの開放要求を発行するかについて，次の点を考慮して適切に判断する必要がある．

- 開放要求タイミング
- プロセッサ要求の優先度と緊急度
- 開放するプロセッサの選出方法

開放要求を発行するタイミングについては，次の場合が考えられる．

- ゲスト OS 上で動作していない，または，利用率の低いプロセッサが存在している
- プロセッサの追加要求が，任意の VM 上の OS ら発行されたが，VM へ割り当てられていないプロセッサが存在しない
- 任意の VM 上のプロセッサの利用率が

高く，プロセッサの追加が必要とされた場合

プロセッサ追加要求の優先度と緊急度については，OS から VMM に対して発行されるプロセッサ追加要求に優先度を付加することで，複数の OS がプロセッサ追加要求を発行した際，どの OS に対してプロセッサを割当てるかを選択することが可能となる．また，プロセッサ追加要求に緊急度を付加することで，開放対象となった OS に対しどのようなタイミングでプロセッサを開放するかを指定することが可能となる．緊急度が高い場合，VMM は，開放対象となった OS がどのような処理をしていてもプロセッサの開放を行うよう開放要求を発行する．緊急度が低い場合，VMM は，開放対象となった OS が一定の基準を満たすまで，プロセッサの開放を待つ．基準としては，その時点での実行プロセスの終了，クリティカルセクション内の処理の完了といったものが挙げられる．

開放を行うプロセッサの選出方法については，一定の順序で開放するプロセッサを選出する方法と，スケジューラによって開放するプロセッサを選出する方法が考えられる．

以上から，VMM からゲスト OS へプロセッサの解放要求が行われ，それに対しゲスト OS のスケジューラが，解放するプロセッサを選出するケースを取り扱う．選出基準としては，各プロセッサ上のランキューに存在するプロセスの数，各プロセッサが獲得しているロック数，ロックの獲得待ち状態であるか否か，といったものが挙げられる．複数のプロセッサが一つのクリティカルセクションにアクセスするためロックを獲得しようとする際，プロセッサは待ち状態に陥り，プロセッサを無駄に占有することとなる．本手法では，このようなロック待ちのプロセッサを優先的に開放する．

(2) 想計算機モニタとゲスト OS が協調する適応的計算機資源割当て方式

本研究課題における協調は，VMM とゲスト OS が互いに情報を交換しながら効率的な資源利用を行うものである．VMM は，PCPU（物理 CPU）と VCPU（仮想 CPU）の割当てをスケジュールしており，これによりゲスト OS に資源を提供する．また，ゲスト OS は，提供された VCPU のスケジューリングを行い，プロセスに資源を割当てる．このような，2 段階のスケジューリングを，VMM とゲスト OS で物理資源の利用率やプロセスの動作状況などの情報を管理することで協調した資源管理を行う．

コアスケジューラによって，資源をスケジュールするためには，VMM・OS 間で通信する機構を実装する．OS は自身のプロセスによる資源の要求，現在の状況などに従い，VMM へと VCP の増減要求を発行する．OS からの VCPU 増減要求を受信した VMM は，全 VM 上で動作している OS の状況を考慮し，VM の資源を管理する．その結果，VM 上の VCPU の増加があった場合，VMM は対象の OS に対し VCPU の増加を通知する．VMM からの通知を受理した OS は VCPU の確保を行い，自身のプロセスを確保している VCPU に対してスケジューリングを行う．また，VMM から VCPU の解放要求があった場合は，OS は自身が確保している VCPU を開放し，VMM が開放された VCPU への PCPU の割当てを解除する．

協調型スケジューリングを行う場合，OS と VMM に，資源の増減に対応した機構を実装するだけでなく，双方向の通信機構を備える必要がある．このようなスケジューリングを行うことにより VMM と OS の連携の取れたスケジューリングが可能であり，結果的に見かけ上パフォーマンスの向上が見込まれる．

協調型仮想計算機システムでは，VMM と OS のそれぞれにおいて以下の 4 つの状況が想定される．

- OS が自身の資源不足を補うため資源の増加を要求したとき

OS が，あるプロセスを実行するに際し，現在割り当てられている VCPU では，必要な処理能力が不足していると判断した場合．具体例として，リアルタイムプロセス等が実行されるような場合に現状の資源ではデッドラインミスを起きるような場合が挙げられる．

- OS が自身の過剰資源を解放するため資源の減少を要求したとき

アイドル状態にある VCPU が複数あり，それらがしばらく利用されない状況にある場合．具体例として，携帯電話のベースバンド OS で通信が終了し，待ち受け状態になった場合が挙げられる．

- VMM がある OS の資源不足を認知したとき

一時的にある OS での処理に負荷がかかり，現状で割当てられている資源ではパフォーマンスの低下が見られる場合．具体例として，サーバ等で一時的に負荷が集中して発生したような場合が挙げられる．

- VMM がある OS の資源過剰を認知したとき

一時的に現状で割り当てられている資源に見合わない処理を行っているような OS が認知できた場合．具体例として，サーバ等で不定期な負荷がかかるような場合，負荷が発生していないような状態のときといったものが挙げられる．

以上で述べたそれぞれの状況において必要となる機能は次の通りである．

- OS が資源不足で資源の要求を行う場合

新たに VCPU を構築する可能性があるため，OS は VM 上に新たに追加された VCPU を初期化する必要がある．また，OS が自身で資源の増

加を要求する場合，資源要求の優先度は高いものと考えられる．しかし，システム上で PCPU が余っているとは限らないため，VMM ではこの資源増加の優先度を考慮し，増加させるか否かを判定する機能が必要となる．

- OS が資源過剰で資源の解放を行う場合

該当 VM に割り当てられている VCPU を停止させ，VMM はその VCPU に割り当てられている PCPU を開放する必要がある．OS 自身が VCPU の開放を行う場合，しばらくの間，その VCPU を利用する可能性が低いと考えられる．しかし，VCPU を VM から削除してしまうと再構築の際，オーバヘッドがかかる．このような状況でも，VCPU を削除するのではなく，利用不可の状態にすべきである．

- VMM がある OS の資源不足を認知した場合

このような状況の場合，VMM では二つの動作が考えられる．

(1) 現時点で余剰の PCPU がある場合
(2) PCPU がすべて VM 上の VCPU に割り当てられている場合

(1)の場合，割り振られていない PCPU を VCPU に割り振ることで要求を満たすことができる．(2)の場合，各 OS の状況を判断し，利用されていない，もしくは，利用率が低い VCPU に割り振られている PCPU を，必要としている VM の VCPU に割り振ることとなる．この場合，各 OS の状況を知るために OS との情報通信機能が必要となる．

- VMM がある OS の資源過剰を認知した場合

一時的に VCPU を停止させる可能性が高く，VCPU は削除せず，該当 VCPU を利用不可にすることで要求を満たす．VCPU を利用不可にするには，利用している OS に対しそのことを通知する必要がある．また，OS が資源を解放する場合がトリガとなって，このような状況が発生することが考えられる．

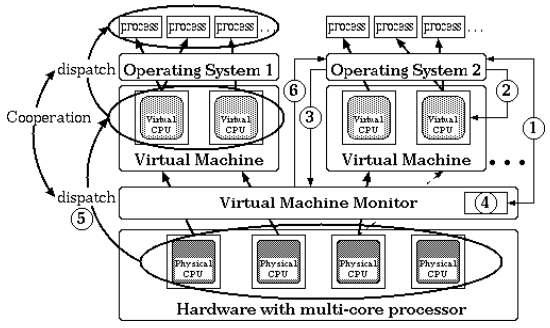

図 3 協調型仮想計算機システム

以上で述べた必要な機能をふまえ，ゲスト OS と VMM に必要な機構について述べる．まず，ゲスト OS に必要な機構を以下に示す．なお，各番号は図 3 中の番号と対応する．

① VMM との通信機構

OS 自身が，プロセッサ利用率，プロセス特性等の情報を VMM に対して送信する．これにより VMM は各 OS の動作状況を詳しく知ることができる．また，VMM から VCPU の割当て状況を受信することで，VCPU をスケジューリング対象から削除し停止状態を実現する．

② VM 上の資源管理機構

VMM によって割当てが変動される，VM の動的な資源の変動に対応する．新たに提供された VCPU の利用や，VCPU の利用停止を実現する．

③ VMM への資源変動要求機構

資源の変動要求を VMM に通知することで，OS から VMM に対して資源の増加，減少要求の発行を実現する．

次に，VMM に必要な機構を以下に示す．

④ OS の情報管理機構

OS との通信によって得られた情報を管理し，システム全体の利用状況を把握する．OS から資源要求があった場合にシステム全体を考慮した PCPU の割当てが可能となる．また，各 OS の不足，過剰資源の認知を実現する．

⑤ VM の資源変動機構

VM 上に割り振られている資源を管理し，実際に VM の資源を変動させる．新たな VCPU の構築，VCPU への PCPU の割当て，解除を実現する．

⑥ 要求に応答する機構

OS に対して VCPU の割当て状況，新規に構築した VCPU の情報などを通知する．これによって，OS は自身が出した要求に VMM がどう対応したかを認知することが可能となり，それらに対応した処理を行うことが可能となる．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計４件）

① 金城 聖，荒木 裕靖，元濱 努，片山 吉章，毛利 公一，仮想計算機モニタを利用したゲスト OS の入出力要求監視手法，情報処理学会研究報告 2008-OS-107，Vol．2008．No．9，pp．9-15，2008，査読なし．

② 荒木 裕靖，金城 聖，元濱 努，片山 吉章，毛利 公一，動的なプロセッサコア資源の変動に適応する CPU スケジューラの検討，情報処理学会研究報告 2008-OS-107，Vol．2008，No．9，pp．119-125，2008，査読なし．

③ 荒木 裕靖，佐治 勇季，金城 聖，毛利 公

二, 協調型仮想計算機システム向け OS におけるプロセッサ間同期手法, 情報処理学会研究報告 2008-OS-108, Vol. 2008, No. 35, pp. 171-178, 2008, 査読なし.

④ 金城 聖, 永島 力, 元濱 努, 片山 吉章, 毛利 公一, リアルタイム仮想化ソフトウェア基盤におけるタイマ割込み通知機構, 情報処理学会研究報告 2008-EMB-10, Vol. 2008, No. 116, pp. 9-16, 2008, 査読なし.

〔学会発表〕(計５件)

① 永島 力, 金城 聖, 荒木 裕靖, 毛利 公一, 仮想計算機上でのリアルタイム OS の動作に対する評価と考察, 情報処理学会第 70 回全国大会, 2008 年 3 月 13 日, 筑波大学 (茨城県つくば市).

② 永島 力, 金城 聖, 毛利 公一, 仮想計算機モニタにおける割込み処理の性能評価, 情報処理学会第 71 回全国大会, 2009 年 3 月 10 日, 立命館大学びわこ・くさつキャンパス (滋賀県草津市).

③ 小野 利直, 金城 聖, 永島 力, 毛利 公一, 組込みシステムを想定した Xen の軽量化手法, 情報処理学会第 71 回全国大会, 2009 年 3 月 10 日, 立命館大学びわこ・くさつキャンパス (滋賀県草津市).

④ 荒木 裕靖, 毛利 公一, 協調型仮想計算機システムにおける協調機構, 情報処理学会第 71 回全国大会, 2009 年 3 月 10 日, 立命館大学びわこ・くさつキャンパス (滋賀県草津市).

⑤ 金城 聖, 永島 力, 毛利 公一, 仮想計算機上の RT-OS と非 RT-OS に対するタイマ割込み管理手法, 情報処理学会第 71 回全国大会, 2009 年 3 月 10 日, 立命館大学びわこ・くさつキャンパス (滋賀県草津市).

〔その他〕
ホームページ等
http://www.lavender.org/

６．研究組織
(1)研究代表者
毛利　公一 (MOURI KOICHI)
立命館大学・情報理工学部・准教授
研究者番号：９０３１３２９６

(2)研究分担者
なし

(3)連携研究者
なし