(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5507660号
(P5507660)
(45)発行日　平成26年5月28日(2014.5.28)　(24)登録日　平成26年3月28日(2014.3.28)

(51)Int.Cl.　　F I
G06F 9/46 (2006.01)　　G06F 9/46 350
G06F 9/50 (2006.01)　　G06F 9/46 462A

請求項の数 20　(全 32 頁)

(21)出願番号　特願2012-502133(P2012-502133)
(86)(22)出願日　平成22年3月19日(2010.3.19)
(65)公表番号　特表2012-521610(P2012-521610A)
(43)公表日　平成24年9月13日(2012.9.13)
(86)国際出願番号　PCT/US2010/028034
(87)国際公開番号　WO2010/111149
(87)国際公開日　平成22年9月30日(2010.9.30)
審査請求日　平成25年2月5日(2013.2.5)
(31)優先権主張番号　12/412,272
(32)優先日　平成21年3月26日(2009.3.26)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100080137
弁理士　千葉　昭男
(74)代理人　100096013
弁理士　富田　博行
(74)代理人　100153028
弁理士　上田　忠

最終頁に続く

(54)【発明の名称】仮想マシン用非一様仮想メモリーアーキテクチャー

(57)【特許請求の範囲】
【請求項１】
仮想マシンをインスタンス化するためのリクエストを受信するステップであって前記リクエストが、前記仮想マシンに関する特性を含んでいるものと、
前記特性に基づいて、前記仮想マシンに関する仮想非一様メモリ・アーキテクチャ（ＮＵＭＡ：Non-Uniform Memory Architecture）ノードトポロジーを選択するステップであって前記仮想ＮＵＭＡノードトポロジーが、複数の仮想ＮＵＭＡノードを含んでいるものと、
前記仮想マシンを計算機システム上でインスタンス化するステップであって前記仮想マシンが、前記複数の仮想ＮＵＭＡノードを含んでいるものと、
前記複数の仮想ＮＵＭＡノードの内の特定の仮想ＮＵＭＡノードにおけるメモリー圧力に基づいて、前記特定の仮想ＮＵＭＡノードに割り当てられるゲストメモリー容量を調整するステップと、を含む方法。
【請求項２】
更に、
前記複数の仮想ＮＵＭＡノードの内の第２の仮想ＮＵＭＡノードにおけるメモリー圧力が所定の値よりも大きいことを決定するステップと、
前記第２の仮想ＮＵＭＡノードを前記計算機システムの第２のＮＵＭＡノードへ移動させるステップと、を含む請求項１記載の方法。
【請求項３】

前記ゲストメモリー容量を調整するステップが更に、
ゲストメモリーの少なくとも１つのメモリーブロックを前記特定の仮想ＮＵＭＡノードからデコミットするステップと、
前記ゲストメモリーの少なくとも１つのデコミットされたメモリーブロックを第２の仮想ＮＵＭＡノードへコミットするステップと、を含むことを特徴とする請求項１記載の方法。
【請求項４】
前記ゲストメモリー容量を調整するステップが更に、
前記特定の仮想ＮＵＭＡノードの少なくとも１つのゲストメモリーブロックがシステムメモリーと分離されていることを決定するステップと、
前記ゲストメモリーの少なくとも１つのメモリーブロックをシステムメモリーの少なくとも１つのメモリーブロック上へマッピングするステップと、を含むことを特徴とする請求項１記載の方法。
【請求項５】
更に、
前記特定の仮想ＮＵＭＡノードを前記計算機システムの第１のＮＵＭＡノード上へマッピングするステップと、
前記特定の仮想ＮＵＭＡノードを前記計算機システムの第２のＮＵＭＡノード上へ移動させるステップと、を含む請求項１記載の方法。
【請求項６】
更に、
仮想プロセッサーを特定の前記仮想ＮＵＭＡノードに追加するステップを含む請求項１記載の方法。
【請求項７】
更に、
前記仮想マシンの仮想プロセッサーと、論理プロセッサーに割り当てられた前記仮想プロセッサーと、ＮＵＭＡノードに割り当てられ前記論理プロセッサーと、及び前記複数の仮想ＮＵＭＡノードの内の１つに割り当てられた前記仮想プロセッサーと、を実行するためのリクエストを受信するステップと、
前記論理プロセッサーが前記仮想プロセッサーを実行不可能なことを決定するステップと、
前記仮想プロセッサーを実行する第２の論理プロセッサーを選択するステップであって前記第２の論理プロセッサーが、第２のＮＵＭＡノードから提供されているもの、を含む請求項１記載の方法。
【請求項８】
計算機システムであって、
仮想マシンを実行するための回路であって前記仮想マシンが、複数の仮想非一様メモリ・アーキテクチャ（ＮＵＭＡ：Non-Uniform Memory Architecture）ノードを含むトポロジーを有しており、前記仮想マシンの前記トポロジーが、前記計算機システムの物理トポロジーから独立して生成されるものと、
前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するための回路と、
前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードそれぞれにおける前記メモリー圧力に基づいて、前記複数の仮想ＮＵＭＡノードの少なくとも１つに割り当てられるゲストメモリーを調整するための回路と、を含む計算機システム。
【請求項９】
更に、
前記仮想マシンを第２の計算機システムへ送信するための回路を含む請求項８記載の計算機システム。
【請求項１０】

更に、
前記複数の仮想ＮＵＭＡノードの内の第１の仮想ＮＵＭＡノードを前記計算機システムの第１のＮＵＭＡノード上へマッピングするための回路と、
前記複数の仮想ＮＵＭＡノードの内の第２の仮想ＮＵＭＡノードを前記計算機システムの前記第１のＮＵＭＡノード上へマッピングするための回路と、を含む請求項８記載の計算機システム。
【請求項１１】
更に、
前記複数の仮想ＮＵＭＡノードの内の第１の仮想ＮＵＭＡノードを前記計算機システムの第１のＮＵＭＡノード上へマッピングするための回路と、
前記複数の仮想ＮＵＭＡノードの内の第２の仮想ＮＵＭＡノードを前記計算機システムの第２のＮＵＭＡノード上へマッピングするための回路と、を含む請求項８記載の計算機システム。
【請求項１２】
更に、
前記第２の仮想ＮＵＭＡノードにおけるメモリー圧力が所定の値よりも大きいことを決定するための回路と、
前記第２の仮想ＮＵＭＡノードを前記計算機システムの第２のＮＵＭＡノードへ移動させるための回路と、を含む請求項１０記載の計算機システム。
【請求項１３】
更に、
前記第２の仮想ＮＵＭＡノードのメモリー圧力が所定の値よりも大きいことを決定するための回路と、
前記仮想マシンの前記第２の仮想ＮＵＭＡノードを前記計算機システムの前記第１のＮＵＭＡノードへ移動させるための回路と、を含む請求項１１記載の計算機システム。
【請求項１４】
プロセッサー実行可能命令を含む計算機可読記憶媒体であって、
第１の仮想マシンを実行するための命令であって前記仮想マシンが、複数の仮想非一様メモリ・アーキテクチャ（ＮＵＭＡ：Non-Uniform Memory Architecture）ノードを含むトポロジーを有しており、前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードそれぞれが、仮想プロセッサー及びゲスト物理アドレスを含んでおり、前記仮想マシンの前記トポロジーが、計算機システムの物理トポロジーから独立して生成されるものと、
前記第１の仮想マシンのランタイム実行中に、増設の仮想プロセッサーを前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードに追加するための命令と、を含む前記計算機可読記憶媒体。
【請求項１５】
更に、
前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するための命令と、
前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードそれぞれにおける前記メモリー圧力に基づいて、複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードの少なくとも１つに割り当てられるゲストメモリーを調整するための命令と、を含む請求項１４記載の計算機可読記憶媒体。
【請求項１６】
更に、
前記複数の仮想ＮＵＭＡノードの内の仮想ＮＵＭＡノードから仮想プロセッサーを移動するための命令を含む請求項１４記載の計算機可読記憶媒体。
【請求項１７】
更に、
前記複数の仮想ＮＵＭＡノードに関するＮＵＭＡ比をゲストオペレーティングシステム

へ報告するための命令を含む請求項１４記載の計算機可読記憶媒体。
【請求項１８】
更に、
前記仮想マシンを第１のＮＵＭＡノードから複数のＮＵＭＡノードへ移動させるための命令を含む請求項１４記載の計算機可読記憶媒体。
【請求項１９】
前記ゲストメモリーを調整するための前記命令が更に、
前記複数の仮想ＮＵＭＡノードの内の第１の仮想ＮＵＭＡノードの現在の前記メモリー圧力が目標の閾値よりも低いという決定に基づいて、前記第１の仮想ＮＵＭＡノードからメモリーをデコミットするための命令を含むことを特徴とする請求項１５記載の計算機可読記憶媒体。
【請求項２０】
前記ゲストメモリーを調整するための前記命令が更に、
ゲストオペレーティングシステムの現在のメモリー圧力が目標の閾値よりも大きいという決定に基づいて、前記複数の仮想ＮＵＭＡノードの内の第１の仮想ＮＵＭＡノードへメモリーをコミットするための命令を含むことを特徴とする請求項１５記載の計算機可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、仮想化技術に関し、具体的には、仮想マシン用のメモリーアーキテクチャーに関する。
【背景技術】
【０００２】
[0001]仮想化技術は、複数のパーティション間においてハードウェア資源を共有することを可能にし、各パーティションがゲストオペレーティングシステムをホスティングすることを可能にする。通常、仮想マシン技術は、サーバーを統合化し、それらの移植性を増大させるために利用され得る。仮想マシンがより大きくなりそれらの作業負荷が増大するにつれて、それらをある計算機システムから別の計算機システムへ容易に統合及び／又は移動させる機能は、より困難になっている。したがって、より大きな仮想マシンを統合化及び／又は移動させる能力を増大する技法が望まれている。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
本発明の目的は、仮想マシン用の仮想ＮＵＭＡアーキテクチャーを達成し、仮想ＮＵＭＡノードにおけるメモリーを調整するための技法を提供することである。
【課題を解決するための手段】
【０００４】
[0002]本開示の実施形態例は、一方法を記述している。この例において、本方法はこれらに限定しないが、仮想マシンをインスタンス化するためのリクエストを受信するステップであってリクエストが、仮想マシンに関する特性を含んでいるものと、前記特性に基づいて仮想マシンに対する仮想ＮＵＭＡノードトポロジーを選択するステップであって仮想ＮＵＭＡノードトポロジーが、複数の仮想ＮＵＭＡノードを含んでいるものと、計算機システム上の仮想マシンをインスタンス化するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含んでいるものと、特定の複数の仮想ＮＵＭＡノードにおけるメモリー圧力に基づいて調整するステップであってゲストメモリー容量が、特定の仮想ＮＵＭＡノードに割り当てられているものと、を含む。前述のものに加えて別の態様が、本開示の一部を形成する請求項、図面、及びテキストに記述される。
【０００５】
[0003]本開示の実施形態例は、一方法を記述している。この例において、本方法はこれ

らに限定しないが、仮想マシンを実行するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含むトポロジーを有しており、前記仮想マシンのトポロジーが、計算機システムの物理トポロジーから独立して生成されるものと、複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するステップと、複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力に基づいて調整するステップであってゲストメモリーが、複数の仮想ＮＵＭＡノードの少なくとも１つに割り当てられているものと、を含む。前述のものに加えて別の態様が、本開示の一部を形成する請求項、図面、及びテキストに記述される。

【０００６】

[0004]本開示の実施形態例は、一方法を記述している。この例において、本方法はこれらに限定しないが、第１の仮想マシンを実行するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含むトポロジーを有しており、複数の仮想ＮＵＭＡノードそれぞれが、仮想プロセッサー及びゲスト物理アドレスを含んでおり、仮想マシントポロジーが、計算機システムの物理トポロジーから独立して生成されるものと、増設の仮想プロセッサーを複数の仮想ＮＵＭＡノードに追加するステップと、を含む。前述のものに加えて別の態様が、本開示の一部を形成する請求項、図面、及びテキストに記述される。

【０００７】

[0005]本開示の１つ以上の様々な態様は、本明細書に参照されている本開示の態様に作用するための回路及び／又はプログラムであって回路及び／又はプログラムが、システム設計者の設計選択に従って本明細書に参照されている態様に効果的になるように構成されたハードウェア、ソフトウェア、及び／又はファームウェアの実際の組み合わせのいずれかであり得るものを含むが、これらに制限しないことが当業者によって十分に理解されよう。

【０００８】

[0006]前述のものは概要であって、かくして必然的に、詳細の簡素化、一般化、及び省略を含む。当業者は、概要が例示に過ぎず、任意の方法に制限することを意図しないことを十分に理解されよう。

【図面の簡単な説明】

【０００９】

【図１】[0007]　本開示の態様が実装され得る計算機システムの例を表している。

【図２】[0008]　本開示の態様を実施するための動作環境を表している。

【図３】[0009]　本開示の態様を実施するための動作環境を表している。

【図４】[0010]　本開示の実施形態においてメモリーがどのように配置され得るか表している。

【図５】[0011]　本開示の態様を実施する動作環境の例を表している。

【図６】[0012]　本開示の態様を実施する動作環境の例を表している。

【図７】[0013]　本開示の態様を実施する動作環境の例を表している。

【図８】[0014]　本開示の態様を例示するブロック図の例を表している。

【図９】[0015]　本開示の態様を実施するための動作手順を表している。

【図１０】[0016]　図９の動作手順の代替実施形態を表している。

【図１１】[0017]　本開示の態様を実施するための動作手順を表している。

【図１２】[0018]　図１１の動作手順の代替実施形態を表している。

【図１３】[0019]　図１２の動作手順の代替実施形態を表している。

【図１４】[0020]　図１２の動作手順の代替実施形態を表している。

【図１５】[0021]　本開示の態様を実施するための動作手順を表している。

【図１６】[0022]　図１５の動作手順の代替実施形態を表している。

【図１７】[0023]　図１６の動作手順の代替実施形態を表している。

【発明を実施するための形態】

【００１０】

[0024]実施形態は１つ以上の計算機上で実行し得る。図１及び以下の論述は、本開示が実装され得る適切な計算環境の簡潔な概説を提供することを意図している。当業者は、図

１の計算機システムがいくつかの実施形態において、計算機システム（２００）、（３００）、（６００）、及び（７００）を達成可能なことを十分に理解できよう。これらの実施形態例において、計算機システムは、図１に記述されたコンポーネント及び本開示の態様を例示化するように構成される回路のいくつか又はすべてを含み得る。

【００１１】

[0025]本開示を介し使用されている用語「回路」は、ハードウェア割り込みコントローラー、ハードドライブ、ネットワークアダプター、グラフィックスプロセッサー、ハードウェアベースのビデオ／オーディオコーデックのようなハードウェアコンポーネント、及びそのようなハードウェアを操作するために使用されるファームウェア／ソフトウェアを含み得る。同一又は別の実施形態において、用語「回路」は、ファームウェア又は一定の方法で設定される切り換えによって機能（単数又は複数）を実行するように構成されるマイクロプロセッサーを含み得る。同一又は別の実施形態例において、用語「回路」は、１つ以上の論理プロセッサー、例えば１つ以上のコアであるマルチコア汎用演算処理装置を含み得る。この例において論理プロセッサー（単数又は複数）は、メモリー、例えば、ＲＡＭ、ＲＯＭ、ファームウェア、及び／又は仮想メモリーからロードされる機能（単数又は複数）を実行するように作動可能なロジックを具体化しているソフトウェア命令によって、構成され得る。回路がハードウェア及びソフトウェアの組み合わせを含む実施形態例において、実装者は、ロジックを具体化しているソースコードを書き出し得、その後、論理プロセッサーによって処理され得る計算機読み出し可能コードにコンパイルされる。最先端技術がハードウェア、ソフトウェア、又はハードウェア／ソフトウェアの組み合わせの間でほとんど差異がない点へ発展していることを当業者は十分に理解し得るので、機能を達成するハードウェア対ソフトウェアの選択は、単なる設計選択に過ぎない。かくして、当業者は、ソフトウェアプロセスが同等のハードウェア構造に変換され得ることと、ハードウェア構造がそれ自体、同等のソフトウェア処理に変換され得ることと、を十分に理解し得るので、ハードウェア実装対ソフトウェア実装の選択は、設計選択の１つであって実装者に任せられている。

【００１２】

[0026]ここで図１を参照すると、例示的な汎用計算システムが表されている。汎用計算システムは、従来の計算機（２０）などを含み得、論理プロセッサー（２１）、システムメモリー（２２）、システムメモリーを含む様々なシステムコンポーネントを論理プロセッサー（２１）と接続する及びシステムバス（２３）を含んでいる。システムバス（２３）は、いくつかのタイプのバス構造のいずれかであり得、メモリーバス、又はメモリーコントローラー、周辺機器用バス、様々なバスアーキテクチャのいずれかを使用するローカルバスを含んでいる。システムメモリーは、読み出し専用メモリー（ＲＯＭ）（２４）及びランダムアクセスメモリー（ＲＡＭ）（２５）を含み得る。始動中など、計算機（２０）内部のエレメント間に情報を送信する支援をする基本的ルーチンを含んでいる基本入力／出力システム（ＢＩＯＳ）（２６）は、ＲＯＭ（２４）にストアされている。計算機（２０）は更に、（図示されていない）ハードディスクから読み込むか又はそれに書き出すためのハードディスクドライブ（２７）、取り外し可能磁気ディスク（２９）から読み出すか又は書き出すための磁気ディスクドライブ（２８）、及びＣＤ－ＲＯＭ又はその他の光学式媒体のような取り外し可能光学式ディスク（３１）から読み出すか又はそれに書き込むための光学式ディスクドライブ（３０）を含み得る。ハードディスクドライブ（２７）、磁気ディスクドライブ（２８）、及び光ディスクドライブ（３０）がそれぞれ、ハードディスクドライブインターフェース（３２）、磁気ディスクドライブインターフェース（３３）、及び光学式ドライブインターフェース（３４）によってシステムバス（２３）に関連付けられるように示されている。ドライブ及びそれらの関連する計算機可読記憶媒体は、計算機（２０）に対する命令、データ構造、プログラムモジュール、及びその他のデータの不揮発性計算機可読記憶装置を提供する。本明細書に説明される例示的環境は、ハードディスク、取り外し可能磁気ディスク（２９）、及び取り外し可能光学式ディスク（３１）を使用しているが、磁気カセット、フラッシュメモリーカード、デジタルビデオ

ディスク、ベルヌーイカートリッジ、ランダムアクセスメモリー（ＲＡＭ）、読み出し専用メモリー（ＲＯＭ）などのような計算機によってアクセス可能なデータをストアし得る別のタイプの計算機可読記憶媒体もまた例示的動作環境において使用され得ることが当業者によって十分に理解される必要がある。通常、実施形態の中にはそのような計算機可読記憶媒体が、本開示の態様を具体化するプロセッサー実行可能命令をストアするために使用され得るものもいくつかある。

【００１３】

[0027]多くのプログラムモジュールは、ハードディスク、磁気ディスク（２９）、光ディスク（３１）、ＲＯＭ（２４）、又はＲＡＭ（２５）上にストアされ得、オペレーティングシステム（３５）、１つ以上のアプリケーションプログラム（３６）、その他のプログラムモジュール（３７）、及びプログラムデータ（３８）を含んでいる。ユーザーは、キーボード（４０）及びポインティングデバイス（４２）のような入力装置を介し、コマンド及び情報を計算機（２０）へ入力し得る。その他の（図示されていない）入力装置は、マイクロフォン、ジョイスティック、ゲームパッド、衛星放送受信機、スキャナーなどを含み得る。これら及びその他の入力装置は多くの場合、システムバスに接続されるシリアルポートインターフェース（４６）を介し、論理プロセッサー（２１）に接続されるが、しかしながらパラレルポート、ゲームポート又は普遍的なシリアルバス（ＵＳＢ）のような別のインターフェースによって接続される。ディスプレイ（４７）又はその他のタイプの表示装置もビデオアダプター（４８）のようなインターフェースを介しシステムバス（２３）に接続され得る。ディスプレイ（４７）に加えて計算機は典型的に、スピーカー及びプリンターのような別の（図示されていない）周辺出力装置を含む。図１の例示的システムは、ホストアダプター（５５）、小型コンピューター用周辺機器インターフェース（ＳＣＳＩ）バス（５６）、及びＳＣＳＩバス（５６）に接続される外部記憶装置（６２）も含む。

【００１４】

[0028]計算機（２０）は、リモートコンピューター（４９）のような１つ以上のリモートコンピューターとの論理接続を利用するネットワーク環境において作動し得る。リモートコンピューター（４９）は、別の計算機、サーバー、ルーター、ネットワークＰＣ、ピア装置又はその他の一般的ネットワークノードであり得、典型的に、前述した計算機（２０）に関連するエレメントの多く又はすべてを含み得るが、メモリー記憶装置（５０）だけが図１に例示されている。図１に表された論理接続は、ローカルエリアネットワーク（ＬＡＮ）（５１）及び広域ネットワーク（ＷＡＮ）（５２）を含み得る。そのようなネットワーク環境は、オフィス、企業規模コンピューターネットワーク、イントラネット、及びインターネットにおいて一般的である。

【００１５】

[0029]ＬＡＮネットワーク環境において利用されるとき、計算機（２０）は、ネットワークインターフェース又はアダプター（５３）を介しＬＡＮ（５１）へ接続され得る。ＷＡＮネットワーク環境において利用されるとき、計算機（２０）は、典型的に、インターネットのような広域ネットワーク（５２）を介した通信を確立するためのモデム（５４）又はその他の手段を含み得る。内蔵又は外付けがあり得るモデム（５４）が、シリアルポートインターフェース（４６）を介しシステムバス（２３）へ接続され得る。ネットワーク環境において、計算機（２０）又はその一部に関連し表されたプログラムモジュールが、リモートメモリー記憶装置にストアされ得る。示されたネットワーク接続が例示的であって、計算機間において通信リンクを確立する別の手段が使用され得ることを十分に理解されよう。更に、本開示の多くの実施形態がコンピューター化されたシステムに対し特に適切であるように描かれているが、本書における開示をそのような実施形態に限定することは意図されていない。

【００１６】

[0030]ここで図２及び図３を参照すると、それらは計算機システムの高水準のブロック図を表している。図面によって示されているように、計算機システム（２００）は、例え

ば、記憶装置（２０８）、ハードドライブ、ネットワークインターフェースコントローラー（ＮＩＣ）（２１０）、グラフィックカード（２３４）、少なくとも１つの論理プロセッサー（２１２）、ランダムアクセスメモリー（ＲＡＭ）（２１４）のような物理ハードウェアデバイスを含み得る。計算機システム（２００）は、図１の計算機（２０）と同様のコンポーネントも含み得る。１つの論理プロセッサーが例示されているが、別の実施形態において、計算機システム（２００）は、複数の論理プロセッサー、例えば、プロセッサーあたり複数の実行コア及び／又は複数の実行コアをそれぞれ有する複数のプロセッサー、を有し得る。図２の説明を続けると、表されているものは、当技術分野において、仮想マシンモニターとしても参照され得るハイパーバイザー（２０２）である。表された実施形態において、ハイパーバイザー（２０２）は、計算機システム（２００）のハードウェアへのアクセスを制御し、仲裁するための実行可能命令を含む。概してハイパーバイザー（２０２）は、子パーティション１から子パーティションＮ（Ｎは１より大きい整数）のようなパーティションと呼ばれる実行環境を生成し得る。実施形態において、子パーティションは、ハイパーバイザー（２０２）によって支援される分離単位と考えられ得、すなわち、子パーティションそれぞれが、ハイパーバイザー（２０２）及び／又は親パーティションの制御下にある一連のハードウェア資源、例えば、メモリー、デバイス、論理プロセッサーサイクルなどへマッピングされ得る。実施形態において、ハイパーバイザー（２０２）は、スタンドアロンソフトウェア製品、オペレーティングシステムの一部、マザーボードのファームウェア内埋め込み、専用集積回路、又はその組み合わせであり得る。

【００１７】

[0031]表された例において、計算機システム（２００）は、オープンソースコミュニティにおいて、ドメイン０としても考えられ得る親パーティション（２０４）を含む。親パーティション（２０４）は、オープンソースコミュニティにおいて、バックエンドドライバーとしても知られている仮想化サービスプロバイダー（２２８）（ＶＳＰ）（複数）を使用することによって、子パーティション（１～Ｎ）を実行しているゲストオペレーティングシステムへリソースを提供するように構成され得る。このアーキテクチャー例において、親パーティション（２０４）は基本ハードウェアへのアクセスを開閉し得る。概してＶＳＰ（２２８）は、オープンソースコミュニティにおいて、フロントエンドドライバーとしても知られている仮想化サービスクライアント（ＶＳＣ）を介し、ハードウェア資源に対するインターフェースを多重化するために使用され得る。子パーティションそれぞれは、ゲストオペレーティングシステム（２２０～２２２）が管理し得、その上で実行するスレッドをスケジューリングし得る仮想プロセッサー（２３０～２３２）のような１つ以上の仮想プロセッサーを含み得る。通常、仮想プロセッサー（２３０～２３２）は、実行可能命令であって、特定のアーキテクチャーを有する物理プロセッサー表現を提供する関連状態情報である。例えば、１つの仮想マシンは、インテルｘ８６プロセッサーの特性を有する仮想プロセッサーを有し得るが、一方で別の仮想プロセッサーは、パワーＰＣプロセッサーの特性を有し得る。この例において、仮想プロセッサーは、仮想プロセッサーを達成する命令が論理プロセッサーによって支援されるように、計算機システムの論理プロセッサーへマッピングされ得る。かくして、これらの実施形態例において、複数の仮想プロセッサーが、例えば、別の論理プロセッサーがハイパーバイザー命令を実行している間、同時に実行し得る。一般的に言えば、図面によって例示されるように、パーティションにおける仮想プロセッサー、様々なＶＳＣ、及びメモリーの組み合わせが、仮想マシン（２４０）又は（２４２）のような仮想マシンであると考えられ得る。

【００１８】

[0032]通常、ゲストオペレーティングシステム（２２０～２２２）は、例えば、マイクロソフト（登録商標）、アップル（登録商標）、オープンソースコミュニティから提供されるオペレーティングシステムのような任意のオペレーティングシステムを含み得る。ゲストオペレーティングシステムは、ユーザー／カーネル動作モードを含み得、スケジューラー、メモリーマネージャーなどを含み得るカーネルを有し得る。ゲストオペレーティングシステム（２２０～２２２）それぞれが、その上にストアされる電子商取引サーバー、

電子メールサーバーのようなアプリケーションを有し得るファイルシステムと、ゲストオペレーティングシステム自体と、を関連付けている。ゲストオペレーティングシステム（２２０～２２２）は、仮想プロセッサー（２３０～２３２）上で実行するスレッドをスケジューリングし得、そのようなアプリケーションのインスタンスが達成され得る。

【００１９】

[0033]ここで図３を参照すると、それは使用され得る代替アーキテクチャーを例示している。図３は、図２のそれと同様のコンポーネントを表しているが、しかし、この実施形態例において、ハイパーバイザー（２０２）は仮想化サービスプロバイダー（２２８）及びデバイスドライバー（２２４）を含み得、親パーティション（２０４）は設定ユーティリティ（２３６）を含み得る。このアーキテクチャーにおいて、ハイパーバイザー（２０２）は、図２のハイパーバイザー（２０２）と同一か又は同様の機能を実行し得る。図３のハイパーバイザー（２０２）は、スタンドアロンのソフトウェア製品、オペレーティングシステムの一部、マザーボードのファームウェア内部に埋め込まれているか、又はハイパーバイザー（２０２）の一部が専用集積回路によって達成され得る。この例において、親パーティション（２０４）は、ハイパーバイザー（２０２）を構成するために使用され得る命令を有し得るが、しかし、ハードウェアアクセスリクエストは、親パーティション（２０４）へ渡される代わりにハイパーバイザー（２０２）によって処理され得る。

【００２０】

[0034]ここで図４を参照すると、それは仮想マシンを含む実施形態において、メモリーがどのように配置され得るか例示している。例えば、計算機システム（２００）のような計算機システムは、メモリーアドレスを有しているＲＡＭ（２１４）を有し得る。システム物理メモリーアドレスを仮想マシンへ報告する代わりに、ハイパーバイザー（２０２）は、システム物理アドレス、例えば、ゲスト物理アドレス（ＧＰＡ）に関する別のアドレスをゲストオペレーティングシステムのメモリーマネージャーへ提示し得る。ゲストオペレーティングシステムがその後、ゲスト物理アドレスを操作し得、ハイパーバイザー（２０２）がＧＰＡ及びＳＰＡによる関係を維持する。図面によって示されるように、実施形態において、ＧＰＡ及びＳＰＡは、メモリーブロックの中へ配置され得る。概してメモリーブロックは、１つ以上のメモリーページを含み得る。ＧＰＡとＳＰＡとの間の関係は、「Ｅｎｈａｎｃｅｄ　Ｓｈａｄｏｗ　Ｐａｇｅ　Ｔａｂｌｅ　Ａｌｇｏｒｉｔｈｍｓ」と題する米国特許出願Ｎｏ．１１／１２８，６６５に記載されているようなシャドウページテーブルによって維持され得、その内容をすべて参照として本明細書に組み込む。作動中、ゲストオペレーティングシステムがブロック１のＧＰＡにデータをストアしたとき、データは、実際にはシステム上のブロック６のような異なるＳＰＡにストアされ得る。

【００２１】

[0035]手短に述べると図５は、本開示の態様を実施するための動作環境を表している。例えば、多くの計算機システム（５０４～５１０）がデータセンター（５００）に共に接続され得る（４つの計算機システムが表されているが、当業者はデータセンター（５００）が、より多いか又はより少ない計算機システムを含み得ることを十分に理解できよう）。表されている計算機システムは、異なるトポロジーを有し得、その上、それらは、異なる特性、例えば、異なるＲＡＭ数、異なるＲＡＭ速度、異なる論理プロセッサー数、及び／又は異なる速度を持つ論理プロセッサーを有し得る。

【００２２】

[0036]管理システム（５０２）は、図１の計算機システム（２０）、及び／又は計算機システム（２００）、（３００）、（６００）、又は（７００）と同様のコンポーネントを有し得る。すなわち、実施形態において、管理システム（５０２）は、図６又は図７に関し後述される対象項目を含む計算機システムであり得る。

【００２３】

[0037]図面の概要を続けると、図６は、完全対称型マルチプロセッシングトポロジー（ＳＭＰ）又は「フラット」トポロジーを有する計算機システム（６００）を表している。通常、ＳＭＰは、単一の共有メモリーに接続される複数のプロセッサーを含むコンピュー

ターアーキテクチャーである。この手続きにおいては、メモリーコントローラー（６０２）が、メモリーへのデータフロー及びメモリーからのデータフローを管理し得る。メモリーアクセスは、論理プロセッサー（２１２Ａ～Ｆ）それぞれに対し一様であり得、論理プロセッサーそれぞれは、メモリー範囲全体、すなわち、システム物理アドレス（６２２～６３２）をアクセスし得る。このトポロジーは、比較的少ない数のプロセッサーを用いた計算機システムに対し十分に動作するが、しかしながら計算機システムは多くのプロセッサーを含んでいて、すべてが共有メモリーバスへのアクセスを求めて競合し、システム性能が低下し得る。その上、計算機システムの複雑さが、著しく増加し、次々に１プロセッサーあたりの価格をつり上げる。

【００２４】

[0038]手短に述べると計算機システム（６００）は、計算機（２００）又は（３００）と同一か又は同様のコンポーネントを含み得る。図面によって示されるように、計算機システム（６００）は、ＲＡＭ（２１４）へのアクセスを開閉するメモリーコントローラー（６０２）を介し連結された複数の論理プロセッサー（２１２Ａ～２１２Ｆ）を有し得る（６つの論理プロセッサーが表されているが、計算機システムはより多いか又は少ないものを有し得る）。前述のものと同様に、論理プロセッサー（２１２Ａ～２１２Ｆ）それぞれは、異なる特性、例えば、クロック速度、キャッシュサイズなどを有し得る。この手続きにおいては、メモリーコントローラー（６０２）が、ＲＡＭ（２１４）へのデータフロー及びＲＡＭ（２１４）からのデータフローを管理し得る。

【００２５】

[0039]ハイパーバイザー（２０２）がインスタンス化され得、それが計算機システム（６００）のハードウェアを制御し得る。ハイパーバイザー（２０２）が１つ以上の仮想マシン（２４０～２４２）を管理し得、それぞれは、仮想ＮＵＭＡノード（６０６～６１２）のような仮想ＮＵＭＡノードを有し得る。仮想ＮＵＭＡノード（６０６～６１２）が使用され得、ゲストアプリケーション又はゲストオペレーティングシステム（２２０）及び（２２２）のようなゲストオペレーティングシステムへ仮想トポロジーを報告することによって、仮想マシンのリソースを統合化し得る。図面によって示されるように、仮想ＮＵＭＡノード（６０６～６１２）それぞれは、１つ以上の仮想プロセッサー（２３０Ａ～Ｄ、２３２Ａ～Ｄ）、及びゲスト物理アドレス（６１４～６１６）及び（６１８～６２０）を有し得る。通常、ハイパーバイザー（２０２）は、１つ以上の論理プロセッサーを有する仮想ＮＵＭＡノード（６０６～６１２）それぞれと、ＲＡＭ（２１４）からのシステム物理アドレスと、を支援し得る。すなわち、ハイパーバイザー（２０２）は、仮想プロセッサースレッドを実行するために使用され得る理想的なプロセッサーとして１つ以上の論理プロセッサーを設定し得る。

【００２６】

[0040]手短に述べると図７は、ＮＵＭＡノード（７０２～７０６）を含むトポロジーを有する計算機システム（７００）を表している。ＮＵＭＡノードを有する計算機システムは、通常、より小さな計算機システムから作り上げられた計算機として考えられ得る。この例において、ＮＵＭＡノードそれぞれ（６０６～６１２）は、１つ以上の論理プロセッサー及びローカルメモリーを含み得る。ＮＵＭＡノードの内部メモリーは、ローカルメモリーであると考えられ、別のＮＵＭＡノードの内部メモリーは、ノード内部のプロセッサーだけが同一のメモリーバスと接続され得るので、リモートメモリーであると考えられる。ＮＵＭＡノードは、キャッシュ一貫性ドメイン相互接続によって相互に接続され、１つのＮＵＭＡノード内のプロセッサーは、首尾一貫した方法で別のＮＵＭＡノード内のメモリーをアクセス可能にする。かくして、システム物理アドレス（６２２～６３２）は、プロセッサーそれぞれに関し一様である。すなわち言い換えると、システム物理アドレス２０，０００は計算機システムにおいて、すべてのプロセッサーに対し同一である。相違は、いくつかのプロセッサーに関し、メモリーアドレス２０，０００が、例えば、それらのＮＵＭＡノード内部のローカルメモリーアドレスであって、別のプロセッサーに対するメモリーアドレス２０，０００が、例えば、それらのＮＵＭＡノードの外部、リモートであ

ることである。通常、ローカルメモリーは、リモートメモリーよりも高速にアクセスされ得、ローカルアクセス時間対リモートアクセス時間の間の関係はＮＵＭＡ比として参照される。ＮＵＭＡ比１対２は、特定のリモートシステム物理アドレスをアクセスするプロセッサーサイクル数が、ローカルシステム物理アドレスよりも２倍かかることを意味する。ＮＵＭＡは、任意の１つのメモリーバス上のプロセッサー数を制限することによって、ＳＭＰシステムによってもたらせられるボトルネックを軽減し、通常、同一の論理プロセッサー数を有するＳＭＰ計算機システムよりも高価にならない。

【００２７】

[0041]計算機システム（７００）は、計算機（２００）又は（３００）と同一か又は同様のコンポーネントを含み得る。図面によって示されるように、この動作環境において、計算機システム（７００）は、相互接続（７０８）によって接続された３つのＮＵＭＡノード（７０２～７０６）を含む（けれども計算機は多いか又は少ないものを有し得る）。図面によって例示されるように、ＮＵＭＡノードそれぞれの内部のプロセッサー数は可変であり得、ノードそれぞれはそれ自身のＲＡＭを有し得る。

【００２８】

[0042]図７と同様にハイパーバイザー（２０２）は、計算機システム（７００）のハードウェアを制御し得る。ゲストオペレーティングシステム又はモノリスアプリケーションがブートしたとき、それが前述したものと同様の仮想マシン（２４０）及び（２４２）のトポロジーを検出し得る。仮想ＮＵＭＡノード（６０６～６１２）それぞれが、仮想プロセッサースレッドを実行するために使用され得る同一のＮＵＭＡノードから１つ以上の理想的なプロセッサー及びシステム物理アドレスを割り当てられ得る。

【００２９】

[0043]計算機システム（６００）及び（７００）が２つの仮想マシン（２４０）及び（２４２）を含んでいるように表されているが、別の実施形態において、それらは、より多いか又はより少ない仮想マシンを実行し得る。その上、仮想マシンそれぞれが２つの仮想ＮＵＭＡノードを有しているように表されているが、別の実施形態において、仮想マシンは、より多いか又はより少ない仮想ＮＵＭＡノードを有し得る。仮想ＮＵＭＡノードも２つの仮想プロセッサーを有するように表されているが、別の実施形態において、仮想ＮＵＭＡノードは、より多いか又はより少ない仮想プロセッサーを有し得る。更に、仮想ＮＵＭＡノードそれぞれは、別の仮想ＮＵＭＡノードと異なるトポロジーを有し得、例えば、ある仮想ＮＵＭＡノードは４つの仮想プロセッサー及び８ギガバイトＲＡＭを有し得るが、別の仮想ＮＵＭＡノードは２つの仮想プロセッサー及び４ギガバイトＲＡＭとを有し得る。

【００３０】

[0044]図８は、本開示の態様において使用され得る環境のブロック図を表している。図面によって示されるように、ダイナミックメモリー仮想化サービスプロバイダー（ＤＭＶＳＰ）（８０２）として知られ得る、仮想マシンに割り当てられるメモリーを管理するコンポーネントが例示されていて、仮想ＮＵＭＡノードにアクセス可能なメモリー容量を調整するために使用され得る。図面によって示されるように、ＤＭＶＳＰ（８０２）は、仮想化サービスクライアント、すなわち、ダイナミックメモリー仮想化サービスクライアント（ＤＭＶＳＣ）（８０４）及び／又は（８０６）として知られ得る１つ以上のバルーニングドライバーに関連付けられ得る（仮想ＮＵＭＡノードあたり１つのＤＭＶＳＣが表されているが、別の実施形態において、１パーティションあたり１つのＤＭＶＳＣが使用され得る）。概してＤＭＶＳＣ（８０４）及び／又は（８０６）は、ＤＭＶＳＰ（８０２）によって使用され得る仮想ＮＵＭＡノードのメモリーを調整するための情報を提供し得、ＤＭＶＳＣそれぞれも、それが関連付けられる仮想ＮＵＭＡノードからメモリーをコミット及びデコミットする支援をし得る。ＤＭＶＳＣ（８０４）、（８０６）、及びＤＭＶＳＰ（８０２）は、「Ｐａｒｔｉｔｉｏｎ　Ｂｕｓ」と題する米国特許出願Ｎｏ．１１／１２８，６４７に記載されている仮想化バスを介し通信し得、その内容全体をすべて参照として組み込む。その上、更にＤＭＶＳＣ及びＤＭＶＳＰの態様は「Ｄｙｎａｍｉｃ　Ｖｉ

ｒｔｕａｌ　Ｍａｃｈｉｎｅ　Ｍｅｍｏｒｙ　Ｍａｎａｇｅｍｅｎｔ」と題する米国特許出願Ｎｏ．１２／３４５，４６９に記載されていて、その内容全体をすべて参照として組み込む。

【００３１】

[0045]図８の説明を続けると、本システムは、ワーカープロセス（８１２）を含み得、子パーティション（単数又は複数）を管理可能に表されている。ワーカープロセス（８１２）は、メモリーを子パーティションに割り当て得る仮想基盤ドライバー（ＶＩＤ）（８１０）と連動して作動し得る。例えば、ＶＩＤ（８１０）は、ゲスト物理アドレスとシステム物理アドレスとの間の関係を確立及び解除し得る。図８は、メモリーマネージャー（８０８）を含み得るゲストオペレーティングシステム（２２０）のような、ゲストオペレーティングシステムを含み得るパーティションも表している。通常、メモリーマネージャー（８０８）は、それらのリクエスト時、メモリーをアプリケーションに割り当て、それがもはやアプリケーションによって必要とされないとき、メモリーを解放し得る。

【００３２】

[0046]以下は、プロセスの実装を表している一連の流れ図である。理解を容易にするため、流れ図は、最初の流れ図が「大画像」の観点を介する実装を表し、その後の流れ図が更なる追加及び／又は詳細を提供するように統合化されている。更に当業者は、点線によって表された動作手順が、任意に考えられることを十分に理解されよう。

【００３３】

[0047]ここで図９に移ると、それは動作（９００～９１０）を含む本開示の態様を実施するための動作手順を表している。動作手順は、動作（９００）で開始し、動作（９０２）は、仮想マシンをインスタンス化するためのリクエストを受信するステップであってリクエストが、仮想マシンに関する特性を含んでいるもの、を例示していている。例えば、図６又は図７を参照すると、ハイパーバイザー（２０２）は、仮想マシン（２４０）などの仮想マシンを作成するためのリクエストを受信し得る例えば、リクエストは、管理システム（５０２）の図２又は図３の親パーティション（２０４）などから受信され得る、リクエストは、新しい仮想マシンに対するものであり得るか又はそれは、前にセーブされた仮想マシンをインスタンス化するためのリクエストであり得る。仮想マシン（２４０）が新しい仮想マシンであるとき、仮想マシンの特性、例えば、仮想マシンに割り当てられるＲＡＭ数、仮想プロセッサー数、又は仮想マシンが有する必要があるＩ／Ｏ装置のタイプ、が例えば管理者によって設定され得る。

【００３４】

[0048]図９の説明を続けると、動作（９０４）は、特性に基づいて仮想マシン用の仮想ＮＵＭＡノードトポロジー選択をするステップであって仮想ＮＵＭＡノードトポロジーが、複数の仮想ＮＵＭＡノードを含んでいるもの、を示している。例えば、親パーティション（２０４）におけるプロセス（及び／又はハイパーバイザー（２０２））は、受信された特性に基づいて仮想マシン（２４０）に関するトポロジーを決定し得る。例えば、親パーティション（２０４）は、仮想ＮＵＭＡノード（６０６）などの仮想ＮＵＭＡノードに関するデフォルトサイズを識別する情報を含み得る。親パーティション（２０４）におけるプロセスは、仮想マシン（２４０）に対する仮想ＮＵＭＡ数を決定するためのデフォルトサイズ及び所望の特性を記述している情報を使用し得る。特定の例において、所望される特性は、１０ギガバイトのＲＡＭを有する６つのプロセッサー仮想マシンであり得る。仮想ＮＵＭＡノードのデフォルトサイズが２つの仮想プロセッサー及び４ギガバイトのＲＡＭを含む場合、管理システム（５０２）は、仮想マシン（２４０）が３つの仮想ＮＵＭＡノードを含むことを示す構成ファイルを生成し得る。

【００３５】

[0049]実施形態において、デフォルトの仮想ＮＵＭＡノードサイズは、管理者によるか又は管理システム（５０２）によって設定され得る。図５に移ると、管理システム（５０２）は、データセンター（５００）における計算機システム（５０４～５１０）の物理トポロジーを識別する情報、例えば、計算機システム（５０４～５１０）それぞれが（もし

あれば）いくつのＮＵＭＡノードを有しているか、ＲＡＭ速度、計算機システム（５０４～５１０）それぞれがどのくらいの容量ＲＡＭを有しているか、ＲＡＭがどのように配列されているか、プロセッサー速度、それぞれのプロセッサーがいくつのコアを有しているかなどを識別する情報、を取得し得る１つ以上のプログラムを実行し得る。

【００３６】

[0050]通常、仮想ＮＵＭＡノードサイズが、データセンター（５００）の仮想マシンの動作に影響する。例えば、仮想ＮＵＭＡノードサイズは、例えば、メモリー及び／又はプロセッサーにおいて増加するにつれて、仮想ＮＵＭＡノードの移植性が減少する。すなわち言い換えると、大きな仮想ＮＵＭＡノードが、仮想マシンを移動させることがより困難とし得る。このことは、仮想ＮＵＭＡノードが、仮想ＮＵＭＡノードを達成するために「フラット」リソースを有するＮＵＭＡノード又は計算機システムのどちらか一方に割り当てられる必要があるため生じる。例えば、仮想ＮＵＭＡノードあまりに大きな、例えば、それが非常に大きなＲＡＭか又はあまりに多くの仮想プロセッサーを有する場合、データセンター（５００）において、より小さいＮＵＭＡノードに適合させることが不可能であって、かくして、仮想マシンを移動させる能力を制限する。その上、より大きな仮想ＮＵＭＡノードが単に、より小さな複数のＮＵＭＡノードに割り当てられた場合、仮想マシンノードの性能は、ローカルメモリーアクセス時間とリモートメモリーアクセス時間との間に存在する相違による理由によって減少する。

【００３７】

[0051]他方では、仮想ＮＵＭＡノードのサイズが減少するにつれて、ゲストオペレーティングシステムの性能は悪影響を及ぼされ得る。この非効率性は、ゲストオペレーティングシステムがアプリケーションを分離しようとし、それが単一の仮想ＮＵＭＡノードに対する自身の実行であるために生じ得る。ゲストオペレーティングシステムは、この場合、抑制され、性能が低下する。

【００３８】

[0052]したがって、実施形態において、管理システム（５０２）は、データセンター（５００）に対する最適な仮想ＮＵＭＡノードサイズを決定することによって、移植性と効率性との間のバランスを決めることができる。例えば、実施形態において、管理システム（５０２）の論理プロセッサーがプログラムを実行し得、データセンターにおける平均ＮＵＭＡノードサイズ、例えば平均論理プロセッサー数、平均ＲＡＭ数などを決定し得、システムにおいて仮想ＮＵＭＡノードのサイズを平均ＮＵＭＡノードと同一か又はそれよりも小さくするように設定し得る。別の実施形態において、プログラムは、データセンター（５００）における仮想ＮＵＭＡノードサイズを最小ＮＵＭＡノードよりもわずかに小さく設定するように構成され得る。実施形態において、仮想ＮＵＭＡノードサイズが平均サイズ又は最小サイズよりわずかに小さいように設定され得、計算機システムが大量にコミットされた場合、２つ以上の仮想ＮＵＭＡノードが単一のＮＵＭＡノードへ割り当てられ得る。特定の例において、最小のＮＵＭＡノードが４つの論理プロセッサー及び８ギガバイトＲＡＭを有する場合、仮想ＮＵＭＡノードサイズは、例えば、２つの仮想プロセッサー及び４ギガバイトＲＡＭに設定され得る。

【００３９】

[0053]動作（９０６）は、計算機システム上に仮想マシンをインスタンス化するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含んでいるもの、を示している。実施形態において、ハイパーバイザー（２０２）が論理プロセッサーによって実行され得、複数の仮想ＮＵＭＡノードを有する仮想マシンがインスタンス化され得る。

そして図６及び／又は図７を参照すると、例えば、仮想ＮＵＭＡノード（６０６～６０８）を有する仮想マシン（２４０）は、計算機システム（６００）又は（７００）によって達成され得る。すなわち、ＶＩＤ（８１０）は、ＲＡＭからのシステム物理アドレスを有する仮想マシン（２４０）のゲスト物理アドレスと、１つ以上の論理プロセッサーを有する仮想プロセッサーと、を支援し得る。例えば、ゲスト物理アドレスブロック（６１４）がシステム物理アドレスブロック（６２２）を用いて支援され得、ゲスト物理アドレスブ

ロック（６１６）がシステム物理アドレスブロック（６２４）によって支援され得る。ハイパーバイザースレッドがその後、仮想プロセッサーを支援している論理プロセッサー上にスケジューリングされ得、仮想プロセッサーの命令指標が実行され得る。図６及び図７によって示されるように、仮想マシンそれぞれのトポロジーが、基本ハードウェアトポロジーから独立して生成され得る。すなわち、仮想マシンのトポロジーそれぞれは、それを達成する計算機システムの基本物理トポロジーから分離されている。

【００４０】

[0054]実施形態において、仮想マシンＢＩＯＳ又はブートファームウェアは、仮想マシンのトポロジーを記述し得、例えば、それは仮想ＮＵＭＡノード、任意の仮想ＮＵＭＡノードサイズ、及び仮想ＮＵＭＡノードに関するＮＵＭＡ比をモノリスアプリケーションのゲストオペレーティングシステムに対し有しているか否か説明し得る。データ構造は処理され得、ゲストＯＳ（２２０）又はアプリケーション及びそれが、仮想ＮＵＭＡノードの存在利点を取り入れるために、ＯＳ又はアプリケーションによって使用され得る。例えば、ゲストオペレーティングシステム（２２０）は、アプリケーションの実行がローカルのままであるように、ＮＵＭＡ非認識アプリケーションスレッドを仮想ＮＵＭＡノードと一体化しようとし得る。別の例において、ＳＱＬサーバーのようなデータベース管理プログラムは、ロックを仮想ＮＵＭＡノードへローカルに割り当て得、データベースが、仮想ＮＵＭＡノード全域の読み出し/書き出しリクエストを分割し得る。更に別の例において、ゲストオペレーティングシステム（２２０）は、仮想マシンに仮想ＮＵＭＡノードそれぞれに関するページプールを生成し得る。

【００４１】

[0055]図９の説明を続けると、動作（９０８）は、特定の仮想ＮＵＭＡノードに割り当てられるゲストメモリー容量を、特定の複数の仮想ＮＵＭＡノードにおけるメモリー圧力に基づいて調整するステップを示している。例えば、論理プロセッサー、例えば、図６又は図７の論理プロセス（２１２Ａ）は、ＤＭＶＳＰ（８０２）の命令指標を実行し得、仮想ＮＵＭＡノード（６０６）のような仮想ＮＵＭＡノードが、利用可能なゲスト物理アドレス量を調整し得る。すなわち、ＤＭＶＳＰ（８０２）が実行され得、メモリーが仮想ＮＵＭＡノードが経験している圧力に基づいてコミット又はデコミットされ得る。

【００４２】

[0056]実施形態において、仮想ＮＵＭＡノード（６０６～６０８）それぞれが、利用可能なメモリー容量によって、ゲストオペレーティングシステム（２２０）の性能がどのくらい影響されるかをメモリー圧力が識別し得る。この情報は、ＤＭＶＳＣ（８０４）及び／又は（８０６）のような例えば、１つ以上のＤＭＶＳＣによってゲストオペレーティングシステム（２２０）のランタイム中に算出され得、ＤＭＶＳＰ（８０２）へ送信され得る。例えば、メモリー圧力は、仮想ＮＵＭＡノードにおける異なるメモリー圧力レベルを識別し得る一連の値によって提示され得る。仮想ＮＵＭＡノードにおけるリソースが、より圧迫されるにつれて、すなわち、仮想ＮＵＭＡノード上の現在の作業負荷を効率的に実行するために要求されるメモリー容量が増加するにつれて、ＤＭＶＳＣ（８０４）が値を改定し得、この情報をＤＭＶＳＰ（８０２）へ伝達し得る。

【００４３】

[0057]実施形態において、メモリー圧力情報が、ゲストオペレーティングシステム（２２０）から受信された情報からＤＭＶＳＣ（８０４）によって算出され得る。例えば、ＤＭＶＳＣ（８０４）が、仮想ＮＵＭＡノード（６０６）に関するオペレーティングシステムのページング情報をメモリーマネージャー（８０８）から受信するように構成され得る。ゲストオペレーティングシステムのページングレートが、メモリーマネージャー（８０８）及びキャッシュマネージャによって公開される２つのカウンター、すなわち、ページングレート及びキャッシュ回転速度を介しモニターされ得る。

【００４４】

[0058]同一又は別の実施形態において、ＤＭＶＳＣ（８０４）が、メモリーマネージャー（８０８）からノード（６０６）を仮想ＮＵＭＡに関連付けられた物理メモリー通知を

受信し、この情報を使用し得、仮想ＮＵＭＡノード（６０６）のメモリー圧力を計算し得る。例えば、メモリーマネージャー（８０８）は、仮想ＮＵＭＡノード（６０６）に関連付けられたゲストオペレーティングシステム（２２０）における活動に基づいて、高いメモリーの通知及び低いメモリーの通知を出力し得る。メモリーマネージャー（８０８）は、低メモリー閾値（ＬＭＴ）及び高メモリー閾値（ＨＭＴ）に基づいてこれらの通知を起動し得る。特定の実施形態例において、低メモリーリソース通知イベントを信号送出する有効メモリーのデフォルトレベルは、およそ４ＧＢあたり約３２ＭＢ、最大６４ＭＢであり得る。例えば、高メモリーリソースの通知イベントを信号送出するデフォルトレベルは、デフォルトの低メモリーの値の３倍であり得る。２つの間の中間メモリーの有効レベルは、高メモリーの閾値レベルと低メモリーの閾値レベルとの間隔を分割することによって決定され得る。当業者は、これらの値が例示的であり得、変更が本開示の趣旨から逸脱せずに実施され得ることを十分に理解できよう。

【００４５】

[0059]これらの通知が、別のものと一緒にＤＭＶＳＣ（８０４）によって使用され得、仮想ＮＵＭＡノード（６０６）のメモリー圧力を算出し得る。例えば、レベルそれぞれは、例えば０～４の値に関連付けられ得、別の任意のパフォーマンスカウンターを考慮に入れた場合、それらの値も関連付けられ得る。パフォーマンスカウンターそれぞれに関する値はその後、仮想ＮＵＭＡノード（６０６）の現在のメモリー圧力を算出し得るために使用され得る。特定の例において、メモリー圧力が、より大きいか又は小さいパフォーマンスカウンター値を取ることによって算出され得る。別の例において、パフォーマンスカウンターの平均値が、メモリー圧力として使用され得る。更に別の実施形態において、より洗練されたアルゴリズムが使用され得、計算において、以前のパフォーマンスカウンターを考慮に入れ、相対重量に作用するスカラーをパフォーマンスカウンターそれぞれに割り当て、メモリー圧力を算出し得る。

【００４６】

[0060]メモリーをコミットするための決定が実行されたとき、ＤＭＶＳＰ（８０２）は、様々な技法を使用し得、その１つがホット（ｈｏｔ）追加動作である。例えば、オペレーティングシステムの中には、物理メモリー範囲が、システムのリブートを必要とせずに実行中のオペレーティングシステムに追加されることを可能にするホット追加を支援し得るものもいくつかある。すなわち、メモリーマネージャー（８０８）は、実行中のシステムへメモリーの動的追加を支援するように構成され得る。ホット追加の実施形態において、ＤＭＶＳＣ（８０４）が、メモリーマネージャー（８０８）のホット追加インターフェースをアクセスするように構成され得、ＤＭＶＳＣ（８０４）が、ホット追加されたＧＰＡ及びそれらがその仮想ＮＵＭＡノードに関連付けられていることを記述しているメッセージをゲストオペレーティングシステム（２２０）へ送信し得る。メモリーマネージャー（８０８）がその後、ゲストオペレーティングシステム（２２０）が利用可能な新しいメモリー、仮想ＮＵＭＡノード（６０６）上で実行しているドライバー、アプリケーション、又は別の任意のプロセスを生成し得る。例えば、ＤＭＶＳＣ（８０４）は、Ｖ１Ｄ（８１０）がＧＰＡとＳＰＡとの間に関係を生成した後、ＤＭＶＳＰ（８０２）からホット追加メモリーアドレスを受信し得る。

【００４７】

[0061]同様に、ホット移動動作が使用され得、仮想ＮＵＭＡノード（６０６）のような仮想ＮＵＭＡノードからメモリーアドレスをデコミットし得る。例えば、ＤＭＶＳＣ（８０４）は、メモリーがホット移動されたことを示すメッセージをゲストオペレーティングシステム（２２０）へ送信し得る。ＤＭＶＳＣ（８０４）は、メモリーマネージャー（８０８）が、仮想ＮＵＭＡノード（６０６）からＧＰＡブロックを移動用に提供するようにリクエストし得る。この例において、ＤＭＶＳＣ（８０４）はその後、メモリーマネージャー（８０８）の移動ＡＰＩを呼び出し得、ゲストオペレーティングシステム（２２０）からＧＰＡを移動し得る。ホット移動が使用される実施形態において、移動されるメモリーは、ゲストの現在のコミットに対し数えられず、メモリーマネージャー（８０８）は、

オペレーティングシステムによってメモリーを移動するために使用される同様の技法を使用し、内部カウンターを調整し得、マザーボードから物理的に移動される。

【００４８】

[0062]別の実施形態において、メモリーは、バルーニング技法を使用することによって、仮想ＮＵＭＡノードへデコミットされ得る。すなわち、メモリーは、仮想ＮＵＭＡ（６０６）ノードにおけるゲスト物理アドレスを、それらを支援している物理アドレスから分離することによってデコミットされ得る。例えば、論理プロセッサー（２１２Ｂ）は、ＤＭＶＳＣ（８０４）の命令指標を実行し得、メモリーマネージャー（８０８）が、ＤＭＶＳＣ（８０４）によって、例えば１つ以上のメモリーブロックを使用するための一定のメモリー容量を予約するリクエストメッセージをメモリーマネージャー（８０８）へ送信し得る。メモリーマネージャー（８０８）は、ＤＭＶＳＣ（８０４）及びＤＭＶＳＣ（８０４）の内部で排他的に使用するためにメモリーをロックし得、メモリーのＧＰＡをＤＭＶＳＰ（８０２）へ送信し得る。この例において、ＤＭＶＳＰ（８０２）は、ＧＰＡをＶＩＤ（８１０）へ送信し得、ＶＩＤ（８１０）は、これらのＧＰＡに関するエントリーをシャドウページテーブル内のＳＰＡへ移動し得る。この例において、メモリーマネージャー（８０８）は、ＧＰＡがまだ利用可能であるが、しかしながら実際には、ＧＰＡがもはやシステム物理アドレスによって支援されないことを識別する情報を含み得る。この例において、メモリーマネージャー（８０８）は、ロックされたＧＰＡを使用しないでそれらを支援するＳＰＡが再び割り当てられ得る。

【００４９】

[0063]分離されたゲスト物理アドレスは、物理アドレスに再度関連付けられ得る。この例において、メモリーページをコミットするためのリクエストがＶＩＤ（８１０）によって受信され得、ＶＩＤ（８１０）は、リクエストを満たしＳＰＡを取得し得、アドレスの範囲をＤＭＶＳＰ（８０２）へ送信し得る。実施形態において、ＶＩＤ（８１０）は、システム効率性を増大するために隣接するＳＰＡの範囲を取得するように構成され得る。この例において、ＶＩＤ（８１０）は、仮想ＮＵＭＡノード（６０６）に関連付けられるＤＭＶＳＣ（８０４）によって、排他的に使用するためにロックされるＧＰＡを有するゲストオペレーティングシステム（２２０）を決定し得る。ＶＩＤ（８１０）は、ロックされたＧＰＡとＳＰＡとの間に関係を生成し得、メッセージをＤＭＶＳＰ（８０２）へ送信し得る。ＤＭＶＳＰ（８０２）はその後、メッセージをＤＭＶＳＣ（８０４）へ送信し得、ＤＭＶＳＣ（８０４）は、ＧＰＡがアンロックされ、仮想ＮＵＭＡノード（６０６）に関連付けられたメモリーマネージャー（８０８）のメモリープールへ返却され得ることを示すメモリーマネージャー（８０８）へメッセージを送信し得る。

【００５０】

[0064]実施形態において、ＶＩＤ（８１０）は、ホット追加技法、又はＧＰＡが膨張するか否かによってバルーニング技法を使用するかどちらか一方を決定し得る。例えば、ＶＩＤ（８１０）がＮＵＭＡノード（６０６）へコミットするＳＰＡを受信したとき、それは、任意のＧＰＡがＤＭＶＳＣ（８０４）によってロックされるか否か決定し得る。ロックされたＧＰＡが存在する例において、ＶＩＤ（８１０）は、それがメモリーをホット追加する前にＳＰＡを用いてそれらを支援し得る。メモリーが仮想ＮＵＭＡノード（６０６）へコミットされる前に、それは０になり得、それに関連するキャッシュラインが、セキュリティのためにフラッシュされ得る。メモリーを０にすることによって、１つのパーティションに以前関付けられたメモリーのコンテンツが別のパーティションへリークされない。

【００５１】

[0065]ここで図１０に移ると、それは図９の追加的動作（１０１０～１０２０）を含んでいる動作手順の代替実施形態を表している。動作（１０１０）は、第２の複数の仮想ＮＵＭＡノードにおけるメモリー圧力が、所定の値よりも大きいことを決定するステップと、第２のＮＵＭＡノードを計算機システムの第２の仮想ＮＵＭＡノードへ移動させるステップと、を示している。そして図７に移ると、例えば、実施形態において、第２の仮想Ｎ

ＵＭＡノード（６０８）におけるメモリー圧力が増大し得る。すなわち、メモリー圧力の指標値が、仮想ＮＵＭＡノード（６０８）が圧迫されていることを示すＤＭＶＳＰ（８０２）によって、受信され得る。この例において、仮想マシン（２４０）又は個々の仮想ＮＵＭＡノード（６０８）は、目標の圧力値を有し得、現在の圧力値が管理者によって設定された目標値よりも大きい場合がある。目標の圧力値は、ＤＭＶＳＰ（８０２）によってアクセスされ得るデータ構造でストアされ得る。実行中の仮想マシン又は仮想ＮＵＭＡノードの現在の圧力値は、その後受信され得る。ＤＭＶＳＰ（８０２）は、実行中の仮想マシン又は仮想ＮＵＭＡノードリストを介し連続的にステップし得、メモリー圧力値を目標の値に減少させるためにメモリーをコミットし得、圧力を目標の値に増大するためにメモリーをデコミットし得る。

【００５２】

[0066]例において、ＤＭＶＳＰ（８０２）は、ＮＵＭＡノード仮想ＮＵＭＡノード（６０６）及び（６０８）を現在ホスティングしていて、例えば、ＮＵＭＡノード（７０２）がその仮想ＮＵＭＡノード双方に対する目標メモリー圧力値を取得するためのメモリーを十分に割り当てることが不可能なことを決定するように構成され得る。この例において、ＤＭＶＳＰ（８０２）は、ハイパーバイザー（２０２）へ信号を送信するように構成され得、ハイパーバイザー（２０２）は、仮想ＮＵＭＡノードの１つをＮＵＭＡノード（７０２）上へ移動しようと試みるように構成され得る。ハイパーバイザー（２０２）は、ＮＵＭＡノード（７０２～７０６）の現在の作業負荷をチェックし得、例えば、ＮＵＭＡノード（７０４）が、仮想ＮＵＭＡノードをホスティングし得、それに十分なリソースを割り当て得、メモリー圧力を目標の値に減少させ得ることを決定し得る。この例において、ハイパーバイザー（２０２）は、仮想ＮＵＭＡノード（６０８）をＮＵＭＡノード（７０４）に再度割り当てるように構成され得る。すなわち、ハイパーバイザー（２０２）はＶＩＤ（８１０）と連動してゲスト物理アドレス（６１６）をシステム物理アドレス（７１４）へ再度マッピングし得、論理プロセッサー（２１２Ｅ及びＦ）を仮想プロセッサー（２３０Ｃ及びＤ）に対する理想的なプロセッサーとして設定し得る。

【００５３】

[0067]図１０の説明を続けると、動作（１０１２）は、ゲストメモリーの少なくとも１つのメモリーブロックを特定の仮想ＮＵＭＡノードからデコミットするステップと、デコミットされたゲストメモリーの少なくとも１つのメモリーブロックを第２の仮想ＮＵＭＡノードへコミットするステップと、を例示している。例えば、ＤＭＶＳＰ（８０２）は、例えば、メモリーを仮想ＮＵＭＡノード（６０６）からデコミットし、メモリーを仮想ＮＵＭＡノード（６０８）へコミットするように構成され得る。この例において、仮想ＮＵＭＡノード（６０６）及び（６０８）は、単一のＮＵＭＡノード又は「フラット」アーキテクチャーによって支援され得る。この実施形態例において、ＤＩＭＩＶＳＰ（８０２）は、例えば、仮想ＮＵＭＡノード（６０８）へコミットされ得る利用可能な有効メモリーが存在しないとき、仮想ＮＵＭＡノード（６０６）からメモリーをフリーにしようと試み得る。別の例において、ＤＭＶＳＰ（８０２）は、例えば、メモリーを仮想ＮＵＭＡノード（６１０）からデコミットし、メモリーを仮想ＮＵＭＡノード（６０８）へコミットするように構成され得る。すなわち、メモリーがある仮想マシンから取得され得、別の仮想マシンに与えられ得る。

【００５４】

[0068]そして図６を参照すると、特定の例において、仮想ＮＵＭＡノード（６０６）及び（６０８）が計算機システム（６００）のリソースへマッピングされ得る。この例において、ＤＭＶＳＰ（８０２）は、別の仮想ＮＵＭＡノードを、例えば仮想マシン（２４０）における低優先順位の仮想ＮＵＭＡノード又は例えば、最低優先順位の仮想マシンを開始するメモリー優先順位でチェックし得る。例えば、目標の閾値よりも小さなメモリー圧力値を有する仮想ＮＵＭＡノード（６０６）のような仮想ＮＵＭＡノードが検出された場合、ＤＭＶＳＰ（８０２）は、メモリーをデコミット開始し得、仮想ＮＵＭＡノード（６０６）からメモリーを移動し得る。デコミットが完了した後、コミット動作が開始され得

、メモリーは、仮想ＮＵＭＡノード（６０８）へホット追加され得るか又は膨張したゲスト物理アドレスがシステム物理アドレスに再度関連付けられ得る。

【００５５】

[0069]そして図７を参照すると、特定の例において、ＤＭＶＳＰ（８０２）は、例えば、メモリーの優先順位で同一のＮＵＭＡノード（７０２）によって支援される別の仮想ＮＵＭＡノードをチェックし得る。例えば、仮想ＮＵＭＡノード（６０８）と同一のＮＵＭＡノード上の仮想ＮＵＭＡノードが、目標の閾値よりも小さなメモリー圧力値を有していることが検出された場合、ＤＭＶＳＰ（８０２）はメモリーをデコミット開始し得る。デコミットが完了した後にコミット動作が開始され得、メモリーが仮想ＮＵＭＡノード（６０８）へホット追加され得るか又は膨張されたゲスト物理アドレスがシステム物理アドレスと再度関連付けられ得る。

【００５６】

[0070]図１０の説明を続けると、動作（１０１４）は、特定の仮想ＮＵＭＡノードの少なくとも１つのゲストメモリーブロックが、システムメモリーと分離されることを決定するステップと、ゲストメモリーの少なくとも１つのメモリーブロックをシステムメモリーの少なくとも１つのメモリーブロック上へマッピングするステップと、を表している。例えば、実施形態において、ＤＭＶＳＰ（８０２）が論理プロセッサーによって実行され得、ＳＰＡ（６２４）を用いて仮想ＮＵＭＡノード（６０６）におけるＧＰＡを支援する決定が実行され得る。例えば、ＧＰＡがＤＭＶＳＣ（８０４）によって予約され得、ＳＰＡが別の仮想ＮＵＭＡノード又は親パーティション（２０４）のどちらか一方へ再度割り当てされ得る。この例において、メモリーページをコミットするためのリクエストがＶＩＤ（８１０）によって受信され得、ＶＩＤ（８１０）がリクエストを満足するＳＰＡを取得し得、アドレス範囲をＤＭＶＳＰ（８０２）へ送信し得る。実施形態において、ＶＩＤ（８１０）は、システム効率を増大するために隣接したＳＰＡの範囲を取得するように構成され得る。ＮＵＭＡ実施形態において、ＶＩＤ（８１０）は、仮想ＮＵＭＡノード（６０６）を実行中の同一のＮＵＭＡノードから隣接したＳＰＡの範囲を取得するように構成され得る。ＶＩＤ（８１０）は、ロックされたＧＰＡとＳＰＡとの間に関係を生成し得、メッセージをＤＭＶＳＰ（８０２）へ送信し得る。ＤＭＶＳＰ（８０２）はその後、メッセージをＤＭＶＳＣ（８０４）へ送信し得、ＤＭＶＳＣ（８０４）は、ＧＰＡがアンロックされ得、仮想ＮＵＭＡノード（６０６）と関連付けられたメモリープールへ返却され得ることを示すメッセージをメモリーマネージャー（８０８）へ送信し得る。

【００５７】

[0071]図１０の説明を続けると、動作（１０１６）は、特定の仮想ＮＵＭＡノードを計算機システムの第１のＮＵＭＡノード上へマッピングするステップと、特定の仮想ＮＵＭＡノードを計算機システムの第２のＮＵＭＡノード上へ移動させるステップと、を例示している。そして図７を参照すると、例えば、ゲストオペレーティングシステム（２２０）は、ＮＵＭＡノード（７０２）及び（７０４）のような少なくとも２つのＮＵＭＡノード全域に拡大され得る。そして図７を参照すると、例えば、ハイパーバイザー（２０２）は、仮想ＮＵＭＡノード（６０６）及び（６０８）が、ＮＵＭＡノード（７０２）上で実行するようにスケジューリングし得る。この例において、ハイパーバイザー（２０２）は、ＮＵＭＡノード（７０２）が圧迫されていることを示す信号を受信し得る。例えば、ゲストオペレーティングシステム（２２０）は、仮想ＮＵＭＡノード（６０６）及び（６０８）がメモリー上において低いことを示す信号を生成し得る。この例において、ハイパーバイザー（２０２）は、ＮＵＭＡノード（７０２）上に仮想ＮＵＭＡノード（６０８）を移動することによって、圧迫されているＮＵＭＡノードの作業負荷を減少させるように構成され得る。

【００５８】

[0072]図１０の説明を続けると、動作（１０１８）は、仮想プロセッサーを特定の仮想ＮＵＭＡノードへ追加するステップを例示している。例えば、実施形態において、仮想プロセッサー（２３０Ｂ）のような仮想プロセッサーが、仮想マシン（２４０）のランタイ

ム実行中に例えば、プロセッサーのホット追加動作を使用して追加され得る。すなわち、仮想ＮＵＭＡノード（６０６）は、ある地点において単一の仮想プロセッサー（２３０Ａ）だけを有し得、その後、別のプロセッサーが追加される。実施形態において、新たに追加されたプロセッサーが、仮想プロセッサー（２３０Ａ）を支援しているプロセッサーへ割り当てられ得るか、又は別の論理プロセッサーが、仮想プロセッサー（２３０Ｂ）のスレッドへ割り当てられ得、実行され得る。ＮＵＭＡ実施形態において、別の論理プロセッサーが使用され、仮想プロセッサー（２３０Ｂ）を支援している場合、それが仮想ＮＵＭＡノード（６０６）において別の仮想プロセッサーを支援している同一のＮＵＭＡノード（７０２）から割り当てられ得る。

【００５９】

[0073]図１０の説明を続けると、動作（１０２０）は、仮想マシンの仮想プロセッサーと、論理プロセッサーに割り当てられた仮想プロセッサーと、ＮＵＭＡノードに割り当てられた論理プロセッサーと、仮想ＮＵＭＡノードに割り当てられた仮想プロセッサーと、を実行するためのリクエストを受信するステップと、論理プロセッサーが仮想プロセッサーを実行不可能であることを決定するステップと、仮想プロセッサーを実行するための第２の論理プロセッサーを選択するステップであって第２の論理プロセッサーが、第２のＮＵＭＡノードから提供されているものと、を例示している。そして図７を参照すると、例えば、実施形態において、ハイパーバイザー（２０２）は、仮想プロセッサー（２３０Ａ）から仮想プロセッサースレッドを実行するためのリクエストを受信し得、理想プロセッサー（２１２Ａ）、例えば仮想プロセッサー（２３０Ａ）を支援するプロセッサー上にスレッドをスケジューリングしようと試み得る。この例において、ハイパーバイザー（２０２）は、論理プロセッサー（２１２Ａ）が過度にコミットされていて、仮想プロセッサースレッドを実行不可能なことを検出し得る。この場合、ハイパーバイザー（２０２）が実行され得、それが、仮想プロセッサースレッドを実行するための別の論理プロセッサーを選択し得る。例えば、ハイパーバイザー（２０２）は、同一のＮＵＭＡノード上において異なる論理プロセッサーを選択しようと試み得る。例えば、ＮＵＭＡノードが過度にコミットされている場合、ハイパーバイザー（２０２）は、仮想プロセッサー（２３０Ａ）、例えば論理プロセッサー（２１２Ｅ）を実行するためのリモートプロセッサーを選択するように構成され得る。この例において、リモートノード上のスレッドを待つか又はスケジューリングすることに関する決定が、ＮＵＭＡノード（７０４）に関連するＮＵＭＡ比を使用し実行され得る。ＮＵＭＡ比が低い場合、かつ、理想的なプロセッサーを待つ予測時間が長い場合、ＮＵＭＡノード（７０４）上にスレッドをスケジューリングする決定が実行され得る。他方では、ＮＵＭＡ比が高い場合、かつ、待つ予測時間が短い場合、待機する決定が実行され得る。

【００６０】

[0074]ここで図１１に移ると、それは動作（１１００）、（１１０２）、（１１０４）、及び（１１０６）を含んでいる本開示の態様を実施するための動作手順を表している。動作（１１００）が動作手順を開始し、動作（１１０２）は、仮想マシンを実行するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含むトポロジーを有しているものと、仮想マシンのトポロジーが、計算機システムの物理トポロジーから独立して生成されていることを示している。例えば、ハイパーバイザー（２０２）は、複数の仮想ＮＵＭＡノードを有する仮想マシンを実行し得る。図６によって示されるような仮想ＮＵＭＡノード（６０６）及び（６０８）を含む仮想マシン（２４０）が作成され得る。仮想ＮＵＭＡノードはそれぞれ、１つ以上の仮想プロセッサー（２３０Ａ～Ｄ）と、ゲスト物理アドレス（６１４）及び（６１６）とを有し得る。この実施形態において、仮想ＮＵＭＡノード（６０６）及び（６０８）が基本ハードウェアトポロジーから独立して生成され得る。すなわち、仮想マシンのトポロジーは、図６及び図７によって表されるような基本ハードウェアと関係しない。かくして、この実施形態において、仮想マシンのトポロジーそれぞれは、それを達成する計算機システムの基本物理トポロジーから分離されている。

【００６１】

[0075]図１１の説明を続けると、動作（１１０４）は、複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するステップを例示している。そして図８を参照すると、例えば、仮想ＮＵＭＡノード（６０６）及び（６０８）それぞれに対するメモリー圧力が取得され得、例えば、生成され及び／又は受信され得る。メモリー圧力情報は、ゲストの実績が仮想ＮＵＭＡノード（６０６～６０８）それぞれが利用可能なメモリー容量によってどのように作用されるか識別し得る。この情報は、例えば、ＤＭＶＳＣ（８０４）及び／又は（８０６）のような１つ以上のＤＭＶＳＣによってゲストオペレーティングシステム（２２０）のランタイム中に算出され得、ＤＭＶＳＰ（８０２）へ送信され得る。すなわち、特定の実施形態において、論理プロセッサーは、ＤＭＶＳＣ（８０４）又は（８０６）の命令指標を実行し得、仮想ＮＵＭＡノードそれぞれに対するメモリー圧力情報を生成し得る。この情報はその後、例えば、ＤＭＶＳＰ（８０２）へ送信され得る。

【００６２】

[0076]実施形態例において、メモリー圧力情報は、０～４の範囲の一連の値を含み得、それぞれの値は、ゲストＯＳが、仮想ＮＵＭＡノード（６０６～６０８）のリソースによる経験をしている異なるレベルのメモリー圧力を識別し得る。ゲストオペレーティングシステムが、より圧迫されるにつれて、すなわち、現在の作業負荷を効率的に実行するために必要なメモリー容量が増加するにつれて、ＤＭＶＳＣ（８０４）及び（８０６）は、それらの値を改め、この情報をＤＭＶＳＰ（８０２）へ伝達し得る。

【００６３】

[0077]図１１の説明を続けると、動作（１１０６）は、複数の仮想ＮＵＭＡノードの少なくとも１つに割り当てられるゲストメモリーを複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力に基づいて調整するステップを示している。動作（１２０６）を含む実施形態において、論理プロセッサー（２１２Ａ）は、ＤＭＶＳＰ（８０２）の命令指標を実行し得、例えば、仮想ＮＵＭＡノード（６０６）におけるゲスト物理アドレス量を調整し得る。すなわち、ＤＭＶＳＰ（８０２）は、仮想ＮＵＭＡノード（６０６）におけるメモリー圧力に基づいてメモリーをコミット又はデコミットし得、例えばプロセスが、圧迫されている仮想ＮＵＭＡノード（６０６）に割り当てられた場合、メモリーがコミットされ得る。

【００６４】

[0078]実施形態において、ＤＭＶＳＰ（８０２）によって構成される論理プロセッサー（２１２）がメモリーをコミットするか又はデコミットする決定をしたとき、それがメモリーブロックベース単位でそのように実行し得る。例えば、ＤＭＶＳＰ（８０２）は、メモリーブロックをコミット又はデコミットし得、メモリーステータスがどのように変化するかチェックし得る。メモリーステータスが変化していない場合、ＤＭＶＳＰ（８０２）は、別のメモリーブロックをコミット又はデコミットし得る。

【００６５】

[0079]ここで図１２に移ると、それは動作（１２０８～１２１６）を含んでいる図１１の動作手順１（１００）の代替実施形態を表している。図面によって例示されるように動作（１２０８）は、仮想マシンを第２の計算機システムへ送信するステップを示している。そして図５を参照すると、例えば、実施形態において、仮想マシンの状態が１つ以上の構成ファイルにセーブされ得、別の計算機システムへ、例えば、計算機（５０４）から（５０６）へ送信され得る。計算機システム（５０６）のハイパーバイザーは、ファイルを読み出し得るか又はファイル仮想マシンをインスタンス化し得る。

【００６６】

[0080]仮想マシンのトポロジーは、仮想マシンへ移動させてリストアする能力に作用する。具体的には、基本ハードウェアトポロジーを検出可能にする決定及び仮想ＮＵＭＡノードサイズが、仮想マシンがどのくらい十分に実行するか及びそれが容易に移動され得るか否かに影響する。例えば、仮想ＮＵＭＡノードサイズは、移動させる機能に作用し、例えば、仮想ＮＵＭＡノードサイズが増加するにつれて仮想ＮＵＭＡノードの移植性が減少し、仮想ＮＵＭＡノードサイズが減少するにつれて仮想マシンの性能も低下する。加える

と、基本計算機のトポロジーを検出し得る仮想マシンは、ＮＵＭＡ認識オペレーティングシステム及びアプリケーションが、ブート時間に仮想マシンが検出する第１のトポロジーに基づいて自らを最適化するという事実のために容易に移動され得ず、これらの最適化は、将来、仮想マシンが移動され得る計算機上において十分に実行し得ない。かくして、仮想ＮＵＭＡノードをゲストオペレーティングシステムに公開することによって、それがブートしたとき、オペレーティングシステムがＮＵＭＡノードを使用するように最適化され得る。仮想ＮＵＭＡノードを正確にサイジングすることによって、仮想マシンが、データセンター（５００）における多くの様々な計算機システムに対し最適化され得る。

【００６７】

[0081]そして図６を参照すると、例えば、仮想マシン（２４０）は２つ以上の仮想ＮＵＭＡノード（６０６）及び（６０８）を含み得る。ハイパーバイザー（２０２）は、論理プロセッサー（２１２Ａ～Ｄ）を用いて仮想プロセッサー（２３０Ａ～Ｄ）を支援し得る。ゲストオペレーティングシステム（２２０）がブートしたとき、それが仮想ＮＵＭＡノード（６０６）及び（６０８）を検出し得、仮想ＮＵＭＡノード（６０６）及び（６０８）を使用するプロセスをスケジューリング及び実行を最適化するように構成され得る。しばらくして仮想マシン（２４０）は、図７によって表されたものと同様の物理トポロジーを有する計算機システムへ移動され得る。図７のハイパーバイザー（２０２）は、論理プロセッサー（２１２Ａ及びＢ）を用いて仮想プロセッサー（２３０Ａ及びＢ）を支援し得、論理プロセッサー（２１２Ｅ及びＦ）を用いて仮想プロセッサー（２３０Ｃ及びＤ）を支援し得る。ゲストオペレーティングシステム（２２０）は、基本計算機トポロジーがＳＭＰからＮＵＭＡへ変更した時でも図６の計算機システム上で実行したときと同一の方法で作動し続け得る。

【００６８】

[0082]図１２の説明を続けると、動作（１２１０）は、第１の複数の仮想ＮＵＭＡノードを計算機システムの第１のＮＵＭＡノードへマッピングするステップと、第２の複数の仮想ＮＵＭＡノードを計算機システムの第１のＮＵＭＡノードへマッピングするステップと、を示している。そして図７を参照すると、例えば、論理プロセッサー（２１２Ａ）のような論理プロセッサーは、ハイパーバイザー命令を実行し得、仮想ＮＵＭＡノード（６０６）及び（６０８）のような仮想ＮＵＭＡノードをＮＵＭＡノード（７０２）のようなＮＵＭＡノードへ一体化し得る。より具体的には、論理プロセッサーは、システム物理アドレスを用いてＮＵＭＡノード（７０２）のＲＡＭ（２１４）からゲスト物理アドレス（６１４）及び（６１６）を支援し得、論理プロセッサー（２１２Ａ～Ｄ）を用いて仮想プロセッサー（２３０Ａ～Ｄ）を支援し得る。

【００６９】

[0083]図１２の説明を続けると、動作（１２１２）は、第１の複数の仮想ＮＵＭＡノードを計算機システムの第１のＮＵＭＡノード上へマッピングするステップと、第２の複数の仮想ＮＵＭＡノードを計算機システムの第２のＮＵＭＡノード上へマッピングするステップと、を示している。そして図７を参照すると、例えば、論理プロセッサー（２１２Ａ）のような論理プロセッサーは、ハイパーバイザー命令を実行し得、仮想ＮＵＭＡノード（６０６）をＮＵＭＡノード（７０２）へ割り当て得、仮想ＮＵＭＡノード（６０８）をＮＵＭＡノード（７０４）へ割り当て得る。この例において、ハイパーバイザー（２０２）が実行されるとき、ハイパーバイザースケジューラーは、仮想プロセッサー（２３０Ａ～Ｂ）からのスレッドを論理プロセッサー（２１２Ａ～Ｄ）上にスケジュールし得、仮想プロセッサー（２３０Ｃ又はＤ）からのスレッドを論理プロセッサー（２１２Ｅ～Ｇ）上にスケジュールし得る。

【００７０】

[0084]ここで図１３に移ると、それは図１２の動作手順の代替実施形態を例示していて、第２の仮想ＮＵＭＡノードにおけるメモリー圧力が、所定の値よりも大きいことを決定するステップと、第２の仮想ＮＵＭＡノードを計算機システムの第２のＮＵＭＡノードに移動させるステップと、を示している動作（１３１４）を含んでいる。そして図７に移る

と、例えば、実施形態において、第２の仮想ＮＵＭＡノード（６０８）におけるメモリー圧力が増大し得る。すなわち、メモリー圧力の値の指標は、仮想ＮＵＭＡノード（６０８）が圧迫されていることを示すＤＭＶＳＰ（８０２）を受信し得る。この例において、仮想ＮＵＭＡノード（６０６～６１２）及び／又は仮想マシン（２４０～２４２）それぞれは、目標の圧力値を有し得、仮想ＮＵＭＡノード（６０８）に対する現在の圧力値が管理者によって設定された目標値よりも大きい場合がある。目標圧力値は、ＤＭＶＳＰ（８０２）によってアクセスされ得るデータ構造でストアされ得る。現在実行中の仮想マシン又は仮想ＮＵＭＡノードの圧力値がその後、受信され得る。ＤＭＶＳＰ（８０２）は、実行中の仮想マシン又は仮想ＮＵＭＡノードの一覧を介し順次にステップし、メモリー圧力値を目標値に減少させるようにメモリーをコミットし、目標値に対する圧力を増やすようにメモリーをデコミットする。

【００７１】

[0085]ここで図１４に移ると、それは図１２の動作手順の代替実施形態を例示していて、第２の仮想マシンメモリー圧力が、所定の値よりも大きいことを決定するステップと、仮想マシンの第２の仮想ＮＵＭＡノードを計算機システムの第１のＮＵＭＡノードへ移動させるステップと、を示している動作（１４１６）を含んでいる。実施形態において、少なくとも２つの仮想マシン、例えば、仮想マシン（２４０）及び（２４２）が実行され得る。

この例において、仮想マシン（２４０）の仮想ＮＵＭＡノードが、例えば、図７のＮＵＭＡノード（７０２）及び（７０４）へマッピングされ得、仮想マシン（２４２）の仮想ＮＵＭＡノードが、例えば、ＮＵＭＡノード（７０６）へマッピングされ得る。この例において、仮想マシン（２４０）及び（２４２）それぞれ及び／又は仮想ＮＵＭＡノード（６０６～６１２）それぞれは目標の圧力値を有し得、ＤＭＶＳＰ（８０２）によってアクセスされ得るデータ構造でストアされ得る。この例において、第２の仮想マシンのメモリー圧力（２４２）は、仮想マシン、例えば、多くの読み出し又は書き出しリクエストを受信した仮想マシン（２４２）における活動のために増大し得、その値がＤＭＶＳＰ（８０２）によって受信され得る。ＤＭＶＳＰ（８０２）が、実行中の仮想マシン又は仮想ＮＵＭＡノードの現在の圧力値を受信し得、実行中の仮想マシン又は仮想ＮＵＭＡノードのリストを介し連続的にステップし得、メモリー圧力を軽減するために、メモリーが仮想マシン（２４２）にコミットされ得るか否か決定し得る。

【００７２】

[0086]圧力が、メモリーをコミット又はデコミットすることによって軽減できない状況において、ＤＭＶＳＰ（８０２）は、ハイパーバイザー（２０２）へ信号を送信するように構成され得、ハイパーバイザー（２０２）は、メモリー圧力を軽減するために、計算機システムのリソースを再度割り当てしようと試み得る。例えば、ハイパーバイザー（２０２）は、ＮＵＭＡノード（７０２～７０６）の現在の作業負荷をチェックし得、例えば、ＮＵＭＡノード（７０２）が、仮想マシン（２４０）から仮想ＮＵＭＡノードをホスティングし得、仮想ＮＵＭＡノード（６０８）をＮＵＭＡノード（７０２）へ再度割当てし得ることを決定できる。すなわち、ハイパーバイザー（２０２）は、ＶＩＤ（８１０）と連動してゲスト物理アドレス（６１６）をシステム物理アドレス（７１２）へ再度マッピングし得、論理プロセッサー（２１２Ａ及びＤ）を仮想プロセッサー（２３０Ｃ及びＤ）に対する理想的なプロセッサーとして設定し得る。その後、ハイパーバイザーは、仮想ＮＵＭＡノード（６１０）をＮＵＭＡノード（７０４）に再度マッピングし得、仮想マシン（２４２）の仮想ＮＵＭＡノード（６１０～６１２）それぞれの内部のメモリーを、そのメモリー圧力を減少させるために調整し得る。

【００７３】

[0087]ここで図１５に移ると、それは動作（１５００）、（１５０２）、及び（１５０４）を含んでいる本開示の態様を実施するための動作手順を例示している。動作（１５００）が動作手順を開始し、動作（１５０２）は、第１の仮想マシンを実行するステップであって仮想マシンが、複数の仮想ＮＵＭＡノードを含むトポロジーを有しているものと、

複数の仮想ＮＵＭＡノードそれぞれが、仮想プロセッサー及びゲスト物理アドレスを含んでいるものと、仮想マシンのトポロジーが、計算機システムの物理トポロジーから独立して生成されていることを示している。例えば、図７のハイパーバイザー（２０２）は、複数の仮想ＮＵＭＡノード（６０８～６１０）を有している仮想マシン（２４０）を実行し得る。仮想ＮＵＭＡノード（６０６）及び（６０８）はそれぞれ、１つ以上の仮想プロセッサー（２３０Ａ～Ｄ）と、ゲスト物理アドレス（６１４）及び（６１６）と、を有し得る。この実施形態において、仮想ＮＵＭＡノード（６０６）及び（６０８）が計算機システム（７００）のリソースへマッピングされ得る。例えば、論理プロセッサー（２１２Ａ及び２１２Ｂ）は仮想プロセッサー（２３０Ａ及び２３０Ｂ）に対する理想的なプロセッサーとして設定され得、ゲスト物理アドレス（６１４）がシステム物理アドレス（７１０）によって支援され得る。同様に論理プロセッサー（２１２Ｅ）及び（２１２Ｆ）が仮想プロセッサー（２３０Ｃ）及び（２３０Ｄ）に対する理想的なプロセッサーとして設定され得、ゲスト物理アドレス（６１６）がシステム物理アドレス（７１４）へマッピングされ得る。この実施形態において、仮想ＮＵＭＡノード（６０６）及び（６０８）は、基本ハードウェアトポロジーから独立している。すなわち、仮想マシンのトポロジーは、図６及び図７によって表されるような基本ハードウェアと関係しない。かくして、この実施形態において、仮想マシンのトポロジーは、それを達成する計算機システムの基本物理トポロジーから分離されている。

【００７４】

[0088]図１５の説明を続けると、動作（１５０４）は、増設の仮想プロセッサーを複数の仮想ＮＵＭＡノードに追加するステップを示している。例えば、実施形態において、増設の仮想プロセッサーが、例えば、仮想ＮＵＭＡノード（６０６）のような仮想ＮＵＭＡノードに追加され得る。

この例において、仮想プロセッサーのような仮想プロセッサーが、例えば、プロセッサーのホット追加動作を使用し仮想マシン（２４０）のランタイム実行中に追加され得る。実施形態において、新しく追加された仮想プロセッサーが仮想プロセッサー（２３０Ａ）を支援しているプロセッサーへ割り当てられ得るか、又は別の論理プロセッサーが新しく追加された仮想プロセッサーのスレッドを実行する理想的なプロセッサーとして設定され得る。ＮＵＭＡ実施形態において、論理プロセッサーは、仮想ＮＵＭＡノード（６０６）を支援している同一のＮＵＭＡノード（７０２）から割り当てられ得る。

【００７５】

[0089]ここで図１６に移ると、それは付加的な動作（１６０６～１６１２）を含んでいる図１５によって表されている動作手順の代替実施形態を示している。動作（１６０６）は、複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するステップと、複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力に基づいて複数の仮想ＮＵＭＡノードの少なくとも１つに割り当てられるゲストメモリーを調整するステップと、を示している。図８を参照すると、仮想ＮＵＭＡノード（６０６）及び（６０８）のそれぞれに対するメモリー圧力が取得され得、例えば生成され及び／又は受信され得る。メモリー圧力情報は、仮想ＮＵＭＡノードそれぞれが利用可能なメモリー容量によってゲスト性能がどのように作用されるか識別し得る。ＤＭＶＳＣ（８０４）及び（８０６）は、例えば、メモリーマネージャー（８０８）から物理メモリー通知及び／又はゲストオペレーティングシステムのページング情報を受信し、それを利用し、仮想ＮＵＭＡノード（６０６）及び（６０８）それぞれのメモリー圧力を計算するように構成され得る。

【００７６】

[0090]本例を続けると、論理プロセッサー（２１２Ａ）は例えば、ＤＭＶＳＰ（８０２）の命令指標を実行し得、例えば、仮想ＮＵＭＡノード（６０６）におけるゲスト物理アドレス量を調整し得る。すなわち、ＤＭＶＳＰ（８０２）は、ゲストＯＳがリソースのために仮想ＮＵＭＡノード（６０６）において経験しているメモリー圧力に基づいて、メモリーをコミット又はデコミットする。

【００７７】

[0091]図１６の説明を続けると、動作（１６０８）は、複数の仮想ＮＵＭＡノードから仮想プロセッサーを移動するステップを例示している。（１７０６）を含む実施形態において、ハイパーバイザー（２０２）が論理プロセッサーによって実行され得、仮想プロセッサーが仮想ＮＵＭＡノード（６０６）から移動され得る。例えば、ハイパーバイザー（２０２）は、ゲストオペレーティングシステム（２２０）のホット移動ＡＰＩをアクセスし得、例えば、仮想プロセッサー（２３０Ｂ）を仮想ＮＵＭＡノード（６０８）から移動し得る。

【００７８】

[0092]ここで動作（１６１０）に移ると、それは複数の仮想ＮＵＭＡノードに関するＮＵＭＡ比をゲストオペレーティングシステムへ報告するステップを示している。例えば、実施形態において、ハイパーバイザー（２０２）は、仮想ＮＵＭＡノード（６０６～６０８）に関するＮＵＭＡ比を生成し得、この情報が図６又は図７のどちらか一方のゲストオペレーティングシステム（２２０）へ報告され得る。実施形態において、ハイパーバイザー（２０２）は、仮想ＮＵＭＡノードに関するＮＵＭＡ比を示しているデータ構造を仮想マシンのファームウェアテーブルに生成し得、ゲストオペレーティングシステム（２２０）がブートしたとき、ゲストが、テーブルを読み出し得、情報を利用し得、スケジューリング決定を実行し得る。例えば、ゲストオペレーティングシステム又はＮＵＭＡ認識アプリケーションが、ＮＵＭＡ比を利用し得、リモートＮＵＭＡノードからのリソースを使用するか否か決定し得る。例えば、オペレーティングシステムは、実行される準備をしている保留中のスレッドを有し得る。この例において、オペレーティングシステムも、フリーになる理想的なプロセッサーを一定の時間、待つように構成され得るか、さもなければ、それは、所定よりも小さなＮＵＭＡ比を有するスレッドをリモートプロセッサー上にスケジューリングする。この場合、スケジューラーが受け入れる時間はＮＵＭＡ比に依存している。

【００７９】

[0093]ここで動作（１６１２）に移ると、それは仮想マシンを第１のＮＵＭＡノードから複数のＮＵＭＡノードへ移動させるステップを示している。例えば、実施形態において、ハイパーバイザー命令は論理プロセッサーによって実行され得、仮想マシン（２４０）は複数のＮＵＭＡノード（７０４）及び（７０６）へマッピングされ得る。この例において、計算機システム（７００）は乱用され得、例えば仮想マシン（２４２）は、リソースの大部分を使用している場合があって、そのため、仮想マシン（２４２）が計算機（７００）から移動される。この状況において、ハイパーバイザー（２０２）は、計算機システム（７００）のリソースを再割り当てし得、仮想マシン（２４０）をＮＵＭＡノード（７０４）及び（７０６）へ再マッピングし得る。

【００８０】

[0094]ここで図１７に移ると、それは図１６の動作手順の動作（１７１４）及び（１７１６）を含む代替実施形態を表している。実施形態において、動作（１６０６）は、仮想ＮＵＭＡノードの現在のメモリー圧力が、目標の閾値よりも低いという決定に基づいて第１の複数の仮想ＮＵＭＡノードからメモリーをデコミットするステップを示している動作（１７１４）を含み得る。例えば、実施形態において、ＤＭＶＳＰ（８０２）は、仮想ＮＵＭＡノード（６０８）のメモリー圧力が、目標範囲よりも低いとき、仮想ＮＵＭＡノード（６０６）からメモリーをデコミットするように構成され得る。例えば、実施形態において、ＤＭＶＳＰ（８０２）は、例えば、コミットされ得る親パーティションにおいてメモリーも利用可能でないとき、及び仮想ＮＵＭＡノード（６０８）が受け入れられない圧力を経験しているとき、仮想ＮＵＭＡノード（６０６）からメモリーをフリーにしようと試み得る。メモリーが仮想ＮＵＭＡノード（６０６）からデコミットされた場合、非同期メッセージが、メモリーをデコミットする指示するためのＤＭＶＳＣ（８０４）が送信され得る。ゲストＯＳ（２２０）に関連するＤＭＶＳＣ（８０４）が応答したとき、それは仮想ＮＵＭＡノード（６０６）内部の新しいメモリー圧力を示し得る。場合によっては、メモリー圧力が、メモリーの移動動作に応答し増大され得るものもある。

【００８１】
[0095]図１７の説明を続けると、動作（１７１６）は、ゲストオペレーティングシステムの現在のメモリー圧力が目標の閾値よりも大きいという決定に基づいて、第１の複数の仮想ＮＵＭＡノードへメモリーをコミットするステップを示している。例えば、実施形態において、ＤＭＶＳＰ（８０２）は、仮想ＮＵＭＡノード（６０６）のメモリー圧力が目標範囲よりも大きいとき、メモリーを仮想ＮＵＭＡノード（６０６）へコミットするように構成され得る。この場合、メモリーが利用可能な場合、それが仮想ＮＵＭＡノード（６０６）に割り当てられ得る。すなわち、ＤＭＶＳＰ（８０２）は、利用可能なメモリー容量によって仮想ＮＵＭＡノード（６０６）の性能がどのように作用されるか識別するメモリー圧力情報を取得し得、メモリーを仮想ＮＵＭＡノード（６０６）へ追加し得る。特定の例において、メモリー圧力情報は数値であり得る。この例において、ＤＭＶＳＰ（８０２）は、現在のメモリー圧力値を仮想ＮＵＭＡノード（６０６）に対する最小値を示す情報テーブルと比較し得、仮想ＮＵＭＡノード（６０６）のメモリー圧力が最小と等しくなるまでメモリーを調整し得る。例えば、管理者は、仮想ＮＵＭＡノード（６０６）及び（６０８）に対する最小値を有する不可欠なアプリケーションを実行するゲストオペレーティングシステムを構成し得る。

【００８２】
[0096]前述の詳細な説明は、例及び／又は動作図面を介し、システム及び／又はプロセスの様々な実施形態を詳細に説明している。そのようなブロック図及び／又は例が１つ以上の機能及び／又は動作を含む場合、そのようなブロック図又は例の中の機能及び／又は動作それぞれは、様々な範囲のハードウェア、ソフトウェア、ファームウェア、又は実際にはその任意の組み合わせによって個別に及び／又はまとめて実装され得ることが当業者によって理解されよう。

【００８３】
[0097]本明細書に記述された本対象事項の具体的な態様が示され、説明されているが、当業者にとって、本明細書の教示に基づく変更及び修正が本明細書に記述された対象項目及びそのより広い態様から離れずに実行され得ることは明らかであって、したがって、添付の請求項は、それらの変更及び修正すべてが本明細書に記述された対象項目の本当の趣旨及び範囲内にあるようにその範囲内に包含される。

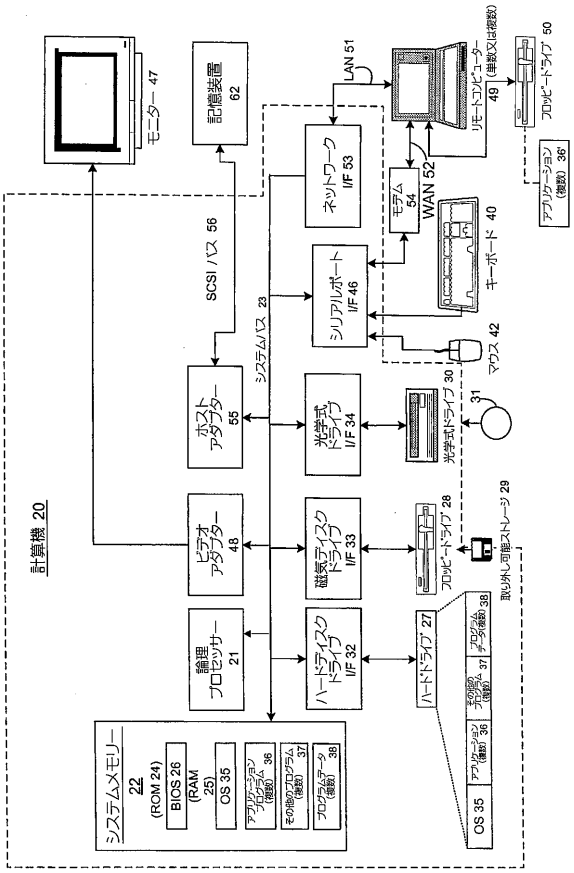
【符号の説明】

【００８４】
２０　計算機システム
２１　論理プロセッサー
２２　システムメモリー
２３　システムバス
２４　読み出し専用メモリー（ＲＯＭ）
２５　ランダムアクセスメモリー（ＲＡＭ）
２６　基本入力／出力システム（ＢＩＯＳ）
２７　ハードディスクドライブ
２８　磁気ディスクドライブ
２９　取り外し可能磁気ディスク
３０　光学式ディスクドライブ
３１　取り外し可能光学式ディスク
３２　ハードディスクドライブインターフェース
３３　磁気ディスクドライブインターフェース
３４　光学式ドライブインターフェース
３５　オペレーティングシステム
３６　アプリケーションプログラム
３７　その他のプログラムモジュール
３８　プログラムデータ

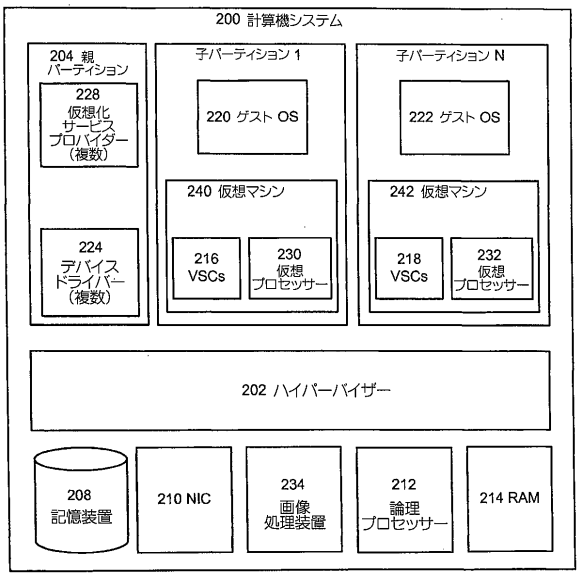
４０　キーボード
４２　ポインティングデバイス
４６　シリアルポートインターフェース
４７　ディスプレイ
４８　ビデオアダプター
４９　リモートコンピューター
５０　メモリー記憶装置
５１　ローカルエリアネットワーク（ＬＡＮ）
５２　広域ネットワーク（ＷＡＮ）
５３　アダプター
５４　モデム
５５　ホストアダプター
５６　小型コンピューター用周辺機器インターフェース（ＳＣＳＩ）バス
６２　外部記憶装置
２００　計算機システム
２０２　ハイパーバイザー
２０４　親パーティション
２０８　記憶装置
２１０　ネットワークインターフェースコントローラー（ＮＩＣ）
２１２　論理プロセッサー
２１４　ランダムアクセスメモリー（ＲＡＭ）
２１６　仮想化サービスクライアント（ＶＳＣ）
２１８　仮想化サービスクライアント（ＶＳＣ）
２２０　ゲストオペレーティングシステム
２２２　ゲストオペレーティングシステム
２２４　デバイスドライバー
２２８　仮想化サービスプロバイダー
２３０　仮想プロセッサー
２３２　仮想プロセッサー
２３４　画像処理装置
２３６　設定ユーティリティ
２４０　仮想マシン
２４２　仮想マシン
３００　計算機システム
５００　データセンター
５０２　管理システム
５０４　計算機システム
５０６　計算機システム
５０８　計算機システム
５１０　計算機システム
６００　計算機システム
６０２　メモリーコントローラー
６０６　仮想ＮＵＭＡノード
６０８　仮想ＮＵＭＡノード
６１０　仮想ＮＵＭＡノード
６１２　仮想ＮＵＭＡノード
６１４　ゲスト物理アドレス
６１６　ゲスト物理アドレス
６１８　ゲスト物理アドレス
６２０　ゲスト物理アドレス

６２２　システム物理アドレスブロック
６２４　システム物理アドレスブロック
６２６　システム物理アドレスブロック
６２８　システム物理アドレスブロック
６３０　システム物理アドレスブロック
６３２　システム物理アドレスブロック
７００　計算機システム
７０２　ＮＵＭＡノード
７０４　ＮＵＭＡノード
７０６　ＮＵＭＡノード
７０８　相互接続
８０２　ダイナミックメモリー仮想化サービスプロバイダー（ＤＭＶＳＰ）
８０４　ダイナミックメモリー仮想化サービスクライアント（ＤＭＶＳＣ）
８０６　ダイナミックメモリー仮想化サービスクライアントＤＭＶＳＣ
８０８　ゲストオペレーティングシステムメモリーマネージャー
８１０　仮想基盤ドライバー（ＶＩＤ）
８１２　ワーカープロセス

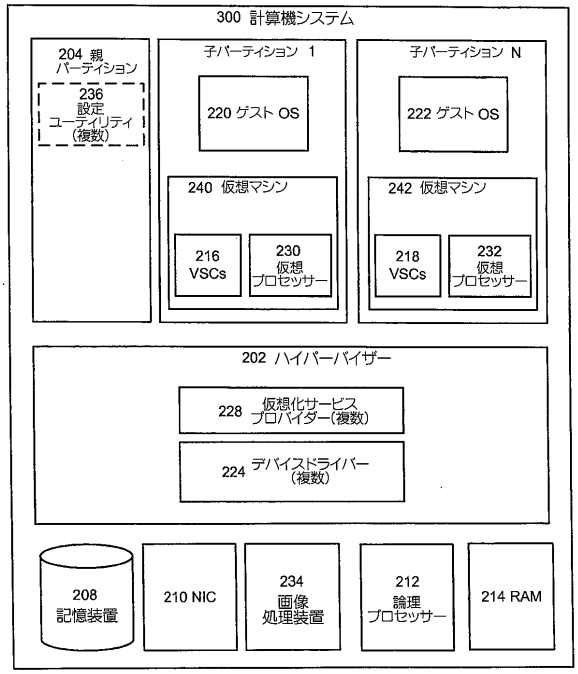
【図１】



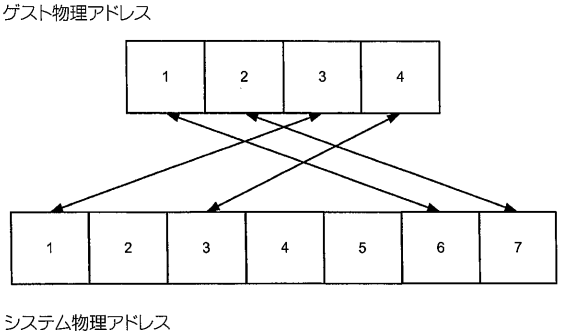
【図２】



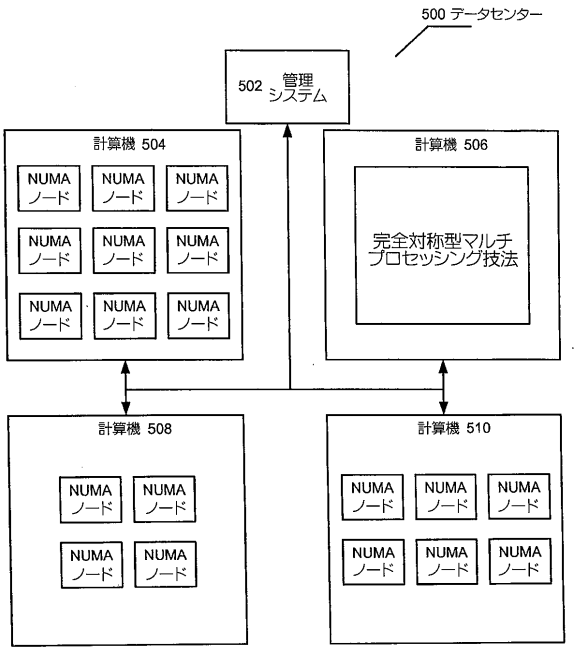
【図３】


300 計算機システム
204 親パーティション
236 設定ユーティリティ（複数）
子パーティション 1
220 ゲスト OS
240 仮想マシン
216 VSCs
230 仮想プロセッサー
子パーティション N
222 ゲスト OS
242 仮想マシン
218 VSCs
232 仮想プロセッサー
202 ハイパーバイザー
228 仮想化サービスプロバイダー（複数）
224 デバイスドライバー（複数）
208 記憶装置
210 NIC
234 画像処理装置
212 論理プロセッサー
214 RAM

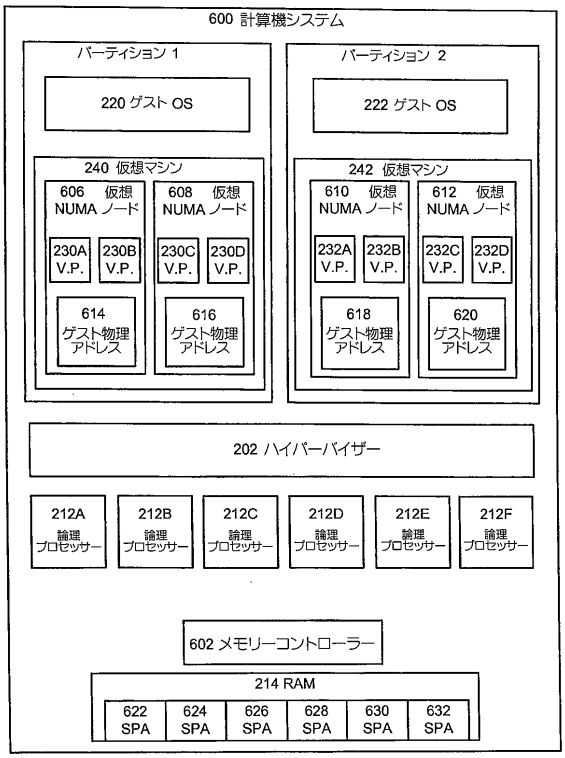
【図４】


ゲスト物理アドレス
1
2
3
4
1
2
3
4
5
6
7
システム物理アドレス

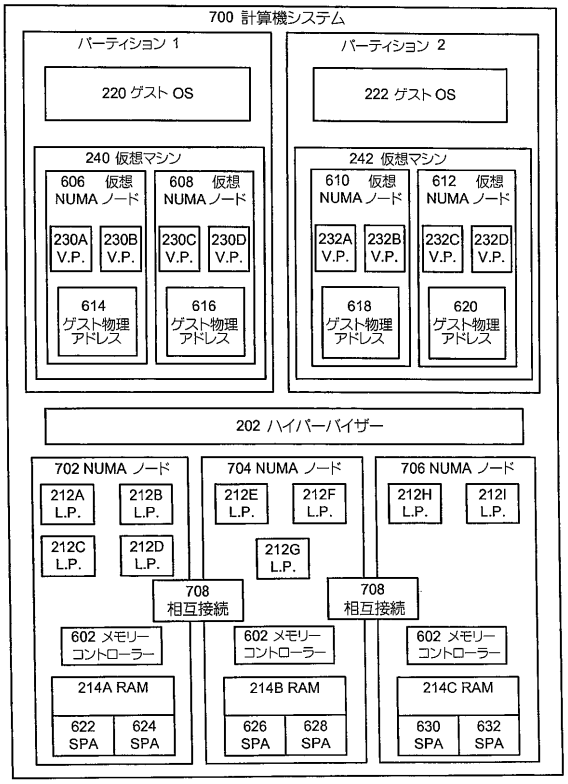
【図５】


500 データセンター
502 管理システム
計算機 504
NUMA ノード
計算機 506
完全対称型マルチプロセッシング技法
計算機 508
計算機 510

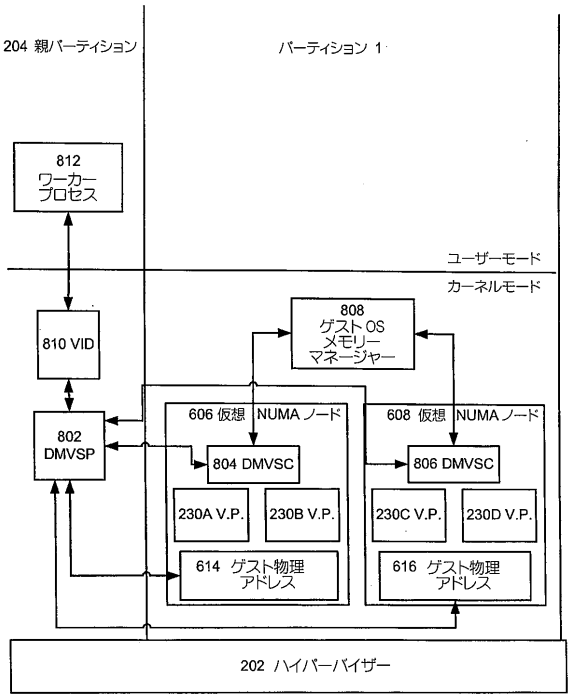
【図６】


600 計算機システム
パーティション 1
220 ゲスト OS
240 仮想マシン
606 仮想 NUMA ノード
608 仮想 NUMA ノード
230A V.P.
230B V.P.
230C V.P.
230D V.P.
614 ゲスト物理アドレス
616 ゲスト物理アドレス
パーティション 2
222 ゲスト OS
242 仮想マシン
610 仮想 NUMA ノード
612 仮想 NUMA ノード
232A V.P.
232B V.P.
232C V.P.
232D V.P.
618 ゲスト物理アドレス
620 ゲスト物理アドレス
202 ハイパーバイザー
212A 論理プロセッサー
212B 論理プロセッサー
212C 論理プロセッサー
212D 論理プロセッサー
212E 論理プロセッサー
212F 論理プロセッサー
602 メモリーコントローラー
214 RAM
622 SPA
624 SPA
626 SPA
628 SPA
630 SPA
632 SPA

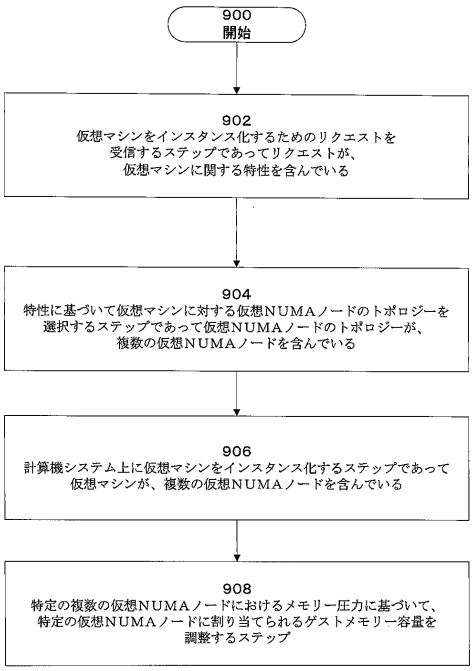
【図７】


700 計算機システム
パーティション 1
パーティション 2
220 ゲスト OS
222 ゲスト OS
240 仮想マシン
242 仮想マシン
606 仮想 NUMA ノード
608 仮想 NUMA ノード
610 仮想 NUMA ノード
612 仮想 NUMA ノード
230A V.P.
230B V.P.
230C V.P.
230D V.P.
232A V.P.
232B V.P.
232C V.P.
232D V.P.
614 ゲスト物理アドレス
616 ゲスト物理アドレス
618 ゲスト物理アドレス
620 ゲスト物理アドレス
202 ハイパーバイザー
702 NUMA ノード
704 NUMA ノード
706 NUMA ノード
212A L.P.
212B L.P.
212C L.P.
212D L.P.
212E L.P.
212F L.P.
212G L.P.
212H L.P.
212I L.P.
708 相互接続
708 相互接続
602 メモリーコントローラー
602 メモリーコントローラー
602 メモリーコントローラー
214A RAM
214B RAM
214C RAM
622 SPA
624 SPA
626 SPA
628 SPA
630 SPA
632 SPA

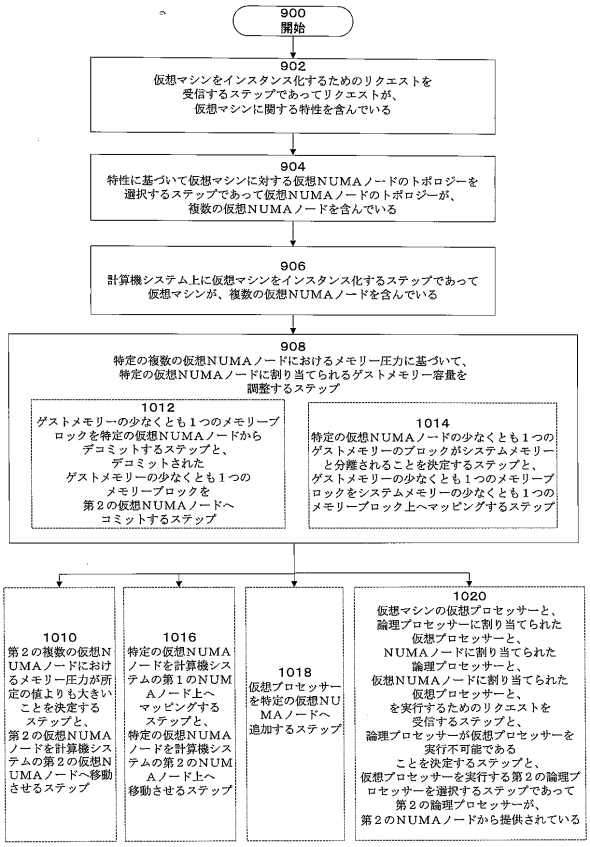
【図８】


204 親パーティション
パーティション 1
812 ワーカープロセス
ユーザーモード
カーネルモード
808 ゲスト OS メモリーマネージャー
810 VID
802 DMVSP
606 仮想 NUMA ノード
608 仮想 NUMA ノード
804 DMVSC
806 DMVSC
230A V.P.
230B V.P.
230C V.P.
230D V.P.
614 ゲスト物理アドレス
616 ゲスト物理アドレス
202 ハイパーバイザー

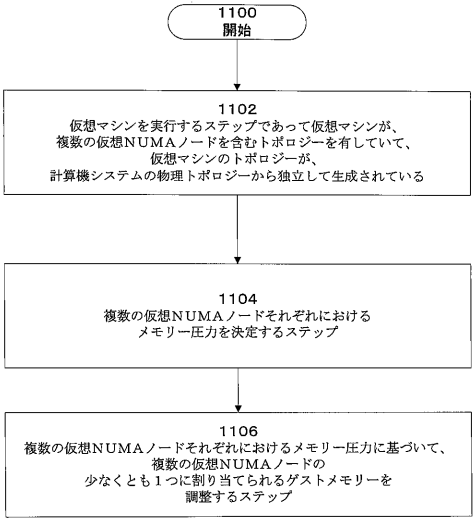
【図９】


900
開始
902
仮想マシンをインスタンス化するためのリクエストを受信するステップであってリクエストが、仮想マシンに関する特性を含んでいる
904
特性に基づいて仮想マシンに対する仮想NUMAノードのトポロジーを選択するステップであって仮想NUMAノードのトポロジーが、複数の仮想NUMAノードを含んでいる
906
計算機システム上に仮想マシンをインスタンス化するステップであって仮想マシンが、複数の仮想NUMAノードを含んでいる
908
特定の複数の仮想NUMAノードにおけるメモリー圧力に基づいて、特定の仮想NUMAノードに割り当てられるゲストメモリー容量を調整するステップ

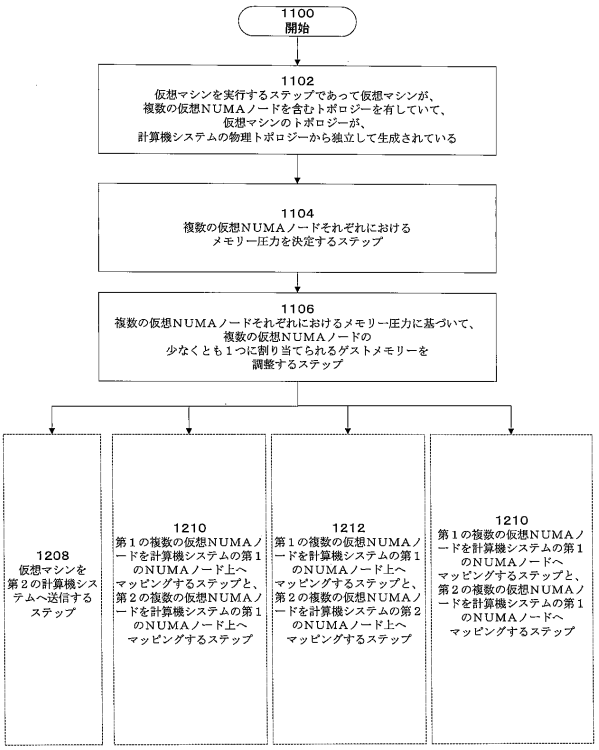
【図１０】


900
開始
902
仮想マシンをインスタンス化するためのリクエストを受信するステップであってリクエストが、仮想マシンに関する特性を含んでいる
904
特性に基づいて仮想マシンに対する仮想NUMAノードのトポロジーを選択するステップであって仮想NUMAノードのトポロジーが、複数の仮想NUMAノードを含んでいる
906
計算機システム上に仮想マシンをインスタンス化するステップであって仮想マシンが、複数の仮想NUMAノードを含んでいる
908
特定の複数の仮想NUMAノードにおけるメモリー圧力に基づいて、特定の仮想NUMAノードに割り当てられるゲストメモリー容量を調整するステップ
1012
ゲストメモリーの少なくとも１つのメモリーブロックを特定の仮想NUMAノードからデコミットするステップと、デコミットされたゲストメモリーの少なくとも１つのメモリーブロックを第２の仮想NUMAノードへコミットするステップ
1014
特定の仮想NUMAノードの少なくとも１つのゲストメモリーのブロックがシステムメモリーと分離されることを決定するステップと、ゲストメモリーの少なくとも１つのメモリーブロックをシステムメモリーの少なくとも１つのメモリーブロック上へマッピングするステップ
1010
第２の複数の仮想NUMAノードにおけるメモリー圧力が所定の値よりも大きいことを決定するステップと、第２の仮想NUMAノードを計算機システムの第２の仮想NUMAノードへ移動させるステップ
1016
特定の仮想NUMAノードを計算機システムの第１のNUMAノード上へマッピングするステップと、特定の仮想NUMAノードを計算機システムの第２のNUMAノード上へ移動させるステップ
1018
仮想プロセッサーを特定の仮想NUMAノードへ追加するステップ
1020
仮想マシンの仮想プロセッサーと、論理プロセッサーに割り当てられた仮想プロセッサーと、NUMAノードに割り当てられた論理プロセッサーと、仮想NUMAノードに割り当てられた仮想プロセッサーと、を実行するためのリクエストを受信するステップと、論理プロセッサーが仮想プロセッサーを実行不可能であることを決定するステップと、仮想プロセッサーを実行する第２の論理プロセッサーを選択するステップであって第２の論理プロセッサーが、第２のNUMAノードから提供されている

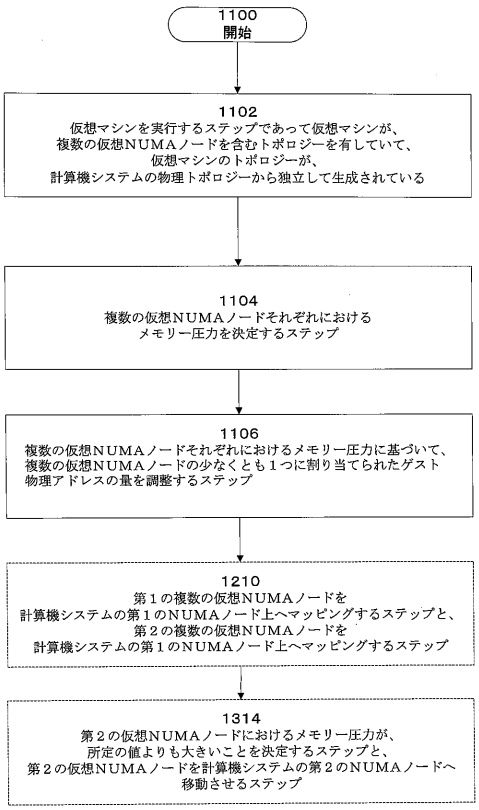
【図１１】



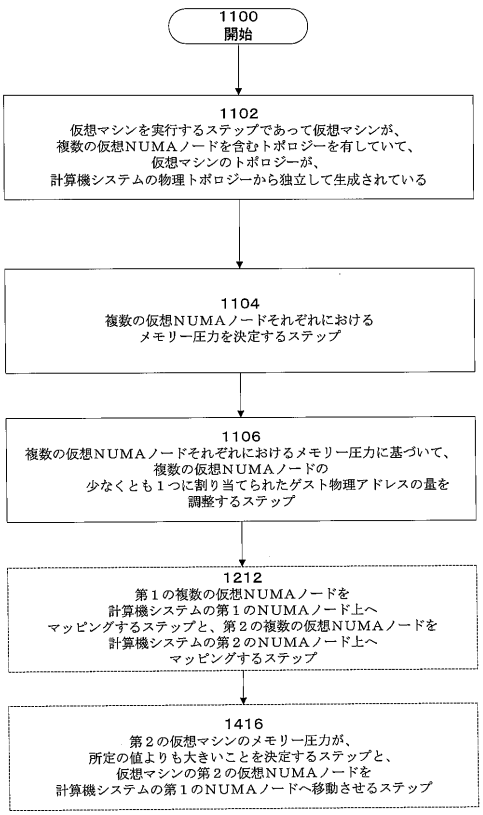
【図１２】



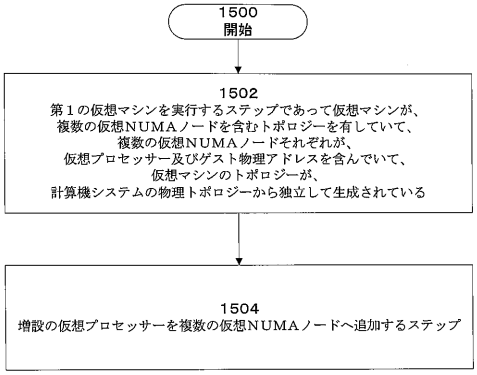
【図１３】



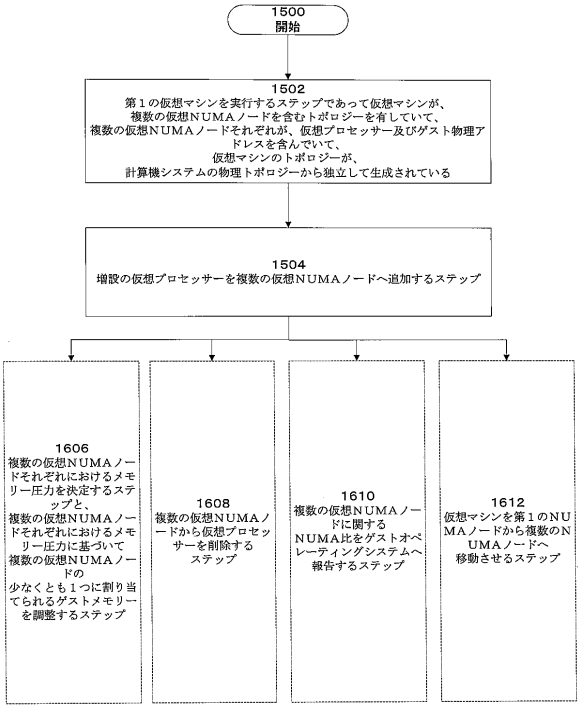
【図１４】



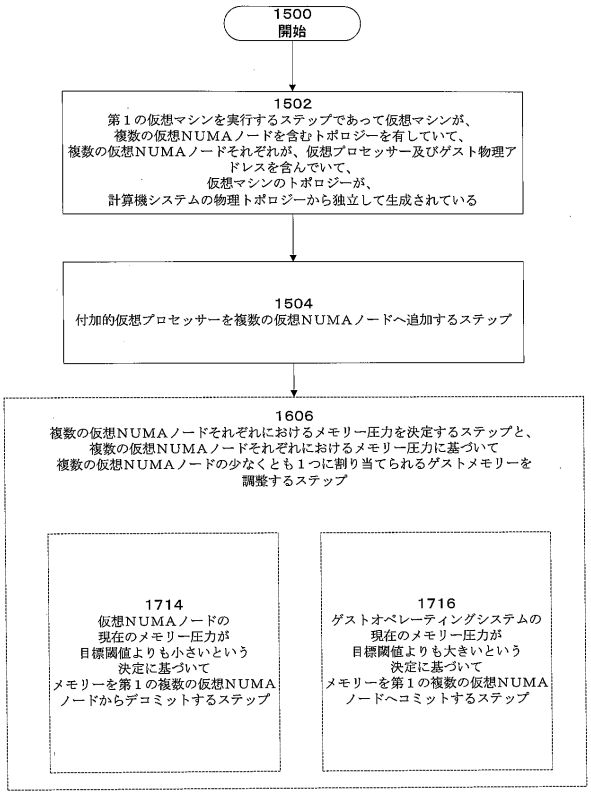
【図１５】


1500
開始
1502
第１の仮想マシンを実行するステップであって仮想マシンが、
複数の仮想ＮＵＭＡノードを含むトポロジーを有していて、
複数の仮想ＮＵＭＡノードそれぞれが、
仮想プロセッサー及びゲスト物理アドレスを含んでいて、
仮想マシンのトポロジーが、
計算機システムの物理トポロジーから独立して生成されている
1504
増設の仮想プロセッサーを複数の仮想ＮＵＭＡノードへ追加するステップ

【図１６】


1500
開始
1502
第１の仮想マシンを実行するステップであって仮想マシンが、
複数の仮想ＮＵＭＡノードを含むトポロジーを有していて、
複数の仮想ＮＵＭＡノードそれぞれが、仮想プロセッサー及びゲスト物理アドレスを含んでいて、
仮想マシンのトポロジーが、
計算機システムの物理トポロジーから独立して生成されている
1504
増設の仮想プロセッサーを複数の仮想ＮＵＭＡノードへ追加するステップ
1606
複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するステップと、
複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力に基づいて
複数の仮想ＮＵＭＡノードの
少なくとも１つに割り当てられるゲストメモリーを調整するステップ
1608
複数の仮想ＮＵＭＡノードから仮想プロセッサーを削除するステップ
1610
複数の仮想ＮＵＭＡノードに関するＮＵＭＡ比をゲストオペレーティングシステムへ報告するステップ
1612
仮想マシンを第１のＮＵＭＡノードから複数のＮＵＭＡノードへ移動させるステップ

【図１７】


1500
開始
1502
第１の仮想マシンを実行するステップであって仮想マシンが、
複数の仮想ＮＵＭＡノードを含むトポロジーを有していて、
複数の仮想ＮＵＭＡノードそれぞれが、仮想プロセッサー及びゲスト物理アドレスを含んでいて、
仮想マシンのトポロジーが、
計算機システムの物理トポロジーから独立して生成されている
1504
付加的仮想プロセッサーを複数の仮想ＮＵＭＡノードへ追加するステップ
1606
複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力を決定するステップと、
複数の仮想ＮＵＭＡノードそれぞれにおけるメモリー圧力に基づいて
複数の仮想ＮＵＭＡノードの少なくとも１つに割り当てられるゲストメモリーを
調整するステップ
1714
仮想ＮＵＭＡノードの
現在のメモリー圧力が
目標閾値よりも小さいという
決定に基づいて
メモリーを第１の複数の仮想ＮＵＭＡ
ノードからデコミットするステップ
1716
ゲストオペレーティングシステムの
現在のメモリー圧力が
目標閾値よりも大きいという
決定に基づいて
メモリーを第１の複数の仮想ＮＵＭＡ
ノードへコミットするステップ

フロントページの続き

(72)発明者　オシンズ，ジェイコブ
アメリカ合衆国ワシントン州９８０５２－６３９９，レッドモンド，ワン・マイクロソフト・ウェイ，マイクロソフト　コーポレーション，エルシーエイ－インターナショナル・パテンツ

審査官　篠塚　隆

(56)参考文献　特開２００６－１７８９３３（ＪＰ，Ａ）
特開２００７－２５７０９７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４６
９／４８
９／５０－９／５２
９／５４