(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4864817号
(P4864817)
(45)発行日　平成24年2月1日(2012.2.1)　(24)登録日　平成23年11月18日(2011.11.18)

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

請求項の数 16 (全 38 頁)

(21)出願番号　特願2007-164892(P2007-164892)
(22)出願日　平成19年6月22日(2007.6.22)
(65)公開番号　特開2009-3749(P2009-3749A)
(43)公開日　平成21年1月8日(2009.1.8)
審査請求日　平成22年2月15日(2010.2.15)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100075513
弁理士　後藤　政喜
(74)代理人　100114236
弁理士　藤井　正弘
(74)代理人　100120260
弁理士　飯田　雅昭
(72)発明者　森木　俊臣
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所　中央研究所内
(72)発明者　服部　直也
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所　中央研究所内
最終頁に続く

(54)【発明の名称】仮想化プログラム及び仮想計算機システム

(57)【特許請求の範囲】
【請求項１】
物理プロセッサとメモリを備えた物理計算機で実行されて、複数の仮想プロセッサを提供する仮想化プログラムにおいて、
第１の仮想マシンマネージャが第１の仮想プロセッサを生成する手順と、
前記第１の仮想プロセッサ上で実行される第２の仮想マシンマネージャが第２の仮想プロセッサを生成する手順と、
前記第２の仮想プロセッサがユーザプログラムを実行する手順と、
前記第１の仮想マシンマネージャが、前記物理プロセッサから当該第１の仮想マシンマネージャの呼び出しを受信する手順と、
前記第１の仮想マシンマネージャが、前記第１の仮想マシンマネージャ呼び出しの要因を解析する手順と、
前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順と、
前記第１の仮想マシンマネージャが、前記判定の結果に基づいて前記第１の仮想プロセッサに前記第２の仮想マシンマネージャまたは前記ユーザプログラムの一方を実行させる手順と、
を前記物理プロセッサに実行させることを特徴とする仮想化プログラム。
【請求項２】
前記第１の仮想マシンマネージャが、前記ユーザプログラムを実行する際の第１の仮想

プロセッサの状態を規定する第１の制御情報を前記メモリに設定する手順と、
前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャを実行する際の前記第１の仮想プロセッサの状態を規定する第２の制御情報を前記メモリに設定する手順と、
前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャまたは前記ユーザプログラムを実行する際の前記物理プロセッサの状態を規定する第３の制御情報を前記メモリに設定する手順と、
をさらに含み、
前記第１の仮想マシンマネージャが、判定の結果に基づいて前記第１の仮想プロセッサに前記第２の仮想マシンマネージャまたは前記ユーザプログラムの一方を実行させる手順は、
前記第１の仮想マシンマネージャが、前記第１の仮想プロセッサに前記ユーザプログラムの実行を開始させるときには、前記第１の制御情報を参照する手順と、
前記第１の仮想マシンマネージャが、前記第１の仮想プロセッサに前記第２の仮想マシンマネージャの実行を開始させるときには、前記第２の制御情報を参照する手順と、
前記第１の仮想マシンマネージャが、前記参照した前記第１の制御情報または第２の制御情報の一方で前記第３の制御情報を更新する手順と、
前記第１の仮想マシンマネージャが、前記物理プロセッサに対して前記第３の制御情報に従って命令の実行を開始するよう指示する第１の制御命令を発行する手順と、
を含むことを特徴とする請求項１に記載の仮想化プログラム。

【請求項３】
前記物理プロセッサは、インテル・アーキテクチャー３２の命令セットに準拠するプロセッサであって、
前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順は、
前記呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＶＭ－ｅｎｔｒｙ命令の発行である場合に、前記ユーザプログラムの実行開始と判定し、
前記呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対してＶＭ－ｅｘｉｔを通知する場合に、前記第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項２に記載の仮想化プログラム。

【請求項４】
前記物理プロセッサは、ＡＭＤ６４の命令セットに準拠するプロセッサであって、
前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順は、
前記呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＶＭＲＵＮ命令の発行である場合に、前記ユーザプログラムの実行開始と判定し、
前記呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対して＃ＶＭＥＸＩＴを通知する場合に、前記第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項２に記載の仮想化プログラム。

【請求項５】
前記物理プロセッサは、アイテニウム・プロセッサ・ファミリの命令セットに準拠するプロセッサであって、
前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順は、
前記呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥ＿ＨＡＮＤＬＥＲプロシジャ呼び出しの発行である場合に、前記ユーザプログラムの実行開始と判定し、
前記呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対して仮想化例外を通知する場合に、前記第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項２に記載の仮想化プログラム。

【請求項６】
前記第１の仮想マシンマネージャが第１の仮想プロセッサを生成する手順は、
前記第１の仮想プロセッサを複数生成する手順を含み、
前記物理プロセッサの状態を規定する第３の制御情報を前記メモリに設定する手順は、
前記第１の仮想プロセッサ毎に、前記第３の制御情報をそれぞれ前記メモリに設定する手順を含み、
前記第１の仮想マシンマネージャが、前記物理プロセッサに対して前記第３の制御情報に従って命令の実行を開始するよう指示する第１の制御命令を発行する手順は、
前記複数の第３の制御情報のうちの一つを選択する第２の制御命令を発行する手順を含むことを特徴とする請求項２に記載の仮想化プログラム。
【請求項７】
前記第２の仮想マシンマネージャが、前記第１の仮想プロセッサで前記第１の制御命令を受付可能か否かを確認する第３の制御命令を発行する手順をさらに含み、
前記第１の仮想マシンマネージャが、前記第１の仮想マシンマネージャ呼び出しの要因を解析する手順は、
前記第１の仮想マシンマネージャ呼び出しの要因が、前記第２の仮想マシンマネージャからの前記第３の制御命令であるときに、予め設定された応答可否情報を参照する手順と、
前記応答可否情報に応じて、前記第３の制御命令に対する応答を決定する手順と、
を含むことを特徴とする請求項２に記載の仮想化プログラム。
【請求項８】
前記第１の仮想マシンマネージャが、前記第１の仮想プロセッサに対する設定要求を受信する手順と、
前記第１の仮想マシンマネージャは、前記設定要求に基づいて前記応答可否情報を設定する手順と、
をさらに含むことを特徴とする請求項７に記載の仮想化プログラム。
【請求項９】
前記物理プロセッサは、インテル・アーキテクチャー３２の命令セットに準拠するプロセッサであって、
前記第３の制御命令は、ＣＰＵＩＤ命令であることを特徴とする請求項７に記載の仮想化プログラム。
【請求項１０】
前記第２の仮想マシンマネージャが、前記第１の仮想マシンマネージャから前記第３の制御命令に対する応答を受信する手順と、
前記第２の仮想マシンマネージャが、前記第１の制御命令が前記第１の仮想プロセッサにて受付可能である場合に、前記第１の仮想プロセッサに対して前記第１の制御命令を受付可能に設定する命令を発行する手順と、
を含むことを特徴とする請求項９に記載の仮想化プログラム。
【請求項１１】
前記第１の仮想マシンマネージャが、前記ユーザプログラムを実行する際の第１の仮想プロセッサの状態を規定する第１の制御情報を前記メモリに設定する手順と、
前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャを実行する際の前記第１の仮想プロセッサの状態を規定する第２の制御情報を前記メモリに設定する手順と、
前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャまたは前記ユーザプログラムを実行する際の前記物理プロセッサの状態を規定する第３の制御情報を前記メモリに設定する手順と、
をさらに含み、
前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順は、

前記解析の結果が、前記第２の仮想マシンマネージャから前記第１の制御情報に対して前記第１の仮想プロセッサの動作状態を反映させる第４の制御命令を第１の仮想マシンマネージャ呼び出しの要因のときに、
前記第１の仮想マシンマネージャが前記物理プロセッサに対して、前記第３の制御情報に前記物理プロセッサの動作状態を反映させる第４の制御命令を発行し、
前記第１の仮想マシンマネージャが前記物理プロセッサの動作状態を反映した前記第３の制御情報を参照し、
前記第１の仮想マシンマネージャが、前記参照した第３の制御情報に基づいて、前記第１の制御情報を更新した後に、前記第２の仮想マシンマネージャの実行を判定することを特徴とする請求項１に記載の仮想化プログラム。

【請求項１２】
前記物理プロセッサは、インテル・アーキテクチャー３２の命令セットに準拠するプロセッサであって、
前記第４の制御命令がＶＭＣＬＥＡＲ命令であることを特徴とする請求項１１に記載の仮想化プログラム。

【請求項１３】
物理プロセッサとメモリを備えた物理計算機上で複数の仮想プロセッサを提供する仮想計算機システムにおいて、
前記物理プロセッサ上で動作して、第１の仮想プロセッサを提供する第１の仮想マシンマネージャと、
前記第１の仮想プロセッサ上で動作して、ユーザプログラムを実行する第２の仮想プロセッサを提供する第２の仮想マシンマネージャと、を備え、
前記第２の仮想マシンマネージャは、
前記ユーザプログラムを実行する際の第１の仮想プロセッサの状態を規定する第１の制御情報と、前記第２の仮想マシンマネージャを実行する際の前記第１の仮想プロセッサの状態を規定する第２の制御情報とを格納する仮想プロセッサ制御データを有し、
前記第１の仮想マシンマネージャは、
前記第２の仮想マシンマネージャまたは前記ユーザプログラムを実行する際の前記物理プロセッサの状態を規定する第３の制御情報を前記メモリに格納する物理プロセッサ制御データと、
前記物理プロセッサから当該第１の仮想マシンマネージャの呼び出しを受信する受信部と、
前記第１の仮想マシンマネージャ呼び出しの要因を解析し、前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する判定部と、
前記判定の結果に基づいて、前記仮想プロセッサ制御データの第１の制御情報と第２の制御情報の何れかを選択する選択部と、
前記選択された第１の制御情報または第２の制御情報で、前記物理プロセッサ制御データの第３の制御情報を更新する更新部と、
前記第３の制御情報に基づいて、前記第１の仮想プロセッサに前記第２の仮想マシンマネージャまたは前記ユーザプログラムの一方を実行させる命令を発行する命令発行部と、
を有することを特徴とする仮想計算機システム。

【請求項１４】
前記物理プロセッサは、インテル・アーキテクチャー３２の命令セットに準拠するプロセッサであって、
前記判定部は、
前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＶＭ－ｅｎｔｒｙ命令の発行である場合に、前記ユーザプログラムの実行開始と判定し、
前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対してＶＭ－ｅｘｉｔを通知する場合に、前記

第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項１３に記載の仮想計算機システム。

【請求項１５】

前記物理プロセッサは、ＡＭＤ６４の命令セットに準拠するプロセッサであって、

前記判定部は、

前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＶＭＲＵＮ命令の発行である場合に、前記ユーザプログラムの実行開始と判定し、

前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対して＃ＶＭＥＸＩＴを通知する場合に、前記第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項１３に記載の仮想計算機システム。

【請求項１６】

前記物理プロセッサは、アイテニウム・プロセッサ・ファミリの命令セットに準拠するプロセッサであって、

前記判定部は、

前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第２の仮想マシンマネージャからのＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥ＿ＨＡＮＤＬＥＲプロシジャ呼び出しの発行である場合に、前記ユーザプログラムの実行開始と判定し、

前記第１の仮想マシンマネージャの呼び出し要因の解析の結果が、前記第１の仮想プロセッサから第２の仮想マシンマネージャに対して仮想化例外を通知する場合に、前記第２の仮想マシンマネージャの実行開始と判定することを特徴とする請求項１３に記載の仮想計算機システム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、仮想計算機システムに関し、特に、仮想化支援機能を備えたプロセッサを用いた仮想計算機システムに関する。

【背景技術】

【０００２】

近年、オープンサーバの普及に伴い多数のサーバが企業の情報システムに導入された。特に価格性能比が高いｉＡ（Intel Architectures）－３２サーバの乱立は、電力やハードウェアの保守費用などを含むサーバの運用管理コストを増大させており、サーバを運用する企業において問題となっている。

【０００３】

サーバの運用管理コストを低減するため、複数台のサーバを１台の物理的なサーバ（物理サーバ）に集約するサーバ統合が有望である。サーバ統合の実現方法としては、計算機資源を仮想化する機能を提供する仮想化ソフトウェアが注目されている。仮想化ソフトウェアは１台の物理サーバのＣＰＵ（プロセッサ）やＩ／Ｏ等の計算機資源を分割し、複数の仮想的なサーバ（仮想サーバ〉に分割した計算機資源を割付ける制御ソフトウェアである。各仮想サーバ上では各々１つのＯＳ（ゲストＯＳ）が稼動できる。仮想化ソフトウェアを用いると、従来、複数の物理的なサーバで稼動していたＯＳやアプリケーションを各仮想サーバに割当てて、一つの物理計算機で複数のサーバを提供するサーバ統合を実現することができる。

【０００４】

仮想化ソフトウェアにおける仮想サーバへの計算機資源の割付方針について説明する。計算機資源のうちＣＰＵの割付について、ｉＡ－３２向けの仮想化ソフトウェアではＶＴ－ｘ（Virtualization Technology for Xeon）機能（または、ＡＭＤ－Ｖ）等の仮想化支援機能を備えたプロセッサを用いる方法が主流である。ＶＴ－ｘは仮想化ソフトウェアとゲストＯＳに異なる動作権限を割当てる機能であり、プロセッサのハードウェアに実装さ

れている（例えば、特許文献１または非特許文献１、２）。ＶＴ－ｘに対応したＣＰＵは、ゲストＯＳ間と仮想化ソフトウェアの動作権限の移行を検出してＣＰＵのレジスタ状態の退避や回復を行い、仮想サーバ毎に独立した動作環境を提供する。

【０００５】

一方、Ｉ／Ｏの割付方針は仮想化ソフトウェアによって異なる。仮想化ソフトウェアのＩ／Ｏの割り付けは、

（１）物理サーバのＩ／Ｏデバイスを直接使用させるＩ／Ｏ直接割付型と、

（２）物理サーバのＩ／Ｏデバイスの種類やリビジョンを隠蔽する仮想Ｉ／Ｏ割付型と、

に大別される。

【０００６】

上記（１）のＩ／Ｏ直接割付型は、物理サーバで動作済みのＩ／Ｏについて、ファイルシステムの再構築等を行うことなく容易にサーバ統合を実現できる長所を有する。一方、上記（２〉の仮想Ｉ／Ｏ割付型では、物理サーバのＩ／Ｏ種類に依らず一定の１／Ｏ構成をゲストＯＳに提供できる長所を有する。

【０００７】

ここで、上述のような仮想化ソフトウェアとしては、次のようなものが知られている。まず、ｉＡ－３２サーバ向けの仮想化ソフトウェアの１つとして、ＶＭｗａｒｅ（登録商標）のＥＳＸ　Ｓｅｒｖｅｒが知られている。これは、ｉＡ－３２のＣＰＵを含む物理サーバで、上述したＶＴ－ｘ機能を利用して、既存のＯＳを複数動作させることが可能である。

【０００８】

汎用機の技術に基づく仮想化ソフトウェアとしては、論理分割運転機能が知られている（ＩＢＭ　Ｓｙｓｔｅｍ３７０等）。これは、単一の物理計算機を複数の論理区画（ＬＰＡＲ）に分割し、各ＬＰＡＲ上で既存のＯＳ及び仮想化管理部（ＶＭＭ＝Virtual Machine Manager）を実行する。論理分割運転機能では、汎用機（物理計算機）が上記ｉＡ－３２のＶＴ－ｘに相当する機能（ＬＰＡＲモード）を用いて、仮想サーバ（ＬＰＡＲ）上での従来のＯＳおよびＶＭＭを稼動させている。

【０００９】

この他、シミュレータによってｉＡ－３２のＶＴ－ｘ機能を提供するものとしてＳｉｍＯＳ（http://simos.stanford.edu/introduction.html）等が知られている。この種のシミュレータは、仮想サーバ上のゲストＯＳの命令列を解釈して任意のサーバおよびＣＰＵ機能を提供するソフトウェアである。

【００１０】

さらに、ｉＡ－６４（ＩＰＦ＝Ｉｔａｎｉｕｍ　Ｐｒｏｃｅｓｓｏｒ　Ｆａｍｉｌｙ）のプロセッサに実装された仮想化支援機能としては、非特許文献３に記載されるＶＴ－ｉ機能が知られている。

【特許文献１】特表２００５－５２９４０１号公報

【非特許文献１】「Intel 64 and IA-32 Architectures Software Developer’s Manual VOL 2B」、[online]、Intel corp. 発行、[平成１９年５月１日検索]、インターネット＜ＵＲＬ：http://www.intel.com/design/processor/manuals/253667.pdf＞

【非特許文献２】「AMD-Virtualization（AMD-V）」、[online]、Advanced Micro Devices, Inc. 発行、[平成１９年５月１日検索]、インターネット＜ＵＲＬ：http://www.amd.com/jp-ja/assets/content_type/DownloadableAssets/jp-ja_AMD-V_general_200702.pdf＞

【非特許文献３】「Intel Itanium Architecture Software Developer's Manual」、[online]、Intel corp. 発行、[平成１９年５月１日検索]、インターネット＜ＵＲＬ：ftp://download.intel.com/design/Itanium/manuals/24531705.pdf＞

【発明の開示】

【発明が解決しようとする課題】

【００１１】

ところで、近年のサーバ統合の要求から、次世代のサーバＯＳ（Windows（登録商標） Server Longhorn またはWindows（登録商標） Server 2008等）では上述の（２）のような仮想Ｉ／Ｏ割付型の仮想化ソフトウェアの機能を搭載することが検討されている。

【００１２】

しかしながら、上記従来の仮想化ソフトウェアでは、次のような問題がある。

【００１３】

上記ＥＳＸサーバ等の従来の仮想化ソフトウェアでは、物理計算機を構成するＣＰＵの仮想化支援（ＶＴ－ｘ）機能を用いて、複数の仮想サーバ上で既存のＯＳを稼動させることは可能であるが、仮想サーバに対してＶＴ－ｘ機能を提供することはできない。このため、ＥＳＸサーバなどでは次世代のサーバＯＳのように仮想化ソフトウェア（仮想化機能）を統合したＯＳを実行することが難しい、という問題がある。

【００１４】

また、上記論理分割運転機能では、ＬＰＡＲ上のＯＳやＶＭＭがＬＰＡＲモードを利用することはできない。このため、仮想化ソフトウェアを統合したＯＳをＬＰＡＲ上で稼動させるのは難しい、という問題があった。

【００１５】

さらに、上記シミュレータの場合は、ＶＴ－ｘ機能を仮想サーバに対して提供することができ、仮想化ソフトウェアを統合した次世代のＯＳを稼動させることは可能である。しかしながら、上記シミュレータは、仮想サーバ（ゲストＯＳ）の命令列を解釈し、物理計算機のＣＰＵが実行可能な命令列に変換（バイナリトランスレーション）するためのオーバーヘッドが発生するため、仮想サーバの性能（処理能力）が低下する、という問題があり、複数の仮想サーバを稼働させるサーバ統合を行うことは現実的ではない。

【００１６】

また、仮想Ｉ／Ｏ割り付け型の仮想化機能を統合した次世代ＯＳでは、開発環境の提供など同一のＩ／Ｏ構成の仮想サーバを多数生成する用途には向いているが、既存のサーバで動作済みの従来のＯＳ（ＶＴ－ｘ機能を利用しないＯＳ＝ＮＴサーバ等）を統合する用途には不向きである。特に、現在、企業が運用するソフトウェア資産は、ほとんどが仮想化機能を含まないＯＳで稼動するソフトウェアである。このため、今後、サーバ統合を進めていく過程では、仮想化機能を含まない既存のＯＳと、新たに導入する仮想化機能を含んだ次世代ＯＳが混在するサーバ統合環境が必要となる。しかし、上述のように、従来例では仮想化機能を統合した次世代ＯＳを仮想計算機上に統合することは難しく、また、次世代ＯＳで既存のＯＳを統合することは難しい、という問題があった。

【００１７】

そこで本発明は、上記問題点に鑑みてなされたもので、仮想計算機の性能を低下させることなく、仮想化機能を統合したＯＳを仮想サーバで実行可能な仮想計算機を提供することを目的とし、さらに、仮想化機能を統合した次世代のＯＳと、プロセッサの仮想化支援機能を利用しない従来のＯＳを一つの物理計算機に統合することを目的とする。

【課題を解決するための手段】

【００１８】

本発明は、物理プロセッサとメモリを備えた物理計算機で実行されて、複数の仮想プロセッサを提供する仮想化プログラムにおいて、第１の仮想マシンマネージャが第１の仮想プロセッサを生成する手順と、前記第１の仮想プロセッサ上で実行される第２の仮想マシンマネージャが第２の仮想プロセッサを生成する手順と、前記第２の仮想プロセッサがユーザプログラムを実行する手順と、前記第１の仮想マシンマネージャが、前記物理プロセッサから当該第１の仮想マシンマネージャの呼び出しを受信する手順と、前記第１の仮想マシンマネージャが、前記第１の仮想マシンマネージャ呼び出しの要因を解析する手順と、前記第１の仮想マシンマネージャが、前記解析の結果に基づいて前記第２の仮想マシンマネージャと、前記ユーザプログラムのいずれを実行するかを判定する手順と、前記第１の仮想マシンマネージャが、前記判定の結果に基づいて前記第１の仮想プロセッサに前記第２の仮想マシンマネージャまたは前記ユーザプログラムの一方を実行させる手順と、を

前記物理プロセッサに実行させる。

【００１９】

また、前記第１の仮想マシンマネージャが、前記ユーザプログラムを実行する際の第１の仮想プロセッサの状態を規定する第１の制御情報を前記メモリに設定する手順と、前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャを実行する際の前記第１の仮想プロセッサの状態を規定する第２の制御情報を前記メモリに設定する手順と、前記第１の仮想マシンマネージャが、前記第２の仮想マシンマネージャまたは前記ユーザプログラムを実行する際の前記物理プロセッサの状態を規定する第３の制御情報を前記メモリに設定する手順と、をさらに含み、前記第１の仮想マシンマネージャが、判定結果に基づいて前記第１の仮想プロセッサに前記第２の仮想マシンマネージャまたは前記ユーザプログラムの一方を実行させる手順は、前記第１の仮想マシンマネージャが、前記第１の仮想プロセッサに前記ユーザプログラムの実行を開始させるときには、前記第１の制御情報を参照する手順と、前記第１の仮想マシンマネージャが、前記第１の仮想プロセッサに前記第２の仮想マシンマネージャの実行を開始させるときには、前記第２の制御情報を参照する手順と、前記第１の仮想マシンマネージャが、前記参照した前記第１の制御情報または第２の制御情報の一方で前記第３の制御情報を更新する手順と、前記第１の仮想マシンマネージャが、前記物理プロセッサに対して前記第３の制御情報に従って命令の実行を開始するよう指示する第１の制御命令を発行する手順と、を含む。

【発明の効果】

【００２０】

したがって、本発明は、第１の仮想マシンマネージャの呼び出しの要因に応じて、第２の仮想プロセッサ上のユーザプログラムを実行するための第１の制御情報と、第２の仮想マシンマネージャを実行するための第２の制御情報の何れかを選択し、物理プロセッサを制御するための第３の制御情報を更新することで、第１の仮想プロセッサ上で仮想化機能を備えた次世代ＯＳをユーザプログラムとして実行することが可能となる。また、ユーザプログラムと第２の仮想マシンマネージャの切り換えは、第１または第２の制御情報で第３の制御情報を更新すればよいので、仮想計算機の性能を低下させることなく次世代ＯＳの稼働を実現できる。

【００２１】

そして、第２の仮想マシンマネージャを持たない既存のＯＳと、第２の仮想マシンマネージャを含む次世代ＯＳを、ひとつの計算機に統合することが可能となる。

【発明を実施するための最良の形態】

【００２２】

以下、本発明の一実施形態を添付図面に基づいて説明する。

【００２３】

図１は、第１の実施形態を示し、本発明を適用する仮想計算機システムのブロック図である。物理サーバ（物理計算機）１０１は、仮想化支援機能を備えて演算処理を実行する物理ＣＰＵ（プロセッサ）１０４と、データやプログラムを格納するメモリ１０５と、物理サーバ１０１の外部の装置とデータの送受信を行うためのＩ／Ｏ装置１０６と、を含んで構成される。なお、Ｉ／Ｏ装置１０６は、ネットワークインターフェースやホストバスアダプタ等で構成される。また、物理ＣＰＵ１０４は、複数のＣＰＵで構成しても良いし、複数の演算コアを備えたＣＰＵで構成しても良い。また、物理ＣＰＵ１０４の仮想化支援機能（ＶＭＸ：Virtual Machine Extensions）は、上述のＶＴ－ｘ（Virtualization Technology for IA－32 Processors）またはＡＭＤ－Ｖ（AMD Virtualization technology）あるいはＶＴ－ｉ（Virtualization Technology for Itanium architecture processors）を含むものである。なお、仮想化支援機能を利用する動作モードをＶＭＸモードとし、仮想化支援機能を利用しない通常の特権レベルによる動作モードは通常動作モードとする。

【００２４】

物理サーバ１０１では、複数の仮想サーバ１０２ａ～１０２ｎを稼動させるため、物理

サーバ１０１の物理的な計算機資源を仮想化した計算機資源に変換し、各仮想サーバ１０２ａ～１０２ｎへ仮想計算機資源を割り当てるホストＶＭＭ（仮想マシンマネージャ：Virtual Machine Manager）１０が実行される。このホストＶＭＭは、メモリ１０５に読み込まれて物理ＣＰＵ１０４で実行されるプログラムとして提供される。ホストＶＭＭ１０は、各仮想サーバ１０２ａ～１０２ｎに対して、仮想ＣＰＵ１０８ａ～１０８ｎを提供し、また、メモリ１０５とＩ／Ｏ装置１０６を各仮想サーバ１０２ａ～１０２ｎに割り当てる。なお、各仮想サーバ１０２ａ～１０２ｎに、物理サーバ１０１の計算機資源を割り当てる手法については、周知または公知のものを適宜用いればよいので、ここでは詳述しない。

【００２５】

物理サーバ１０１は、ユーザーインターフェース３０１を提供する管理コンソール３００に接続され、管理者などが計算機資源の割り当てなどの設定をユーザーインターフェース３０１を介してホストＶＭＭ１０へ入力する。また、ユーザーインターフェース３０１は、ホストＶＭＭ１０から受信した設定状態等を管理コンソール３００の表示装置へ出力する。

【００２６】

物理サーバ１０１のホストＶＭＭ１０上で稼動する仮想サーバ１０２ａ～１０２ｎは、ユーザプログラム１１０ａ～１１０ｎとして、ゲストＯＳ１１１ａ～１１１ｎがそれぞれ稼動し、各ゲストＯＳ１１１ａ～１１１ｎ上ではアプリケーション１１２ａ～１１２ｎが実行されている。各ゲストＯＳ１１１ａ～１１１ｎは、ホストＶＭＭ１０が提供する仮想ＣＰＵ１０８ａ～１０８ｎで実行される。なお、仮想ＣＰＵ１０８ａ～１０８ｎは、一つの仮想サーバに対して複数の仮想ＣＰＵを割り当てるようにすることができる。

【００２７】

そして、仮想サーバ１０２ａでは、ゲストＯＳ１１１ａとして仮想化機能（ゲストＶＭＭ２０）を統合した次世代ＯＳが実行され、仮想サーバ１０２ｎでは、ゲストＯＳ１１１ｎとして仮想化機能を利用しない既存のＯＳ（例えば、ＮＴサーバ）が実行される。

【００２８】

ホストＶＭＭ１０は、既存のＯＳを実行する仮想サーバ１０２ｎに対しては、仮想ＣＰＵ１０８ｎと、管理コンソール３００から設定された計算機資源を割り当て、既存のゲストＯＳ１１１ｎと、アプリケーション１１２ｎを実行する。

【００２９】

一方、ホストＶＭＭ１０は、次世代ＯＳを実行する仮想サーバ１０２ａへ割り当てる仮想ＣＰＵ１０８ａに、仮想化支援機能を提供する。仮想ＣＰＵ１０８ａ上ではゲストＶＭＭ（第２仮想マシンマネージャ）２０が稼働し、このゲストＶＭＭ２０は仮想ＣＰＵ２０８ａ～２０８ｉを提供する。次世代ＯＳが稼働する仮想サーバ１０２ａでは、第１の仮想ＣＰＵ１０８ａ上で第２の仮想ＣＰＵ２０８ａ～２０８ｉが複数提供され、各仮想ＣＰＵ２０８ａ～２０８ｉでは、複数のユーザプログラム１１０ａ（ゲストＯＳ１１１ａ及びアプリケーション１１２ａ）～１１０ｉ（ゲストＯＳ１１１ｉ及びアプリケーション１１２ｉ）が実行される。

【００３０】

以下、本第１実施形態では、物理ＣＰＵ１０４にＶＴ－ｘ機能を備え、仮想サーバ１０２ａのゲストＯＳ１１１ａが仮想化機能を統合した次世代ＯＳの例について説明する。

【００３１】

ＶＴ－ｘ機能を利用するホストＶＭＭ１０は、メモリ１０５の所定の領域に仮想サーバ１０２ａ～１０２ｎの状態と物理ＣＰＵ１０４を制御するための制御情報を格納するホストＶＭＭ保持データ１１を格納する。そして、ホストＶＭＭ保持データ１１には、物理ＣＰＵ１０４を制御するための物理ＣＰＵ制御データ１３が格納される。物理ＣＰＵ制御データ１３は、仮想化支援機能を利用する仮想ＣＰＵ１０８ａ～１０８ｎのステータスを示すデータ構造で、ＶＭＣＢ（Ｖｉｒｔｕａｌ Ｍａｃｈｉｎｅ Ｃｏｎｔｒｏｌ Ｂｌｏｃｋ）またはＶＭＣＳ（Ｖｉｒｔｕａｌ Ｍａｃｈｉｎｅ Ｃｏｎｔｒｏｌ Ｓｔｒｕｃ

ｔｕｒｅ）という。なお、本実施形態では、ホストＶＭＭ保持データ１１内の物理ＣＰＵ制御データ１３を、図２で示すようにシャドウＶＭＣＢ_０～_ｎとし、ゲストＶＭＭ２０が操作する仮想ＣＰＵ制御データ２１をゲストＶＭＣＢとして区別する。また、物理ＣＰＵ１０４は、物理ＣＰＵ制御データ１３のシャドウＶＭＣＢ_０～_ｎを参照するためのポインタ１１５を備える。

【００３２】

ホストＶＭＭ１０は、物理ＣＰＵ制御データ１３内のシャドウＶＭＣＢ_０～_ｎを書き換えることで、物理ＣＰＵ１０４の動作モードを、ユーザプログラム１１０ａまたはゲストＶＭＭ２０を実行する動作モード（ＶＭＸノンルートモード）と、ホストＶＭＭ１０を実行する動作モード（ＶＭＸルートモード）の何れかに設定する。

【００３３】

仮想サーバ１０２ａでは、ホストＶＭＭ１０が提供する仮想ＣＰＵ１０８ａ上で、次世代ＯＳであるゲストＯＳ１１１ａに統合された仮想化機能（仮想化ソフトウェア）が、ゲストＶＭＭ２０として稼動する。ゲストＶＭＭ２０には、仮想ＣＰＵ１０８ａの仮想化支援機能を制御するためのＶＭＣＢ（ゲストＶＭＣＢ２２）を含む仮想ＣＰＵ制御データ２１が格納される。ゲストＶＭＣＢ２２は、ホストＶＭＭ１０によって割り当てられたメモリ１０５上の所定の領域に格納される。

【００３４】

また、仮想ＣＰＵ１０８ａは、ゲストＶＭＭ２０の仮想ＣＰＵ制御データ２１（ゲストＶＭＣＢ２２）を参照するためのポインタ１１６を備える。このポインタ１１６は、ＶＴ－ｘ機能に対応するゲストＯＳ１１１ａが保持する仮想ＣＰＵ１０８ａの制御データ構造（ゲストＶＭＣＢ２２）へのポインタであり、ゲストＯＳ１１１ａからのＶＭＰＴＲＬＤ命令発行時に初期化される。なお、ポインタ１１６の初期化は、ゲストＶＭＭ２０が実行する。

【００３５】

図１において、次世代ＯＳが稼動する仮想サーバ１０２ａの仮想ＣＰＵ２０８ａ～２０８ｉは、次世代ＯＳに統合されたゲストＶＭＭ２０が提供するものであり、各仮想ＣＰＵ２０８ａ～２０８ｉ上でユーザプログラム１１０ａ～１１０ｉを実行することができる。なお、仮想サーバ１０２ａのゲストＶＭＭ２０は、ゲストＯＳ１１１ａのアドインソフトウェアとして機能するものであっても良い。

【００３６】

＜本発明の概要＞

仮想化を支援するＶＴ－ｘ機能では、ホストＶＭＭ１０が物理サーバ１０１のメモリ１０５上に確保するシャドウＶＭＣＢを用いて、物理ＣＰＵ１０４の動作モードを制御する。仮想化支援機能としてＶＴ－ｘ機能を備えた物理ＣＰＵ１０４は、通常の動作モードと仮想化支援機能を提供するＶＭＸ（Virtual Machine Extensions）モードを備え、ＶＭＸモードでは、ホストＶＭＭ１０が動作するホストモード（以下、ＶＭＸルートモード）と、ゲストＶＭＭ２０またはユーザプログラム（ゲストＯＳ１１１ａまたはアプリケーション１１２ａ）が動作するゲストモード（以下、ＶＭＸノンルートモード）の何れかに切り替える。

【００３７】

物理ＣＰＵ制御データ１３のシャドウＶＭＣＢ中には仮想サーバ１０２ａ上のユーザプログラム１１０ａの動作状態を規定するフィールド（ゲスト状態エリア１３１）が１種類しかなく、単純には次世代ＯＳの仮想化機能であるゲストＶＭＭ２０と、ユーザプログラム１１０ａ（ゲストＯＳ１１１ａまたはアプリケーション１１２ａ）を区分できない。

【００３８】

そこで、本発明では、１つの仮想ＣＰＵが、ゲストＶＭＭ２０とユーザプログラム１１０ａを同時に実行することがないことに着目し、ホストＶＭＭ１０が仮想化機能を統合したゲストＯＳ１１１ａを実行する仮想サーバ１０２ａのゲストＶＭＭ２０と、ユーザプログラム１１０ａの実行の切替を監視して、動作モードの切替時にホストＶＭＭ保持データ

１１のシャドウＶＭＣＢ_０～_ｎ内のゲスト状態エリア１３１を書き換えることにより、仮想サーバ１０２ａ上で仮想化機能を稼動させるものである。

【００３９】

このため、ホストＶＭＭ１０は、仮想化機能を統合した次世代ＯＳであるゲストＯＳ１１１ａを監視して、物理ＣＰＵ１０４の動作モードを、ホストＶＭＭ１０が動作するＶＭＸルートモードと、ゲストＶＭＭ２０またはユーザプログラム１１０ａが動作するＶＭＸノンルートモードモードに切り替え、所定の条件（ＶＭ－ｅｘｉｔ）でＶＭＸルートモードではホストＶＭＭ１０がゲストＶＭＭ２０やゲストＯＳの命令をエミュレートする。これにより、ゲストＯＳ１１１ａに対して、仮想ＣＰＵ１０８ａが仮想化支援機能を提供するように見せかける。

【００４０】

＜ホストＶＭＭの構成＞

ホストＶＭＭ１０は、上述のホストＶＭＭ保持データ１１の他に、仮想サーバ１０２ａ～１０２ｎを監視して、物理ＣＰＵ１０４の動作モードをＶＭＸルートモードとＶＭＸノンルートモードの何れかに切り替えるＣＰＵ制御部１２を備える。

【００４１】

さらに、ホストＶＭＭ１０は、ＣＰＵ制御部１２が仮想サーバ１０２ａ～１０２ｎの状態を取得し、各仮想サーバ１０２ａ～１０２ｎへ指令を送信するためのインターフェースである制御・通信インターフェース１４と、ＣＰＵ制御部１２が物理ＣＰＵ１０４へ命令を発行するための命令発行インターフェース１５と、物理ＣＰＵ１０４が物理ＣＰＵ制御データ１３を参照または更新するための参照・更新インターフェース１６と、Ｉ／Ｏ装置１０６からの割り込み要求などを受け付けて、この要求に応答するためのＩ／Ｏ要求・応答インターフェース１７を備える。

【００４２】

図２は、ホストＶＭＭ１０とゲストＶＭＭ２０の機能を示すブロック図である。ホストＶＭＭ１０は、物理ＣＰＵ１０４の仮想化支援機能（ＶＴ－ｘ機能）を利用するため、メモリ１０５の所定の領域にホストＶＭＭ保持データ１１を設定する。

【００４３】

ホストＶＭＭ保持データ１１は、ゲストＯＳ１１１ａ～１１１ｎの仮想化支援機能の利用の有無や、仮想ＣＰＵ１０８ａ～１０８ｎの状態を示すフラグ類を格納する領域と、仮想ＣＰＵ１０８ａ～１０８ｎ毎のステータス等を格納するシャドウＶＭＣＢ_０～_ｎを保持する物理ＣＰＵ制御データ１３の領域を含んで構成される。

【００４４】

ホストＶＭＭ保持データ１１のゲストＯＳや仮想ＣＰＵの状態を示すフラグとしては、例えば、ゲストＯＳ１１１ａ～１１１ｎ毎に物理ＣＰＵ１０４の仮想化支援機能を利用可能か否かを識別する仮想化機能有効フラグ１４１と、仮想ＣＰＵ１０８ａ～１０８ｎ毎に仮想化支援機能を利用中か否かを設定するＶＭＸＯＮフラグ１４２と、仮想ＣＰＵ１０８ａ～１０８ｎ仮想化支援機能がＶＭＸルートモードまたはＶＭＸノンルートモードの何れで動作しているのかを示す動作モードフラグ１４３が挙げられる。

【００４５】

仮想化機能有効フラグ１４１は、ゲストＯＳ１１１ａ～１１１ｎ毎に設定されて「１」であれば該当するゲストＯＳが仮想化支援機能を利用可能であることを示し、「０」であればゲストＯＳが仮想化支援機能を利用しないことを示す。この仮想化機能有効フラグ１４１は、ゲストＯＳ１１１ａ～１１１ｎ毎に管理コンソール３００から設定したり、あるいは、予め設定したファイルなどから設定する。

【００４６】

ＶＭＸＯＮフラグ１４２は、各仮想ＣＰＵ１０８ａ～１０８ｎ毎にＶＭＸモードの動作状態を示すフラグで、「１」のときには該当する仮想ＣＰＵ１０８ａ～１０８ｎの動作モードがＶＭＸモードであることを示し、「０」のときには該当する仮想ＣＰＵ１０８ａ～１０８ｎの動作モードが仮想化支援機能を利用しない通常動作モードであることを示す。

ＶＭＸＯＮフラグ１４２は、ゲストＯＳ１１１ａ～１１１ｎがＶＭＸＯＮ命令を発行したときにホストＶＭＭ１０が「１」をセットし、ゲストＯＳ１１１ａ～１１１ｎがＶＭＸＯＦＦ命令を発行したときにホストＶＭＭ１０が「０」にリセットする。

【００４７】

動作モードフラグ１４３は、仮想ＣＰＵ１０８ａ～１０８ｎ上で動作するプログラムの動作モードを追跡するフラグである。この動作モードフラグ１４３は、ゲストＯＳ１１１ａ～１１１ｎのＶＭ－ｅｎｔｒｙ時にホストＶＭＭ１０が「１」にセットし、ゲストＯＳのＶＭ－ｅｘｉｔ時にホストＶＭＭ１０が「０」にリセットする。つまり、この動作モードフラグ１４３は、ＶＭＸＯＮフラグ１４２が「１」のときに各仮想ＣＰＵ１０８ａ～１０８ｎ毎のＶＭＸモードの種別を示すもので、動作モードフラグ１４３が「０」のときには仮想ＣＰＵ１０８ａ～１０８ｎがゲストＶＭＭ２０を実行する状態（仮想ＣＰＵのＶＭＸルートモード）を示し、動作モードフラグ１４３が「１」のときには仮想ＣＰＵがユーザプログラム１１０ａ（ゲストＯＳ１１１ａまたはアプリケーション１１２ａ）を実行する状態（仮想ＣＰＵのＶＭＸノンルートモード）を示す。

【００４８】

ここで、ＶＭＸモードのＶＭＸルートモードとＶＭＸノンルートモードの遷移については、上記非特許文献１に記載されるとおりである。ここでは、概要のみについて説明する。通常動作モードからＶＭＸモードへ移行する際には、ホストＶＭＭ１０がＶＭＸＯＮ命令を発行し、物理ＣＰＵ１０４の動作モードをＶＭＸモードに移行させる。そして、ＶＭＸモードへ移行したホストＶＭＭ１０は、該当する仮想ＣＰＵ１０８ａ～１０８ｎのシャドウＶＭＣＢ_０～_ｎ－１にユーザプログラム１１０ａを実行させるための情報を書き込んでからＶＭ－ｅｎｔｒｙ命令（ＶＭＬＡＵＮＣＨ命令またはＶＭＲＥＳＵＭＥ命令）を発行し、ＶＭＸルートモードからＶＭＸノンルートモードへ移行する。このＶＭＸルートモードからＶＭＸノンルートモードへの遷移をＶＭ－ｅｎｔｒｙという。

【００４９】

逆に、ＶＭＸノンルートモードからＶＭＸルートモードへの遷移は、ＶＭ－ｅｘｉｔという。このＶＭ－ｅｘｉｔは、ゲストＯＳ１１１ａ～１１１ｎが特権命令を発行したときなどの所定の要因で物理ＣＰＵ１０４がホストＶＭＭ１０へＶＭ－ｅｘｉｔを通知する。ホストＶＭＭ１０のＣＰＵ制御部１２がＶＭ－ｅｘｉｔを検知すると、所定のエミュレーションを行ってゲストＶＭＭ２０またはゲストＯＳの処理を完了した後、シャドウＶＭＣＢを必要に応じて書き換えてからＶＭ－ｅｎｔｒｙ命令（第１制御命令）を発行してＶＭＸノンルートモードからＶＭＸルートモードへ切り替える。

【００５０】

本発明では、次世代ＯＳであるゲストＯＳ１１１ａのＶＭ－ｅｎｔｒｙの際に、ホストＶＭＭ１０がゲストＶＭＣＢ２２のゲスト状態エリア２２１とホスト状態エリア２２２を読み込んで、ゲストＯＳ１１１ａの動作に応じた一方のエリアの内容をシャドウＶＭＣＢ_０のゲスト状態エリア１３１へ設定することで、仮想サーバ１０２ａでゲストＯＳ１１１ａの仮想化機能を実現するのである。

【００５１】

ＶＴ－ｘ機能では以上のような、ＶＭ－ｅｎｔｒｙとＶＭ－ｅｘｉｔの遷移により、ホストＶＭＭ１０とゲストＶＭＭ２０またはユーザプログラム１１０ａを切り替える。このため、ＶＭ－ｅｎｔｒｙとＶＭ－ｅｘｉｔの前後の物理ＣＰＵ１０４のステータス等を保持するために、物理ＣＰＵ制御データ１３のデータ構造であるシャドウＶＭＣＢ_０～_ｎを使用する。

【００５２】

物理ＣＰＵ制御データ１３は、仮想ＣＰＵ１０８ａ～１０８ｎ毎にシャドウＶＭＣＢ_０～_ｎが設定され、各シャドウＶＭＣＢには次のようなデータが格納される。

【００５３】

ゲスト状態エリア１３１には、図３で示すように、仮想ＣＰＵ１０８ａ～１０８ｎのレジスタ状態などのステータスが格納される。すなわち、後述するように、ゲストＶＭＭ２

０のステータスまたはユーザプログラム１１０ａのステータスが選択的に格納される。
【００５４】
ホスト状態エリア１３２には、図３で示すように、ホストＶＭＭ１０の物理ＣＰＵ１０４のレジスタ状態などのステータスが格納される。ＶＭ実行制御領域１３３には、仮想サーバ１０２ａ～１０２ｎの設定情報が格納され、例えば、例外ビットマップやＩ／Ｏビットマップなどの情報が含まれる。ＶＭ－ｅｘｉｔ制御領域１３４には、ＶＭ－ｅｘｉｔの要因などの情報が格納される。ＶＭ－ｅｎｔｒｙ制御領域１３５には、ＶＭ－ｅｎｔｒｙの動作を制御するための情報が格納される。そして、ＶＭ－ｅｘｉｔ情報領域１３６には、ＶＭ－ｅｘｉｔを発生する要因（命令またはイベント）を格納する。ＶＭ－ｅｘｉｔを発生させる要因としては、例えば、図４に示す「description」に示すものがＶＭ－ｅｘｉｔ情報領域１３６に設定される。
【００５５】
以上のようなシャドウＶＭＣＢ_０～_ｎにより、ホストＶＭＭ１０は各仮想サーバ１０２ａ～１０２ｎの制御を行う。
【００５６】
一方、仮想サーバ１０２ａのゲストＶＭＭ２０が管理する仮想ＣＰＵ制御データ２１には、上記物理ＣＰＵ制御データ１３のシャドウＶＭＣＢ_０と同様のデータ構造であるゲストＶＭＣＢ２２が格納される。
【００５７】
ゲストＶＭＭ２０は、ユーザプログラム１１０ａ（ゲストＯＳ１１１ａまたはアプリケーション１１２ａ）を実行する仮想ＣＰＵ１０８ａのレジスタ状態などのステータス等を格納するゲスト状態エリア２２１と、ゲストＶＭＭ２０を実行する仮想ＣＰＵ１０８ａのレジスタ状態などのステータスを格納するホスト状態エリア２２２と、仮想サーバ１０２ａの設定情報を格納するＶＭ実行制御領域２２３と、仮想サーバ１０２ａにおけるＶＭ－ｅｘｉｔの要因などの情報を格納するＶＭ－ｅｘｉｔ制御領域２２４と、仮想サーバ１０２ａにおけるＶＭ－ｅｎｔｒｙの動作を制御するための情報を格納するＶＭ－ｅｎｔｒｙ制御領域２２５と、仮想サーバ１０２ａにおけるＶＭ－ｅｘｉｔの原因を特定する情報を格納するＶＭ－ｅｘｉｔ情報領域２２６とを備える。
【００５８】
ゲスト状態エリア２２１には、上記図３と同様に、仮想ＣＰＵ１０８ａのレジスタ状態などのステータスが格納される。すなわち、後述するように、ゲストＯＳ１１１ａのステータスまたはアプリケーション１１２ａのステータスが選択的に格納される。
【００５９】
仮想サーバ１０２ａの状態を制御するホストＶＭＭ１０のシャドウＶＭＣＢ_０では、ゲスト状態エリア１３１にゲストＶＭＭ２０のステータスまたはユーザプログラム１１０ａ（ゲストＯＳ１１１ａまたはアプリケーション１１２ａ）のステータスが格納され、ホスト状態エリア１３２にはホストＶＭＭ１０のステータスが格納される。
【００６０】
一方、ゲストＶＭＭ２０のゲストＶＭＣＢ２２では、ゲスト状態エリア２２１にゲストＯＳ１１１ａのステータスまたはアプリケーション１１２ａのステータスが格納され、ホスト状態エリア２２２にはゲストＶＭＭ２０のステータスが格納される点でシャドウＶＭＣＢ_０と相違する。
【００６１】
次に、ホストＶＭＭ１０のＣＰＵ制御部１２の構成について説明する。ＣＰＵ制御部１２は、管理コンソール３００からの入力などに基づいて各仮想サーバ１０２ａ～１０２ｎへ物理サーバ１０１の計算機資源を割り当てるリソース管理部（図示省略）に加えて、ゲストＯＳ１１１ａの仮想化機能（ゲストＶＭＭ２０）を稼動させるため、仮想ＣＰＵのステータスの読み込み先を選択する状態エリア選択部１２１と、エミュレータ１２２と、シャドウＶＭＣＢ参照・更新部１２３と、ＶＭ－ｅｘｉｔハンドラ１２４と、ＶＭ－ｅｎｔｒｙ命令発行部１２５と、ユーザインターフェース３０１とを備える。なお、リソース管

理部については、上述のように公知または周知の技術を適用すればよいので、本実施形態では詳述しない。

【００６２】

ＶＭ－ｅｘｉｔハンドラ１２４は、物理ＣＰＵ１０４からＶＭ－ｅｘｉｔを受信して、エミュレータ１２２を起動する。

【００６３】

エミュレータ１２２は、ＶＭ－ｅｘｉｔハンドラ１２４から受信したＶＭ－ｅｘｉｔの発生要因となった命令またはイベントを特定し、後述するように特定した発生要因に対応するモジュールを起動してゲストＶＭＭ２０やユーザプログラム１１０ａに代わって処理を実行する。

【００６４】

エミュレータ１２２が、仮想ＣＰＵ１０８ａの動作モードの変更（ＶＭＸルートとＶＭＸノンルートの切り替え）を検出した場合は、特定した要因に応じたモジュール（ＶＭ－ｅｎｔｒｙ命令実行モジュール、ＶＭＣＬＥＡＲ命令実行モジュール、ＣＰＵＩＤ命令実行モジュール、ＶＭ－ｅｘｉｔ条件検出モジュール）を起動して、状態エリア選択部１２１を起動する。

【００６５】

ここで、ＶＭ－ｅｘｉｔの発生要因は、図４に示すように、シャドウＶＭＣＢ_０のＶＭ－ｅｘｉｔ情報領域１３６の「Ｅｘｉｔ　ｒｅａｓｏｎ」に設定されたものである。図４に示すＶＭ－ｅｘｉｔの要因リストには、ＶＭ－ｅｎｔｒｙ命令の発行に起因する要因１３６１と、ＶＭＣＬＥＡＲ命令（第４制御命令）の発行に起因する要因１３６２と、ゲストＶＭＭ２０へのＶＭ－ｅｘｉｔ通知の有無を設定した通知条件１３６３が予め設定される。

【００６６】

例えば、仮想ＣＰＵ１０８ａの動作モードを切り替える際のＶＭ－ｅｘｉｔの要因として、ＶＭ－ｅｎｔｒｙ命令（ＶＭＬＡＵＮＣＨ命令またはＶＭＲＥＳＵＭＥ命令）を検出した場合には、図４の要因１３６１に該当するので、ホストＶＭＭ１０はＶＭ－ｅｎｔｒｙ命令実行モジュールでエミュレーションを行い、ゲストＶＭＭ２０またユーザプログラム１１０ａに代わって処理を行う。

【００６７】

あるいは、ＶＭ－ｅｘｉｔの要因が、図４の通知条件１３６３に該当するゲストＶＭＭへの通知条件の場合には、ＶＭ－ｅｘｉｔ条件検出モジュールを起動して同様にエミュレーションを行う。

【００６８】

状態エリア選択部１２１は、ホストＶＭＭ保持データ１１からＶＭ－ｅｘｉｔを発生した仮想ＣＰＵ（この例では１０８ａ）の動作モードフラグ１４３を読み込んで、ＶＭＸルートモードとＶＭＸノンルートモードの何れであるかを判定する。動作モードフラグ１４３が「０」のＶＭＸルートモードであればゲストＶＭＭ２０が動作していたので、状態エリア選択部１２１は、ＶＭ－ｅｘｉｔを発生した仮想サーバ（この例では１０２ａ）のゲストＶＭＣＢ２２からホスト状態エリア２２２を読み込む。

【００６９】

一方、動作モードフラグ１４３が「１」のＶＭＸノンルートモードであればユーザプログラム１１０ａが動作していたので、状態エリア選択部１２１は、該当する仮想サーバ１０２ａのゲストＶＭＣＢ２２からゲスト状態エリア２２１を読み込む。

【００７０】

ゲストＶＭＣＢ２２からの読み込みが完了すると、ＣＰＵ制御部１２ではシャドウＶＭＣＢ参照・更新モジュール１２３を起動する。シャドウＶＭＣＢ参照・更新モジュール１２３は、状態エリア選択部１２１が読み込んだゲストＶＭＣＢ２２の情報を、ＶＭ－ｅｘｉｔの処理対象である仮想ＣＰＵ１０８ａに対応するシャドウＶＭＣＢ_０のゲスト状態エリア１３１に書き込んでシャドウＶＭＣＢを更新する。

【００７１】
シャドウＶＭＣＢ_０のゲスト状態エリア１３１の更新が完了すると、ＣＰＵ制御部１２は仮想ＣＰＵ１０８ａの動作モードをＶＭＸルートモードからＶＭＸノンルートモードへ切り替えるため、動作モードフラグ１４３を「１」に更新し、物理ＣＰＵ１０４のポインタ１１５に操作対象の仮想ＣＰＵ１０８ａのシャドウＶＭＣＢ_０のアドレスをセットする。
【００７２】
そして、ＣＰＵ制御部１２は、ＶＭ－ｅｎｔｒｙ命令発行モジュール１２５を起動して、物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令（ＶＭＲＥＳＵＭＥ命令）を発行する。
【００７３】
物理ＣＰＵ１０４はＶＭ－ｅｎｔｒｙ命令を受け付けると、ポインタ１１５が指し示すシャドウＶＭＣＢ_０のゲスト状態エリア１３１を読み込んで、状態エリア選択部１２１が選択した仮想サーバ１０２ａのゲストＶＭＭ２０またはユーザプログラム１１０ａを実行する。
【００７４】
以上のように、仮想サーバ１０２ａで仮想化機能を統合した次世代ＯＳのゲストＯＳ１１１ａが稼動している場合、ホストＶＭＭ１０のＣＰＵ制御部１２は、物理ＣＰＵ１０４のＶＭ－ｅｘｉｔを検出すると、シャドウＶＭＣＢ_０の動作モードフラグ１４３を参照して、仮想ＣＰＵ１０８ａで実行中のプログラムがゲストＶＭＭ２０とユーザプログラム１１０ａの何れであるかを判定する。そして、ＣＰＵ制御部１２は、仮想ＣＰＵ１０８ａで実行していたプログラムに応じてゲスト状態エリア２２１またはホスト状態エリア２２２の内容をホストＶＭＭ１０のシャドウＶＭＣＢ_０のゲスト状態エリア１３１へ書き込んでからＶＭ－ｅｎｔｒｙ命令を実行する。
【００７５】
こうして、仮想サーバ１０２ａのゲストＶＭＭ２０が保持するゲストＶＭＣＢ２２の情報で、ホストＶＭＭ１０が保持するシャドウＶＭＣＢ_０のゲスト状態エリア１３１を更新することにより、ゲストＯＳ１１１ａの仮想化機能を稼働させる仮想サーバ１０２ａと、仮想化機能を用いない既存のＯＳを実行する仮想サーバ１０２ｎをひとつの物理サーバ１０１に統合することが可能となる。
【００７６】
＜ホストＶＭＭの処理の詳細＞
次に、ホストＶＭＭ１０で行われる処理について、図５を参照しながら以下に説明する。図５は、仮想サーバ１０２ａ～１０２ｎの稼働中に物理ＣＰＵ１０４からのＶＭ－ｅｘｉｔを受信したときにホストＶＭＭ１０のＣＰＵ制御部１２が行う処理の一例を示すフローチャートである。なお、この例では、ＶＭ－ｅｘｉｔの要因がＶＭ－ｅｎｔｒｙ命令である場合を示す。
【００７７】
まず、Ｓ１では、ホストＶＭＭ１０のＣＰＵ制御部１２は、ＶＭ－ｅｘｉｔの要因となった操作対象のゲストＯＳ１１１ａ～１１１ｎ（以下、ゲストＯＳとする）に対応するホストＶＭＭ保持データ１１の仮想化機能有効フラグ１４１を参照し、ＶＭ－ｅｘｉｔを発生したゲストＯＳがＶＴ－ｘ機能を利用可能であるか否かを判定する。操作対象のゲストＯＳがＶＴ－ｘ機能を利用可能である場合にはＳ２へ進む。一方、操作対象のゲストＯＳがＶＴ－ｘ機能を利用しない場合（ＮＴサーバまたは２０００サーバなど）には、Ｓ１５へ進んで、ホストＶＭＭ１０は前記従来例に示した特許文献１または非特許文献１に記載されるような仮想マシン処理を行う。
【００７８】
Ｓ２では、ホストＶＭＭ１０がＶＭ－ｅｘｉｔの要因となった仮想ＣＰＵ１０８ａ～１０８ｎ（以下、仮想ＣＰＵとする）のＶＭＸＯＮフラグ１４２を参照し、この操作対象の仮想ＣＰＵがＶＭＸモードであるか否かを判定する。ＶＭＸＯＮフラグ１４２が「１」で

あれば仮想ＣＰＵはＶＭＸモードであるのでＳ３へ進む。一方、ＶＭＸＯＮフラグ１４２が「０」の場合には、仮想ＣＰＵが通常の動作モードであるので、上記と同様にＳ１５へ進み、ホストＶＭＭ１０は既存の仮想マシン処理を実行する。

【００７９】

Ｓ３では、ホストＶＭＭ１０のＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードフラグ１４３を参照し、仮想ＣＰＵの動作モードがＶＭＸルートモードとＶＭＸノンルートモードの何れであるかを判定する。動作モードフラグ１４３が「０」の場合には仮想ＣＰＵがＶＭＸルートモードであると判定してＳ４へ進む。一方、動作モードフラグ１４３が「１」の場合には、仮想ＣＰＵがＶＭＸノンルートモードであると判定してＳ１１へ進む。

【００８０】

Ｓ４では、ＣＰＵ制御部１２が物理ＣＰＵ１０４から受信したＶＭ－ｅｘｉｔの発生要因を特定する。この例では、仮想ＣＰＵの動作モードがＶＭＸルートモードであるので、ＣＰＵ制御部１２は、ゲストＶＭＭ２０の状態（ステータス）を格納したゲスト状態エリア１３１を参照し、ゲストＶＭＭ２０が発行したＶＭ－ｅｎｔｒｙ命令をＶＭ－ｅｘｉｔの要因として特定する。

【００８１】

次に、Ｓ５ではＣＰＵ制御部１２は、エミュレータ１２２からＶＭ－ｅｎｔｒｙ命令実行モジュールを実行し、操作対象の仮想ＣＰＵをＶＭＸノンルートモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）をゲストＶＭＭ２０に代わってエミュレートする。

【００８２】

次に、Ｓ６ではホストＶＭＭ１０のＣＰＵ制御部１２は、仮想ＣＰＵの動作モードをＶＭＸルートモードからＶＭＸノンルートモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「１」に更新する。そして、Ｓ７では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０の仮想ＣＰＵ制御データ２１からゲスト状態エリア２２１に格納されているゲストＯＳまたはアプリケーションのステータスを読み込む。

【００８３】

次に、Ｓ８において、ＣＰＵ制御部１２は、物理ＣＰＵ１０４に対してＶＭＰＴＬＤＲ命令を発行して、操作対象の仮想ＣＰＵに対応するシャドウＶＭＣＢをアクティブに設定し、アクティブにしたシャドウＶＭＣＢのアドレスをポインタ１１５に設定する。このＶＭＰＴＬＤＲ命令（第２制御命令）により、ホストＶＭＭ１０は、複数のシャドウＶＭＣＢ_０～_ｎ－１の中から操作対象の仮想ＣＰＵ（仮想サーバ）のシャドウＶＭＣＢを選択する。

【００８４】

Ｓ９では、ＣＰＵ制御部１２が、上記Ｓ７で読み込んだゲスト状態エリア２２１の情報で操作対象のシャドウＶＭＣＢ_０のゲスト状態エリア１３１を更新する。そして、Ｓ１０でＣＰＵ制御部１２は、物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行する。

【００８５】

ＶＭ－ｅｎｔｒｙ命令を受け付けた物理ＣＰＵ１０４は、ポインタ１１５で指定されたシャドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのユーザプログラム（ゲストＯＳまたはアプリケーション）を実行する。なお、物理ＣＰＵ１０４は、ＶＭ－ｅｎｔｒｙ命令を受け付けると、実行していたホストＶＭＭ１０のステータスをシャドウＶＭＣＢのホスト状態エリア１３２へ格納し、次回の呼び出しに備える。

【００８６】

一方、上記Ｓ３で動作モードフラグ１４３が「１」と判定された場合は、操作対象の仮想ＣＰＵがユーザプログラムを実行中のＶＭＸノンルートモードであるので、Ｓ１１の処理を行う。

【００８７】

Ｓ１１では、ＣＰＵ制御部１２が図４のＶＭ－ｅｘｉｔの要因リストを参照し、通知条件１３６３からゲストＶＭＭへのＶＭ－ｅｘｉｔ通知条件を検索する。この例では、ＶＭ－ｅｘｉｔの要因となったＶＭ－ｅｎｔｒｙ命令（ＶＭＬＡＵＮＣＨ、ＶＭＲＥＳＵＭＥ命令）であるので、ＶＭ－ｅｘｉｔ通知条件に一致する。

【００８８】

そして、ＣＰＵ制御部１２は、Ｓ１２で図２のＶＭ－ｅｘｉｔ条件検出モジュールを実行し、操作対象の仮想ＣＰＵをＶＭＸルートモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）をユーザプログラムに代わってエミュレートする。

【００８９】

次に、Ｓ１３では、操作対象の仮想ＣＰＵの動作モードをＶＭＸノンルートモードからＶＭＸルートモードへ移行させるため、ＣＰＵ制御部１２が操作対象の仮想ＣＰＵの動作モードフラグ１４３を「０」にリセットする。そして、Ｓ１４では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０の仮想ＣＰＵ制御データ２１からホスト状態エリア２２１に格納されているゲストＶＭＭ２０のステータスを読み込む。

【００９０】

Ｓ１４の処理が完了すると、上記Ｓ８～Ｓ１０を実行し、ＣＰＵ制御部１２は、上記Ｓ１４で読み込んだホスト状態エリア２２２の情報で操作対象のシャドウＶＭＣＢのゲスト状態エリア１３１を更新し、ゲストＶＭＭ２０のステータスをセットしてから物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行する。

【００９１】

この結果、ＶＭ－ｅｎｔｒｙ命令を受け付けた物理ＣＰＵ１０４は、ポインタ１１５で指定されたシャドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのゲストＶＭＭ２０を実行する。

【００９２】

このように、ゲストＯＳが仮想化機能を統合した次世代ＯＳである場合、ホストＶＭＭ１０は、仮想ＣＰＵの動作モードと、発生したＶＭ－ｅｘｉｔの要因に応じてシャドウＶＭＣＢのゲスト状態エリア１３１に書き込むステータスを、ゲストＶＭＭまたはユーザプログラムの何れか一方から選択する。そして、ホストＶＭＭ１０が物理ＣＰＵ１０４（第１の仮想ＣＰＵ）に対してＶＭ－ｅｎｔｒｙ命令を発行することで、仮想サーバ上でゲストＶＭＭと、このゲストＶＭＭが提供する第２の仮想ＣＰＵ上で稼動するユーザプログラムを切り替えて実行することが可能となり、ゲストＶＭＭは仮想サーバ上に複数の仮想化環境（ユーザプログラム）を提供することができる。

【００９３】

なお、上記Ｓ１の判定でゲストＯＳが仮想化機能を利用しない場合、または上記Ｓ２の判定で、仮想ＣＰＵが物理ＣＰＵ１０４のＶＴ－ｘ機能を利用しない場合は、Ｓ１５において、前記従来例に示した特許文献１または非特許文献１の記載に準じた仮想マシン処理をホストＶＭＭ１０で行えばよい。

【００９４】

Ｓ１５の仮想マシン処理は、例えば、図１に示した仮想サーバ１０２ｎのゲストＯＳ１１１ｎは、既存のＯＳであり、このゲストＯＳ１１１ｎまたはアプリケーション１１２ｎ（ユーザプログラム１１０ｎ）が特権命令などの所定の命令を実行すると、上述したように物理ＣＰＵ１０４はＶＭ－ｅｘｉｔの発生をホストＶＭＭ１０に通知する。

【００９５】

ホストＶＭＭ１０は、物理ＣＰＵ１０４からのＶＭ－ｅｘｉｔの通知を受けると、ユーザプログラム１１０ｎ（仮想ＣＰＵ１０８ｎ）のステータスをシャドウＶＭＣＢ_ｎ－１のゲスト状態エリア１３１に格納する。そして、ホストＶＭＭ１０は、ポインタ１１５のアドレスをホストＶＭＭ１０のステータスを格納したホスト状態エリアに設定して、所定の処理を実行する。

【００９６】

ホストＶＭＭ１０は、特権命令など所定の処理が完了すると、ホストＶＭＭ１０のステ

ータスをホスト状態エリア１３２へ格納し、ポインタ１１５のアドレスにゲスト状態エリア１３１を設定してから、ＶＭ－ｅｎｔｒｙ命令（ＶＭＲＥＳＵＭＥ命令）を発行して、制御を仮想ＣＰＵ１０８ｎに移動してユーザプログラム１１０ｎの実行を再開する。

【００９７】

このように、本発明によれば、仮想化機能を統合した次世代ＯＳと、既存のＯＳをひとつの物理サーバ１０１へ統合することが可能となって、物理サーバの数を削減してサーバの運用管理コストを削減することが可能となる。

【００９８】

さらに、上述のように、ホストＶＭＭ１０は次世代ＯＳに対して仮想ＣＰＵがＶＴ－ｘ機能を提供するように見せかけることができ、仮想化ソフトウェアを統合したＯＳを確実に稼働させることが可能となる。また、本発明のホストＶＭＭ１０によれば、前記従来例のシミュレータのように、命令列の変換などのオーバーヘッドがなく、仮想計算機の性能を低下させることなく、仮想化機能を統合したＯＳを仮想サーバで実行することが可能となるのである。

【００９９】

また、本発明では、複数の仮想ＣＰＵ１０８ａ～１０８ｎに対応して、複数のシャドウＶＭＣＢ_０～_ｎ－１を設けたので、物理サーバ１０１で複数のゲストＶＭＭ２０を実行する場合でも、シャドウＶＭＣＢ_０～_ｎ－１を切り替えるだけで迅速に処理を切り替えることが可能となる。これにより、物理サーバ１０１に複数の次世代ＯＳを統合した場合でも、仮想サーバの性能を維持することが可能となる。

【０１００】

図６は、仮想サーバの稼働中にＶＭＣＬＥＡＲ命令の実行に起因して、物理ＣＰＵ１０４からのＶＭ－ｅｘｉｔを受信したときにホストＶＭＭ１０のＣＰＵ制御部１２が行う処理の一例を示すフローチャートである。

【０１０１】

Ｓ２１～Ｓ２３は、上記図５のＳ１～Ｓ３と同様であり、ホストＶＭＭ１０のＣＰＵ制御部１２が、ＶＭ－ｅｘｉｔの要因となった操作対象のゲストＯＳがＶＴ－ｘ機能を利用可能であるか否かを判定し、操作対象の仮想ＣＰＵのＶＭＸＯＮフラグ１４２を参照して、操作対象の仮想ＣＰＵがＶＴ－ｘ機能を使用中か否かを判定する。操作対象のゲストＯＳがＶＴ－ｘ機能を利用しない場合や、ＶＭＸＯＮフラグ１４２が「０」である仮想ＣＰＵが通常の動作モードの場合は、Ｓ３５へ進み、ホストＶＭＭ１０は既存の仮想マシン処理を実行する。

【０１０２】

さらに、ＣＰＵ制御部１２は、操作対象の仮想ＣＰＵの動作モードフラグ１４３を参照し、仮想ＣＰＵの動作モードがＶＭＸルートモードとＶＭＸノンルートモードの何れであるかを判定し、ＶＭＸルートモードであればＳ２４へ進み、ＶＭＸノンルートモードであればＳ３０へ進む。

【０１０３】

Ｓ２４では、ＣＰＵ制御部１２は、物理ＣＰＵ１０４から受信したＶＭ－ｅｘｉｔから要因を特定する。この例では、ＣＰＵ制御部１２がゲスト状態エリア１３１を参照して、ＶＭ－ｅｘｉｔの要因としてゲストＶＭＭ２０のＶＭＣＬＥＡＲ命令を特定する。

【０１０４】

次に、Ｓ２５ではＣＰＵ制御部１２のエミュレータ１２２からＶＭＣＬＥＡＲ命令実行モジュールを起動してエミュレーションを行う。Ｓ２６では、エミュレーションにより、物理ＣＰＵ１０４の動作状態（ステータス）を読み込んで、操作対象の仮想ＣＰＵに対応するシャドウＶＭＣＢを更新する。これにより、物理ＣＰＵ１０４のステータスをシャドウＶＭＣＢに反映させる。

【０１０５】

Ｓ２７では、ＣＰＵ制御部１２が操作対象の仮想ＣＰＵに対応するシャドウＶＭＣＢのゲスト状態エリア１３１に格納されているステータスを読み込む。

【０１０６】
次に、Ｓ２８において、ＣＰＵ制御部１２は、シャドウＶＭＣＢのゲスト状態エリア１３１から読み込んだステータスをゲストＶＭＭ２０のゲスト状態エリア２２１へ書き込んで更新する。これにより、ゲストＶＭＭ２０のゲスト状態エリア２２１は、物理ＣＰＵ１０４のステータスと同期する。
【０１０７】
Ｓ２９では、ＣＰＵ制御部１２が、ポインタ１１５のアドレスを操作対象の仮想ＣＰＵに対応するシャドウＶＭＣＢのゲスト状態エリア１３１に設定してから、物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行する。
【０１０８】
ＶＭ－ｅｎｔｒｙ命令を受け付けた物理ＣＰＵ１０４は、ポインタ１１５で指定されたシャドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのユーザプログラム（ゲストＯＳまたはアプリケーション）を実行する。
【０１０９】
一方、上記Ｓ２３で動作モードフラグ１４３が「１」と判定された場合は、操作対象の仮想ＣＰＵがＶＭＸノンルートモードであるので、Ｓ３０の処理を行う。Ｓ３０では、図４のＶＭ－ｅｘｉｔの要因リストを参照し、ＶＭ－ｅｘｉｔの要因を解析する。ＣＰＵ制御部１２は、通知条件１３６３からゲストＶＭＭへのＶＭ－ｅｘｉｔ通知条件を検索する。この例では、ＶＭ－ｅｘｉｔの要因となったＶＭＣＬＥＡＲ命令は、ＶＭ－ｅｘｉｔ通知条件に一致する。
【０１１０】
そして、Ｓ３１ではＣＰＵ制御部１２がゲストＶＭＭ２０に対して規定外命令エラーを通知する。すなわち、ユーザプログラムの権限ではＶＭＣＬＥＡＲ命令を実行できなかったことを、ゲストＶＭＭ２０へ通知する。
【０１１１】
次に、Ｓ３２ではＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードをＶＭＸノンルートモードからＶＭＸルートモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「０」にリセットする。
【０１１２】
Ｓ３３では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０の仮想ＣＰＵ制御データ２１から、ホスト状態エリア２２１に格納されているゲストＶＭＭ２０のステータスを読み込む。
【０１１３】
Ｓ３４では、ＣＰＵ制御部１２が、上記Ｓ３３で読み込んだホスト状態エリア２２２のステータスを、操作対象のシャドウＶＭＣＢのゲスト状態エリア１３１に書き込んでシャドウＶＭＣＢを更新する。その後、上記Ｓ２９では、ＣＰＵ制御部１２がポインタ１１５のアドレスをゲスト状態エリア１３１に設定してから、物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行する。
【０１１４】
この結果、ＶＭ－ｅｎｔｒｙ命令を受け付けた物理ＣＰＵ１０４は、ポインタ１１５で指定されたシャドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのゲストＶＭＭ２０を実行する。
【０１１５】
なお、上記Ｓ２１の判定でゲストＯＳが仮想化機能を利用しない場合、または上記Ｓ２２の判定で、仮想ＣＰＵが物理ＣＰＵ１０４のＶＴ－ｘ機能を利用しない場合は、Ｓ３５において、前記図５のＳ１５と同様にして、従来例に示した特許文献１または非特許文献１の記載に準じた仮想マシン処理をホストＶＭＭ１０で行えばよい。
【０１１６】
このように、ゲストＶＭＭ２０がＶＭＣＬＥＡＲ命令を発行した場合には、ホストＶＭＭ１０は、物理ＣＰＵ１０４の動作状態（ステータス）を取得してシャドウＶＭＣＢのゲ

スト状態エリア１３１を、取得した動作状態で更新する。そして、ホストＶＭＭ１０は、シャドウＶＭＣＢのゲスト状態エリア１３１のステータスを、ゲストＶＭＣＢのゲスト状態エリア２２１に反映させることで、物理ＣＰＵ１０４のステータスとホストＶＭＭ１０のゲスト状態エリア１３１及びゲストＶＭＭ２０上のゲスト状態エリア２２１のステータスを同期させることができる。

【０１１７】

これにより、仮想化機能を統合した次世代ＯＳは、仮想サーバ上で仮想環境を円滑に提供することが可能となる。

【０１１８】

図７は、仮想サーバの稼働中にＣＰＵＩＤ命令（第３制御命令）の実行に起因して、物理ＣＰＵ１０４からのＶＭ－ｅｘｉｔを受信したときにホストＶＭＭ１０のＣＰＵ制御部１２が行う処理の一例を示すフローチャートである。

【０１１９】

まず、Ｓ４１では、図５のＳ３と同様に、ホストＶＭＭ１０のＣＰＵ制御部１２は、ＶＭ－ｅｘｉｔの要因となった操作対象の仮想ＣＰＵの動作モードフラグ１４３を参照し、仮想ＣＰＵの動作モードがＶＭＸルートモードとＶＭＸノンルートモードの何れであるかを判定する。動作モードフラグ１４３が「０」の場合には仮想ＣＰＵがＶＭＸルートモードであると判定してＳ４２へ進む。一方、動作モードフラグ１４３が「１」の場合には、仮想ＣＰＵがＶＭＸノンルートモードであると判定してＳ４１へ進む。

【０１２０】

Ｓ４２では、図５のＳ４と同様にして、ＣＰＵ制御部１２が物理ＣＰＵ１０４から受信したＶＭ－ｅｘｉｔの発生要因をゲスト状態エリア１３１の値から特定する。この例では、ＶＭＸルートモードであるので、ＣＰＵ制御部１２は、ゲストＶＭＭ２０のＣＰＵＩＤ命令をＶＭ－ｅｘｉｔの発生要因として特定する。

【０１２１】

Ｓ４３では、図５のＳ１と同様に、ＣＰＵ制御部１２は、ＶＭ－ｅｘｉｔの要因となった操作対象のゲストＯＳに対応するホストＶＭＭ保持データ１１の仮想化機能有効フラグ１４１を参照し、ＶＭ－ｅｘｉｔを発生したゲストＯＳがＶＴ－ｘ機能を利用可能であるか否かを判定する。操作対象のゲストＯＳがＶＴ－ｘ機能を利用可能である場合にはＳ４４へ進む。一方、操作対象のゲストＯＳがＶＴ－ｘ機能を利用しない場合には、Ｓ４５へ進む。

【０１２２】

Ｓ４４では、ＣＰＵＩＤ命令の戻り値を示すリターンレジスタ（ＩＡ－３２の場合ＥＣＸ）の所定のビット（ＣＰＵＩＤ．１．ＥＣＸ［５］＝１）を１に設定して、ＶＴ－ｘ機能が有効であることを設定する。

【０１２３】

一方、Ｓ４５では、ＣＰＵＩＤ命令の戻り値を示すリターンレジスタ（ＩＡ－３２の場合ＥＣＸ）の所定のビットを０に設定して、ＶＴ－ｘ機能が無効であることを設定する。

【０１２４】

次に、Ｓ４６においてＣＰＵ制御部１２は、エミュレータ１２２からＣＰＵＩＤ命令実行モジュールを起動して、ゲストＶＭＭ２０もしくはユーザプログラムへ通知すべき例外のチェック等を行い、チェック結果をゲスト状態エリア１３１に格納する。

【０１２５】

Ｓ４７では、ＣＰＵ制御部１２が物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行して、仮想サーバに制御を移す。

【０１２６】

一方、上記Ｓ４１の判定で、動作モードフラグ１４３が「１」となるユーザプログラムを実行していた場合（ＶＭＸノンルートモード）にはＳ４８へ進み、ユーザプログラムを実行している期間中のＶＭ－ｅｘｉｔの要因を解析する。Ｓ４８では、ＣＰＵ制御部１２が物理ＣＰＵ１０４から受信したＶＭ－ｅｘｉｔの発生要因をゲスト状態エリア１３１の

値から特定する。この例では、ＶＭＸノンルートモードであるので、ＣＰＵ制御部１２は、ゲストＯＳまたはアプリケーションのＣＰＵＩＤ命令をＶＭ－ｅｘｉｔの発生要因として特定する。

【０１２７】

Ｓ４９では、ＣＰＵ制御部１２が、ユーザプログラム実行中のＣＰＵＩＤ命令に起因するＶＭ－ｅｘｉｔについて、ゲストＶＭＭ２０への通知が必要であるか否かを判定する。この判定は、ＣＰＵ制御部１２が、仮想ＣＰＵ制御データ２１のゲストＶＭＣＢ２２のＶＭ実行制御領域２２３に予め設定された情報を参照して、判定を行う。ＣＰＵＩＤ命令に伴うＶＭ－ｅｘｉｔをゲストＶＭＭ２０へ通知する必要があればＳ５０へ進む。一方、ゲストＶＭＭ２０へのＶＭ－ｅｘｉｔの通知が不要であればＳ４３へ進み、上述のＳ４３移行の処理を実行する。

【０１２８】

Ｓ５０では、ゲストＶＭＭ２０へＶＭ－ｅｘｉｔの通知を行うため、ＣＰＵ制御部１２はＶＭ－ｅｘｉｔエミュレータを実行し、Ｓ５１で操作対象の仮想ＣＰＵの動作モードフラグ１４３を「０」にリセットしてＶＭＸルートモードへ切り替える。

【０１２９】

次に、Ｓ５２でＣＰＵ制御部１２は、操作対象のゲストＶＭＣＢ２２のホスト状態エリア２２２のステータスを読み込む。Ｓ５３ではＣＰＵ制御部１２が、上記読み込んだステータスを、操作対象の仮想ＣＰＵのシャドウＶＭＣＢのゲスト状態エリア１３１に書き込んで更新を行う。その後、Ｓ４７で、ＣＰＵ制御部１２が物理ＣＰＵ１０４に対してＶＭ－ｅｎｔｒｙ命令を発行して制御をゲストＶＭＭ２０へ移す。

【０１３０】

以上のように、ゲストＶＭＭ２０がＣＰＵＩＤ命令を発行した場合と、ＣＰＵＩＤ命令に起因するＶＭ－ｅｘｉｔをゲストＶＭＭ２０へ通知しない場合では、仮想化機能有効フラグ１４１の設定に応じてリターンレジスタの値を設定した後、ＣＰＵ制御部１２は、ＶＭ－ｅｎｔｒｙ命令を発行して、ゲストＶＭＭ２０またはユーザプログラムへ制御を戻すことができる。また、ユーザプログラムがＣＰＵＩＤ命令を発行し、ゲストＶＭＭ２０へＶＭ－ｅｘｉｔの通知が必要な場合では、シャドウＶＭＣＢのゲスト状態エリア１３１を、ゲストＶＭＣＢ２２のホスト状態エリア２２２の内容で更新して、制御をゲストＶＭＭ２０へ移すことができる。

【０１３１】

以上のように、本第１実施形態によれば、ホストＶＭＭ１０がゲストＯＳやアプリケーションの状態と仮想ＣＰＵの状態を監視してシャドウＶＭＣＢのゲスト状態エリア１３１を書き換えることで、仮想サーバの性能を低下させることなく、仮想化ソフトウェアを統合したＯＳを確実に稼働させることが可能となる。そして、ひとつの物理サーバ１０１で仮想化機能を統合した次世代ＯＳと、既存のＯＳを共存させることが可能となって、サーバ統合を効率よく行うことで、サーバの運用コストを削減することが可能となる。

【０１３２】

なお、ホストＶＭＭ１０が管理コンソール３００に提供するユーザーインターフェースとしては、例えば、図８に示すようなＧＵＩで構成される。図８は、各仮想サーバ毎に仮想化機能を有効にするか無効にするかを設定するユーザーインターフェースを示す。図中「ＶＭ_」は、仮想サーバの識別子を示し、「現在設定」は「ＯＮ」であれば仮想化機能が有効出ることを示し、「ＯＦＦ」であれば仮想化機能が無効であることを示す。そして、「新規設定」に「ＯＮ」または「ＯＦＦ」を設定してから、画面左下の「ＯＫ」をクリックすることにより、所望の仮想サーバについて仮想化機能の有効または無効を切り替えることができる。すなわち、ホストＶＭＭ１０は、ユーザーインターフェース３０１から入力された「新規設定」の値に基づいて、ホストＶＭＭ保持データ１１のうち、該当する仮想サーバの仮想化機能有効フラグ１４１に「１」または「０」を設定する。これにより、ユーザーインターフェース３０１から入力された設定値を任意の仮想サーバへ反映させることができる。

【０１３３】
＜第２実施形態＞
図９は、第２の実施形態を示し、前記第１実施形態に示した物理ＣＰＵ１０４のＶＴ－ｘ機能を、前記従来例（非特許文献２）に示したＡＭＤ－Ｖ（Virtualization）に置き換えたものである。
【０１３４】
図９において、ＡＭＤ－Ｖを実装した物理ＣＰＵ１０４ａは、前記第１実施形態に示したホストＶＭＭ１０のシャドウＶＭＣＢのアドレスを指し示すポインタ１１５に加えて、ホスト状態エリア１３２のアドレスを指し示すポインタ１１８を備える。
【０１３５】
ＡＭＤ－Ｖを備える物理ＣＰＵ１０４ａは、ＶＭ－ｅｎｔｒｙ命令に代わって、ＶＭＲＵＮ命令を実行するとホストＶＭＭ１０を実行するホストモードからゲストＶＭＭ２０またはユーザプログラムを実行するゲストモードへの切り換え行う。そして、ホストＶＭＭ１０は、ＡＭＤ－Ｖの仕様に応じて、次の点が前記第１実施形態のＶＴ－ｘ機能と相違する。
【０１３６】
ホストＶＭＭ保持データ１１のうち、前記第１実施形態のＶＭＸＯＮフラグ１４２を削除し、仮想サーバ毎の仮想化機能有効フラグ１４１と、仮想ＣＰＵ１０８ａ～ｎ毎の動作モードフラグ１４３を備える。そして、ホストＶＭＭ保持データ１１の物理ＣＰＵ制御データは、図１０で示すように、仮想ＣＰＵ毎のホスト状態エリア１３２と、仮想ＣＰＵ毎のシャドウＶＭＣＢ_０～_ｎ－１に、ゲスト状態エリア１３１と制御エリア１３７を備える。なお、シャドウＶＭＣＢ_０～_ｎ－１の制御エリア１３７は、前記第１実施形態のＶＭ－ｅｘｉｔ情報領域１３６に相当する情報を格納する。
【０１３７】
すなわち、制御エリア１３７には、図１１で示すように、予め設定された_ＶＭＥＸＩＴ（前記第１実施形態のＶＭ－ｅｘｉｔに相当）の発生要因のコードを示すＥＸＩＴＣＯＤＥ１３７１と、コードの名称１３７２と、このコードがＶＭＲＵＮ命令に起因することを示す判定結果１３７３と、_ＶＭＥＸＩＴをゲストＶＭＭ２０へ通知することを示す通知条件１３７４が含まれている。
【０１３８】
ＣＰＵ制御部１２のエミュレータ１２２は、ＶＭ－ｅｎｔｒｙ命令に代わって、ＶＭＲＵＮ命令実行モジュールを備え、ＶＭＣＬＥＡＲ命令実行モジュールを削除した。また、ＣＰＵ制御部１２は、前記第１実施形態のＶＭ－ｅｎｔｒｙ命令発行部に代わって、物理ＣＰＵ１０４にＶＭＲＵＮ命令を発行するＶＭＲＵＮ命令発行部１２５Ａを備える。
【０１３９】
次に、ホストＶＭＭ１０が仮想サーバ１０２ａ～１０２ｎへ提供する仮想ＣＰＵ１０８ａ～１０８ｎも、物理ＣＰＵ１０４ａの構成の変更に応じて、ゲストＶＭＭ２０内のゲスト状態エリア２２１のアドレスを指し示すポインタ１１６に加えて、ホスト状態エリア２２２のアドレスを指し示すポインタ１１７を備える。なお、ゲストＶＭＭ２０は、前記第１実施形態の仮想サーバ１０２ａと同じく、仮想化機能を統合した次世代ＯＳをゲストＯＳ１１１ａに採用した場合に仮想サーバ１０２ａで稼働するものである。
【０１４０】
ゲストＶＭＭ２０の仮想ＣＰＵ制御データ２１は、ホストＶＭＭ１０のホストＶＭＭ保持データ１１と同様に、ＡＭＤ－Ｖの仕様に応じてゲストＶＭＭ２０のステータスを保持するホスト状態エリア２２２と、ユーザプログラムのステータスを格納するゲスト状態エリア２２１及び制御エリア２２６を備えたゲストＶＭＣＢ２２から構成される。
【０１４１】
以上のような構成により、ホストＶＭＭ１０は、物理ＣＰＵ１０４ａから_ＶＭＥＸＩＴを受信すると、シャドウＶＭＣＢ_０～_ｎ－１のゲスト状態エリア１３１に、ゲストＶＭＭ２０のゲスト状態エリア２２１またはホスト状態エリア２２２の内容を書き込むこと

で、仮想サーバ１０２ａで稼働する次世代ＯＳに統合されたゲストＶＭＭ２０と、このゲストＶＭＭ２０上で稼動するユーザプログラムを切り替えることができる。

【０１４２】

次に、ホストＶＭＭ１０で行われる処理について、図１２を参照しながら以下に説明する。図１２は、仮想サーバ１０２ａの稼働中に物理ＣＰＵ１０４ａからの_ＶＭＥＸＩＴを受信したときにホストＶＭＭ１０のＣＰＵ制御部１２が行う処理の一例を示すフローチャートである。なお、この例では、_ＶＭＥＸＩＴの要因がＶＭＲＵＮ命令である場合を示す。

【０１４３】

まず、Ｓ６１では、ホストＶＭＭ１０のＣＰＵ制御部１２は、_ＶＭＥＸＩＴの要因となった操作対象のゲストＯＳ１１１ａ（以下、ゲストＯＳとする）に対応するホストＶＭＭ保持データ１１の仮想化機能有効フラグ１４１を参照し、_ＶＭＥＸＩＴを発生したゲストＯＳがＡＭＤ－Ｖ機能を利用可能であるか否かを判定する。仮想化機能有効フラグ１４１が「１」であれば操作対象のゲストＯＳがＡＭＤ－Ｖ機能を利用可能であると判定してＳ６２へ進む。一方、仮想化機能有効フラグ１４１が「０」であれば操作対象のゲストＯＳがＡＭＤ－Ｖ機能を利用しないと判定してＳ７３へ進む。Ｓ７３では前記第１実施形態の図５に示したＳ１５と同様に、ホストＶＭＭ１０が前記従来例に示した仮想マシン処理を行う。

【０１４４】

Ｓ６２では、ホストＶＭＭ１０のＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードフラグ１４３を参照し、仮想ＣＰＵの動作モードがホストモード（前記第１実施形態のＶＭＸルートモード）とゲストモード（前記第１実施形態のＶＭＸノンルートモード）の何れであるかを判定する。動作モードフラグ１４３が「０」の場合には仮想ＣＰＵがホストモードであると判定してＳ６３へ進む。一方、動作モードフラグ１４３が「１」の場合には、仮想ＣＰＵがゲストモードであると判定してＳ６９へ進む。

【０１４５】

Ｓ６３では、ＣＰＵ制御部１２が物理ＣＰＵ１０４ａから受信した_ＶＭＥＸＩＴの発生要因を特定する。この例では、ホストモードであるので、ＣＰＵ制御部１２は、ゲストＶＭＭ２０の状態（ステータス）を格納したゲスト状態エリア１３１を参照し、ゲストＶＭＭ２０が発行したＶＭＲＵＮ命令を_ＶＭＥＸＩＴの要因として特定する。

【０１４６】

次に、Ｓ６４ではＣＰＵ制御部１２のエミュレータ１２２からＶＭＲＵＮ命令実行モジュールを実行し、操作対象の仮想ＣＰＵをゲストモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）をゲストＶＭＭ２０に代わってエミュレートする。

【０１４７】

次に、Ｓ６５ではホストＶＭＭ１０のＣＰＵ制御部１２は、仮想ＣＰＵの動作モードをホストモードからゲストモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「１」に更新する。

【０１４８】

そして、Ｓ６６では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０のゲストＶＭＣＢ２２からゲスト状態エリア２２１に格納されているユーザプログラム（ゲストＯＳまたはアプリケーション）のステータスを読み込む。

【０１４９】

Ｓ６７では、ＣＰＵ制御部１２が、上記Ｓ６６で読み込んだゲスト状態エリア２２１の情報で操作対象のシャドウＶＭＣＢ_０のゲスト状態エリア１３１を更新する。そして、Ｓ６８でＣＰＵ制御部１２は、物理ＣＰＵ１０４ａ（操作対象の仮想ＣＰＵ１０８ａ）に対してＶＭＲＵＮ命令を発行する。

【０１５０】

ＶＭＲＵＮ命令を受け付けた物理ＣＰＵ１０４ａは、ポインタ１１５で指定されたシャ

ドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのユーザプログラム（ゲストＯＳまたはアプリケーション）を実行する。なお、物理ＣＰＵ１０４ａは、ＶＭＲＵＮ命令を受け付けると、ホストＶＭＭ１０のステータスをホストＶＭＭ保持データ１１のホスト状態エリア１３２へ格納し、次回の呼び出しに備える。

【０１５１】

一方、上記Ｓ６２で動作モードフラグ１４３が「１」と判定されたゲストモードの場合は、操作対象の仮想ＣＰＵがユーザプログラムを実行中であるので、Ｓ６９の処理を行う。Ｓ６９では、ＣＰＵ制御部１２がＥＸＩＴＣＯＤＥを取得して、図１１に示した_ＶＭＥＸＩＴの要因リストを参照し、操作対象の仮想サーバでユーザプログラムを実行中であるので、通知条件１３７４からゲストＶＭＭへの_ＶＭＥＸＩＴ通知条件を検索する。この例では、_ＶＭＥＸＩＴの要因がＶＭＲＵＮ命令が、_ＶＭＥＸＩＴ通知条件に一致する。そして、Ｓ７０ではＣＰＵ制御部１２がエミュレータ１２２から図９の_ＶＭＥＸＩＴ条件検出モジュールを実行し、操作対象の仮想ＣＰＵをホストモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）をユーザプログラムに代わってエミュレートする。

【０１５２】

次に、Ｓ７１ではＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードをゲストモードからホストモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「０」にリセットする。

【０１５３】

そして、Ｓ７２では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０から仮想ＣＰＵ制御データ２１のホスト状態エリア２２１に格納されているゲストＶＭＭ２０のステータスを読み込む。

【０１５４】

Ｓ７２の処理が完了すると、上記Ｓ６７～Ｓ６８を実行し、ＣＰＵ制御部１２は、上記Ｓ７２で読み込んだホスト状態エリア２２２の情報で操作対象のシャドウＶＭＣＢのゲスト状態エリア１３１を更新し、物理ＣＰＵ１０４ａ（操作対象の仮想ＣＰＵ１０８ａ）に対してＶＭＲＵＮ命令を発行する。この結果、ＶＭＲＵＮ命令を受け付けた物理ＣＰＵ１０４ａは、ポインタ１１５で指定されたシャドウＶＭＣＢのゲスト状態エリア１３１の内容に基づいて、操作対象の仮想サーバのゲストＶＭＭ２０を実行する。

【０１５５】

このように、仮想サーバで稼働するゲストＯＳが仮想化機能を統合した次世代ＯＳである場合、ホストＶＭＭ１０は、仮想ＣＰＵの動作モードと、発生した_ＶＭＥＸＩＴの要因に応じてシャドウＶＭＣＢのゲスト状態エリア１３１に書き込むステータスを、ゲストＶＭＭまたはユーザプログラムの何れか一方から選択する。そして、ホストＶＭＭ１０が物理ＣＰＵ１０４ａに対してＶＭＲＵＮ命令を発行することで、仮想サーバ上で第１の仮想ＣＰＵ上で稼動するゲストＶＭＭと、このゲストＶＭＭが提供する第２の仮想ＣＰＵ上で稼動するユーザプログラムを切り替えて実行することが可能となり、ゲストＶＭＭは仮想サーバ上で仮想化環境を提供することができる。

【０１５６】

なお、上記Ｓ６１の判定でゲストＯＳが仮想化機能を利用しない場合は、Ｓ１５において、前記従来例に示した特許文献１または非特許文献１の記載に準じた仮想マシン処理をホストＶＭＭ１０で行えばよい。

【０１５７】

Ｓ７３の仮想マシン処理は、例えば、図１に示した仮想サーバ１０２ｎのゲストＯＳ１１１ｎは既存のＯＳであり、このゲストＯＳ１１１ｎまたはアプリケーション１１２ｎ（ユーザプログラム１１０ｎ）が特権命令などの所定の命令を実行すると、上述したように物理ＣＰＵ１０４ａは_ＶＭＥＸＩＴの発生をホストＶＭＭ１０に通知する。

【０１５８】

ホストＶＭＭ１０は、物理ＣＰＵ１０４ａからの_ＶＭＥＸＩＴの通知を受けると、ユ

ーザプログラム１１０ｎ（仮想ＣＰＵ１０８ｎ）のステータスをシャドウＶＭＣＢ_ｎ－１のゲスト状態エリア１３１に格納する。

【０１５９】

ホストＶＭＭ１０は、特権命令など所定の処理が完了すると、ホストＶＭＭ１０のステータスをホスト状態エリア１３２へ格納し、ポインタ１１５のアドレスにゲスト状態エリア１３１を設定してから、ＶＭＲＵＮ命令を発行して、制御を仮想ＣＰＵ１０８ｎに移動してユーザプログラム１１０ｎの実行を再開する。

【０１６０】

このように、第２の実施形態によれば、前記第１実施形態と同様に、仮想化機能を統合した次世代ＯＳをホストＶＭＭ１０上で稼動させることが可能となる。そして、仮想化機能を統合した次世代ＯＳと、既存のＯＳをひとつの物理サーバ１０１へ統合することが可能となって、物理サーバの数を削減してサーバの運用管理コストを削減することが可能となる。

【０１６１】

さらに、上述のように、ホストＶＭＭ１０は第１の仮想ＣＰＵ上で稼動するゲストＶＭＭに対して第１の仮想ＣＰＵがＡＭＤ－Ｖ機能を提供するように見せかけることができ、仮想化ソフトウェアを統合したＯＳを確実に稼働させることが可能となる。また、本発明のホストＶＭＭ１０によれば、前記従来例のシミュレータのように、命令列の変換などのオーバーヘッドがなく、仮想計算機の性能を低下させることなく、仮想化機能を統合したＯＳを仮想サーバで実行することが可能となるのである。

【０１６２】

＜第３実施形態＞

図１３は、第３の実施形態を示し、前記第１実施形態に示した物理ＣＰＵ１０４の仮想化支援機能を、ＶＴ－ｘ機能から前記従来例（非特許文献３）に示したＩＰＦ（Ｉｔａｎｉｕｍ　Ｐｒｏｃｅｓｓｏｒ　Ｆａｍｉｌｙ）のＶＴ－ｉ機能に置き換えたものである。

【０１６３】

図１３において、ＶＴ－ｉ機能を実装した物理ＣＰＵ１０４ｂでは、ゲスト（ゲストＶＭＭ２０またはユーザプログラム１１０ａ）を実行中に予め設定した命令やイベントがあると仮想化例外（Ｖｉｒｔｕａｌｉｚａｔｉｏｎ　Ｆａｕｌｔ）を発生し、ホストＶＭＭ１０に制御を移す。ホストＶＭＭ１０では、仮想化例外を検知すると、予め設定された仮想化プロシジャ（ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャなど）を実行（エミュレーション）し、仮想ＣＰＵ１０８ａから０８ｎを制御するためのシャドウＶＰＤ（Virtual Processor Descriptor）を更新してから物理ＣＰＵ１０４ｂへ仮想化プロシジャ（ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャ）を発行し、ゲストに制御を移す。なお、ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャは、ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥ＿ＮＯＲＭＡＬプロシジャ呼び出し、またはＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥ＿ＨＡＮＤＬＥＲプロシジャ呼び出しの総称とする。

【０１６４】

すなわち、物理ＣＰＵ１０４ｂは、前記第１実施形態のＶＭ－ｅｘｉｔに代わって仮想化例外を発生し、ＶＭ－ｅｎｔｒｙ命令に代わって仮想化プロシジャを実行することで、ホストＶＭＭ１０とゲストの制御を切り替えるアーキテクチャである。詳細については前記非特許文献３に詳しいので、本アーキテクチャの詳細な説明は省略する。

【０１６５】

ホストＶＭＭ１０のホストＶＭＭ保持データ１１には、前記第１実施形態と同様の仮想化機能有効フラグ１４１と動作モードフラグ１４３に加え、仮想ＣＰＵ１０８ａ～１０８ｎのステータスを格納するアーキテクチャ状態１５１を含む仮想ＣＰＵ制御データＢ１５０を設ける。

【０１６６】

そして、物理ＣＰＵ制御データ１３は、ＶＴ－ｉ機能の仕様に応じて以下の情報を格納する。物理ＣＰＵ制御データ１３には、仮想ＣＰＵ１０８ａ～１０８ｎ毎のステータスを

格納するアーキテクチャ状態１３１０を含むシャドウＶＰＤ_０～_ｎ－１と、仮想ＣＰＵ１０８ａ～１０８ｎ毎に仮想化例外の発生の抑止を設定するＶＤＣ（Virtualization Disable Control）領域_０～_ｎ－１（１３２０）と、仮想ＣＰＵ１０８ａ～１０８ｎが実行する命令またはイベントのうち、物理ＣＰＵ１０４ｂで仮想化例外を発生する命令またはイベントを予め定義した仮想化例外要因１３３０と、を備える。

【０１６７】

ホストＶＭＭ保持データ１１に格納されるアーキテクチャ状態は、図１４で示すようになる。すなわち、仮想ＣＰＵ制御データＢ１５０のアーキテクチャ状態１５１には、仮想ＣＰＵ１０８ａ～１０８ｎが実行するゲスト（ゲストＶＭＭ２０またはユーザプログラム）のステータスが格納され、シャドウＶＰＤのアーキテクチャ状態１３１０には、物理ＣＰＵ１０４ｂで実行されるゲストまたはホストＶＭＭ１０のステータスが格納される。

【０１６８】

また、ホストＶＭＭ保持データ１１の物理ＣＰＵ制御データ１３に格納される仮想化例外要因１３３０は、例えば、図１５の仮想化例外要因リストで示すように設定される。図１５において、仮想化例外要因リストには、仮想化例外の要因を示すコード１３３１と、コードに対応する仮想化例外の内容１３３２と、制御をゲストへ移すためのＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャの呼び出し設定１３３３と、ゲストＶＭＭ２０への仮想化例外の通知設定１３３４から構成される。設定１３３３では、図中「○」が設定された命令のときに、制御をゲストへ移すＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを呼び出すことを意味する。また、通知設定１３３４は、図中「○」が設定された命令のときに、ゲストＶＭＭ２０へ通知を行うことを意味し、ゲストＶＭＭ２０のＶＤＣ領域２２０２が仮想化例外の発生を抑止していないときにゲストＶＭＭ２０への通知を行うことを意味する。

【０１６９】

ゲストＶＭＭ２０が保持する仮想ＣＰＵ１０８ａの状態は、仮想ＣＰＵ制御データＡ２１０に格納される。仮想ＣＰＵ制御データＡ２１０は、仮想ＣＰＵ１０８ａのステータスを格納するアーキテクチャ状態２２１０を含むゲストＶＰＤ２２０と、ゲストＯＳまたはアプリケーションが実行した命令またはイベントのうち、仮想ＣＰＵ１０８ａで仮想化例外を発生する命令またはイベントを定義した仮想化例外要因２２０１と、ゲストＶＭＭ２０を実行する仮想ＣＰＵ１０８ａがＶＴ－ｉ機能を利用するか否かを示すＶＤＣ領域２２０２とを含む。

【０１７０】

ＣＰＵ制御部１２は、物理ＣＰＵ１０４ｂからの仮想化例外を検知し、エミュレータ１２２へ通知する仮想化例外ハンドラ１２４０と、物理ＣＰＵ１０４ｂが提供するＰＡＬ（Processor Abstraction Layer）で仮想化例外となった処理を実行させるエミュレータ１２２と、仮想ＣＰＵ１０８ａの稼働状態基づいて、ゲストＶＰＤのアーキテクチャ状態２２１０と、ホストＶＭＭ保持データ１１の仮想ＣＰＵ制御データＢ１５０のアーキテクチャ状態１５１の何れかを選択する状態エリア選択部１２１と、状態エリア選択部１２１で選択された情報でシャドウＶＰＤのアーキテクチャ状態１３１０を更新するシャドウＶＰＤ更新部１２５０と、ホストＶＭＭ１０からゲストＶＭＭ２０またはゲストＶＭＭ２０上の第２の仮想ＣＰＵ２０８ａ上で稼動するユーザプログラムへ制御を切り替えるため、物理ＣＰＵ１０４ｂへ所定のプロシジャの実行を指令する仮想化プロシジャ指令部１２５０と、を備える。

【０１７１】

次に、ホストＶＭＭ１０のＣＰＵ制御部１２で行われる処理の一例について、図１６のフローチャートを参照しながら以下に説明する。なお、以下の説明では、前記第１実施形態の図１に示した仮想サーバ１０２ａで次世代ＯＳのゲストＯＳ１１１ａとアプリケーション１１２ａ及びゲストＶＭＭ２０が稼働している例について説明する。

【０１７２】

図１６は、仮想サーバ１０２ａの稼働中に物理ＣＰＵ１０４ｂからの仮想化例外を受信したときにホストＶＭＭ１０のＣＰＵ制御部１２が行う処理の一例を示すフローチャート

である。なお、この例では、仮想化例外の要因がＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャからのｖｍｓｗ命令である場合を示す。

【０１７３】

まず、Ｓ８１では、ホストＶＭＭ１０のＣＰＵ制御部１２が、仮想化例外の要因となった操作対象のゲストＯＳ１１１ａ（以下、ゲストＯＳとする）に対応するホストＶＭＭ保持データ１１の仮想化機能有効フラグ１４１を参照し、仮想化例外を発生したゲストＯＳがＶＴ－ｉ機能を利用可能であるか否かを判定する。仮想化機能有効フラグ１４１が「１」であれば操作対象のゲストＯＳがＶＴ－ｉ機能を利用可能であると判定してＳ８２へ進む。一方、仮想化機能有効フラグ１４１が「０」であれば操作対象のゲストＯＳがＶＴ－ｉ機能を利用しないと判定してＳ９４へ進む。Ｓ９４では、ホストＶＭＭ１０が前記従来例の非特許文献３に示したＶＴ－ｉ機能を用いて仮想マシン処理を行う。

【０１７４】

Ｓ８２では、ホストＶＭＭ１０のＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードフラグ１４３を参照し、仮想ＣＰＵの動作モードがホストモード（前記第１実施形態のＶＭＸルートモード）とゲストモード（前記第１実施形態のＶＭＸノンルートモード）の何れであるかを判定する。動作モードフラグ１４３が「０」の場合には仮想ＣＰＵがゲストＶＭＭ２０を実行するホストモードであると判定してＳ８３へ進む。一方、動作モードフラグ１４３が「１」の場合には、仮想ＣＰＵがユーザプログラムを実行するゲストモードであると判定してＳ９０へ進む。

【０１７５】

Ｓ８３では、ＣＰＵ制御部１２が物理ＣＰＵ１０４ｂから受信した仮想化例外の発生要因を特定する。この例では、ホストモードであるので、ＣＰＵ制御部１２は、ゲストＶＭＭ２０の状態（ステータス）を格納したホストＶＭＭ保持データ１１内の仮想ＣＰＵ制御データＢ１５０のアーキテクチャ状態１５１を参照し、ゲストＶＭＭ２０がＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャから発行したｖｍｓｗ命令を仮想化例外の要因として特定する。

【０１７６】

次に、Ｓ８４ではＣＰＵ制御部１２が、仮想化例外の要因となったＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャモジュールを実行し、所定の処理をゲストＶＭＭ２０に代わって実行し、操作対象の仮想ＣＰＵをゲストモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）を実行する。

【０１７７】

次に、Ｓ８５ではＣＰＵ制御部１２が、仮想ＣＰＵの動作モードをホストモードからゲストモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「１」に更新する。

【０１７８】

そして、Ｓ８６では、ＣＰＵ制御部１２が操作対象のゲストＶＭＭ２０の仮想ＣＰＵ制御データＡ２１０内のゲストＶＰＤ２２０からアーキテクチャ状態２２１０に格納されているユーザプログラム（ゲストＯＳまたはアプリケーション）のステータスを読み込む。

【０１７９】

Ｓ８７では、ＣＰＵ制御部１２が物理ＣＰＵ１０４ｂに対してＰＡＬ＿ＶＰＳ＿ＳＡＶＥ／ＲＥＳＴＯＲＥプロシジャを発行し、仮想化例外の発行要因となったシャドウＶＰＤ_０を選択する。

【０１８０】

次に、Ｓ８８では、ＣＰＵ制御部１２が、上記Ｓ８６で読み込んだアーキテクチャ状態２２１０のステータスを、上記Ｓ８７で選択した操作対象のシャドウＶＰＤ_０のアーキテクチャ状態１３１０に書き込んで更新する。そして、Ｓ８８においてＣＰＵ制御部１２は、物理ＣＰＵ１０４ｂに対してＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを発行する。

【０１８１】

ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを受け付けた物理ＣＰＵ１０４ｂは、Ｓ８７

で選択されたシャドウＶＰＤのアーキテクチャ状態１３１０の内容に基づいて、操作対象の仮想サーバのユーザプログラム（ゲストＯＳまたはアプリケーション）を実行する。

【０１８２】

一方、上記Ｓ８２で動作モードフラグ１４３が「１」と判定されたゲストモードの場合は、操作対象の仮想ＣＰＵがユーザプログラムを実行中であるので、Ｓ９０の処理を行う。Ｓ９０では、ＣＰＵ制御部１２が仮想化例外の要因を取得して、図１５に示した仮想化例外の要因リストを参照し、通知条件１３３４からゲストＶＭＭへの仮想化例外通知条件を検索する。この例では、仮想化例外の要因がｖｍｓｗ命令であるので、仮想化例外通知条件はゲストＶＭＭ２０のＶＤＣ領域２２０２の設定によるものとなる。ここでは、ゲストＶＭＭ２０へ仮想化例外を通知するものとして扱う。そして、Ｓ９１では図１３の仮想化例外条件検出モジュールを実行し、操作対象の仮想ＣＰＵをホストモードへ切り替えるのに必要な所定の処理（状態エリア選択部１２１の起動など）をゲストＶＭＭ２０に代わってエミュレートする。

【０１８３】

次に、Ｓ９２ではＣＰＵ制御部１２が、操作対象の仮想ＣＰＵの動作モードをゲストモードからホストモードへ移行させるので、操作対象の仮想ＣＰＵの動作モードフラグ１４３を「０」にリセットする。

【０１８４】

そして、Ｓ９３では、ＣＰＵ制御部１２がホストＶＭＭ保持データ１１内の操作対象の仮想ＣＰＵに対応する仮想ＣＰＵ制御データＢ１５０のアーキテクチャ状態１５１に格納されているゲストＶＭＭ２０のステータスを読み込む。

【０１８５】

Ｓ９３の処理が完了すると、上記Ｓ８７～Ｓ８９を実行し、ＣＰＵ制御部１２は、上記Ｓ９３で読み込んだホストＶＭＭ保持データ１１の仮想ＣＰＵ制御データＢ１５０のアーキテクチャ状態１５１の情報で操作対象のシャドウＶＰＤのアーキテクチャ状態１３１０を更新し、物理ＣＰＵ１０４ｂに対してＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを発行する。この結果、ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを受け付けた物理ＣＰＵ１０４ｂは、シャドウＶＰＤ_０のアーキテクチャ状態１３１０の内容に基づいて、操作対象の仮想サーバのゲストＶＭＭ２０を実行する。

【０１８６】

このように、仮想サーバで稼働するゲストＯＳが仮想化機能を統合した次世代ＯＳである場合、ホストＶＭＭ１０は、仮想ＣＰＵの動作モードと、発生した仮想化例外の要因に応じて、シャドウＶＰＤのアーキテクチャ状態１３１０に書き込むステータスを、ゲストＶＭＭまたはユーザプログラムの何れか一方から選択する。そして、ホストＶＭＭ１０が物理ＣＰＵ１０４ｂに対してＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを発行することで、仮想サーバ上でゲストＶＭＭと、このゲストＶＭＭが提供する第２の仮想ＣＰＵ２０８ａ上で稼動するユーザプログラムを切り替えて実行することが可能となり、ゲストＶＭＭは仮想サーバ上で仮想化環境を提供することができる。

【０１８７】

なお、上記Ｓ８１の判定でゲストＯＳが仮想化機能を利用しない場合は、Ｓ９４において、前記従来例に示した特許文献１または非特許文献３の記載に準じた仮想マシン処理をホストＶＭＭ１０で行えばよい。

【０１８８】

Ｓ９４の仮想マシン処理は、例えば、図１に示した仮想サーバ１０２ｎのゲストＯＳ１１１ｎは既存のＯＳであり、このゲストＯＳ１１１ｎまたはアプリケーション１１２ｎ（ユーザプログラム１１０ｎ）が仮想化例外要因の命令を実行すると、上述したように物理ＣＰＵ１０４ｂは仮想化例外の発生をホストＶＭＭ１０に通知する。

【０１８９】

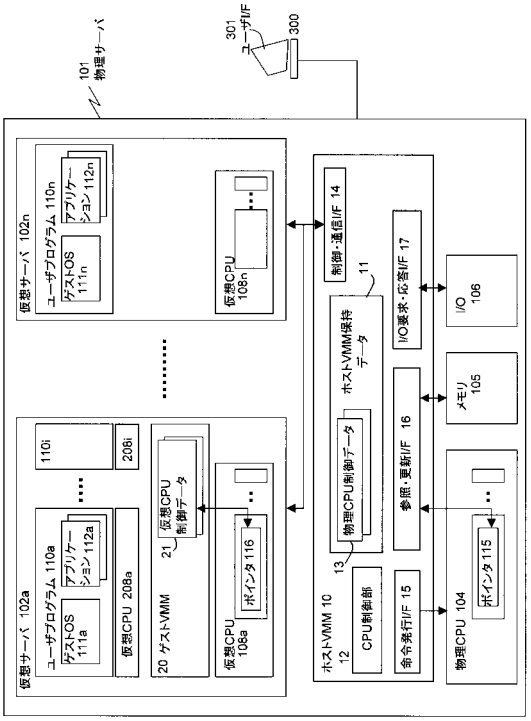
ホストＶＭＭ１０は、物理ＣＰＵ１０４ｂからの仮想化例外の通知を受けると、ユーザプログラム１１０ｎ（仮想ＣＰＵ１０８ｎ）のステータスを仮想ＣＰＵ制御データＢ１５

０のアーキテクチャ状態１５１に格納する。
【０１９０】
ホストＶＭＭ１０は、仮想化例外の要因となった命令をエミュレートした後、ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャを発行して、制御を仮想ＣＰＵ１０８ｎに移動してユーザプログラム１１０ｎの実行を再開する。
【０１９１】
このように、第３の実施形態によれば、前記第１実施形態と同様に、仮想化機能を統合した次世代ＯＳと、既存のＯＳをひとつの物理サーバ１０１へ統合することが可能となって、物理サーバの数を削減してサーバの運用管理コストを削減することが可能となる。
【０１９２】
さらに、上述のように、ホストＶＭＭ１０は次世代ＯＳに対して仮想ＣＰＵがＶＴ－ｉ機能を提供するように見せかけることができ、仮想化ソフトウェアを統合したＯＳを確実に稼働させることが可能となる。また、本発明のホストＶＭＭ１０によれば、前記従来例のシミュレータのように、命令列の変換などのオーバーヘッドがなく、仮想計算機の性能を低下させることなく、仮想化機能を統合したＯＳを仮想サーバで実行することが可能となるのである。
【０１９３】
なお、上記各実施形態において、物理ＣＰＵ１０４、１０４ａ、１０４ｂとして記載したプロセッサは、マルチコアプロセッサの構成でも良く、また、ホモジニアスなプロセッサやヘテロジニアスのプロセッサを採用することができる。すなわち、物理ＣＰＵ１０４、１０４ａ、１０４ｂとして複数の汎用プロセッサコア（ＣＰＵ）と特定用途プロセッサコアを含むヘテロジニアス・マルチコア・プロセッサを用いる場合は、汎用プロセッサコアが仮想化支援機能を備えていれば本発明を適用することができる。
【産業上の利用可能性】
【０１９４】
以上のように、本発明は、複数の仮想サーバを提供する仮想計算機システムに適用することができる。また、本発明は、物理計算機上で複数の仮想サーバを提供する仮想化管理（ＶＭＭ）ソフトウェアに適用することができる。
【図面の簡単な説明】
【０１９５】
【図１】第１の実施形態を示し、本発明を適用する仮想計算機システムのブロック図である。
【図２】第１の実施形態を示し、ホストＶＭＭとゲストＶＭＭの機能を示すブロック図である。
【図３】第１の実施形態を示し、ホストＶＭＭ保持データのシャドウＶＭＣＢの制御データの一例を示すブロック図である。
【図４】第１の実施形態を示し、ＶＭ－ｅｘｉｔの要因リストの一例を示す説明図である。
【図５】第１の実施形態を示し、ホストＶＭＭで行われる処理の一例を示すフローチャートで、ＶＭ－ｅｎｔｒｙ命令に起因するＶＭ－ｅｘｉｔの処理を示す。
【図６】第１の実施形態を示し、ホストＶＭＭで行われる処理の一例を示すフローチャートで、ＶＭＣＬＥＡＲ命令に起因するＶＭ－ｅｘｉｔの処理を示す。
【図７】第１の実施形態を示し、ホストＶＭＭで行われる処理の一例を示すフローチャートで、ＣＰＵＩＤ命令に起因するＶＭ－ｅｘｉｔの処理を示す。
【図８】第１の実施形態を示し、ユーザーインターフェースの一例を示す画面イメージである。
【図９】第２の実施形態を示し、物理ＣＰＵがＡＭＤ－Ｖ機能をサポートする場合のホストＶＭＭとゲストＶＭＭの機能を示すブロック図である。
【図１０】第２の実施形態を示し、ホストＶＭＭ保持データのシャドウＶＭＣＢの制御データの一例を示すブロック図である。

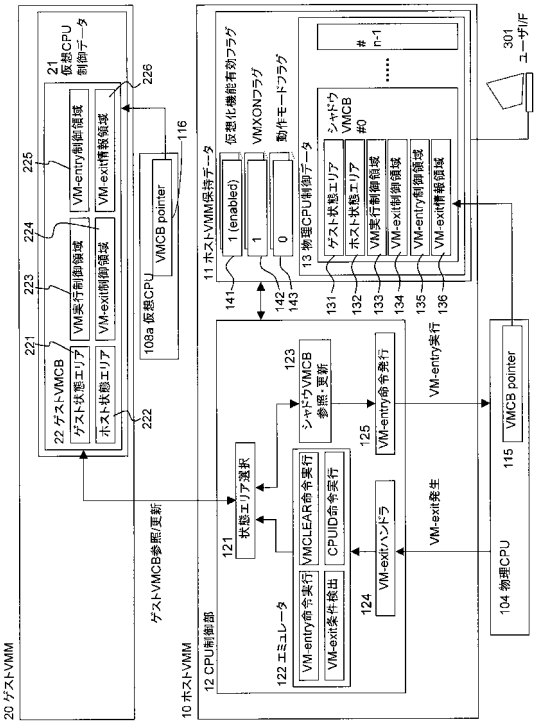
【図１１】第２の実施形態を示し、_ＶＭＥＸＩＴの要因リストの一例を示す説明図である。
【図１２】第２の実施形態を示し、ホストＶＭＭで行われる処理の一例を示すフローチャートで、ＶＭＲＵＮ命令に起因する_ＶＭＥＸＩＴの処理を示す。
【図１３】第３の実施形態を示し、物理ＣＰＵがＶＴ－ｉ機能をサポートする場合のホストＶＭＭとゲストＶＭＭの機能を示すブロック図である。
【図１４】第３の実施形態を示し、ホストＶＭＭ保持データのシャドウＶＰＤ及び仮想ＣＰＵ制御データＢの制御データの一例を示すブロック図である。
【図１５】第３の実施形態を示し、仮想化例外の要因リストの一例を示す説明図である。
【図１６】第３の実施形態を示し、ホストＶＭＭで行われる処理の一例を示すフローチャートで、ＰＡＬ＿ＶＰＳ＿ＲＥＳＵＭＥプロシジャ実行に起因する仮想化例外処理を示す。

【符号の説明】
【０１９６】
１０　ホストＶＭＭ
１１　ホストＶＭＭ保持データ
１２　ＣＰＵ制御部
１３　物理ＣＰＵ制御データ
２０　ゲストＶＭＭ
２１　仮想ＣＰＵ制御データ
１０１　物理サーバ（物理計算機）
１０２ａ～１０２ｎ　仮想サーバ
１０８ａ～１０８ｎ　仮想ＣＰＵ
１０４　ＣＰＵ
１０５　メモリ
１１０ａ～１１０ｎ　ユーザプログラム
１１１ａ～１１１ｎ　ゲストＯＳ
１１２ａ～１１２ｎ　アプリケーション
２０８　仮想ＣＰＵ
３０１　ユーザーインターフェース

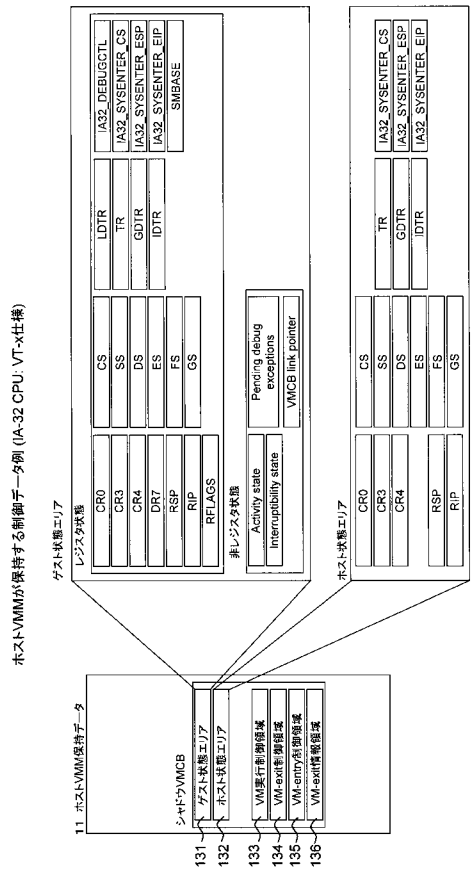
【図１】


101 物理サーバ
301 ユーザI/F
300
仮想サーバ 102a
ユーザプログラム 110a
ゲストOS 111a
アプリケーション 112a
仮想CPU 208a
20 ゲストVMM
21 仮想CPU制御データ
仮想CPU 108a
ポインタ 116
110i
206i
仮想サーバ 102n
ユーザプログラム 110n
ゲストOS 111n
アプリケーション 112n
仮想CPU 108n
ホストVMM 10
12 CPU制御部
13 物理CPU制御データ
ホストVMM保持データ 11
命令実行I/F 15
参照・更新I/F 16
制御・通信I/F 14
I/O要求・応答I/F 17
物理CPU 104
ポインタ 115
メモリ 105
I/O 106

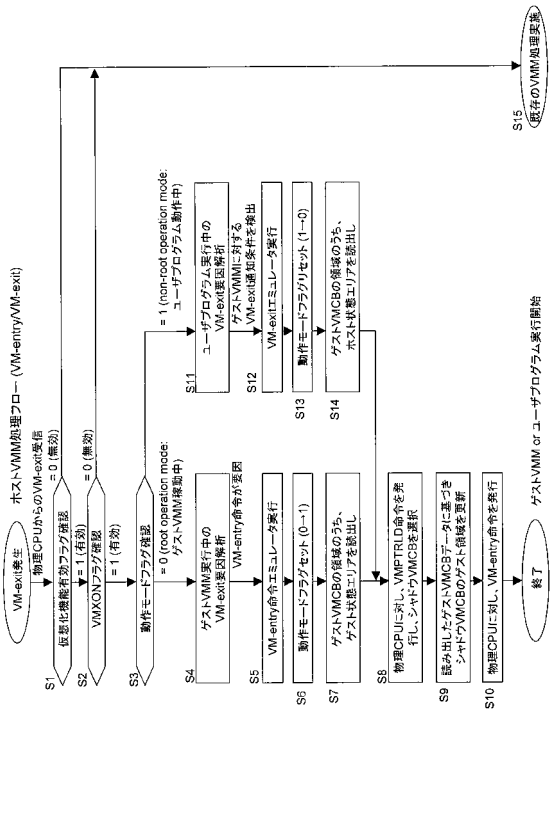
【図２】


20 ゲストVMM
22 ゲストVMCB
ゲスト状態エリア
ホスト状態エリア
VM実行制御領域
VM-exit制御領域
VM-exit情報領域
21 仮想CPU制御データ
VM-entry制御領域
VM-exit情報領域
226
VMCB pointer
116
108a 仮想CPU
222
ゲストVMCB参照・更新
10 ホストVMM
12 CPU制御部
121 状態エリア選択
122 エミュレータ
VM-entry命令実行
VMCLEAR命令実行
VM-exit条件検出
CPUID命令実行
124 VM-exitハンドラ
123 シャドウVMCB参照・更新
125 VM-entry命令実行
VM-exit発生
VM-entry実行
11 ホストVMM保持データ
仮想化機能有効フラグ
1 (enabled)
VMXONフラグ
動作モードフラグ
13 物理CPU制御データ
ゲスト状態エリア
ホスト状態エリア
VM実行制御領域
VM-exit制御領域
VM-entry制御領域
VM-exit情報領域
シャドウVMCB #0
n-1
141
142
143
131
132
133
134
135
136
104 物理CPU
115 VMCB pointer
301 ユーザI/F

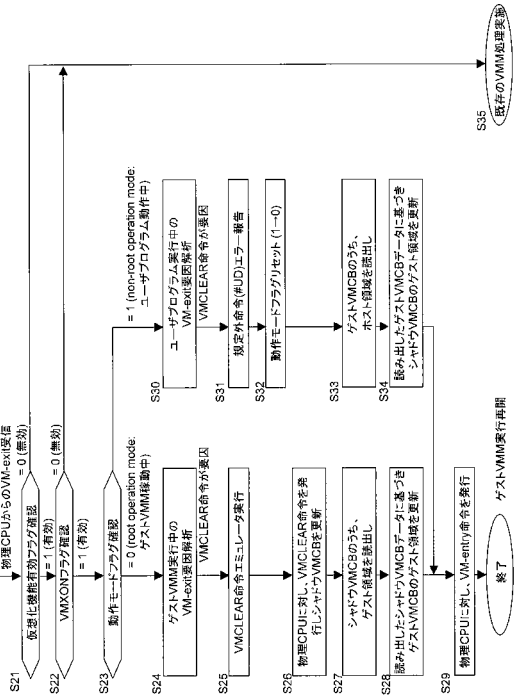
【図３】


ホストVMMが保持する制御データ例 (IA-32 CPU, VT-x仕様)
ゲスト状態エリア
レジスタ状態
CR0
CR3
CR4
DR7
RSP
RIP
RFLAGS
CS
SS
DS
ES
FS
GS
LDTR
TR
GDTR
IDTR
IA32_DEBUGCTL
IA32_SYSENTER_CS
IA32_SYSENTER_ESP
IA32_SYSENTER_EIP
SMBASE
非レジスタ状態
Activity state
Interruptibility state
Pending debug exceptions
VMCB link pointer
ホスト状態エリア
11 ホストVMM保持データ
シャドウVMCB
ゲスト状態エリア
ホスト状態エリア
VM実行制御領域
VM-exit制御領域
VM-entry制御領域
VM-exit情報領域
131
132
133
134
135
136

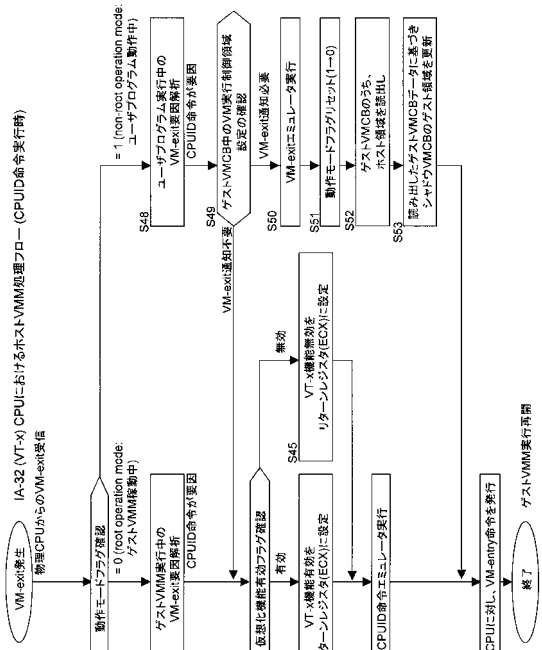
【図５】


ホストVMM処理フロー (VM-entry/VM-exit)
VM-exit発生
物理CPUからのVM-exit受信
S1
仮想化機能有効フラグ確認
= 1 (有効)
= 0 (無効)
S2
VMXONフラグ確認
= 1 (有効)
= 0 (無効)
S3
動作モードフラグ確認
= 0 (root operation mode: ゲスト-VMM稼動中)
= 1 (non-root operation mode: ユーザプログラム動作中)
S4
ゲストVMM実行中のVM-exit原因解析
VM-entry命令が要因
S5
VM-entry命令エミュレータ実行
S6
動作モードフラグセット (0→1)
S7
ゲストVMCBの領域のうち、ゲスト状態エリアを読出し
S8
物理CPUに対し、VMPTRLD命令を発行し、シャドウVMCBを選択
S9
読み出したゲストVMCBデータに基づきシャドウVMCBのゲスト領域を更新
S10
物理CPUに対し、VM-entry命令を発行
終了
ゲストVMM or ユーザプログラム実行開始
S11
ユーザプログラム実行中のVM-exit原因解析
S12
ゲストVMMに対するVM-exit (継続) 条件を検出
VM-exitエミュレータ実行
S13
動作モードフラグリセット (1→0)
S14
ゲストVMCBの領域のうち、ホスト状態エリアを読出し
S15
終了のVMM処理実施

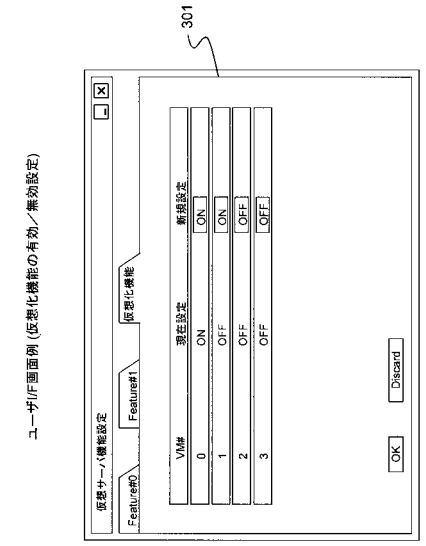
【図６】


IA-32 (VT-x) CPUにおけるホストVMM処理フロー (VMCLEAR)
VM-exit発生
物理CPUからのVM-exit受信
仮想化機能有効フラグ確認
= 0 (無効)
= 1 (有効)
VMXONフラグ確認
= 0 (無効)
= 1 (有効)
動作モードフラグ確認
= 0 (root operation mode: ゲストVMM稼動中)
= 1 (non-root operation mode: ユーザプログラム動作中)
ゲストVMM実行中の VM-exit要因解析
VMCLEAR命令が要因
ユーザプログラム実行中の VM-exit要因解析
VMCLEAR命令が要因
VMCLEAR命令エミュレータ実行
規定外命令(#UD)エラー報告
動作モードフラグリセット (1→0)
物理CPUに対し、VMCLEAR命令を発行しシャドウVMCBを更新
シャドウVMCBのうち、ゲスト領域を読出し
読み出したシャドウVMCBデータに基づきゲストVMCBのゲスト領域を更新
ゲストVMCBのうち、ホスト領域を読出し
読み出したゲストVMCBデータに基づきシャドウVMCBのゲスト領域を更新
物理CPUに対し、VM-entry命令を発行
終了
ゲストVMM実行再開
既存のVMM処理実施
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35

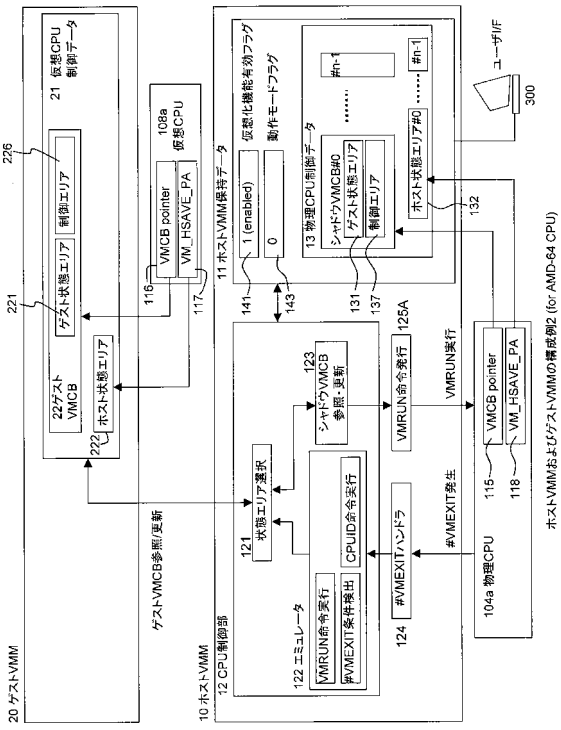
【図７】


IA-32 (VT-x) CPUにおけるホストVMM処理フロー (CPUID命令実行時)
VM-exit発生
物理CPUからのVM-exit受信
動作モードフラグ確認
= 0 (root operation mode: ゲストVMM稼動中)
= 1 (non-root operation mode: ユーザプログラム動作中)
ゲストVMM実行中の VM-exit要因解析
CPUID命令が要因
ユーザプログラム実行中の VM-exit要因解析
CPUID命令が要因
仮想化機能有効フラグ確認
有効
無効
VM-exit通知不要
VM-exit通知必要
VT-x機能有効を リターンレジスタ(ECX)に設定
VT-x機能無効を リターンレジスタ(ECX)に設定
CPUID命令エミュレータ実行
VM-exitエミュレータ実行
動作モードフラグリセット(1→0)
ゲストVMCBのうち、ホスト領域を読出し
読み出したゲストVMCBデータに基づきシャドウVMCBのゲスト領域を更新
ゲストVMCB中のVM実行制御領域設定の情報
物理CPUに対し、VM-entry命令を発行
終了
ゲストVMM実行再開
S41
S42
S43
S44
S45
S46
S47
S48
S49
S50
S51
S52
S53

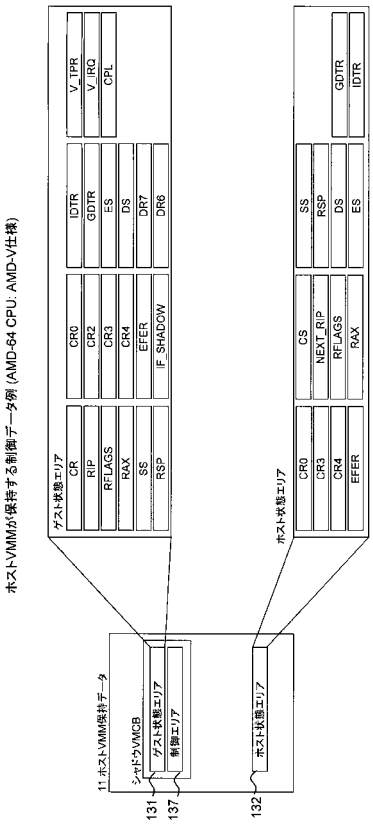
【図８】


ユーザI/F画面例 (仮想化機能の有効／無効設定)
301
仮想サーバ機能設定
Feature#0
Feature#1
仮想化機能
VM#
現在設定
新規設定
0
ON
ON
1
OFF
ON
2
OFF
OFF
3
OFF
OFF
OK
Discard

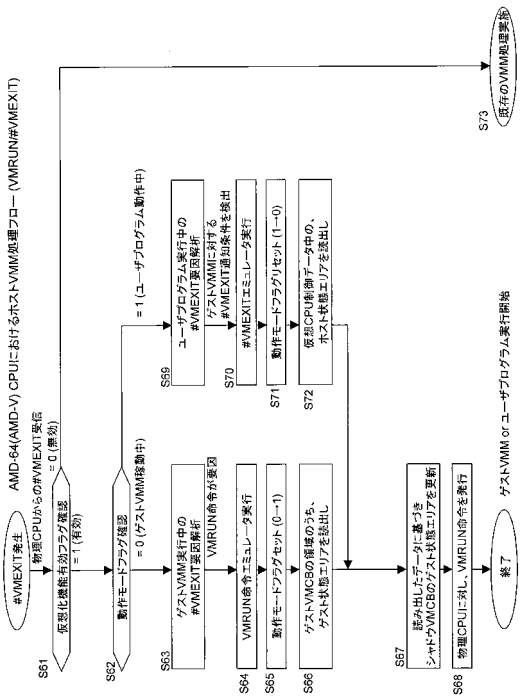
【図９】


20 ゲストVMM
21 仮想CPU制御データ
22ゲストVMCB
ゲスト状態エリア
制御エリア
ホスト状態エリア
222
221
226
116
117
VMCB pointer
VM_HSAVE_PA
108a 仮想CPU
ゲストVMCB参照・更新
10 ホストVMM
12 CPU制御部
11 ホストVMM保持データ
仮想化機能有効フラグ
1 (enabled)
0
141
143
動作モードフラグ
13 物理CPU制御データ
シャドウVMCB#0
ゲスト状態エリア
制御エリア
#n-1
ホスト状態エリア#0
ホスト状態エリア#n-1
131
137
132
ユーザI/F
300
121 状態エリア選択
122 エミュレータ
VMRUN命令実行
#VMEXIT条件検出
CPUID命令実行
123 シャドウVMCB参照・更新
125A VMRUN命令発行
VMRUN実行
124 #VMEXITハンドラ
#VMEXIT発生
104a 物理CPU
115
118
VMCB pointer
VM_HSAVE_PA
ホストVMMおよびゲストVMMの構成例2 (for AMD-64 CPU)

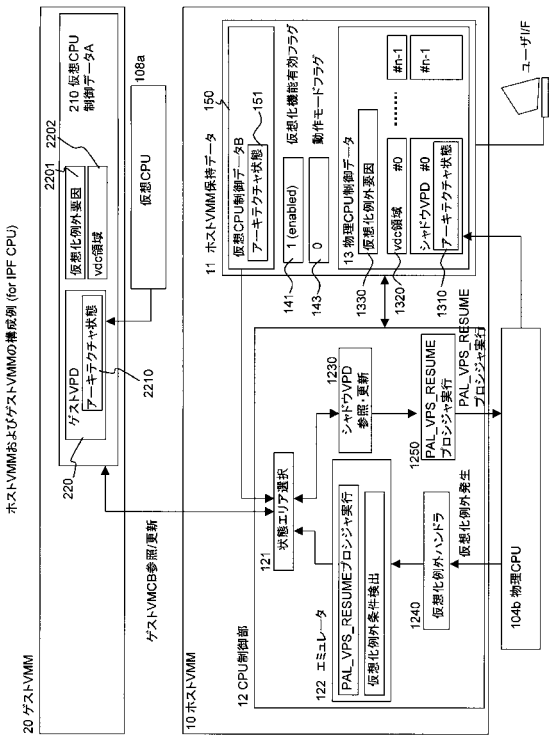
【図１０】



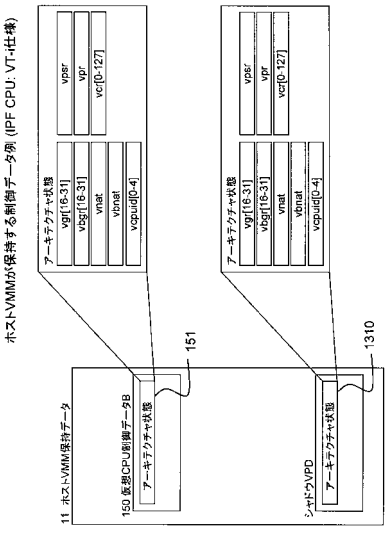
【図１２】



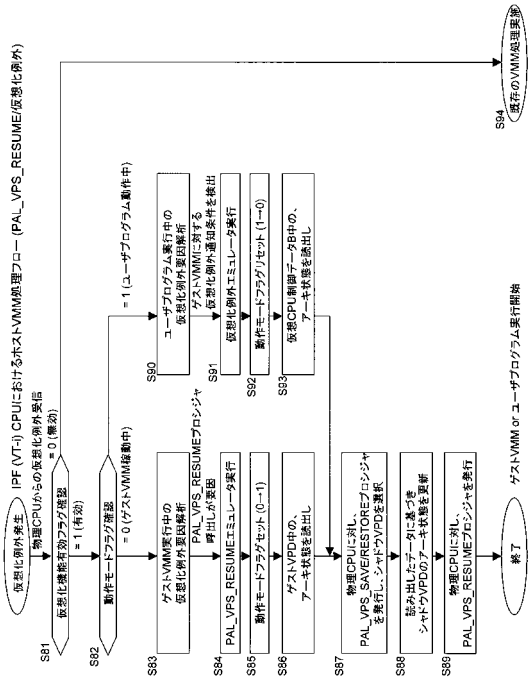
【図１３】



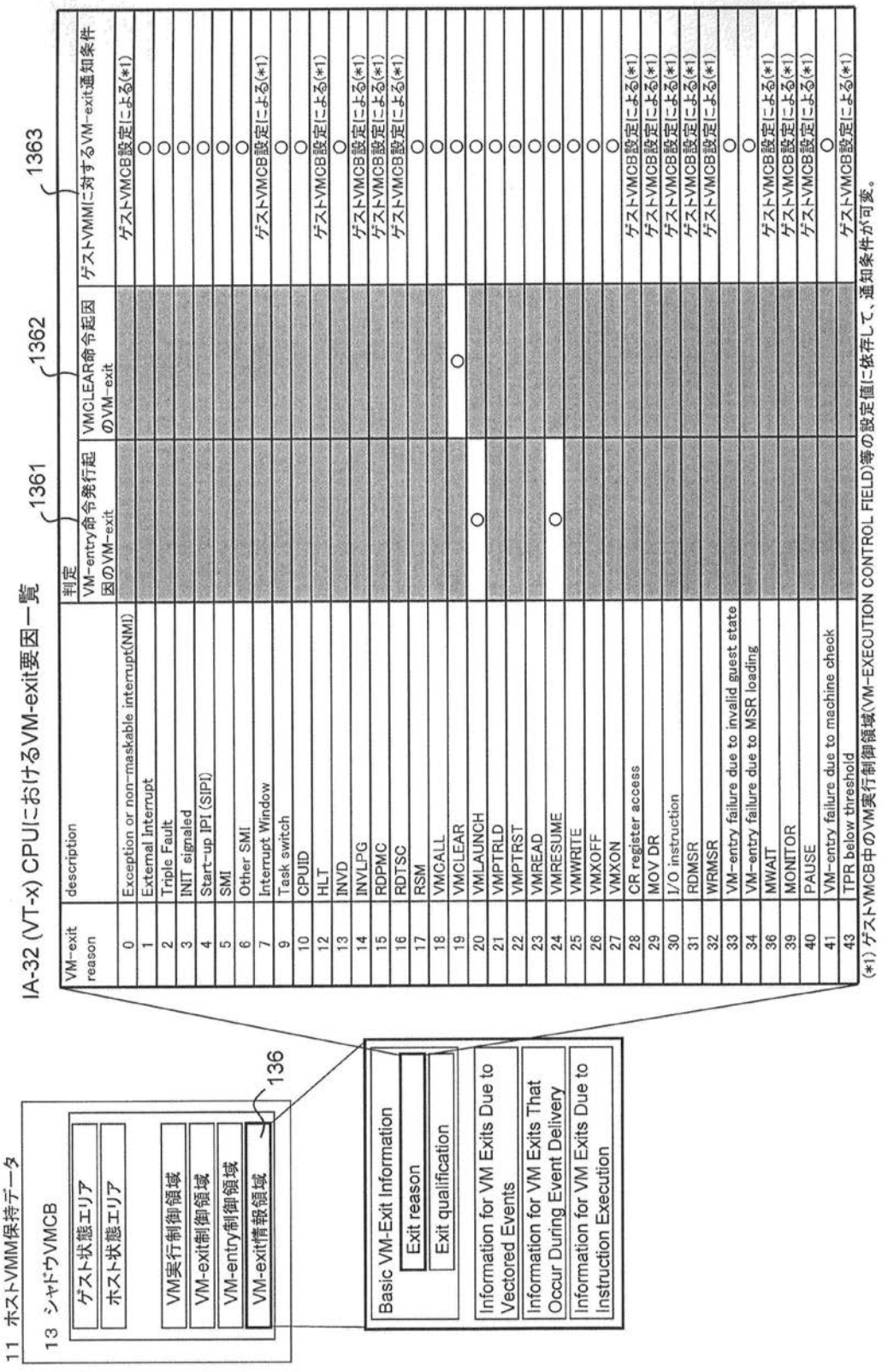
【図１４】



【図１６】


仮想化例外発生
IPF (VT-i) CPUにおけるホストVMM処理フロー (PAL_VPS_RESUME/仮想化例外)
物理CPUからの仮想化例外要因
= 0 (無効)
= 1 (有効)
仮想化機能有効フラグ確認
動作モードフラグ確認
= 0 (ゲストVMM稼動中)
= 1 (ユーザプログラム動作中)
S81
S82
S83
S84
S85
S86
S87
S88
S89
S90
S91
S92
S93
S94
ゲストVMM実行中の仮想化例外要因解析
PAL_VPS_RESUMEプロシジャ呼出しが要因
PAL_VPS_RESUMEエミュレータ実行
動作モードフラグセット (0→1)
ゲストVPD中の、アーキ状態を読出し
物理CPUに対し、PAL_VPS_SAVE/RESTOREプロシジャを実行し、シャドウVPDを選択
読み出したデータに基づきシャドウVPDのアーキ状態を更新
物理CPUに対し、PAL_VPS_RESUMEプロシジャを実行
終了
ユーザプログラム実行中の仮想化例外要因解析
ゲストVMMに対する仮想化例外通知条件を検出
仮想化例外エミュレータ実行
動作モードフラグリセット (1→0)
仮想CPU制御データBH中の、アーキ状態を読出し
既存のVMM処理実施
ゲストVMM or ユーザプログラム実行開始

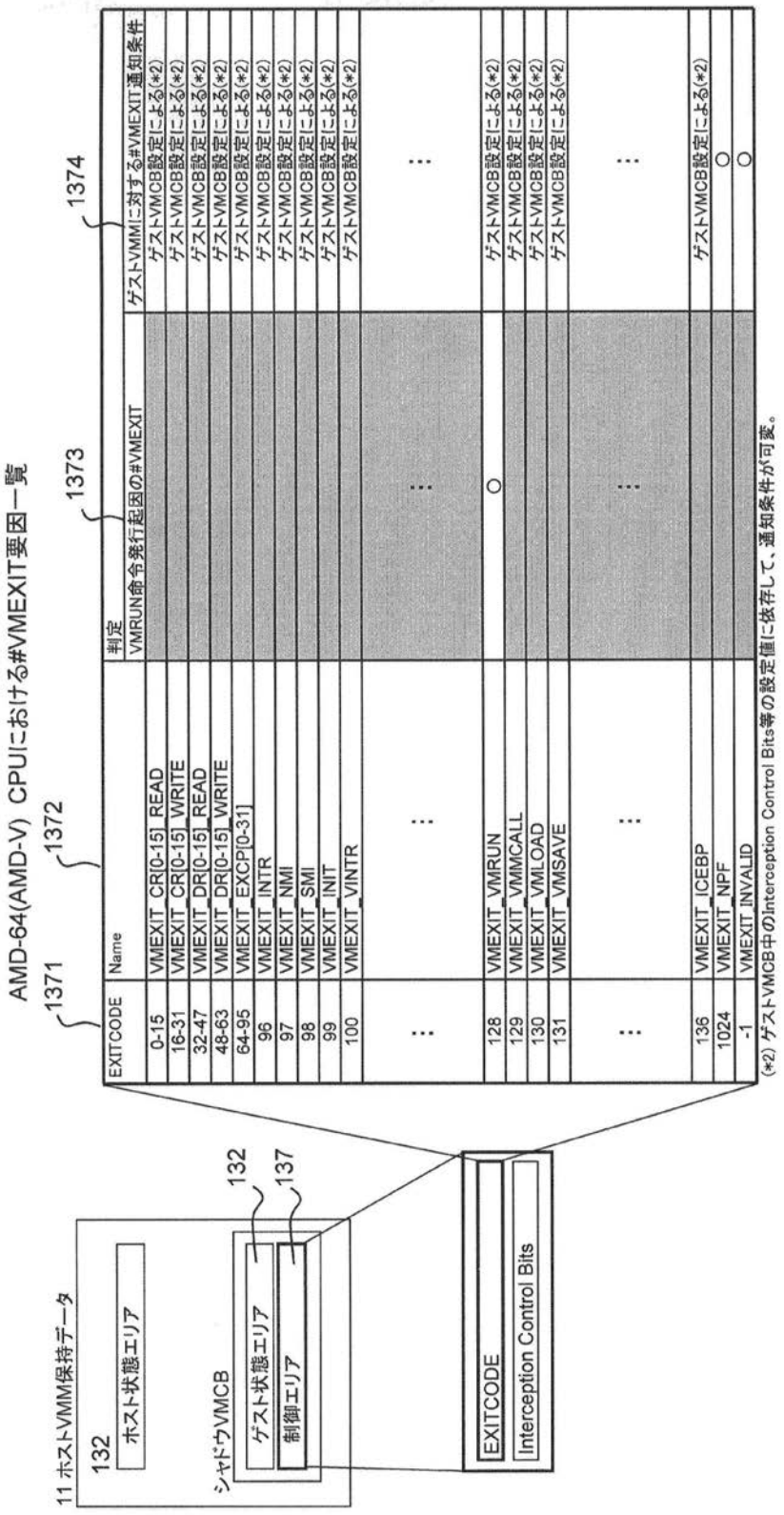
【図４】



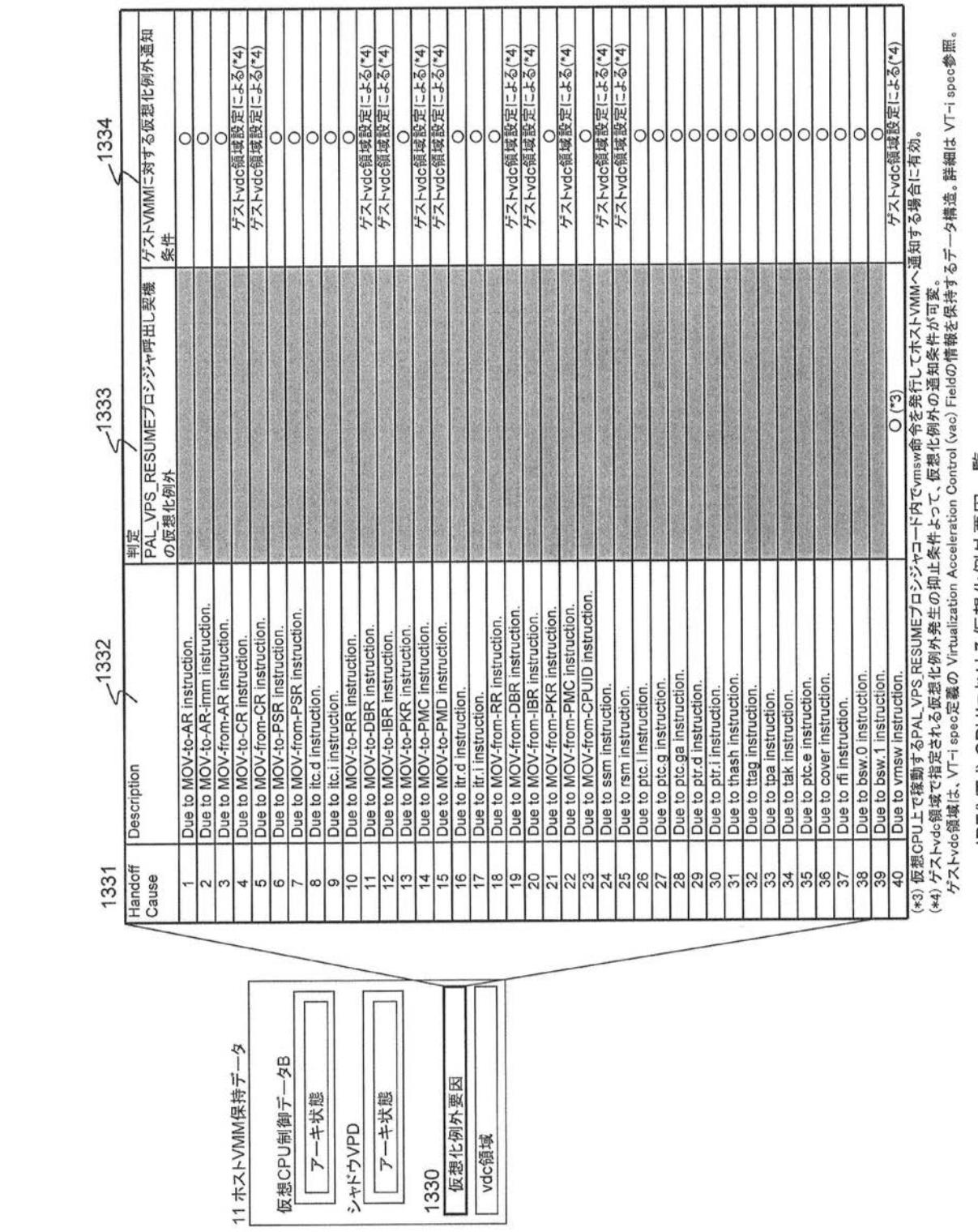
VM-exit reason	description	判定 VM-entry命令発行起因のVM-exit	VMCLEAR命令起因のVM-exit	ゲストVMMに対するVM-exit通知条件
0	Exception or non-maskable interrupt(NMI)			ゲストVMCB設定による(*1)
1	External Interrupt			○
2	Triple Fault			○
3	INIT signaled			○
4	Start-up IPI (SIPI)			○
5	SMI			○
6	Other SMI			○
7	Interrupt Window			ゲストVMCB設定による(*1)
9	Task switch			○
10	CPUID			○
12	HLT			ゲストVMCB設定による(*1)
13	INVD			○
14	INVLPG			ゲストVMCB設定による(*1)
15	RDPMC			ゲストVMCB設定による(*1)
16	RDTSC			ゲストVMCB設定による(*1)
17	RSM			○
18	VMCALL			○
19	VMCLEAR		○	○
20	VMLAUNCH	○		○
21	VMPTRLD			○
22	VMPTRST			○
23	VMREAD			○
24	VMRESUME	○		○
25	VMWRITE			○
26	VMXOFF			○
27	VMXON			○
28	CR register access			ゲストVMCB設定による(*1)
29	MOV DR			ゲストVMCB設定による(*1)
30	I/O instruction			ゲストVMCB設定による(*1)
31	RDMSR			ゲストVMCB設定による(*1)
32	WRMSR			ゲストVMCB設定による(*1)
33	VM-entry failure due to invalid guest state			○
34	VM-entry failure due to MSR loading			
36	MWAIT			ゲストVMCB設定による(*1)
39	MONITOR			ゲストVMCB設定による(*1)
40	PAUSE			ゲストVMCB設定による(*1)
41	VM-entry failure due to machine check			○
43	TPR below threshold			ゲストVMCB設定による(*1)

(*1) ゲストVMCB中のVM実行制御領域(VM-EXECUTION CONTROL FIELD)等の設定値に依存して、通知条件が可変。

【図１１】


AMD-64(AMD-V) CPUにおける#VMEXIT要因一覧
1371
1372
1373
1374
EXITCODE
Name
判定
VMRUN命令発行起因の#VMEXIT
ゲストVMMに対する#VMEXIT通知条件
0-15 VMEXIT_CR[0-15]_READ ゲストVMCB設定による(*2)
16-31 VMEXIT_CR[0-15]_WRITE ゲストVMCB設定による(*2)
32-47 VMEXIT_DR[0-15]_READ ゲストVMCB設定による(*2)
48-63 VMEXIT_DR[0-15]_WRITE ゲストVMCB設定による(*2)
64-95 VMEXIT_EXCP[0-31] ゲストVMCB設定による(*2)
96 VMEXIT_INTR ゲストVMCB設定による(*2)
97 VMEXIT_NMI ゲストVMCB設定による(*2)
98 VMEXIT_SMI ゲストVMCB設定による(*2)
99 VMEXIT_INIT ゲストVMCB設定による(*2)
100 VMEXIT_VINTR ゲストVMCB設定による(*2)
128 VMEXIT_VMRUN ○ ゲストVMCB設定による(*2)
129 VMEXIT_VMMCALL ゲストVMCB設定による(*2)
130 VMEXIT_VMLOAD ゲストVMCB設定による(*2)
131 VMEXIT_VMSAVE ゲストVMCB設定による(*2)
136 VMEXIT_ICEBP ゲストVMCB設定による(*2)
1024 VMEXIT_NPF ○
-1 VMEXIT_INVALID ○
(*2) ゲストVMCB中のInterception Control Bits等の設定値に依存して、通知条件が可変。
11 ホストVMM保持データ
132
ホスト状態エリア
シャドウVMCB
132
ゲスト状態エリア
137
制御エリア
EXITCODE
Interception Control Bits

【図１５】



(*3) 仮想CPU上で稼動するPAL_VPS_RESUMEプロシジャコード内でvmsw命令を発行してホストVMMへ通知する場合に有効。
(*4) ゲストvdc領域で指定される仮想化例外発生の抑止条件によって、仮想化例外の通知条件が可変。
ゲストvdc領域は、VT-i spec定義の Virtualization Acceleration Control (vac) Fieldの情報を保持するデータ構造。詳細は VT-i spec参照。

IPF(VT-i) CPUにおける仮想化例外要因一覧

フロントページの続き

(72)発明者　對馬　雄次
東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

審査官　吉田　美彦

(56)参考文献　特開２００７－０３５０４５（ＪＰ，Ａ）
特表２００７－５１３４０５（ＪＰ，Ａ）
特開２００７－１８３９５１（ＪＰ，Ａ）
特開２００６－０９９３３２（ＪＰ，Ａ）
米国特許出願公開第２００７／０００６２２８（ＵＳ，Ａ１）
伊藤　宏通，ＣＰＵの仮想化技術をＸｅｎで活用する　ＶＭＸドメイン徹底研究，Ｌｉｎｕｘ　ＷＯＲＬＤ　第５巻　第４号，日本，（株）ＩＤＧジャパン，２００６年　４月　１日，第5巻，Ｐ.136～141

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４６