(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5785564号
(P5785564)

(45)発行日　平成27年9月30日(2015.9.30)　(24)登録日　平成27年7月31日(2015.7.31)

(51)Int.Cl.　F I
G06F 9/46 (2006.01)　G06F 9/46 350
G06F 9/445 (2006.01)　G06F 9/06 610A

請求項の数 21　(全 17 頁)

(21)出願番号　特願2012-552120 (P2012-552120)
(86)(22)出願日　平成23年2月4日(2011.2.4)
(65)公表番号　特表2013-519167 (P2013-519167A)
(43)公表日　平成25年5月23日(2013.5.23)
(86)国際出願番号　PCT/US2011/023798
(87)国際公開番号　WO2011/097518
(87)国際公開日　平成23年8月11日(2011.8.11)
審査請求日　平成26年2月4日(2014.2.4)
(31)優先権主張番号　12/699,901
(32)優先日　平成22年2月4日(2010.2.4)
(33)優先権主張国　米国(US)

(73)特許権者　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　９８０５２　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　100101373
弁理士　竹内　茂雄
(74)代理人　100118902
弁理士　山本　修

最終頁に続く

(54)【発明の名称】拡張可能なアプリケーション仮想化サブシステム

(57)【特許請求の範囲】
【請求項１】
アプリケーションの仮想化の準備のためにコンピューターにより実行される方法であって、
コンピューティング装置によって、仮想化されるべきアプリケーションのインストールの開始を検出するステップと、
前記仮想化されるべきアプリケーションのインストールの開始の検出に応答して、前記コンピューティング装置上で実行中のコア仮想化システムによって、仮想化サブシステムの記憶装置から複数の登録された仮想化サブシステムを識別するステップと、
前記複数の登録された仮想化サブシステムの識別に基づいて、前記コンピューティング装置上で実行中の前記コア仮想化システムによって、前記インストールを監視するための仮想化サブシステムをプログラム的に決定するステップであって、
前記複数の登録された仮想化サブシステムから仮想化サブシステムを選択するステップと、
前記選択された仮想化サブシステムに前記インストールの開始を報知するステップと、
前記選択された仮想化サブシステムが前記インストールを監視すべきであるか否かを確認するステップと、
を含む、プログラム的に決定するステップと、
前記決定された仮想化サブシステムによって、前記インストールを監視するステップで

あって、前記仮想化されるべきアプリケーションに関連する構成情報を収集するステップを含み、前記構成情報は、前記決定された仮想化サブシステムに特有である少なくともいくつかの構成情報を含む、監視するステップと、
前記コンピューティング装置によって、前記収集された構成情報を前記仮想化されるべきアプリケーションに関連するアプリケーションパッケージ内に格納するステップと、
を含む方法。
【請求項２】
コア仮想化エンジンをスタートするステップと、
前記コア仮想化システムを前記インストールのプロセスに注入するステップと、
前記インストールによってなされたＡＰＩコールを監視するステップと、
を更に含む請求項１に記載の方法。
【請求項３】
前記複数の登録された仮想化サブシステムを識別するステップは、オペレーティングシステムレジストリ内に登録された仮想化サブシステムに関する格納された情報をロードするステップを含む請求項１に記載の方法。
【請求項４】
前記コア仮想化システムを変更することなしに、前記コア仮想化システムが新しい仮想化サブシステムによって拡張されることを可能とする請求項１に記載の方法。
【請求項５】
前記収集された構成情報を格納するステップは、前記収集された構成情報をＸＭＬ（extensible markup language）マニフェストに格納するステップを含む請求項１に記載の方法。
【請求項６】
前記仮想化サブシステムをプログラム的に決定するステップは、前記登録された仮想化サブシステムの一つが前記インストールを監視することを示すまで、前記複数の登録された仮想化サブシステムから仮想化サブシステムを順次選択することを含む請求項１に記載の方法。
【請求項７】
前記選択された仮想化サブシステムに報知するステップは、前記選択された仮想化サブシステムをロードする関数を呼び出すステップと、前記選択されたサブシステムに前記インストールに関する情報を渡すステップと、を含む請求項１に記載の方法。
【請求項８】
前記構成情報を収集するステップは、Ｗｅｂサーバー仮想化サブシステムによって、Ｗｅｂサーバーの構成情報を収集するステップを含む請求項１に記載の方法。
【請求項９】
前記収集された構成情報を格納するステップは、前記決定された仮想化サブシステムからオブジェクトモデルインターフェースへのコールを受け取るステップを含む請求項１に記載の方法。
【請求項１０】
前記仮想化されたアプリケーションを後で実行するために使用される前記仮想化サブシステムが示すサブシステム識別子を、前記アプリケーションパッケージに格納するステップを更に含む請求項１に記載の方法。
【請求項１１】
アプリケーションレベルの仮想化のためのコンピューターシステムであって、
ソフトウエア命令を実行するように構成されたプロセッサーおよびメモリーと、
少なくとも一つの仮想化サブシステムから登録要求を受け取るように構成されたサブシステムカタログコンポーネントであって、登録によって、コア仮想化システムが前記仮想化サブシステムを使用して、前記コア仮想化システムの命令を更新することなく、アプリケーションを仮想化することを可能とするサブシステムカタログコンポーネントと、
コア仮想化サブシステムであって、

仮想化のために用意されているアプリケーションのインストールの開始を検出することに応答して、前記サブシステムカタログコンポーネントから複数の登録された仮想化サブシステムを識別し、

前記複数の登録された仮想化サブシステムの識別に基づいて、選択プロセスであって、前記複数の登録された仮想化サブシステムから仮想化サブシステムを選択するステップと、前記選択された仮想化サブシステムに前記インストールの開始を報知するステップと、前記仮想化のために用意されているアプリケーションを監視するために前記選択された仮想化サブシステムを呼び出すステップと、を含む選択プロセスによって、登録された仮想化サブシステムをプログラム的に選択する、

ように構成されたコア仮想化サブシステムと、

前記仮想化のために用意されているアプリケーションに関係する構成データを格納するためのパッケージを作成するように構成されたパッケージ作成コンポーネントと、

を含むコンピューターシステム。

【請求項１２】

前記サブシステムカタログコンポーネントは、前記登録された仮想化サブシステムの一つまたは複数を記述する情報をオペレーティングシステムの構成データストアに格納する請求項１１に記載のコンピューターシステム。

【請求項１３】

前記コア仮想化サブシステムは、前記複数の登録された仮想化サブシステムから仮想化サブシステムを順次選択して、特定の登録された仮想化サブシステムが前記仮想化のために用意されているアプリケーションを扱うべきであるか否かを決定するように更に構成された請求項１１に記載のコンピューターシステム。

【請求項１４】

前記呼び出された仮想化サブシステムは、前記アプリケーションのインストール中になされたサブシステム特有の設定の変化を監視するように構成される請求項１１に記載のシステム。

【請求項１５】

前記コア仮想化サブシステムは、前記選択された仮想化サブシステムがシステム構成情報のスタートアップとシャットダウンのスナップショットをとることを可能とするように更に構成される請求項１１に記載のコンピューターシステム。

【請求項１６】

前記パッケージ作成コンポーネントは、サブシステム特有の構成情報を格納するためのＡＰＩを提供するように更に構成される請求項１１に記載のシステム。

【請求項１７】

仮想化されたアプリケーションをクライアントコンピューターシステムに配置するように構成されたアプリケーション配置コンポーネントを更に含み、前記アプリケーション配置コンポーネントは、前記仮想化されたアプリケーションに関連する前記仮想化サブシステムを呼び出すことによって、前記クライアントコンピューターシステム上のアプリケーション特有の設定を行うように更に構成される請求項１１に記載のシステム。

【請求項１８】

仮想化されたアプリケーションをクライアントコンピューティングシステムのホストオペレーティングシステムから隔離し、登録された仮想化サブシステムを呼び出して仮想化されたアプリケーションのサブシステム特有の実行時処理を提供するように構成されたアプリケーション実行環境を更に含む請求項１１に記載のシステム。

【請求項１９】

仮想化されたアプリケーションを実行するための動作を実施するようにコンピューターシステムを制御するための命令が格納されたコンピューター読み取り可能メモリーであって、前記動作は、

前記仮想化されたアプリケーションをクライアントコンピューターシステム上で実行する要求を指示するアプリケーション実行指示を受け取るステップと、

前記アプリケーション実行指示が受け取られた前記仮想化されたアプリケーションに関連するアプリケーションパッケージをロードするステップと、
前記仮想化されたアプリケーションを実行するための前記アプリケーションパッケージに関連する仮想化サブシステムを識別するステップと、
前記識別された仮想化サブシステムに前記仮想化されたアプリケーションを実行する前記要求を報知するステップと、
実行されるべき前記仮想化されたアプリケーションに関連するサブシステム特有の情報を、前記識別されたサブシステムによって前記アプリケーションパッケージから取得するステップと、
前記取得されたサブシステム特有の情報を用いて、前記識別された仮想化サブシステムによって前記仮想化されたアプリケーションを実行するステップと、
を含むコンピューター読み取り可能メモリー。
【請求項２０】
前記サブシステム特有の情報を取得するステップは、前記アプリケーションパッケージへのおよび前記アプリケーションパッケージからの情報のシリアル化および逆シリアル化のための汎用インターフェースを提供するステップと、前記識別された仮想化サブシステムによって前記アプリケーションパッケージに以前格納された設定を取り出すコールを、前記識別された仮想化サブシステムから前記汎用インターフェース上で受け取るステップと、を含む請求項１９に記載のコンピューター読み取り可能メモリー。
【請求項２１】
以前に仮想化されたアプリケーションは、
仮想化されるべきアプリケーションのインストールの開始を検出するステップと、
前記仮想化されるべきアプリケーションのインストールの開始の検出に応答して、仮想化サブシステムの記憶装置から複数の登録された仮想化サブシステムを識別するステップと、
前記複数の登録された仮想化サブシステムの識別に基づいて、前記インストールを監視するための仮想化サブシステムをプログラム的に決定するステップであって、
前記複数の登録された仮想化サブシステムから仮想化サブシステムを選択するステップと、
前記選択された仮想化サブシステムに前記インストールの開始を報知するステップと、
前記選択された仮想化サブシステムが前記インストールを監視すべきであるか否かを確認するステップと、
を含む、プログラム的に決定するステップと、
前記決定された仮想化サブシステムによって、前記インストールを監視するステップと、
を含むプロセスによって仮想化されている請求項１９に記載のコンピューター読み取り可能メモリー。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、アプリケーション仮想化技術に関する。
【背景技術】
【０００２】
仮想化は、物理ハードウエアによって仮想マシーンを実行し、その後仮想マシーン上でオペレーティングシステムとアプリケーションを動作させることを指している。仮想マシーンはハードウエアの機能の最小の共通点を表しても良く、又はオペレーティングシステムとアプリケーションを用意するのが容易な良く知られた構成を表しても良い。多数のデータセンターがメンテナンスサイクルのためにリソースの要求が増加するのに応じて、物理サーバーの負荷をバランスさせるために仮想マシーンを新しい物理ハードウエアに容易

に移動させることが可能であるように仮想化を使用している。仮想化は多くの状況で有用であるが、多くの仮想マシーンが（例えば、中央処理装置（ＣＰＵ）、メモリー、およびネットワークインターフェースカード（ＮＩＣ）等の同一のリソースを取り合うために生じる限界を課す可能性も有る。

【０００３】

アプリケーションの仮想化は、仮想マシーンがＯＳを基礎となるハードウエアから隔離するのと似た方法で、アプリケーションを基礎となるＯＳから隔離する単一アプリケーションレベルでの仮想環境を提供する。例えば、オペレーティングシステムは本来のものとして幾つかのアプリケーションを動作させる一方、他のアプリケーションを動作させる仮想環境を提供することができる。これにより、例えばオペレーティングシステムが異なるオペレーティングシステムのために設計されたアプリケーションを動作させることが可能になり得る。アプリケーションの仮想化は、ユーザーにとってホストオペレーティングシステム内で本来のものとして動作しているアプリケーションと仮想環境内で動作しているアプリケーションとの区別を曖昧にする。例えば、両方の種類のアプリケーションがタスクバー又はオペレーティングシステムシェルによって提供されたメニュー内に並ぶことも可能である。マイクロソフト（登録商標）のアプリケーション仮想化（ＡＰＰ－Ｖ）は、例えば、アプリケーションを、インストールされず他のアプリケーションと競合しない中央管理された仮想サービスに変換する。物理環境において、全てのアプリケーションはメモリーの割り当て、装置ドライバー、およびもっと多くのものを含む種々のサービスをそのオペレーティングシステム（ＯＳ）に依存している。アプリケーションとそのＯＳとの間の非互換性はサーバーの仮想化又はプレゼンテーションの仮想化のどちらかによって扱うことができるが、一つのＯＳの同一のインスタンスにインストールされた２つのアプリケーションの間の非互換性はアプリケーションの仮想化によって解決される。

【０００４】

アプリケーション仮想化製品の開発者は、しばしば製品内のアプリケーション仮想化サブシステムを拡張する。例えば、開発者は、アプリケーションの隔離性を高めるために以前に仮想化されていないオペレーティングシステムの一部分を仮想化しようと思うかも知れない。マイクロソフトのウインドウズ（登録商標）システムを用いた一つの例は、仮想化ＣＯＭ＋のサポートを追加するものである。ＣＯＭ＋はマイクロソフトコンポーネントオブジェクトモデル（Component Object Model）（ＣＯＭ）およびマイクロソフトトランザクションサーバー（ＭＴＳ）を発展させたものである。既存のアプリケーション仮想化ソリューションにおいて、仮想化サブシステムは製品と強固に結合している。既存の製品において、このような結合は、ＣＯＭ＋のような新しい拡張ポイントへのサービスを追加することは製品のかなりの量の再加工を必要とすることを意味している。更に、サブシステムを実装することは、かなりの専門的知識と、（単にサブシステムに特定した知識ではなく）オペレーティングシステム内部構成の理解を必要とする。

【発明の概要】

【０００５】

拡張可能な仮想化システムが本明細書に記述されており、それはオブジェクトモデルを提供し、製品自体の再加工無しに仮想化製品を拡張するために新しい仮想化サブシステムを追加することを可能にする方法で仮想アプリケーションのライフサイクルを管理するものである。アプリケーションを仮想化することは通常以下の３つのステップを含み、それらはメタデータ抽出、メタデータ格納および再構成、並びに要求のランタイム管理である。拡張可能な仮想化システムは、アプリケーション準備セッションの通知を受け取り、仮想化サブシステムがアプリケーションがクライアント上で動作するのに使用する各サブシステムに特有の構成情報（configuration information）を収集するためにセッションを監視することを可能にする。各サブシステムは収集した情報を、拡張可能な仮想化システムに提供し、拡張可能な仮想化システムは、仮想化されるべきアプリケーションが配置されるまで収集された情報を格納する。アプリケーションが配置されるとき、システムは同一の仮想化サブシステムを呼び出し、サブシステムに格納された情報を提供する。こうし

て拡張可能な仮想化システムは、そのシステムを実施する仮想化製品がその製品に対してより少ない影響でより容易に拡張されることを可能にする多くの種類の仮想化サブシステムにとって、有用な汎用モデルを提供する。

【０００６】

この概要は以下の詳細な記述において更に記述された概念の一つの選択を単純化した形式で導入するために提供されたものである。この概要は請求項に記載された主題の主要な特徴又は必須な特徴を特定することを意図したものではなく、特許請求の主題の範囲を限定するために用いられることを意図したものでもない。

【図面の簡単な説明】

【０００７】

【図１】一実施形態における拡張可能な仮想化システムのコンポーネントを示すブロック図である。

【図２】一実施形態における拡張可能な仮想化システムのアプリケーションを監視する処理を示すフロー図である。

【図３】一実施形態における仮想化されたアプリケーションを配置するための拡張可能な仮想化システムの処理を示すフロー図である。

【図４】一実施形態における拡張可能な仮想化システムの動作環境および実装コンポーネントを示すブロック図である。

【発明を実施するための形態】

【０００８】

拡張可能な仮想化システムが本明細書に記述されており、そのシステムはオブジェクトモデルを提供し、製品自体の再加工無しに仮想化製品を拡張するために新しい仮想化サブシステムが追加されることを可能にする方法で仮想アプリケーションのライフサイクルを管理する。アプリケーションを仮想化することは通常３つのステップを含み、それらはメタデータ抽出、メタデータの格納および再構成、並びに要求のランタイム管理である。メタデータの抽出は、アプリケーションが仮想化のための準備ができた時点を知り、構成情報を抽出するためにアプリケーションを監視するプロセスである。構成情報は、アプリケーションがそれ自身をオペレーティングシステム又は他のアプリケーションに結びつけるどのような方法も含むことができる。拡張可能な仮想化システムは、アプリケーション準備セッションの通知を受け取り、仮想化サブシステムが、アプリケーションがクライアント上で動作するために使用する各サブシステムに固有の構成情報を収集するためにセッションを監視することを可能にする。各サブシステムは、収集された情報を拡張可能な仮想化システムに提供し、拡張可能な仮想化システムは、仮想化されるべきアプリケーションが配置されるまで、収集された情報を格納する。アプリケーションが配置されるとき、システムは同じ仮想化サブシステムを呼び出し、格納された情報をサブシステムに提供する。サブシステムはこの情報とクライアントからの情報を使用してメタデータを再構成し、この知識からアプリケーションを仮想化するためにサブシステム特有のステップを実行する方法を知ることができる。

【０００９】

例えば、もし管理者がマイクロソフトＩＩＳ（Internet Information Server）Ｗｅｂアプリケーションを仮想化しようと思ったならば、ＩＩＳは「アプリケーションプール」識別子を使用してＷｅｂアプリケーションのインスタンスを管理する。この場合、拡張可能な仮想化システムの監視部が、登録された仮想化サブシステムを見つける。この場合、システムはＩＩＳ仮想化サブシステムを新しいプロセスに注入する。プロセスが作成されたとき、拡張可能な仮想化システムは通知を受け取り、サブシステム特有の方法で、ＩＩＳがＷｅｂアプリケーションを登録していることを検出する。これに代えて、又はこれに加えて、サブシステムはアプリケーションのインストール前と後に構成情報のスナップショットをとり、その差分をアプリケーションメタデータとして格納しても良い。ＩＩＳ仮想化サブシステムはＷｅｂアプリケーションプールの名称等のＷｅｂアプリケーションを仮想化するのに必要とされる情報を抽出し、他のクライアント上で動作するために（例え

ば、アプリケーションプール名称が既に使用されていないことを確認することなどによって）必要に応じて要求を変更し、要求を継続させる。Ｗｅｂアプリケーションが配置されるとき、拡張可能な仮想化システムは、この仮想アプリケーションがＩＩＳ仮想化サブシステムに依存していることを発見し、そのサブシステムをロードし、その後サブシステムにＷｅｂアプリケーションが作成されていること通知する。拡張可能な仮想化システムは、ＩＩＳ仮想化サブシステムにサブシステムが監視中にＷｅｂアプリケーションについて集めた情報を提供する。実行時にＩＩＳがＷｅｂアプリケーションプールをその特定の名称で作成するとき、ＩＩＳ仮想化サブシステムは特定の仮想アプリケーションを発見して作成するために充分な知識を得ているであろう。同一のステップは、他の様々な種類のアプリケーションおよびサブシステムに仮想化を提供するために使用することができる。こうして拡張可能な仮想化システムは、そのシステムを実施する仮想化製品がその製品に対してより少ない影響でより容易に拡張されることを可能にする多くの種類の仮想化サブシステムにとって、有用な汎用モデルを提供する。更に、このシステムを用いることにより、仮想化サブシステムの作者は、サブシステム特有の動作により集中することができ、サブシステムの実装は、拡張可能な仮想化システムによって扱われるオペレーティングシステム特有の知識のうち、必要となる知識はより少なくなる。

【００１０】

アプリケーション仮想化サブシステムの主な責務は、監視、視覚化、登録、および実行時（ランタイム）仮想化にある。これらの各々についてここで個別に説明する。

【００１１】

監視はインストールプロセスを注視してコンピューターシステムに対して加えられた変更を検知することを含む。幾つかの実施形態において、シーケンサーと呼ばれるアプリケーションがアプリケーション仮想化の監視段階の責任を負う。幾つかのサブシステムがアプリケーションプログラミングインターフェース（ＡＰＩｓ）をフックしてインストールを監視し、他のサブシステムが単にインストール前後のコンピューターシステムの状態を比較するようにしても良い。前者の一例は、仮想サービスサブシステムがCreateService（）ＡＰＩをフックして新しいオペレーティングシステムサービスがインストーラーによって追加された時点を検出することである。一方、前の例で記述されたＩＩＳサブシステムは、インストール前後のＩＩＳ構成を比較することによってそれ自身が使用する情報を収集する。

【００１２】

視覚化は仮想化のためのアプリケーションを用意している管理者にアプリケーションによって生じた変化の視覚的な表示を示すことを含む。シーケンサーは監視の間に検出された変化を一連のタブとして表示する。拡張可能な仮想化システムは、視覚化を実行する責務を各サブシステムに与える。各サブシステムはそのサブシステムにとって適当な視覚化のためのユーザーインターフェースを提供する。これによってシーケンサーが視覚化段階の間に各サブシステムの明示的な知識を持っている必要が無くなる。サブシステムの視覚化インターフェースが拡張可能な仮想化システムの視覚化アプリケーションによってそれらの為に作成されたウインドウ内でそれらの結果を表示する責任を負う。

【００１３】

登録は仮想アプリケーションを実行するためのクライアントコンピューターシステムを用意することを含む。仮想アプリケーションはクライアント上にインストールされないが、シームレスなユーザーエクスペリエンス（例えば、アプリケーションがスタートメニューに表示されるように）を提供するために情報がクライアントに公開される。拡張可能な仮想化システムは、監視の間にサブシステムによって収集された情報を、クライアント上の適当な構成位置に公開する。例えば、仮想化されたＩＩＳＷｅｂアプリケーションは、クライアントのＩＩＳ構成を変更して、クライアントにアプリケーションが存在しているという知識を提供する（ユーザーがアプリケーションをスタートできるように）ようにしても良い。

【００１４】

登録に深く関係するサブシステムの他の責務は、設定（configuration）についてである。登録は一度行われる。しかしながら、設定はサブシステムがそのコンポーネントを登録した後に何回も行われ得る。登録の一例は、ＩＩＳＷｅｂサイト、アプリケーション、および本明細書に記述されたアプリケーションプールを作成することである。設定項目の一例は、データベース接続文字列である。アプリケーションのためのバックエンドデータベースが移動されたときには、アプリケーションが登録された後にこの値は変更されなければならない。サブシステムは、例えばサブシステムの特定の知識を適用することを必要とする設定（例えば、特定のＡＰＩを呼び出すことによって設定されねばならないかもしれない）などの、サブシステムに特有のあらゆる設定値を適用する責任を持つ。

【００１５】

実行時（ランタイム）仮想化は仮想化されたアプリケーションの実行時の間に機能をフックし、機能の動作を変更してアプリケーションがあたかもクライアント上にローカルにインストールされたかのようにそのリソースにアクセスすることを可能にすることを指している。実行時仮想化はまた、或るシステムプロセス（即ち、仮想アプリケーションのパッケージの一部分でないプロセス）が仮想化されるべきであるか否かについての決定を行うことを指している。例えば、ＩＩＳサブシステムは、ＩＩＳ作業プロセスがコマンドラインに渡されたアプリケーションプール名に基づいて仮想化されるべきであるか否かを決定する。

【００１６】

図１は、一実施形態における拡張可能な仮想化システムの構成要素を示すブロック図である。システム１００は、サブシステムカタログコンポーネント１１０、サブシステム監視コンポーネント１２０、パッケージ作成コンポーネント１３０、アプリケーションパッケージストア１４０、アプリケーション配置コンポーネント１５０、およびアプリケーション実行環境１６０を含む。ここで、これらのコンポーネントの各々について更に記述する。

【００１７】

サブシステムカタログコンポーネント１１０は、少なくとも一つの仮想化サブシステムから登録要求を受取り、登録によってコア仮想化システムが仮想化サブシステムを使用するアプリケーションをコア仮想化システムの命令を更新すること無しに仮想化することが可能になる。例えば、クライアントコンピューターにインストールされたＣＯＭ＋仮想化サブシステムがそれ自身をコア仮想化システムに登録しても良い。ユーザーがＣＯＭ＋アプリケーションを仮想化することを要求したとき、仮想化システムがＣＯＭ＋仮想化サブシステムを呼び出し、それによって設定の変化を検出するためのサブシステム特有のＣＯＭ＋アプリケーションの監視が提供される。更に、仮想化されたアプリケーションがクライアント上で動作するとき、コア仮想化システムが再び仮想化サブシステムを呼び出してサブシステム特有のアプリケーションのランタイム処理が行われる。

【００１８】

サブシステム監視コンポーネント１２０は、仮想化のための準備が行われているアプリケーションを監視するために、登録された仮想化サブシステムを呼び出す。サブシステム監視コンポーネント１２０は、登録されたサブシステムを通して各サブシステムにサブシステムが現在のアプリケーションを処理することに興味を持っているか否かを問い合わせることを反復するようになっていても良い。コンポーネント１２０は、アプリケーションプロセスに興味を示した一つ以上のサブシステムを呼び出し、サブシステムがアプリケーションによって行われたサブシステム特有の設定の変化を監視することができる。サブシステム監視コンポーネント１２０は、サブシステムにアプリケーションのライフタイムについて報知して、個々のサブシステムがサブシステムに関係したスタートアップおよびシャットダウンのタスクを実行できるようにしても良い。例えば、幾つかのサブシステムがアプリケーションの実行前後の構成データのスナップショットを撮ることによって監視するようにしても良い。

【００１９】

パッケージ作成コンポーネント１３０は、仮想化のための用意が行われているアプリケーションに関係した構成データを格納するためのパッケージを作成する。パッケージは、圧縮、認証、暗号化、又はパッケージを小さくするかパッケージの作者のセキュリティもしくは確実性を提供する等の他の処理を含みうる種々のコンテナファイルフォーマットを含んでも良い。例えば、パッケージはＺＩＰ、ＣＡＢ、又は多くのファイルおよび設定を単一のファイルに格納するのに適した他のアーカイブフォーマットとすることができる。パッケージ作成コンポーネント１３０は、新しいアプリケーションが仮想化のために用意されているときにパッケージを作成し、サブシステムがサブシステム特有の構成情報をパッケージ内に格納するためのＡＰＩを提供する。例えば、コンポーネント１３０は、拡張可能な仮想化システムがサブシステムを呼び出したときにデータを格納するためのインターフェースへのポインタを渡しても良い。

【００２０】

アプリケーションパッケージストア１４０は、仮想化のために用意されているアプリケーションの監視と仮想化されたアプリケーションの１以上のクライアントコンピューターシステムへの配置との間に、アプリケーションパッケージを格納する。アプリケーションパッケージストア１４０は、ファイルシステム、ネットワークベースストレージ、クラウドベース格納サービス、データベース、等の種々の記憶媒体を含むことができる。

【００２１】

アプリケーション配置コンポーネント１５０は、クライアントコンピューターシステムが仮想化されたアプリケーションを呼び出すことができるようにアプリケーションパッケージをクライアントコンピューターシステムに配置する。コンポーネント１５０は、クライアントシステム上でファイルタイプ関連付けの追加、アプリケーションパッケージへのリンクの追加、オペレーティングシステムサービス設定の実行、およびＷｅｂサーバー構成情報の追加などのアプリケーション特有の設定を実行するようになっていても良い。アプリケーション配置コンポーネント１５０がアプリケーションパッケージに関連したサブシステムを呼び出して仮想化されたアプリケーションがクライアントコンピューターシステム上で動作するように用意するためのサブシステム特有の登録タスクを実行するようになっていても良い。

【００２２】

アプリケーション実行環境１６０は、仮想化されたアプリケーションとクライアントコンピューターシステムのホストオペレーティングシステムとの間の、ある間接レベル（level of indirection）を提供する。ラッパー（wrapper）は非常に薄く（thin）、アプリケーションがホストオペレーティングシステム上で動作するように設計されているときのように、アプリケーションに殆ど本来の状態で動作させることが可能である。これに代えて、又はこれに加えて、ラッパーはＡＰＩを提供し、他のオペレーティングシステム又はオペレーティングシステムの他のバージョンの為に設計されたアプリケーションが予期するような制約を満足するようにしても良い。このようにしてアプリケーション実行環境１６０は、仮想アプリケーションに、そのためにアプリケーションがホストオペレーティングシステムの利用可能なリソースを使用するように設計された環境を提供する。アプリケーション実行環境１６０はまた、適当なサブシステム（又は複数のサブシステム）を呼び出して仮想化されたアプリケーションの実行時のサブシステム特有の処理を提供する。

【００２３】

拡張可能な仮想化システムが実装されるコンピューティング装置は、中央処理装置、メモリー、入力装置（例えば、キーボードおよびポインティング装置）、出力装置（例えば、表示装置）、および記憶装置（例えばディスクドライバ又は他の不揮発性記憶媒体）を含む。メモリーおよび記憶装置は、システムを実現又は可能化するコンピューターにより実行可能な命令（例えば、ソフトウエア）でエンコードされていても良いコンピューターにより読み出し可能な記憶媒体である。加えて、データ構造およびメッセージ構造は通信リンク上の信号等のデータ伝送媒体を介して格納又は伝送することができる。インターネット、ローカルエリアネットワーク、広域ネットワーク、ポイント間ダイヤルアップアッ

プ接続、携帯電話ネットワーク等の種々の通信リンクが使用可能である。

【００２４】

システムの実施形態はパーソナルコンピューター、サーバーコンピューター、ハンドヘルド又はラップトップ装置、マルチプロセッサシステム、マイクロプロセッサベースシステム、プログラム可能な個人用電化製品、デジタルカメラ、ネットワークＰＣ、ミニコンピューター、メインフレームフレームコンピューター、上記のシステム又は装置の何れかを含む分散コンピューティング環境等を含む種々の動作環境で実装することが可能である。コンピューターシステムは携帯電話、パーソナルデジタルアシスタント、スマートフォン、パーソナルコンピューター、プログラム可能な個人用電化製品、デジタルカメラ、等であっても良い。

【００２５】

システムはプログラムモジュール等の、一以上のコンピューター又は他の装置によって実行されるコンピューターにより実行可能な命令の一般的なコンテキストで記述することができる。通常プログラムモジュールは特定のタスクを実行するか特定の抽象的なデータタイプを実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを含む。通常、プログラムモジュールの機能は種々の実施形態において必要に応じて組み合わされるか又は分散されていても良い。

【００２６】

図２は、一実施形態における拡張可能な仮想化システムのアプリケーションを監視する処理を示すフロー図である。処理はブロック２１０において開始し、システムは、新しいプロセスの作成を検出してコア仮想化システムをプロセスに注入する。例えば、もし管理者が監視を起動させてデスクトップアプリケーションを開始すると、システムがデスクトップアプリケーションの開始を検出し、コア仮想化エンジンをスタートさせ、コア仮想化システムをプロセスに注入し、例えば、システムがＡＰＩコールおよび他のプロセスの状態を監視できるようになる。続くブロック２２０において、システムは仮想化サブシステムのリストから登録された仮想化サブシステムをロードする。例えば、システムはシステムに登録された仮想化サブシステムについての情報をオペレーティングシステムレジストリに格納するようにしても良い。この方法で、システムは単にシステムの設定を行うのみでシステムを変更したり再構成すること無く（例えば、新しいサブシステムが必要とされる度に新しいコアリリースを出荷すること無く）新しいシステムと共に動作することができる。

【００２７】

続くブロック２３０において、システムは検出されたプロセスと関連した仮想化されたアプリケーションのための構成情報を格納するためのアプリケーションパッケージを作成する。例えば、パッケージは設定の設定値のためのＸＭＬ（extensible markup language）又は他のコンテナを含んでも良い。続くブロック２４０において、システムは第１の登録された仮想化サブシステムを選択する。その後の繰り返しにおいて、システムは次の登録されたサブシステムを選択する。続くブロック２５０において、システムは選択された仮想化サブシステムに検出されたプロセスの作成を報知し、仮想化サブシステムがプロセスを仮想化することを望んでいるか否かを判別する。例えば、システムは作成関数（creation function）を呼び出して仮想化サブシステムをロードし、次にサブシステムにプロセスについての情報を渡す。サブシステムは、サブシステムがプロセスを仮想化することに興味を持っているか否かを示す、作成関数からの戻り値を提供するようにしても良い。異なるタイプのプロセスが異なるサブシステムによって仮想化され、こうして、典型的には、単一のサブシステムは、検出されたプロセスについての責任を示す。

【００２８】

続く判断ブロック２６０において、もしシステムが、選択されたサブシステムが検出されたプロセスを扱うことができると判断したときには、システムはブロック２７０において動作を続ける。そうでなければシステムはブロック２４０にループして次の仮想化サブシステムを選択する。プロセス（図示せず）を扱うことができるサブシステムが無ければ

、システムは終了し、プロセスは本来の状態で動作する（即ち、仮想化されない）。続く判断ブロック２７０において、仮想化サブシステムは、プロセスが動作している間に検出されたプロセスについてのサブシステム特有の構成情報を収集する。例えば、サブシステムは、プロセスによって変更されたファイルと登録キー、同様に、マイクロソフトアクティブディレクトリ（MICROSOFT ACTIVE DIRECTORY）又はＩＩＳメタベース（metabase）の変化等の他の設定の変化を検出しても良い。続くブロック２８０において、システムは収集された構成情報を、作成されたパッケージ内に格納する。サブシステムがサブシステム特有の情報を、作成されたパッケージに、又は作成されたパッケージからシリアル化（serialize）および逆シリアル化（deserialize）することができるように、システムは、サブシステムにインターフェースを供給しても良い。このようにして、パッケージはコア仮想化システムに共通な情報とサブシステム特有の情報とを一つの場所に格納する。

【００２９】

図３は、一実施形態における仮想化されたアプリケーションを配置するための拡張可能な仮想化システムの処理を示したフロー図である。処理はブロック３１０において開始し、システムは仮想化されたアプリケーションのインスタンスをクライアントコンピューターシステム上に登録する。例えば、拡張可能な仮想化システムは、クライアントのユーザーが仮想化されたアプリケーションを開始することを可能にするリンク、ファイルタイプ関連付け又は他のクライアントエントリーを作成しても良い。続くブロック３２０において、システムは、仮想化されたアプリケーションを開始するための要求を表示するアプリケーション実行指示を受け取る。デスクトップアプリケーションの場合には、その指示はオペレーティングシステムシェルを使用しているユーザーから発せられる。サーバーアプリケーションの場合には、その指示は（例えば、サーバーによりホストされたＷｅｂページをアクセスするための）ネットワーク又は他のソースを介して受け取った要求から発せられる。

【００３０】

続くブロック３３０において、システムは、受け取ったアプリケーション実行指示に関連したアプリケーションパッケージをロードする。例えば、その指示は、システムが正しい仮想化されたアプリケーションを見つけて開始することができるようにするためにアプリケーションＧＵＩＤ又は他の識別子を特定する、呼び出されたリンクを含んでいても良い。システムはまた、ロードされたパッケージからコア仮想化設定の設定値を取り込むようにしても良い。続くブロック３４０において、システムは、アプリケーションパッケージに関連した仮想化サブシステムを識別する。例えば、システムは、プロセスコマンドラインから登録されたサブシステムに問い合わせることによってプロセスが仮想化されるべきか否かを決定するようなっていても良い。クライアントは、特にパッケージ名を引用するコマンドライン引数により又は各サブシステムに問い合わせることによってプロセスがパッケージにおいてそれらのパスで仮想化されるべきか否かを動的に決定しても良い。もし仮想化されれば、全てのサブシステムが仮想プロセス内で機能することになる。例えば、システムは、図２のステップ２４０から２６０と同様のステップを辿って登録されたサブシステムを通して動作を繰り返し、サブシステムが、利用できる情報に基いてプロセスを仮想化する選択を示すことができるようにしても良い。

【００３１】

続くブロック３５０において、システムは識別された仮想化サブシステムをスタートさせる。例えば、システムは、サブシステムに関連したバイナリー実行可能モジュールを特定し、モジュールをロードし、サブシステムをスタートさせるエントリーポイントファンクション（entry point function）を実行するようにしても良い。続くブロック３６０において、システムは、識別された仮想化サブシステムにアプリケーションパッケージを実行する指令について報知する。例えば、システムは、サブシステムの開始時にパッケージのデータへのポインタをサブシステムに渡すようにしても良い。サブシステムは、サブシステムが仮想化されたアプリケーションの実行を扱うことを可能にするサブシステム特有のフックを設定するようにしても良い。

【００３２】
続くブロック３７０において、システムは、識別されたサブシステムによる要求の時点でアプリケーションパッケージからサブシステム特有の情報を取り込む。拡張可能な仮想化システムは、アプリケーションパッケージへのおよびアプリケーションパッケージからの情報のシリアル化および逆シリアル化のための汎用インターフェースを提供し、サブシステムがインターフェースを使用して、本明細書に記述された監視処理の間にサブシステムによってパッケージに格納された設定を取り出すようにしても良い。続くブロック３８０において、システムは、識別されたサブシステムと取り込まれたサブシステム特有の情報を使用して仮想化されたアプリケーションを実行する。例えば、システムは、仮想化環境がロードできるようにCreateProcessコール（CreateProcess call）の間に一時停止されていたプロセスに、実行を継続させるようにしても良い。アプリケーションの実行にしたがって、識別された仮想化サブシステムによって導入されたフック又は他の間接指定処理によって、サブシステムが、仮想化によって提供された分離を許容するリダイレクト又は他の処理を必要とするアプリケーションの要求を扱うことが可能になる。ブロック３８０の後、これらのステップは終了する。
【００３３】
図４は、一実施形態における拡張可能な仮想化システムの動作環境および実装コンポーネントを示すブロック図である。幾つかの実施形態において、拡張可能な仮想化システムは、アプリケーションを仮想化するためにオペレーティングシステムドライバー４３０およびユーザーモードコンポーネントを使用する。仮想化ドライバー４３０（例えば、sftplay.sys）は、オペレーティングシステムのカーネルレベルの仮想環境を管理する。ドライバーのプロセスマネージャーコンポーネントは、仮想環境マッピングへのプロセスＩＤを維持する。プロセスが生成されている間、プロセスマネージャーコンポーネントは、自動的に親プロセスの仮想環境に子プロセスを追加する。プロセスマネージャーコンポーネントはまた、ユーザーモード仮想化ライブラリー４２０に仮想プロセスの作成、終了、仮想環境中で動作しているプロセスが無くなった時点について報知する。
【００３４】
ドライバー４３０の役割の一つは、レジストリとファイルシステムの仮想化を実行することである。仮想環境が仮想化ライブラリー４２０を使用して最初に作成されるときに、レジストリとファイルマッピング情報はドライバー４３０にアップロードされる。ドライバー４３０はこの情報を利用してレジストリとファイルシステムＡＰＩの動作を変化させ、仮想アプリケーション４４０にはアプリケーションがクライアントコンピューターシステム上に局所的にインストールされたように見える。
【００３５】
ユーザーモード仮想化ライブラリー４２０（例えばosguard.lib）は、仮想環境とプロセスを管理するＡＰＩを含んでおり、ユーザーモードと仮想化ドライバー４３０との間のインターフェースである。仮想化ライブラリー４２０はまた、種々の拡張可能なサブシステムに亘って使用される仮想ファイルシステムおよび仮想レジストリを含む、それに組みこまれる仮想サブシステムを有している。仮想化ライブラリー４２０は、仮想アプリケーションをパッケージ化するとき、および実行時に使用される。シーケンサー４１０は、アプリケーションインストールプロセスを監視することによって仮想アプリケーションをパッケージ化するときに管理者に使用されるアプリケーションであり、リスナー（図示せず）は仮想アプリケーションを実行時に管理するオペレーティングシステムのサービスである。
【００３６】
仮想化実行時モジュール（例えばsftldr.dll）は、子プロセスを作成するときに仮想化ライブラリー４２０又はそれ自身によって各仮想プロセスに注入されるライブラリーである。迂回路（detours）ライブラリーが仮想アプリケーション４４０のプロセス内の機能をフックするために用いられる。幾つかのコールがレジストリとファイルシステムの仮想化のために仮想化ドライバー４３０にリダイレクトされ、その他はＲＰＣコールを仮想化

ライブラリー４２０に作成してそれらの各々のサブシステムの仮想化を実行する。

【００３７】

その初期化の間、仮想化ライブラリー４２０はレジストリ内にリストされた各サブシステムモジュールを動的にロードし、それらのファクトリー（factory）インターフェースへのポインタを格納する。シーケンサー４１０又はリスナーが仮想環境マネージャーを使用して仮想環境を作成するとき、各サブシステムのインスタンスを作成するためにファクトリーが使用される。サブシステムのインスタンスは仮想環境オブジェクト内に格納され、シーケンサー又はリスナーによって取り込まれうる。

【００３８】

多くのサブシステムはそれらの仮想化を実行するために仮想ファイルシステムおよびレジストリにアクセスする必要がある。例えば、仮想ＣＯＭおよび仮想サービスサブシステムの両方が仮想ＣＯＭオブジェクトを作成し、仮想サービスを開始するためにそれぞれ仮想レジストリーキーにアクセスする必要がある。したがって、幾つかの実施形態では、仮想レジストリおよびファイルシステムサブシステムは汎用化されず、寧ろコアコンポーネントである。サブシステムは、それらの監視および実行時インターフェースを介して渡された仮想環境オブジェクトを通して、これらのコアサブシステムへのアクセスを得ることができる。

【００３９】

監視インターフェースは、それが検出したサブシステムの変化をマニフェスト４５０ファイル内に格納する。このマニフェスト４５０は、表示を行いこの情報の編集を可能にする視覚化インターフェース、およびコンポーネントを登録するためにその情報を用いる登録インターフェースを含む他のインターフェースに渡される。

【００４０】

幾つかの実施形態において、拡張可能な仮想化システムによって格納されるマニフェストはＸＭＬファイルである。ＸＭＬファイルは情報を叙述的なフォーマットで階層的に格納し、仮想化のために使用される種々のコンポーネントおよびサブシステムがそのサブシステムによって必要とされる情報を個々に格納することを可能にする。実行時に、各コンポーネント又はサブシステムは、ＸＭＬファイルからそれ自身の、格納された情報を容易に特定し、抽出することができる。

【００４１】

幾つかの実施形態において、拡張可能な仮想化システムは、仮想化サブシステムと対話するためのオブジェクトモデルを提供する。システムは、それぞれのサブシステムの責務のためにインターフェースを提供し、サブシステムを表すインターフェースは、他のインターフェースをラップ（wrap）する。監視と実行時はそれぞれ２つのインターフェースを使用し、一つのインターフェースは仮想化されたプロセス内で使用され、本明細書に記述された仮想化ライブラリーによって使用されるもう一つのインターフェースと通信する。その結果のインターフェースは以下の通りである。

```
class subsystem
{
public:
   // サブシステム名を返す（Returns the subsystem name）.
   virtual const std::wstring& name（） const =0;
   // サブシステムの視覚化インターフェースを返す（Returns the subsystem's visualization interface）.
   virtual subsystem_visualizer* visualizer（） =0;
   // サブシステムの登録インターフェースを返す（Returns the subsystem's registration interface）.
   virtual subsustem_registrator* registrator（） =0;
   // サブシステムの監視インターファースを返す（Returns the subsystem's monitoring interface）.
```

```
        virtual subsystem_monitor* monitor () =0;
        // 仮想プロセス内で使用されたサブシステムの監視インターフェースを返す（R
eturns the subsystem's monitoring interface used within the virtual process）.
        virtual subsystem_process_monitor* process_monitor () =0;
        // サブシステムの実行時インターフェースを返す（Returns the subsystem's r
untime interface）.
        virtual subsystem_runtime* runtime () =0;
        // 仮想プロセス内で使用されたサブシステムの実行時インターフェースを返す
（Returns the subsystem's runtime interface used within the virtual process）.
        virtual subsystem_process_runtime* process_runtime () =0;
        // サブシステムのＶＥラッパーインターフェースを返す（Returns the subsyst
em's VE wrappper interface）.
        virtual subsystem_ve_wrapper* ve_wrapper () =0;
    };
```

【００４２】

幾つかの実施形態において、拡張可能な仮想化システムは、サブシステムと共に動作するために、仮想化サブシステムからファクトリー（factory)インターフェースを受け取る。幾つかのサブシステムは、仮想環境においてプロセスによって共有された状態を格納する必要がある。そこで、サブシステムオブジェクトのインスタンスは、ファクトリーインターフェースを使用して各仮想環境に作成される。各サブシステムモジュールは、モジュールによって実装されたサブシステム毎にひとつのファクトリーオブジェクトのリストを返すファンクションをエクスポートする。仮想化ライブラリーは、ファクトリーを使用して仮想環境が作成される毎にサブシステムのインスタンスを作成する。構成情報（例えば、レジストリーキー）は、どのサブシステムモジュールが仮想化ライブラリーによってロードされるかについて制御を行う。ファクトリーインターフェースの定義は以下の通りである。

```
    class subsystem_factory
    {
    public:
        // このファクトリが作成するサブシステムのための名前を返す（Returns the n
ame for the subsystem this factory will create）.
        virtual const std: :wstring& name () const =0;
        // サブシステムオブジェクトのインスタンスを作成する（Creates an instance
 of a subsystem object.）
       virtual shared _ptr<subsystem> create () const =0;
    };
```

【００４３】

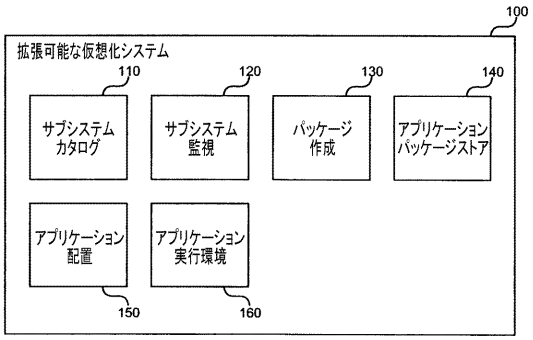
幾つかの実施形態において、拡張可能な仮想化システムは、仮想化サブシステムをテストのために呼び出す。仮想サブシステムを実装することに加えて、これらのモジュールはテストとデバッグのために用いることができる。仮想アプリケーションをパッケージ化することの最も難しい点の一つは、アプリケーションが仮想環境内で正しく動作しないときの診断の問題である。サブシステムモジュールが仮想プロセスに注入されているため、モジュールがＡＰＩをフックして具体的なエラーを探すことによって仮想プロセスを監視することができる。これらのモジュールの他の使用法は、仮想化ライブラリーおよびシーケンサーの機能のテストを行うことである。モジュールは、仮想レジストリおよびファイルシステムが生データを検査することによる監視の間に正しい情報を捕捉していることを確認することができる。それらはまた、それらのインターフェースからエラーを生成することによって、シーケンサーおよびリスナーの耐障害性をテストすることができる。

【００４４】

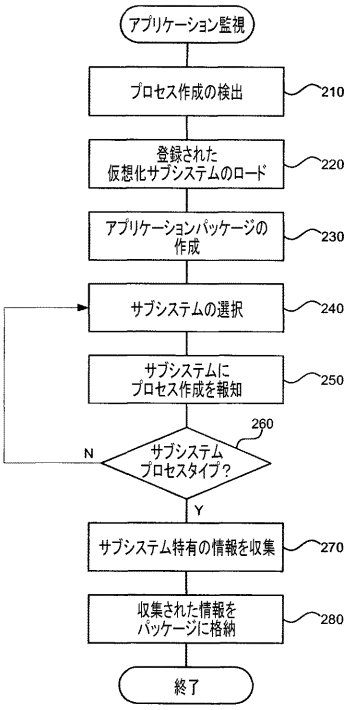
以上の記述から、拡張可能な仮想化システムの特定の実施形態が単なる例示として本明

細書に記述されてきたが、発明の趣旨および範囲から逸脱することなく様々な変形が可能であることが理解されるであろう。従って、本発明は添付の特許請求の記載による以外には限定されない。

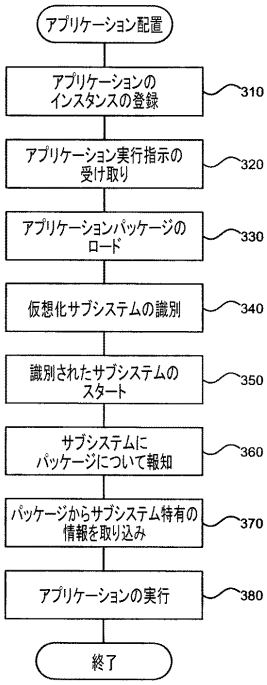
【図１】



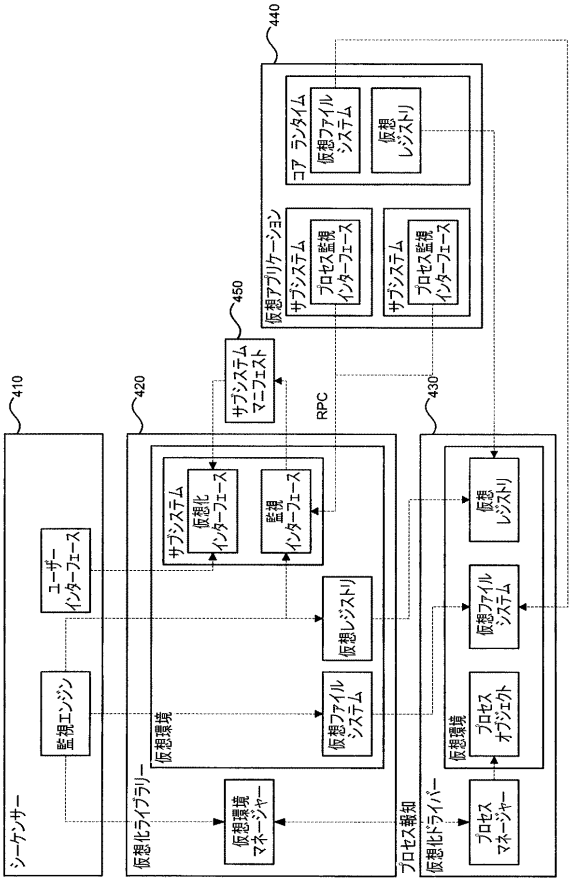
【図２】



【図３】


アプリケーション配置
アプリケーションのインスタンスの登録
310
アプリケーション実行指示の受け取り
320
アプリケーションパッケージのロード
330
仮想化サブシステムの識別
340
識別されたサブシステムのスタート
350
サブシステムにパッケージについて報知
360
パッケージからサブシステム特有の情報を取り込み
370
アプリケーションの実行
380
終了

【図４】


440
コア ランタイム
仮想ファイルシステム
仮想レジストリ
仮想アプリケーション
サブシステム
プロセス監視インターフェース
サブシステム
プロセス監視インターフェース
450
サブシステムマニフェスト
410
420
430
RPC
ユーザーインターフェース
サブシステム仮想化インターフェース
監視インターフェース
仮想レジストリ
仮想ファイルシステム
監視エンジン
仮想環境
仮想環境マネージャー
シーケンサー
仮想化ライブラリー
仮想レジストリ
仮想ファイルシステム
プロセスオブジェクト
仮想環境
プロセスマネージャー
プロセス報知
仮想化ドライバー

フロントページの続き

(74)代理人 100153028
弁理士 上田 忠
(74)代理人 100120112
弁理士 中西 基晴
(74)代理人 100196508
弁理士 松尾 淳一
(74)代理人 100147991
弁理士 鳥居 健一
(74)代理人 100119781
弁理士 中村 彰吾
(74)代理人 100162846
弁理士 大牧 綾子
(74)代理人 100173565
弁理士 末松 亮太
(74)代理人 100138759
弁理士 大房 直樹
(72)発明者 ニール エー．ヤコブソン
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 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 エルシーエー－インターナショナル パテンツ内
(72)発明者 ジョン シーハン
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 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 エルシーエー－インターナショナル パテンツ内
(72)発明者 エリック ジェワート
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 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 エルシーエー－インターナショナル パテンツ内

審査官 篠塚 隆

(56)参考文献 特表２００８－５１５１００（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４４５
Ｇ０６Ｆ９／４６