(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4288295号
(P4288295)
(45)発行日　平成21年7月1日(2009.7.1)　(24)登録日　平成21年4月3日(2009.4.3)

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

請求項の数 11　(全 19 頁)

(21)出願番号　特願2007-513913 (P2007-513913)
(86)(22)出願日　平成17年5月18日 (2005.5.18)
(65)公表番号　特表2008-500619 (P2008-500619A)
(43)公表日　平成20年1月10日 (2008.1.10)
(86)国際出願番号　PCT/EP2005/052279
(87)国際公開番号　WO2005/119444
(87)国際公開日　平成17年12月15日 (2005.12.15)
審査請求日　平成20年4月9日 (2008.4.9)
(31)優先権主張番号　10/855,726
(32)優先日　平成16年5月27日 (2004.5.27)
(33)優先権主張国　米国 (US)

早期審査対象出願

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】論理パーティションにおけるメモリへのアクセス

(57)【特許請求の範囲】

【請求項１】

データ処理システムにおいて、第１の論理パーティションにおける第１のプロセスが第２の論理パーティションにおける遠隔メモリにアクセスするための方法であって、

前記遠隔メモリのための、前記遠隔メモリの記述を与える拡張メモリ間記述子を検索するステップであって、前記拡張メモリ間記述子が、異なる論理パーティションにおける遠隔メモリを記述するものとして前記拡張メモリ間記述子を示す第１のフィールドを含む、ステップと、

前記拡張メモリ間記述子に基づいて、前記第１の論理パーティションにおける前記第１のプロセスが、前記第２の論理パーティションにおける前記遠隔メモリにアクセスするステップと、

を含む、方法。

【請求項２】

前記拡張メモリ間記述子が、前記第２の論理パーティションにおける前記遠隔メモリを識別する第２のフィールドと、前記遠隔メモリのサイズを示す第３のフィールドと、前記遠隔メモリにおける開始アドレスを示す第４のフィールドと、を更に含む、請求項１に記載の方法。

【請求項３】

前記拡張メモリ間記述子が、前記第１の論理パーティションにおけるオペレーティング・システムが初期化されるときに発生される、請求項１に記載の方法。

【請求項４】
前記拡張メモリ間記述子が、他の論理パーティションの遠隔メモリを前記第１の論理パーティションのオペレーティング・システムにアタッチするオペレーティング・システム・カーネル・サービスによって発生される、請求項１に記載の方法。
【請求項５】
前記オペレーティング・システム・カーネル・サービスが、前記第１の論理パーティションにおける前記オペレーティング・システムが初期化されるときに前記オペレーティング・システム・カーネル・サービスによってパースされる前記第２の論理パーティションのためのデバイス・ツリーに基づいて前記拡張メモリ間記述子を発生する、請求項４に記載の方法。
【請求項６】
前記デバイス・ツリーがハイパーバイザによって維持される、請求項５に記載の方法。
【請求項７】
前記第１の論理パーティションがサーバ・コンピューティング・デバイスにあり、前記第２の論理パーティションがクライアント・コンピューティング・デバイスにある、請求項１に記載の方法。
【請求項８】
前記第２の論理パーティションにおける前記遠隔メモリにアクセスするステップが、前記拡張メモリ間記述子を用いて直接メモリ・アクセス動作を実行することを含む、請求項１に記載の方法。
【請求項９】
前記拡張メモリ間記述子に基づいて前記第２の論理パーティションにおける前記遠隔メモリにアクセスするステップが、
前記拡張メモリ間記述子を入出力（Ｉ／Ｏ）スタックを通して物理デバイス・ドライバに渡すステップと、
前記拡張メモリ間記述子に基づいて直接メモリ・アクセス動作を発生するステップと、
前記直接メモリ・アクセス動作をＩ／Ｏアダプタに提示するステップと、
を含み、前記Ｉ／Ｏアダプタは前記直接メモリ・アクセス動作を前記第２の論理パーティションに送信する、請求項１に記載の方法。
【請求項１０】
第１の論理パーティションにおける第１のプロセスが第２の論理パーティションにおける遠隔メモリにアクセスするためのプログラムであって、
前記遠隔メモリのための、前記遠隔メモリの記述を与える拡張メモリ間記述子を検索するステップであって、前記拡張メモリ間記述子が、異なる論理パーティションにおける遠隔メモリを記述するものとして前記拡張メモリ間記述子を示す第１のフィールドを含む、ステップと、
前記拡張メモリ間記述子に基づいて、前記第１の論理パーティションにおける前記第１のプロセスが、前記第２の論理パーティションにおける前記遠隔メモリにアクセスするステップと、
をコンピュータに実行させるプログラム。
【請求項１１】
第１の論理パーティションにおける第１のプロセスが第２の論理パーティションにおける遠隔メモリにアクセスするためのシステムであって、
前記遠隔メモリのための、前記遠隔メモリの記述を与える拡張メモリ間記述子を検索するための手段であって、前記拡張メモリ間記述子が、異なる論理パーティションにおける遠隔メモリを記述するものとして前記拡張メモリ間記述子を示す第１のフィールドを含む、手段と、
前記拡張メモリ間記述子に基づいて、前記第１の論理パーティションにおける前記第１のプロセスが、前記第２の論理パーティションにおける前記遠隔メモリにアクセスするための手段と、

を含む、システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、改良したデータ処理システムを対象とする。更に具体的には、本発明は、メモリ間記述子を拡張して、論理的にパーティションに分割したコンピューティング環境において、別のパーティションからメモリを記述するために使えるようにするためのシステムおよび方法を対象とする。
【背景技術】
【０００２】
ＡＩＸ（Ｒ）オペレーティング・システムは、ＩＢＭ社のｐＳｅｒｉｅｓ（Ｒ）マシンにおいて、論理分割（ＬＰＡＲ）をサポートする。ＬＰＡＲは、単一プロセッサ・コンプレックス上での多数のシステム・イメージの実行を可能とする。かかる各イメージは、充分なＣＰＵ（専用または共用）、中央ストレージ、拡張ストレージおよびチャネルを有する。ＬＰＡＲでは、パーティション間でリソースは明確に分離され、あるパーティションは他のいかなるパーティションのシステム環境も不安定にすることができないようになっている。ＩＢＭ社のマシンでは、パーティションのためのリソースのこの明らかな分離を可能とする機構は、ハイパーバイザである。
【０００３】
各パーティションはそれ自身の分離したリソースの組を有するので、各パーティションは、周囲のパーティションから隠れたそれ自身の「ローカル」メモリを有する。仮想入出力（ＶＩＯ）を生成するまで、あるパーティションが別のパーティションのメモリに直接アクセスする正当な理由はなかった。
【０００４】
ＶＩＯによって、ポイント・ツー・ポイントの直接メモリ・アクセス（ＤＭＡ）動作およびパーティション間のメモリ・コピー動作の概念が導入された。ＤＭＡ動作によって、あるメモリから別のメモリへの直接データ転送が可能となり、このデータ転送を実行するために中央演算処理装置（ＣＰＵ）を用いずに済む。ＤＭＡ動作では、データ転送にＣＰＵを用いなければならない場合に比べて高速のデータ転送が可能となる。メモリ・コピー動作によって、メモリの一部を、あるメモリから別のものに直接コピーすることができる。
【０００５】
このため、ＶＩＯによって、例えばＩ／Ｏアダプタのような物理リソースをパーティション間で共有することが可能となる。リソースは１つのみのパーティション（「サーバ」パーティションと呼ぶ）に属することができるので、他のパーティションは、そのリソースを用いるためにサーバ・パーティションを通過しなければならない（他のパーティションを「クライアント」パーティションと呼ぶ）。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
共有リソースがＩ／Ｏアダプタである場合、サーバ・パーティションのＩ／Ｏアダプタを用いるために、サーバ・パーティションはＩ／ＯアダプタからクライアントのメモリへのＤＭＡ動作をセットアップする必要がある。しかしながら、ＤＭＡ動作をセットアップするために、サーバ・パーティションは、クライアント・パーティションのメモリの構造に関する情報を知らなければならない。更に、サーバ・パーティションは、クライアント・パーティションのメモリ構造に関する情報を、そのＩ／Ｏスタックに渡すことができなければならない。このＩ／Ｏスタックは、規定のソフトウェア・レイヤの階層であり、使用可能なフォーマットでＩ／Ｏ動作を実行するためには、この階層を介してデータを処理しなければならない（例えば論理ボリューム・マネージャ（ＬＶＭ）、ヘッド・ドライバ、アダプタ・ドライバ、バス・ドライバ、カーネル等）。従って、サーバ・パーティショ

ンの環境においてクライアント・パーティションのメモリを記述し、この記述をサーバ・パーティションのＩ／Ｏスタックに容易に渡すことができるようなシステムおよび方法を有することは有利であろう。

【課題を解決するための手段】

【０００７】

本発明は、サーバ・パーティション環境においてメモリ間記述子の使用を拡張して、例えばクライアント・パーティションのメモリ（以降、「遠隔」メモリと称する）のような別のパーティションのメモリを記述するために使えるようにするシステムおよび方法を提供する。このシステムおよび方法によって、論理パーティションの立ち上げの間等に、論理的にパーティションに分けたコンピューティング・システムにおけるオペレーティング・システムを初期化すると、デバイス・ドライバのローディング等の起動動作を実行するために、オペレーティング・システム・カーネル・サービスを呼び出す。これらのオペレーティング・システム・カーネル・サービスの一部として、あるカーネル・サービスを呼び出して、他の論理パーティションの遠隔メモリを本オペレーティング・システムにアタッチし、遠隔メモリ・コピーおよびＤＭＡ動作が他の論理パーティションの遠隔メモリによって実行可能であるようにする。

【０００８】

アタッチ・カーネル・サービスは、コンピューティング・システムのデバイス・ツリーを調査する。これは、オープン・ファームウェア等によって生成可能であり、サーバ・パーティションにおけるオペレーティング・システムがアクセスできるようにサーバ・パーティションのメモリに維持することができる。論理パーティション当たり１つのデバイス・ツリー・インスタンスがある。デバイス・ツリーは、そのパーティションにおけるリソースのためのノードを含む。これらのリソースは、パーティションごとのローカル・メモリ、ＰＣＩ／ＩＳＡ　Ｉ／Ｏスロット、プロセッサ、サポートされたファームウェア・コール等を含む。

【０００９】

コンピューティング環境においてＶＩＯをサポートする場合、典型的な実施形態において、サーバ・パーティションのデバイス・ツリーにおける／ｖデバイス・ノードまたはその子ノードは、クライアント・パーティションのローカル・メモリの記述を含む。この記述は、クライアントのローカル・メモリの論理識別子、ローカル・メモリの開始アドレス、およびローカル・メモリの長さを含む。この情報から、各論理パーティションのローカル・メモリごとの拡張メモリ間記述子を生成することができる。次いで、これらの拡張メモリ間記述子は、遠隔メモリ・コピーおよびＤＭＡ動作を実行する際に用いるため、現行パーティションのオペレーティング・システムによって維持される。

【００１０】

動作を実行するために遠隔メモリに対するアクセスが必要である場合、サーバ・パーティションのオペレーティング・システムは、遠隔メモリ動作の対象であるクライアント・パーティションの「遠隔」メモリを記述する拡張メモリ間記述子を用いる。本発明の拡張メモリ間記述子は、メモリ間記述子を用いるパーティションのローカル・メモリのみを記述するために通常用いられるデータ構造である既存のメモリ間記述子の拡張である。

【００１１】

本発明においては、メモリ間記述子のフィールドを拡張して、別のパーティションのローカル・メモリすなわち「遠隔」メモリに関する情報をストアするために用いる。特に、拡張メモリ間記述子のフィールドは、アドレス空間識別子、遠隔メモリのサイズ、遠隔メモリを識別するための識別子（いくつかのクライアント・パーティションがある場合）、および、遠隔メモリ内の開始アドレスを含む。アドレス空間識別子は、メモリ間記述子を用いて「遠隔」メモリを表す場合、遠隔メモリ記述としてメモリ間記述子を識別する。このため、メモリ間記述子を調べるＩ／Ｏスタックまたはメモリ管理サービスにおけるいずれのインタフェースも、ここで、アドレス空間識別子を見て、これが遠隔メモリ記述であることを認識することができる。この結果、インタフェースは、ＤＭＡ動作をセットアッ

プして「遠隔」メモリにアクセスするための適切なアクションを取ることができる。
【００１２】
本発明のこれらおよび他の特徴および利点は、好適な実施形態の以下の詳細な説明において記載され、または、それを考慮することによって当業者に明らかとなろう。
【００１３】
ここで、本発明の好適な実施形態について、添付図面を参照して、一例としてのみ説明する。
【発明を実施するための最良の形態】
【００１４】
提供されるのは、第１のパーティション環境においてメモリ間記述子を拡張することによって、別のパーティションのローカル・メモリを記述するようにして、直接メモリ・アクセスまたは遠隔メモリ・コピー動作を必要とする動作を容易にするための機構である。このため、本発明の実施形態は、好ましくは、論理分割機構を用いてリソースがパーティションに分けられているコンピューティング・デバイスにおいて実装される。
【００１５】
ここで図面、特に図１を参照すると、本発明の実施形態を実装可能なデータ処理システムのブロック図が示されている。データ処理システム１００は、システム・バス１０６に接続された複数のプロセッサ１０１、１０２、１０３、および１０４を含む対称マルチプロセッサ（ＳＭＰ）・システムとすることができる。例えば、データ処理システム１００は、ＩＢＭ社（ニューヨーク州アーモンク）の製品であるＩＢＭ（Ｒ） ｅＳｅｒｖｅｒ（Ｒ）とし、ネットワーク内のサーバとして実装することができる。あるいは、単一プロセッサ・システムを用いることも可能である。また、システム・バス１０６には、複数のローカル・メモリ１６０～１６３に対するインタフェースを提供するメモリ・コントローラ／キャッシュ１０８も接続されている。Ｉ／Ｏバス・ブリッジ１１０が、システム・バス１０６に接続され、Ｉ／Ｏバス１１２に対するインタフェースを提供する。メモリ・コントローラ／キャッシュ１０８およびＩ／Ｏバス・ブリッジ１１０は、図示するように統合することができる。
【００１６】
データ処理システム１００は、論理パーティション化（ＬＰＡＲ）データ処理システムである。このため、データ処理システム１００は、同時に動作する複数の異種オペレーティング・システム（または単一オペレーティング・システムの複数インスタンス）を有することができる。これらのオペレーティング・システムの各々は、その内部で実行する任意の数のソフトウェア・プログラムを有することができる。データ処理システム１００は、論理的にパーティション化して、異なるＰＣＩ Ｉ／Ｏアダプタ１２０～１２１、１２８～１２９、および１３６、グラフィック・アダプタ１４８、ならびにハード・ディスク・ドライブ１４９を、異なる論理パーティションに割り当てることができるようになっている。この場合、グラフィック・アダプタ１４８は、ディスプレイ・デバイス（図示せず）に対する接続を与え、ハード・ディスク・アダプタ１４９は、ハード・ディスク１５０を制御するための接続を与える。
【００１７】
このため、例えば、データ処理システム１００が３つの論理パーティションすなわちＰ１、Ｐ２、およびＰ３に分割されると仮定する。ＰＣＩ Ｉ／Ｏアダプタ１２０～１２１、１２８～１２９、１３６の各々、グラフィック・アダプタ１４８、ハード・ディスク・アダプタ１４９、ホスト・プロセッサ１０１～１０４の各々、およびローカル・メモリ１６０～１６３からのメモリが、３つのパーティションの各々に割り当てられる。これらの例では、メモリ１６０～１６３は、デュアル・インライン・メモリ・モジュール（ＤＩＭＭ）の形態を取ることができる。ＤＩＭＭは、通常、ＤＩＭＭごとにパーティションに割り当てられない。その代わり、パーティションは、プラットフォームから見たメモリ全体の一部分を得る。例えば、プロセッサ１０１、ローカル・メモリ１６０～１６３からのメモリの一部、およびＩ／Ｏアダプタ１２０、１２８、および１２９は、論理パーティショ

ンＰ１に割り当てることができる。プロセッサ１０２～１０３、ローカル・メモリ１６０～１６３からのメモリの一部、Ｉ／Ｏアダプタ１２１および１３６は、パーティションＰ２に割り当てることができる。プロセッサ１０４、ローカル・メモリ１６０～１６３からのメモリの一部、グラフィック・アダプタ１４８、およびハード・ディスク・アダプタ１４９は、論理パーティションＰ３に割り当てることができる。

【００１８】

データ処理システム１００内で実行する各オペレーティング・システムは、異なる論理パーティションに割り当てられる。このため、データ処理システム１００内で実行する各オペレーティング・システムは、その論理パーティション内にあるＩ／Ｏユニットにのみアクセスすることができる。このため、例えば、ＡＩＸ（Ｒ）オペレーティング・システムの１つのインスタンスをパーティションＰ１内で実行し、ＡＩＸ（Ｒ）オペレーティング・システムの第２のインスタンス（イメージ）をパーティションＰ２内で実行し、Ｌｉｎｕｘ（Ｒ）またはＯＳ／４００（Ｒ）オペレーティング・システムを論理パーティションＰ３内で動作することができる。

【００１９】

Ｉ／Ｏバス１１２に接続された周辺機器相互接続（ＰＣＩ）ホスト・ブリッジ１１４は、ＰＣＩローカル・バス１１５に対するインタフェースを提供する。複数のＰＣＩ入力／出力アダプタ１２０～１２１を、ＰＣＩ－ＰＣＩブリッジ１１６、ＰＣＩバス１１８、ＰＣＩバス１１９、Ｉ／Ｏスロット１７０、およびＩ／Ｏスロット１７１を介して、ＰＣＩバス１１５に接続することができる。ＰＣＩ－ＰＣＩブリッジ１１６は、ＰＣＩバス１１８およびＰＣＩバス１１９に対するインタフェースを与える。ＰＣＩ　Ｉ／Ｏアアプタ１２０および１２１は、Ｉ／Ｏスロット１７０および１７１内にそれぞれ配置される。典型的なＰＣＩバス実装は、４個乃至８個のＩ／Ｏアダプタ（すなわちアドイン・コネクタのための拡張スロット）をサポートする。各ＰＣＩ　Ｉ／Ｏアダプタ１２０～１２１は、データ処理システム１００と、例えばデータ処理システム１００のクライアントである他のネットワーク・コンピュータ等の入出力デバイスとの間のインタフェースを与える。

【００２０】

追加のＰＣＩホスト・ブリッジ１２２は、追加のＰＣＩバス１２３のためのインタフェースを提供する。ＰＣＩバス１２３は、複数のＰＣＩ　Ｉ／Ｏアダプタ１２８～１２９に接続されている。ＰＣＩ　Ｉ／Ｏアダプタ１２８～１２９は、ＰＣＩ－ＰＣＩブリッジ１２４、ＰＣＩバス１２６、ＰＣＩバス１２７、Ｉ／Ｏスロット１７２、およびＩ／Ｏスロット１７３を介して、ＰＣＩバス１２３に接続することができる。ＰＣＩ－ＰＣＩブリッジ１２４は、ＰＣＩバス１２６およびＰＣＩバス１２７に対するインタフェースを提供する。ＰＣＩ　Ｉ／Ｏアダプタ１２８および１２９は、Ｉ／Ｏスロット１７２および１７３内にそれぞれ配置される。このように、例えばモデムまたはネットワーク・アダプタ等の追加のＩ／Ｏデバイスを、ＰＣＩ　Ｉ／Ｏアダプタ１２８～１２９の各々を介してサポートすることができる。このように、データ処理システム１００は、複数のネットワーク・コンピュータに対する接続を可能とする。

【００２１】

Ｉ／Ｏスロット１７４に挿入されたメモリ・マップ・グラフィック・アダプタ１４８は、ＰＣＩバス１４４、ＰＣＩ－ＰＣＩブリッジ１４２、ＰＣＩバス１４１、およびＰＣＩホスト・ブリッジ１４０を介してＩ／Ｏバス１１２に接続することができる。ハード・ディスク・アダプタ１４９は、ＰＣＩバス１４５に接続されたＩ／Ｏスロット１７５内に配置される。そして、このバスは、ＰＣＩ－ＰＣＩブリッジ１４２に接続され、これは、ＰＣＩバス１４１によってＰＣＩバス・ブリッジ１４０に接続されている。

【００２２】

ＰＣＩホスト・ブリッジ１３０は、ＰＣＩバス１３１のためのインタフェースを提供して、Ｉ／Ｏバス１１２に接続する。ＰＣＩ　Ｉ／Ｏアダプタ１３６は、Ｉ／Ｏスロット１７６に接続され、これは、ＰＣＩバス１３３によってＰＣＩ－ＰＣＩブリッジ１３２に接続されている。ＰＣＩ－ＰＣＩブリッジ１３２は、ＰＣＩバス１３１に接続されている。

また、このＰＣＩバスは、ＰＣＩホスト・ブリッジ１３０を、サービス・プロセッサ・メールボックス・インタフェースおよびＩＳＡバス・アクセス通過論理１９４およびＰＣＩ－ＰＣＩブリッジ１３２に接続する。サービス・プロセッサ・メールボックス・インタフェースおよびＩＳＡバス・アクセス通過論理１９４は、ＰＣＩ／ＩＳＡブリッジ１９３に宛てられたＰＣＩアクセスを転送する。ＮＶＲＡＭストレージ１９２は、ＩＳＡバス１９６に接続されている。サービス・プロセッサ１３５は、サービス・プロセッサ・メールボックス・インタフェースおよびＩＳＡバス・アクセス通過論理１９４に、そのローカルＰＣＩバス１９５を介して結合されている。また、サービス・プロセッサ１３５は、複数のＪＴＡＧ／I^2Cバス１３４を介してプロセッサ１０１～１０４に接続されている。ＪＴＡＧ／I^2Cバス１３４は、ＪＴＡＧ／スキャン・バス（ＩＥＥＥ１１４９．１を参照）およびＰｈｉｌｌｉｐｓ　I^2Cバスを組み合わせたものである。しかしながら、代替として、ＪＴＡＧ／I^2Cバス１３４は、Ｐｈｉｌｌｉｐｓ　I^2CバスのみまたはＪＴＡＧ／スキャン・バスのみによって置換することができる。ホスト・プロセッサ１０１、１０２、１０３、および１０４の全てのＳＰ－ＡＴＴＮ信号は、共に、サービス・プロセッサの割り込み入力信号に接続されている。サービス・プロセッサ１３５は、それ自身のローカル・メモリ１９１を有し、ハードウェアＯＰパネル１９０に対するアクセスを有する。

【００２３】

データ処理システム１００に最初に電源投入した場合、サービス・プロセッサ１３５は、ＪＴＡＧ／I^2Cバス１３４を用いて、システム（ホスト）・プロセッサ１０１～１０４、メモリ・コントローラ／キャッシュ１０８、およびＩ／Ｏブリッジ１１０に質問する。このステップが完了すると、サービス・プロセッサ１３５は、データ処理システム１００のインベントリおよびトポロジの理解を得る。また、サービス・プロセッサ１３５は、ホスト・プロセッサ１０１～１０４、メモリ・コントローラ／キャッシュ１０８、およびＩ／Ｏブリッジ１１０に質問することによって見出された全ての要素に対して、内蔵自己テスト（ＢＩＳＴ）、基本検証テスト（ＢＡＴ）、およびメモリ・テストを実行する。ＢＩＳＴ、ＢＡＴ、およびメモリ・テストの間に検出された故障についてのエラー情報は全て、サービス・プロセッサ１３５によって収集されて報告される。

【００２４】

ＢＩＳＴ、ＢＡＴ、およびメモリ・テストの間に故障と見なされた要素を除外した後に、システム・リソースの意味のある有効な構成が可能である場合、データ処理システム１００は、先に進んで、実行可能コードをローカル（ホスト）・メモリ１６０～１６３にロードすることができる。次いで、サービス・プロセッサ１３５は、ローカル・メモリ１６０～１６３にロードされたコードを実行するため、ホスト・プロセッサ１０１～１０４を解放する。ホスト・プロセッサ１０１～１０４がデータ処理システム１００内の各オペレーティング・システムからのコードを実行している間、サービス・プロセッサ１３５はエラーの監視および報告のモードに入る。サービス・プロセッサ１３５が監視するアイテムのタイプは、例えば、冷却ファンの速度および動作、熱センサ、電源レギュレータ、ならびに、プロセッサ１０１～１０４、ローカル・メモリ１６０～１６３、およびＩ／Ｏブリッジ１１０によって報告された回復可能および回復不可能エラーを含む。

【００２５】

サービス・プロセッサ１３５は、データ処理システム１００内の監視されている全アイテムに関するエラー情報を保存し報告する責任がある。また、サービス・プロセッサ１３５は、エラーのタイプおよび規定された閾値に基づいてアクションを取る。例えば、サービス・プロセッサ１３５は、プロセッサのキャッシュ・メモリ上の過度の回復可能エラーを記録し、これがハード故障を予測するものであることを決定することができる。この決定に基づいて、サービス・プロセッサ１３５は、現在の実行セッションおよび今後の初期プログラム・ロード（ＩＰＬ）の間に構成解除するためにそのリソースをマークすることができる。また、ＩＰＬは、時に、「ブート」または「ブートストラップ」と呼ばれる。

【００２６】

データ処理システム１００は、様々な市販のコンピュータ・システムを用いて実施可能である。例えば、データ処理システム１００は、ＩＢＭ社から入手可能なＩＢＭ（Ｒ）ｅＳｅｒｖｅｒ（Ｒ） iＳｅｒｉｅｓ（Ｒ） Ｍｏｄｅｌ ８４０システムを用いて実装することができる。かかるシステムは、ＯＳ／４００（Ｒ）オペレーティング・システムを用いた論理分割をサポートすることができる。ＯＳ／４００（Ｒ）オペレーティング・システムも、ＩＢＭ社から入手可能である。

【００２７】

図１に示すハードウェアが様々に変化し得ることは、当業者には認められよう。例えば、図示したハードウェアに加えて、またはその代わりに、光ディスク・ドライブ等の他の周辺デバイスも使用可能である。図示した例は、本発明に対してアーキテクチャ上の限定を加えることを意味しない。

【００２８】

ここで図２を参照すると、本発明の実施形態を実装することができる例示的な論理パーティション化プラットフォームのブロック図が示されている。論理パーティション化プラットフォーム２００におけるハードウェアは、例えば図１のデータ処理システム１００として実装可能である。論理パーティション化プラットフォーム２００は、パーティション化ハードウェア２３０、オペレーティング・システム２０２、２０４、２０６、２０８、および、パーティション管理ファームウェア２１０を含む。オペレーティング・システム２０２、２０４、２０６、および２０８は、単一オペレーティング・システムの複数コピー、または、論理パーティション化プラットフォーム２００上で同時に動作する複数の異種オペレーティング・システムとすることができる。これらのオペレーティング・システムは、ＯＳ／４００（Ｒ）を用いて実装可能であり、これは、ハイパーバイザ等のパーティション管理ファームウェアとインタフェースするように設計されている。ＯＳ／４００（Ｒ）は、これらの例示的な実施形態において、一例としてのみ用いられる。むろん、具体的な実施に応じて、ＡＩＸ（Ｒ）およびＬｉｎｕｘ（Ｒ）等の他のタイプのオペレーティング・システムも使用可能である。オペレーティング・システム２０２、２０４、２０６、および２０８は、パーティション２０３、２０５、２０７、および２０９に配置されている。ハイパーバイザ・ソフトウェアは、パーティション管理ファームウェア２１０を実装するために使用可能であるソフトウェアの一例であり、ＩＢＭ社から入手可能である。ファームウェアは、例えばリード・オンリ・メモリ（ＲＯＭ）、プログラマブルＲＯＭ（ＰＲＯＭ）、消去可能プログラマブルＲＯＭ（ＥＰＲＯＭ）、電気的消去可能プログラマブルＲＯＭ（ＥＥＰＲＯＭ）、および不揮発性ランダム・アクセス・メモリ（不揮発性ＲＡＭ）等の、電力なしでその内容を保持するメモリ・チップにストアされる「ソフトウェア」である。

【００２９】

更に、これらのパーティションは、パーティション・ファームウェア２１１、２１３、２１５、および２１７も含む。パーティション・ファームウェア２１１、２１３、２１５、および２１７は、初期ブート・ストラップ・コード、ＩＥＥＥ－１２７５標準オープン・ファームウェア、および、ＩＢＭ社から入手可能なランタイム・アブストラクション・ソフトウェア（ＲＴＡＳ）を用いて実装可能である。パーティション２０３、２０５、２０７、および２０９をインスタンス化するとき、プラットフォーム・ファームウェア２１０によって、ブート・ストラップ・コードのコピーがパーティション２０３、２０５、２０７、および２０９にロードされる。その後、制御はブート・ストラップ・コードに移され、次いでブート・ストラップ・コードは、オープン・ファームウェアおよびＲＴＡＳをロードする。次いで、パーティションに関連するまたはパーティションに割り当てられたプロセッサが、パーティションのメモリにディスパッチされて、パーティション・ファームウェアを実行する。

【００３０】

パーティション化ハードウェア２３０は、複数のプロセッサ２３２～２３８、複数のシステム・メモリ・ユニット２４０～２４６、複数の入出力（Ｉ／Ｏ）アダプタ２４８～２

６２、およびストレージ・ユニット２７０を含む。プロセッサ２３２～２３８、メモリ・ユニット２４０～２４６、ＮＶＲＡＭストレージ２９８、およびＩ／Ｏアダプタ２４８～２６２の各々は、論理パーティション化プラットフォーム２００内の複数のパーティションのうち１つに割り当てることができる。パーティションの各々は、オペレーティング・システム２０２、２０４、２０６、および２０８に対応する。

【００３１】

パーティション管理ファームウェア２１０は、パーティション２０３、２０５、２０７、および２０９のための多数の機能およびサービスを実行して、論理パーティション化プラットフォーム２００のパーティショニングを実施する。パーティション管理ファームウェア２１０は、基礎にあるハードウェアと同一のファームウェア実装仮想マシンである。このため、パーティション管理ファームウェア２１０は、論理パーティション化プラットフォーム２００の全てのハードウェア・リソースを仮想化することによって、独立したＯＳイメージ２０２、２０４、２０６、および２０８の同時実行を可能とする。

【００３２】

サービス・プロセッサ２９０を用いて、パーティションにおけるプラットフォーム・エラーの処理等、様々なサービスを提供することができる。また、これらのサービスは、ＩＢＭ社等のベンダーにエラーを報告するサービス・エージェントとして機能することができる。異なるパーティションの動作は、ハードウェア管理コンソール２８０等のハードウェア管理コンソールを介して制御することができる。ハードウェア管理コンソール２８０は、別個のデータ処理システムであり、これによって、システム・アドミニストレータは、異なるパーティションへのリソース再割り振りを含む様々な機能を実行することができる。

【００３３】

かかる論理パーティション化環境において、カーネルと、現在のプロセス・アドレス空間以外のアドレス空間との間でデータを動かそうとする場合はいつでも、メモリ間カーネル・サービスを利用してこのデータ移動を実行する。アドレス空間の１つの領域内のデータ・エリアが、メモリ間動作のためのユーザ・バッファにアタッチするｘｍａｔｔａｃｈカーネル・サービスを呼び出すことによってアタッチされる。呼び出されると、ｘｍａｔｔａｃｈカーネル・サービスはメモリ間記述子を発生する。その後、他のメモリ間カーネル・サービスを利用して、カーネルから現在のプロセス・アドレス以外のアドレス空間に、データを移動またはコピーすることができる。例えば、ｘｍｅｍｉｎカーネル・サービスは、指定されたアドレス空間からカーネル・グローバル・メモリにデータをコピーすることによってメモリ間移動を実行する。ｘｍｅｍｏｕｔカーネル・サービスは、カーネル・グローバル・メモリから指定されたアドレス空間にデータをコピーすることによってメモリ間移動を実行する。ｘｍｅｍｄｍａカーネル・サービスは、ＤＭＡ　Ｉ／Ｏのためのメモリのページを準備するか、または、ＤＭＡ　Ｉ／Ｏが完了した後にページを処理する。

【００３４】

既知のコンピューティング・システムにおいて、メモリ間記述子は、ローカル・メモリを記述するために用いられるデータ構造である。ローカル・メモリのメモリ間記述子は、オペレーティング・システムの仮想メモリ管理（ＶＭＮ）コンポーネントによって維持される情報から発生される。

【００３５】

本発明の好適な実施形態による拡張メモリ間記述子は、ブート時にオペレーティング・システムによって発生されパースされるデバイス・ツリーに基づいてカーネル・サービスによって発生される。デバイス・ツリーは、システム・ハードウェアおよびユーザ構成選択を記述する階層データ構造であるオープン・ファームウェア・デバイス・ツリーと同様である。また、オープン・ファームウェア・デバイス・ツリーは、ハードウェア・ドライバと、これらのドライバによって用いるサポート・ルーチンを含む。以下は、オープン・ファームウェア・デバイス・ツリーの一例である。

【００３６】

```
root /
ff8885f8 /rtas
ff866bf4/rom@ff000000
ff8627c0/flash@fff00000
ff8513d0 /cpus
ff88a1c8 /PowerPC,604ev@0
ff88a788 /l2-cache
ff84d8e0 /pci
ff89552c /ethernet@4
ff8952e0 /display
ff88cf44 /mac-io@2
ff893e68 /misc@0
ff894688 /iic
ff893d7c /via@16000
ff8939fc /escc@13000
ff893c2c/ch-b@13000
ff893af4/ch-a@13020
ff88fd08 /scsi@10000
ff892e1c /tape
ff8924d0 /disk
ff88f944 /escc-legacy@12000
ff88fba8/ch-b@12000
ff88fa60/ch-a@12002
ff88d78c /adb@11000
ff88f364 /mouse@3
ff88e6dc/keyboard@2
ff88d638 /open-pic@40000
ff88aad8 /ide@1,1
ff88c504 /disk
ff85a534 /isa@1
ff864208 /sound@i534
ff8640e8 /midi@i330
ff863ff8 /game@i200
ff863368 /gpio@i800
ff863008 /nvram@me0000
ff862aa4 /rtc@i70
ff85f644 /8042@i60
ff8618b0 /mouse@aux
ff860260 /keyboard@
ff85d804 /floppy@i3f0
ff85d3b4 /parallel@i3bc
ff85c704 /serial@i2f8
ff85b9f4 /serial@i3f8
ff85b490 /timer@i40
ff85b01c /interrupt-controller@i20
ff85ae08 /dma-controller@i00
ff84a650 /mmu
ff83f2e4/memory@0
```

【００３７】

本発明の一実施形態によって、同様のデバイス・ツリー表現を用いて、拡張メモリ間記述子を発生させるための情報を得る。デバイス・ツリーは、オブジェクト指向であり、デバイス・ツリー内のエントリの各々が関連する特性を有し、これを調べることができるようになっている。デバイス・ツリーにおいて、遠隔メモリに関する情報をオペレーティング・システムに提示することができるノードの一例は、以下に示す通りである。

【００３８】

```
/devicenode
 …
remote-memory-info10000000 00200000 00040000
 20000000 003D0000 00020000
 …
…
```

【００３９】

ここで、「remote-memory-info（遠隔メモリ情報）」は、ノード「/devicenode」の特性である。クライアント－サーバ仮想Ｉ／Ｏモデルの典型的な実装において、ノード「/devicenode」は、サーバ・デバイスのノードを表す。以下に述べるように、このノードはパースすることができ、このノードのパースから得た情報を用いて、本発明の好適な実施形態に従った拡張メモリ間記述子を発生することができる。このため、「10000000」エントリは、メモリの論理識別子であり、「00200000」エントリは、メモリの開始アドレスであり、「0040000」エントリは、最初のクライアントについてのバイト単位のメモリ・サイズである。同様に、「20000000」は、メモリの論理識別子であり、「03D00000」は、メモリ内の開始アドレスであり、「0020000」は、第２のクライアントについてのバイト単位のメモリ・サイズである。このトリプレット（論理ｉｄ、開始アドレス、サイズ）は、サーバが対応するクライアントごとに繰り返すことができる。

【００４０】

このデバイス・ツリー情報から、システム・ハードウェアおよびユーザ構成選択の詳細な記述を得ることができる。特に、先に示したように、デバイス・ツリーのメモリ・ノードは、メモリの論理識別子、メモリの開始アドレス、およびメモリの長さを示す特性を含む。この情報は、カーネル・サービスによって用いて、コンピューティング・システムのローカル・メモリ・リソースのためのメモリ間記述子を発生させることができる。

【００４１】

メモリ間記述子を、対象のメモリの所有者の知識を必要とせずに、２つの無関係なコンテキストからメモリにアクセスするための方法として用いる。例えば、割り込みハンドラがデータをプロセスに渡す必要がある場合、これは、メモリ間記述子を用いて、プロセスのメモリ空間に関する情報を得て、プロセスのメモリ空間にデータをコピーする。これが必要である理由は、割り込みハンドラが、必ずしもそれが所有するデータのプロセスに関連して動作しているわけではなく、従って、どのプロセスにデータが属するかに関して知らないからである。割り込みハンドラが知っているのは、データをコピーしなければならないメモリの記述、すなわちメモリ間記述子である。

【００４２】

ローカル・メモリのメモリ間記述子の使用の別の例は、ディスクＩ／Ｏである。例えば、ユーザ・レベルのアプリケーションは、ＩＯＣＴＬを呼び出して、ディスクに関する情報を読むことができる。この情報は、ユーザ空間内にある、アプリケーションが供給するバッファにストアされる。ディスク・ドライバが呼び出されると、ディスク・ドライバはｘｍａｔｔｃｈを呼び出してアプリケーションのバッファをアタッチする。これは全て、ディスク・ドライバの上半分によって行われる。

【００４３】

将来のある時点で、Ｉ／Ｏ要求が完了し、データが利用可能となった場合、ディスク・ドライバの下半分内にあるディスク・ドライバの「ｉｏｄｏｎｅ」ルーチンが、ｘｍｏｕｔを呼び出して、メモリ間記述子を用いてユーザ空間にデータをコピーする。メモリ間記

述子は、ディスク・ドライバによってストアされ、上半分および下半分の双方に利用可能とする。

【００４４】

メモリ間記述子は、オペレーティング・システムの初期化の際にカーネル・サービスによって発生され、必要に応じて、コンピューティング・デバイスのＩ／Ｏスタックにおいて、インタフェースからインタフェース（またはメモリ管理サービス）へと渡される。Ｉ／Ｏスタックのインタフェースおよびメモリ管理サービスは、メモリ間記述子を調べて、メモリ間動作を実行するために必要な情報を得る。

【００４５】

オペレーティング・システムは、別のパーティションの遠隔メモリに関して同様の問題に直面する。なぜなら、遠隔メモリ・コピー／ＤＭＡ動作は、別のパーティションにおける所有者デバイスの知識を持たないからである。このため、本明細書中に記載する手法は、メモリ間記述子の拡張バージョンを利用して、遠隔メモリ・コピー／ＤＭＡ動作を実行する場合に、別のパーティションの遠隔メモリに関する情報を提供する。

【００４６】

ローカル・メモリ間記述子は、４つの主フィールドから成る。すなわち、アドレス空間識別子、セグメント数、開始セグメント識別子、およびセグメント内の開始アドレスである。アドレス空間識別子は、データをコピーするメモリの一意の識別子を提供する。セグメント数は、アドレス空間識別子が識別するアドレス空間のセグメントの合計数を提供する。開始セグメント識別子は、セグメントの合計数内で、コピー対象データの書き込みを始めるセグメントを識別する。セグメント内の開始アドレスは、開始セグメント識別子が識別したセグメント内で、コピー対象データの書き込みを始めるアドレスを識別する。

【００４７】

以下は、これらの主フィールドを例示するメモリ間データ構造の一例である。

【００４８】

```
enum asid={LOCAL=0, REMOTE=1};
structcross_memory {
 enumasid address_space_id;  /* LOCAL, REMOTE, etc. */
intnumber_of_segments; /* number of segmentsif address_space_id=LOCAL*/
long segment_id;  /* segmentnumber if address_space_id=LOCAL */
char *vaddr;  /*starting address within the segment if address_space_id=LOCAL*/
uint flags;  /* anyspecial attributes of the segment oraddress range */
};
#defineremote_size (number_of_segments) /* size ofremote memory if address_space
_id=REMOTE */
#defineremote_logical_id (segment_id) /*remote memory's logical identifier*/
#define remote_start_addr (vaddr) /*starting address within the remote memory */
```

【００４９】

上記のフィールドから明らかなのは、特定のアドレス空間の所有者を知る必要がなく、従って、アドレス空間にアクセスするために所有者にデータを渡す必要なしにメモリ間動作を実行するために使用可能であるという特異なメモリの記述を、メモリ間記述子が与えることである。本発明の実施形態は、この既知のデータ構造を利用して、完了のためにリソースの所有権に関する情報を必要としない異なる動作を容易にする。すなわち、メモリ間記述子の拡張バージョンを用いて、あるパーティションから別のパーティションに対する遠隔メモリ・コピー／ＤＭＡ動作を容易にする（ここで、「遠隔」メモリは、現在実行しているプロセスのパーティションとは異なるパーティションにおけるメモリである）。かかる動作により、あるパーティションにおけるプロセスは、仮想入出力（ＶＩＯ）プロセスの一部として別のパーティションのメモリにアクセスすることができ、遠隔メモリの所有権を有する特定のプロセスにデータを渡す必要はない。この結果、拡張メモリ間記述子を用いることによって、パーティション間の遠隔メモリ・コピーおよびＤＭＡ動作が可

能となる。

【００５０】

メモリ間記述子フィールドを用いて、遠隔メモリに関する情報をストアする。メモリ間記述子の元の４つのフィールドをオーバーロードして、遠隔メモリを記述し、これによって、以下の４つのフィールド、すなわち、アドレス空間識別子、遠隔メモリのサイズ、遠隔メモリを識別する識別子、および遠隔メモリ内の開始アドレスを有する拡張メモリ間記述子を発生する。拡張メモリ間記述子のアドレス空間識別子は、元のメモリ間記述子のアドレス空間識別子と同様であるが、異なる点は、このフィールドに所定の値をストアして、拡張メモリ間記述子がクライアント・パーティションの遠隔メモリに対応することを示すことである。

【００５１】

遠隔メモリのサイズは、バイトまたは他の任意のメモリ・サイズ単位で指定することができ、アクセスを実行する遠隔メモリの合計サイズを指定する。遠隔メモリ識別子は、特定のクライアント・パーティションにおける特定の遠隔メモリを識別する。これは、複数のクライアント・パーティションがあり、どの遠隔メモリにアクセスしているかを指定する必要がある場合に重要である。遠隔メモリ内の開始アドレスは、遠隔メモリ識別子が指定する遠隔メモリ内のアクセスを開始する位置を提供する。遠隔メモリ・コピー／ＤＭＡ動作を実行するため、これらの４つのフィールドから、遠隔メモリの全体像を得る。

【００５２】

拡張メモリ間記述子は、元のメモリ間記述子と同じ方法で、Ｉ／Ｏスタックを通してインタフェースからインタフェースへ渡すことができる。拡張メモリ間記述子を調べる必要があるインタフェースおよびメモリ管理サービスは、拡張メモリ間記述子を処理するためのコードを含むように増強する。すなわち、それらのインタフェースおよびメモリ管理サービスは、アドレス空間識別子が拡張メモリ間記述子を識別する所定の値を含む場合、それを遠隔メモリ・アクセス要求のための遠隔メモリを記述するとして認識するコードを含むように増強する。次いで、インタフェースおよびメモリ管理サービスは、拡張メモリ間記述子が提供する遠隔メモリのサイズ、遠隔メモリの識別子、および遠隔メモリ内の開始アドレスに基づいて、適切なアクションを実行することができる。例えば、d_map_pageおよびd_map_listインタフェースは、遠隔メモリ間記述子を用いて、サーバ・パーティションが所有する物理デバイスから遠隔メモリへのＤＭＡを正しくセットアップする。このため、遠隔メモリ間記述子は、物理デバイスのドライバによって、d_map_page/listインタフェースに渡されて、ＤＭＡをセットアップする。サーバ・パーティションの定義により、物理デバイスはサーバ・パーティションに属する。この結果、遠隔メモリ・コピー／ＤＭＡ動作をパーティション間で実行することができる。

【００５３】

元のメモリ間記述子と同じフォーマットを本質的に有するが、メモリ間記述子のフィールドにストアされた異なるデータおよびこの異なるデータに関連する異なる含意（connotation）を有する拡張メモリ間記述子に、既存のメモリ間記述子をオーバーロードすることによって、本発明の実施形態は、メモリ間記述子を必要とするがメモリ間記述子の内容を調べないインタフェースおよびメモリ管理サービスの多くを、変更なしで動作させることができる。すなわち、拡張メモリ間記述子によって、オペレーティング・システムは、メモリ間記述子を必要とするが実際にそれを調べないインタフェースおよびメモリ管理サービスとのバイナリ互換性を維持することができる。例えば、ほとんどのデバイス・ドライバは、メモリ間記述子を必要とするが、それを調べない。それらは、単に、メモリ間記述子をカーネルまたはＰＣＩバス・ドライバに渡すだけである。かかるデバイス・ドライバは、メモリ間記述子に対する変更のため、再コンパイルする必要がなく、それらは、いつもと変わらず動作し続けることができる。

【００５４】

図３は、本発明の１つの例示的な実施形態に従った拡張メモリ間記述子を発生させるための機構の一例を示す例示的な図である。図３に示すように、論理パーティション３１０

のブートアップの間等、論理パーティション化コンピューティング・システムにおいて論理パーティション３１０のオペレーティング・システム３２０を初期化すると、デバイス・ドライバのローディング等のブートアップ動作を実行するため、オペレーティング・システム・カーネル・サービス３２５が呼び出される。これらのオペレーティング・システム・カーネル・サービス３２５の一部として、カーネル・サービスを呼び出し、他の論理パーティション３１２～３１６の遠隔メモリを、本オペレーティング・システムにアタッチし、遠隔メモリ・コピーおよびＤＭＡ動作が他の論理パーティションの遠隔メモリによって実行可能であるようにする。本発明の１つの例示的な実施形態では、遠隔メモリをアタッチするカーネル・サービスは、xmattach_remioと称し、既知のxmattachメモリ間カーネル・サービスの拡張バージョンである。xmattach_remioは、既知のxmemoutおよびxmeminメモリ間カーネル・サービスならびにd_map_pageおよびd_map_listカーネル・サービスの拡張バージョンによって実行可能であるメモリ間動作のための遠隔メモリをアタッチする。

【００５５】

拡張アタッチ・カーネル・サービスは、ハイパーバイザ３３０等に維持することができるコンピューティング・システムのデバイス・ツリー３３２～３３８を調べる。ハイパーバイザ３３０は、論理パーティション化コンピューティング・システムにおいて複数の仮想マシンを提供し管理する機構である。論理パーティションごとの個々のデバイス・ツリー３３２～３３８は、ハイパーバイザ３３０に維持される。更に、論理パーティションのためのデバイス・ツリー３３２～３３８のコピーは、論理パーティションのオペレーティング・システム・カーネルに利用可能であるように、その論理パーティションのローカル・メモリにストアすることができる。オペレーティング・システムは、デバイス・ツリー３３２～３３８をパースするためにカーネル拡張（例えばデバイス・ドライバ）およびユーザ・レベル・コードが使用可能である関数のライブラリを提供する。

【００５６】

デバイス・ツリー３３２～３３８は、コンピューティング・システムおよびそれらのリソースのパーティションごとのノードを含む。これらのリソースは、パーティションごとのローカル・メモリ３４０～３６０を含む。パーティションのこれらのローカル・メモリを表すデバイス・ツリー３３２～３３８のノードは、ローカル・メモリの論理識別子、ローカル・メモリの開始アドレス、およびローカル・メモリの長さを含むローカル・メモリの特性を含む。この情報から、論理パーティションごとのローカル・メモリ（これは、現在の論理パーティションに対して「遠隔」メモリである）のサイズ、論理パーティションごとのローカル・メモリの識別子、および論理パーティションごとのローカル・メモリ内の開始アドレスを得ることができる。この情報は、各論理パーティションのローカル・メモリごとに、拡張メモリ間記述子３７０～３９０にパッケージすることができる。これらの拡張メモリ間記述子３７０～３９０は、次いで、他の論理パーティション３１２～３１６に対する遠隔メモリ・コピーおよびＤＭＡ動作を実行する際に用いるため、現行論理パーティション３１０のオペレーティング・システム３２０によって維持される。

【００５７】

図４は、拡張メモリ間記述子を用いて遠隔メモリ・アクセスを必要とする動作を実行するための、本発明の１つの例示的な実施形態に従った機構の一例を示す例示的な図である。図４に示すように、クライアント・デバイス４００のアプリケーション４１０等のプロセスが、例えば、read()システム・コールによって、遠隔メモリ・アクセス動作を実行することを求める入出力（Ｉ／Ｏ）要求を、オペレーティング・システム・カーネル空間４２０に発生すると、このＩ／Ｏ要求は、Ｉ／Ｏスタック４２２からクライアント仮想デバイス・ドライバ４２４に至るパスで処理される。クライアント仮想デバイス・ドライバ４２４は、この要求を、サーバ４８０の対応するもの、すなわちサーバ仮想ドライバ４３０に転送する。

【００５８】

サーバ仮想ドライバ４３０は、ブート時に、例えば論理パーティション２のような遠隔

パーティションのメモリ４１５のための拡張メモリ間記述子４４０をすでに生成している。例えば、サーバ仮想ドライバ４３０は、ブート時に、xmattach_remioカーネル・サービスを呼び出して、拡張メモリ間記述子を発生することができる。サーバ仮想ドライバ４３０は、拡張メモリ間記述子４４０と共に、Ｉ／Ｏ要求を、サーバ側のＩ／Ｏスタック４５０に渡す。

【００５９】

Ｉ／Ｏ要求は、最終的に、Ｉ／Ｏスタック４５０を介して、例えばＳＣＳＩアダプタ・ドライバのような物理デバイス・ドライバ４６０に渡される。これは、そのＩ／ＯアダプタＩ／Ｏアダプタ４７０を使用するＤＭＡ動作をスケジューリングする。Ｉ／ＯアダプタＩ／Ｏアダプタ４７０によってＤＭＡが完了すると、データは遠隔メモリ４１５に書き込まれる。このため、先の例に示したように、データ自体に加えて、遠隔直接メモリ・アクセスを実行するために要求される他の必要なエレメントは、拡張メモリ間記述子によって与えられる遠隔メモリの記述である。

【００６０】

図５および図６は、本発明の例示的な実施形態による、拡張メモリ間記述子を発生し用いるための例示的なプロセスを示すフローチャートである。フローチャート図の各ブロックおよびフローチャート図のブロックの組み合わせが、コンピュータ・プログラム命令によって実装可能であることは、理解されよう。これらのコンピュータ・プログラム命令を、プロセッサまたは他のプログラマブル・データ処理装置に提供して、マシンを生成し、それにより、プロセッサまたは他のプログラマブル・データ処理装置上で実行する命令で、フローチャートのブロックに明示される機能を実装するための手段を生成することができる。また、これらのコンピュータ・プログラム命令は、コンピュータ読み取り可能メモリまたはストレージ媒体にストアして、プロセッサまたは他のプログラマブル・データ処理装置を特定の方法で機能させることができ、それにより、コンピュータ読み取り可能メモリまたはストレージ媒体にストアされた命令が、フローチャートのブロックに明示される機能を実装する命令手段を含む製造物を生成するようになっている。

【００６１】

従って、フローチャート図のブロックは、明示した機能を実行するための手段の組み合わせ、明示した機能を実行するためのステップの組み合わせ、および、明示した機能を実行するためのプログラムの組み合わせをサポートする。また、フローチャート図の各ブロックおよびフローチャート図のブロックの組み合わせは、明示した機能またはステップを実行する特殊目的のハードウェア・ベースのコンピュータ・システムによって、または、特殊目的のハードウェアおよびコンピュータ命令の組み合わせによって実装可能であることは理解されよう。

【００６２】

図５は、本発明の１つの例示的な実施形態に従った拡張メモリ間記述子を発生するための例示的なプロセスの概要を示すフローチャートである。図５に示すように、動作は、オペレーティング・システムの初期化によって開始する（ステップ５１０）。次いで、オペレーティング・システム・カーネル・サービスは、他の論理パーティションの遠隔メモリを表すノードを探してコンピュータ・システムのためのデバイス・ツリーをパースする（ステップ５２０）。次いで、これらの遠隔メモリ・ノードのための特性を検索し（ステップ５３０）、他のパーティションの遠隔メモリごとに、識別子、サイズ、および開始アドレスを求めるために用いる（ステップ５４０）。次いで、識別子、サイズ、および開始アドレスを、遠隔識別子と共にパッケージし、他の論理パーティションの各々の遠隔メモリごとの拡張メモリ間記述子とする（ステップ５５０）。次いで、これらの拡張メモリ間記述子を、遠隔メモリ・アクセスを実行する際に用いるためにストアする（ステップ５６０）。

【００６３】

図６は、本発明の１つの例示的な実施形態に従って拡張メモリ間記述子を用いるための例示的なプロセスの概要を示すフローチャートである。図６に示すように、動作は、遠隔

メモリ・アクセスを求める要求を受信することによって開始する（ステップ６１０）。次いで、この要求のターゲットである遠隔メモリのための拡張メモリ間記述子を検索する（ステップ６２０）。次いで、拡張メモリ間記述子を要求と共にＩ／Ｏスタックに渡す（ステップ６３０）。これは、拡張メモリ間記述子および要求をそれに応じて処理する。次いで、拡張メモリ間記述子を用いて、Ｉ／Ｏアダプタによって、遠隔メモリ・アクセスを実行する（ステップ６４０）。

【００６４】

従って、既知のメモリ間記述子を拡張して、現在プロセスが実行しているパーティションとは異なるパーティションの遠隔メモリを記述することができるようにするための機構が提供される。本発明の実施形態が提供する拡張は、既知のメモリ間記述子と同一のフォーマットを用いるが、データおよびデータに関連する含意は異なるので、コンピューティング・デバイスに存在するインタフェースおよびメモリ管理サービスの多くは、拡張メモリ間記述子を扱うために変更する必要はない。メモリ間記述子を調べる他のインタフェースおよびメモリ管理サービスは、拡張メモリ間記述子を処理するためのコードを含むように増強させて、遠隔メモリ・アクセスを実行するための機能を提供することができる。

【００６５】

本発明の例示的な実施形態について、典型的なＤＭＡ動作との関連で説明してきたが、本発明の実施形態はこれに限定されないことは認められよう。例えば、拡張メモリ間記述子は、遠隔コピー動作にも用いることができる。コピー動作は、ハイパーバイザ、または他のハードウェアもしくはソフトウェア機構またはその両方によって、容易にすることができる。この場合、サーバ・パーティションは、例えばハイパーバイザのような機構に対して、オペレーティング・システムとハイパーバイザとの間の詳細に記録されたインタフェースを介して情報を供給する。供給される情報は、拡張メモリ間記述子からのものである（例えば、クライアント・パーティションの遠隔メモリを識別する識別子）。次いで、ハイパーバイザは、サーバからクライアントに、またはクライアントからサーバに、実際のコピー動作を実行することができる。

【００６６】

動作は、サーバ側によって開始することができ、従って、サーバは、データをプルする（例えばｘｍｅｎｉｎシステム・コールによって）か、またはクライアント・パーティションにデータをプッシュする（例えばｘｍｅｍｏｕｔシステム・コールによって）ことができる。ハイパーバイザは、実際のコピー動作を実行することができる。なぜなら、その上で動作しているオペレーティング・システムよりも高い管理特権を有するからである。これによって、ハイパーバイザは、どのパーティションのどのメモリにもアクセスすることが可能となる。こうした遠隔コピー動作によって、サーバとクライアントとの間のいっそう洗練されたデータ交換プロトコルの設計および実装が容易になる。

【００６７】

本発明の実施形態は、完全に機能するデータ処理システムとの関連で説明してきたが、このプロセスは、命令のコンピュータ読み取り可能媒体の形態で、および、様々な形態で分配することができ、この分配を実行するために実際に用いる信号担持媒体の特定のタイプには無関係に、ここに記載した手法が等しく適用されることは、当業者には認められよう。コンピュータ読み取り可能媒体の例は、フレキシブル・ディスク、ハード・ディスク・ドライブ、ＲＡＭ、ＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ等の記録可能タイプの媒体、ならびに、デジタルおよびアナログ通信リンク、例えば無線周波数および光波伝送等の伝送形態を用いた有線または無線通信リンク等の伝送タイプの媒体を含む。コンピュータ読み取り可能媒体は、特定のデータ処理システムにおいて実際に用いるために復号される符号化フォーマットの形態を取ることができる。

【００６８】

ここでの記載は、例示および説明の目的のために提示するものであり、網羅的であったり、開示した形態に本発明を限定したりすることは意図していない。当業者には、多くの変更および変形が明らかであろう。実施形態は、本発明の原理、実際の適用を最良に説明

するため、および、考えられる特定の使用法に適した様々な変更を用いる様々な実施形態について当業者が本発明を理解することを可能とするために、選択して記載した。

【図面の簡単な説明】

【００６９】

【図１】本発明の実施形態を実装可能なデータ処理システムのブロック図である。

【図２】本発明の実施形態を実装可能である例示的な論理パーティション化プラットフォームのブロック図である。

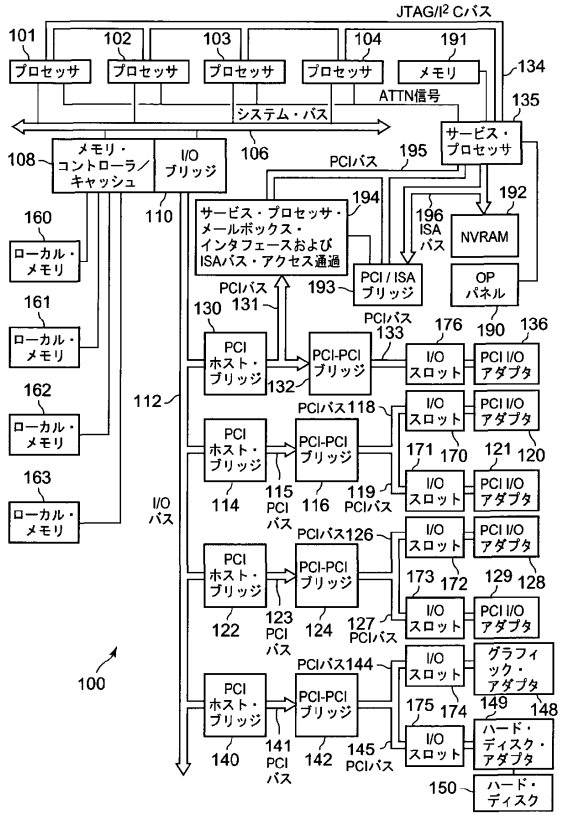
【図３】本発明の１つの例示的な実施形態に従って拡張メモリ間記述子を発生させるための機構の一例を示す例示的な図である。

【図４】拡張メモリ間記述子を用いて遠隔メモリ・アクセスを必要とする動作を実行するための、本発明の１つの例示的な実施形態に従った機構の一例を示す例示的な図である。

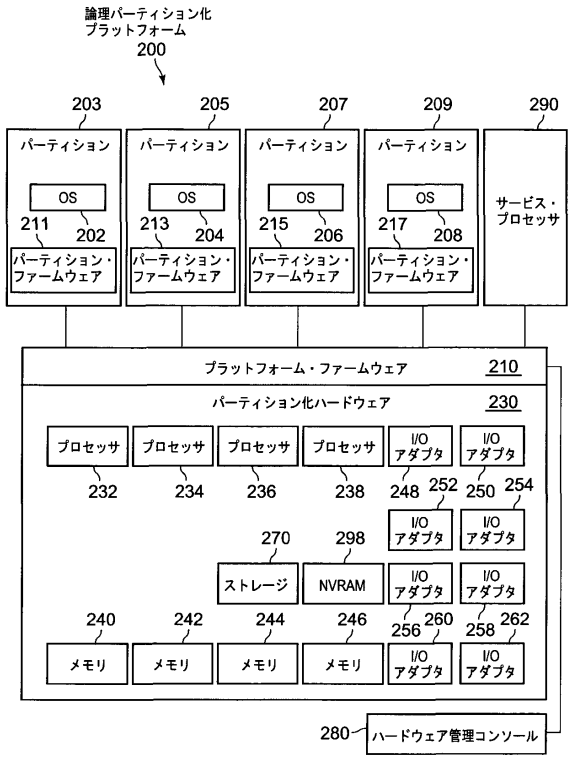
【図５】本発明の１つの例示的な実施形態に従って拡張メモリ間記述子を発生させるための例示的なプロセスの概要を示すフローチャートである。

【図６】本発明の１つの例示的な実施形態に従って拡張メモリ間記述子を用いるための例示的なプロセスの概要を示すフローチャートである。

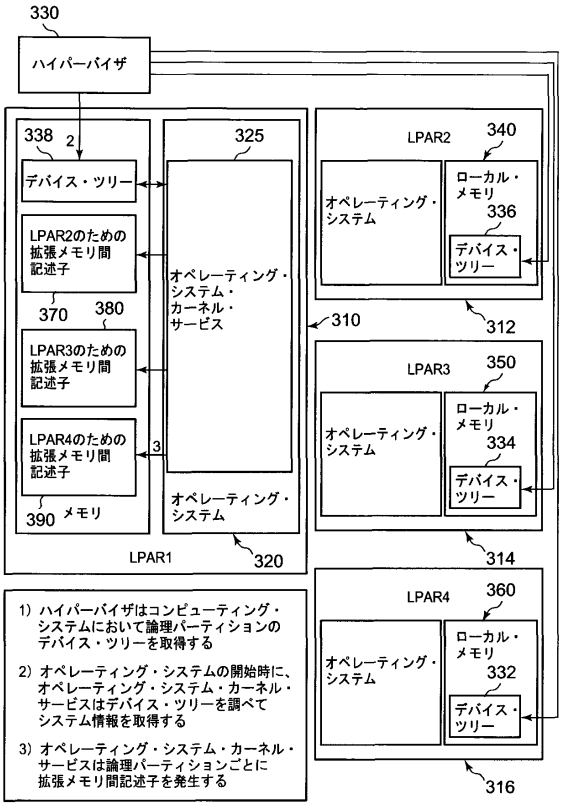
【図１】



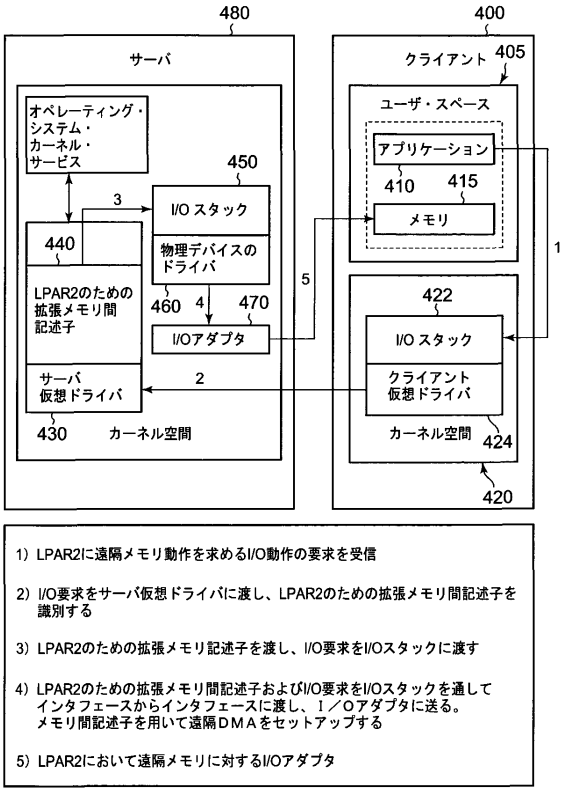
【図２】



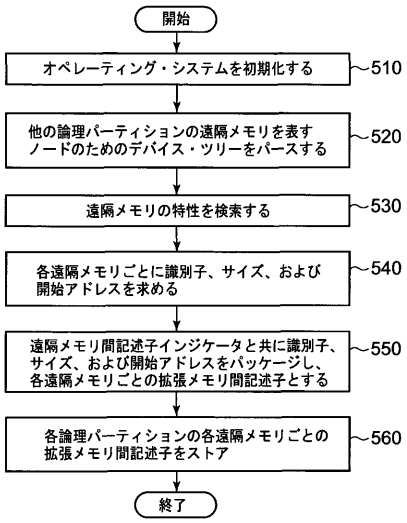
【図３】


330
ハイパーバイザ
338
2
デバイス・ツリー
LPAR2のための拡張メモリ間記述子
370
380
LPAR3のための拡張メモリ間記述子
LPAR4のための拡張メモリ間記述子
3
390
メモリ
325
オペレーティング・システム・カーネル・サービス
オペレーティング・システム
LPAR1
320
LPAR2
340
オペレーティング・システム
ローカル・メモリ
336
デバイス・ツリー
310
312
LPAR3
350
オペレーティング・システム
ローカル・メモリ
334
デバイス・ツリー
314
LPAR4
360
オペレーティング・システム
ローカル・メモリ
332
デバイス・ツリー
316
1) ハイパーバイザはコンピューティング・システムにおいて論理パーティションのデバイス・ツリーを取得する
2) オペレーティング・システムの開始時に、オペレーティング・システム・カーネル・サービスはデバイス・ツリーを調べてシステム情報を取得する
3) オペレーティング・システム・カーネル・サービスは論理パーティションごとに拡張メモリ間記述子を発生する

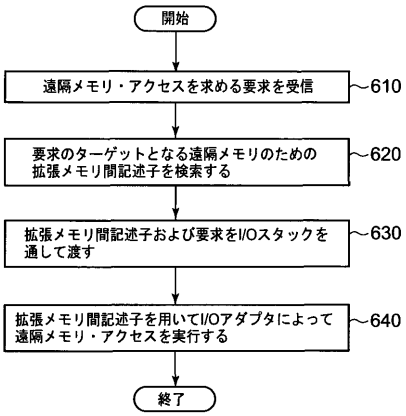
【図４】


480
サーバ
オペレーティング・システム・カーネル・サービス
450
3
I/O スタック
440
物理デバイスのドライバ
LPAR2のための拡張メモリ間記述子
460
4
470
I/Oアダプタ
5
サーバ仮想ドライバ
2
430
カーネル空間
400
クライアント
405
ユーザ・スペース
アプリケーション
410
415
メモリ
1
422
I/O スタック
クライアント仮想ドライバ
カーネル空間
424
420
1) LPAR2に遠隔メモリ動作を求めるI/O動作の要求を受信
2) I/O要求をサーバ仮想ドライバに渡し、LPAR2のための拡張メモリ間記述子を識別する
3) LPAR2のための拡張メモリ記述子を渡し、I/O要求をI/Oスタックに渡す
4) LPAR2のための拡張メモリ間記述子およびI/O要求をI/Oスタックを通してインタフェースからインタフェースに渡し、I／Oアダプタに送る。メモリ間記述子を用いて遠隔ＤＭＡをセットアップする
5) LPAR2において遠隔メモリに対するI/Oアダプタ

【図５】


開始
オペレーティング・システムを初期化する
510
他の論理パーティションの遠隔メモリを表すノードのためのデバイス・ツリーをパースする
520
遠隔メモリの特性を検索する
530
各遠隔メモリごとに識別子、サイズ、および開始アドレスを求める
540
遠隔メモリ間記述子インジケータと共に識別子、サイズ、および開始アドレスをパッケージし、各遠隔メモリごとの拡張メモリ間記述子とする
550
各論理パーティションの各遠隔メモリごとの拡張メモリ間記述子をストア
560
終了

【図６】


開始
遠隔メモリ・アクセスを求める要求を受信
610
要求のターゲットとなる遠隔メモリのための拡張メモリ間記述子を検索する
620
拡張メモリ間記述子および要求をI/Oスタックを通して渡す
630
拡張メモリ間記述子を用いてI/Oアダプタによって遠隔メモリ・アクセスを実行する
640
終了

フロントページの続き

(74)代理人 100086243
弁理士 坂口 博
(72)発明者 アスロット、ヴィシャル、チットランジャン
アメリカ合衆国７８７５８ テキサス州オースティン グレーシー・ファームズ・レーン２６００
アパートメント＃１３１２
(72)発明者 メアリー、ブルース
アメリカ合衆国７８７５９ テキサス州オースティン アミー・コート３９１４

審査官 鈴木 修治

(56)参考文献 特開２００２－０７３３５８（ＪＰ，Ａ）
特開２００２－１５７１３３（ＪＰ，Ａ）
特開２００４－００５４４３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46-9/54
G06F 12/10
G06F 13/28