(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6920281号
(P6920281)
(45)発行日　令和3年8月18日(2021.8.18)　(24)登録日　令和3年7月28日(2021.7.28)

(51)Int.Cl.　　F I
G06F 9/455 (2006.01)　　G06F 9/455 150
G06F 15/177 (2006.01)　　G06F 15/177 A

請求項の数 7　(全 41 頁)

(21)出願番号　特願2018-510841(P2018-510841)
(86)(22)出願日　平成29年1月26日(2017.1.26)
(65)公表番号　特表2019-509529(P2019-509529A)
(43)公表日　平成31年4月4日(2019.4.4)
(86)国際出願番号　PCT/US2017/015169
(87)国際公開番号　WO2017/132399
(87)国際公開日　平成29年8月3日(2017.8.3)
審査請求日　令和1年12月2日(2019.12.2)
(31)優先権主張番号　62/287,712
(32)優先日　平成28年1月27日(2016.1.27)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　15/415,668
(32)優先日　平成29年1月25日(2017.1.25)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　ヨンセン，ビョルン・ダグ
ノルウェー、０６８７　オスロ、ビルベルクグレンダ、９
(72)発明者　ホゥ，ハーラル
ノルウェー、１３２６　リュサケール、ピィ・オゥ・ボックス・３８４

最終頁に続く

(54)【発明の名称】高性能コンピューティング環境において仮想マシンの仮想マシンファブリックプロファイルを規定するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
仮想マシン（ＶＭ）の仮想マシンファブリックプロファイルを作成する方法であって、
プロセッサを含むノードが、仮想マシン識別子（ＶＭ－ｉｄ）を生成するステップと、
前記ノードが、仮想ホストチャネルアダプタ（ｖＨＣＡ）を特定するｖＨＣＡインスタンスＩＤを生成するステップと、
前記ノードが、グローバル一意識別子のプールからの仮想グローバル一意識別子（ｖＧＵＩＤ）を、前記ｖＨＣＡの仮想ポートの現在のファブリックアドレスとして割当てるステップと、
前記ノードが、管理パーティションを規定するＰ＿Ｋｅｙと前記ｖＧＵＩＤとの間の第１の関係を作成するステップとを含み、前記Ｐ＿Ｋｅｙと前記ｖＧＵＩＤとの間の前記第１の関係は、前記ｖＧＵＩＤが前記現在のファブリックアドレスとして割当てられている前記仮想ポートを、前記Ｐ＿Ｋｅｙによって規定される前記管理パーティションのメンバとして規定し、
前記ノードが、前記ＶＭ－ｉｄと前記ｖＨＣＡインスタンスＩＤとの間の第２の関係を作成するステップを含み、前記第２の関係により、前記ｖＨＣＡインスタンスＩＤを、前記ＶＭ－ｉｄへのアクセスを通して取出すことができ、
前記ノードが、前記ＶＭ－ｉｄと前記ｖＧＵＩＤとの間の第３の関係を作成するステップを含み、前記第３の関係により、前記ｖＧＵＩＤを、前記ＶＭ－ｉｄへのアクセスを通して取出すことができ、

前記ノードが、前記ＶＭ－ｉｄ、前記ｖＨＣＡインスタンスＩＤ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、および前記第３の関係を、前記ＶＭ－ｉｄ、前記ｖＨＣＡ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、および前記第３の関係を保存するフォーマットで、永続化するステップを含む、方法。

【請求項２】

前記ＶＭ－ｉｄ、前記ｖＨＣＡインスタンスＩＤ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、前記第３の関係、および前記Ｐ＿Ｋｅｙは、サブネットのサブネットマネージャのキャッシュに置かれ、前記Ｐ＿Ｋｅｙによって規定される前記管理パーティションは前記サブネット内に定められる、請求項１に記載の方法。

【請求項３】

前記キャッシュに置かれた前記ＶＭ－ｉｄ、前記ｖＨＣＡインスタンスＩＤ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、前記第３の関係、および前記Ｐ＿Ｋｅｙは、コピーである、請求項２に記載の方法。

【請求項４】

前記キャッシュは揮発性メモリである、請求項２または３に記載の方法。

【請求項５】

前記サブネットマネージャが、前記キャッシュから、前記ｖＨＣＡインスタンスＩＤおよび前記ｖＧＵＩＤを取出すステップと、

前記サブネットマネージャが、前記ｖＨＣＡインスタンスＩＤおよび前記ｖＧＵＩＤを物理ホストチャネルアダプタに送るステップとをさらに含む、請求項２～４のいずれか一項に記載の方法。

【請求項６】

仮想マシン（ＶＭ）の仮想マシンファブリックプロファイルを作成し前記仮想マシンファブリックプロファイルにアクセスするためのシステムであって、

プロセッサを含むノードと、

前記ノードにアクセス可能なメモリとを備え、

前記ノードは、

仮想マシン識別子（ＶＭ－ｉｄ）を生成するように動作し、

仮想ホストチャネルアダプタ（ｖＨＣＡ）を特定するｖＨＣＡインスタンスＩＤを生成するように動作し、

グローバル一意識別子のプールからの仮想グローバル一意識別子（ｖＧＵＩＤ）を、前記ｖＨＣＡの仮想ポートの現在のファブリックアドレスとして割当てるように動作し、

管理パーティションを規定するＰ＿Ｋｅｙと前記ｖＧＵＩＤとの間の第１の関係を作成するように動作し、前記Ｐ＿Ｋｅｙと前記ｖＧＵＩＤとの間の前記第１の関係は、前記ｖＧＵＩＤが前記現在のファブリックアドレスとして割当てられている前記仮想ポートを、前記Ｐ＿Ｋｅｙによって規定される前記管理パーティションのメンバとして規定し、前記ノードはさらに、

前記ＶＭ－ｉｄと前記ｖＨＣＡインスタンスＩＤとの間の第２の関係を作成するように動作し、前記第２の関係により、前記ｖＨＣＡインスタンスＩＤを、前記ＶＭ－ｉｄへのアクセスを通して取出すことができ、前記ノードはさらに、

前記ＶＭ－ｉｄと前記ｖＧＵＩＤとの間の第３の関係を作成するように動作し、前記第３の関係により、前記ｖＧＵＩＤを、前記ＶＭ－ｉｄへのアクセスを通して取出すことができ、前記ノードはさらに、

前記ＶＭ－ｉｄ、前記ｖＨＣＡインスタンスＩＤ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、および前記第３の関係を、前記ＶＭ－ｉｄ、前記ｖＨＣＡ、前記ｖＧＵＩＤ、前記第１の関係、前記第２の関係、および前記第３の関係を保存するフォーマットで、永続化するように動作する、システム。

【請求項７】

コンピュータシステムに請求項１～５のいずれか一項に記載の方法を実行させるためのコンピュータプログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
著作権に関する注意
本特許文献の開示の一部には、著作権保護の対象となるものが含まれている。著作権者は、この特許文献または特許開示の何者かによる複製が、特許商標庁の特許ファイルまたは記録にある限り、それに対して異議を唱えないが、そうでなければ、いかなる場合もすべての著作権を留保する。
【０００２】
発明の分野
本発明は、概してコンピュータシステムに関し、具体的には仮想マシンの仮想マシンファブリックプロファイルを規定することに関する。
【背景技術】
【０００３】
背景
導入されるクラウドコンピューティングアーキテクチャがより大規模になるのに応じて、従来のネットワークおよびストレージに関する性能および管理の障害が深刻な問題になってきている。クラウドコンピューティングファブリックのための基礎としてインフィニバンド^{TM}（InfiniBand：ＩＢ）技術などの高性能な無損失相互接続を用いることへの関心がますます高まってきている。これは、本発明の実施形態が対応するように意図された一般領域である。
【発明の概要】
【課題を解決するための手段】
【０００４】
概要
本明細書に記載されているのは、仮想マシンの仮想マシンファブリックプロファイルを規定するためのシステムおよび方法である。ある例示的な実施形態は、仮想マシン識別子、仮想ホストチャネルアダプタインスタンスＩＤ、および仮想グローバル一意識別子を提供することができる。仮想マシン識別子、仮想ホストチャネルアダプタインスタンスＩＤ、および仮想グローバル一意識別子は、仮想マシン識別子にアクセスすることにより、仮想ホストチャネルアダプタインスタンスＩＤおよび仮想グローバル一意識別子を取出すことができるように、相互にマッピングすることができる。さらに、管理パーティションを規定するＰ＿Ｋｅｙと仮想グローバル一意識別子との間の関係を作成することができ、上記Ｐ＿Ｋｅｙと仮想グローバル一意識別子との間の関係は、仮想グローバル一意識別子を、Ｐ＿Ｋｅｙによって規定される管理パーティションのメンバとして規定する。
【図面の簡単な説明】
【０００５】
【図１】一実施形態に従うインフィニバンド^{TM}環境の一例を示す図である。
【図２】一実施形態に従う分割されたクラスタ環境の一例を示す図である。
【図３】一実施形態に従うネットワーク環境におけるツリートポロジーの一例を示す図である。
【図４】一実施形態に従う例示的な共有ポートアーキテクチャを示す図である。
【図５】一実施形態に従う例示的なｖＳｗｉｔｃｈアーキテクチャを示す図である。
【図６】一実施形態に従う例示的なｖＰｏｒｔアーキテクチャを示す図である。
【図７】一実施形態に従うＬＩＤが予めポピュレートされている例示的なｖＳｗｉｔｃｈアーキテクチャを示す図である。
【図８】一実施形態に従う動的ＬＩＤ割当てがなされた例示的なｖＳｗｉｔｃｈアーキテクチャを示す図である。
【図９】一実施形態に従う、ｖＳｗｉｔｃｈに動的ＬＩＤ割当てがなされかつＬＩＤが予めポピュレートされている、例示的なｖＳｗｉｔｃｈアーキテクチャを示す図である。

【図１０】一実施形態に従う例示的なマルチサブネットインフィニバンド^{TM}ファブリックを示す図である。
【図１１】一実施形態に従う、例示的な物理および論理サブネットリソースを含む例示的なインフィニバンド^{TM}ファブリックおよびサブネットを示す図である。
【図１２】一実施形態に従う、異なる管理パーティションのメンバとしての例示的なサブネットリソースを含む例示的なインフィニバンド^{TM}ファブリックおよびサブネットを示す図である。
【図１３】一実施形態に従う、管理パーティションおよびリソースドメイン双方を含む、階層管理スキームのメンバとしての例示的なサブネットリソースを含む例示的なインフィニバンド^{TM}ファブリックおよびサブネットを示す図である。
【図１４】一実施形態に従う、ファブリックレベルリソースドメインに対応付けられた管理パーティションのメンバおよび対応付けられたリソースに対して相互アクセス権を割当てるための方法のフローチャートを示す図である。
【図１５】一実施形態に従う、ＶＭファブリックプロファイル情報を格納するための例示的なデータベース構造を示す図である。
【図１６】一実施形態に従う、ＶＭファブリックプロファイルをサブネットリソースが利用できるようにするためのフローチャートを示す図である。
【図１７】一実施形態に従う、仮想マシンの仮想マシンファブリックプロファイルを作成するためのフローチャートを示す図である。
【発明を実施するための形態】
【０００６】
詳細な説明
本発明は、同様の参照番号が同様の要素を指している添付図面の図において、限定のためではなく例示のために説明されている。なお、この開示における「ある」または「１つの」または「いくつかの」実施形態への参照は必ずしも同じ実施形態に対するものではなく、そのような参照は少なくとも１つを意味する。特定の実現例が説明されるが、これらの特定の実現例が例示的な目的のためにのみ提供されることが理解される。当業者であれば、他の構成要素および構成が、この発明の範囲および精神から逸脱することなく使用され得ることを認識するであろう。
【０００７】
図面および詳細な説明全体にわたって同様の要素を示すために、共通の参照番号が使用され得る。したがって、ある図で使用される参照番号は、要素が別のところで説明される場合、そのような図に特有の詳細な説明において参照される場合もあり、または参照されない場合もある。
【０００８】
本明細書に記載されているのは、仮想マシンの仮想マシンファブリックプロファイルを規定するためのシステムおよび方法である。
【０００９】
この発明の以下の説明は、高性能ネットワークの一例として、インフィニバンド^{TM}（ＩＢ）ネットワークを使用する。以下の説明全体にわたり、インフィニバンド^{TM}の仕様（インフィニバンド仕様、ＩＢ仕様、またはレガシーＩＢ仕様など、さまざまな呼ばれ方がある）を引用することがある。このような引用は、２０１５年３月に発表され、http://www.inifinibandta.orgから入手可能な、本明細書にその全体を引用により援用するInfiniBand Trade Association Architecture Specification, Volume 1, Version 1.3を引用することであると理解される。他のタイプの高性能ネットワークが何ら限定されることなく使用され得ることが、当業者には明らかであるだろう。以下の説明ではまた、ファブリックトポロジーについての一例として、ファットツリートポロジーを使用する。他のタイプのファブリックトポロジーが何ら限定されることなく使用され得ることが当業者には明らかであるだろう。
【００１０】

インフィニバンド^{TM}

インフィニバンド^{TM}（ＩＢ）は、インフィニバンド^{TM}・トレード・アソシエーション（InfiniBand^{TM} Trade Association）によって開発されたオープン標準無損失ネットワーク技術である。この技術は、特に高性能コンピューティング（high-performance computing：ＨＰＣ）アプリケーションおよびデータセンタを対象とする、高スループットおよび少ない待ち時間の通信を提供するシリアルポイントツーポイント全二重相互接続（serial point-to-point full-duplex interconnect）に基づいている。

【００１１】

インフィニバンド^{TM}・アーキテクチャ（InfiniBand Architecture：ＩＢＡ）は、２層トポロジー分割をサポートする。低層では、ＩＢネットワークはサブネットと呼ばれ、１つのサブネットは、スイッチおよびポイントツーポイントリンクを使用して相互接続される一組のホストを含み得る。より高いレベルでは、１つのＩＢファブリックは、ルータを使用して相互接続され得る１つ以上のサブネットを構成する。

【００１２】

１つのサブネット内で、ホストは、スイッチおよびポイントツーポイントリンクを使用して接続され得る。加えて、サブネットにおける指定されたデバイス上に存在する、１つのマスター管理エンティティ、すなわちサブネットマネージャ（subnet manager：ＳＭ）があり得る。サブネットマネージャは、ＩＢサブネットを構成し、起動し、維持する役割を果たす。加えて、サブネットマネージャ（ＳＭ）は、ＩＢファブリックにおいてルーティングテーブル計算を行なう役割を果たし得る。ここで、たとえば、ＩＢネットワークのルーティングは、ローカルサブネットにおけるすべての送信元と宛先とのペア間の適正な負荷バランシングを目標とする。

【００１３】

サブネット管理インターフェイスを通して、サブネットマネージャは、サブネット管理パケット（subnet management packet：ＳＭＰ）と呼ばれる制御パケットを、サブネット管理エージェント（subnet management agent：ＳＭＡ）と交換する。サブネット管理エージェントは、すべてのＩＢサブネットデバイス上に存在する。ＳＭＰを使用することにより、サブネットマネージャは、ファブリックを発見し、エンドノードおよびスイッチを構成し、ＳＭＡから通知を受信することができる。

【００１４】

一実施形態に従うと、ＩＢネットワークにおけるサブネット内のルーティングは、スイッチに格納されたリニアフォワーディングテーブル（linear forwarding table）（ＬＦＴ）に基づき得る。ＬＦＴは、使用中のルーティングメカニズムに従って、ＳＭによって計算される。サブネットでは、エンドノード上のホストチャネルアダプタ（Host Channel Adapter：ＨＣＡ）ポートおよびスイッチが、ローカル識別子（ＬＩＤ）を使用してアドレス指定される。ＬＦＴにおける各エントリは、宛先ＬＩＤ（destination LID：ＤＬＩＤ）と出力ポートとからなる。テーブルにおけるＬＩＤごとに１つのエントリのみがサポートされる。パケットがあるスイッチに到着すると、その出力ポートは、そのスイッチのフォワーディングテーブルにおいてＤＬＩＤを検索することによって判断される。所与の送信元－宛先ペア（ＬＩＤペア）間のネットワークにおいてパケットは同じ経路を通るため、ルーティングは決定論的である。

【００１５】

一般に、マスタサブネットマネージャを除く他のすべてのサブネットマネージャは、耐故障性のために待機モードで作動する。しかしながら、マスタサブネットマネージャが故障した状況では、待機中のサブネットマネージャによって、新しいマスタサブネットマネージャが取り決められる。マスタサブネットマネージャはまた、サブネットの周期的なスイープ（sweep）を行なってあらゆるトポロジー変化を検出し、それに応じてネットワークを再構成する。

【００１６】

さらに、サブネット内のホストおよびスイッチは、ローカル識別子（ＬＩＤ）を用いて

アドレス指定され得るとともに、単一のサブネットは４９１５１個のユニキャストＬＩＤに制限され得る。サブネット内で有効なローカルアドレスであるＬＩＤの他に、各ＩＢデバイスは、６４ビットのグローバル一意識別子（global unique identifier：ＧＵＩＤ）を有し得る。ＧＵＩＤは、ＩＢレイヤー３（Ｌ３）アドレスであるグローバル識別子（global identifier：ＧＩＤ）を形成するために使用され得る。

【００１７】

ＳＭは、ネットワーク初期化時間に、ルーティングテーブル（すなわち、サブネット内のノードの各ペア間の接続／ルート）を計算し得る。さらに、トポロジーが変化するたびに、ルーティングテーブルは、接続性および最適性能を確実にするために更新され得る。通常動作中、ＳＭは、トポロジー変化をチェックするためにネットワークの周期的なライトスイープ（light sweep）を実行し得る。ライトスイープ中に変化が発見された場合、または、ネットワーク変化を信号で伝えるメッセージ（トラップ）をＳＭが受信した場合、ＳＭは、発見された変化に従ってネットワークを再構成し得る。

【００１８】

たとえば、ＳＭは、リンクがダウンした場合、デバイスが追加された場合、またはリンクが除去された場合など、ネットワークトポロジーが変化する場合に、ネットワークを再構成し得る。再構成ステップは、ネットワーク初期化中に行なわれるステップを含み得る。さらに、再構成は、ネットワーク変化が生じたサブネットに制限されるローカルスコープを有し得る。また、ルータを用いる大規模ファブリックのセグメント化は、再構成スコープを制限し得る。

【００１９】

一実施形態に従ったインフィニバンド^{TM}環境１００の一例を示す図１に、インフィニバンド^{TM}ファブリックの一例を示す。図１に示す例では、ノードＡ１０１～Ｅ１０５は、インフィニバンド^{TM}ファブリック１２０を使用して、それぞれのホストチャネルアダプタ１１１～１１５を介して通信する。一実施形態に従うと、さまざまなノード（たとえばノードＡ１０１～Ｅ１０５）はさまざまな物理デバイスによって表わすことができる。一実施形態に従うと、さまざまなノード（たとえばノードＡ１０１～Ｅ１０５）は仮想マシンなどのさまざまな仮想デバイスによって表わすことができる。

【００２０】

インフィニバンド^{TM}におけるデータパーティション

一実施形態に従うと、ＩＢネットワークは、ネットワークファブリックを共有するシステムの論理グループを分離するためのセキュリティメカニズムとしてのパーティショニングをサポートし得る。ファブリックにおけるノード上の各ＨＣＡポートは、１つ以上のパーティションのメンバである可能性がある。一実施形態に従うと、本開示は、ＩＢサブネット内に規定することができる２種類のパーティションとして、データパーティション（以下の段落でより詳細に説明する）と管理パーティション（本開示において後に詳細に説明する）とを提供する。

【００２１】

データパーティションメンバーシップは、ＳＭの一部であり得る集中型パーティションマネージャによって管理される。ＳＭは、各ポートに関するデータパーティションメンバーシップ情報を、１６ビットのパーティションキー（partition key：Ｐ＿Ｋｅｙ）のテーブルとして構成することができる。ＳＭはまた、これらのポートを介してデータトラフィックを送信または受信するエンドノードに関連付けられたＰ＿Ｋｅｙ情報を含むデータパーティション実施テーブルを用いて、スイッチポートおよびルータポートを構成することができる。加えて、一般的な場合には、スイッチポートのデータパーティションメンバーシップは、（リンクに向かう）出口方向に向かってポートを介してルーティングされたＬＩＤに間接的に関連付けられたすべてのメンバーシップの集合を表わし得る。

【００２２】

一実施形態に従うと、データパーティションはポートの論理グループであり、あるグループのメンバは同じ論理グループの他のメンバとしか通信できない。ホストチャネルアダ

プタ（ＨＣＡ）およびスイッチにおいて、データパーティションメンバーシップ情報を用いてパケットをフィルタリングすることにより、分離を実施することができる。無効なパーティショニング情報を有するパケットは、当該パケットが入口ポートに達すると直ちにドロップすることができる。パーティショニングされたＩＢシステムにおいて、データパーティションを用いることにより、テナントクラスタを作成できる。データパーティションを適所で実施すると、ノードは異なるテナントクラスタに属する他のノードと通信することができない。このようにして、欠陥があるまたは悪意があるテナントノードが存在していても、システムのセキュリティを保証することができる。

【００２３】

一実施形態に従うと、ノード間の通信のために、マネージメントキューペア（ＱＰ０およびＱＰ１）を除き、キューペア（Queue Pair：ＱＰ）およびエンドツーエンドコンテキスト（End-to-End context：ＥＥＣ）を特定のデータパーティションに割当てることができる。次に、Ｐ＿Ｋｅｙ情報を、送信されたすべてのＩＢトランスポートパケットに追加することができる。パケットがＨＣＡポートまたはスイッチに到着すると、そのＰ＿Ｋｅｙ値を、ＳＭによって構成されたテーブルに対して確認することができる。無効のＰ＿Ｋｅｙ値が見つかった場合、そのパケットは直ちに廃棄される。このようにして、通信は、データパーティションを共有するポート間でのみ許可される。

【００２４】

一実施形態に従い、データがパーティショニングされたクラスタ環境の一例を示す図２に、ＩＢパーティションの一例が示される。図２に示す例では、ノードＡ１０１～Ｅ１０５は、インフィニバンド™ファブリック１２０を使用して、それぞれのホストチャネルアダプタ１１１～１１５を介して通信する。ノードＡ～Ｅは、データパーティション、すなわち、データパーティション１　１３０、データパーティション２　１４０、およびデータパーティション３　１５０に配置されている。データパーティション１はノードＡ　１０１とノードＤ　１０４とを含む。データパーティション２はノードＡ　１０１とノードＢ　１０２とノードＣ　１０３とを含む。データパーティション３はノードＣ　１０３とノードＥ　１０５とを含む。データパーティションのこの配置により、ノードＤ　１０４およびノードＥ　１０５は、１つのデータパーティションを共有していないので、通信することができない。一方、たとえばノードＡ　１０１およびノードＣ　１０３は、どちらもデータパーティション２　１４０のメンバなので、通信することができる。

【００２５】

インフィニバンド™における仮想マシン

過去１０年の間に、ハードウェア仮想化サポートによってＣＰＵオーバーヘッドが実質的に排除され、メモリ管理ユニットを仮想化することによってメモリオーバーヘッドが著しく削減され、高速ＳＡＮストレージまたは分散型ネットワークファイルシステムの利用によってストレージオーバーヘッドが削減され、シングルルートＩ／Ｏ仮想化（Single Root Input/Output Virtualization：ＳＲ－ＩＯＶ）のようなデバイス・パススルー技術を使用することによってネットワークＩ／Ｏオーバーヘッドが削減されてきたことに応じて、仮想化された高性能コンピューティング（High Performance Computing：ＨＰＣ）環境の将来見通しが大幅に改善されてきた。現在では、クラウドが、高性能相互接続ソリューションを用いて仮想ＨＰＣ（virtual HPC：ｖＨＰＣ）クラスタに対応し、必要な性能を提供することができる。

【００２６】

しかしながら、インフィニバンド™（ＩＢ）などの無損失ネットワークと連結されたとき、仮想マシン（ＶＭ）のライブマイグレーションなどのいくつかのクラウド機能は、これらのソリューションにおいて用いられる複雑なアドレス指定およびルーティングスキームのせいで、依然として問題となる。ＩＢは、高帯域および低レイテンシを提供する相互接続ネットワーク技術であり、このため、ＨＰＣおよび他の通信集約型の作業負荷に非常によく適している。

【００２７】

ＩＢデバイスをＶＭに接続するための従来のアプローチは直接割当てされたＳＲ－ＩＯＶを利用することによるものである。しかしながら、ＳＲ－ＩＯＶを用いてＩＢホストチャネルアダプタ（ＨＣＡ）に割当てられたＶＭのライブマイグレーションを実現することは難易度の高いものであることが判明した。各々のＩＢが接続されているノードは、３つの異なるアドレス（すなわちＬＩＤ、ＧＵＩＤおよびＧＩＤ）を有する。ライブマイグレーションが発生すると、これらのアドレスのうち１つ以上が変化する。マイグレーション中のＶＭ（VM-in-migration）と通信する他のノードは接続性を失う可能性がある。これが発生すると、ＩＢサブネットマネージャ（Subnet Manager：ＳＭ）にサブネット管理（Subnet Administration：ＳＡ）経路記録クエリを送信することによって、再接続すべき仮想マシンの新しいアドレスを突きとめることにより、失われた接続を回復させるように試みることができる。

【００２８】

ＩＢは３つの異なるタイプのアドレスを用いる。第１のタイプのアドレスは１６ビットのローカル識別子（ＬＩＤ）である。少なくとも１つの固有のＬＩＤは、ＳＭによって各々のＨＣＡポートおよび各々のスイッチに割当てられる。ＬＩＤはサブネット内のトラフィックをルーティングために用いられる。ＬＩＤが１６ビット長であるので、６５５３６個の固有のアドレス組合せを構成することができ、そのうち４９１５１個（０×０００１－０×ＢＦＦＦ）だけをユニキャストアドレスとして用いることができる。結果として、入手可能なユニキャストアドレスの数は、ＩＢサブネットの最大サイズを定義することとなる。第２のタイプのアドレスは、製造業者によって各々のデバイス（たとえば、ＨＣＡおよびスイッチ）ならびに各々のＨＣＡポートに割当てられた６４ビットのグローバル一意識別子（ＧＵＩＤ）である。ＳＭは、ＨＣＡポートに追加のサブネット固有ＧＵＩＤを割当ててもよく、これは、ＳＲ－ＩＯＶが用いられる場合に有用となる。第３のタイプのアドレスは１２８ビットのグローバル識別子（ＧＩＤ）である。ＧＩＤは有効なＩＰｖ６ユニキャストアドレスであり、少なくとも１つが各々のＨＣＡポートに割当てられている。ＧＩＤは、ファブリックアドミニストレータによって割当てられたグローバルに固有の６４ビットプレフィックスと各々のＨＣＡポートのＧＵＩＤアドレスとを組合わせることによって形成される。

【００２９】

ファットツリー（Fat Tree：ＦＴｒｅｅ）トポロジーおよびルーティング

一実施形態に従うと、ＩＢベースのＨＰＣシステムのいくつかは、ファットツリートポロジーを採用して、ファットツリーが提供する有用な特性を利用する。これらの特性は、各送信元宛先ペア間の複数経路の利用可能性に起因する、フルバイセクション帯域幅および固有の耐故障性を含む。ファットツリーの背後にある初期の概念は、ツリーがトポロジーのルート（root）に近づくにつれて、より利用可能な帯域幅を用いて、ノード間のより太いリンクを採用することであった。より太いリンクは、上位レベルのスイッチにおける輻輳を回避するのに役立てることができ、バイセクション帯域幅が維持される。

【００３０】

図３は、一実施形態に従った、ネットワーク環境におけるツリートポロジーの例を示す。図３に示すように、ネットワークファブリック２００において、１つ以上のエンドノード２０１～２０４が接続され得る。ネットワークファブリック２００は、複数のリーフスイッチ２１１～２１４と複数のスパインスイッチまたはルート（root）スイッチ２３１～２３４とを含むファットツリートポロジーに基づき得る。加えて、ネットワークファブリック２００は、スイッチ２２１～２２４などの１つ以上の中間スイッチを含み得る。

【００３１】

また、図３に示すように、エンドノード２０１～２０４の各々は、マルチホームノード、すなわち、複数のポートを介してネットワークファブリック２００のうち２つ以上の部分に接続される単一のノードであり得る。たとえば、ノード２０１はポートＨ１およびＨ２を含み、ノード２０２はポートＨ３およびＨ４を含み、ノード２０３はポートＨ５およびＨ６を含み、ノード２０４はポートＨ７およびＨ８を含み得る。

【００３２】
加えて、各スイッチは複数のスイッチポートを有し得る。たとえば、ルートスイッチ２３１はスイッチポート１～２を有し、ルートスイッチ２３２はスイッチポート３～４を有し、ルートスイッチ２３３はスイッチポート５～６を有し、ルートスイッチ２３４はスイッチポート７～８を有し得る。
【００３３】
一実施形態に従うと、ファットツリールーティングメカニズムは、ＩＢベースのファットツリートポロジーに関して最も人気のあるルーティングアルゴリズムのうちの１つである。ファットツリールーティングメカニズムはまた、ＯＦＥＤ（Open Fabric Enterprise Distribution：ＩＢベースのアプリケーションを構築しデプロイするための標準ソフトウェアスタック）サブネットマネージャ、すなわちＯｐｅｎＳＭにおいて実現される。
【００３４】
ファットツリールーティングメカニズムの目的は、ネットワークファブリックにおけるリンクにわたって最短経路ルートを均一に広げるＬＦＴを生成することである。このメカニズムは、索引付け順序でファブリックを横断し、エンドノードの目標ＬＩＤ、ひいては対応するルートを各スイッチポートに割当てる。同じリーフスイッチに接続されたエンドノードについては、索引付け順序は、エンドノードが接続されるスイッチポートに依存し得る（すなわち、ポートナンバリングシーケンス）。各ポートについては、メカニズムはポート使用カウンタを維持することができ、新しいルートが追加されるたびに、ポート使用カウンタを使用して使用頻度が最小のポートを選択することができる。
【００３５】
一実施形態に従うと、パーティショニングされたサブネットでは、共通のデータパーティションのメンバではないノードは通信することを許可されない。実際には、これは、ファットツリールーティングアルゴリズムによって割当てられたルートのうちのいくつかがユーザトラフィックのために使用されないことを意味する。ファットツリールーティングメカニズムが、それらのルートについてのＬＦＴを、他の機能的経路と同じやり方で生成する場合、問題が生じる。この動作は、リンク上でバランシングを劣化させるおそれがある。なぜなら、ノードが索引付けの順序でルーティングされているからである。データパーティションに気づかずにルーティングが行なわれるため、ファットツリーでルーティングされたサブネットにより、概して、データパーティション間の分離が不良なものとなる。
【００３６】
一実施形態に従うと、ファットツリーは、利用可能なネットワークリソースでスケーリングすることができる階層ネットワークトポロジーである。さらに、ファットツリーは、さまざまなレベルの階層に配置された商品スイッチを用いて容易に構築される。さらに、ｋ－ａｒｙ－ｎ－ｔｒｅｅ、拡張された一般化ファットツリー（Extended Generalized Fat-Tree：ＸＧＦＴ）、パラレルポート一般化ファットツリー（Parallel Ports Generalized Fat-Tree：ＰＧＦＴ）およびリアルライフファットツリー（Real Life Fat-Tree：ＲＬＦＴ）を含むファットツリーのさまざまな変形例が、一般に利用可能である。
【００３７】
また、ｋ－ａｒｙ－ｎ－ｔｒｅｅは、ｎレベルのファットツリーであって、k^nエンドノードと、$n \cdot k^{n-1}$スイッチとを備え、各々が２ｋポートを備えている。各々のスイッチは、ツリーにおいて上下方向に同数の接続を有している。ＸＧＦＴファットツリーは、スイッチのための異なる数の上下方向の接続と、ツリーにおける各レベルでの異なる数の接続とをともに可能にすることによって、ｋ－ａｒｙ－ｎ－ｔｒｅｅを拡張させる。ＰＧＦＴ定義はさらに、ＸＧＦＴトポロジーを拡張して、スイッチ間の複数の接続を可能にする。多種多様なトポロジーはＸＧＦＴおよびＰＧＦＴを用いて定義することができる。しかしながら、実用化するために、現代のＨＰＣクラスタにおいて一般に見出されるファットツリーを定義するために、ＰＧＦＴの制限バージョンであるＲＬＦＴが導入されている。ＲＬＦＴは、ファットツリーにおけるすべてのレベルに同じポートカウントスイッチ

を用いている。

【００３８】

入出力（Ｉ／Ｏ）仮想化

一実施形態に従うと、Ｉ／Ｏ仮想化（I/O Virtualization：ＩＯＶ）は、基礎をなす物理リソースに仮想マシン（ＶＭ）がアクセスすることを可能にすることによって、Ｉ／Ｏを利用可能にすることができる。ストレージトラフィックとサーバ間通信とを組合わせると、シングルサーバのＩ／Ｏリソースにとって抗し難い高い負荷が課され、結果として、データの待機中に、バックログが発生し、プロセッサがアイドル状態になる可能性がある。Ｉ／Ｏ要求の数が増えるにつれて、ＩＯＶにより利用可能性をもたらすことができ、最新のＣＰＵ仮想化において見られる性能レベルに匹敵するように、（仮想化された）Ｉ／Ｏリソースの性能、スケーラビリティおよび融通性を向上させることができる。

【００３９】

一実施形態に従うと、Ｉ／Ｏリソースの共有を可能にして、ＶＭからリソースへのアクセスが保護されることを可能にし得るようなＩＯＶが所望される。ＩＯＶは、ＶＭにエクスポーズされる論理装置を、その物理的な実装から分離する。現在、エミュレーション、準仮想化、直接的な割当て（direct assignment：ＤＡ）、およびシングルルートＩ／Ｏ仮想化（ＳＲ－ＩＯＶ）などのさまざまなタイプのＩＯＶ技術が存在し得る。

【００４０】

一実施形態に従うと、あるタイプのＩＯＶ技術としてソフトウェアエミュレーションがある。ソフトウェアエミュレーションは分離されたフロントエンド／バックエンド・ソフトウェアアーキテクチャを可能にし得る。フロントエンドはＶＭに配置されたデバイスドライバであり得、Ｉ／Ｏアクセスをもたらすためにハイパーバイザによって実現されるバックエンドと通信し得る。物理デバイス共有比率は高く、ＶＭのライブマイグレーションはネットワークダウンタイムのわずか数ミリ秒で実現可能である。しかしながら、ソフトウェアエミュレーションはさらなる不所望な計算上のオーバーヘッドをもたらしてしまう。

【００４１】

一実施形態に従うと、別のタイプのＩＯＶ技術として直接的なデバイスの割当てがある。直接的なデバイスの割当てでは、Ｉ／ＯデバイスをＶＭに連結する必要があるが、デバイスはＶＭ間では共有されない。直接的な割当てまたはデバイス・パススルーは、最小限のオーバーヘッドでほぼ固有の性能を提供する。物理デバイスはハイパーバイザをバイパスし、直接、ＶＭに取付けられている。しかしながら、このような直接的なデバイスの割当ての欠点は、仮想マシン間で共有がなされないため、１枚の物理ネットワークカードが１つのＶＭと連結されるといったように、スケーラビリティが制限されてしまうことである。

【００４２】

一実施形態に従うと、シングルルートＩＯＶ（Single Root IOV：ＳＲ－ＩＯＶ）は、ハードウェア仮想化によって、物理装置がその同じ装置の複数の独立した軽量のインスタンスとして現われることを可能にし得る。これらのインスタンスは、パススルー装置としてＶＭに割当てることができ、仮想機能（Virtual Function：ＶＦ）としてアクセスすることができる。ハイパーバイザは、（１つのデバイスごとに）固有の、十分な機能を有する物理機能（Physical Function：ＰＦ）によってデバイスにアクセスする。ＳＲ－ＩＯＶは、純粋に直接的に割当てする際のスケーラビリティの問題を軽減する。しかしながら、ＳＲ－ＩＯＶによって提示される問題は、それがＶＭマイグレーションを損なう可能性があることである。これらのＩＯＶ技術の中でも、ＳＲ－ＩＯＶは、ほぼ固有の性能を維持しながらも、複数のＶＭから単一の物理デバイスに直接アクセスすることを可能にする手段を用いてＰＣＩ　Ｅｘｐｒｅｓｓ（ＰＣＩｅ）規格を拡張することができる。これにより、ＳＲ－ＩＯＶは優れた性能およびスケーラビリティを提供することができる。

【００４３】

ＳＲ－ＩＯＶは、ＰＣＩｅデバイスが、各々のゲストに１つの仮想デバイスを割当てる

ことによって複数のゲスト間で共有することができる複数の仮想デバイスをエクスポーズすることを可能にする。各々のＳＲ－ＩＯＶデバイスは、少なくとも１つの物理機能（ＰＦ）と、１つ以上の関連付けられた仮想機能（ＶＦ）とを有する。ＰＦは、仮想マシンモニタ（virtual machine monitor：ＶＭＭ）またはハイパーバイザによって制御される通常のＰＣＩｅ機能であるのに対して、ＶＦは軽量のＰＣＩｅ機能である。各々のＶＦはそれ自体のベースアドレス（base address：ＢＡＲ）を有しており、固有のリクエスタＩＤが割当てられている。固有のリクエスタＩＤは、Ｉ／Ｏメモリ管理ユニット（I/O memory management unit：ＩＯＭＭＵ）がさまざまなＶＦへの／からのトラフィックストリームを区別することを可能にする。ＩＯＭＭＵはまた、メモリを適用して、ＰＦとＶＦとの間の変換を中断する。

【００４４】

しかし、残念ながら、直接的デバイス割当て技術は、仮想マシンのトランスペアレントなライブマイグレーションがデータセンタ最適化のために所望されるような状況においては、クラウドプロバイダにとって障壁となる。ライブマイグレーションの本質は、ＶＭのメモリ内容がリモートハイパーバイザにコピーされるという点である。さらに、ＶＭがソースハイパーバイザにおいて中断され、ＶＭの動作が宛先において再開される。ソフトウェアエミュレーション方法を用いる場合、ネットワークインターフェイスは、それらの内部状態がメモリに記憶され、さらにコピーされるように仮想的である。このため、ダウンタイムは数ミリ秒にまで減らされ得る。

【００４５】

しかしながら、ＳＲ－ＩＯＶなどの直接的デバイス割当て技術が用いられる場合、マイグレーションはより困難になる。このような状況においては、ネットワークインターフェイスの内部状態全体は、それがハードウェアに結び付けられているのでコピーすることができない。代わりに、ＶＭに割当てられたＳＲ－ＩＯＶ　ＶＦが分離され、ライブマイグレーションが実行されることとなり、新しいＶＦが宛先において付与されることとなる。インフィニバンド^{TM}およびＳＲ－ＩＯＶの場合、このプロセスがダウンタイムを数秒のオーダでもたらす可能性がある。さらに、ＳＲ－ＩＯＶ共有型ポートモデルにおいては、ＶＭのアドレスがマイグレーション後に変化することとなり、これにより、ＳＭにオーバーヘッドが追加され、基礎をなすネットワークファブリックの性能に対して悪影響が及ぼされることとなる。

【００４６】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャ－共有ポート

さまざまなタイプのＳＲ－ＩＯＶモデル（たとえば共有ポートモデル、仮想スイッチモデルおよび仮想ポートモデル）があり得る。

【００４７】

図４は、一実施形態に従った例示的な共有ポートアーキテクチャを示す。図に示されるように、ホスト３００（たとえばホストチャネルアダプタ）はハイパーバイザ３１０と対話し得る。ハイパーバイザ３１０は、さまざまな仮想機能３３０、３４０および３５０をいくつかの仮想マシンに割当て得る。同様に、物理機能はハイパーバイザ３１０によって処理することができる。

【００４８】

一実施形態に従うと、図４に示されるような共有ポートアーキテクチャを用いる場合、ホスト（たとえばＨＣＡ）は、物理機能３２０と仮想機能３３０、３５０、３５０との間において単一の共有ＬＩＤおよび共有キュー対（Queue Pair：ＱＰ）のスペースがあるネットワークにおいて単一のポートとして現われる。しかしながら、各々の機能（すなわち、物理機能および仮想機能）はそれら自体のＧＩＤを有し得る。

【００４９】

図４に示されるように、一実施形態に従うと、さまざまなＧＩＤを仮想機能および物理機能に割当てることができ、特別のキュー対であるＱＰ０およびＱＰ１（すなわちインフィニバンド^{TM}管理パケットのために用いられる専用のキュー対）が物理機能によって所

有される。これらのＱＰはＶＦにも同様にエクスポーズされるが、ＶＦはＱＰ０を使用することが許可されておらず（ＶＦからＱＰ０に向かって入来するすべてのＳＭＰが廃棄され）、ＱＰ１は、ＰＦが所有する実際のＱＰ１のプロキシとして機能し得る。

【００５０】

一実施形態に従うと、共有ポートアーキテクチャは、（仮想機能に割当てられることによってネットワークに付随する）ＶＭの数によって制限されることのない高度にスケーラブルなデータセンタを可能にし得る。なぜなら、ネットワークにおける物理的なマシンおよびスイッチによってＬＩＤスペースが消費されるだけであるからである。

【００５１】

しかしながら、共有ポートアーキテクチャの欠点は、トランスペアレントなライブマイグレーションを提供することができない点であり、これにより、フレキシブルなＶＭ配置についての可能性が妨害されてしまう。各々のＬＩＤが特定のハイパーバイザに関連付けられており、かつハイパーバイザ上に常駐するすべてのＶＭ間で共有されているので、マイグレートしているＶＭ（すなわち、宛先ハイパーバイザにマイグレートする仮想マシン）は、そのＬＩＤを宛先ハイパーバイザのＬＩＤに変更させなければならない。さらに、ＱＰ０アクセスが制限された結果、サブネットマネージャはＶＭの内部で実行させることができなくなる。

【００５２】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャモデル－仮想スイッチ（ｖＳｗｉｔｃｈ）

図５は、一実施形態に従った例示的なｖＳｗｉｔｃｈアーキテクチャを示す。図に示されるように、ホスト４００（たとえばホストチャネルアダプタ）はハイパーバイザ４１０と対話することができ、当該ハイパーバイザ４１０は、さまざまな仮想機能４３０、４４０および４５０をいくつかの仮想マシンに割当てることができる。同様に、物理機能はハイパーバイザ４１０によって処理することができる。仮想スイッチ４１５もハイパーバイザ４０１によって処理することができる。

【００５３】

一実施形態に従うと、ｖＳｗｉｔｃｈアーキテクチャにおいては、各々の仮想機能４３０、４４０、４５０は完全な仮想ホストチャネルアダプタ（virtual Host Channel Adapter：ｖＨＣＡ）であり、これは、ハードウェアにおいて、ＶＦに割当てられたＶＭに、ＩＢアドレス一式（たとえばＧＩＤ、ＧＵＩＤ、ＬＩＤ）および専用のＱＰスペースが割当てられていることを意味する。残りのネットワークおよびＳＭについては、ＨＣＡ４００は、仮想スイッチ４１５を介して追加のノードが接続されているスイッチのように見えている。ハイパーバイザ４１０はＰＦ４２０を用いることができ、（仮想機能に付与された）ＶＭはＶＦを用いる。

【００５４】

一実施形態に従うと、ｖＳｗｉｔｃｈアーキテクチャは、トランスペアレントな仮想化を提供する。しかしながら、各々の仮想機能には固有のＬＩＤが割当てられているので、利用可能な数のＬＩＤが速やかに消費される。同様に、多くのＬＩＤアドレスが（すなわち、各々の物理機能および各々の仮想機能ごとに１つずつ）使用されている場合、より多くの通信経路をＳＭによって演算しなければならず、それらのＬＦＴを更新するために、より多くのサブネット管理パケット（ＳＭＰ）をスイッチに送信しなければならない。たとえば、通信経路の演算は大規模ネットワークにおいては数分かかる可能性がある。ＬＩＤスペースが４９１５１個のユニキャストＬＩＤに制限されており、（ＶＦを介する）各々のＶＭとして、物理ノードおよびスイッチがＬＩＤを１つずつ占有するので、ネットワークにおける物理ノードおよびスイッチの数によってアクティブなＶＭの数が制限されてしまい、逆の場合も同様に制限される。

【００５５】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャモデル－仮想ポート（ｖＰｏｒｔ）

図６は、一実施形態に従った例示的なｖＰｏｒｔの概念を示す。図に示されるように、

ホスト３００（たとえばホストチャネルアダプタ）は、さまざまな仮想機能３３０、３４０および３５０をいくつかの仮想マシンに割当てることができるハイパーバイザ４１０と対話することができる。同様に、物理機能はハイパーバイザ３１０によって処理することができる。

【００５６】

一実施形態に従うと、ベンダーに実装の自由を与えるためにｖＰｏｒｔ概念は緩やかに定義されており（たとえば、当該定義では、実装がＳＲＩＯＶ専用とすべきであるとは規定されていない）、ｖＰｏｒｔの目的は、ＶＭがサブネットにおいて処理される方法を標準化することである。ｖＰｏｒｔ概念であれば、空間ドメインおよび性能ドメインの両方においてよりスケーラブルであり得る、ＳＲ－ＩＯＶ共有のポートのようなアーキテクチャおよびｖＳｗｉｔｃｈのようなアーキテクチャの両方、または、これらのアーキテクチャの組合せが規定され得る。また、ｖＰｏｒｔはオプションのＬＩＤをサポートするとともに、共有のポートとは異なり、ＳＭは、ｖＰｏｒｔが専用のＬＩＤを用いていなくても、サブネットにおいて利用可能なすべてのｖＰｏｒｔを認識する。

【００５７】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャモデル－ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈ

一実施形態に従うと、本開示は、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャを提供するためのシステムおよび方法を提供する。

【００５８】

図７は、一実施形態に従った、ＬＩＤが予めポピュレートされた例示的なｖＳｗｉｔｃｈアーキテクチャを示す。図に示されるように、いくつかのスイッチ５０１～５０４は、ネットワーク切替環境６００（たとえばＩＢサブネット）内においてインフィニバンド^{TM}ファブリックなどのファブリックのメンバ間で通信を確立することができる。ファブリックはホストチャネルアダプタ５１０、５２０、５３０などのいくつかのハードウェアデバイスを含み得る。さらに、ホストチャネルアダプタ５１０、５２０および５３０は、それぞれ、ハイパーバイザ５１１、５２１および５３１と対話することができる。各々のハイパーバイザは、さらに、ホストチャネルアダプタと共に、いくつかの仮想機能５１４、５１５、５１６、５２４、５２５、５２６、５３４、５３５および５３６と対話し、設定し、いくつかの仮想マシンに割当てることができる。たとえば、仮想マシン１　５５０はハイパーバイザ５１１によって仮想機能１　５１４に割当てることができる。ハイパーバイザ５１１は、加えて、仮想マシン２　５５１を仮想機能２　５１５に割当て、仮想マシン３　５５２を仮想機能３　５１６に割当てることができる。ハイパーバイザ５３１は、さらに、仮想マシン４　５５３を仮想機能１　５３４に割当てることができる。ハイパーバイザは、ホストチャネルアダプタの各々の上で十分な機能を有する物理機能５１３、５２３および５３３を介してホストチャネルアダプタにアクセスすることができる。

【００５９】

一実施形態に従うと、スイッチ５０１～５０４の各々はいくつかのポート（図示せず）を含み得る。いくつかのポートは、ネットワーク切替環境６００内においてトラフィックを方向付けるためにリニアフォワーディングテーブルを設定するのに用いられる。

【００６０】

一実施形態に従うと、仮想スイッチ５１２、５２２および５３２は、それぞれのハイパーバイザ５１１、５２１、５３１によって処理することができる。このようなｖＳｗｉｔｃｈアーキテクチャにおいては、各々の仮想機能は完全な仮想ホストチャネルアダプタ（ｖＨＣＡ）であり、これは、ハードウェアにおいて、ＶＦに割当てられたＶＭに、ＩＢアドレス一式（たとえばＧＩＤ、ＧＵＩＤ、ＬＩＤ）および専用のＱＰスペースが割当てられていることを意味する。残りのネットワークおよびＳＭ（図示せず）については、ＨＣＡ５１０、５２０および５３０は、仮想スイッチを介して追加のノードが接続されているスイッチのように見えている。

【００６１】

一実施形態に従うと、本開示は、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャを提供するためのシステムおよび方法を提供する。図７を参照すると、ＬＩＤは、さまざまな物理機能５１３、５２３および５３３に、さらには、仮想機能５１４～５１６、５２４～５２６、５３４～５３６（その時点でアクティブな仮想マシンに関連付けられていない仮想機能であっても）にも、予めポピュレートされている。たとえば、物理機能５１３はＬＩＤ１が予めポピュレートされており、仮想機能１　５３４はＬＩＤ１０が予めポピュレートされている。ネットワークがブートされているとき、ＬＩＤはＳＲ－ＩＯＶ　ｖＳｗｉｔｃｈ対応のサブネットにおいて予めポピュレートされている。ＶＦのすべてがネットワークにおけるＶＭによって占有されていない場合であっても、ポピュレートされたＶＦには、図７に示されるようにＬＩＤが割当てられている。

【００６２】

一実施形態に従うと、多くの同様の物理的なホストチャネルアダプタが２つ以上のポートを有することができ（冗長性のために２つのポートが共用となっている）、仮想ＨＣＡも２つのポートで表わされ、１つまたは２つ以上の仮想スイッチを介して外部ＩＢサブネットに接続され得る。

【００６３】

一実施形態に従うと、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャにおいては、各々のハイパーバイザは、それ自体のための１つのＬＩＤをＰＦを介して消費し、各々の追加のＶＦごとに１つ以上のＬＩＤを消費することができる。ＩＢサブネットにおけるすべてのハイパーバイザにおいて利用可能なすべてのＶＦを合計すると、サブネットにおいて実行することが可能なＶＭの最大量が得られる。たとえば、サブネット内の１ハイパーバイザごとに１６個の仮想機能を備えたＩＢサブネットにおいては、各々のハイパーバイザは、サブネットにおいて１７個のＬＩＤ（１６個の仮想機能ごとに１つのＬＩＤと、物理機能のために１つのＬＩＤ）を消費する。このようなＩＢサブネットにおいては、単一のサブネットについて理論上のハイパーバイザ限度は利用可能なユニキャストＬＩＤの数によって規定されており、（４９１５１個の利用可能なＬＩＤをハイパーバイザごとに１７個のＬＩＤで割って得られる）２８９１であり、ＶＭの総数（すなわち限度）は（ハイパーバイザごとに２８９１個のハイパーバイザに１６のＶＦを掛けて得られる）４６２５６である（実質的には、ＩＢサブネットにおける各々のスイッチ、ルータまたは専用のＳＭノードが同様にＬＩＤを消費するので、実際これらの数はより小さくなる）。なお、ｖＳｗｉｔｃｈが、ＬＩＤをＰＦと共有することができるので、付加的なＬＩＤを占有する必要がないことに留意されたい。

【００６４】

一実施形態に従うと、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャにおいては、ネットワークが一旦ブートされると、すべてのＬＩＤについて通信経路が計算される。新しいＶＭを始動させる必要がある場合、システムは、サブネットにおいて新しいＬＩＤを追加する必要はない。それ以外の場合、経路の再計算を含め、ネットワークを完全に再構成させ得る動作は、最も時間を消費する要素となる。代わりに、ＶＭのための利用可能なポートはハイパーバイザのうちの１つに位置し（すなわち利用可能な仮想機能）、仮想マシンは利用可能な仮想機能に付与されている。

【００６５】

一実施形態に従うと、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャはまた、同じハイパーバイザによってホストされているさまざまなＶＭに達するために、さまざまな経路を計算して用いる能力を可能にする。本質的には、これは、ＬＩＤを連続的にすることを必要とするＬＭＣの制約によって拘束されることなく、１つの物理的なマシンに向かう代替的な経路を設けるために、このようなサブネットおよびネットワークがＬＩＤマスク制御ライク（LID-Mask-Control-like：ＬＭＣライク）な特徴を用いることを可能にする。ＶＭをマイグレートしてその関連するＬＩＤを宛先に送達する必要がある場合、不連続なＬＩＤを自由に使用できることは特に有用となる。

【００６６】

一実施形態に従うと、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈアーキテクチャについての上述の利点と共に、いくつかの検討事項を考慮に入れることができる。たとえば、ネットワークがブートされているときに、ＳＲ－ＩＯＶ　ｖＳｗｉｔｃｈ対応のサブネットにおいてＬＩＤが予めポピュレートされているので、（たとえば起動時の）最初の経路演算はＬＩＤが予めポピュレートされていなかった場合よりも時間が長くかかる可能性がある。

【００６７】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャモデル－動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈ

一実施形態に従うと、本開示は、動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈアーキテクチャを提供するためのシステムおよび方法を提供する。

【００６８】

図８は、一実施形態に従った、動的ＬＩＤ割当てがなされた例示的なｖＳｗｉｔｃｈアーキテクチャを示す。図に示されるように、いくつかのスイッチ５０１～５０４は、ネットワーク切替環境７００（たとえばＩＢサブネット）内においてインフィニバンド^{TM}ファブリックなどのファブリックのメンバ間で通信を確立することができる。ファブリックは、ホストチャネルアダプタ５１０、５２０、５３０などのいくつかのハードウェアデバイスを含み得る。ホストチャネルアダプタ５１０、５２０および５３０は、さらに、ハイパーバイザ５１１、５２１および５３１とそれぞれ対話することができる。各々のハイパーバイザは、さらに、ホストチャネルアダプタと共に、いくつかの仮想機能５１４、５１５、５１６、５２４、５２５、５２６、５３４、５３５および５３６と対話し、設定し、いくつかの仮想マシンに割当てることができる。たとえば、仮想マシン１　５５０はハイパーバイザ５１１によって仮想機能１　５１４に割当てることができる。ハイパーバイザ５１１は、加えて、仮想マシン２　５５１を仮想機能２　５１５に割当て、仮想マシン３　５５２を仮想機能３　５１６に割当てることができる。ハイパーバイザ５３１はさらに、仮想マシン４　５５３を仮想機能１　５３４に割当てることができる。ハイパーバイザは、ホストチャネルアダプタの各々の上において十分な機能を有する物理機能５１３、５２３および５３３を介してホストチャネルアダプタにアクセスすることができる。

【００６９】

一実施形態に従うと、スイッチ５０１～５０４の各々はいくつかのポート（図示せず）を含み得る。いくつかのポートは、ネットワーク切替環境７００内においてトラフィックを方向付けるためにリニアフォワーディングテーブルを設定するのに用いられる。

【００７０】

一実施形態に従うと、仮想スイッチ５１２、５２２および５３２は、それぞれのハイパーバイザ５１１、５２１および５３１によって処理することができる。このようなｖＳｗｉｔｃｈアーキテクチャにおいては、各々の仮想機能は完全な仮想ホストチャネルアダプタ（ｖＨＣＡ）であり、これは、ハードウェアにおいて、ＶＦに割当てられたＶＭに、ＩＢアドレス一式（たとえばＧＩＤ、ＧＵＩＤ、ＬＩＤ）および専用のＱＰスペースが割当てられていることを意味する。残りのネットワークおよびＳＭ（図示せず）については、ＨＣＡ５１０、５２０および５３０は、仮想スイッチを介して、追加のノードが接続されているスイッチのように見えている。

【００７１】

一実施形態に従うと、本開示は、動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈアーキテクチャを提供するためのシステムおよび方法を提供する。図８を参照すると、ＬＩＤには、さまざまな物理機能５１３、５２３および５３３が動的に割当てられており、物理機能５１３がＬＩＤ１を受取り、物理機能５２３がＬＩＤ２を受取り、物理機能５３３がＬＩＤ３を受取る。アクティブな仮想マシンに関連付けられたそれらの仮想機能はまた、動的に割当てられたＬＩＤを受取ることもできる。たとえば、仮想マシン１　５５０がアクティブであり、仮想機能１　５１４に関連付けられているので、仮想機能５１４にはＬＩＤ５が割当てられ得る。同様に、仮想機能２　５１５、仮想機能３　５１６および仮想機能

１　５３４は、各々、アクティブな仮想機能に関連付けられている。このため、これらの仮想機能にＬＩＤが割当てられ、ＬＩＤ７が仮想機能２　５１５に割当てられ、ＬＩＤ１１が仮想機能３　５１６に割当てられ、ＬＩＤ９が仮想機能１　５３４に割当てられている。ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈとは異なり、アクティブな仮想マシンにその時点で関連付けられていない仮想機能はＬＩＤの割当てを受けない。

【００７２】

一実施形態に従うと、動的ＬＩＤ割当てがなされていれば、最初の経路演算を実質的に減らすことができる。ネットワークが初めてブートしており、ＶＭが存在していない場合、比較的少数のＬＩＤを最初の経路計算およびＬＦＴ分配のために用いることができる。

【００７３】

一実施形態に従うと、多くの同様の物理的なホストチャネルアダプタが２つ以上のポートを有することができ（冗長性のために２つのポートが共用となっている）、仮想ＨＣＡも２つのポートで表わされ、１つまたは２つ以上の仮想スイッチを介して外部ＩＢサブネットに接続され得る。

【００７４】

一実施形態に従うと、動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈを利用するシステムにおいて新しいＶＭが作成される場合、どのハイパーバイザ上で新しく追加されたＶＭをブートすべきであるかを決定するために、自由なＶＭスロットが発見され、固有の未使用のユニキャストＬＩＤも同様に発見される。しかしながら、新しく追加されたＬＩＤを処理するためのスイッチのＬＦＴおよびネットワークに既知の経路が存在しない。新しく追加されたＶＭを処理するために新しいセットの経路を演算することは、いくつかのＶＭが毎分ごとにブートされ得る動的な環境においては望ましくない。大規模なＩＢサブネットにおいては、新しい１セットのルートの演算には数分かかる可能性があり、この手順は、新しいＶＭがブートされるたびに繰返されなければならないだろう。

【００７５】

有利には、一実施形態に従うと、ハイパーバイザにおけるすべてのＶＦがＰＦと同じアップリンクを共有しているので、新しいセットのルートを演算する必要はない。ネットワークにおけるすべての物理スイッチのＬＦＴを繰返し、（ＶＭが作成されている）ハイパーバイザのＰＦに属するＬＩＤエントリから新しく追加されたＬＩＤにフォワーディングポートをコピーし、かつ、特定のスイッチの対応するＬＦＴブロックを更新するために単一のＳＭＰを送信するだけでよい。これにより、当該システムおよび方法では、新しいセットのルートを演算する必要がなくなる。

【００７６】

一実施形態に従うと、動的ＬＩＤ割当てアーキテクチャを備えたｖＳｗｉｔｃｈにおいて割当てられたＬＩＤは連続的である必要はない。各々のハイパーバイザ上のＶＭ上で割当てられたＬＩＤをＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈと動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈとで比較すると、動的ＬＩＤ割当てアーキテクチャにおいて割当てられたＬＩＤが不連続であり、そこに予めポピュレートされたＬＩＤが本質的に連続的であることが分かるだろう。さらに、ｖＳｗｉｔｃｈ動的ＬＩＤ割当てアーキテクチャにおいては、新しいＶＭが作成されると、次に利用可能なＬＩＤが、ＶＭの生存期間の間中ずっと用いられる。逆に、ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈにおいては、各々のＶＭは、対応するＶＦに既に割当てられているＬＩＤを引継ぎ、ライブマイグレーションのないネットワークにおいては、所与のＶＦに連続的に付与されたＶＭが同じＬＩＤを得る。

【００７７】

一実施形態に従うと、動的ＬＩＤ割当てアーキテクチャを備えたｖＳｗｉｔｃｈは、いくらかの追加のネットワークおよびランタイムＳＭオーバーヘッドを犠牲にして、予めポピュレートされたＬＩＤアーキテクチャモデルを備えたｖＳｗｉｔｃｈの欠点を解決することができる。ＶＭが作成されるたびに、作成されたＶＭに関連付けられた、新しく追加されたＬＩＤで、サブネットにおける物理スイッチのＬＦＴが更新される。この動作のた

めに、１スイッチごとに１つのサブネット管理パケット（ＳＭＰ）が送信される必要がある。各々のＶＭがそのホストハイパーバイザと同じ経路を用いているので、ＬＭＣのような機能も利用できなくなる。しかしながら、すべてのハイパーバイザに存在するＶＦの合計に対する制限はなく、ＶＦの数は、ユニキャストＬＩＤの限度を上回る可能性もある。このような場合、当然、アクティブなＶＭ上でＶＦのすべてが必ずしも同時に付与されることが可能になるわけではなく、より多くの予備のハイパーバイザおよびＶＦを備えることにより、ユニキャストＬＩＤ限度付近で動作する際に、断片化されたネットワークの障害を回復および最適化させるための融通性が追加される。

【００７８】

インフィニバンド^{TM}ＳＲ－ＩＯＶアーキテクチャモデル－動的ＬＩＤ割当てがなされかつＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈ

図９は、一実施形態に従った、動的ＬＩＤ割当てがなされてＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈを備えた例示的なｖＳｗｉｔｃｈアーキテクチャを示す。図に示されるように、いくつかのスイッチ５０１～５０４は、ネットワーク切替環境８００（たとえばＩＢサブネット）内においてインフィニバンド^{TM}ファブリックなどのファブリックのメンバ間で通信を確立することができる。ファブリックはホストチャネルアダプタ５１０、５２０、５３０などのいくつかのハードウェアデバイスを含み得る。ホストチャネルアダプタ５１０、５２０および５３０は、それぞれ、さらに、ハイパーバイザ５１１、５２１および５３１と対話することができる。各々のハイパーバイザは、さらに、ホストチャネルアダプタと共に、いくつかの仮想機能５１４、５１５、５１６、５２４、５２５、５２６、５３４、５３５および５３６と対話し、設定し、いくつかの仮想マシンに割当てることができる。たとえば、仮想マシン１　５５０は、ハイパーバイザ５１１によって仮想機能１　５１４に割当てることができる。ハイパーバイザ５１１は、加えて、仮想マシン２　５５１を仮想機能２　５１５に割当てることができる。ハイパーバイザ５２１は、仮想マシン３　５５２を仮想機能３　５２６に割当てることができる。ハイパーバイザ５３１は、さらに、仮想マシン４　５５３を仮想機能２　５３５に割当てることができる。ハイパーバイザは、ホストチャネルアダプタの各々の上において十分な機能を有する物理機能５１３、５２３および５３３を介してホストチャネルアダプタにアクセスすることができる。

【００７９】

一実施形態に従うと、スイッチ５０１～５０４の各々はいくつかのポート（図示せず）を含み得る。これらいくつかのポートは、ネットワーク切替環境８００内においてトラフィックを方向付けるためにリニアフォワーディングテーブルを設定するのに用いられる。

【００８０】

一実施形態に従うと、仮想スイッチ５１２、５２２および５３２は、それぞれのハイパーバイザ５１１、５２１、５３１によって処理することができる。このようなｖＳｗｉｔｃｈアーキテクチャにおいては、各々の仮想機能は、完全な仮想ホストチャネルアダプタ（ｖＨＣＡ）であり、これは、ハードウェアにおいて、ＶＦに割当てられたＶＭに、ＩＢアドレス一式（たとえばＧＩＤ、ＧＵＩＤ、ＬＩＤ）および専用のＱＰスペースが割当てられていることを意味する。残りのネットワークおよびＳＭ（図示せず）については、ＨＣＡ５１０、５２０および５３０は、仮想スイッチを介して、追加のノードが接続されているスイッチのように見えている。

【００８１】

一実施形態に従うと、本開示は、動的ＬＩＤ割当てがなされＬＩＤが予めポピュレートされたハイブリッドｖＳｗｉｔｃｈアーキテクチャを提供するためのシステムおよび方法を提供する。図９を参照すると、ハイパーバイザ５１１には、予めポピュレートされたＬＩＤアーキテクチャを備えたｖＳｗｉｔｃｈが配置され得るとともに、ハイパーバイザ５２１には、ＬＩＤが予めポピュレートされて動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈが配置され得る。ハイパーバイザ５３１には、動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈが配置され得る。このため、物理機能５１３および仮想機能５１４～５１６には、それ

らのＬＩＤが予めポピュレートされている（すなわち、アクティブな仮想マシンに付与されていない仮想機能であってもＬＩＤが割当てられている）。物理機能５２３および仮想機能１　５２４にはそれらのＬＩＤが予めポピュレートされ得るとともに、仮想機能２　５２５および仮想機能３　５２６にはそれらのＬＩＤが動的に割当てられている（すなわち、仮想機能２　５２５は動的ＬＩＤ割当てのために利用可能であり、仮想機能３　５２６は、仮想マシン３　５５２が付与されているので、１１というＬＩＤが動的に割当てられている）。最後に、ハイパーバイザ３　５３１に関連付けられた機能（物理機能および仮想機能）にはそれらのＬＩＤを動的に割当てることができる。これにより、結果として、仮想機能１　５３４および仮想機能３　５３６が動的ＬＩＤ割当てのために利用可能となるとともに、仮想機能２　５３５には、仮想マシン４　５５３が付与されているので、９というＬＩＤが動的に割当てられている。

【００８２】

ＬＩＤが予めポピュレートされたｖＳｗｉｔｃｈおよび動的ＬＩＤ割当てがなされたｖＳｗｉｔｃｈがともに（いずれかの所与のハイパーバイザ内で独立して、または組合わされて）利用されている、図８に示されるような一実施形態に従うと、ホストチャネルアダプタごとの予めポピュレートされたＬＩＤの数はファブリックアドミニストレータによって定義することができ、（ホストチャネルアダプタごとに）０＜＝予めポピュレートされたＶＦ＜＝総ＶＦの範囲内になり得る。動的ＬＩＤ割当てのために利用可能なＶＦは、（ホストチャネルアダプタごとに）ＶＦの総数から予めポピュレートされたＶＦの数を減じることによって見出すことができる。

【００８３】

一実施形態に従うと、多くの同様の物理的なホストチャネルアダプタが２つ以上のポートを有することができ（冗長性のために２つのポートが共用となっている）、仮想ＨＣＡも２つのポートで表わされ、１つまたは２つ以上の仮想スイッチを介して外部ＩＢサブネットに接続され得る。

【００８４】

インフィニバンド^{TM}－サブネット間通信

一実施形態に従うと、１つのサブネット内にインフィニバンド^{TM}ファブリックを提供することに加え、本開示の実施形態は、２つ以上のサブネットにまたがるインフィニバンド^{TM}ファブリックを提供することもできる。

【００８５】

図１０は、一実施形態に従う例示的なマルチサブネットインフィニバンド^{TM}ファブリックを示す。この図に示されるように、サブネットＡ　１０００の内部の多数のスイッチ１００１～１００４は、サブネットＡ　１０００（たとえばＩＢサブネット）内におけるインフィニバンド^{TM}ファブリックなどのファブリックのメンバ間の通信を提供することができる。このファブリックは、たとえばチャネルアダプタ１０１０などの多数のハードウェアデバイスを含み得る。ホストチャネルアダプタ１０１０は、ハイパーバイザ１０１１と対話することができる。ハイパーバイザは、対話の相手であるホストチャネルアダプタとともに、多数の仮想機能１０１４をセットアップすることができる。加えて、ハイパーバイザは、仮想マシンを仮想機能各々に割当てることができる。たとえば、仮想マシン１　１０１５は仮想機能１　１０１４に割当てられる。ハイパーバイザは、その対応付けられたホストチャネルアダプタに、各ホストチャネルアダプタ上の物理機能１０１３などの十分な機能を有する物理機能を通して、アクセスすることができる。

【００８６】

さらに図１０を参照して、一実施形態に従うと、多数のスイッチ１０２１～１０２４は、サブネットＢ　１０４０（たとえばＩＢサブネット）内におけるインフィニバンド^{TM}ファブリックなどのファブリックのメンバ間の通信を提供することができる。このファブリックは、たとえばホストチャネルアダプタ１０３０などの多数のハードウェアデバイスを含み得る。ホストチャネルアダプタ１０３０は、ハイパーバイザ１０３１と対話することができる。ハイパーバイザは、対話の相手であるホストチャネルアダプタとともに、多

数の仮想機能１０３４をセットアップすることができる。加えて、ハイパーバイザは、仮想マシンを仮想機能各々に割当てることができる。たとえば、仮想マシン２　１０３５は仮想機能２　１０３４に割当てられる。ハイパーバイザは、その対応付けられたホストチャネルアダプタに、各ホストチャネルアダプタ上の物理機能１０３３などの十分な機能を有する物理機能を通して、アクセスすることができる。なお、各サブネット（すなわちサブネットＡおよびサブネットＢ）内に示されているホストチャネルアダプタは１つだけであるが、各サブネット内に複数のホストチャネルアダプタおよびそれらに対応するコンポーネントが含まれていてもよいことが、理解されるはずである。

【００８７】

一実施形態に従うと、各ホストチャネルアダプタはさらに、仮想スイッチ１０１２および仮想スイッチ１０３２などの仮想スイッチに対応付けられていてもよく、上記のように各ＨＣＡは異なるアーキテクチャモデルでセットアップされてもよい。図１０のサブネットはどちらもＬＩＤが予めポピュレートされているｖＳｗｉｔｃｈのアーキテクチャモデルを使用するものとして示されているが、これは、このようなサブネット構成すべてが同様のアーキテクチャモデルに従う必要があることを示唆しようとしているのではない。

【００８８】

一実施形態に従うと、各サブネット内の少なくとも１つのスイッチがルータに対応付けられていてもよい。たとえば、サブネットＡ　１０００内のスイッチ１００２はルータ１００５に対応付けられ、サブネットＢ　１０４０内のスイッチ１０２１はルータ１００６に対応付けられている。

【００８９】

一実施形態に従うと、サブネットＡ内の仮想マシン１などの発信ソースにおけるトラフィックを、サブネットＢ内の仮想マシン２などの異なるサブネットを宛先としてそれに向ける場合、トラフィックは、サブネットＡ内のルータ、すなわち、ルータ１００５に向ければよく、そうすると、ルータ１００５はこのトラフィックをルータ１００６とのリンクを介してサブネットＢに送ることができる。

【００９０】

ファブリックマネージャ

上述のように、インフィニバンド™ファブリックなどのネットワークファブリックは、ファブリックの各サブネット内の相互接続されたルータを用いることで、複数のサブネットにまたがることができる。一実施形態に従うと、ファブリックマネージャ（図示せず）はホスト上で実現することができる。このホストは、ネットワークファブリックのメンバであり、ファブリック内で使用されることにより、ファブリックの一部である物理リソースおよび論理リソース双方を管理することができる。たとえば、特に、ファブリックリソースを発見すること、物理サーバ間の接続性を制御すること、リアルタイムのネットワーク統計を収集して観察すること、災害復旧（disaster recovery）、およびサービス品質（quality of service：ＱｏＳ）設定を設定することなどの管理タスクを、ユーザは、ファブリックマネージャを通して実行することができる。一実施形態に従うと、ファブリックマネージャは、ファブリック内に規定されたすべてのサブネットにまたがっていてもよい。すなわち、ファブリックマネージャは、リソースがどのサブネットのメンバであるかに関係なく、ファブリックのメンバであるかまたは少なくともファブリックに対応付けられている物理リソースおよび論理リソースを管理することができる。

【００９１】

一実施形態に従うと、ファブリックマネージャはグラフィカルユーザインターフェイス（graphical user interface：ＧＵＩ）を含み得る。ユーザは、ＧＵＩを通して管理機能を実行することができる。ファブリックマネージャＧＵＩは、ユーザがファブリックリソースをモニタリングし制御することを可能にする可視化ツールを取入れてもよい。たとえば、一実施形態において、ユーザは、ファブリック全体のサーバについて、サーバ接続、構成設定およびパフォーマンス統計を、ファブリックインターフェイスを通して見ることができる。ファブリックマネージャＧＵＩを通してモニタリングおよび／または管理する

ことができるファブリック機能のその他の例は、サブネット間ファブリックトポロジーを発見すること、これらのトポロジーの視覚表現を見ること、ファブリックプロファイル（たとえば仮想マシンファブリックプロファイル）を作成すること、および、ファブリックマネージャデータベースを構築し管理することを含む。ファブリックマネージャデータベースは、ネットワークファブリックが必要としネットワークファブリックに関連する、ファブリックプロファイル、メタデータ、構成設定、およびその他のデータを格納することができる。一実施形態に従うと、ファブリックマネージャデータベースはファブリックレベルデータベースである。

【００９２】

加えて、ファブリックマネージャは、どのサブネットがどのルータポートを介してどのパーティション番号を用いて通信することを許可されるかについて、法的なサブネット間接続性を規定することができる。一実施形態に従うと、ファブリックマネージャは、集中型ファブリック管理ユーティリティである。上に挙げた例は限定を意図しているのではない。

【００９３】

一実施形態に従うと、ファブリックマネージャの機能の一部はユーザによって開始されることができ、その他の機能はユーザから引き離されてもよくまたは自動化されてもよい（たとえば、機能の一部は、起動に応じてまたはその他所定のイベント時に、ファブリックマネージャによって実行されてもよい）。

【００９４】

管理イベントの例示的な実施形態において、ユーザは、ファブリックマネージャインターフェイスで、ネットワークファブリックデバイス向けの構成変更を開始してもよい。構成変更要求を受けた後に、ファブリックマネージャは、この構成変更要求が適切に実行されることを保証してもよい。たとえば、ファブリックマネージャは、要求をデバイスに伝え構成変更がデバイスの構成に書込まれることを保証してもよい。一実施形態において、物理デバイスは、構成変更が無事に終了したことをファブリックマネージャに伝える。一実施形態に従うと、それに応じて、ファブリックマネージャはインターフェイスを更新し、要求が実行されたことを視覚的に確認できるようにしてもよい。さらに、ファブリックマネージャは、たとえば災害復旧のためまたはそれ以外の目的のために、デバイスの構成をファブリックマネージャデータベースに永続化（persist）してもよい。

【００９５】

一実施形態に従うと、ファブリックマネージャは、ＧＵＩのいくつかの、すべての、またはより多くの機能を含む、コマンドラインインターフェイスなどのその他のインターフェイスを有し得る。

【００９６】

ファブリックレベルリソースドメイン

上述のように、ファブリックマネージャは、ユーザがファブリックマネージャのインターフェイスを用いてネットワークファブリック全体にわたって管理タスクを実行できるようにする。一実施形態に従うと、ファブリックマネージャのその他の機能は、階層的なロール（role）ベースのアクセス制御を容易にすることである。一実施形態において、ロールベースのアクセス制御は、ファブリックレベルリソースドメインを通して実現される。

【００９７】

一実施形態に従うと、ロールベースのアクセス制御は、ファブリックユーザという概念に基づく。人間であるアドミニストレータからのアクセスおよび外部管理アプリケーションからのアクセスはいずれも、ファブリックインフラストラクチャまたはファブリックリソースのすべてまたはサブセットに対する法的処理を定める認証されたコンテキストを表わすことができる。たとえば、ユーザは、ファブリックにおいて、ユーザプロファイルによって表わすことができる。すなわち、ファブリック内において、ユーザを、そのユーザのプロファイルを作成し属性をこのプロファイルに割当てることによって定義することができる。ユーザプロファイルには、ユーザ名属性およびパスワード属性を割当てることが

でき、ユーザ名は、ファブリック内で固有であるため、ユーザを一意に特定する。さらに、ユーザプロファイルは、ファブリック内のさまざまなリソースに特定のアクセスレベルを割当てる、ファブリック内に規定された特定の役割に、対応付けられてもよい。一実施形態に従うと、ユーザプロファイルのセットアップは、ファブリックマネージャインターフェイスを通して実現することができる。ユーザプロファイルのすべてまたは一部を、ファブリックマネージャデータベースに格納することができる。加えて、一実施形態において、ファブリックマネージャは、マイクロソフト（登録商標）のアクティブディレクトリまたはＬＤＡＰディレクトリなどの周知のユーザディレクトリと一体化してもよく、または、たとえば遠隔認証のためのＲＡＤＩＵＳネットワーキングプロトコルと一体化してもよい。

【００９８】

一実施形態に従うと、ファブリックマネージャは、ファブリックレベルリソースドメイン（本明細書では「リソースドメイン」または簡単に「ドメイン」と呼ぶ）を通して自身が発見したファブリックリソースを管理することができる。リソースドメインは、ファブリックレベルで規定されたファブリックリソースの論理的なグルーピングである。ファブリックリソースは、物理リソースおよび論理リソース双方を含む。リソースの例の一部は、特に、ファブリックデバイス（ＨＣＡ、物理ノード、およびスイッチなど）、ファブリックプロファイル（仮想マシンファブリックプロファイル、およびユーザプロファイルなど）、仮想マシン、クラウド、および入出力モジュールを含む。

【００９９】

一実施形態に従うと、ファブリックマネージャによって発見され管理されるすべてのファブリックリソースは、デフォルトでファブリック内に存在する（すなわちセットアップまたは構成する必要がない）デフォルトドメイン内にあり、これらのファブリックリソースには、ファブリックマネージャインターフェイスを通してアクセスすることができる。デフォルトドメインは、最高レベルのドメインである、すなわち、その他すべてのリソースドメインに対する親（parent）ドメインであり、その他すべてのリソースドメインはデフォルトドメイン内に存在する。デフォルトドメインはファブリックレベルアドミニストレータに対応付けられる。ファブリックレベルアドミニストレータもデフォルトで存在し、デフォルトでデフォルトドメイン内の管理上の権利を有するように構成される。

【０１００】

一実施形態に従うと、リソースドメインは、階層形式のリソース管理を表わす。たとえば、デフォルトドメインを構成し管理するプロセスは、ファブリックレベルアドミニストレータしか利用できない。しかしながら、子（child）ドメインは、ファブリックレベルアドミニストレータが、デフォルトドメイン内に作成することができる。たとえば、ファブリックレベルアドミニストレータは、子ドメインを作成することができ、ドメインリソースを子ドメインに追加することができる。加えて、ファブリックレベルアドミニストレータは、ドメインレベルの「ドメイン管理」ユーザを作成することができ、ドメイン管理ユーザを子ドメインに追加する（すなわち対応付ける）ことができる。ドメイン管理ユーザをリソースドメインのメンバにすることにより、ドメイン管理ユーザが、子ドメインとそれに含まれるファブリックリソースのサブセットとを管理できるようにする。一実施形態に従うと、ドメイン管理ユーザは、子メインの外部にあるリソース（すなわちこのドメイン管理が対応付けられているレベルと同様のまたはそれよりも高いレベルのリソース）を管理することはできない。しかしながら、ドメイン管理は、リソースドメインの子ドメインとして作成されたリソースドメインに含まれているリソースを管理することができる。一実施形態に従うと、ファブリックマネージャは、リソースドメインの境界が厳密に守られることを保証するセキュリティを提供する役割を担う。

【０１０１】

図１１は、リソースドメインの階層構造を示す。図示のように、デフォルトドメイン１１０２がネットワークファブリック１１００内に存在する。ファブリックレベルアドミニストレータ１１１０は、ファブリックレベルリソース１１１２、１１２４、および１１３

４の管理のためのアクセス権を有する。ファブリックレベルアドミニストレータ１１１０はまた、新たなリソースドメインをデフォルトドメイン１１０２内に作成し管理することができる。リソースドメイン１１２０および１１３０と、対応するドメインレベルドメイン管理ユーザ１１２２および１１３２とは、ファブリックレベルアドミニストレータ１１１０によって作成されたものである。ドメイン管理ユーザ１１２２は、（ファブリックレベルアドミニストレータ１１１０によりリソースドメイン１１２０に割当てられた）ファブリックリソース１１２４を管理するためのアクセス権を有するが、（より高いレベルの）ファブリックリソース１１１２または（同様のレベルの）ドメインリソース１１３４を管理するためのアクセス権は有しない。同様に、ドメイン管理ユーザ１１３２は、（ファブリックレベルアドミニストレータ１１１０によりリソースドメイン１１３０に割当てられた）ファブリックリソース１１３４を管理するためのアクセス権を有するが、（より高いレベルの）ファブリックリソース１１１２または（同様のレベルの）ドメインリソース１１２４を管理するためのアクセス権は有しない。

【０１０２】

管理パーティション

一実施形態に従うと、リソースドメインは、アドミニストレーションまたは（本明細書における呼称である）「管理」パーティションにより、サブネットレベルで表わすことができる。管理パーティションは、サブネットレベルでアクセス権をサブネットリソースに与えるグループメンバーシップを表わす。一実施形態に従うと、管理パーティションのメンバは、管理パーティションに対応付けられているいずれのサブネットリソースへのアクセス権も有するという点において、特権があると考えられる。ファブリックマネージャレベルにおいて、管理パーティションは、リソースドメインと対応するドメイン管理ユーザとに対応付けられる。このようにして、マルチテナント環境においてユーザとロールの分離をサブネットレベルで保証することができる。さらに、特定のリソースドメインに対応付けられている管理パーティションのメンバであるリソースがリソースドメインのメンバにもなるように、リソースドメインメンバーシップを管理パーティションメンバーシップに対応付けることができる。

【０１０３】

一実施形態に従うと、管理パーティションは、サブネットレベルにおいて、データパーティションの規定と同じやり方で規定することができるが、作成されたパーティションが管理パーティションであることを示す追加の属性を伴う。（先に詳述した）データパーティションと同様、管理パーティションは、一実施形態に従い、アドミニストレータにより、ファブリックマネージャインターフェイスを通して作成することができる。一実施形態において、ファブリックマネージャは、パーティションの作成中、「管理パーティション」フラグを任意のパラメータとしてサポートすることができる。作成したアドミニストレータによって選択される場合、ファブリックマネージャは、新たに作成されたパーティションが管理パーティションであることを示す追加の属性を含みかつファブリックマネージャおよびローカルマスタサブネットマネージャによって管理パーティションとして扱われるであろう。

【０１０４】

一実施形態に従うと、ファブリックマネージャは、作成されたリソースドメインごとに対応する管理パーティションを自動的に作成し、自動的に作成したパーティションを対応するリソースドメインに対応付けるように構成されてもよい。このような実施形態において、ファブリックレベルリソースがリソースドメインに追加されると、ファブリックマネージャはまた、これらを、自動的に作成されリソースドメインに対応付けられた管理パーティションに対応付ける。このように、リソースドメインに追加されたリソースは、アドミニストレータ（たとえばファブリックレベルアドミニストレータまたはドメイン管理）によるさらに他の動作を伴うことなく、リソースドメインに追加されると互いにサブネットレベルアクセス権を有することになる。

【０１０５】

加えて、一実施形態に従うと、ネットワークのサブネット全体は、トップレベルドメイン（たとえばデフォルトドメイン）を有するドメイン階層における特別なリソースドメインを表わすことができる。たとえば、ドメイン階層において、デフォルトドメインがトップレベルドメインを表わす場合、ネットワークファブリックの各サブネットは、ファブリックマネージャにより、デフォルトドメインの子ドメインとして認識されることができる。サブネット全体をトップレベルドメインの子ドメインとして認識することを、ファブリックマネージャのデフォルト挙動として構成してもよく、または、これらデフォルトドメインがアドミニストレータによって手作業で定められてもよい。ここでも、ユーザとロールを分離し、ドメイン境界を守り、サブネットレベルにおけるリソースの対応付けを実施するために、サブネットリソースドメイン全体に対応する管理パーティションを規定することができる。一実施形態に従うと、サブネット内に規定され当該サブネット内の各リソース（メンバであるかまたは管理パーティションに対応付けられている）を含む管理パーティションは、「ドメイングローバル」管理パーティションと名付けることができる。なぜなら、この構成の中で、サブネット内のすべてのリソースはその他すべてのリソースへのアクセス権を有するからである。

【０１０６】

一実施形態に従うと、管理パーティションは、ドメイン管理にとってトランスペアレントであってもよい。上述のように、ドメイングローバル管理パーティションは、ファブリックマネージャレベルでリソースドメインに対して自動的に作成することができ、その後、このドメインの範囲に割当てられたまたはこのドメインの範囲内に作成されたすべてのリソースを対応する管理パーティションに自動的に対応付けることができる。しかしながら、別の実施形態において、ドメイン管理は、関連するリソースドメイン内に種々の管理パーティションを明示的に作成することができ、その後、当該ドメイン内のリソースを、リソースドメインに対してデフォルトで作成された管理パーティションの代わりに明示的に作成された管理パーティションに対応付けることができる。

【０１０７】

一実施形態に従うと、ファブリックマネージャは、共有管理パーティションおよびプライベート管理パーティション双方の作成をサポートできる。デフォルトドメイン内のファブリックレベルアドミニストレータによって作成された管理パーティションは、個々のリソースドメインが利用できるようにすることができる共有パーティションであってもよい。ドメイン管理がメンバであるドメインによって作成された管理パーティションは、そのコンテキストに管理パーティションが作成されている特定のリソースドメインのみに対応付けられ当該特定のリソースドメインのみが利用できるプライベートパーティションであってもよい。

【０１０８】

一実施形態に従うと、ＨＣＡおよびｖＨＣＡのエンドポートは、データパーティションのメンバであってもよいのと同じく、管理パーティションのメンバであってもよい。しかしながら、一実施形態に従うと、管理パーティションはその他のサブネットリソースに対応付けることができるという点で、管理パーティションはデータパーティションと区別される。たとえば、データパーティションは管理パーティションに対応付けることができる。さらに、一実施形態に従うと、管理パーティションを子または親としての別の管理パーティションに対応付けることにより、これらの管理パーティションは階層的概念となり、対応付けられているリソースドメインの階層に対応させることができる。

【０１０９】

技術的な問題として、従来の専門用語では、ＨＣＡ（およびｖＨＣＡ）のエンドポートをパーティションの「メンバ」と呼ぶことができ、一実施形態に従うと、その他のリソースは管理パーティション「に対応付ける」ことができる。以下、これら２つの概念の技術的相違点について説明する。しかしながら、便宜上かつ読み易さのために、本文献では、管理パーティションについて、「メンバ」および「～に対応付ける」という表現を区別なく使用する場合がある。これらの表現が区別なく使用されていても、それとは関係なく、

管理パーティションへのエンドポート／ＨＣＡのメンバーシップと、管理パーティションに対応付けられたリソースとの間の技術的相違が読み手によって一貫して適用されることを意図していることが理解されるはずである。

【０１１０】

一実施形態に従うと、管理パーティションは、データパーティションの規定と同じく、Ｐ＿Ｋｅｙによって規定される。しかしながら、エンドポートは、自身がメンバであるデータパーティションを認識しているものの、エンドポートは自身がメンバであるのがどの管理パーティションなのかを認識している必要はない。よって、一実施形態において、管理パーティションを規定するＰ＿Ｋｅｙは、メンバのエンドポートのＰ＿Ｋｅｙテーブルに入力されない。このように、管理パーティションがＩＢパケットトラッフィック用に使用されなければ、管理パーティションの作成が、限りあるリソースであるＰ＿Ｋｅｙテーブルエントリを無駄に使うことはない。しかしながら、別の実施形態において、管理パーティションは管理パーティションとしてもデータパーティションとしても機能する場合がある。このような実施形態において、管理パーティションのメンバであるエンドポートのＰ＿Ｋｅｙテーブルすべてが、それぞれのＰ＿Ｋｅｙテーブルの管理パーティションのＰ＿Ｋｅｙエントリを有することができる。一実施形態に従うと、データパーティションは、管理パーティションではない何らかのパーティションであると定義してもよい。

【０１１１】

一実施形態に従うと、データパーティションは１つ以上の管理パーティションに対応付けることができる。たとえば、Ｐ＿Ｋｅｙ値によって規定されるデータパーティションは、自身の固有のＰ＿Ｋｅｙ値によって規定される管理パーティションに対応付けることができる。加えて、このデータパーティションは、別の固有のＰ＿Ｋｅｙ値によって規定される第２の管理パーティションに対応付けることができる。一実施形態に従うと、データパーティションを特定の管理パーティションに対応付けることにより、当該特定の管理パーティションのメンバであるエンドポートの最大メンバーシップレベルを規定することができる。

【０１１２】

上述の通り、管理パーティションは、サブネットリソースにアクセス権を与えるグループメンバーシップを表わす。一実施形態に従うと、管理パーティションのどのエンドポートメンバも、当該管理パーティションへのエンドポートのメンバーシップのみに基づいて、同じ管理パーティションに対応付けられているいずれのサブネットリソースへのアクセス権も有する。したがって、管理パーティションのメンバであるエンドポートはいずれも、同じ管理パーティションに対応付けられているいずれのデータパーティションへのアクセス権も有する。なお、これは、メンバエンドポートが、対応付けられたデータパーティションのメンバであることを必ずしも意味しておらず、対応付けられたデータパーティションへのアクセス権を有しそれ故にこのデータパーティションのメンバであり得ることを意味している。

【０１１３】

このような手法により、アドミニストレータが、たとえばデータパーティションへのアクセス権を、エンドポートに、データパーティションのＰ＿Ｋｅｙをこのエンドポートの Ｐ＿Ｋｅｙテーブルに手作業で含めることによって与える必要は、なくなる。一実施形態において、エンドポートがサブネット内で初期化されると、マスタサブネットマネージャは、管理パーティションの規定（たとえばＰ＿Ｋｅｙ）を保持するとともに規定された管理パーティションへのメンバーシップを規定しかつ規定された管理パーティションとの対応付けを規定する関係を保持するデータストア（たとえば以下で説明する管理パーティションレジストリ）にクエリすることによって、このエンドポートがメンバであるのはどの管理パーティションかを判断することができる。次に、サブネットマネージャはさらに、エンドポートがメンバである管理パーティションに対応付けられているデータパーティションがあるか否かを確かめるためにチェックすることができる。ＳＭが、１）エンドポートが管理パーティションのメンバであること、および、２）当該管理パーティションがデ

ータパーティションに対応付けられていることを見出した場合、ＳＭは、対応付けられているデータパーティションのＰ＿Ｋｅｙをエンドポートの Ｐ＿Ｋｅｙテーブルに自動的に置くことにより、データパーティションへアクセス権をエンドポートの自動的に与える。このように、管理パーティションは、アドミニストレータによる手作業のパーティションマッピングと比較して、より簡単でよりスケーラブルな解決策を表わす。

【０１１４】

図１２は、管理パーティションおよびデータパーティション双方を有する例示的なネットワークファブリックを示す。図１２に示されるように、管理パーティション１２３０、１２４０、および１２５０がファブリック内において規定されている。ノードＡ　１２０１～Ｅ　１２０５が、それぞれのＨＣＡ１２１１～１２１５によってファブリックに物理的に接続されている。加えて、各ＨＣＡは、少なくとも１つの管理パーティションのメンバである。ＨＣＡ１２１１およびＨＣＡ１２１４は、管理パーティション１２３０のメンバである。ＨＣＡ１２１１はまた、ＨＣＡ１２１２および１２１３とともに管理パーティション１２４０のメンバである。ＨＣＡ１２１３はさらに、ＨＣＡ１２１５とともに管理パーティション１２５０のメンバである。

【０１１５】

さらに図１２を参照して、一実施形態に従うと、データパーティション１２３２、１２４２、および１２５２がファブリック内に規定されている。データパーティション１２３２は管理パーティション１２３０に対応付けられ、データパーティション１２４２は管理パーティション１２４０に対応付けられ、データパーティション１２５２は管理パーティション１２５０に対応付けられている。一実施形態に従うと、ＨＣＡ１２１１およびＨＣＡ１２１４は、管理パーティション１２３０へのそれぞれのメンバーシップに基づく、データパーティション１２３２へのメンバーシップへのアクセス権を有する。同様に、ＨＣＡ１２１１～１２１３は、管理パーティション１２４０へのそれぞれのメンバーシップに基づく、データパーティション１２４２へのメンバーシップへのアクセス権を有する。加えて、ＨＣＡ１２１３および１２１５は、管理パーティション１２５０へのそれぞれのメンバーシップに基づく、データパーティション１２５２へのメンバーシップへのアクセス権を有する。

【０１１６】

一実施形態に従うと、管理パーティションは、物理ＨＣＡの仮想機能をｖＨＣＡに登録できるか否かを判断するために使用することもできる。一実施形態に従うと、ｖＨＣＡは、特定の仮想マシン（ＶＭ）のために計画され構成されたホストチャネルアダプタを表わす。ｖＨＣＡはＶＭとともにマイグレートするのに対し、ＶＦは物理アダプタとともに留まっているという点において、ｖＨＣＡは仮想機能（ＶＦ）と異なる。しかしながら、上述のように、物理ＨＣＡもｖＨＣＡも（また、より低いレベルでは、これらの（ｖ）ＨＣＡのエンドポートも）管理パーティションのメンバになり得る。よって、一実施形態に従うと、管理パーティションへのメンバーシップを、ＳＭが使用することにより、物理ＨＣＡからの要求である、この要求を出している物理ＨＣＡの仮想機能をｖＨＣＡに登録することを求める要求が許可可能か否かを、判断することができる。

【０１１７】

図１３は、ＨＣＡおよびｖＨＣＡを管理パーティションのメンバとして有する例示的なネットワークファブリックを示す。図１３に示されるように、サブネット１３０２はネットワークファブリック１３００の一部である。ＨＣＡ１３１０、１３２４、１３３２、および１３４４は、サブネット１３０２内でそれぞれのエンドポートを通してネットワークファブリック１３００に物理的に接続されている物理ＨＣＡを表わす。ＨＣＡ１３１０は、物理機能（ＰＦ）１３１２と仮想機能（ＶＦ）１３１４および１３１６とに対応付けられている。ＨＣＡ１３２４は、ＰＦ１３２６とＶＦ１３２８および１３２９とに対応付けられている。ＨＣＡ１３３２は、ＰＦ１３３４とＶＦ１３３６および１３３８とに対応付けられている。ＨＣＡ１３４４は、ＰＦ１３４６とＶＦ１３４８および１３４９とに対応付けられている。さらに、ｖＨＣＡ１３２０は、ＶＦ１３１４が登録され仮想マシン（Ｖ

Ｍ）１３１８に対応付けられたものとして示されている（すなわち、ＶＭ１３１８は、ネットワークファブリック１３００へのアクセス権を、ｖＨＣＡ１３２０を通して取得し、最終的には物理ＨＣＡ１３１０を通して取得する）。ｖＨＣＡ１３４０は、ＶＦ１３３７が登録されＶＭ１３３８に対応付けられている。

【０１１８】

引き続き図１３を参照して、図示の通り、ＨＣＡ１３１０および１３２４ならびにｖＨＣＡ１３２０は、管理パーティション１３５０のメンバである。加えて、ＨＣＡ１３３２および１３４４ならびにｖＨＣＡ１３４０は、管理パーティション１３６０のメンバである。結果として、ＨＣＡ１３１０および１３２４ならびにｖＨＣＡ１３２０は各々管理パーティション１３５０のメンバであるため、ｖＨＣＡ１３２０に、ＨＣＡ１３１０のＶＦ１３１４もしくは１３１６を、またはＨＣＡ１３２４のＶＦ１３２８もしくは１３２９を正式に登録することができる。同様に、ＨＣＡ１３３２および１３２４ならびにｖＨＣＡ１３４０が各々管理パーティション１３６０のメンバであるため、ｖＨＣＡ１３４０に、ＨＣＡ１３３０のＶＦ１３３６もしくは１３３８を、またはＨＣＡ１３４４のＶＦ１３４８もしくは１３４９を、正式に登録することができる。

【０１１９】

上述のように、ファブリックレベルのファブリックデータベースは、ファブリックおよびファブリックリソースに関連する情報を保持し、ファブリックマネージャによって管理される。一実施形態に従うと、ファブリックデータベースは、ファブリックリソースのインベントリーの「完全な知識」を有することができる（すなわち、ネットワークファブリックの一部であるすべてのリソースは、少なくともファブリックデータベースに保持されている記録によって表わされる）。さらに、ファブリック内の各リソースに対応付けられたアクセス権およびネームスペースは、ファブリックデータベースに格納されても、ファブリックデータベースに含まれる情報および関係から導出することも、可能である。

【０１２０】

たとえば、一実施形態に従うと、管理パーティションへのメンバーシップおよび／または管理パーティションに対するリソースの対応付けに関する情報は、ファブリックデータベースに格納できる。この情報を保持するテーブルと、これらのテーブルをまとめてリンクする関係とは、ファブリックデータベースのサブセットであってもよく、管理パーティションレジストリと呼ぶことができる。一実施形態に従うと、管理パーティションレジストリは、管理パーティショングループリソースを集めたものである。たとえば、管理パーティションレジストリ内の管理パーティショングループは、特定の管理パーティションのＨＣＡメンバ（ｖＨＣＡを含む）と、対応付けられたリソースとを集めたものであってもよく、このグループは、当該特定の管理パーティションを規定するＰ＿Ｋｅｙによってルックアップされる。加えて、管理パーティショングループのメンバおよび対応付けられたリソースを、メンバＨＣＡもしくはｖＨＣＡそれぞれに対するＧＵＩＤもしくはｖＧＵＩＤなどのキー、または、対応付けられたデータパーティションに対するＰ＿Ｋｅｙを用いて、レジストリ内でルックアップすることができる。管理パーティションのＰ＿Ｋｅｙと、メンバまたは対応付けられたリソースの固有識別子との間の関係はそれぞれ、管理パーティションへのメンバーシップまたは対応付けを規定し、管理パーティションレジストリによって維持され、より高いレベルではファブリックデータベースによって維持される。

【０１２１】

一実施形態に従うと、管理パーティションレジストリのすべてまたは一部は、記録としてＳＭのキャッシュに保持されていてもよい。たとえば、特定のサブネットのリソースに対応する管理パーティションレジストリの記録を、当該特定のサブネットのサブネットマネージャ（たとえばマスタサブネットマネージャ）の常駐メモリのキャッシュに複製することができる。管理パーティションレジストリの記録は、（たとえばＳＭがブートするときは）ＳＭによってファブリックデータベースから取出され（すなわちコピーされ）てもよく、または、ファブリックデータベースに永続化される前にキャッシュに置かれてもよい。キャッシュは、揮発性または不揮発性メモリであってもよい。いつレジストリ記録が

キャッシュに置かれるかには関係なく、管理パーティションレジストリのキャッシュされたコピーと、ファブリックレベルデータベースで発見された管理パーティションレジストリのコピーとの間には、同期が成立し得る。

【０１２２】

クエリを受けるたびに管理パーティション情報を永続化状態から（すなわちファブリックデータベースから）取出すのではなく、ＳＭ上の高速キャッシュ内の管理パーティションレジストリのうちのすべてまたはサブネット関連部分を保持することにより、管理パーティション情報のルックアップがＳＭに課すオーバーヘッドは最小になる。これは特に、サブネットリソース間でアクセス権が自動的に割当てられるときのサブネット初期化中において重要である。

【０１２３】

一実施形態に従うと、論理名または識別子は、リソースドメイン内のリソースに割当てることができる（たとえばファブリックレベルまたはドメインレベルの管理ユーザによって）。これらの論理名は、リソースドメインにとってプライベートなものであってもよい。ファブリックマネージャは、ファブリックデータベースを通して、ファブリック内で使用される固有の識別子（たとえばｖＧＵＩＤおよびＰ＿Ｋｅｙ）をファブリック内のリソースに与えられた論理または識別名にマッピングする関係を作成することができる。

【０１２４】

たとえば、一実施形態に従うと、ファブリックデータベースは、リソースの記録、およびリソースのドメインメンバーシップおよび／または管理パーティションメンバーシップを格納することができる。論理名は、ファブリックマネージャによってリソースが発見されると、リソースに割当てることができる。これらの名称は、ファブリックデータベース内のファブリックリソースの固有識別子にリンクさせることができる。加えて、ファブリックマネージャは、ファブリックマネージャデータベース内の関係を通して、リソースドメインおよび管理パーティションへの各リソースのメンバーシップを追跡することができる。これらの記録および関係を用いて、ファブリックマネージャは、全く異なるリソースドメインおよび管理パーティションにおける同様の論理名を可能にする。一実施形態に従うと、論理ドメイン名スキームは、特定のドメインリソースがメンバである１つまたは複数のリソースドメインの階層を反映することができる。このような実施形態において、論理リソース名は、リソースがメンバである最高レベルのリソースドメインに固有であってもよい。

【０１２５】

一実施形態に従うと、ファブリックにおけるリソースの識別子は、この識別子が何であるかに関わらず、管理パーティションの範囲内で固有であってもよい。そうすると、リソース名を、対応する管理パーティションでプレフィックスすることにより、グローバルな固有性（すなわちファブリックレベルで）を得ることができる。

【０１２６】

図１４は、一実施形態に従う、リソースドメインおよび管理ドメイン双方を有する例示的なネットワークファブリックを示す。図１４に示されるように、ファブリックマネージャ１４０２はネットワークファブリック１４００上で実行している。一実施形態に従うと、ファブリックマネージャ１４０２は、ネットワークファブリック１４００のノード（図示せず）から実行することができる。ファブリックマネージャ１４０２は、ファブリックレベルアドミニストレータ１４０４によって管理され、ファブリックマネージャデータベース１４１４を含む。管理パーティションレジストリ１４１６は、論理名テーブル１４１８と同様、ファブリックマネージャデータベース１４１４の一部である。

【０１２７】

引き続き図１４を参照すると、サブネット１４２０がネットワークファブリック１４００内に規定されている。サブネットマネージャ１４２２は、サブネット１４２０に対応付けられており、一実施形態に従うと、ネットワークファブリック１４００における動作のためにサブネット１４２０が必要とするセマンティックランタイム動作を実行する。サブ

ネット１４２０が必要とするセットアップおよび管理タスクは、ファブリックレベルアドミニストレータ１４０４およびファブリックマネージャ１４０２によって実行されてもよい。

【０１２８】

ノード１４４４、１４５４、１４７４および１４８４は、 サブネット１４２０の一部である。ＨＣＡ１４４６は、ノード１４４４に対応付けられ、ＰＦ１４４８とＶＦ１４５０および１４５２とを含む。同様に、ＨＣＡ１４５６は、ノード１４５４に対応付けられ、ＰＦ１４５８とＶＦ１４６０および１４６２とを含む。ＨＣＡ１４７６は、ノード１４７４に対応付けられ、ＰＦ１４７８とＶＦ１４８０および１４８２とを含む。さらに、ＨＣＡ１４８６は、ノード１４８４に対応付けられ、ＰＦ１４８８とＶＦ１４９０および１４９２とを含む。ＶＭ１４４０はノード１４４４上で実行しており、ＶＭ１４７０はノード１４７４上で実行している。ｖＨＣＡ１４４２は、ＶＭ１４４０のために計画および構成されており、ＶＭ１４４０に対応付けられ、ＨＣＡ１４４６の仮想機能１４５２が登録されている。ｖＨＣＡ１４７２は、ＶＭ１４７０のために計画および構成されており、ＶＭ１４７０に対応付けられ、ＨＣＡ１４７６の仮想機能１４８２が登録されている。

【０１２９】

一実施形態に従うと、ＨＣＡ１４４６、１４５６、１４７６、および１４８６はドメインリソースとみなされ、各々の記録はファブリックマネージャデータベース１４１４に格納される。この記録は、ファブリック内のＨＣＡリソースを特定するために使用されるＧＵＩＤなどの識別子を含み得る。さらに、ｖＨＣＡ１４４２および１４７２も、ドメインリソースとみなされ、各々の記録はファブリックマネージャデータベース１４１４に格納される。この記録は、ｖＨＣＡを特定するために使用されるＧＵＩＤなどの識別子を含み得る。

【０１３０】

さらに図１４を参照すると、一実施形態に従い、リソースドメイン１４１０およびリソースドメイン１４１２がファブリックマネージャ１４０２内に作成されている。一実施形態に従うと、ファブリックレベルアドミニストレータ１４０４は、リソースドメイン１４１０およびリソースドメイン１４１２を作成する役割を担う。加えて、ドメイン管理１４０６は、リソースドメイン１４１０に対応付けられたドメインレベルアドミニストレータである。同様に、ドメイン管理１４０８は、リソースドメイン１４１２に対応付けられたドメインレベルアドミニストレータである。一実施形態に従うと、ファブリックレベルアドミニストレータ１４０４は、リソースドメインの階層性に従う、それぞれのリソースドメインの管理として、ドメイン管理１４０６および１４０８を作成することができる。

【０１３１】

一実施形態に従うと、管理パーティション１４２４および管理パーティション１４２６は、サブネット１４２０内に規定されている。管理パーティション１４２４はリソースドメイン１４１０に対応付けられ、管理パーティション１４２６はリソースドメイン１４１２に対応付けられている。

【０１３２】

図１４に示されるように、ｖＨＣＡ１４４２とＨＣＡ１４４６および１４５６とは、リソースドメイン１４１０のメンバである。一実施形態に従うと、管理パーティション１４２４はリソースドメイン１４１０に対応付けられているので、ｖＨＣＡ１４４２とＨＣＡ１４４６および１４５６とが、リソースドメイン１４１０のメンバとして追加されると、これらは管理パーティション１４２４のメンバにもなり、管理パーティション１４２４を規定するＰ＿ＫｅｙとＨＣＡ１４４６、１４５６およびｖＨＣＡ１４４２の識別子との間の関係が、管理パーティションレジストリ１４１６内に作成される。一実施形態に従うと、この関係は、ＨＣＡ１４４６、１４５６およびｖＨＣＡ１４４２を、管理パーティション１４２４のメンバとして定義する。

【０１３３】

同様に、ｖＨＣＡ１４７２とＨＣＡ１４７６および１４８６とは、リソースドメイン１

４１２のメンバである。一実施形態に従うと、管理パーティション１４２６はリソースドメイン１４１０に対応付けられているので、ｖＨＣＡ１４７２とＨＣＡ１４６６および１４８６とが、リソースドメイン１４１２のメンバとして追加されると、これらは管理パーティション１４２６のメンバにもなり、管理パーティション１４２６を規定するＰ＿Ｋｅｙと、ＨＣＡ１４７６、１４８６およびｖＨＣＡ１４７２の識別子との間の関係が、管理パーティションレジストリ１４１６内に生成される。一実施形態に従うと、この関係は、ＨＣＡ１４７６、１４８６およびｖＨＣＡ１４７２を、管理パーティション１４２６のメンバとして定義する。

【０１３４】

上述のように、一実施形態に従うと、ＶＭ１４４０（ｖＨＣＡ１４４２を含む）、ノード１４４４（ＨＣＡ１４４６を含む）およびノード１４５４（ＨＣＡ１４５６を含む）は、リソースドメイン１４１０のメンバである。本発明の一実施形態において、ファブリックレベルアドミニストレータ１４０４は、ノード１４４４およびノード１４５４をリソースドメイン１４１０に追加する役割を担う。たとえば、ファブリックレベルアドミニストレータ１４０４は、ファブリックマネージャ１４０２のインターフェイスを通して、ノード１４４４および１４５４をリソースドメイン１４１０に追加することができる。ファブリックレベルアドミニストレータ１４０４がノード１４４４および１４５４をリソースドメイン１４１０に追加すると、ドメイン管理１４０６はノード１４４４および１４５４に対して管理タスクを実行することができる。しかしながら、リソースドメインの階層スキームに従うと、ドメイン管理１４０６は、ノード１４４４および１４５４に対して、これらがリソースドメイン１４１０に追加される前に（すなわちこれらがより高いレベルのデフォルトドメイン（図示せず）のメンバである間）、管理タスクを実行することはできない。さらに、一実施形態に従うと、ドメイン管理１４０８は、ノード１４４４および１４５４に対して管理タスクを実行できない。なぜなら、ノード１４４４および１４５４は、ドメイン管理１４０８が対応付けられていないパラレルレベルのリソースドメインのメンバであるからである。

【０１３５】

引き続き図１４を参照すると、一実施形態に従い、管理パーティション１４２４および１４２６がサブネット１４２０内に規定されている。リソースドメインの階層スキームに従い、一実施形態では、管理パーティション１４２４および１４２６はファブリックレベルアドミニストレータ１４０４によって規定されたものである。別の実施形態では、ドメイン管理１４０６が管理パーティション１４２４を規定し、ドメイン管理１４０８が管理パーティション１４２６を規定している。一実施形態に従うと、管理パーティション１４２４はリソースドメイン１４１０に対応付けられ、管理パーティション１４２６はリソースドメイン１４１２に対応付けられる。上述のように、一実施形態に従うと、管理パーティション１４２４および１４２６はそれぞれ、サブネットレベルにおいてリソースドメイン１４１０および１４１２を表わす。それぞれのリソースドメインに対応付けられることに加えて、管理パーティション１４２４および１４２６は、それぞれドメイン管理１４０６および１４０８（すなわち、管理パーティション各々が対応付けられているリソースドメインの対応する管理ユーザ）に対応付けられる。上述のように、一実施形態に従うと、管理パーティションとドメインレベル管理との間のこの対応付けによって、サブネットレベルにおけるマルチテナント環境内のユーザとロールの分離を保証することができる。

【０１３６】

一実施形態に従い、データパーティション１４２８および１４３０がサブネット１４２０内に規定されている。リソースドメインの階層スキームに従い、一実施形態では、データパーティション１４２８および１４３０がファブリックレベルアドミニストレータ１４０４によって規定されている。別の実施形態では、ドメイン管理１４０６がデータパーティション１４２８を規定し、ドメイン管理１４０８がデータパーティション１４３０を規定している。図１４に示されるように、データパーティション１４２８は管理パーティション１４２４に対応付けられ、データパーティション１４３０は管理パーティション１４

２６に対応付けられている。加えて、先に説明し図１４に示しているように、ＨＣＡ１４４６、１４５６およびｖＨＣＡ１４４２は、管理パーティション１４２４のメンバである。結果として、一実施形態に従うと、ＨＣＡ１４４６、１４５６およびｖＨＣＡ１４４２は、データパーティション１４２８が対応付けられている管理パーティション（すなわち管理パーティション１４２４）のメンバであるため、データパーティション１４２８へのアクセス許可を有する。

【０１３７】

一実施形態に従い、データパーティション１４２８が管理パーティション１４２４に対応付けられると、データパーティション１４２８の識別子（たとえばデータパーティション１４２８のＰ＿Ｋｅｙ）と、管理パーティション１４２４のＰ＿Ｋｅｙとの間の関係が、管理パーティションレジストリ１４１６内に作成される。この関係は、データパーティション１４２８を、管理パーティション１４２４に対応付けられているものとして定義する。同様に、データパーティション１４３０が管理パーティション１４２６に対応付けられると、データパーティション１４３０の識別子（たとえばデータパーティション１４３０のＰ＿Ｋｅｙ）と管理パーティション１４２６のＰ＿Ｋｅｙとの間の関係が管理パーティションレジストリ１４１６内に作成される。この関係は、データパーティション１４３０を、管理パーティション１４２６に対応付けられているものとして定義する。

【０１３８】

一実施形態に従うと、データパーティション１４２８に加わることを求める要求をＨＣＡ１４４６および１４５６またはｖＨＣＡ１４４２から受けた場合、ＳＭ１４２２は、管理パーティションレジストリ１４１６をチェックし、ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２が管理パーティション１４２４のメンバであることと、データパーティション１４２８が管理パーティション１４２４に対応付けられていることとを見出すことができる。そうすると、ＳＭ１４２２は、ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２が管理パーティション１４２４のメンバでありデータパーティション１４２８が管理パーティション１４２４に対応付けられていることに基づいて、ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２が、データパーティション１４２８のメンバになることを許可することができる。ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２をデータパーティション１４２８に加わらせるのに、ファブリックレベルアドミニストレータ１４０４またはドメインレベルアドミニストレータ１４０６からのマッピングは不要であろう。

【０１３９】

加えて、ｖＨＣＡ１４４２には、ＨＣＡ１４４６のＶＦ１４５２もしくは１４５０を、または、ＨＣＡ１４５６のＶＦ１４６２もしくは１４６０を登録することができる。なぜなら、ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２は、管理パーティション１４２４のメンバであるからである（ｖＨＣＡ１４４２はＶＦ１４５２が登録されたものとして示されている）。ここでもまた、ＳＭ１４２２は、管理パーティションレジストリ１４１６をチェックしてＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２が管理パーティション１４２４のメンバであることを見出すことができる。ＨＣＡ１４４６および１４５６ならびにｖＨＣＡ１４４２が管理パーティション１４２４のメンバであることを発見すると、ＳＭ１４２２は、いかなるファブリックユーザからも介入されることなく、ｖＨＣＡ１４４２に、ＶＦ１４５２、１４５０、１４６２、および１４６０のうちのいずれかを登録することができる。

【０１４０】

仮想マシンファブリックプロファイル

上述のように、効率的なハードウェアリソースの利用およびスケーラビリティを改善するために、ＩＢファブリック等のファブリックにおいて仮想マシン（ＶＭ）を用いることができる。しかしながら、仮想マシン（ＶＭ）のライブマイグレーションは、これらのソリューションにおいて用いられるアドレス指定およびルーティングスキームのせいで、依然として問題である。一実施形態に従う方法およびシステムは、このようなＶＭのマイグ

レーションの問題を克服することを目的とし、アドレス指定スキームをサポートすることが可能な、予め定められ、非常に入手し易く、かつ物理的な場所に左右されない仮想マシンファブリックプロファイルを、容易にする。一実施形態に従うと、ＶＭファブリックプロファイルにより、ファブリック接続性を用いてＶＭの集中型セットアップおよび構成管理を可能にし、かつ、ＳＲ－ＩＯＶに基づいて、ＶＭに対し、最適化されたＶＭマイグレーションをサポートする。一実施形態に従うと、ＶＭファブリックプロファイルは、仮想マシンに関する、詳細なファブリック構成情報の、単一の集中型リポジトリを表わす。ファブリックマネージャに対応付けられたデータベース（たとえばファブリックデータベース）は、ＶＭファブリックプロファイルを構成する情報を永続化することができる。

【０１４１】

一実施形態に従うと、ＶＭファブリックプロファイルは、ＩＢファブリック等のネットワークファブリックにおいて、仮想マシン識別子（ＶＭ－ｉｄ）により、特定することができる。一実施形態において、ＶＭ－ｉｄは、ファブリック全体にわたって固有であってもよい、汎用一意識別子（universally unique identifier：ＵＵＩＤ）である固有の１２８ビット数である。しかしながら、必要なＶＭ－ｉｄの固有性は、異なる態様で管理される複数のＶＭマネージャドメインにわたる固有性のみである（たとえばＶＭ－ｉｄは管理パーティション内で固有であってもよい）。したがって、他の実施形態において、ＶＭ－ｉｄは、少なくともこのようなドメインにわたって固有である何か他の適切なタイプのＩＤであってもよい。一実施形態に従うと、ファブリックレベルまたはサブネットレベルのすべての管理エンティティは、ＶＭファブリックプロファイルのＶＭ－ｉｄを参照することにより、ＶＭファブリックプロファイルに関する情報をルックアップする。

【０１４２】

図１５は、ＶＭファブリックプロファイル情報を格納するための例示的なデータベース構造を示す。図１５は、従来のリレーショナルデータベース設計のいくつかのテーブルを示す。しかしながら、何らかの適切なデータ構造を用いて、ＶＭファブリックプロファイルデータ（たとえばフラットファイルテーブル、または境界が定められた構造等）を格納することができる。図１５において、アスタリスク（＊）はキー値を示す。図１５は、ＶＭファブリックプロファイルデータを、より大きなファブリックデータベース１５００の一部として示しているが、その他の実施形態において、ＶＭプロファイルデータは自身のデータベースに含まれていてもよい、または、ファブリックデータベース１５００にアクセスできる別のデータベースであってもよい。

【０１４３】

図１５に示されるように、ＶＭファブリックプロファイルのコンテンツは、ルックアップキーとして使用される仮想マシン識別子（ＶＭ－ｉｄ）１５０２と、ファブリックの管理における使い易さのためおよび品質改善のための論理名１５０４と、たとえばＶＭファブリックプロファイルと、ファブリックに対して規定されたその他のプロファイルとを区別するために使用されるプロファイルタイプ１５０６と、ファブリックに対して規定されたすべてのプロファイルからなる組の中の固有ｉｄであるプロファイルＩＤ１５０８と、最高シーケンスが最も最近の更新を表わすプロファイルのシーケンス番号であってもよいコンテンツ更新エニュメレータ（ＣＵＥ）１５１０とを含み得るが、これらに限定されない。図１５に示されるように、このＶＭファブリックプロファイルのコンテンツは、ＶＭプロファイルテーブル１５１２に格納することができ、ＶＭ－ｉｄは、各ＶＭファブリックプロファイルを特定するための固有のキーの働きをする。

【０１４４】

上述のように、仮想ＨＣＡ（ｖＨＣＡ）を、物理ＨＣＡの仮想機能とともに使用することにより、ＶＭへのネットワークアクセスを提供してもよい。一実施形態に従うと、各ＶＭファブリックプロファイルは、少なくとも１つのｖＨＣＡにも対応付けられる。ｖＨＣＡは、特定のＶＭに対して計画および構成することができ、この構成は、ｖＨＣＡが構成されたＶＭのファブリックプロファイルとともに、含まれ格納されてもよい。そうすると、構成されたｖＨＣＡは、物理ＨＣＡによって規定することができ物理ＨＣＡとともに常

駐できる仮想機能と異なり、仮想マシンとともにマイグレートすることができる。さらに図１５を参照すると、ｖＨＣＡは、ファブリックマネージャデータベースにおいて、ＶＭ－ｉｄ１５０２とｖＨＣＡインスタンスＩＤ１５１４との固有の組合わせとして表わすことができる。加えて、この組合わせは、ＶＭ－ｉｄキー１５０２を用いて、ＶＭプロファイルテーブル１５１２に関連するｖＨＣＡテーブルに格納することができる。

【０１４５】

物理ＨＣＡと同様に、ｖＨＣＡは複数の（仮想）ポートを有し得る。一実施形態に従うと、これらの仮想ポートは、物理ポートと全く同じように、ネットワーク環境におけるエンドポートとして働くことができる。一例として、ｖＨＣＡポートを含むすべてのエンドポートに、ＧＵＩＤ（たとえばＩＢネットワークで使用される６４ビットＧＵＩＤ）を割当てることができる。このＧＵＩＤを用いて、ＳＭのルーティングテーブルのＬＩＤ宛先アドレスを要求することができる。一実施形態に従うと、仮想ＧＵＩＤ（ｖＧＵＩＤ）は、各ｖＨＣＡポートの現在のファブリックアドレスを表わし得る。一実施形態において、ｖＧＵＩＤは、上述のように、ＶＭファブリックプロファイルに割当てられかつＶＭファブリックプロファイルとともに格納されるＧＵＩＤのリストからのｖＨＣＡポートに割当てることができる。ｖＧＵＩＤは、ファブリックマネージャＧＵＩＤポリシーに従ってファブリックマネージャにより所有され制御されるＧＵＩＤの専用プールからのファブリックプロファイルに割当てることができる。たとえば、フリーでファブリックの広さの固有ｖＧＵＩＤは、ファブリックプロファイルがファブリックマネージャに作成されたときにＶＭのために構成されたｖＨＣＡに割当てることができる。

【０１４６】

引き続き図１５を参照して、一実施形態に従うと、ｖＨＣＡポートは、ファブリックマネージャデータベースにおいて、ｖＨＣＡインスタンスＩＤ１５１４とｖＧＵＩＤ１５２２との固有の組合わせとして表わすことができる。ｖＨＣＡ構成はまた、ｖＨＣＡポート番号１５２０を含み得る。この構成は、ｖＨＣＡインスタンスＩＤ＊１５１４キーによってｖＨＣＡテーブルに関連付けることができ最終的にはＶＭ－ｉｄキー１５０２によってＶＭプロファイルテーブル１５１２に関連付けることができるｖＨＣＡポートテーブル１５１８に格納することができる。

【０１４７】

一実施形態に従うと、ｖＨＣＡは、データパーティションおよび管理パーティション双方のメンバであってもよい。ｖＨＣＡのパーティションメンバーシップは、ＶＭファブリックプロファイルにおいて、ｖＨＣＡがメンバであるパーティション（管理パーティションおよびデータパーティション双方）にｖＨＣＡ記録をリンクするファブリックデータベース内の関係によって表わすことができる。一実施形態において、たとえばｖＧＵＩＤキー１５２２は、データとネットワークファブリックにおいて規定された管理パーティションＰ＿Ｋｅｙとを含むテーブル（図示せず）に関連付けることができる。一実施形態において、ｖＧＵＩＤキーは、関係により、（上記）管理パーティションレジストリにリンクされる。一実施形態において、これに代えてまたはこれに加えて、管理パーティションレジストリをｖＨＣＡインスタンスＩＤ１５１４にリンクする関係があってもよい。これらの関係により、ファブリックコンポーネントは、ｖＨＣＡがメンバであるのはどのデータパーティションおよび管理パーティションであるかを識別することができる。

【０１４８】

図１５は、専ら例示を目的として提供されており、当業者は、ＶＭファブリックプロファイルを構成するデータの記憶装置を設計および監理する数多くのやり方が存在することを理解するであろう。さらに、上記ＶＭファブリックプロファイルコンテンツのリストは、限定ではなく例示を意図したものであり、仮想マシンファブリックプロファイルのその他の実施形態は、より多くのまたはより少ないその他のコンテンツを含み得る。一実施形態に従うと、図１５に示されるデータベースコンポーネントは、ファブリックについてその他の関連情報を保持するより大きなファブリックマネージャデータベースの一部であってもよく、その他のテーブルと相互に関連付けることにより、ファブリックマネージャお

よびその他のファブリックコンポーネントの機能を向上させることができる。

【０１４９】

一実施形態において、ユーザは、ファブリックマネージャのインターフェイスを通して仮想マシンファブリックプロファイルと対話する。たとえば、ユーザは、ファブリックマネージャを通してＶＭファブリックプロファイルを作成、編集、および削除し得る。一実施形態に従うと、ファブリックプロファイル情報の一部は、ＶＭファブリックプロファイル（たとえばＶＭファブリックプロファイルの論理名）を作成または編集するユーザによって提供され、この情報のその他の部分は、ファブリックマネージャによって、または、ファブリックマネージャが作成されているサブネットのローカルＳＭによって提供される（たとえばＶＭ－ｉｄ、ｖＨＣＡインスタンスＩＤ、またはｖＧＵＩＤ）。

【０１５０】

一実施形態に従うと、仮想マシンファブリックプロファイルの作成は、ＶＭファブリックプロファイルが対応付けられているファブリックリソースの管理特権を表わす管理コンテキストにおいて行なわれてもよい。たとえば、ドメイン管理は、ＶＭファブリックプロファイルに対して構成されているｖＨＣＡと同じリソースドメイン（および同じ管理パーティション）のメンバであるＨＣＡを有するノードが使用するために、ＶＭファブリックプロファイルを作成する。作成されたＶＭファブリックプロファイルは、（論理）リソースでありかつそれが作成されたリソースドメインのメンバであるとみなされる。このように、同じ管理パーティションのメンバであるおかげで、ＶＭファブリックプロファイルのｖＨＣＡは、リソースドメインのメンバでもあるＨＣＡのＶＦのうちいずれかとともに登録される許可を得る。こうして管理オーバーヘッドは緩和される。

【０１５１】

一実施形態に従うと、ファブリックレベルまたはドメインレベルアドミニストレータユーザは、「仮想マシンマネージャ」（Virtual Machine Manager：ＶＭＭ）と名付けられたファブリックマネージャのコンポーネントを用いることにより、ＶＭファブリックプロファイルをセットアップし構成することができる。ＶＭＭは、ＶＭファブリックプロファイルの作成においてファブリックＲＥＳＴ　ＡＰＩを使用することができる。ユーザは、たとえばファブリックマネージャのＧＵＩを通してＶＭＭにアクセスすることができる。一実施形態に従うと、管理ユーザは、ＶＭファブリックプロファイルに関連する特定のパラメータ（プロファイルに対応付けられるｖＨＣＡの論理名、プロファイルタイプ、および数など）を提供することができ、その他のパラメータ（ＶＭ－ｉｄ、ｖＧＵＩＤ、およびプロファイルに対応付けられた各ｖＨＣＡのｖＨＣＡインスタンスＩＤなど）は、ＶＭＭによって自動的に生成し割当てることができる。ＶＭファブリックプロファイルの更新および削除などのその他のＣＲＵＤアクションは、ファブリックマネージャ、具体的にはＶＭＭを通して、アドミニストレータユーザが利用することもできる。

【０１５２】

一実施形態に従うと、ＶＭファブリックプロファイルの構築に必要なパラメータすべてがＶＭＭを介して供給されると、ファブリックマネージャは、アドミニストレータユーザおよびＶＭＭによって指定された属性を有するＶＭファブリックプロファイルオブジェクトのインスタンスを作成し、このＶＭファブリックプロファイルオブジェクトをファブリックレベルデータベースに永続化することができる。

【０１５３】

一実施形態に従うと、オペレーショナルネットワークファブリックにおいて、ＶＭファブリックプロファイルは、ＳＭのキャッシュに、１つまたは複数の記録として保持することができる。ＶＭファブリックプロファイルは、（たとえばＳＭが起動したときに）ＳＭによってファブリックデータベースから取出され（すなわちコピーされ）てもよく、または、ファブリックデータベースに永続化される前にキャッシュに置かれてもよい。このキャッシュは、揮発性メモリであっても不揮発性メモリであってもよい。ＶＭ－ｉｄを、キャッシュにクエリするためのキーとして用いることにより、特定のＶＭファブリックプロファイルの属性を取出してもよい。

【０１５４】
クエリを受けるたびに、ＶＭファブリックプロファイルを、永続化された状態から（すなわちファブリックデータベースから）取出すのではなく、ＳＭ上の高速キャッシュに保持することによって、ＶＭファブリックプロファイル属性のルックアップがＳＭに対して課す可能性があるオーバーヘッドを最小にすることができる。これは、ＶＭおよびホストの起動中に、どのＨＣＡ　ＶＭとペアにすることができどのＨＣＡ　ＶＭとペアにするかを確定するのにファブリックプロファイルデータが必要なときは、特に重要である。
【０１５５】
図１６は、ＶＭファブリックプロファイルをサブネットリソースが利用できるようにするためのフローチャートである。
【０１５６】
ステップ１６１０において、ＶＭのセットアップパラメータおよび構成を含むＶＭファブリックプロファイルが規定される。
【０１５７】
ステップ１６２０において、ＶＭファブリックプロファイルはファブリックレベルデータベースに格納される。
【０１５８】
ステップ１６３０において、ＶＭファブリックプロファイルは、サブネットマネージャ上の高速メモリキャッシュを通して利用できるようにされる。
【０１５９】
ステップ１６４０において、ＶＭファブリックプロファイルデータが、高速メモリキャッシュに向けられたＶＭ－ｉｄルックアップ要求に基づいてサブネットマネージャから戻される。
【０１６０】
図１７は、仮想マシンの仮想マシンファブリックプロファイルを作成するためのフローチャートである。
【０１６１】
ステップ１７１０において、仮想マシン識別子（ＶＭ－ｉｄ）が生成される。
ステップ１７２０において、仮想ホストチャネルアダプタ（ｖＨＣＡ）を特定する仮想ホストチャネルアダプタインスタンスＩＤが生成される。
【０１６２】
ステップ１７３０において、仮想グローバル一意識別子（ｖＧＵＩＤ）が、グローバル一意識別子のプールから、ｖＨＣＡの仮想ポートの現在のファブリックアドレスとして割当てられる。
【０１６３】
ステップ１７４０において、管理パーティションを規定するＰ＿ＫｅｙとｖＧＵＩＤとの間の第１の関係が作成される。このＰ＿ＫｅｙとｖＧＵＩＤとの間の関係は、ｖＧＵＩＤが現在のファブリックアドレスとして割当てられている仮想ポートを、Ｐ＿Ｋｅｙによって規定される管理パーティションのメンバとして規定する。
【０１６４】
ステップ１７５０において、ＶＭ－ｉｄとｖＨＣＡインスタンスＩＤとの間の第２の関係が作成される。この第２の関係により、ｖＨＣＡインスタンスＩＤを、ＶＭ－ｉｄへのアクセスを通して取出すことができる。
【０１６５】
ステップ１７６０において、ＶＭ－ｉｄとｖＧＵＩＤとの間の第３の関係が作成される。この第３の関係により、ｖＧＵＩＤを、ＶＭ－ｉｄへのアクセスを通して取出すことができる。
【０１６６】
ステップ１７７０において、ＶＭ－ｉｄ、仮想ホストチャネルアダプタインスタンスＩＤ、ｖＧＵＩＤ、第１の関係、第２の関係、および第３の関係を、ＶＭ－ｉｄ、仮想ホス

トチャネルアダプタ、ｖＧＵＩＤ、第１の関係、第２の関係、および第３の関係を保存するフォーマットで永続化する。

【０１６７】

本発明の多数の特徴は、ハードウェア、ソフトウェア、ファームウェア、またはこれらを組合わせたものにおいて、これを用いて、またはこれに支援されて、実施することができる。したがって、本発明の特徴は、処理システム（たとえば１つ以上のプロセッサを含む）を用いて実現し得る。

【０１６８】

この発明の特徴は、ここに提示された特徴のうちのいずれかを行なうように処理システムをプログラミングするために使用可能な命令を格納した記憶媒体またはコンピュータ読取可能媒体であるコンピュータプログラムプロダクトにおいて、それを使用して、またはその助けを借りて実現され得る。記憶媒体は、フロッピー（登録商標）ディスク、光ディスク、ＤＶＤ、ＣＤ－ＲＯＭ、マイクロドライブ、および光磁気ディスクを含む任意のタイプのディスク、ＲＯＭ、ＲＡＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、ＤＲＡＭ、ＶＲＡＭ、フラッシュメモリ装置、磁気カードもしくは光カード、ナノシステム（分子メモリＩＣを含む）、または、命令および／もしくはデータを格納するのに好適な任意のタイプの媒体もしくは装置を含み得るものの、それらに限定されない。

【０１６９】

この発明の特徴は、機械読取可能媒体のうちのいずれかに格納された状態で、処理システムのハードウェアを制御するために、および処理システムがこの発明の結果を利用する他の機構とやり取りすることを可能にするために、ソフトウェアおよび／またはファームウェアに取込まれ得る。そのようなソフトウェアまたはファームウェアは、アプリケーションコード、装置ドライバ、オペレーティングシステム、および実行環境／コンテナを含み得るものの、それらに限定されない。

【０１７０】

この発明の特徴はまた、たとえば、特定用途向け集積回路（application specific integrated circuit：ＡＳＩＣ）などのハードウェアコンポーネントを使用して、ハードウェアにおいて実現されてもよい。ここに説明された機能を行なうようにハードウェアステートマシンを実現することは、関連技術の当業者には明らかであろう。

【０１７１】

加えて、この発明は、この開示の教示に従ってプログラミングされた１つ以上のプロセッサ、メモリおよび／またはコンピュータ読取可能記憶媒体を含む、１つ以上の従来の汎用または特殊デジタルコンピュータ、コンピューティング装置、マシン、またはマイクロプロセッサを使用して都合よく実現され得る。ソフトウェア技術の当業者には明らかであるように、この開示の教示に基づいて、適切なソフトウェアコーディングが、熟練したプログラマによって容易に準備され得る。

【０１７２】

この発明のさまざまな実施形態が上述されてきたが、それらは限定のためではなく例示のために提示されたことが理解されるべきである。この発明の精神および範囲から逸脱することなく、形状および詳細のさまざまな変更を行なうことができることは、関連技術の当業者には明らかであろう。

【０１７３】

この発明は、特定された機能およびそれらの関係の実行を示す機能的構築ブロックの助けを借りて上述されてきた。説明の便宜上、これらの機能的構築ブロックの境界は、この明細書中ではしばしば任意に規定されてきた。特定された機能およびそれらの関係が適切に実行される限り、代替的な境界を規定することができる。このため、そのようないかなる代替的な境界も、この発明の範囲および精神に含まれる。

【０１７４】

この発明の前述の説明は、例示および説明のために提供されてきた。それは、網羅的であるよう、またはこの発明を開示された形態そのものに限定するよう意図されてはいない

。この発明の幅および範囲は、上述の例示的な実施形態のいずれによっても限定されるべきでない。多くの変更および変形が、当業者には明らかになるだろう。これらの変更および変形は、開示された特徴の関連するあらゆる組合せを含む。実施形態は、この発明の原理およびその実用的応用を最良に説明するために選択され説明されたものであり、それにより、考えられる特定の使用に適したさまざまな実施形態についての、およびさまざまな変更例を有するこの発明を、当業者が理解できるようにする。この発明の範囲は、請求項およびそれらの同等例によって定義されるよう意図されている。

【図１】

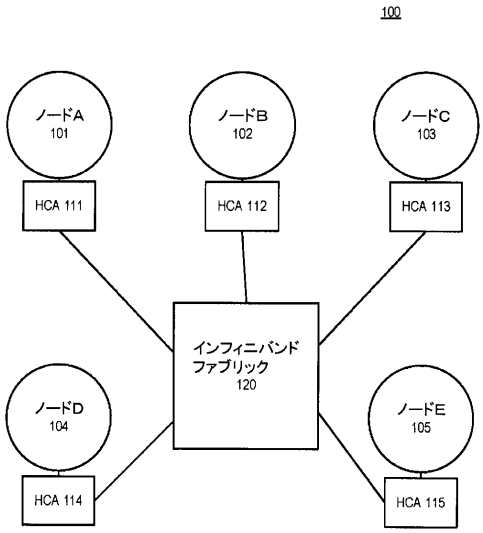

FIGURE 1

【図２】

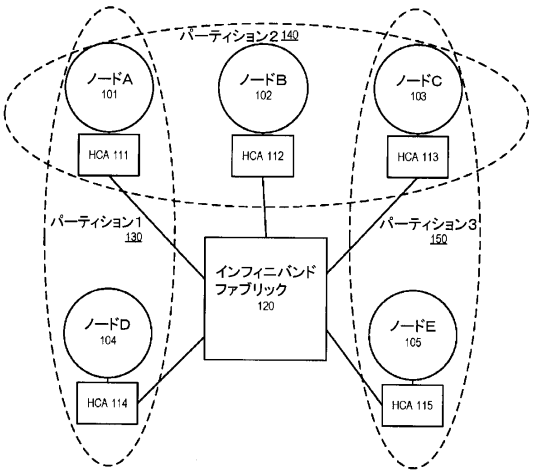

FIGURE 2

【図３】

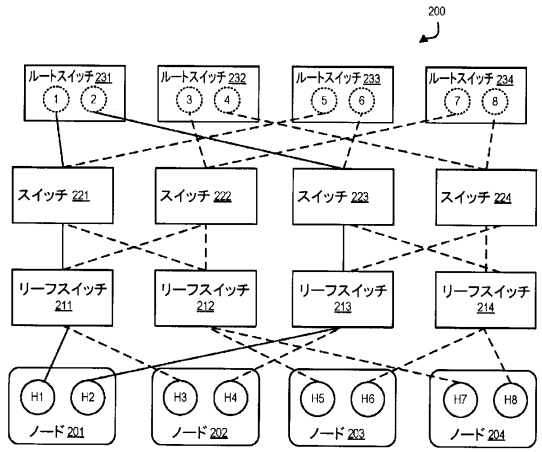

FIGURE 3

【図４】

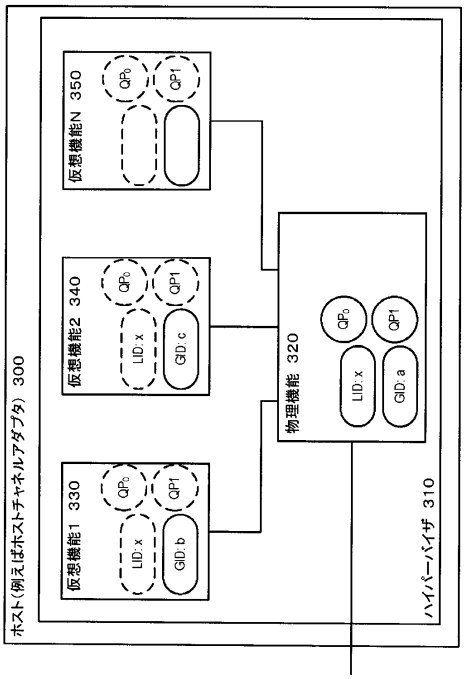

FIGURE 4

【図５】

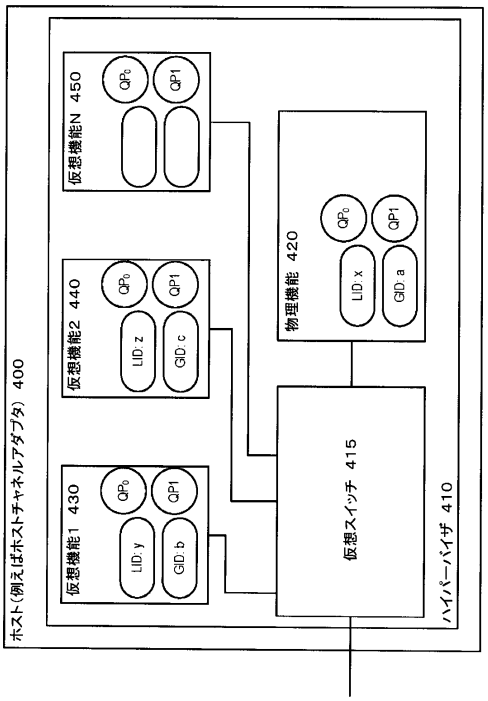

FIGURE 5

【図６】

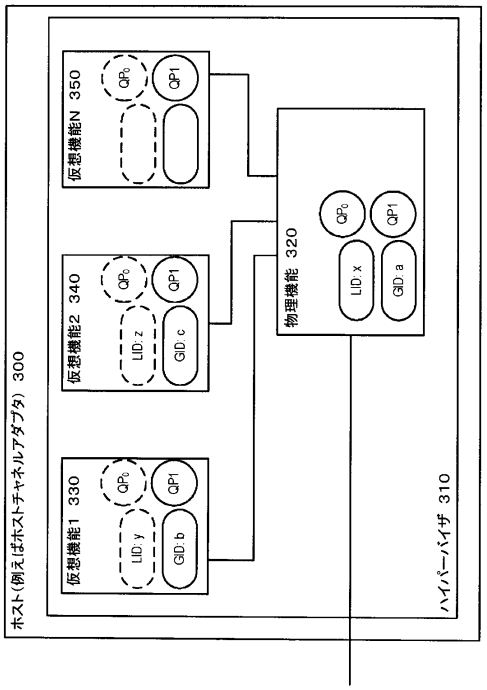

FIGURE 6

【図７】

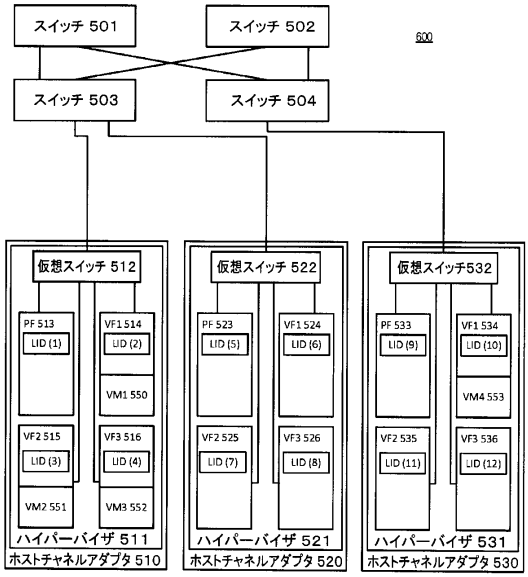

FIGURE 7

【図８】

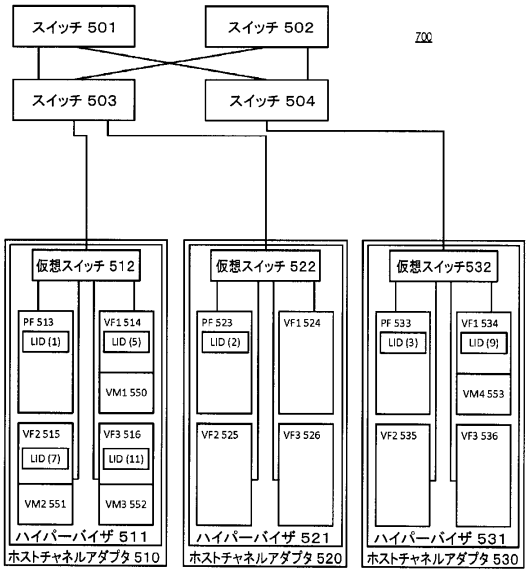

FIGURE 8

【図９】

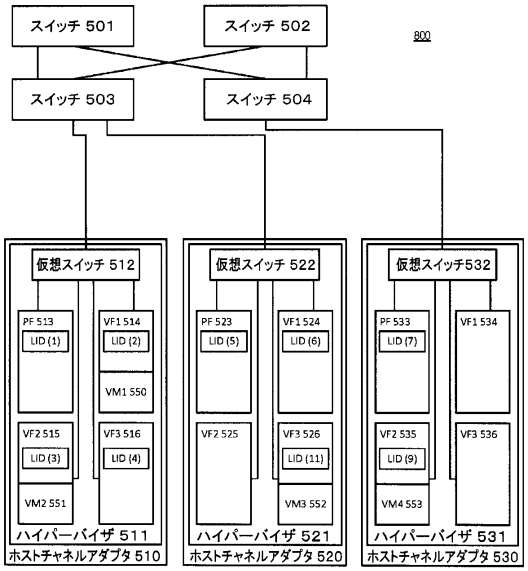

FIGURE 9

【図１０】

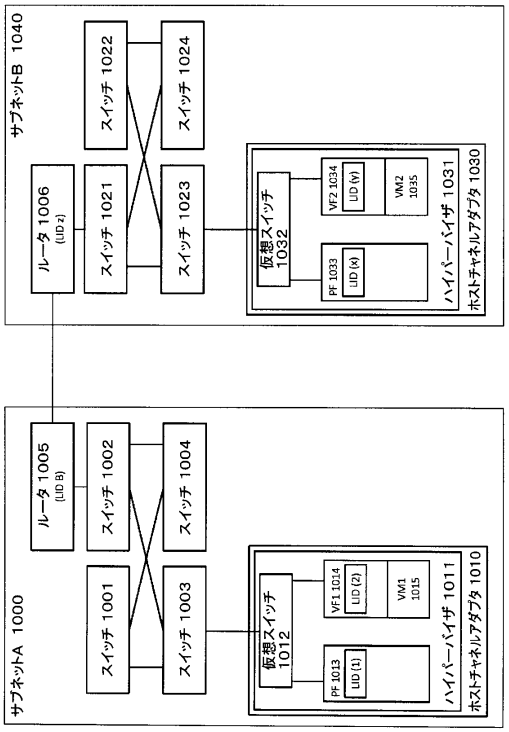

FIGURE 10

【図１１】

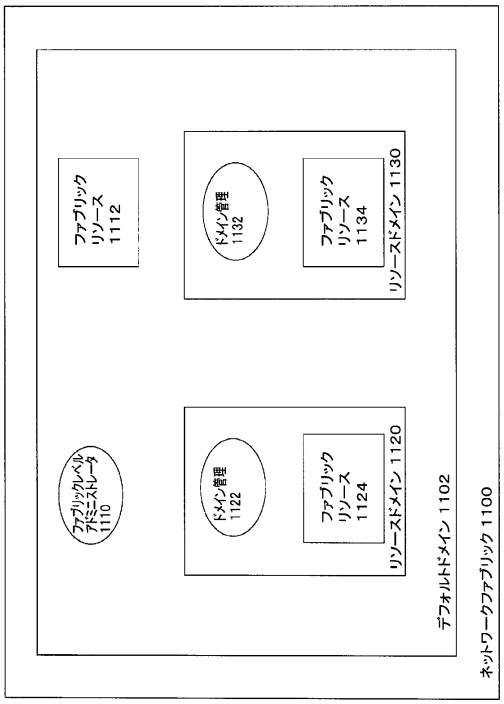

FIGURE 11

【図１２】

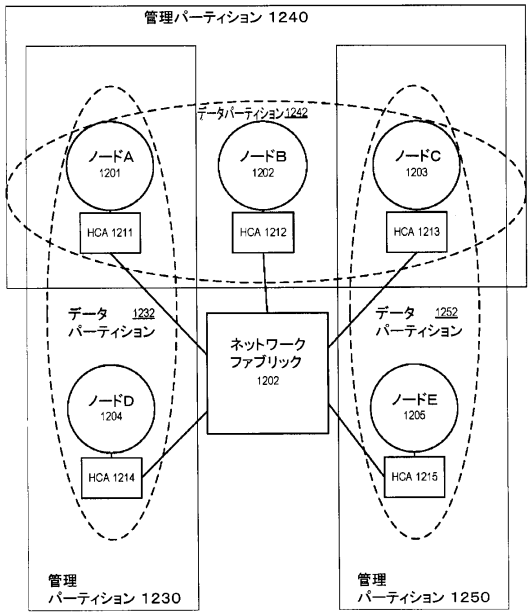

FIGURE 12

【図１３】

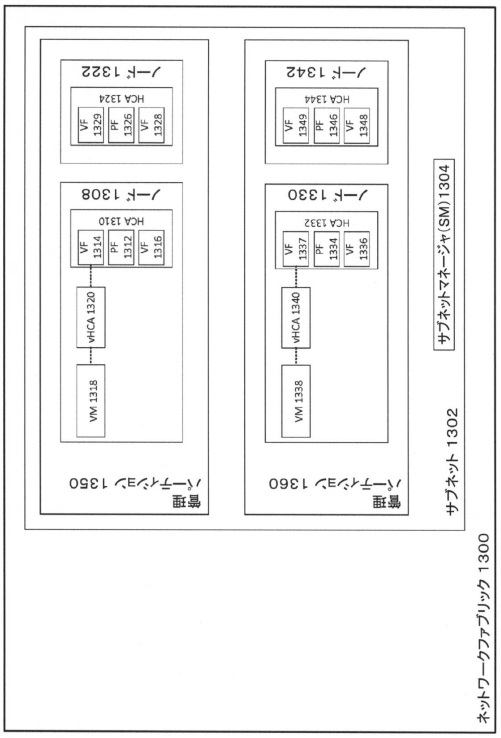

FIGURE 13

【図１４】

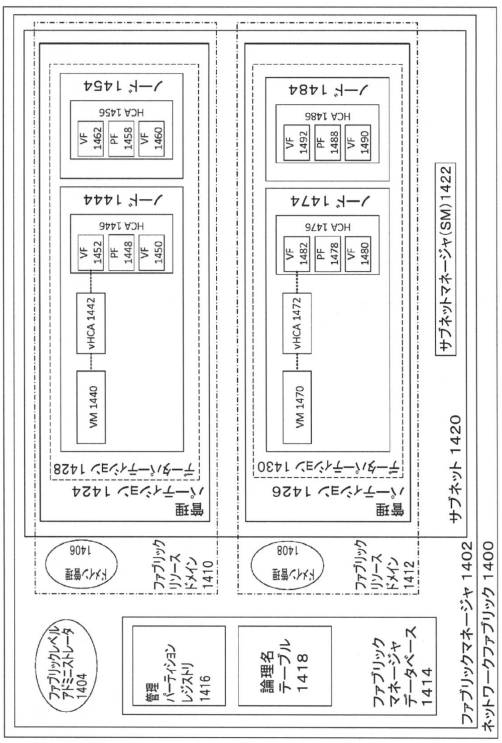

FIGURE 14

【図１５】

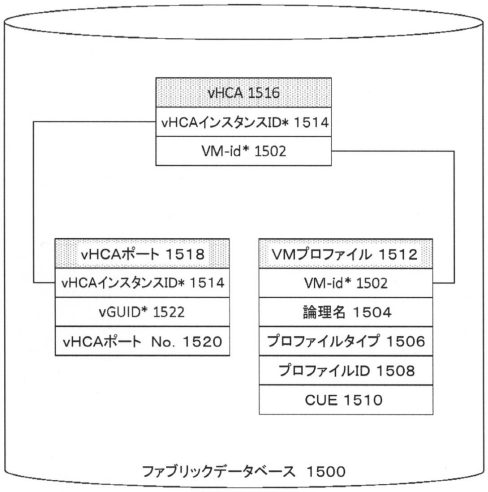

FIGURE 15

【図１６】

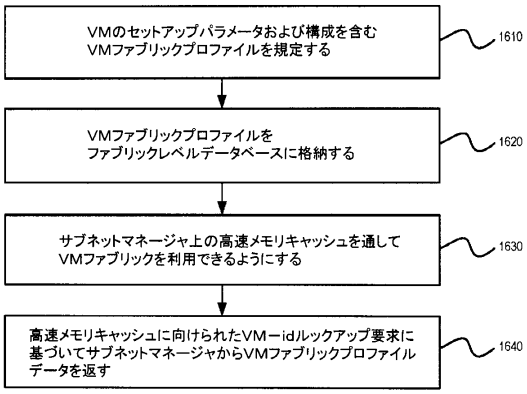

FIGURE 16

【図１７】

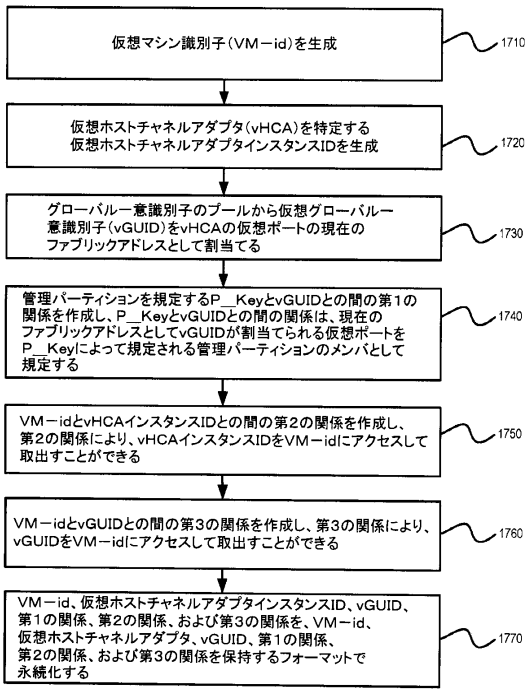

FIGURE 17

フロントページの続き

(72)発明者　ホレン，リネ
ノルウェー、１９００　フェツンド、ビタセン、１７

審査官　井上　宏一

(56)参考文献　特表２０１５－５１８６０２（ＪＰ，Ａ）
特表２００４－５３１１７５（ＪＰ，Ａ）
特表２０１５－５１５６８３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４５５－９／５４
Ｇ０６Ｆ　１５／１７７