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

特許第6752815号
(P6752815)

(45)発行日　令和2年9月9日(2020.9.9)　(24)登録日　令和2年8月21日(2020.8.21)

(51)Int.Cl.			F I		
H04L	12/715	(2013.01)	H04L	12/715	
G06F	9/50	(2006.01)	G06F	9/50	150A
H04L	12/70	(2013.01)	G06F	9/50	150C
			H04L	12/70	D

請求項の数 15　(全 34 頁)

(21)出願番号　特願2017-554456 (P2017-554456)
(86)(22)出願日　平成28年9月9日 (2016.9.9)
(65)公表番号　特表2018-537004 (P2018-537004A)
(43)公表日　平成30年12月13日 (2018.12.13)
(86)国際出願番号　PCT/US2016/050992
(87)国際公開番号　WO2017/065905
(87)国際公開日　平成29年4月20日 (2017.4.20)
審査請求日　令和1年6月24日 (2019.6.24)
(31)優先権主張番号　62/240,986
(32)優先日　平成27年10月13日 (2015.10.13)
(33)優先権主張国・地域又は機関
米国 (US)
(31)優先権主張番号　62/242,211
(32)優先日　平成27年10月15日 (2015.10.15)
(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　ザヒド，フェロツ
ノルウェー、エヌ－１３２５　リュサケール、ピィ・オゥ・ボックス・１３４
(72)発明者　グラン，アーンスト・ガンナー
ノルウェー、エヌ－１３２５　リュサケール、ピィ・オゥ・ボックス・１３４

最終頁に続く

(54)【発明の名称】マルチテナントクラスタ環境における効率的なネットワーク分離および負荷バランシングのためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
マルチテナントクラスタ環境においてネットワーク分離をサポートするための方法であって、
前記マルチテナントクラスタ環境内の１つ以上のテナントをサポートするステップと、
前記１つ以上のテナントの各々を複数のパーティションのうちのあるパーティションに関連付けるステップと、
前記複数のパーティションの各々を複数のノードのうちの１つ以上のノードに関連付けるステップとを備え、前記複数のノードの各々は、複数のスイッチのうちのリーフスイッチに関連付けられ、前記複数のスイッチは、複数のリーフスイッチと、別のレベルにおける少なくとも１つのスイッチとを備え、前記方法はさらに、
前記複数のパーティションの各々を複数のポリシーパラメータのうちのあるポリシーパラメータでマーキングするステップと、
前記複数のノードの各ノードにパーティション化順序を割り当てるステップとを備え、前記パーティション化順序は、少なくとも各ノードに関連付けられた前記パーティションにマーキングされた前記ポリシーパラメータに基づき、前記方法はさらに、
少なくとも前記複数のパーティションのうちの前記パーティションの前記マーキングに基づいて、前記マルチテナントクラスタ環境で使用される１つ以上のリニアフォワーディングテーブルを生成するステップを備える、方法。
【請求項２】

前記複数のパーティションの各々をグローバルなポリシーパラメータでマーキングするステップをさらに備える、請求項１に記載の方法。
【請求項３】
前記複数のポリシーパラメータは、
厳格なパラメータと、
厳格な仮想レーンパラメータと、
ベストエフォートパラメータとを備える、請求項１または２に記載の方法。
【請求項４】
厳格なパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序は、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序より前であり、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序は、ベストエフォートパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序より前である、請求項３に記載の方法。
【請求項５】
１つ以上のリニアフォワーディングテーブルを生成するステップは、
前記複数のリーフスイッチの各々について、各ノードの前記パーティション化順序に従って前記複数のノードを順序付けるステップと、
前記ノードの順序で複数のエンドノードをルーティングするステップとを備え、前記ルーティングするステップは、
各ノードについて少なくとも１つの下りポートおよび少なくとも１つの上りポートを選択するステップを備える、請求項４に記載の方法。
【請求項６】
前記ノードの順序で前記複数のエンドノードをルーティングするステップは、厳格なパラメータでマーキングされたパーティションに関連付けられるそれらのノードに対して厳格な分離を提供する、請求項５に記載の方法。
【請求項７】
前記ノードの順序で前記複数のエンドノードをルーティングするステップは、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるそれらのノードに対して仮想レーン上で分離を提供する、請求項５または６に記載の方法。
【請求項８】
マルチテナントクラスタ環境においてネットワーク分離をサポートするシステムであって、
１つ以上のプロセッサと、
命令を格納したメモリとを備え、前記命令は、前記１つ以上のプロセッサによって実行されたときに前記１つ以上のプロセッサにステップを実行させ、前記ステップは、
前記マルチテナントクラスタ環境内の１つ以上のテナントをサポートするステップと、
前記１つ以上のテナントの各々を複数のパーティションのうちのあるパーティションに関連付けるステップと、
前記複数のパーティションの各々を複数のノードのうちの１つ以上のノードに関連付けるステップとを備え、前記複数のノードの各々は、複数のスイッチのうちのリーフスイッチに関連付けられ、前記複数のスイッチは、複数のリーフスイッチと、別のレベルにおける少なくとも１つのスイッチとを備え、前記ステップはさらに、
前記複数のパーティションの各々を複数のポリシーパラメータのうちのあるポリシーパラメータでマーキングするステップと、
前記複数のノードの各ノードにパーティション化順序を割り当てるステップとを備え、前記パーティション化順序は、少なくとも各ノードに関連付けられた前記パーティションにマーキングされた前記ポリシーパラメータに基づき、前記ステップはさらに、
少なくとも前記複数のパーティションのうちの前記パーティションの前記マーキング

に基づいて、前記マルチテナントクラスタ環境で使用される１つ以上のリニアフォワーディングテーブルを生成するステップを備える、システム。
【請求項９】
前記１つ以上のプロセッサは、前記複数のパーティションの各々をグローバルなポリシーパラメータでマーキングするステップを備えるさらに他のステップを実行するように動作する、請求項８に記載のシステム。
【請求項１０】
前記複数のポリシーパラメータは、
厳格なパラメータと、
厳格な仮想レーンパラメータと、
ベストエフォートパラメータとを備える、請求項８または９に記載のシステム。
【請求項１１】
厳格なパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序は、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序より前であり、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序は、ベストエフォートパラメータでマーキングされたパーティションに関連付けられるノードのパーティション化順序より前である、請求項１０に記載のシステム。
【請求項１２】
１つ以上のリニアフォワーディングテーブルを生成するステップは、
前記複数のリーフスイッチの各々について、各ノードの前記パーティション化順序に従って前記複数のノードを順序付けるステップと、
前記ノードの順序で複数のエンドノードをルーティングするステップとを備え、前記ルーティングするステップは、
各ノードについて少なくとも１つの下りポートおよび少なくとも１つの上りポートを選択するステップを備える、請求項１１に記載のシステム。
【請求項１３】
前記ノードの順序で前記複数のエンドノードをルーティングするステップは、厳格なパラメータでマーキングされたパーティションに関連付けられるそれらのノードに対して厳格な分離を提供する、請求項１２に記載のシステム。
【請求項１４】
前記ノードの順序で前記複数のエンドノードをルーティングするステップは、厳格な仮想レーンパラメータでマーキングされたパーティションに関連付けられるそれらのノードに対して仮想レーン上で分離を提供する、請求項１２または１３に記載のシステム。
【請求項１５】
コンピュータプログラムであって、前記コンピュータプログラムは、マルチテナントクラスタ環境においてネットワーク分離をサポートするためのプログラム命令を備え、前記プログラム命令は、実行されたときにコンピュータに請求項１～７のいずれか１項に記載の方法を実行させる、コンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
著作権表示：
この特許文献の開示の一部は、著作権保護の対象となる題材を含んでいる。著作権所有者は、特許商標庁の包袋または記録に掲載されるように特許文献または特許情報開示を誰でも複製できることに対して異議はないが、その他の点ではすべてのいかなる著作権をも保有する。
【０００２】
発明の分野：
本発明は、一般にコンピュータシステムに関し、特にマルチテナントクラスタ環境に関

する。
【背景技術】
【０００３】
背景：
マルチテナンシは、利用可能なシステムリソースの利用度を高くすることを約束し、サービスプロバイダのために費用効果の高い動作を維持することに役立つ。しかし、マルチテナント高性能コンピューティング（high-performance computing：ＨＰＣ）インフラストラクチャは、固有の課題をもたらし、当該固有の課題は、性能分離をテナントに提供することにもネットワークファブリック全体にわたる効率的な負荷バランシングを達成することにも関連付けられる。
【発明の概要】
【課題を解決するための手段】
【０００４】
概要：
マルチテナントクラスタ環境においてネットワーク分離をサポートするためのシステムおよび方法である。例示的な方法は、上記マルチテナントクラスタ環境内の１つ以上のテナントをサポートし得る。上記方法は、上記１つ以上のテナントの各々を複数のパーティションのうちのあるパーティションに関連付け得て、また、上記複数のパーティションの各々を複数のノードのうちの１つ以上のノードに関連付け得て、上記複数のノードの各々は、複数のスイッチのうちのリーフスイッチに関連付けられ、上記複数のスイッチは、複数のリーフスイッチと、別のレベルにおける少なくとも１つのスイッチとを備える。上記方法は、上記複数のパーティションの各々を複数のポリシーパラメータのうちのあるポリシーパラメータでマーキングし得る。上記方法は、上記複数のノードの各ノードにパーティション化順序を割り当て得て、上記パーティション化順序は、少なくとも各ノードに関連付けられた上記パーティションにマーキングされたポリシーパラメータに基づく。最後に、上記方法は、少なくとも上記複数のパーティションのうちの上記パーティションの上記マーキングに基づいて、上記マルチテナントクラスタ環境で使用される１つ以上のリニアフォワーディングテーブルを生成し得る。
【図面の簡単な説明】
【０００５】
【図１】実施形態に係るマルチテナントクラスタ環境の図である。
【図２】本開示の実施形態を実施することができるネットワーク環境におけるツリートポロジの図である。
【図３】本開示の実施形態を実施することができるマルチテナントクラスタ環境におけるルーティングの図である。
【図４】本開示の実施形態を実施することができるマルチテナントクラスタ環境におけるルーティングの図である。
【図５】実施形態に係るマルチテナントクラスタ環境においてパーティション認識ルーティングをサポートすることの図である。
【図６】実施形態に係るマルチテナントクラスタ環境においてパーティション認識ルーティングをサポートすることの図である。
【図７】実施形態に係るマルチテナントクラスタ環境においてパーティション認識ルーティングをサポートすることの図である。
【図８】実施形態に係るマルチテナントクラスタ環境においてパーティション認識ルーティングをサポートすることの図である。
【図９】実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポートすることの図である。
【図１０】実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポートすることの図である。
【図１１】実施形態に係るマルチテナントクラスタ環境において重み付けされたパーティ

ション認識ルーティングをサポートすることの図である。
【図１２】実施形態に係るマルチテナントクラスタ環境において重み付けされたパーティション認識ルーティングをサポートすることの図である。
【図１３】実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポートするための方法のフローチャートである。
【図１４】本発明のさまざまな実施形態に係るコンピュータシステムを示す一般化された概略図である。
【発明を実施するための形態】
【０００６】
詳細な説明：
以下の詳細な説明では、本発明は、限定としてではなく一例として添付の図面の図に示される。なお、本開示における「ある」または「１つの」または「いくつかの」実施形態への言及は、必ずしも同一の実施形態に言及するものではなく、このような言及は少なくとも１つを意味する。具体的な実現例を記載しているが、当該具体的な実現例は例示の目的で提供されているに過ぎないということが理解される。本発明の範囲および精神から逸脱することなく他の構成要素および構成を使用してもよいということを当業者は認識するであろう。
【０００７】
図面および詳細な説明全体にわたって、共通の参照番号を使用して同様の要素を示す。したがって、要素が他の場所に記載されている場合には、図で使用される参照番号は、当該図を対象とした詳細な説明の中で言及されてもされなくてもよい。
【０００８】
本発明の以下の説明では、高性能ネットワークの一例としてインフィニバンド（商標）（Infiniband：ＩＢ）ネットワークを使用する。他のタイプの高性能ネットワークが制限なく使用されてもよいということが当業者に明らかであろう。また、以下の説明では、ファブリックトポロジの一例としてファットツリートポロジを使用する。他のタイプのファブリックトポロジが制限なく使用されてもよいということが当業者に明らかであろう。
【０００９】
インフィニバンド（商標）
インフィニバンド（商標）（ＩＢ）は、インフィニバンド（商標）トレードアソシエーションによって開発された開かれた標準的な無損失ネットワーク技術である。当該技術は、特にＨＰＣアプリケーションおよびデータセンタを対象とした、高スループットおよび低遅延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に基づく。
【００１０】
インフィニバンド（商標）アーキテクチャ（InfiniBand Architecture：ＩＢＡ）は、二層トポロジ分割をサポートする。下位層では、ＩＢネットワークはサブネットと称され、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た一組のホストを含み得る。上位レベルでは、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。
【００１１】
サブネット内において、ホスト同士は、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る。また、１つのマスタ管理エンティティ、すなわちサブネットマネージャ（subnet manager：ＳＭ）があり、当該サブネットマネージャは、サブネット内の指定されたサブネットデバイス上に存在している。サブネットマネージャは、ＩＢサブネットを構成、起動および維持することを担う。また、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実行することを担ってもよい。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内の全てのソース・宛先ペア間で負荷バランシングが適切になることを目指している。
【００１２】
サブネット管理インターフェイスを介して、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）と称さ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）とやりとりする。サブネット管理エージェントは、どのＩＢサブネットデバイス上にも存在している。ＳＭＰを使用して、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から通知を受け取ることができる。

【００１３】

一般に、マスタサブネットマネージャ以外の全ての他のサブネットマネージャは、フォールトトレランスのために待機モードで動作する。しかし、マスタサブネットマネージャが機能しなくなる状況では、待機中のサブネットマネージャによって新たなマスタサブネットマネージャが取り決められる。また、マスタサブネットマネージャは、サブネットの定期的なスイープを実行して、いかなるトポロジ変化も検出し、それに応じてネットワークを再構成する。

【００１４】

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（local identifier：ＬＩＤ）を使用してアドレス指定可能であり、単一のサブネットは、４９１５１個のＬＩＤに制限される。サブネット内で有効なローカルアドレスであるＬＩＤに加えて、各ＩＢデバイスは、その不揮発性メモリに焼き付けられた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を使用して、ＩＢ層３（Ｌ３）アドレスであるグローバル識別子（global identifier：ＧＩＤ）を形成することができる。ＧＩＤは、６４ビットのサブネット識別子（ＩＤ）を６４ビットのＧＵＩＤと連結してＩＰｖ６のような１２８ビットのアドレスを形成することによって作成することができる。たとえば、ＩＢファブリックに接続されたポートにさまざまなポートＧＵＩＤが割り当てられてもよい。

【００１５】

ＳＭは、ネットワーク初期化時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／経路）を計算し得る。さらに、接続性および最適性能を保証するために、トポロジが変化するたびにルーティングテーブルが更新され得る。通常動作中、ＳＭは、ネットワークの定期的な軽いスイープを実行して、トポロジ変化をチェックし得る。軽いスイープ中に変化が発見されるか、またはネットワーク変化を知らせるメッセージ（トラップ）がＳＭによって受信されると、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。

【００１６】

たとえば、ＳＭは、リンクがダウンしたとき、デバイスが追加されたとき、またはリンクが除去されたときなど、ネットワークトポロジが変化したときにネットワークを再構成してもよい。再構成ステップは、ネットワーク初期化中に実行さ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限定されるローカルスコープを有し得る。また、ルータを用いた大きなファブリックのセグメント化は、再構成スコープを制限する可能性がある。

【００１７】

実施形態によれば、ＩＢネットワークは、ルータを使用して相互接続された１つ以上のサブネットで構成され得る。サブネット内において、ホスト同士は、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る。各ＩＢサブネット内に、１つのマスタ管理エンティティ、すなわちサブネットマネージャ（ＳＭ）があってもよく、当該サブネットマネージャは、任意の指定されたサブネットデバイス上に存在し、ＩＢサブネットを構成、起動および維持する。

【００１８】

サブネット管理インターフェイスを介して、ＳＭは、サブネット管理パケット（ＳＭＰ）と呼ばれる制御パケットを、どのＩＢデバイス上にも存在しているサブネット管理エージェント（ＳＭＡ）とやりとりする。ＳＭＰを使用して、ＳＭ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から通知を受け取ることができる。また

、ＳＭは、サブネットの定期的な軽いスイープを実行して、いかなるトポロジ変化も検出し、それに応じてネットワークを再構成することができる。

【００１９】

実施形態によれば、ＩＢネットワークにおけるサブネット内ルーティングは、スイッチに記憶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、使用時に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。サブネットにおいて、エンドノード上の全てのＨＣＡポートおよび全てのスイッチは、ローカル識別子（ＬＩＤ）を使用してアドレス指定される。ＬＦＴへの各入力は、宛先ＬＩＤ（ＤＬＩＤ）および出力ポートで構成される。テーブルへのＬＩＤ当たり１つの入力のみがサポートされる。パケットがスイッチに到達すると、当該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その出力ポートが決定される。パケットは所与のソース・宛先ペア（ＬＩＤペア）間でネットワーク内の同一のパスを通るので、ルーティングは決定論的である。

【００２０】

実施形態によれば、パーティション化は、ネットワークファブリックを共有しているシステムの論理グループの分離を提供するためにＩＢによってサポートされるセキュリティメカニズムである。ファブリック内のノード上の各ＨＣＡポートは、１つ以上のパーティションのメンバであり得る。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、各ポートについての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（Ｐ＿Ｋｅｙ）のテーブルとして構成し得る。また、ＳＭは、ＬＩＤに関連付けられるＰ＿Ｋｅｙ値を含むパーティション実行（partition enforcement）テーブルを用いてスイッチおよびルータを構成し得る。

【００２１】

実施形態によれば、ノード間での通信のために、管理キューペア（ＱＰ０およびＱＰ１）以外に、キューペア（ＱＰ）およびエンドツーエンドコンテキスト（ＥＥＣ）が特定のパーティションに割り当てられ得る。次いで、送信されたどのＩＢトランスポートパケットにもＰ＿Ｋｅｙ情報が追加され得る。パケットがＨＣＡポートまたはスイッチに到達すると、そのＰ＿Ｋｅｙ値は、ＳＭによって構成されたテーブルに対して有効化され得る。無効なＰ＿Ｋｅｙ値が見つかれば、パケットはすぐに廃棄される。このようにして、パーティションを共有しているポート間でのみ通信が可能になる。

【００２２】

ＩＢパーティションの一例が図１に示されており、図１は、実施形態に係るマルチテナントクラスタ環境の図である。図１に示される例では、ノードＡ～Ｅ　１０１～１０５は、インフィニバンドファブリック１００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、パーティション、すなわちパーティション１　１１０、パーティション２　１２０およびパーティション３　１３０の中に配置されている。パーティション１は、ノードＡ　１０１とノードＤ　１０４とを備える。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とを備える。パーティション３は、ノードＣ　１０３とノードＥ　１０５とを備える。このようなパーティションの配置のために、ノードＤ　１０４およびノードＥ　１０５は、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、両方ともがパーティション２　１２０の一部であるので通信することができる。

【００２３】

実施形態によれば、パーティションは、ネットワークファブリックを共有しているシステムの論理グループの分離を実施するためのセキュリティメカニズムとして提供されることができる。ＩＢパーティションは、同様の分離特徴をイーサネット（登録商標）８０２．１Ｑ　ＶＬＡＮとして提供することができる。ファブリック内のノード上の各ＨＣＡポートは、１つ以上のパーティションのメンバであり得る。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することができる。ＳＭは、各ポートについての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（Ｐ＿Ｋｅｙ）のテーブルとして構成し得る。また、ＳＭは、Ｐ＿Ｋｅｙ値を含むパーティション実行テーブルを用いてスイッチおよびルータを構成し得る。

【００２４】

実施形態によれば、ノード間での通信のために、管理キューペア（ＱＰ０およびＱＰ１）以外に、キューペア（ＱＰ）およびエンドツーエンドコンテキスト（ＥＥＣ）が特定のパーティションに割り当てられ得る。次いで、送信されたどのトランスポートパケットにもＰ＿Ｋｅｙ情報が追加され得る。パケットがＨＣＡポートまたはスイッチに到達すると、そのＰ＿Ｋｅｙ値は、ＳＭによって構成されたテーブルに対して有効化され得る。無効なＰ　Ｋｅｙ値が見つかれば、パケットは廃棄される。このようにして、パーティションを共有しているポート間でのみ通信が可能になる。

【００２５】

実施形態によれば、ＩＢは、ＶＬを使用して各物理リンクを複数の仮想チャネルに分割することができる層状アーキテクチャである。各ＶＬは、それ自体のバッファリング、フロー制御および輻輳管理リソースを有し得る。一組の区別されたトラフィッククラス、すなわちＳＬ、を介してＱｏＳを提供することができる。ＳＬは、ネットワーク内でパケットが受信することができるサービスのクラスを表わす。各ＳＬは、構成されたＳＬ－ＶＬマッピングテーブルに基づいてリンク上のＶＬにマッピングされる。ＩＢは、１６個までのＶＬをサポートする。しかし、最後のＶＬは、サブネット管理トラフィックのために取っておかれて、通常はユーザアプリケーションによって使用されることはない。

【００２６】

ＩＢシステムにおけるマルチテナンシ

ネットワーキングの観点から、マルチテナンシは、ネットワークリソースの利用度を高くして、サービスプロバイダのために費用効果の高い動作を維持することに役立つことができる。しかし、マルチテナントインフラストラクチャは、いくつかの重要なセキュリティ問題も課し、最も困難なもののうちの１つは、テナントに対して性能分離を提供することに関連する。各テナントは、システム内の他のテナントのワークロードの影響を受けない予測可能なネットワーク性能を備えているべきである。ＩＢシステムにおけるネットワーク分離は、パーティション化によって提供することができる。

【００２７】

実施形態によれば、パーティションは、グループのメンバが同一グループの他のメンバとのみ通信できるようなポートの論理グループである。ホストチャネルアダプタ（ＨＣＡ）およびスイッチにおいて、パーティションメンバーシップ情報を使用してパケットをフィルタリングして分離を実施することができる。無効なパーティション化情報を有するパケットは、受信ポートに到達するとすぐに削除され得る。しかし、ＨＰＣシステムで使用されるルーティングアルゴリズムは、一般に、ネットワーク内のこのようなパーティションを認識しない。したがって、異なるパーティションに属するトラフィックフローがネットワークファブリック内のリンクを共有する可能性がある。

【００２８】

マルチテナントＩＢシステムでは、パーティションを使用してテナントクラスタを作成することができる。パーティション実行を実施すると、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は通信できなくなる。このようにして、感染したまたは不正な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。

【００２９】

一般に、ＩＢルーティングは、スイッチに記憶されたリニアフォワーディングテーブル（ＬＦＴ）に基づき得る。ＬＦＴは、パーティション化情報を考慮に入れることなくサブネットマネージャ（ＳＭ）によって計算される。したがって、中間ネットワークリンクは、さまざまなパーティションに属するトラフィックを搬送し得る。このように中間リンクを共有することは、パーティション干渉につながるおそれがある。その結果、テナントは

予測不可能なネットワーク性能を経験することになる。さらに、パーティション化されたサブネットでは、ルーティングアルゴリズムのバランシング特徴も影響を受ける。その理由は、たとえパーティション境界を横断するリンクがユーザトラフィックに利用されなくても、これらのリンクが他の機能的リンクと同じようにルーティングされる（したがって、バランシングの際に考慮される）からである。バランシングの質が低下することにより、有効帯域幅および準最適ネットワーク利用度が低下する可能性がある。

【００３０】

ＩＢは、一般に、他のパーティションにおけるノードにかかわらず、あるシェアの利用可能な帯域幅を各パーティションに保証するために使用できるサービス品質（ＱｏＳ）特徴を提供する。次いで、各パーティションにサービスレベル（service level：ＳＬ）と呼ばれる利用可能な区別されたトラフィッククラスを割り当てることによって帯域幅保証が提供される。次いで、各ＳＬは、ＳＬ－ＶＬマッピングテーブルに従ってリンク上の利用可能な１５個の仮想レーン（ＶＬ）のうちの１つにマッピングされる。

【００３１】

ＳＬをパーティションに割り当てる際に問題が生じる可能性がある。なぜなら、当該システムは１５個のＶＬのみを利用してネットワーク内に個別のパーティションを作成することができる一方、ＩＢネットワークは一般に多数のパーティションを有し得る（たとえば、各ポートは３２，７６８個までのパーティションのメンバであり得る）からである。さらに、既存のＩＢハードウェアでは、（サブネット管理のために取っておかれるものを含む）９個のＶＬのみをサポートすることが一般的である。さらに、ＳＬは希少なリソースであるので、他の目的、たとえばネットワーク内でフォールトトレランスまたはサービス差別化を提供する目的のためにできるだけ多くのＳＬを空けておくことが望ましいであろう。

【００３２】

ファットツリー（ＦＴｒｅｅ）トポロジおよびルーティング

実施形態によれば、ＩＢベースのＨＰＣシステムのうちのいくつかは、ファットツリートポロジを利用して、有用な特性であるファットツリーの提供を活用する。これらの特性は、各ソース・宛先ペア間で複数のパスを利用できることに起因した全二分割帯域幅および固有のフォールトトレランスを含む。ファットツリーの背後にある最初の考え方は、ツリーがトポロジのルートの方に移動するにつれて、利用可能な帯域幅が増加していき、ノード間のリンクが太くなっていくことを利用するというものであった。リンクが太くなることは、上位のスイッチでの輻輳の回避に役立つことができ、二分割帯域幅が維持される。

【００３３】

図２は、本開示の実施形態を実施することができるネットワーク環境におけるツリートポロジの図である。図２に示されるように、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を接続することができる。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（spine）スイッチまたはルートスイッチ２３１～２３４とを含むファットツリートポロジに基づき得る。また、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。

【００３４】

また、図２に示されるように、エンドノード２０１～２０４の各々は、マルチホームのノード、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２つ以上の部分に接続される単一のノードであってもよい。たとえば、ノード２０１はポートＨ１およびＨ２を含み得て、ノード２０２はポートＨ３およびＨ４を含み得て、ノード２０３はポートＨ５およびＨ６を含み得て、ノード２０４はポートＨ７およびＨ８を含み得る。

【００３５】

また、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を有し得て、ルートスイッチ２３２はスイッチポート３～４

を有し得て、ルートスイッチ２３３はスイッチポート５～６を有し得て、ルートスイッチ２３４はスイッチポート７～８を有し得る。

【００３６】

実施形態によれば、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のための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、ＯＦＥＤ（オープン・ファブリック・エンタープライズ・ディストリビューション：ＩＢベースのアプリケーションを構築およびデプロイするための標準的な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、でも実現される。

【００３７】

ファットツリールーティングメカニズムは、ネットワークファブリック内のリンク全体にわたって最短パス経路を均一に分散させるＬＦＴを生成することを目指している。当該メカニズムは、インデックス化順序でファブリックを横断し、エンドノード、したがって対応する経路、のターゲットＬＩＤを各スイッチポートに割り当てる。同一のリーフスイッチに接続されたエンドノードでは、インデックス化順序は、エンドノードが接続されているスイッチポート（すなわち、ポート番号付けシーケンス）に左右され得る。各ポートについて、当該メカニズムは、ポート使用カウンタを維持することができ、このポート使用カウンタを使用して、新たな経路が追加されるたびに最も使われていないポートを選択することができる。同一の２個のスイッチを接続する複数のポートがある場合には、当該ポートはポートグループを形成する。その場合、新たな経路を追加するために最小負荷ポートグループの最小負荷ポートが選択される。

【００３８】

上記のように、パーティション化されたサブネットでは、共通のパーティションのメンバではないノード同士は通信することができない。これは、事実上、ファットツリールーティングアルゴリズムによって割り当てられた経路のうちのいくつかがユーザトラフィック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他の機能的パスと同じようにそれらの経路についてＬＦＴを生成する場合に問題が生じる。この挙動は、リンク上のバランシングの質の低下を生じさせる可能性がある。なぜなら、ノードがインデックス化順序でルーティングされるからである。ルーティングがパーティションに気付かずになされるので、ファットツリールーティングが行われたサブネットでは、一般にパーティション間の分離が不十分である。

【００３９】

図３は、本開示の実施形態を実施することができるマルチテナントクラスタ環境におけるルーティングの図である。より具体的に、図３は、負荷バランシングの質の低下および不十分な分離の問題について詳細に説明する。

【００４０】

図３は、４個のスイッチ、すなわちルートスイッチ３２５～３２６およびリーフスイッチ３２０～３２１と、３個の重なり合うパーティションにおける６個のエンドノード、すなわちノードＡ～Ｆ　３０１～３０６とを有する２レベルファットツリートポロジを示す。パーティション１は、ノードＢ　３０２とノードＣ　３０３とを備える。パーティション２は、ノードＡ　３０１とノードＣ　３０３とノードＤ　３０４とノードＦ　３０６とを備える。最後に、パーティション３は、ノードＤ　３０４とノードＥ　３０５とを備える。

【００４１】

実施形態によれば、パーティション１および３は、それぞれリーフスイッチ３２０および３２１（すなわち、単一リーフスイッチパーティション）内に完全に限定されている。このため、パーティション１および３内のノード間の通信は、トラフィックをルートスイッチ３２５または３２６に移動させることなく、それらの対応するリーフスイッチを介して行われる。このトポロジがファットツリールーティングメカニズムによってルーティングされると、リーフスイッチ３２０および３２１に接続されたノードに向かう経路がルー

トスイッチに割り当てられ、そのため、リーフスイッチ間のフローはそれらの宛先に到達することができる。負荷バランシングのために、ＡおよびＣに向かう経路がルートスイッチ３２５に割り当てられる（リンクｐとして図示）一方、ルートスイッチ３２６はノードＢの方にトラフィックをルーティングする（リンクｑとして図示）。同様に、リーフスイッチ３２１では、リーフスイッチ間パーティション２内のノードＤおよびＦに向かうトラフィックがルートスイッチ３２５を介してルーティングされ（リンクｒとして図示）、ノードＥに向かうトラフィックがルートスイッチ３２６を介してルーティングされる（リンクｓとして図示）。

【００４２】

実施形態によれば、ルートスイッチ上でのエンドポートの選択は、ノード識別子を有する小さな円として図に示されている。（ファットツリールーティングメカニズムを使用して）パーティション化情報を考慮に入れることなくルーティングがなされるので、サブネット内のパスは適切なバランスがとられない。リンクｐおよびｒはオーバーサブスクライブされる（oversubscribed）一方、リーフスイッチ内のフローはリンクｑまたはｓを使用することはない。ノードＢおよびＥに向かう割り当てられた経路は、（管理トラフィックが比較的低い場合を除いて）利用されない。なぜなら、これらのノードは両方とも、パーティション化によりそれらのリーフスイッチの外部からのいかなる通信も受信できないからである。このバランシング問題は、パーティションの通信がトポロジ内のレベルのうちのいくつかのレベルのみに制限される場合にはファットツリーでも生じる。

【００４３】

ここで、本開示の実施形態を実施することができるマルチテナントクラスタ環境におけるルーティングの図である図４を参照する。より具体的に、図４は、ファットツリー内での不十分な分離に関連する問題について詳細に説明する。

【００４４】

図４は、４個のスイッチ、すなわちルートスイッチ４２５～４２６およびリーフスイッチ４２０～４２１と、８個のエンドノード、すなわちノードＡ～Ｇ　４０１～４０８とを有する２レベルファットツリートポロジを示す。同様に、エンドノードは２個のパーティションに分割されている。パーティション１は、ノードＡ　４０１とノードＢ　４０２とノードＧ　４０７とノードＨ　４０８とを備える。パーティション２は、ノードＣ　４０３とノードＤ　４０４とノードＥ　４０５とノードＦ　４０６とを備える。

【００４５】

パーティションの各々は、２個のリーフスイッチの各々に接続された２個のノードを有する。ファットツリールーティングメカニズムは、図に示されるように、ルートスイッチ４２５および４２６に下りポートを割り当てる。ファットツリールーティングメカニズムの性質のために、各ルートスイッチは、両方のパーティションに属するノードの方にトラフィックをルーティングし、これは分離を不十分にし、パーティション化された環境では望ましくないものである。たとえば、ノードＡおよびＣに向かうトラフィックは、共有リンクｐに沿ってルーティングされる。異なるパーティションのノード間での中間リンクの共有は、それらの間に干渉を引き起こすおそれがある。ネットワークがパーティション間の完全な分離を提供するのに十分なリソースをルートレベルで有しているにもかかわらず、ファットツリールーティングメカニズムは望ましい分離を提供しない。

【００４６】

パーティション認識ファットツリー（ｐＦＴｒｅｅ）ルーティング

実施形態によれば、パーティション認識ファットツリールーティングメカニズム（本明細書では別名でｐＦＴｒｅｅと称される）は、ＩＢネットワークにおけるマルチテナンシに関連する所望の目的を達成することができる。たとえば、ｐＦＴｒｅｅメカニズムは、ツリー内のリンク全体にわたって均一に経路を分散させることによって、バランスのとれたリニアフォワーディングテーブルをファットツリートポロジに提供することができる。また、ｐＦＴｒｅｅは、リンク上の経路をバランスのとれた状態に維持しながら、異なるパーティションに属するパス間の干渉を取り除くことができる。

【００４７】
実施形態によれば、ｐＦＴｒｅｅメカニズムは、サブネットについてのパーティション化情報を使用して、パーティション内のノードが、他のパーティションでのワークロードの影響を受けない予測可能なネットワーク性能を受け取ることを確実にすることができる。トポロジが（負荷バランシングに関して妥協することなく）各レベルにおいてパーティション分離を提供するのに十分な利用可能なリンクを持たない状況では、ｐＦＴｒｅｅは、ＶＬを割り当てて干渉の影響を軽減することができる。
【００４８】
実施形態によれば、ｐＦＴｒｅｅメカニズムは、各エンドノードに関連付けられるＬＩＤについて全ての関連するスイッチ上でＬＦＴをセットアップするように再帰的に機能し得る。これは、以下の疑似コード（本明細書ではリスト１と称される）で示される。
【００４９】
【数１】

```
1: for each sw ∈ leafSwitches[] do
2:   Load partitioning information
3:   Filter leaf-switch only partitions
4:   Sort end nodes in partitioning specific order
5:   for each cn ∈ endNodes[] do
6:     Get lid of cn
7:     Get partition key of the cn.hca_port
8:     Set LFT[lid] ⇐ cn:hca port on sw
9:  ROUTEDOWNGOINGBYASCENDINGTD() on sw
10:  end for
11: end for
12: ASSIGNVIRTUALLANESTD()
```

【００５０】
実施形態によれば、ROUTEDOWNGOINGBYASCENDINGTD()は、以下の疑似コード（本明細書ではリスト２と称される）で例示される。
【００５１】

【数２】

```
1:  Get least-loaded ports from sw.UpGroups[] as uplist[]
2:  selected port ⇐ upList.get_port_max_guid()
3:  for each port in upList[] do
4:    r_sw ⇐ port:get_remote_switch()
5:    if r_sw is marked with partition_key then
6:      selected_port ⇐ port
7:      break
8:    end if
9:  end for
10: if r_sw is not marked then
11:    Mark it with partition_key in DWN direction
12: end if
13: Set LFT[lid] ⇐ selected_port on r_sw
14: ROUTEUPGOINGBYDESCENDINGTD() on sw
15: ROUTEDOWNGOINGBYASCENDINGTD() on r_sw
```

【００５２】

実施形態によれば、ROUTEUPGOINGBYDESCENDINGTD()は、以下の疑似コード（本明細書ではリスト３と称される）で例示される。

【００５３】

【数３】

```
1:  Get least-loaded ports from sw.DownGroups[] as dwnlist[]
2:  selected port ⇐ dwnList.get_port_max_guid()
3:  for each port in dwnList[] do
4:    r_sw ⇐ port:get_remote_switch()
5:    if r_sw is marked with partition_key then
6:      selected_port ⇐ port
7:      break
8:    end if
9:  end for
10: if r_sw is not marked then
11:    Mark it with partition_key in UP direction
12: end if
13: Set LFT[lid] ⇐ selected_port on r_sw
14: ROUTEUPGOINGBYDESCENDINGTD() on r_sw
```

【００５４】

実施形態によれば、ASSIGNVIRTUALLANESTD()は、以下の疑似コード（本明細書では別名でリスト４と称される）で例示される。

【００５５】
【数４】

```
1:  vlanes_needed ⇐ 1
2:  max_vlanes ⇐ get_max_lanes()
3:  strict ⇐ get_is_strict()
4:  for each partition in partition_tbl do
5:  check if any intermediate communication link in this
    partition share a switch with a partition that has not
    been assigned a virtual lane
6:  if require a separate vl then
7:    if vlanes_needed = max_vlanes and strict = false then
8:      vlanes_needed ⇐ 1
9:    else
10:     error: routing failed
11:     return
12:   end if
13:   vlanes_needed++
14:   partition.vlane ⇐ vlanes_needed
15:  end if
16: end for
```

【００５６】
実施形態によれば、単一リーフスイッチパーティション（すなわち、もっぱら単一のリーフスイッチ内で通信することができるパーティション）をフィルタリングして除去した後、当該メカニズムは、各リーフスイッチについて、（たとえば、固有のパーティション化順序番号を有する各パーティションを介して）接続されたエンドノードをパーティション化に特有の順序でソートし得る（上記のリスト１の第４行）。この順序付けは、リーフスイッチにおける利用可能な上りポートの数を考慮に入れてノードがそれらのパーティションに従ってルーティングされることを確実なものとする助けとなるであろう。次いで、ｐＦＴｒｅｅメカニズムは、ROUTEDOWNGOINGBYASCENDINGTD（上記のリスト１の第９行）などの関数を呼び出して、リスト２に示されるようにツリーを上って次のレベルのポートを選択して、ＬＩＤをルーティングし得る。
【００５７】
実施形態によれば、ポートの選択は、最小数の既に割り当てられている経路に基づく。これは、利用可能なパス全体にわたって負荷を分散させることを確実なものとする助けとなるであろう。しかし、いくつかのポートが同一の負荷で利用可能である場合には、当該関数は、これらの最小負荷ポートで繰返し処理を実行して、ルーティングされているノードのパーティションキーで既にマーキングされているスイッチに接続されているポートを選択し得る（リスト２の第３～９行）。マーキングされているスイッチがない場合（これは、このパーティションの第１のノードがルーティングされていることを意味し得る）、当該システムは、デフォルトで最も高いグローバル一意識別子（ＧＵＩＤ）を有するポートの選択を実行し得る（リスト２の第２行）。パーティションにとって初めてスイッチが選択されると、当該スイッチはパーティションキーで下り方向にマーキングされる（リスト２の第１１行）。

【００５８】
実施形態によれば、スイッチにおけるＬＩＤに対して下りポートが設定された後、当該メカニズムは、（リスト３のROUTEUPGOINGBYDESCENDINGTD）を呼び出すツリーを下ることによって全ての接続された下りスイッチ上のそれに対して上りポートを割り当て得る。やはり、上りポートの選択は、まず負荷基準に基づき、次いで上り方向へのリモートスイッチのパーティションマーキングに基づき得る。次いで、全てのＬＦＴが設定されるまで、次のレベルまでツリーを上ることによってプロセスが繰返され得る。なお、スイッチは複数のパーティションキーでマーキングされてもよい。ｐＦＴｒｅｅメカニズムは、各パーティションについてルーティングされるノードの個数を記憶するテーブルを各スイッチについて維持し得る。マーキングされたパーティションを有するいくつかのスイッチがノードのルーティングに利用できる場合には、このカウンタを使用してポートの選択を決定することができる。パーティションについて最大数の既にルーティングされているノードを有するスイッチを選択することができる。

【００５９】
実施形態によれば、パーティション分離基準を保持しながらルーティングテーブルが生成されると、当該メカニズムは、リンクのうちのいくつかが異なるパーティション内のノードに向かうフローに使用されているか否かを確認することに進み得る。そのような場合、当該メカニズムは、干渉しているパーティションにＶＬを割り当てて分離を提供することができる。ＶＬ割り当てメカニズムの一例はリスト４に示されている。

【００６０】
実施形態によれば、ＶＬ割り当てメカニズムは、パーティションで繰返し処理を実行して、パーティション内のノードによって使用されるいずれかの中間通信リンクが、別個のＶＬを割り当てられていない別のパーティションと中間リンクを共有しているか否かを確認し得る。このような状況に遭遇すると、新たなＶＬが割り当てられ得る。ｐＦＴｒｅｅルーティングメカニズムは、ＶＬ選択のために２つのモード、すなわち厳格モードおよび通常モード、をサポートし得る。

【００６１】
実施形態によれば、厳格モードでは、ｐＦＴｒｅｅルーティングに必要とされるＶＬの数がシステム内の利用可能なＶＬを超えると、ルーティングは失敗する可能性がある（リスト４の第１０行）。

【００６２】
実施形態によれば、通常モードでは、アルゴリズムはパーティションへのＶＬの割り当てをVL_1から再開し得る（リスト４の第８行）。

【００６３】
実施形態によれば、ＩＢベースのファットツリーネットワークのための効率的なパーティション認識ルーティングメカニズム（別名でｐＦＴｒｅｅと称される）が提供される。ｐＦＴｒｅｅメカニズムは、ファットツリートポロジのためのパーティションのネットワーク全体にわたる分離を提供することができる。また、ｐＦＴｒｅｅは、スイッチのためにバランスのとれたＬＦＴを生成する。ネットワークリソースが十分であると仮定すると、ｐＦＴｒｅｅは、もっぱら物理リンクレベルでパーティションを分離することができる。たとえば、ファットツリーが２個の重なり合わない等しいサイズのパーティションを有している場合には、ｐＦＴｒｅｅは、ルーティング自体に基づいて、中間ネットワークリンクを２個の等しいサイズの論理サブネットワークに分割することができる。さらに、ネットワークが完全なパーティション分離を提供するのに十分な利用可能なリソースを持たない場合には、ｐＦＴｒｅｅは、物理的分離と連動して機能する相補的なＶＬベースの分離スキームを利用することができる。

【００６４】
実施形態によれば、ｐＦＴｒｅｅルーティングメカニズムは、２つの主な目的を達成することを目指している。第一に、当該メカニズムは、ツリー内のリンク全体にわたって均一に経路を分散させることによってファットツリートポロジのためにバランスのとれたＬ

ＦＴを生成することができる。第二に、ｐＦＴｒｅｅは、リンク上の経路をバランスのとれた状態に維持しながら、異なるパーティションに属するパス間の干渉を取り除く。ｐＦＴｒｅｅは、サブネットについてのパーティション化情報を使用することができ、パーティション内のノードが、他のパーティションでのワークロードの影響を受けない予測可能なネットワーク性能を受け取ることを確実にする。トポロジが（負荷バランシングに関して妥協することなく）各レベルにおいてパーティション分離を提供するのに十分な利用可能なリンクを持たない場合、ｐＦＴｒｅｅは、ＶＬを使用して干渉の影響を軽減することができる。

【００６５】

実施形態によれば、ｐＦＴｒｅｅメカニズムは、各エンドノードに関連付けられるＬＩＤについて全ての関連するスイッチ上でＬＦＴをセットアップするように再帰的に機能し得る。単一リーフスイッチパーティションをフィルタリングして除去した後、当該メカニズムは、各リーフスイッチについて、接続されたエンドノードをパーティション化に特有の順序でソートし得る。この順序付けは、リーフスイッチで利用可能な上りポートの数を考慮に入れてノードがそれらのパーティションに従ってルーティングされることを確実にする。各レベルでのポートの選択は、負荷が利用可能なパス全体にわたって分散されることを確実にするように、最小数の既に割り当てられている経路に基づき得る。しかし、いくつかのポートが同一の負荷で利用可能である場合、当該関数は、これらの最小負荷ポートで繰返し処理を実行して、ルーティングされているノードのパーティションキーで既にマーキングされているスイッチに接続されているポートを選択する。マーキングされているスイッチがない場合（すなわち、特定のパーティションの第１のノードのルーティング）、ｐＦＴｒｅｅは、最も高いグローバル一意識別子（ＧＵＩＤ）を有するポートのデフォルト選択の状態になり得る。パーティションにとって初めてスイッチが選択されると、当該スイッチはパーティションキーでマーキングされ得る。このようにして、当該メカニズムは、十分なパスがバランシングに利用可能であると仮定して、１つのパーティションに属するノードが同一のスイッチおよび対応するリンクを介してルーティングされることを確実なものとする助けとなるであろう。パーティション分離基準を保持しながらルーティングテーブルが生成されると、当該メカニズムは、リンクのうちのいくつかが異なるパーティション内のノードに向かうフローに使用されているか否かを確認することに進み得る。そのような場合、当該メカニズムは、干渉しているパーティションにＶＬを割り当てて分離を提供することができる。

【００６６】

ここで、実施形態に係るマルチテナントクラスタ環境においてパーティション認識ルーティングをサポートすることの図である図５～図８を参照する。

【００６７】

実施形態によれば、ｐＦＴｒｅｅルーティングにおけるポート選択メカニズムは、オーバーサブスクライブされたファットツリーネットワークの単純なセクションによって図５～図８に示されている。

【００６８】

ここで、図５を参照する。図５は、４個のスイッチ、すなわちルートスイッチ５２５～５２６およびリーフスイッチ５２０～５２１と、８個のエンドノード、すなわちノードＡ～Ｇ　５０１～５０８とを有する２レベルファットツリートポロジを示す。同様に、エンドノードは２個のパーティションに分割されている。パーティション１は、ノードＡ　５０１とノードＤ　５０４とノードＧ　５０７とノードＨ　５０８とを備える。パーティション２は、ノードＢ　５０２とノードＣ　５０３とノードＥ　５０５とノードＦ　５０６とを備える。

【００６９】

実施形態によれば、図５に示されるように、例示的なセクションは、２個のリーフスイッチ（５２０および５２１）で構成され、各リーフスイッチは、４個のエンドノードと、リーフスイッチの次のレベルの２個のスイッチ、すなわちルートスイッチ（５２５および

５２６）とに接続されている。下り方向の割り当てられた経路の数を表わす変数ｄｏｗｎおよびｍａｘ、ならびに、各リンク上で適切なバランシングを確実にするようにルーティングされ得るノードの最大数もそれぞれ図に示されている。

【００７０】

実施形態によれば、ルーティングすべき４個のエンドノードを有する各リーフスイッチに２個の上りポートがあると仮定すると、アップリンクの各々は、リンクのバランスをとることを確実にするために２個のエンドノードを下方にルーティングすべきである（すなわち、ｍａｘ＝２）。

【００７１】

実施形態に従って、リーフスイッチ５２０についての、最初の２個のノード、すなわちノードＡおよびノードＢ、のルーティングが図６に示されている。ルーティングメカニズムは、ルートスイッチ５２５を選択して、ノードＡの方にトラフィックをルーティングし、図に「（パーティション１）」と示されているノードＡのパーティションキーで当該スイッチをマーキングし得る。同様に、ノードＢのために、ルートスイッチ５２６が選択され、図に「（パーティション２）」と示されているノードＢのパーティションキーでマーキングされ得る。また、２個の下りリンクの各々上の単一のルーティングされるノードを数えるように変数ｄｏｗｎも更新される。

【００７２】

実施形態によれば、図７に示されるように、ノードＣおよびＤのために、対応するパーティションキーで既にマーキングされているスイッチが選択され得る。結果として生じる経路は、ルートスイッチ５２５を介して同一のリンクによって、第１のパーティションに属するノード、すなわちノードＡおよびＤ、の方に流れる。同様に、第２のパーティションのノード、すなわちノードＢおよびＣ、がルートスイッチ５２６を介して下り方向にルーティングされ得る。この経路の分離により、２個のパーティションのトラフィックフロー間の干渉が回避される。なお、各リンクに沿って下り方向にルーティングされるノードの数はｍａｘ変数を超えることはなく、これはルーティングが依然としてバランスがとれていることを意味する。

【００７３】

最後に、実施形態に従って、図８は、リーフスイッチ５２１に接続されたエンドノードのためのルーティングを示す。ここでも、第２のレベルのスイッチが第１のリーフスイッチのルーティングからのパーティションキーで既にマーキングされているので、対応するスイッチが選択されて、ノード、すなわちノードＥ，Ｆ，ＧおよびＨ、の各々をルーティングすることができる。図に示されるように、最終的なルーティングは、ルーティングに基づいて中間ネットワークリンクを２個の等しいサイズの論理サブネットワークに分割することによって２個のパーティションを分離することができる。

【００７４】

拡張型ｐＦＴｒｅｅ

実施形態によれば、ネットワークがもっぱら物理リンクレベルでパーティションを分離するのに十分なリソースを持たない場合、ｐＦＴｒｅｅルーティングアルゴリズムは、ＶＬを使用してパーティション間の干渉を減少させる。しかし、対応するＳＬＡ（サービス水準合意）またはＱｏＳ要件によっては、パーティションが異なれば分離ニーズも異なる可能性がある。たとえば、ネットワーク内のパーティションのうちのいくつかは、重大な動作を実行しているかもしれず、いかなる場合でも完全な物理的分離を必要とするかもしれない。同様に、多くのネットワークでは、ＶＬの利用可能性によっては、いくつかのパーティションは別のパーティションとＶＬを共有しなければならないかもしれず、これは通信集中的なワークロードにとっては望ましくないであろう。上記のｐＦＴｒｅｅアルゴリズムは、ルーティングにおける上記のパーティション面の要件を指定することができず、全てのパーティションは、同様のＱｏＳ要件を想定して等しい優先度で処理される。

【００７５】

ここで、実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポ

ートすることの図である図９を参照する。より具体的に、図９は、３個の異なるテナントパーティション（さまざまなシェーディングによって図示）に９個のノード（すなわち、ノードＡ～Ｉ　９０１～９０９）を有するファットツリーネットワークの一例を示す。ノードＡ　９０１およびＥ　９０５はパーティション１に属し、ノードＢ　９０２、Ｆ　９０６、Ｇ　９０７およびＩ　９０９はパーティション２に属し、ノードＣ　９０３、Ｄ　９０４およびＨ　９０８はパーティション３に属している。また、ファットツリーネットワークは、ルートスイッチ９２５および９２６と、リーフスイッチ９２０，９２１および９２２とを含む。

【００７６】

実施形態によれば、パーティション１は非常に高いＱｏＳ要件を有し得て、このパーティションでのワークロードがいかなるパーティション間の干渉の影響も受けないことが極めて重要である。しかし、所与のファットツリーネットワークが３つの異なるテナントパーティションを有しながらルートスイッチを２つだけ有している、すなわちルートスイッチ９２５およびルートスイッチ９２６を有しているので、もっぱら物理レベルでこれらのパーティションを分離することは不可能である。上記のように、このような場合、ｐＦＴｒｅｅルーティングアルゴリズムは、ＶＬを使用してパーティションの分離を進めることができる。デフォルトのｐＦＴｒｅｅアルゴリズム（上記）を使用して得られるルーティングも図に示されており、ここではスイッチの直下の小さなノード円を使用して宛先ノードに向かうフローを示している。パーティション１のノードＡに向かうトラフィックは、現在のところ、ルートスイッチ９２５とリーフスイッチ９２０との間のリンクを、異なるパーティション（すなわち、パーティション３）に属するノードＣに向かうトラフィックと共有していることが分かる。同様に、パーティション１（高いＱｏＳを必要とするパーティション）のＥに向かうトラフィックは、パーティション２のノードＦに向かうトラフィックとリンクを共有している。ルーティングメカニズムは、分離を保つために、これらのリンクの各々上で別個の仮想レーンを利用することができる。しかし、ＶＬを使用することにより干渉が減少するとしても、このような干渉が完全に排除されることはない。

【００７７】

分離ポリシー

実施形態によれば、ｐＦＴｒｅｅルーティングメカニズムは、パーティションワイズの（partition-wise）分離ポリシーおよびグローバルな分離ポリシーを含むように拡張可能である。各パーティションについて、分離ポリシーは、どのようにしてパーティション内のノードが他のパーティションに属するノードとネットワークリソースを共有できるようにするかを決定し得る。グローバルなポリシーは、所与のネットワークについて全てのパーティションワイズの分離ポリシーを満たすことができない場合にルーティングが機能しなくなるかベストエフォート型分離を継続するかを判断し得る。

【００７８】

実施形態によれば、拡張型ｐＦＴｒｅｅルーティングメカニズムのためのさまざまなポリシーパラメータを提供することができる。各パーティションは、３個のパーティションワイズのポリシーパラメータのうちの１つでマーキングされ得る。パーティションをｐｈｙ－分離（本明細書では厳格なパラメータとも称される）でマーキングすることにより、ルーティングアルゴリズムが特に当該パーティションのためにネットワークリソースを取っておくことを保証することができ、当該パーティション内のノードは、異なるパーティション内のその他のノードといかなるリンクも共有しない。パーティションをｖｌａｎｅ－分離というパラメータ（本明細書では厳格な仮想レーンパラメータとも称される）でマーキングすることにより、マーキングされたパーティションは、別個のＶＬのみを使用して他のパーティションとネットワークリソースを共有することができる。パーティションをｄｅｆ－分離（本明細書ではベストエフォートパラメータとも称される）スキームでマーキングすることにより、マーキングされたパーティションのベストエフォート型分離が実現される。

【００７９】

実施形態によれば、ポリシーパラメータは、グローバルなポリシーパラメータも含み得る。グローバルなポリシーパラメータ、すなわち厳格なパラメータおよびベストエフォートパラメータ、は、所与のサブネットにおいてパーティションワイズのポリシーパラメータを満たすことができない場合にルーティングメカニズムが機能しなくなるかベストエフォート型ルーティングに戻るかを規定し得る。たとえば、ネットワークが所望の分離を提供するための十分なリンクまたはＶＬを持たない場合、パーティション構成ファイルを使用してポリシーパラメータがルーティングメカニズムに提供されてもよい。

【００８０】

拡張型ｐＦＴｒｅｅメカニズム

実施形態によれば、拡張型ｐＦＴｒｅｅルーティングメカニズム（本明細書では別名で「ｐＦＴｒｅｅ－Ｅｘｔ」とも称される）は、ファブリックを再帰的に横断して、各エンドノードに関連付けられるＬＩＤについて全てのスイッチにおいてＬＦＴをセットアップすることによって、（上記の）当初のｐＦＴｒｅｅと同様の態様で動作する。しかし、ｐＦＴｒｅｅとは異なって、ｐＦＴｒｅｅ－Ｅｘｔは、経路を割り当てる際に規定のグローバルな分離ポリシーおよびパーティションワイズの分離ポリシーも考慮に入れることができる。

【００８１】

実施形態に従って、ｐＦＴｒｅｅ－Ｅｘｔルーティングメカニズムの疑似コード（本明細書では別名でリスト５とも称される）を以下に示す。

【００８２】

【数５】

```
Ensure: The LFTs are generated for the switches conforming isolation
policies
1: global_param ⇐ get_global_isolation_policy()
2: partitions_info ⇐ get_partition_information()
3: ORDERCOMPUTENODES()
4: for each sw ∈ leafSwitches[] do
5:    for each cn ∈ computeNodes[] do
6:       Get lid of cn
7:       Get partition_key of the cn.hca_port
8:       Set LFT[lid] ⇐ cn:hca_port on sw
9:       ROUTEDOWNGOINGBYASCENDINg() on sw
10:    end for
11: end for
12: ASSIGNVIRTUALLANES()
13: VALIDATEPOLICIES()
```

【００８３】

実施形態によれば、当該メカニズムは決定論的であり、経路は宛先ノードから開始して後向きに計算される。当該メカニズムは、まず、コンピュートノードをパーティションに特有の順序でソートし得る（リスト５の第３行）。パーティションに特有の順序は、メカニズムのさらに高速な実行を確実にすることができる。なぜなら、ノードは、一旦順序付けられると、各々の下りおよび上りポート上に最大カウンタを維持することなく繰返しルーティングできるからである。以下のORDERCOPMUTENODESのための疑似コードに示されるように、各リーフスイッチについて、ORDERCOMPUTENODESは、まず、エンドノードをそれらのパーティションポリシー優先度の低い方から順にソートする（リスト６の第４行（以

下を参照））。ｐｈｙ－分離パラメータでマーキングされたパーティションに属するノードが最初に付加され得て、ｖｌａｎｅ－分離によるパーティションが２番目に付加され得る。最後に、ｄｅｆ－分離のポリシーパラメータ値を有するパーティションノードがコンピュートノードのリストに付加される。次いで、当該メカニズムは、ノードのパーティション化情報を使用してルーティング順序を生成し、当該ルーティング順序では、１つのパーティションに属するノードは、反復ルーティングの際にネットワーク内の同一の上りリンクを提案するインデックスを得る傾向がある。これは、パーティションキーテーブルを使用して、利用可能な上りポートの数を、パーティションに属する第１のノードをルーティングするように選択されたインデックスに付加することによってなされる（リスト６の第１４～２８行）。しかし、このようなインデックスが既に取得されている場合、または当該インデックスがコンピュートアレイ境界を超える場合、第１のフリーインデックスが選択されて、その後の選択のためにパーティションキーでマーキングされ得る（リスト６の第２４行）。ORDERCOMPUTENODESのための疑似コード（本明細書では別名でリスト６とも称される）をここに示す。

【００８４】

【数６】

```
Require: List of switches and attached compute nodes
Ensure: The compute nodes are ordered for the pFTree-Ext routing
algorithm
1: for each sw in leafswitches[] do
2:     num_up_ports ⇐ count(sw → upPorts[])
3:     num_cns ⇐ count(sw → computeNodes[])
4:     Sort nodes in increasing order of partition isolation policy
       (phy > vlane > def)
5:     if num_cns ≥ num_up_ports then
6:         return
7:     end if
8:     index arr[] = array(num_cns)
9:     taken[] = array(num_cns)
10:    pkey_tbl[] = map()
11:    id ⇐ 0
12:    for each cn in sw → computeNodes[] do
13:        pkey ⇐ cn → get_partition_key()
14:        if pkey not found in pkey_tbl then
15:            if taken[id] ≠ false then
16:                id ⇐ get_free_id()
17:            end if
18:            index_arr[cn[i]] ⇐ id
19:            taken[id] = true
20:            insert pkey in pkey_tbl
21:        else {pkey is already in pkey_tbl}
22:            id ⇐ id(pkey) + num_up_ports
23:            if id ≥ num_cns or taken[id] = true then
24:                id ⇐ get_free_id()
25:            end if
26:            index_arr[cn[i]] ⇐ id
27:            taken[id] = true
28:            update pkey_tbl
29:        end if
30:    end for
31:    Sort sw → computeNodes[] with respect to index_arr[]
32: end for
```

【００８５】

実施形態によれば、ノードが適切に順序付けられると、ROUTEDOWNGOINGBYASCENDINGのための以下の疑似コード（本明細書では別名でリスト７とも称される）に例示されるよう

に、ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、ROUTEDOWNGOINGBYASCENDING（リスト５の第９行）を呼び出して、ツリーを上って次のレベルのポートを選択して、ＬＩＤを下り方向にルーティングし得る。

【００８６】

【数７】

```
Require: A switch sw, an end node lid and partition_key
1: Sort sw.upPorts[] with increasing load and then GUID
2: Get least loaded ports as leastLoadedList[]
3: partition_param ⇐ get_isolation_policy(partition_key)
4: selected_port ⇐ null
5: for each port in leastLoadedList[] do
6:     r_sw ⇐ port.get_remote_switch()
7:     if r_sw is marked with partition_key then
8:         selected_port ⇐ port
9:         break
10:     end if
11: end for
12: if selected_port = null then
13:    while selected_port = null do
14:        port ⇐ sw.upPorts[].get_next()
15:        r_sw ⇐ port.get_remote_switch()
16:        if r_sw is marked with a partition with isolation policy >
           partition_param then
17:            continue
18:        end if
19:        selected_port ⇐ port
20:    end while
21: end if
22: Set LFT[lid] ⇐ selected_port on r_sw
23: if r_sw is not marked then
24:    Mark it with partition_key in DWN direction
25: end if
26: ROUTEUPGOINGBYDESCENDING() on sw
27: ROUTEDOWNGOINGBYASCENDING() on r_sw
```

【００８７】

実施形態によれば、ポートの選択は、まず、ソートされた利用可能な上りポートから得られる最小負荷ポートリストに基づく（リスト７の第１～２行）。当該関数は、これらの最小負荷ポートで繰返し処理を実行して、ルーティングされているノードのパーティションキーで既にマーキングされているスイッチに接続されているポートを選択する（リスト７の第５～１１行）。マーキングされているスイッチがないことが分かると、アルゴリズムは、上りポートで繰返し処理を実行して、ＬＩＤに適した経路を見つける。上りポート

リストは、ポートに対する現在の負荷の低い方から順にソートされる。同一の負荷を有するポートでは、決定論的なままであるようにするために、ソートはそれらのグローバル一意識別子（ＧＵＩＤ）の高い方から順になされる。さらに、当該関数は、ルーティングされるノードよりも高い分離ポリシーパラメータを有するパーティションキーで既にマーキングされているポートを選択しない（リスト７の第１６～１７行）。最後に、ポートが選択されると、対応するスイッチがパーティションキーで下り方向にマーキングされる（リスト７の第２４行）。

【００８８】

実施形態によれば、スイッチにおけるＬＩＤに対して下りポートが設定されると、ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、ROUTEUPGOINGBYDESCENDING（本明細書では別名でリスト８とも称される）を呼び出すツリーを下ることによって全ての接続された下りスイッチ上でそれに対して上りポートを割り当てる。ROUTEUPGOINGBYDESCENDINGのための疑似コードをここに示す。

【００８９】

【数８】

```
Require: A switch sw, an end node lid and partition_key
1: Get least-loaded ports from sw.dwnPorts[] as dwnlist[]
2: selected_port ⇐ dwnList.get port_max_guid()
3: for each port in dwnList[] do
4:     r_sw ⇐ port.get_remote_switch()
5:     if r_sw is marked with partition_key then
6:         selected_port ⇐ port
7:         break
8:     end if
9: end for
10: if r_sw is not marked then
11:    Mark it with partition_key in UP direction
12: end if
13: Set LFT[lid] ⇐ selected_port on r_sw
14: ROUTEUPGOINGBYDESCENDING() on r_sw
```

【００９０】

実施形態によれば、上りポートの選択は、まず負荷基準に基づき、次いで今回は上り方向へのリモートスイッチのパーティションマーキングに基づく。次いで、全てのＬＦＴが設定されるまで、次のレベルまでツリーを上ることによってプロセスが繰返される。なお、スイッチは複数のパーティションキーでマーキングされてもよい。ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、各パーティションについてルーティングされるノードの個数を記憶するテーブルを各スイッチについて維持する。マーキングされたパーティションを有するいくつかのスイッチがノードのルーティングに利用できる場合には、このカウンタを使用してポートの選択を決定する。パーティションについて最大数の既にルーティングされているノードを有するスイッチが選択される。

【００９１】

実施形態によれば、パーティション分離基準を保持しながらルーティングテーブルが生成されると、ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、リンクのうちのいくつかが異なるパーティション内のノードに向かうフローに使用されているか否かを確認することに進む。そのような場合、ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、干渉しているパーティションにＶＬ

を割り当てて分離を提供することができる。ＶＬ割り当て関数、すなわちASSIGNVIRTUALLANES、のための疑似コード（本明細書では別名でリスト９と称される）を以下に示す。

【００９２】

【数９】

```
Require: The pFTree-Ext routing tables have been generated
Require: Switches have been marked with the partition keys
Require: Global policy parameter, strict or best-effort
Ensure: A partitions marked with vl-isolation has a separate VL
Ensure: No two partitions with the same SL share a link
1: vlanes_needed ⇐ 1
2: max_vlanes ⇐ get_max_lanes()
3: strict ⇐ get_is_strict()
4: for each partition in partition_tbl do
5:     check if the isolation policy of the partition is vl-isolation
and
       any intermediate communication link in this partition share a
       switch with a partition that has not been assigned a virtual
lane
6:     if require a separate vl then
7:        if vlanes_needed = max_vlanes and global_param = strict then
8:            vlanes_needed ⇐ 1
9:        else
10:           error: routing failed
11:           return
12:       end if
13:       vlanes_needed++
14:       partition.vlane ⇐ vlanes_needed
15:   end if
16: end for
```

【００９３】

実施形態によれば、仮想レーン割り当て関数は、全てのパーティションで繰返し処理を実行して、パーティションがｖｌ－分離ポリシーパラメータでマーキングされているか否かおよびパーティション内のノードによって使用されるいずれかの中間通信リンクが、別個のＶＬを割り当てられていない別のパーティションと中間リンクを共有しているか否かを確認し得る。そうであれば、新たなＶＬが割り当てられる。また、ＶＬ割り当て関数は、２つのモード、すなわち厳格モードおよびベストエフォートモード、を有するグローバルなポリシーパラメータを使用し得る。厳格モードでは、ｐＦＴｒｅｅ－Ｅｘｔルーティングメカニズムに必要とされるＶＬの数がシステム内の利用可能なＶＬの数を超えると、ルーティングは失敗する（リスト９の第１０行）。ベストエフォートモードでは、仮想レーン割り当て関数はパーティションへのＶＬの割り当てをVL_1から再開し得る（リスト９の第８行）。

【００９４】

実施形態によれば、ｐＦＴｒｅｅ－Ｅｘｔルーティングメカニズムは、全ての利用可能なＶＬではなくＶＬの特定のグループを考慮に入れるように容易に変更することができる。同様に、より高い分離ポリシーを有するパーティションがＶＬを共有しにくくするため

に、全ての利用可能なＶＬが使用されると、（$ＶＬ_1$を選択する代わりに）割り当てられたパーティションの優先度を選択のために減少させることによってＶＬリストを順序付けることができる。ＶＬが割り当てられた後、ｐＦＴｒｅｅ－Ｅｘｔルーティングアルゴリズムは、全てのパーティションワイズのポリシーおよびグローバルなポリシーが満たされているか否かを確認する（リスト５の第１３行）。

【００９５】

実施形態によれば、ｐＦＴｒｅｅ－Ｅｘｔメカニズムは、分離ポリシーをルーティングアルゴリズムに組み込むことができる。各リーフスイッチについてルーティングの前にエンドノードをパーティションに特有の順序でソートするｐＦＴｒｅｅとは異なって、ｐＦＴｒｅｅ－Ｅｘｔルーティングメカニズムは、まず、エンドノードをそれらのパーティション優先度の順序でソートする。ｐｈｙ－分離でマーキングされたパーティション内のエンドノードは、最大の優先度を得る。その後、当該メカニズムは、上記のようにエンドノードをパーティションに特有の順序でソートすることに取りかかる。最も高いパーティション優先度を有するノードが最初にルーティングされることを確実にするために、付加的なソートが事前になされる。

【００９６】

実施形態によれば、ｐＦＴｒｅｅ－Ｅｘｔメカニズムはさらに、新たなノードをルーティングするためのポートの選択方法を変更することによって分離ポリシーをルーティングアルゴリズムに組み込むことができる。たとえば、いくつかの候補ポートの中から下りポートを選択するために、ｐＦＴｒｅｅ－Ｅｘｔは、ポートに対する現在の負荷を確認することに加えて、対応するスイッチが現在ルーティングされているノードのパーティションよりも高い優先度を有するパーティションのキーで既にマーキングされている任意のポートグループを除去する。

【００９７】

実施形態によれば、および加えて、利用可能なネットワークリソースがパーティションワイズのポリシーパラメータを満たすことを許可しない場合、ｐＦＴｒｅｅ－Ｅｘｔルーティングメカニズムは、機能しなくなるかまたはグローバルなポリシーパラメータに従って続行するかのいずれかであり得る。その場合、当初のｐＦＴｒｅｅルーティングアルゴリズムは、利用可能なＶＬを考慮に入れるのみである。

【００９８】

ここで、実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポートすることの図である図１０を参照する。より具体的に、図１０は、３個の異なるテナントパーティション（さまざまなシェーディングによって図示）に９個のノード（すなわち、ノードＡ～Ｉ　９０１～９０９）を有するファットツリーネットワークの一例を示す。ノードＡ　９０１およびＥ　９０５はパーティション１に属し、ノードＢ　９０２、Ｆ　９０６、Ｇ　９０７およびＩ　９０９はパーティション２に属し、ノードＣ　９０３、Ｄ　９０４およびＨ　９０８はパーティション３に属している。また、ファットツリーネットワークは、ルートスイッチ９２５および９２６と、リーフスイッチ９２０，９２１および９２２とを含む。

【００９９】

実施形態によれば、図１０は、ｐＦＴｒｅｅ－Ｅｘｔメカニズムを使用したサブネットルーティングを示しており、当該メカニズムでは、パーティション１（すなわち、ノードＡ　９０１およびノードＥ　９０５）はｐｈｙ－分離などの高い優先度でマーキングされており、ルーティングメカニズムが特に当該パーティションのためにネットワークリソースを取っておくことを保証することができ、当該パーティション内のノードは、異なるパーティション内のその他のノードといかなるリンクも共有しない。結果として生じるルーティングが図１０に示されている。パーティション１がｐｈｙ－分離などの高い優先度でマーキングされているので、パーティション１のノード（すなわち、ノードＡおよびＥ）はいずれも、その他のパーティションとリンクを共有しない。しかし、このようなポリシーはパーティション２および／またはパーティション３には適用されなかったので、これ

らのパーティションはスイッチ９２６からの全ての下りリンクを共有する。

【０１００】

重み付けされたｐＦＴｒｅｅルーティングメカニズム

実施形態によれば、ｐＦＴｒｅｅルーティングメカニズムの第２の拡張機能は、サブネットにおけるトラフィック特性の重みを説明することができる。これは、重み付けされたｐＦＴｒｅｅルーティングメカニズム（ｐＦＴｒｅｅ－Ｗｔ）と称されることができる。ｐＦＴｒｅｅ－Ｗｔは、各コンピュートノードに関連付けられる重みの概念に基づく。これらの重みを使用して、経路を計算する際に既知のまたは学習したトラフィック特性を考慮に入れる。パーティション化にかかわらず、ノードの重みは、ルーティングテーブルを計算する際にノードに向かうフローが受ける優先度を反映する。たとえば、考えられる構成は、ネットワーク内でどれぐらいのトラフィックをノードが受けることが分かっているかに応じて［１，１００］の範囲内の重みをノードに割り当てるというものであり得る。このようなスキームでは、ほとんどトラフィックを受けないノード（たとえば、主にトラフィックジェネレータ）に対しては重み＝１を割り当て、リンク容量に近いトラフィックを受けるノードに対しては重み＝１００を割り当てることができる。その結果、中間の値、すなわち１＜ｘ＜１００は、ネットワーク内でノードが受けることが見込まれるトラフィックの割合を反映することができる。

【０１０１】

実施形態によれば、コンピュートノードについての管理情報が利用できない場合には、ポートデータカウンタベースのスキームを使用して重みを計算することができる。たとえば、ＯＦＥＤ（オープン・ファブリック・エンタープライズ・ディストリビューション）では、データカウンタを読取るためにibdatacountと呼ばれるユーティリティが提供される。全てのノードについて初期重みが等しい状態でネットワークをセットアップした後、所定の期間が経過すると新たな重みを学習することができる。

【０１０２】

実施形態によれば、Ｂがある期間にわたって測定される全てのノードの受信帯域幅セットを表わすとすると、各ノードの重みは、以下のように、線形変換を使用して［ａ，ｂ］の範囲内で割り当てられることができる。

【０１０３】

【数１０】

$$W(x) = (x - a)\frac{b - a}{\max(B) - \min(B)} + a, \forall x \in B$$

【０１０４】

実施形態によれば、ｐＦＴｒｅｅ－Ｗｔルーティングメカニズムを使用して、各コンピュートノードは、重みというパラメータを割り当てられることができる。ポートに対する負荷が上りおよび下り方向へのノードに向かう割り当てられた経路の数を表わす当初のｐＦＴｒｅｅルーティングとは異なって、ｐＦＴｒｅｅ－Ｗｔルーティングスキームにおけるポートに対する負荷は、当該ポートから各方向にルーティングされるコンピュートノードの累積重みである。各リーフスイッチについて、１つのパーティション内のノードは、ルーティングの前にそれらの重みによってもソートされる。コンピュートノードをルーティングするためにスイッチにおける下りポートが選択されると、ｐＦＴｒｅｅ－Ｗｔは、対応するコンピュートノードの重みを付加することによって、選択されたポートに対する現在の負荷を更新する。同様に、上りリンクでは、各ポートに対して上り方向の負荷が維持される。ポート選択基準はｐＦＴｒｅｅルーティングと同様であり、ノードのパーティションも考慮する。しかし、ポートカウンタとは異なって、ｐＦＴｒｅｅ－Ｗｔにおける各レベルでのポートの選択は、全ての利用可能なポートに対する最小累積重みに基づく。いくつかのポートが同一の負荷で利用可能である場合、当該メカニズムは、これらの最小負荷ポートで繰返し処理を実行して、ルーティングされているノードのパーティションキーで既にマーキングされているスイッチに接続されているポートを選択する。ルーティン

グテーブルが生成されると、ｐＦＴｒｅｅ－Ｗｔは、ＶＬ割り当てを実行して、ネットワーク内でリンクを共有している異なるパーティションに関連付けられたノードに異なるＶＬが割り当てられることを確実にすることができる。

【０１０５】

図１１は、実施形態に係るマルチテナントクラスタ環境において重み付けされたパーティション認識ルーティングをサポートすることの図である。具体的に、図１１は、４個のスイッチ、すなわちルートスイッチ１１２５～１１２６およびリーフスイッチ１１２０～１１２１と、８個のエンドノード、すなわちノードＡ～Ｇ　１１０１～１１０８とを有する２レベルファットツリートポロジを示す。同様に、エンドノードは２個のパーティションに分割されている。パーティション１は、ノードＡ　１１０１とノードＤ　１１０４とノードＧ　１１０７とノードＨ　１１０８とを備える。パーティション２は、ノードＢ　１１０２とノードＣ　１１０３とノードＥ　１１０５とノードＦ　１１０６とを備える。

【０１０６】

実施形態によれば、図１１における各ノードは、重みを割り当てられている。ノードＡ　１１０１は１００という重みを割り当てられている一方、残りのノードは１という重みを割り当てられている。ｐＦＴｒｅｅ－Ｗｔを使用したリーフスイッチ１１２０への下りルーティングが図１１に示されている。リーフスイッチ１１２０に接続されたノードをルーティングする際、スイッチ１１２５および１１２６にそれぞれ接続された２個の上りポートが利用可能である。ノードＡは、１００に等しい重みを有しているので、それらのリンクのうちの１つ、すなわちスイッチ１１２５→スイッチ１１２０を割り当てられる一方、他の３個のノードは他のリンク、すなわちスイッチ１１２６→スイッチ１１２０を共有する。その理由は、他の３個のノードの重みの合計が１００よりも低い３に過ぎないからである。たとえ選択されたスイッチがパーティションキーでマーキングされたとしても、重み付けされたパーティション認識ルーティングにより、依然としてサブネットにおいてパーティションを分離することはできない。

【０１０７】

図１２は、実施形態に係るマルチテナントクラスタ環境において重み付けされたパーティション認識ルーティングをサポートすることの図である。具体的に、図１２は、４個のスイッチ、すなわちルートスイッチ１１２５～１１２６およびリーフスイッチ１１２０～１１２１と、８個のエンドノード、すなわちノードＡ～Ｇ　１１０１～１１０８とを有する２レベルファットツリートポロジを示す。同様に、エンドノードは２個のパーティションに分割されている。パーティション１は、ノードＡ　１１０１とノードＤ　１１０４とノードＧ　１１０７とノードＨ　１１０８とを備える。パーティション２は、ノードＢ　１１０２とノードＣ　１１０３とノードＥ　１１０５とノードＦ　１１０６とを備える。

【０１０８】

実施形態によれば、図１２における各ノードは、重みを割り当てられている。ノードＡ　１１０１は１００という重みを割り当てられている一方、残りのノードは１という重みを割り当てられている。ｐＦＴｒｅｅ－Ｗｔを使用したリーフスイッチ１１２１への下りルーティングが図１２に示されている。リーフスイッチ１１２０へのルーティングとは異なって、リーフスイッチ１１２１に接続された各ノードは、同一の重み（すなわち、１）を有している。このため、パーティションはリンク上で分離されたままである可能性がある。同一のパーティションに属するノードＧおよびＨは、スイッチ１１２５→スイッチ１１２１間のリンクを介して下り方向にルーティングされることができる。等しい重みを有するパーティション２のノードＥおよびＦは、スイッチ１１２６→スイッチ１１２１間のリンクを介して下り方向にルーティングされることができる。

【０１０９】

実施形態によれば、ｐＦＴｒｅｅ－Ｗｔは、パーティションが分離した状態をできるだけ保持しながらリンク上の重み付けされた負荷バランシングを満足させる。なお、最終的なルーティングは、図１２に示されるように、２個のパーティションのノードによって共有されるリンクを１つだけ有する。

【０１１０】
図１３は、実施形態に係るマルチテナントクラスタ環境においてネットワーク分離をサポートするための方法のフローチャートである。
【０１１１】
ステップ１３０１において、当該方法は、マルチテナントクラスタ環境内の１つ以上のテナントをサポートし得る。
【０１１２】
ステップ１３０２において、当該方法は、１つ以上のテナントの各々を複数のパーティションのうちのあるパーティションに関連付け得る。
【０１１３】
ステップ１３０３において、当該方法は、複数のパーティションの各々を複数のノードのうちの１つ以上のノードに関連付け得て、複数のノードの各々は、複数のスイッチのうちのリーフスイッチに関連付けられ、複数のスイッチは、複数のリーフスイッチと、別のレベルにおける少なくとも１つのスイッチとを備える。
【０１１４】
ステップ１３０４において、当該方法は、複数のパーティションの各々を複数のポリシーパラメータのうちのあるポリシーパラメータでマーキングし得る。
【０１１５】
ステップ１３０５において、当該方法は、複数のノードの各ノードにパーティション化順序を割り当て得て、パーティション化順序は、少なくとも各ノードに関連付けられるパーティションにマーキングされたポリシーパラメータに基づく。
【０１１６】
ステップ１３０６において、当該方法は、少なくとも複数のパーティションのうちのパーティションのマーキングに基づいて、マルチテナントクラスタ環境で使用される１つ以上のリニアフォワーディングテーブルを生成し得る。
【０１１７】
図１４は、本発明の実施形態を実現することができる例示的なコンピュータシステム１４００を示すブロック図である。コンピュータシステム１４００は、バス１４９０を介して電気的に結合され得るハードウェア要素を備えるものとして示されている。ハードウェア要素は、１つ以上の中央処理ユニット１４１０と、１つ以上の入力デバイス１４２０（たとえば、マウス、キーボードなど）と、１つ以上の出力デバイス１４３０（たとえば、ディスプレイデバイス、プリンタなど）とを含んでいてもよい。
【０１１８】
コンピュータシステム１４００は、１つ以上の記憶装置１４４０も含んでいてもよい。一例として、記憶装置１４４０は、ランダムアクセスメモリ（「ＲＡＭ」）および／またはリードオンリメモリ（「ＲＯＭ」）などの、ディスクドライブ、光学式記憶装置、ソリッドステート記憶装置であってもよく、これらはプログラム可能、フラッシュ更新可能などであってもよい。記憶装置は、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、フラッシュメモリデバイス、磁気もしくは光カード、（分子メモリＩＣを含む）ナノシステム、または、命令および／もしくはデータを格納するのに適した任意のタイプの媒体もしくはデバイスを含み得るが、それらに限定されるものではない。
【０１１９】
また、コンピュータシステム１４００は、コンピュータ読取可能記憶媒体読取装置１４５０と、通信システム１４６０（たとえば、モデム、ネットワークカード（無線または有線）、赤外線通信デバイス、ブルートゥース（登録商標）デバイス、セルラー通信デバイスなど）と、上記のＲＡＭおよびＲＯＭデバイスを含み得るワーキングメモリ１４８０とを含んでいてもよい。いくつかの実施形態では、コンピュータシステム１４００は、デジタル信号プロセッサ、特殊用途プロセッサなどを含み得る処理加速ユニット１４７０も含

んでいてもよい。
【０１２０】
コンピュータ読取可能記憶媒体読取装置１４５０はさらに、コンピュータ読取可能な記憶媒体に接続可能であり、ともに（および任意に、記憶装置１４４０と組み合わせて）コンピュータ読取可能な情報を一時的におよび／またはより永久的に収容するためのリモートの、ローカルな、固定式のおよび／またはリムーバブルな記憶装置プラス記憶媒体を包括的に表わす。本発明の特徴は、機械読取可能な媒体のいずれか１つに格納されて、ソフトウェアおよび／またはファームウェアに組み込まれてもよく、当該ソフトウェアおよび／またはファームウェアは、コンピュータシステムのハードウェアを制御して、本発明の結果を利用する他のメカニズムとコンピュータシステムとを対話させるためのものである。こ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、デバイスドライバ、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が、それらに限定されるものではない。通信システム１４６０は、上記のネットワーク、システム、コンピュータおよび／または他の構成要素とデータをやりとりすることを可能にし得る。
【０１２１】
コンピュータシステム１４００は、ソフトウェア要素も備えていてもよく、当該ソフトウェア要素は、オペレーティングシステム１４８８および／または他のコード１４８４を含むワーキングメモリ１４８０内に現在のところ置かれているものとして示されている。コンピュータシステム１４００の代替的な実施形態は上記のものに対する多数の変形例を有していてもよいということが理解されるべきである。たとえば、カスタマイズされたハードウェアも使用されてもよく、および／または、特定の要素がハードウェア、（アプレットなどのポータブルソフトウェアを含む）ソフトウェア、またはそれら両方で実現されてもよい。さらに、ネットワーク入出力デバイスおよびデータ取得デバイスなどの他のコンピューティングデバイスとの接続も行われてもよい。
【０１２２】
コンピュータシステム１４００のソフトウェアは、本明細書に記載されているアーキテクチャのさまざまな要素のいずれかまたは全ての機能を実現するためのコード１４８４を含んでいてもよい。たとえば、システム１４００などのコンピュータシステムに格納され、および／または、システム１４００などのコンピュータシステムによって実行されるソフトウェアが、上記のような本発明の機能および／または他の構成要素を提供してもよい。これらの構成要素のうちのいくつかのソフトウェアによって実現可能な方法については、より詳細に上記している。
【０１２３】
また、本発明の特徴は、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で実現されてもよい。本明細書に記載されている機能を実行するためのハードウェア状態機械の実現は、当業者に明らかであろう。
【０１２４】
本発明のさまざまな実施形態を上記してきたが、それらは限定としてではなく一例として提示されているということが理解されるべきである。本発明の精神および範囲から逸脱することなく形態および詳細のさまざまな変更を行ってもよいということは当業者に明らかであろう。
【０１２５】
特定の機能の実行およびそれらの関係を示す機能的構成要素を用いて本発明を上記してきた。これらの機能的構成要素の範囲は、説明の便宜上、本明細書では多くの場合任意に規定されている。特定の機能およびそれらの関係が適切に実行される限り、代替的な範囲が規定されてもよい。したがって、いかなるこのような代替的な範囲も本発明の範囲および精神の範囲内である。
【０１２６】
本発明の上記の説明は、解説および説明の目的で提供されている。本発明の上記の説明

は、網羅的であることを意図したものではなく、本発明を開示されている厳密な形態に限定することを意図したものでもない。本発明の幅および範囲は、上記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はない。多くの変更例および変形例が当業者に明らかであろう。当該変更例および変形例は、開示されている特徴のいずれかの関連性のある組み合わせを含む。本発明の原理およびその実際の適用を最良に説明するために実施形態が選択されて説明されており、それによって、さまざまな実施形態についておよび意図される特定の使用に適したさまざまな変更例とともに当業者は本発明を理解することができる。本発明の範囲は以下の特許請求の範囲およびそれらの等価物によって規定されることが意図される。

【図１】

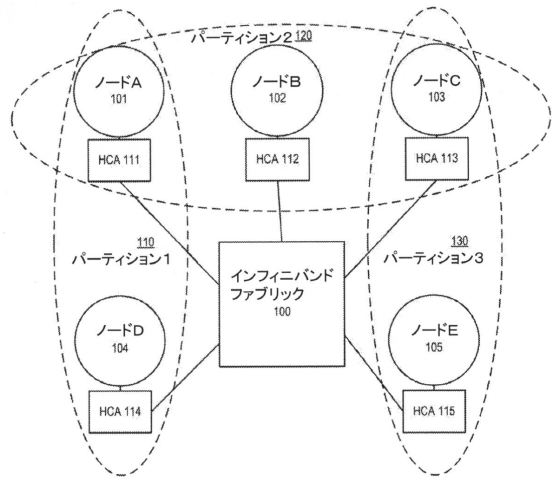

FIGURE 1

【図２】

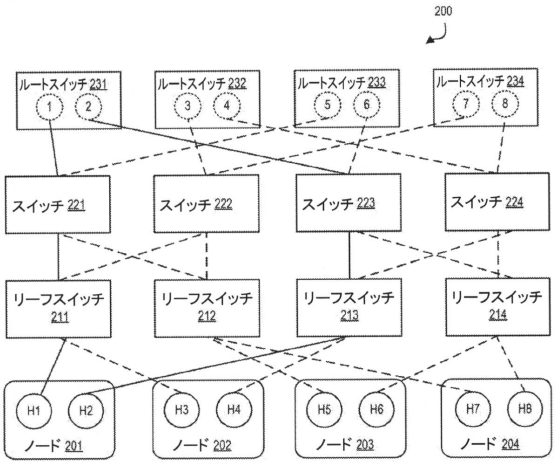

FIGURE 2

【図３】

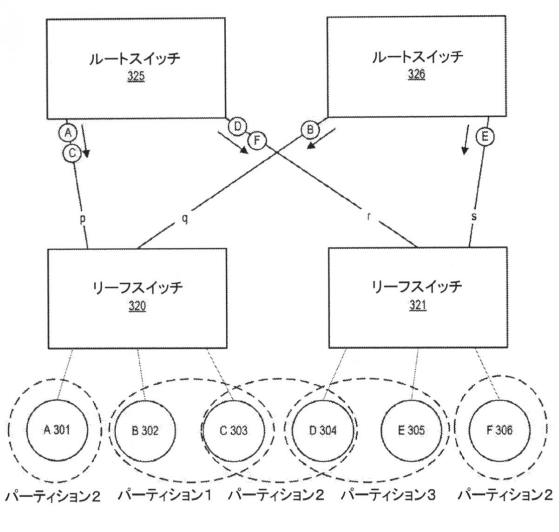

FIGURE 3

【図４】

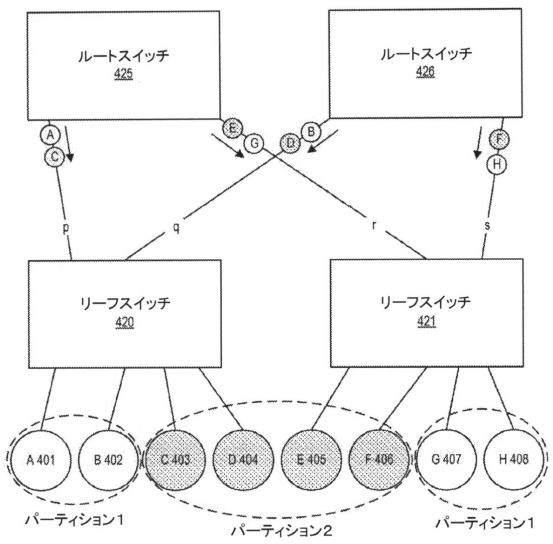

FIGURE 4

【図５】

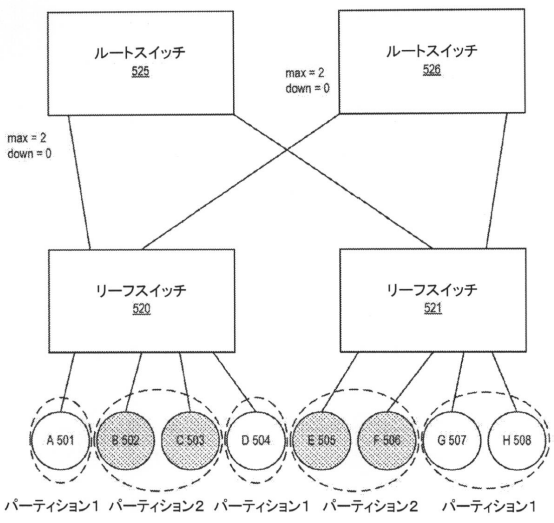

FIGURE 5

【図６】

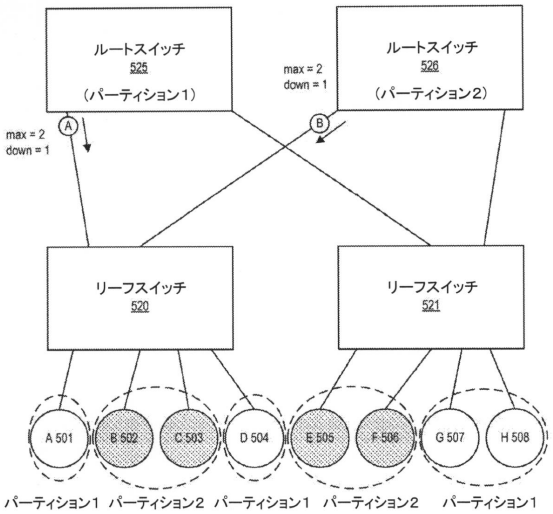

FIGURE 6

【図７】

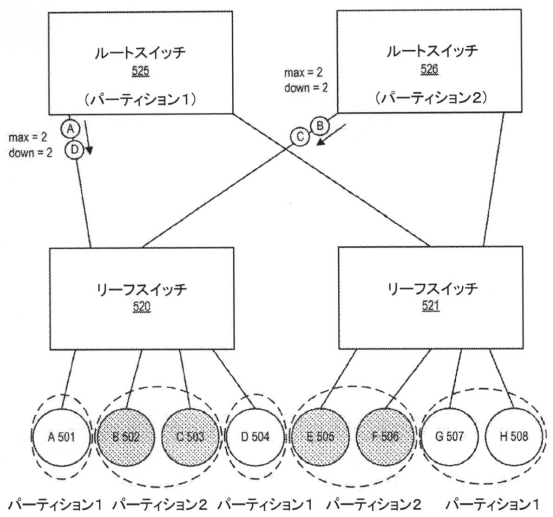

FIGURE 7

【図８】

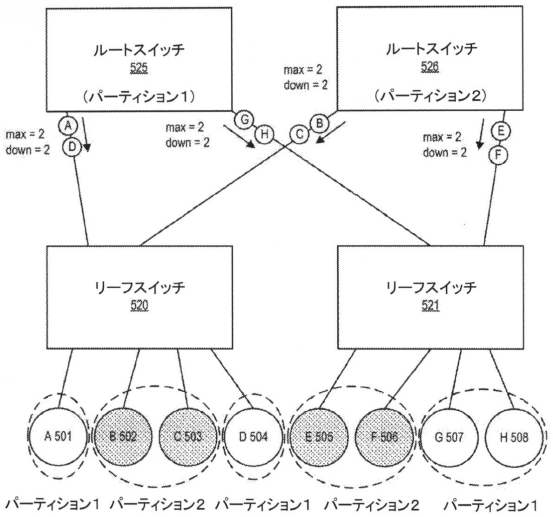

FIGURE 8

【図９】

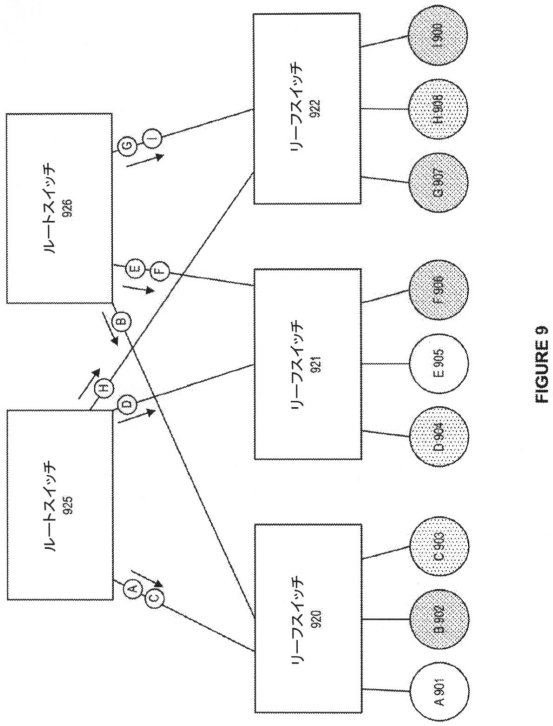

FIGURE 9

【図１０】

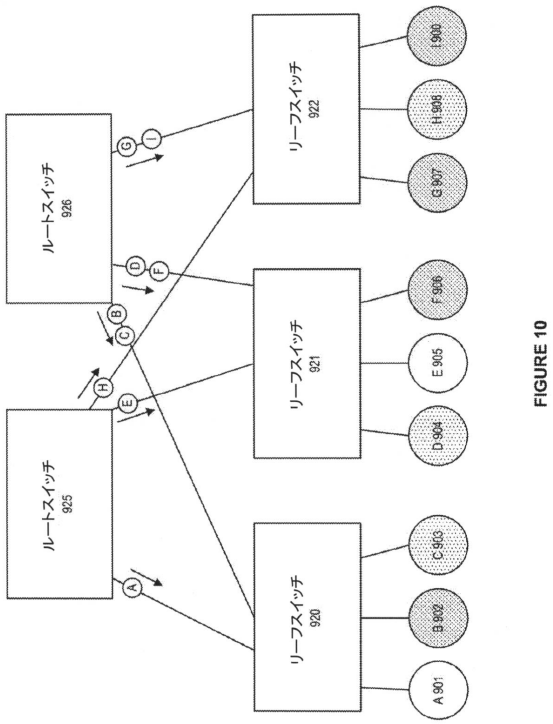

FIGURE 10

【図１１】

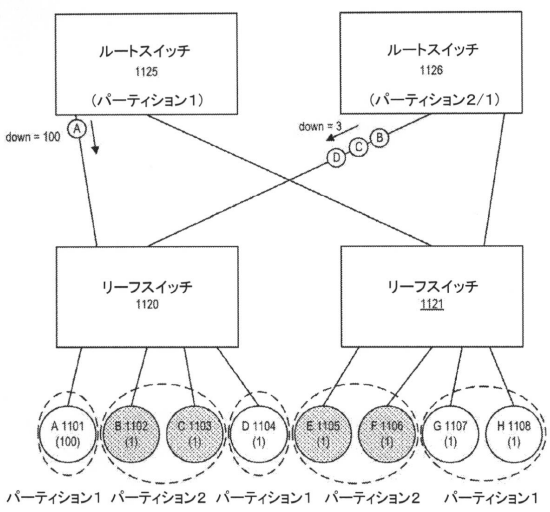

FIGURE 11

【図１２】

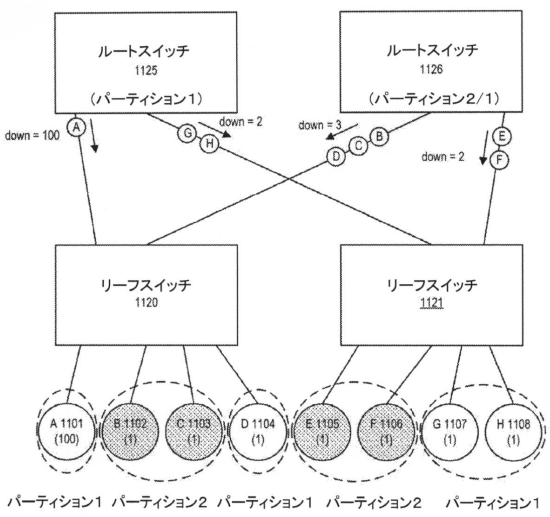

FIGURE 12

【図１３】

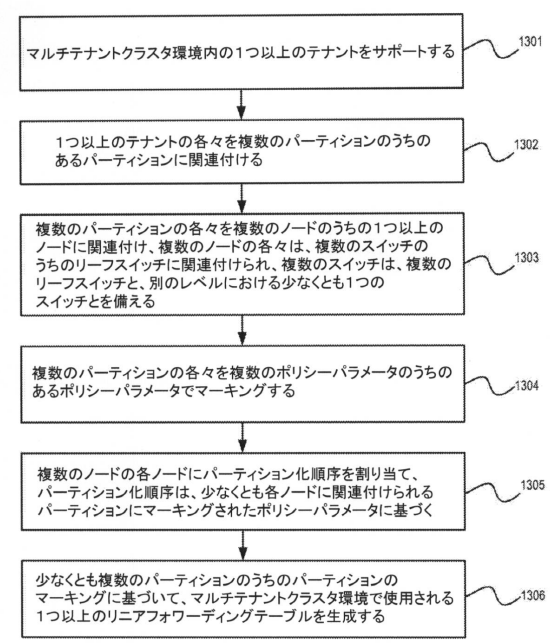

FIGURE 13

【図１４】

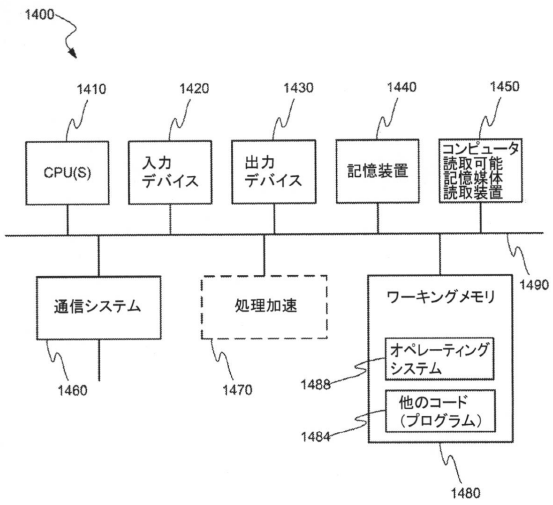

FIGURE 14

フロントページの続き

(31)優先権主張番号　15/182,383
(32)優先日　平成28年6月14日(2016.6.14)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　15/182,397
(32)優先日　平成28年6月14日(2016.6.14)
(33)優先権主張国・地域又は機関
米国(US)

(72)発明者　ボグダンスキー，バルトシュ
ノルウェー、エヌ－０２７５　オスロ、ホフ・テラッセ、１５、エイチ・０２０３
(72)発明者　ヨンセン，ビョルン・ダグ
ノルウェー、エヌ－０６８７　オスロ、ビルベルクグレンダ、９

審査官　安藤　一道

(56)参考文献　国際公開第２０１５／０２０６６５（WO，Ａ１）
国際公開第２０１３／０７１３３０（WO，Ａ１）
国際公開第２０１５／１３０３７２（WO，Ａ１）
国際公開第２０１５／０３１３７１（WO，Ａ１）
米国特許出願公開第２０１３／０１２１１４９（ＵＳ，Ａ１）
国際公開第２０１６／０６９９２７（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ　１２／７１５
Ｇ０６Ｆ　９／５０
Ｈ０４Ｌ　１２／７０