

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号

特許第7085565号

(P7085565)

(45)発行日 令和4年6月16日(2022.6.16)

(24)登録日 令和4年6月8日(2022.6.8)

(51)国際特許分類

F I

G 0 6 F 9/50 (2006.01)

G 0 6 F 9/50 1 5 0 Z

G 0 6 F 9/54 (2006.01)

G 0 6 F 9/54 D

H 0 4 L 45/50 (2022.01)

H 0 4 L 45/50

H 0 4 L 61/00 (2022.01)

H 0 4 L 61/00

請求項の数 15 (全23頁)

(21)出願番号 特願2019-554611(P2019-554611)

(86)(22)出願日 平成30年4月3日(2018.4.3)

(65)公表番号 特表2020-515987(P2020-515987
A)

(43)公表日 令和2年5月28日(2020.5.28)

(86)国際出願番号 PCT/US2018/025951

(87)国際公開番号 WO2018/187371

(87)国際公開日 平成30年10月11日(2018.10.11)

審査請求日 令和3年3月8日(2021.3.8)

(31)優先権主張番号 62/481,421

(32)優先日 平成29年4月4日(2017.4.4)

(33)優先権主張国・地域又は機関
米国(US)

(31)優先権主張番号 15/611,437

(32)優先日 平成29年6月1日(2017.6.1)

最終頁に続く

(73)特許権者 303039534

ネットアップ、インコーポレイテッド
アメリカ合衆国、カリフォルニア州 9
4 0 8 9, サニーベール, クロスマン・
アベニュー 1 3 9 5

(74)代理人 100107766

弁理士 伊東 忠重

(74)代理人 100070150

弁理士 伊東 忠彦

(74)代理人 100087642

弁理士 古谷 聡

(74)代理人 100082946

弁理士 大西 昭広

(74)代理人 100195693

弁理士 細井 玲

最終頁に続く

(54)【発明の名称】 分離されたネットワークスタックにわたるインテリジェントなスレッド管理

(57)【特許請求の範囲】

【請求項1】

複数のソケットのうちの第1のソケットにおいてネットワーク処理のためにデータを検出し、

前記第1のソケットに対応する複数の分離されたネットワークスタックインスタンスのうちの第1の分離されたネットワークスタックインスタンスを識別し、前記複数のネットワークスタックインスタンスが、アプリケーションインスタンスによって作成されたものであり、

複数のスレッドのうちの1つが、前記第1の分離されたネットワークスタックインスタンスに既に割り当てられているか否かを判定し、前記複数のスレッドが、前記アプリケーションインスタンスに割り当てられており、

前記複数のスレッドのうちの第1のスレッドが、前記第1の分離されたネットワークスタックインスタンスに割り当てられているという判定に基づいて、前記第1の分離されたネットワークスタックインスタンスにしたがってデータを処理する仕事を前記第1のスレッドに割り当て、

前記第1の分離されたネットワークスタックインスタンスにスレッドが割り当てられていないという判定に基づいて、システムコールを実施することにより、前記複数のスレッドのうちの1つを前記第1の分離されたネットワークスタックインスタンスに割り当てること、

を含み、

前記第1のソケットにおいてネットワーク処理のためにデータを検出することは、オペレーティングシステムの監視サービスから、前記第1のソケットにデータが書き込まれたという通知を受信すること、を含む、

方法。

【請求項2】

複数のソケットのうちの第1のソケットにおいてネットワーク処理のためにデータを検出し、

前記第1のソケットに対応する複数の分離されたネットワークスタックインスタンスのうちの第1の分離されたネットワークスタックインスタンスを識別し、前記複数のネットワークスタックインスタンスが、アプリケーションインスタンスによって作成されたものであり、

10

複数のスレッドのうちの1つが、前記第1の分離されたネットワークスタックインスタンスに既に割り当てられているか否かを判定し、前記複数のスレッドが、前記アプリケーションインスタンスに割り当てられており、

前記複数のスレッドのうちの第1のスレッドが、前記第1の分離されたネットワークスタックインスタンスに割り当てられているという判定に基づいて、前記第1の分離されたネットワークスタックインスタンスにしたがってデータを処理する仕事を前記第1のスレッドに割り当て、

前記第1の分離されたネットワークスタックインスタンスにスレッドが割り当てられていないという判定に基づいて、システムコールを実施することにより、前記複数のスレッドのうちの1つを前記第1の分離されたネットワークスタックインスタンスに割り当てること、

20

を含み、

前記第1のソケットにおいてネットワーク処理のためにデータを検出することは、前記複数のソケットをオペレーティングシステムの監視サービスに登録させること、を含む、方法。

【請求項3】

前記第1の分離されたネットワークスタックインスタンスを識別することは、前記第1のソケットのソケット構造にアクセスすることにより、前記第1の分離されたネットワークスタックインスタンスを識別する変数を読み出すことを含む、請求項1または2に記載の方法。

30

【請求項4】

前記変数は、前記第1の分離されたネットワークインスタンスを前記複数の分離されたネットワークインスタンスのうちの他のものから分離する論理カプセル化の識別子を用いて、前記第1の分離されたネットワークインスタンスを識別する、請求項3に記載の方法。

【請求項5】

前記複数のソケットを前記監視サービスに登録することをさらに含む、請求項1に記載の方法。

【請求項6】

前記複数のスレッドのうちの1つが、前記第1の分離されたネットワークスタックインスタンスに既に割り当てられているか否かを判定することは、前記複数のスレッドのスレッドローカルストレージにアクセスすることにより、前記スレッドローカルストレージ内の変数が前記第1の分離されたネットワークスタックインスタンスを識別するか否かを判定することを含む、請求項1または2に記載の方法。

40

【請求項7】

前記第1の分離されたネットワークスタックインスタンスを識別することは、前記第1のソケットに関連する論理ネットワーク識別子を判定することを含み、
前記第1の分離されたネットワークスタックインスタンスを識別することは、前記第1の分離されたネットワークスタックインスタンスが、前記論理ネットワーク識別子にも関連していることを判定することを含む、請求項1または2に記載の方法。

50

【請求項 8】

共有ノード上でデータトラフィックの分離を維持するためのプログラムコードを含む 1 以上の非一時的マシン読み取り可能媒体であって、前記プログラムコードは、アプリケーションインスタンスのために複数のネットワークスタックコンテナを作成し、複数のソケットの各々を、前記複数のネットワークスタックコンテナの異なる 1 つに割り当て、前記複数のソケットの各々が、異なる論理ネットワーク又はサブネットワークに対応しており、

前記複数のソケットのうちのネットワーク処理のためのデータを有するソケットの識別に基づいて、

前記複数のネットワークスタックコンテナのうちのどれが前記識別されたソケットをカプセル化するかを識別し、

前記アプリケーションインスタンスに割り当てられた複数のスレッドのうちのあるスレッドが、前記識別されたネットワークスタックコンテナに割り当てられているか否かを判定し、

前記複数のスレッドのうちのあるスレッドが、前記識別されたネットワークスタックコンテナに割り当てられているという判定に基づいて、前記識別されたネットワークスタックコンテナ内のデータを処理する仕事を前記スレッドに割り当て、

前記複数のスレッドのうちのあるスレッドが、前記識別されたネットワークスタックコンテナに割り当てられていないという判定に基づいて、前記複数のスレッドのうちのあるスレッドの前記識別されたネットワークスタックコンテナへの割り当てを要求し、割り当て後に、前記識別されたネットワークスタックコンテナ内のデータを処理する仕事を前記スレッドに割り当て、さらに、

前記複数のソケットを、アプリケーションインスタンスをサポートするオペレーティングシステムの監視サービスに登録する、

ものである、1 以上の非一時的マシン読み取り可能媒体。

【請求項 9】

前記複数のスレッドのうちのあるスレッドの前記識別されたネットワークスタックコンテナへの割り当てを要求するための前記プログラムコードは、前記ネットワークスタックコンテナを識別する引数を用いてシステムコールを実施するプログラムコードを含む、請求項 8 に記載の 1 以上の非一時的マシン読み取り可能媒体。

【請求項 10】

前記監視サービスは、アプリケーションインスタンスへのソケットを、前記ソケットについてデータが検出されたときに識別する、請求項 8 に記載の 1 以上の非一時的マシン読み取り可能媒体。

【請求項 11】

前記複数のネットワークスタックコンテナのうちのどれが前記識別されたソケットをカプセル化するかを識別するための前記プログラムコードは、前記複数のソケットのソケット構造にアクセスすることにより、対応するネットワークスタックコンテナの識別子を読み出すためのプログラムコードを含む、請求項 8 に記載の 1 以上の非一時的マシン読み取り可能媒体。

【請求項 12】

前記アプリケーションインスタンスに割り当てられた複数のスレッドのうちのあるスレッドが、前記識別されたネットワークスタックコンテナに割り当てられているか否かを判定するための前記プログラムコードは、前記複数のスレッドのうちのスレッドローカルストレージにアクセスすることにより、前記ネットワークスタックコンテナの識別子を読み出すためのプログラムコードを含む、請求項 8 に記載の 1 以上の非一時的マシン読み取り可能媒体。

【請求項 13】

プロセッサと、請求項 1 乃至 7 の何れか一項に記載の方法を装置に実施させるために、前記プロセッサによって実行可能なプログラム命令を含むマシン読み取り可能媒体とを含む

10

20

30

40

50

、装置。

【請求項 1 4】

前記マシン読み取り可能媒体は、前記アプリケーションインスタンスのためのアプリケーションプログラム命令をさらに含み、前記アプリケーションインスタンスのための前記アプリケーションプログラム命令は、前記プログラム命令を含む、請求項 1 3 に記載の装置。

【請求項 1 5】

前記マシン読み取り可能媒体は、前記分離されたネットワークスタックインスタンスを作成するために前記プロセッサによって実行可能なプログラム命令をさらに含む、請求項 1 3 に記載の装置。

【発明の詳細な説明】

【背景技術】

【0001】

L i n u x（登録商標）オペレーティングシステム（OS）のようなオペレーティングシステムは、リソースを分離するためのメカニズムを提供する。こうしたメカニズムの例としては、j a i l、ゾーン、及びコンテナが挙げられる。これらの分離メカニズムの各々の構成要素は、名前空間である。L i n u x（登録商標）OSは、マウント、プロセス識別子、ネットワークスタック、ユーザーのような名前空間を提供する。各名前空間は、異なるブランチのノードが互いに分離された階層と見なされることがある。その結果、種々の名前空間にわたる可視性が妨げられる。分離のもう一つの構成要素は、コントロールグループである。コントロールグループによれば、階層構造を用いたリソースの計測や制限が可能となる。コントロールグループの例としては、プロセッサコントロールグループ、メモリーコントロールグループ、及びブロック入出力コントロールグループが挙げられる。名前空間とコントロールグループとを合わせたものが、コンテナの構成要素である。

【0002】

仮想ローカルエリアネットワーク（V L A N）によれば、種々のリモートコンピューティングシステムは、共通のローカルエリアネットワーク上にあるかのように通信することができる。したがって、セキュアでないネットワークを介して通信が行われる場合であっても、ネットワークセキュリティ対策により、種々のリモートコンピューティングシステム間でセキュアな通信が行われることが保証される。インターネットスモールコンピュータシステムインターフェース（i S C S I）をV L A Nと組み合わせて使用すれば、ストレージシステム上の種々のボリュームにセキュアな形でアクセスすることができる。i S C S Iによれば、データのボリュームがストレージシステムのどこに記憶されているかを抽出することができる。接続を行うために、デバイスは、特定のボリュームの探索要求をi S C S Iターゲットに対して発行する。この探索要求に応答し、i S C S Iターゲットは、そのボリュームの場所を判定し、そのボリュームの場所のインターネットプロトコル（I P）アドレスを提供する。その結果、エンドユーザーは、データにアクセスする前に、そのデータが実際に配置されている場所を知る必要なく、データのボリュームにアクセスすることができる。このプロセスは、i S C S Iリダイレクトと呼ばれる。

【図面の簡単な説明】

【0003】

本開示の態様は、添付の図面を参照することにより、よりよく理解され得る。

【図 1】 i S C S Iリダイレクトをサポートするストレージシステムを示す図である。

【図 2】複数のV L A Nを用いてi S C S Iリダイレクションをサポートするストレージシステムを示す図である。

【図 3】複数のV L A Nを用いてi S C S Iリダイレクションをサポートする別のストレージシステムを示す図である。

【図 4】複数のV L A Nを用いてセキュアな形でi S C S Iリダイレクションをサポートする方法を示すフロー図である。

【図 5】ネットワーク名前空間内の複数のネットワークスタックインスタンスにわたってスレッドを管理する例示的アプリケーションインスタンスを示す概念図である。

10

20

30

40

50

【図6】分離されたネットワークスタックインスタンスの受信データを処理する仕事をスレッドにインテリジェントに割り当てるための例示的動作を示すフロー図である。

【図7】分離されたネットワークスタックインスタンスからの送信用データを処理する仕事をスレッドにインテリジェントに割り当てるための例示的動作を示すフロー図である。

【図8】トラフィック分離を維持するためのスレッドマネージャーを有する例示的コンピュータシステムを示す図である。

【発明を実施するための形態】

【0004】

説明

以下の説明には、本開示の種々の態様を具現化するシステム、方法、技術、及びプログラムフローの例が含まれる。しかしながら、本開示は、これらの具体的詳細なしに実施されてもよいものと理解される。例えば、本開示は、トラフィック分離のための仮想ローカルエリアネットワーク（VLAN）技術と併せて、そのトラフィック分離を継続するために、共有ノードの分離されたネットワークスタックインスタンスにわたるスレッドのインテリジェントな管理にも言及する。本開示の種々の態様によれば、他の技術を使用して、種々のテナント／顧客間におけるトラフィックを分離し、その分離を共有ノード上で維持することができる。他の技術の例としては、仮想プライベートネットワーク（VPN）やトンネリング技術が挙げられる。他の例では、説明を分かりにくくしないために、周知の命令インスタンス、プロトコル、構造、及び技術については、詳細に示されない。

【0005】

概要

異なる顧客又はテナントのデータトラフィックを、互いに分離された状態で、共有ノードにおいて効率的に処理することができる。アプリケーションの複数のインスタンスのためのコンテナを作成したり、複数のネットワークスタックの各々についてスレッドを生成したりする代わりに、アプリケーションインスタンスは、互いに分離された複数のネットワークスタックインスタンスを作成し、分離されたネットワークスタックインスタンスにわたってスレッドをインテリジェントに管理することができる。分離されたネットワークスタックインスタンスにわたってスレッドをインテリジェントに管理するために、各スレッドは、そのスレッドが現在割り当てられているネットワーク名前空間を識別するデータを保持する。この情報を用いることで、アプリケーションは、データトラフィックを処理するネットワークスタックインスタンスのネットワーク名前空間に既に存在しているスレッドをインテリジェントに使用し、スレッドを名前空間に割り当てるシステムコールのパフォーマンスへの影響を回避することができる。

【0006】

VLANストレージシステム

図1は、iSCSIエリダイレクトをサポートするストレージシステム104を示している。ストレージシステム104は、ノード108、110、及び112を含む。各ノードは、1以上のデータボリュームに関する情報を含む場合がある。例えば、ノード3 112は、ボリューム3に関連するデータを含む。このデータは、そのボリュームに記憶されたデータの場所に関する情報を含む場合がある。ボリュームのデータは、異なるノードにわたって記憶される場合がある。一実施形態において、ボリュームのデータは、ストレージシステム104の全ノードにわたってランダムに記憶される。複数の異なるクライアントが、ストレージシステムにアクセスすることができる。これらのクライアントは、互いに独立している場合がある。したがって、各クライアントに関連するデータは、他のクライアントからはアクセス不能にされる。クライアントデータを互いに分離したままにする1つの方法は、各クライアントについて個別の仮想インターネットプロトコル（VIP）アドレスを使用することである。この実施形態では、各VIPが、ノードのクラスターに対して使用される。種々のクライアントクラスターで使用されるノードは、重複する可能性があるが、異なるVIPを使用しているため、データは分離された常態に維持される。例えば、VIP 106を使用してクラスターにアクセスするクライアントを別のVIP（

図1には示されていない)を用いて認証することは、できないであろう。したがって、クライアントは、そのクライアントに関連するクラスター上のボリュームにのみアクセスすることができる。

【0007】

iSCSIを使用して、各ボリュームにアクセスすることができる。エンドユーザーは、コンピューティングデバイス102を使用して、エンドユーザーに関連するボリュームにアクセスすることができる。例えば、クライアント102は、ボリューム3にアクセスすることができる。これを行うためには、クライアントは、ストレージシステム104に関連するIPアドレス106を知っていなければならない。この目的のために、仮想IPアドレス(VIP)が使用される。このIPアドレスが仮想的であると見なされるのは、このVIPを宛先とするデータを受信する物理デバイスが、変わるからである。クライアント102のようなiSCSIイニシエータは、最初に、エンドポイントとしてのVIPアドレス106に接続する。iSCSI機能をサポートするために、VIPアドレス106は、複数のクライアントからのすべての初期iSCSI要求を処理する責任を有する。このアドレスの実際の物理的宛先である実際のノード又は他のコンピューティングシステムは、変更することができる。例えば、ホスティングコンピューティングデバイスを変更することにより、iSCSI機能を処理する負荷のバランスを取ることができる。重要なのは、如何なる時点においても、単一のノードのみが、VIPを有することである。VIPで受信されたデータを処理するノードは何れも、入ってくるiSCSI要求に備えて、VIPの既知のポート(例えば、3260)をリッスンする。

【0008】

種々のノードがVIPのエンドポイントとして機能できるようにすることで、現在VIPを有しているノードがクラッシュした場合、別のノードがVIPになることができる。顧客の観点からは、VIPは、常に利用可能であり、顧客は、どのノードがVIPとして機能しているかを、知っている必要はない。このように、VIPは、クライアント102がiSCSIストレージに接続するために使用するアドレスである。

【0009】

VIPの1つの機能は、クライアントを、要求されたボリュームを記憶しているノードに導くことである。これによって、現在VIPとして機能しているノードとは異なるノードに、ボリュームを置くことが可能となる。例えば、図1は、ボリューム3へのアクセスを要求するクライアント102を示している。最初に、クライアント102は、VIPに向けて要求を送信する(150)。図1では、ノード108がVIPとして機能しているため、この要求は、ノード108によって処理される。ノード1は、どのノードが、ボリューム3に対するI/O要求を処理するかを判定する。例えば、データベースは、ボリューム名からノード名又はIPアドレスへのマッピング(対応付け)を記憶している場合がある。この例では、ノード112が、ボリューム3に対するI/O要求を処理する。したがって、ノード108は、ノード112のIPアドレス(例えば192.168.133.93)及びボリューム3に対するiSCSIコマンドを受け入れるポートを含むリダイレクト応答を、クライアント102に送信する(152)。これを受信すると、クライアント102は、ノード112に対して新しいログインを直接実施する(154)。

【0010】

このリダイレクトには、2つの異なるタイプのプロセスが含まれる。第1のタイプのプロセスは、VIPプロセスである。第2のタイプのプロセスは、特定のネットワークで発生するiSCSIコマンドをリッスンするプロセスである。図1では、各ノードが、iSCSIリスニングプロセスとして機能する1つのプロセスを有している。各プロセスは、そのノードが有するボリュームにアクセスするためのiSCSIコマンドをリッスンする。特定のボリュームが現在のノード上に保有されていない場合、現在のノードは、iSCSIイニシエータを正しいノードにリダイレクトすることができる。これは、iSCSIイニシエータを正しいノードにリダイレクトするVIPプロセスとは異なる。むしろ、各iSCSIリスニングプロセスは、あるノードから別のノードへ移動するボリュームを

10

20

30

40

50

考慮して、iSCSIイニシエーターをリダイレクトすることができる。このように、2つのタイプのプロセスの1つの主な違いは、各iSCSIリスニングプロセスは、すべてのクライアントが最初に通信するリダイレクタプロセスであることを意図していない点である。VIPプロセスは、すべてのクライアントが、特定のボリュームにアクセスしようとするときに最初に接続するプロセスである。

【0011】

iSCSIRダイレクトは、VLANと組み合わせて使用することができる。図2は、複数のVLANを用いてiSCSIRダイレクトをサポートするストレージシステム104を示している。具体的には、ストレージシステム104は、3つの異なるiSCSIエンドポイント、すなわち、VLAN1、クラスター、及びVLAN2を含む。クライアント102は、VIP106を使用して、そのクライアントのクラスター上の種々のボリュームにアクセスすることができる。これは、図1で説明したようにして達成される。図1とは対照的に、図2は、2つのVLANを含む。各VLANは、そのVLANに固有の専用VLAN IPアドレスを有する専用VLANネットワークインターフェースが、すべてのノード上に構成されていることを必要とする。各VLANについて異なるネットワークインターフェースを有することにより、異なるネットワークからのパケットを、互いに分離することが可能となる。あるVLANについて入ってくるトラフィックや出て行くトラフィックはすべて、そのVLANに関連する専用インターフェース及びIPアドレスを介して送受信されなければならない。また、VLANトラフィックは、非VLANトラフィックや、別のVLAN上のトラフィックを見ることができない。このVLANデータの分離を確保するために、図2では、2つのVIPが追加されている。1つは、VLAN1 206用であり、もう1つは、VLAN2 208用である。したがって、VLAN1クライアント202は、VIP206を使用して、自分のクラスターにアクセスすることができる。同様に、VLAN2クライアント204は、VIP 208を使用して、自分のクラスターにアクセスすることができる。

【0012】

VIP206及び208の追加の他に、各ノードは、各VLAN用のiSCSIRリスニングプロセスをさらに含む。VIPプロセスは、各VLAN用にも使用される場合がある。図2において、プロセスP1、P3、及びP4は、ノードのクラスター用のiSCSIRリスニングプロセスである。プロセスP5、P8、及びP11は、VLAN1用のiSCSIRリスニングプロセスであり、P7、P10、及びP12は、VLAN2用のiSCSIRリスニングプロセスである。ノード210上のP2、ノード210上のP6、及びノード212上のP9は、それぞれ、クラスター、VLAN1、及びVLAN2用のVIPプロセスである。

【0013】

図2に示したアプローチでは、異なるVLANからのトラフィックを、異なるプロセスを使用して分離している。そのため、新たなVLANを追加すると、各ノード上で動作するプロセスの数は、増加することになる。VLANの数が少ないときは、これによって問題が発生することはない。しかしながら、サポートされるVLANが多数（例えば、100又は1000）になると、システムリソースに、大きな負担がかかり始まる。多数のプロセスは、競合の問題、並びに膨大なメモリアーオーバーヘッドを引き起こすことがある。また、各プロセスが、余分なスレッド及びソケットを必要とする。さらに、VLANの追加や削除も、問題となる。多数のノードを有するストレージシステムでは、各ノードにプロセスを追加する必要がある。そのため、VLANの動的追加を実施することはできない。例えば、各ノード上にプロセスをアトミック（微小サイズ）に作成しようとする、競合状態がよく発生する。さらに、IPアドレスを割り当てる方法も、問題になる。

【0014】

図3は、一実施形態による、複数のVLANを用いてiSCSIRダイレクトをサポートするストレージシステムを示している。図3では、各ノード上に、単一のワイルドカードプロセスが存在する。このプロセスは、何れかのVLAN又はクラスターに対してVIP

10

20

30

40

50

プロセスとして動作し、かつ、すべてのクラスター及びVLANに対してiSCSIリスニングプロセスとして動作する。これを実現するために、マシンのすべてのインターフェースにバインドされたIPアドレスを使用する。例えば、IPADDR_ANY IPアドレス（例えば、0.0.0.0又は::）を使用することができる。このように、1つのプロセスは、マシンのすべてのネットワークインターフェースにわたって、特定のポート（例えば、3260）をリッスンする。種々の実施形態において、iSCSIトラフィックは、異なるIPアドレスを使用してVLANを区別するが、同じポートを使用する。異なるポートを使用する場合、すべてのポートの各々について1つのプロセスが必要である。IPADDR_ANY IPアドレスは、プロセスがリッスンすることができるワイルドカードアドレスとして機能し、これによってプロセスは、マシンの特定ポート上の何れかのインターフェースを宛先とするすべてのパケットを受信することになる。ただし、物理インターフェースと仮想インターフェースは、依然として分離されている。あるノードに入ってきたトラフィックは、依然として、それ自体のインターフェース上に留まっている。カーネル内の最終層でのみ、すべての入ってきたトラフィックが、IPADDR_ANYにバインドされた前記1つのポートをリッスンしている1つのプロセスに集約される。したがって、異なるVLAN間でデータを分離するというVLAN要件が達成される。

【0015】

このように、IPADDR_ANYアドレスを使用すると、ノードごとに、単一のプロセスを実行することが可能となる。この1つのプロセスが、すべてのクラスター及びVLANトラフィックを処理する。また、iSCSIリスニングプロセスをVIPプロセスと結合することができる。そのため、ストレージシステムでサポートされるVLANの数に関係なく、各ノードは、単一のプロセスのみを有する。この1つのプロセスが、すべての非VLANパケットも処理する。なお、各VLAN及びクラスターは、固有のIPアドレスを有しており、固有のIPアドレスは、外部クライアントによるVLAN又はクラスター上の種々のボリュームへのアクセスに使用される。

【0016】

上記のように、図3に示したアプローチによれば、VLANトラフィックが、適当に分離された状態に保たれる。したがって、VLANのセキュリティが維持される。顧客のiSCSIデータが、フィルタリングされていない多目的のワイルドカードインターフェース及びポートを通過することはない。ワイルドカードインターフェースは、種々のVLANに関連するデータを受信することができるため、ワイルドカードプロセスは、iSCSI要求及びデータを適当に処理する方法を判定しなければならない。データベースを使用して、パケットを適当にルーティングするために使用されるデータを記憶してもよい。データベースは、ストレージシステムにおける各ボリューム及びノードに関するデータを含む場合がある。ワイルドカードプロセスのリダイレクタ部分は、この情報を使用して、どのノードがそのボリュームを有しているかを探索することができる。次に、そのノードのすべてのIPアドレスを判定することができる。

【0017】

図1の例を繰り返すと、クライアントは、ボリューム3にログインすることができる。VLAN1クライアントから、ボリューム3にアクセスするために、探索要求が送信される。クライアントは、パケットをVLAN1のIPアドレス10.10.5.200に送信する。ノード1がVLAN1のVIPである場合、探索要求は、ノード1で実行されている1つのワイルドカードプロセスによって処理される。ボリューム3は、ノード3にある。ただし、ノード3は、3つのIPアドレスでアドレス指定可能であるため、どのようなIPアドレスを返すべきかが、問題となる。3つのIPアドレスとは、すなわち、クラスター用の192.168.133.93、VLAN1用の10.10.5.3、及びVLAN2用の10.10.6.3である。以前は、各インターフェースについて1つのプロセスがあった。そのため、返すべきアドレスは、分かっている場合があった。なぜなら、各VLANについて、例えば3つのIPアドレスの各々について、1つのプロセスしかなかったからである。今度は、単一のプロセスが実行されているため、返すべき正しいIP

10

20

30

40

50

アドレスを判定しなければならない。

【0018】

返すべき正しいIPアドレスを判定するために、パケットが到着したローカルエンドポイントを判定することができる。例えば、`getsockname()` メソッドの呼び出しを行うことができる。上記の例では、トラフィックがVLAN1のVIPで受信されたため、10.10.5.200が返される場合がある。この情報を使用して、VLANの名前を、データベースから判定することができる。さらに、ボリューム3がノード3にあることも、判定することができる。次に、このVLANの名前を使用して、VLAN1に関連するノード3上のIPアドレス、すなわち、10.10.5.3を判定することができる。これが、クライアントに返されるアドレスである。クライアントは、その後、10.10.5.3に直接接続することにより、ボリューム3にアクセスすることができる。

10

【0019】

クライアントがボリューム3のデータにアクセスするときに、ワイルドカードプロセスは、I/O要求を処理する。これらの要求はiSCSI探索要求ではないため、ワイルドカードプロセスに対応するiSCSILisenerが、そのI/O要求を処理する。ワイルドカードプロセスのこの部分は、クライアントがノード3への接続に使用したIPアドレスを判定する。この情報を使用して、ワイルドカードプロセスは、クライアントがVLANに接続できることを、確認することができる。

【0020】

種々の実施形態によれば、VLANをサポートするために必要なプロセスの削減に加えて、VLANをアトミック（微小サイズ）に構成することが可能となる。VLANをクラスターに追加するために、IPアドレスの1以上のブロックを、クライアントデバイスから受信する。例えば、VLANセットアッププロセスは、IPアドレスのブロックを受信することができる。さらに、新しいVLANの名前、及びそのVLANに必要となるVIPを、受信することができる。これらのIPアドレスは、各ノードに1つのIPアドレスを割り当てるために使用される。新しいVLANに関連する各IPアドレスは、現在のところ、クラスターにおいて使用することができない。VLANを作成できるようにするために、IPアドレスのブロックにおいて現在使用されているIPアドレスは、すべてフィルタリングされ、若しくは使用中であるものとしてマーキングされる。その後、未使用のIPアドレスの数を、判定することができる。未使用のIPアドレスの数がクラスター内のノードの数よりも少ない場合、VLANを作成することはできない。この事例では、VLANを作成するためにIPアドレスの別のブロックが必要であることを示すメッセージを、クライアントデバイスに返すことができる。IPアドレスの数がクラスター内のノードの数以上である場合、VLANの作成を続けることができる。

20

30

【0021】

アトミック機能をサポートするデータベースは、VLANをアトミック（微小サイズ）に作成するために使用される。各ノードに割り当てられたIPアドレス、及びノードの識別子は、データベースに記憶される。これにより、クラスター内の各ノードについて、このVLAN用のIPアドレスを判定することができる。このアトミック機能によれば、VLANの追加と同時にノードがクラスターに追加された場合でも、そのVLANを、その新しいノードを用いて正常に作成することができる。VLANが正常に追加される前に新しいノードが追加されたために、VLANが最初に正常にインストールされなかった場合、新しいVLANの追加を再試行してもよい。この事例では、新しいノードが追加されていないか、既存のノードが削除されているか、あるいは、1つのIPアドレスの使用が重複している場合に限り、新しいVLANの追加は成功する。VLANがデータベースに追加された後、各ノード用にネットワークインターフェースを作成し、適当なポートにバインドすることができる。さらに、VLANのVIPは、初期ノードにバインドされ、新しいVLAN上でiSCSI探索要求をリッスンする。

40

【0022】

この構成のもう1つの利点は、多数の異なるクライアントが、ストレージシステムを使用

50

することができる点である。単一のクライアントは、それ自体が複数の顧客を有することがある。ただし、クライアントは、各顧客のデータが他の顧客から分離され、セキュアであることを確保する必要がある場合がある。これは、各顧客に独自のVLANを提供することにより、達成することができる。上記のように、あるVLAN内のデータを、すべての他のVLANのデータから区別することができる。

【0023】

セキュアモードVLAN

例えば図3を参照して実施され、上で説明されたVLANシステムでは、各ノード又はボリュームにおいてワイルドカードアドレスが使用されるため、リスニングノードは、VLAN1クライアント202、クライアント102、及び／又はVLAN2クライアント204から、データ（又はボリューム）の探索要求を受信する可能性がある。上で説明したように、探索要求を受信したノードはさらに、要求の結果を返すために、getsockname（）メソッドの呼び出しを使用して正しいIPアドレスを判定する。その後、クライアントから要求されたデータ／ボリュームのアドレスをクライアントに返すことにより、クライアントを適切なノードにリダイレクトし、そのデータにアクセスするためにiSCSI要求を開始する。iSCSIプロシージャは、クライアントがアクセスできないデータ／ボリュームにクライアントがアクセスすることを、防止することができる。ただし、ノードは、複数のクライアントから探索要求を受信し、返すことができるため、探索要求の応答は、特定のクライアントによるアクセスが許可されていないデータ／ボリュームを参照する可能性がある。つまり、図3のノードは、種々のクライアントからの種々の要求をリッスンする単一のプロセスを使用して動作しているため、これらのノードが、単一クライアントについての探索要求に応答してデータ／ボリュームの場所を判定すること

10

20

【0024】

図4は、例示的实施形態による、複数のVLANを用いてセキュアな形でiSCSIリダイレクトをサポートする方法400のフロー図を示している。代替実施形態では、もっと少ない、追加の、及び／又は異なるステップが実施されてもよい。また、フロー図の使用は、実施されるステップの順序に関して制限的であることを意図していない。

30

【0025】

動作402では、データ／ボリュームの探索要求が、クライアントから発信される。一例として、探索要求は、図3に示され、図3を参照して説明されたように、VLAN1クライアント202から発信される場合がある。図1、図2、及び図3に示され、それらを参照して上で説明された手順と同様に、探索要求は、例えば図3のノード314のような第3のノードにあるデータ／ボリュームに対するものである場合がある。ただし、例えば図3のノード310のような別のノードが、VLAN1クライアント202のVIPであってもよい。したがって、VLAN1クライアント202からの探索要求は、動作404で、ノード310上のワイルドカードプロセスによって処理されることになる。

40

【0026】

動作404では、図3を参照して上で説明したように、VIPは、要求の結果を返すために、正しいIPアドレスを判定することができる。この例では、ノード310は、探索要求がVLAN1クライアント202から発信されたことを判定し、要求の対象であるボリューム／データが記憶されている場所を判定した後、ノード314上の要求されたデータ／ボリュームの場所を、VLAN1クライアント202に送り返す。この判定は、動作406で行われる。

【0027】

ただし、要求されたデータ／ボリュームの場所をクライアントに返す前に（及び、その後

50

クライアントを第3のノードにリダイレクトし、iSCSIプロシージャを開始することにより、第3のノードからデータ／ボリュームを読み出す前に)、動作408において、第1のノード310は、その探索要求をクライアント固有のボリュームリスト410と照合する。クライアント固有のボリュームリスト410は、VLAN1クライアント202がアクセスできるすべてのボリュームのインデックス又はデータベースである。一部のボリュームは、VLAN1クライアント202に専用であってもよく、他のボリュームは、VLAN1クライアント202及び他のクライアントからアクセス可能であってもよい。さらに他のボリュームは、他のクライアントからはアクセス可能であるが、VLAN1クライアント202からはアクセスできない場合がある。この事例では、そのようなボリュームは、VLAN1クライアント202についてのクライアント固有のボリュームリスト410には現れない場合がある。代替実施形態では、クライアント固有のボリュームリストは、探索要求が行われたときに照合される複数のクライアント固有のボリュームリストを含む場合がある。例えば、あるクライアントが異なるデータ／ボリュームへのアクセスを許可する異なるグループ／セキュリティドメインの一部である場合、そのクライアントは、実質的に、複数の固有のボリュームリストを有する場合がある。実際には、探索要求を許可する際に、システムは、それらのボリュームリストを結合して一回だけ照合を実行し、あるいは、各ボリュームリストを同様に順番に調べることにより、特定のデータ／ボリュームの位置情報(ボリュームID)を要求することをクライアントに許可すべきか否かを判定することができる。実施形態によっては、システムは、クライアント固有のボリュームリスト(単数又は複数)全体を調べなくてもよい。例えば、システムがクライアント固有のボリュームリスト(複数可)上で一致するものを見つけた場合、そのボリュームリスト(複数可)の残りを調べる必要はない。このような方法によれば、システムのリソースを温存することができる。

【0028】

図4は、ウェブボリュームリスト412も示している。クライアント固有のボリュームリスト410は、VLAN1クライアント202に割り当てられ、すなわち、VLAN1クライアント202からアクセス可能なボリュームIDを示しているのに対し、ウェブボリュームリストは、ストレージシステム(例えば、図1～図3のストレージシステム104)に存在するすべてのボリュームIDを示している。したがって、ストレージシステムに何かを記憶しているすべてのボリュームIDが、ボリュームがどのノードに記憶されているか、及びボリュームに何個のノードがあるかとは関係なく、Webボリュームリストに示されている。ボリュームIDは、特定の物理的記憶場所を指し示している。したがって、何が記憶されていたとしても、特定の場所をボリュームIDで示すことができる。クライアント固有のボリュームリスト410上にあるボリュームIDはすべて、Webボリュームリスト412(例えば、1、2、3、4、11)上に現れる。ウェブボリュームリスト412は、VLAN1クライアント202には割り当てられておらず、VLAN1クライアント202からはアクセスすることができない、クライアント固有のボリュームリスト410上にはないボリュームID(例えば、6、7、8、9、10)を、さらに含む。ただし、これらのボリュームIDは、Webボリュームリスト412にあるため、少なくとも1つの他のクライアント及び／又はVLANからはアクセス可能である。同様に、ウェブボリュームリスト412とクライアント固有のボリュームリスト410との両方に現れるボリュームIDは、他のクライアント／VLANボリュームリスト上にある場合があり、したがって、それらの他のクライアント／VLANからアクセス可能である場合がある。このような状況は、複数のクライアントが同じ(等しい)ボリュームを記憶している場合に発生することがある。したがって、ストレージシステムにとっては、データの1つのインスタンスのみを記憶の方が効率的である。ストレージシステムにおいてデータ／ボリュームを記憶することができる他のボリュームIDは、空である場合があり、現在のところ、データ(例えば、5)を含まない場合がある。そのようなボリュームIDは、以前はデータを記憶していた可能性があるが、特定のボリュームIDにあるデータにリンクされたクライアントがないと判断された後、ガベージコレクタープロセスによって削除さ

10

20

30

40

50

れた可能性がある。例えば、探索要求がボリュームID 5にあるデータの要求を含む場合、システムは、エラーメッセージを返してもよいし、そのボリュームIDに新しいデータを動的に記憶／作成してもよいし、あるいは、要求されたデータが実際にどこか他の場所に記憶されている（別のボリュームIDにある）か否かを判定するプロセスを実行してもよい。

【0029】

上述のように、動作406でクライアントが判定された場合、動作408で、ノードは、探索要求において要求されたデータ／ボリュームが、クライアント固有のボリュームリスト410上にあるか否かを判定する。データ／ボリュームがクライアント固有のボリュームリスト410上にあるか否かの判定は、動作414でクライアントがデータ／ボリュームの実際の場所にリダイレクトされる前に行われる。動作408を動作414よりも前に行うことで、クライアントが許可されていないボリュームに関する情報や、クライアントが許可されていないボリューム位置へのリダイレクトに関する情報が、そのクライアントに送信されないことを保証する。

【0030】

さらに、動作414で、クライアントは、本明細書に開示されたようなノードからの情報によってリダイレクトされ、探索要求において要求されたデータ／ボリュームを読み出し、及び／又はアクセスすることができる。例えば、動作414は、第3のノード314からデータをアクセスし、及び読み出すためのiSCSIプロセスを開始することができる。このようなプロセスは、図1に示したプロセスと同様であってもよく、上述のように、ノード310（動作408における検証後）は、アクセスが要求されたボリュームに対するiSCSIコマンドを受け入れるノード314のポートのIPアドレスを含むリダイレクト応答を、VLAN1クライアント202に送信する。このリダイレクト応答は、例えば、図1に152で示されている。これを受け取ると、VLAN1クライアント202は、ノード314への新しいログインを直接実施することができる。この直接ログインは、例えば、図1に154で示されている。

【0031】

本明細書に開示されたシステム及び方法は、ストレージシステムに柔軟性とセキュリティを提供する。例えば、複数のクライアントについてのVLANを一つにグループ化することで、ボリュームを、異なるVLANから見えるようにすることができる。例えば、単一エンティティの一部であるクライアントのグループ／セキュリティドメインは、同じボリュームのすべてにアクセスすることができる。別の例では、クライアントのグループ／セキュリティドメインのサブセットが、同じボリュームのすべてにアクセスすることができる。その結果、複数のグループ化されたクライアントが、本明細書に開示されるようなストレージシステムにアクセスするときに、共通のVLANを使用することができる。これにより、各クライアントデバイスについて個別の多数のVLANを必要とすることなく、クライアントのグループが、ストレージシステムをセキュアに使用することが可能となる。

【0032】

さらに、クライアントのグループ化を利用することにより、より少ない数のクライアント固有のボリュームリストを維持し、及び使用することができる。これは、共通のVLANを使用するようにクライアントをグループ化するか否かに関係なく行うことができる。この例では、複数のグループ化されたクライアントは、それらのクライアントから発信された探索要求を、同じクライアント固有のボリュームリストと照合する場合がある。そのため、このリストは、より正確には、クライアントグループ固有のボリュームリスト若しくはセキュリティドメイン固有のボリュームリストとして特徴付けられる場合がある。複数のクライアントについて1つのボリュームリストを使用することにより、ストレージシステムは、所与の数のクライアントに対してより少ない数のボリュームリスト（各クライアントについて1つのボリュームリストを有する場合に比べて少ない数）を記憶し、及び維持することができる。これらのグループ化実施形態では、探索要求が行われる前に、クライアントをグループ化しなければならない（共通のVLANにより、及び／又は、複数の

10

20

30

40

50

VLANを共通のボリュームリストに関連付けることにより)。これによって、探索要求が行われた後、システムが探索リストを、要求されたデータ／ボリューム及びそれらの各々の場所に関する情報をクライアントに送信すべきか否かを示す適当なボリュームリストと照合することが、保証される。

【0033】

本明細書に記載されたシステム及び方法によれば、データが実際にアクセスされる iSCSI 層ではなく、システムのネットワーク層のためのセキュリティ／認証も定義される。これによって、ストレージシステムにデータを記憶するクライアントのためのセキュリティ及び保護に関するさらに別の層が追加される。

【0034】

ノードは、クライアントがアクセスを許可されていないデータ／ボリュームに関する情報をクライアントに提供するような形でノードがクライアントからの要求に応えることを、防止することができる。この手続は、異なるクライアントが一部の共通ボリュームにアクセスできる場合であっても、使用することができる。共通ボリュームは、一部の共通データを共有するクライアントのグループ又はセキュリティドメインにとって、特に有用である。

【0035】

分離されたネットワークスタックにわたる限られたスレッドのインテリジェントな管理
前述のように、アプリケーションを、異なるネットワーク名前空間にわたってスレッドを管理するようにプログラムすることで、共有ノードによって処理される異なる VLAN のトラフィックの分離を維持することができる。スレッドを管理するために、アプリケーションインスタンスは、VLAN とネットワーク名前空間との間の関連付けを維持し、スレッドによってスレッドレベルのストレージの形で（例えば、スレッドのスタック／メモリ空間内に変数の形で）維持されたコンテキスト情報を読み出す。本明細書の説明では、ネットワーク名前空間を作成し、ネットワーク名前空間にわたってスレッドを管理するアプリケーションインスタンスのこの機能を、「コンテナ化動作」と呼ぶ。これは、リソースをアプリケーションの複数のインスタンスで消費する代わりに、複製、すなわちクローニングを、ネットワークスタックに制限するからである。

【0036】

「ノード」という用語は、クライアント要求を処理するプロセスの集合を指して使用され、ホストハードウェア（コンピューター、ブレード、マシン、アプライアンスなど）を含んでもよいし、含まなくてもよい。例えば、ストレージノードは、クライアントからのストレージ要求に対処するように協調及び／又は協働するプロセスの集合である。あるノードのためのプロセスの集合は、アプリケーションインスタンスの 1 以上のプロセスを含む。

【0037】

「ネットワークスタックインスタンス」という用語は、ネットワーク名前空間に割り当てられた、又はネットワーク名前空間内にある、ネットワーク関連リソースの集合を指して使用される。こうしたリソースの例としては、インターフェース、アドレス空間、及びルーティングテーブルなどが挙げられる。ネットワーク名前空間が作成されると、オペレーティングシステムは、そのネットワーク名前空間のためにデフォルトネットワークスタック（例えば、デフォルトの仮想インターフェースや IP アドレス空間など）をインスタンス化することができる。さらに別のリソースを、ネットワーク名前空間に割り当ててもよい。

【0038】

図 5 は、ネットワーク名前空間内の複数のネットワークスタックインスタンスにわたってスレッドを管理する例示的アプリケーションインスタンスの概念図である。ホスト 501 は、オペレーティングシステム 503 及びアプリケーションインスタンス 511 を有している。図 5 では、アプリケーションインスタンス 511 は、複数のネットワーク名前空間 515、517、519 を既に作成している。各ネットワーク名前空間は、ネットワークスタックのインスタンスを含む。さらに、アプリケーションインスタンス 511 は、それ

10

20

30

40

50

らのネットワーク名前空間に、ソケット及びVLANを既に割り当てている。

【0039】

ソケットを割り当てるために、アプリケーションインスタンス511は、システムコールを実施することにより、ネットワーク名前空間の各々のアドレス空間に基づくソケットアドレスを有するソケットを作成した。例えば、アプリケーションインスタンス511は、ソケット、並びに、ネットワーク名前空間515に含まれるネットワークスタックのアドレス空間内のネットワークアドレスの指示を作成するための、オペレーティングシステム503への要求を用いて、ソケット541を作成した。アプリケーションインスタンス511はさらに、そのソケットをそのネットワーク名前空間に割り当てるためのコマンドを呼び出した。

10

【0040】

ソケット作成後のある時点で、アプリケーションインスタンス511は、イベント監視サービス514を呼び出す（例えば、`epoll()` システムコマンドを呼び出す）。アプリケーションインスタンス511は、ソケット識別子のリストをイベント監視サービス514に渡すことができる。`epoll()` 呼び出しに関連して、アプリケーションインスタンス511は、1以上のコマンドを実行することにより、監視のために、ソケット541、543、545の一組のファイル記述子を登録することができる。実施形態によっては、分離されたネットワークスタックインスタンスの各々の中でソケット又はバッファを可視化する、サポートオペレーティングシステムの他のサービス又はシステムコールを使用してもよい。

20

【0041】

VLANを割り当てるために、アプリケーションインスタンスは、ネットワーク名前空間の各々の中で、各VLANをVLANタグによってインターフェースif0にリンクした。図5は、ネットワークスタックインスタンスのいくつかの例示的要素を、ネットワーク名前空間へのVLANの割り当てとともに示している。ネットワーク名前空間515は、ルーティングテーブル521、ソケット541、及び、インターフェースif0にリンクされたVLAN100の表現を含む。ネットワーク名前空間517は、ルーティングテーブル523、ソケット543、及び、インターフェースif0にリンクされたVLAN101の表現を含む。ネットワーク名前空間519は、ルーティングテーブル525、ソケット545、及び、インターフェースif0にリンクされたVLAN nの表現を含む。インターフェースif0は、ホスト501の物理インターフェースに対応する。異なるVLANをインターフェースif0にリンクすることにより、インターフェースif0は、異なるインターフェースに論理的に分割される。各VLANが、論理的に異なるネットワーク又はサブネットワークを表しているからである。したがって、図5は、オペレーティングシステム503（例えば、オペレーティングシステムのネットワーク名前空間）内に表されたインターフェースif0、及び、異なるネットワーク名前空間515、517、519におけるインターフェースif0と異なるVLANタグとの結合を示している。

30

【0042】

アプリケーションインスタンス511は、アプリケーションインスタンス511に提出されたコマンドに応答して、及び／又は設定ファイルに基づいて、ネットワーク名前空間の作成、及びネットワーク名前空間へのリソースの割り当てを実施することができる。例えば、設定ファイルは、作成するネットワーク名前空間の数、ネットワーク名前空間の名前、及びネットワーク名前空間へのリソースの割り当てを指定することができる。アプリケーションインスタンス511は、動作開始時に、この設定を読み出すことにより、数百の名前空間を作成し、設定することができる。その結果、互いに分離された数百のネットワークスタックがインスタンス化される。これにより、分離されたデータトラフィックを大規模にセキュアに処理するための十分なネットワークスタックを有するアプリケーションの効率的起動が可能となる（例えば、1秒あたり数百万の入出力動作（IOPS））。

40

【0043】

多数のネットワークスタックは、大規模なトラフィックを分離することができるが、ホス

50

ト501のリソースを使い果たすことなく、ネットワークスタックの各々にスレッドを割り当てることはできない。大規模なトラフィックを処理しながらリソース消費のバランスをとるために、オペレーティングシステム503は、スレッドプール513をアプリケーションインスタンス511に割り当てる。スレッドプール513は、トラフィックからの負荷に適応するように調整される場合がある。調整とは無関係に、スレッドプール513のスレッドの数は、ネットワーク名前空間の数よりも少なくなる場合があり、コンテキストの切り替えが行われる場合がある。数百のネットワーク名前空間のために数百のスレッドを維持することは、リソースを使い果たす可能性があるが、あらゆるネットワークランザクションについて、オペレーティングシステムへのシステムコールを行い、スレッドをネットワーク名前空間ごとに切り替えることも、大規模なトラフィックを処理する場合、コストがかかることになる。したがって、アプリケーションインスタンス511は、スレッドプール513からネットワーク名前空間515、517、519へのスレッドの切り替えをインテリジェントに管理することにより、オペレーティングシステム503へのシステムコールを回避する。アプリケーションインスタンス511によって管理される各スレッドは、スレッドローカルストレージ内に、そのスレッドが存在するネットワーク名前空間（すなわち、そのスレッドが実行されているネットワーク名前空間（アクティブ状態の場合）、又はそのスレッドが実行されていたネットワーク名前空間（待機状態の場合））を保持している。この情報から、アプリケーションインスタンス511は、スレッドをネットワーク名前空間に割り当てるためのコールを使用することなく、何時ネットワーク処理の仕事をスレッドに割り当てることができるかを、判定することができる。

【0044】

図5は、一連の文字A、B、Cを使用して、3つの一般的ステージを示している。これらのステージの各々は、特定の動作ではなく、一組の動作を表している。

【0045】

ステージAは、ホスト501のインターフェースif0でのデータトラフィック505の受信を表している。データトラフィック505は、異なるVLANからのトラフィックで構成されている。ステージBは、トラフィックユニット内のVLANタグ及びソケットアドレスにしたがって、適当なソケットに書き込まれてるトラフィックを表している。VLAN100のトラフィックの場合、インターフェースif0のデバイスドライバは、ネットワーク名前空間515内に含まれるソケット541にデータを書き込む。デバイスドライバは、VLAN101のデータをネットワーク名前空間517内に含まれるソケット543に書き込み、VLANnのデータをネットワーク名前空間519内に含まれるソケット545に書き込む。ステージCは、スレッドプール513からネットワーク名前空間515、517、519にスレッドをインテリジェントに割り当てる種々の動作を実施することにより種々のVLANのトラフィック505を処理する、アプリケーションインスタンスを表している。この例では、スレッド529が、ネットワーク名前空間519に最後に割り当てられた。ネットワーク名前空間515、517には、スレッドが割り当てられていない。ネットワーク名前空間517と519との間の省略記号は、実用性を得るために、多数のネットワーク名前空間のうちのほんのいくつかのみが、描かれていることを示している。アプリケーションインスタンス511は、ネットワーク名前空間又はヌル値を示している、各スレッドのスレッドローカルストレージ内の変数を読み出す。アプリケーションインスタンス511は、スレッド529がネットワーク名前空間519に既に割り当てられていることを判定した場合、ソケット545のデータを処理する仕事をスレッド529に割り当てる。例えば、アプリケーションインスタンス511は、ネットワーク通信プロトコル処理コードへの参照及びソケット545のソケット識別子を含む引数を用いてスレッド529を起動（ウェイクアップ）する関数を呼び出す。アプリケーションインスタンス511は、ネットワーク名前空間515にスレッドが割り当てられていないことを判定する。アプリケーションインスタンス511は、システムコールを行い、スレッドプール513からアイドルスレッド527をネットワーク名前空間515に割り当て、スレッド527を起動し、ソケット541のデータを処理する仕事をスレッド527に割り

当てる。アプリケーションインスタンス 5 1 1 は、ネットワーク名前空間 5 1 7 のソケット 5 4 3 のデータに対して同じことを行ってもよく（すなわち、スレッドプール 5 1 3 からスレッドを割り当て、スレッドに仕事を割り当てる）、あるいは、スレッドプール 5 1 3 にまだ何も割り当てられていない場合、別のネットワーク名前空間からのアイドルスレッドと入れ替えてもよい。

【0046】

図 5 の例では、ネットワークスタックインスタンスを分離するためのネットワーク名前空間に言及しているが、実施形態によっては、他の分離メカニズム（例えば、Linux コンテナ又は LXC など）を使用してもよい。実施形態に関係なく、ネットワークスタックインスタンスは、互いに分離され、アプリケーションインスタンスによって管理される。さらに、図 5 では、ホストデバイス、すなわち、OS、アプリケーションインスタンス、及びネットワークスタックインスタンスを有するホストマシンに言及している。ただし、ホストは、仮想ホストであってもよい。図 6～図 7 は、図 5 で使用された用語よりも一般的な用語を使用して、分離されたネットワークスタックインスタンスにわたってトラフィックを処理するための例示的動作を示すフロー図である。

【0047】

図 6 は、分離されたネットワークスタックインスタンスの受信データを処理する仕事をスレッドにインテリジェントに割り当てるための例示的動作を示すフロー図である。図 6 では、これらの動作を実施するアプリケーションインスタンスにも言及している。

【0048】

ブロック 6 0 1 で、アプリケーションインスタンスは、ネットワーク処理のためにソケットにおいてデータの受信を検出する。アプリケーションインスタンスは、種々のネットワークスタックインスタンスに関する大局的視点を有するバックグラウンドプロセス又はバックグラウンドサービスからの通知に基づいて、このデータの受信を検出する。アプリケーションインスタンスは、監視のために、そのソケットを他のソケットとともに、前もって登録している。監視には、例えば、複数のソケットにわたってコールバックや反復処理を使用してもよい。

【0049】

ブロック 6 0 3 で、アプリケーションインスタンスは、そのソケットに対応するネットワークスタックインスタンスを識別する。アプリケーションインスタンスが受信した通知により、ソケットは識別される。アプリケーションインスタンスは、ソケット識別子を用いてソケット構造を読み出すことにより、そのアプリケーションインスタンスが割り当てられたネットワークスタックインスタンスを判定する。このネットワークスタックインスタンスは、ソケットの作成時に配置されたものである。ソケット構造は、ネットワークスタックインスタンスを分離する論理カプセル化の識別子（例えば、名前空間識別子、コンテナ識別子など）により、ネットワークスタックインスタンスを識別することができる。

【0050】

ブロック 6 0 5 で、アプリケーションインスタンスは、識別されたネットワークスタックインスタンスに既にスレッドがあるか否かを判定する。ソケットと同様に、アプリケーションインスタンスは、アプリケーションインスタンスに割り当てられたスレッドのスレッドローカルストレージにアクセスすることができる。アプリケーションインスタンスは、そのソケット構造において識別されたネットワークスタックインスタンスとの一致が見付かるか、あるいは、スレッドネットワークスタック割り当て変数の探索が完了するまで、各スレッドのスレッドローカルストレージ内のネットワークスタック割り当て変数を読み出すことができる。別の例示的实施形態として、アプリケーションインスタンスは、自分自身に対するスレッドの割り当てを追跡することができる。アプリケーションインスタンスは、システムコールを実施することにより、スレッドを割り当てるからである。アプリケーションインスタンスは、ネットワークスタックインスタンス識別子に対するスレッド識別子の割り当てのリストを維持してもよく、そのリストを調べることにより、ソケット構造において識別されたネットワークスタックインスタンスにソケットが既に割り当てら

10

20

30

40

50

れているか否かを、判定することができる。識別されたネットワークスタックインスタンスに既にスレッドが割り当てられてはいない場合、プロセスは、ブロック 607 へと進む。そうでなければ、プロセスは、ブロック 609 へと進む。

【0051】

ブロック 607 で、アプリケーションインスタンスは、識別されたネットワークスタックインスタンスに非アクティブなスレッドを割り当てる。アプリケーションインスタンスは、システムコールを実施する（すなわち、OS の機能又は OS ツールの機能呼び出す）ことにより、非アクティブなスレッドを識別されたネットワークスタックインスタンスに割り当てる。プロセスは、ブロック 609 へと続く。

【0052】

ブロック 609 で、アプリケーションインスタンスは、スレッドに、受信データを処理する仕事を割り当てる。スレッドが汎用スレッドである場合、アプリケーションインスタンスは、識別されたソケットの入力ストリームへのポインター、及びネットワークプロトコル処理プログラムコードへの参照を渡す関数を呼び出すことができる。スレッドのコードには、ネットワークプロトコル処理コードが既に組み込まれている場合がある。この場合、アプリケーションインスタンスは、入力ストリームポインターを渡す。場合によっては、アプリケーションインスタンスは、独立したファンクションコールを実施することにより、非アクティブなスレッドを起動する場合がある。

【0053】

図 7 は、分離されたネットワークスタックインスタンスからの送信用データを処理する仕事をスレッドにインテリジェントに割り当てるための例示的動作を示すフロー図である。図 7 では、動作を実施するアプリケーションインスタンスにも言及している。

【0054】

ブロック 701 で、アプリケーションインスタンスは、送信するデータ、及びそのデータのソケットアドレスを判定する。アプリケーションインスタンスは、特定の VLAN のための特定のボリュームを指定する読み出し要求を処理している場合がある。そのボリュームに対応するストレージデバイスからデータを取得し、ネットワーク通信プロトコルに関係しない他の処理（例えば、暗号化）を実施した後、アプリケーションインスタンスは、その読み出し要求から、主張されたソケット識別子を読み出す。例えば、ソケット識別子は、ネットワークスタックインスタンスの出力から生成されたストレージコマンドとともに送られてくる場合がある。ソケット識別子は、ホストシステムのソケットを一意に識別する内部識別子であり、ソケットアドレスとは異なる。

【0055】

ブロック 703 で、アプリケーションインスタンスは、取得したデータを、そのソケット識別子に対応するソケットに書き込む。その結果、アプリケーションインスタンスは、監視プロセス／サービスから、そのソケットで行うべき作業の通知を、ソケット構造内のネットワークスタックインスタンスの識別情報とともに、受信する。

【0056】

ブロック 705 で、アプリケーションインスタンスは、識別されたネットワークスタックインスタンスに既にスレッドがあるか否かを判定する。アプリケーションインスタンスは、アプリケーションインスタンスに割り当てられたスレッドのスレッドローカルストレージにアクセスすることができる。アプリケーションインスタンスは、そのソケット構造において識別されたネットワークスタックインスタンスとの一致が見付かるか、あるいは、スレッドネットワークスタック割り当て変数の探索が完了するまで、各スレッドのスレッドローカルストレージ内のネットワークスタック割り当て変数を読み出すことができる。別の例示的实施形態として、アプリケーションインスタンスは、自分自身に対するスレッドの割り当てを追跡することができる。アプリケーションインスタンスは、システムコールを実施することにより、スレッドを割り当てるからである。アプリケーションインスタンスは、ネットワークスタックインスタンス識別子に対するスレッド識別子の割り当てのリストを維持してもよく、そのリストを調べることにより、ソケット構造において識別さ

10

20

30

40

50

れたネットワークスタックインスタンスにスレッドが既に割り当てられているか否かを、判定することができる。識別されたネットワークスタックインスタンスに既にスレッドが割り当てられてはいない場合、プロセスは、ブロック 707 へと進む。そうでなければ、プロセスは、ブロック 709 へと進む。

【0057】

ブロック 707 で、アプリケーションインスタンスは、識別されたネットワークスタックインスタンスに、非アクティブなスレッドを割り当てる。アプリケーションインスタンスは、システムコールを実施する（すなわち、OS の機能又は OS ツールの機能呼び出す）ことにより、非アクティブなスレッドを識別されたネットワークスタックインスタンスに割り当てる。プロセスは、ブロック 709 へと続く。

【0058】

ブロック 709 で、アプリケーションインスタンスは、送信用データを処理する仕事をスレッドに割り当てる。場合によっては、アプリケーションインスタンスは、独立したファンクションコールを実施することにより、非アクティブなスレッドを起動する場合がある。

【0059】

単なる一例として、コンテナ化動作は、システム 104 や古いクライアント及び新しいクライアントに大きな変更を加える必要なく、図 3 のストレージシステム 104 にさらに別のクライアントを統合することに役立つことがある。例えば、新しいクライアントは、現在ローカルに記憶されたストレージシステムを使用している場合がある。新しいクライアントは、クラウドや、インターネットベースのストレージシステムに統合されることが、望ましい場合がある。ストレージシステム 104 が既にクラウドに基づいている場合、古いローカルシステムのルーティングプロセス、IP アドレス、及び通信の一部は、そのクラウドで既に使用中の古いクライアントのものに類似する場合がある。ストレージを管理する OS やアプリケーションを複製することなく、新しいクライアントをストレージシステム 104 に統合するために、新しいクライアントを、本明細書に開示したコンテナ化動作方法を使用して、統合することができる。このようにすると、新しいクライアントと古いクライアントとの両方が、同じアドレス及びポート（例えば、3260）を使用して、ストレージシステム 104 との間で送受信を行うことができる。ただし、本明細書に開示したシステム及び方法を使用すると、異なる VLAN を使用するクライアントに関連するトラフィックを、コンテナ化動作を使用して、分離することができる。換言すれば、複数のクライアントが同じストレージシステム 104 を同じ形で使用している場合であっても、ある VLAN に関連するトラフィックを、他の VLAN に関連するクライアントから見ることにはできない。また、これによってストレージシステム 104 は、より多くのクライアントに対して、より効率的にサービスを提供することが可能となる。なぜなら、ストレージシステムは、ネットワークスタックをインスタンス化する際に、各ネットワークスタックに対して自分自身のインスタンスを書き換える必要がなくなるからである。そのような方法は、大きなメモリ占有領域を使用する場合があり、また、非常に大きな分散システムを大規模に（すなわち、多数のクライアントと共に）動作させる必要がある場合がある。クライアントのトラフィックは、異なるコンテナに（そしてその後、異なる独立したソケット又はネットワークスタックに）分離される。システムをこのように構成することで、多数の異なるクライアントが、同じストレージシステム 502 を使用することができる。ストレージシステム 502 は、共通のノード／ボリューム、オペレーティングシステム（OS）、及びアプリケーション（例えば、iSCSI 要求に応答し、iSCSI 要求を満足させるためのもの）を含むことができる。システムの共通の側面を利用することで、使用されるシステムリソース全体を、削減又は合理化することができる。ただし、ストレージシステムとクライアントとの間の通信が、他のクライアントのトラフィックから分離されるため、クライアントが、セキュリティやプライバシーを犠牲にすることはない。

【0060】

本明細書に開示されたシステム及び方法のユーザは、1 以上のクライアントに関連する場合がある。これらのクライアントは、1 以上の VLAN に関連する場合がある。VLAN

10

20

30

40

50

に基づいて動作をコンテナ化することにより、ユーザーは、自分のクライアントに対して複数のVLANを設定することにより、自分のクライアントのトラフィックを互いに分離することができる。例えば、あるユーザー組織内の異なるタイプのアカウントは、異なる情報に対して異なるレベルのアクセス権を有する場合がある。そのため、そうした個々のアカウントのクライアントトラフィックを異なるVLANを用いて分離し、コンテナ化された動作を使用して、それらのアカウントのトラフィックが互いに分離されるようにすることが望ましい場合がある。さらに、本明細書に開示されたシステム及び方法の他のユーザーは、異なる、さらに別のVLANに関連する場合がある。したがって、異なるユーザーのトラフィックも、本明細書に開示されたシステム及び方法を使用して、分離されることになる。実施形態によっては、システムは、複数のVLANを特定のスペース又はコンテナに関連付ける場合がある。このようにして、通信監視モジュール504は、ある動作が、あるコンテナに関連するVLANの何れか1つに関連している場合、その動作がそのコンテナに関連しているものと、判定することができる。

【0061】

図3は、iSCSI要求（例えば、3260）用に指定されたポートを通して3つのクライアント（例えば、202、102、204）のすべてから来るすべての要求をリスンするすべてのノード（例えば、310、312、314）を示している。この機能は、上で説明したように、各ノード上で実行されるプロセスの数を最小限に抑えることに役立つ。また、この機能によれば、図3を参照して説明したように、各ノード上のあらゆる場所について個別のリスニングプロセスを実行する必要なしに、システム全体を、より効率的に機能させることができる。したがって、図5を参照して上で説明した方法及びシステムによれば、異なるクライアントが、ストレージシステムの同じポート（例えば、3260）を用いて通信することが可能となり、そのトラフィックが、互いに可視化されることはない。換言すれば、本明細書に開示されたシステム及び方法によれば、システムは、トラフィックを、異なるVLANに関連する異なるコンテナ又は異なる名前空間に分離することが可能となる。このトラフィックは、クライアントに別のポートを指し示す必要もなければ、ストレージシステムを管理するソフトウェアアプリケーションの異なるインスタンスを作成する必要もなく、適当に分離されることができる。換言すれば、アプリケーションのネットワークコンポーネント動作のコンテナ化は、複製され、コンテナ化されることがあるが、iSCSI要求の処理などの処理を実行しているアプリケーション全体が、複製され、又はコンテナ化されることはない。このようにして、各クライアント及び／又はVLANは、同じポート（3260など）を指し示す続けることがあるが、そのポートを通過するトラフィックは依然として、各クライアントのVLANにしたがって、異なるコンテナに分離されたままとなる。上述のように、これが有用となるのは、クライアントのストレージが、クラウドに移動される場合である。もし、異なるVLAN用のストレージが、前もって物理的に分離されていた場合、トラフィックもまた、分離されている（たとえば異なるVLANを有する異なるクライアントによって使用されるポートが、3260のような類似の名前を有していたとしても）。ただし、クラウドネットワーク及び／又はクライアントが再構成されない限り、すべてのデータをクラウドに移動すると、異なるVLANを有する異なるクライアントは、要求／トラフィックを、同じ宛先ポートに送信する可能性がある。ネットワークコンポーネント（ポート）をVLANにしたがってコンテナ化することにより、ストレージシステムにおいて、クライアント側及びデータ管理側を変更する必要がなくなる。代わりに、本明細書に開示されたようなコンテナ化ネットワーク動作を実施することにより、クライアント及びストレージシステムの構成を同じに保ちながら、異なるVLANにしたがってトラフィックを分離することが可能となる。もう1つの利点は、iSCSIのような特定のプロトコルでは、使用する特定のポート（例えば、3260ポート）が指定されることである。種々のVLAN上のクライアントは、自分達のトラフィックを互いに公開することなく、特定のポートを指し示し続けることができる。これにより、プロトコル標準を引き続き使用する機会が得られ、クライアント間でトラフィックを分離するために、プロトコル標準を再定義する必要がなくなる。

10

20

30

40

50

【0062】

マシン読み取り可能媒体上に具現化されたプログラムコードは、限定はしないが、無線、有線、光ファイバケーブル、RFなど、又はこれらの適当な組み合わせを含む、任意の適当な媒体を使用して送信される場合がある。機械に対して特定の形で動作するように命令することができるプログラムコード／命令が、マシン読み取り可能媒体に記憶されてもよく、それによって、マシン読み取り可能媒体に記憶された命令は、機械に、フロー図及び／又はブロック図の種々のブロック（単数又は複数）に示された機能／動作を実施する種々の命令を含む製品を、生成させてもよい。

【0063】

図8は、トラフィック分離を維持するためのスレッドマネージャーを有する例示的コンピュータシステムを示している。コンピュータシステム又はホストは、プロセッサ801を含む。コンピュータシステムは、メモリ807を含む。メモリ807は、システムであってもよいし、あるいは、上で既に説明したマシン読み取り可能媒体の種々の実現可能形態のうちの何れか1つ又は複数であってもよい。コンピュータシステムは、ネットワークインターフェース805をさらに含む。コンピュータシステムは、スレッドマネージャ811をさらに含む。スレッドマネージャ811は、アプリケーションに組み込まれてもよいし、あるいは、複数の分離されたネットワークスタックを作成し、管理するアプリケーションによって使用されてもよい。アプリケーションは、複数の分離されたネットワークスタックを使用して、コンピュータシステムで受信された分離されたトラフィックの分離状態を維持する。各ネットワークスタックは、異なる論理ネットワーク若しくはサブネットワーク、又はそのグループに対応することがある。スレッドマネージャ811は、既存のスレッド割り当てを考慮して分離されたネットワークスタックにスレッドを割り当てることにより、ネットワークスタックへのスレッドの切り替え／割り当てのための、基礎となるオペレーティングシステムへの呼び出しのコストを回避する。前述の機能のうちの何れか1つは、ハードウェアで及び／又はプロセッサ801上で、部分的に（又は完全に）実施されてもよい。例えば、こうした機能は、特定用途向け集積回路を用いて実施されてもよいし、プロセッサ801において論理回路の形で実施されてもよく、あるいは、周辺機器又はカード上でコプロセッサ等の形で実施されてもよい。また、実現形態によっては、図8に示されていない少数の、又は追加のコンポーネント（例えば、ビデオカード、オーディオカード、追加のネットワークインターフェース、周辺機器など）を含む場合がある。プロセッサ801及びネットワークインターフェース805は、バス803に結合されている。メモリ807は、バス803に結合されるものとして例示されているが、プロセッサユニット801に結合されてもよい。本システムは、一組のストレージデバイス815をさらに含む。ストレージデバイス815としては、固体ストレージデバイス、ディスクストレージデバイス、及び、異なるタイプのストレージデバイスの混合などが挙げられる。ストレージデバイス815は、ネットワークインターフェースではないインターフェースを介して本システムに接続されてもよいし、あるいは、ネットワークインターフェース805を介して接続されてもよい。

10

20

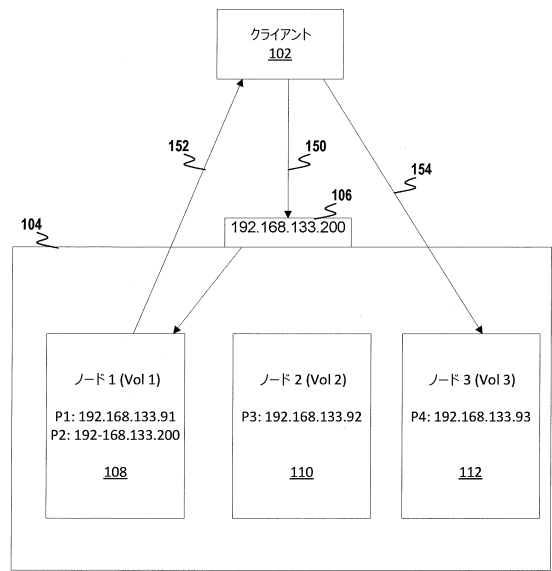
30

40

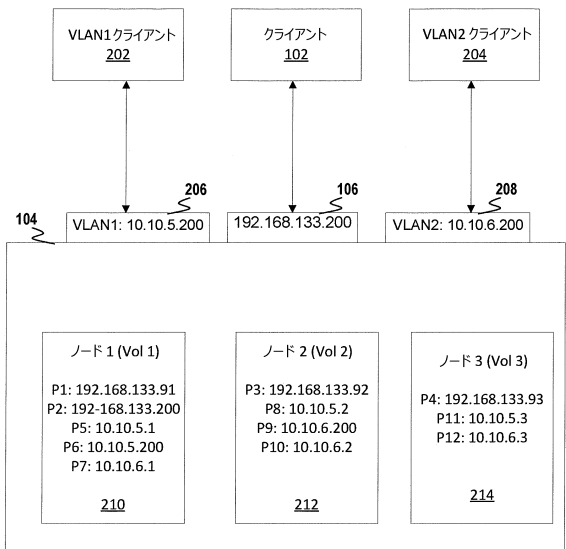
50

【図面】

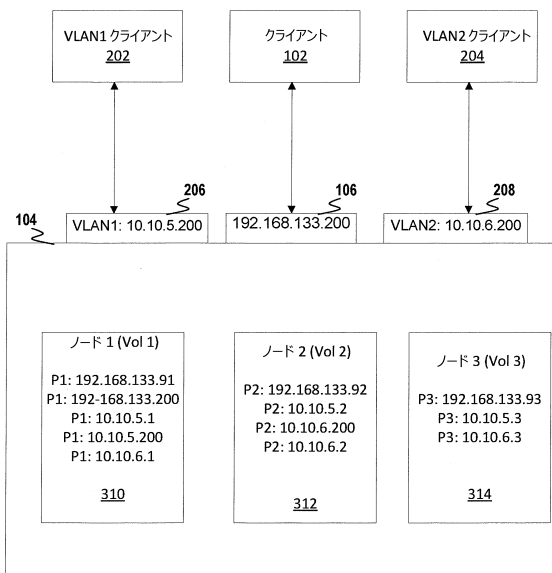
【図 1】



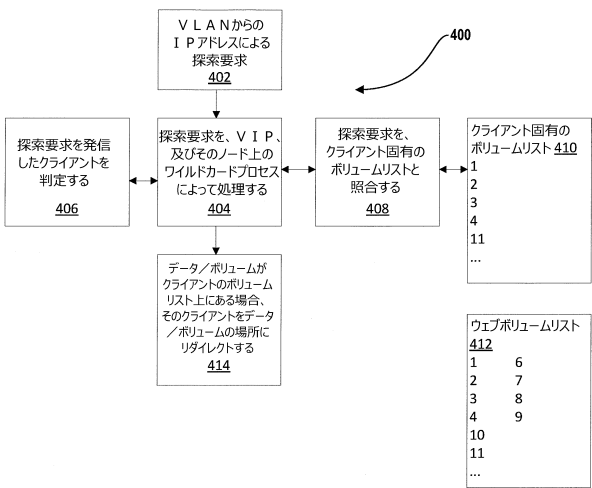
【図 2】



【図 3】



【図 4】



10

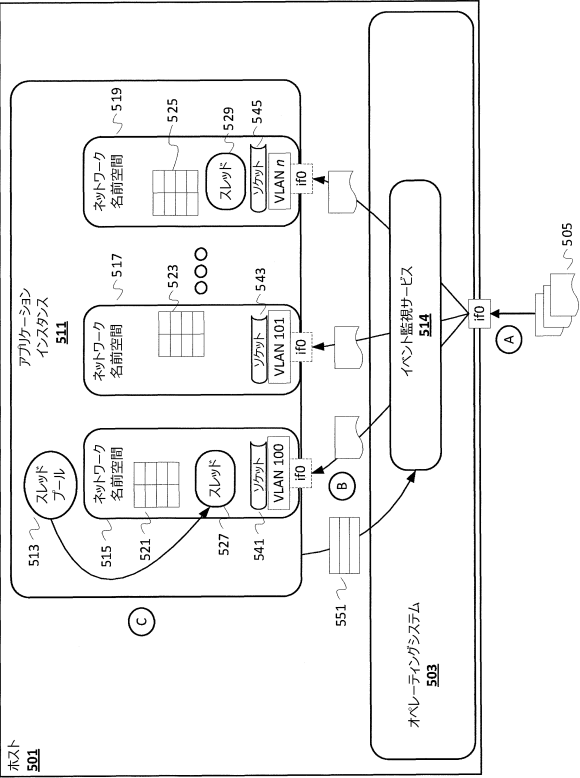
20

30

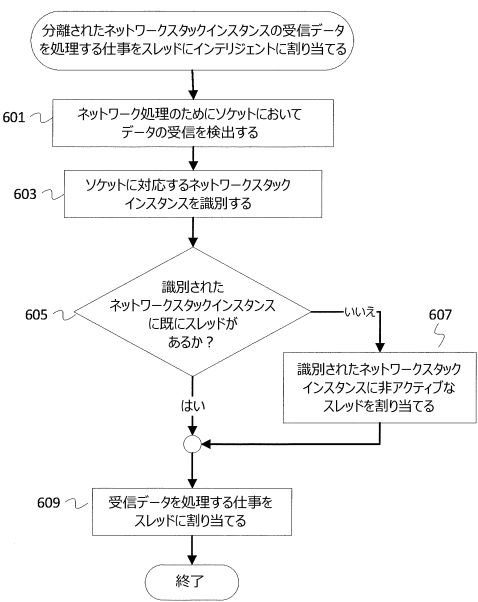
40

50

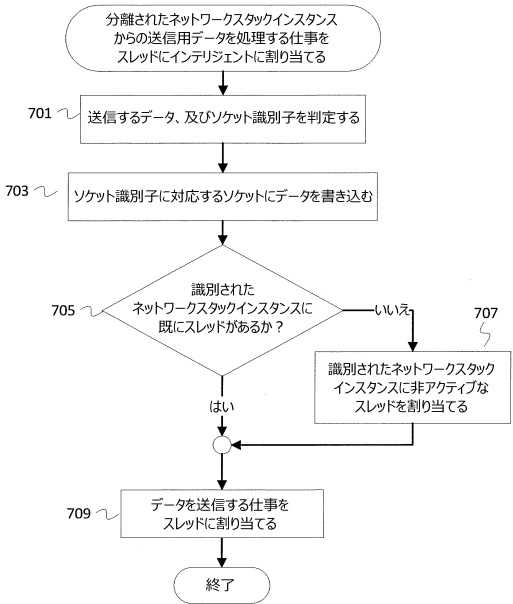
【図 5】



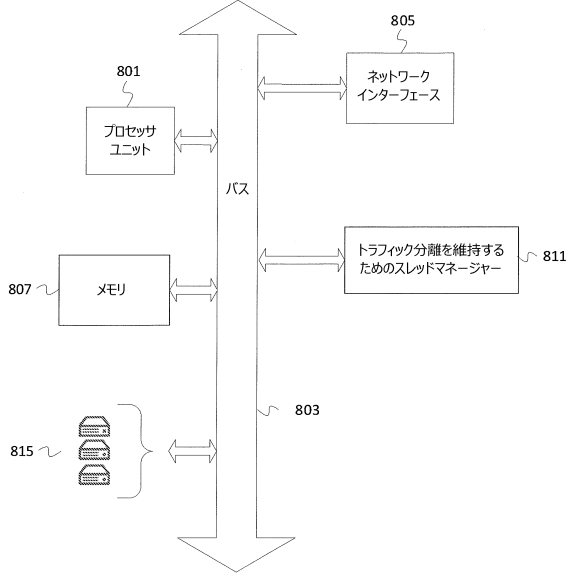
【図 6】



【図 7】



【図 8】



10

20

30

40

50

フロントページの続き

(33)優先権主張国・地域又は機関

米国(US)

(72)発明者 ヴァスキエヴィチ, ピーター, ピー, ジュニア

アメリカ合衆国オレゴン州97219, ポートランド, サウスウエスト・シックスティフォース・
アベニュー・12445

(72)発明者 キャントウェル, ジャレド

アメリカ合衆国コロラド州80301, ボルダー, セイント・ピーターズバーグ・ストリート・3
968

(72)発明者 マクマレン, マーシャル

アメリカ合衆国コロラド州80403, ゴールデン, クレセント・レイク・ロード・382

(72)発明者 シーリー, カール, ジェフレイ

アメリカ合衆国ネバダ州89521, レノ, カスケード・フォールズ・ドライブ・10100

審査官 漆原 孝治

(56)参考文献 PJ Waskiewicz, Scaling With Multiple Network Namespaces in a Single Application, 2016
年12月12日, <https://legacy.netdevconf.info/1.2/papers/pj-netdev-1.2.pdf>

(58)調査した分野 (Int.Cl., DB名)

G06F 9/50

H04L 61/00

H04L 45/50

G06F 9/54