Red Hat Enterprise Linux 7
ロードバランサーの管理

Red Hat Enterprise Linux 向け Load Balancer Add-On

Red Hat Enterprise Linux 7 ロードバランサーの管理

Red Hat Enterprise Linux 向け Load Balancer Add-On

法律上の通知

概要

Load Balancer Add-On システムを構築すると、Keepalived および HAProxy を用いて設定されたルーティングや負荷分散技術に対して特化した Linux Vertual Servers (LVS) を使用し、本番サービスに可用性および拡張性が高いソリューションが提供されます。本書では、Red Hat Enterprise Linux 7 のロードバランサー技術を使用した高パフォーマンスシステムおよびサービスの設定について説明します。

目次

第1章 ロードバランサーの概要

ロードバランサーは、複数の実サーバー間で IP トラフィックを分散する統合されたソフトウェアコンポーネントです。Keepalived と HAProxy の 2 つの技術を使用してクラスターメンバーとクラスターサービスを監視します。Keepalived は LVS を使用して負荷分散とフフェイルオーバーのタスクを実行します。HAProxy は TCP および HTTP アプリケーションに対して負荷分散および高可用性サービスを実行します。

1.1. keepalived

keepalived デーモンは、アクティブおよびパッシブ LVS ルーターの両方で実行されます。**keepalived** で実行されるすべてのルーターは*Virtual Redundancy Routing Protocol* (VRRP) を使用します。アクティブルーターは定期的な間隔で VRRP アドバタイズメントを送信します。バックアップルーターが VRRP アドバタイズメントを正常に受信しなかった場合、新しいアクティブルーターが選出されます。

アクティブルーターでは、**keepalived** は実サーバーの負荷分散タスクも実行できます。

Keepalived は LVS ルーターに関連する制御プロセスです。起動時、デーモンは **/etc/keepalived/keepalived.conf** 設定ファイルを読み取る **systemctl** コマンドによって起動されます。アクティブルーターでは、**keepalived** デーモンは LVS サービスを開始し、設定されたトポロジーを基にサービスの健全性を監視します。アクティブルーターは VRRP を使用して周期的なアドバタイズメントをバックアップルーターへ送信します。バックアップルーターでは、VRRP インスタンスがアクティブルーターの稼働状況を判断します。ユーザーが設定した間隔の後にアクティブルーターのアドバタイズメントが失敗すると、Keepalived はフェイルオーバーを開始します。フェイルオーバー中に仮想サーバーがクリアになります。新しいアクティブルーターが VIP を制御して、ARP メッセージを送信し、IPVS テーブルエントリー (仮想サーバー) を設定します。さらに、ヘルスチェックを開始し、VRRP アドバタイズメントの送信を開始します。

Keepalived は、TCP が接続ベースのデータ送信を実行するレイヤー4 (トランスポート層) でフェイルオーバーを実行します。実サーバーが単純なタイムアウト TCP 接続に応答しなかった場合、**keepalived** はサーバーの失敗を検出し、そのサーバーをサーバープールから削除します。

1.2. haproxy

HAProxy は、インターネットに接続されたサービスや Web ベースのアプリケーションなどの HTTP および TCP ベースのサービスに負荷分散されたサービスを提供します。選択したロードバランサースケジューリングアルゴリズムに応じて、**haproxy** は 1 つの仮想サーバーとして動作する複数の実サーバーのプール全体で、複数のイベントを数千個の接続で処理できます。スケジューラーは接続の量を判断し、非加重のスケジュールで均等に割り当てするか、加重を認識するアルゴリズムで高容量の処理が可能なサーバーにより多くの接続量を割り当てします。

HAProxy では、ユーザーは複数のプロキシーサービスを定義でき、プロキシーに対してトラフィックの負荷分散サービスを実行します。プロキシーは、1 つのフロントエンドと 1 つ以上のバックエンドで構成されます。フロントエンドは IP アドレス (VIP)、プロキシーがリッスンするポート、および特定のプロキシーに使用するバックエンドを定義します。

バックエンドは実サーバーのプールで、負荷分散アルゴリズムを定義します。

HAProxy はレイヤー 7 (アプリケーション層) で負荷分散管理を実行します。ほとんどの場合で、管理者は HTTP ベースの負荷分散 (ビジネスの継続に高可用性インフラストラクチャーが必要となる本番 Web アプリケーションなど) に HAProxy をデプロイします。

1.3. keepalived および haproxy

管理者は Keepalived と HAProxy を両方使用して堅牢でスケーラブルな高可用性環境を実現できます。Keepalived のフェイルオーバーサービスとともに、HAProxy のスピードとスケーラビリティーを使用して HTTP およびその他の TCP ベースのサービスに負荷分散を実行すると、実サーバー間で負荷を分散して可用性を高めることができ、バックアップルーターへのフェイルオーバーによってルーターが停止してもサービスが継続されます。

第2章 Keepalived の概要

Keepalived は*アクティブ LVS ルーター*や 1 つ以上の任意の*バックアップ LVS ルーター*で実行されます。アクティブ LVS ルーターには 2 つの役割があります。

- 複数の実サーバー全体の負荷分散
- それぞれの実サーバー上にあるサービスの整合性チェック

アクティブ (マスター) ルーターは、*Virtual Router Redundancy Protocol* (VRRP) を使用してアクティブな状態をバックアップルーターに通知します。マスタールーターは、通常の間隔でアドバタイズメントを送信する必要があります。アクティブルーターがアドバタイズメントの送信を停止すると、新しいマスターが選出されます。

本章は以下の項で構成され、Load Balancer コンポーネントおよび機能の概要を説明します。

- 「基本的な Keepalived ロードバランサーの設定」
- 「3 層の keepalived ロードバランサーの設定」
- 「keepalived スケジューリングの概要」
- 「ルーティングメソッド」
- 「Keepalived の永続性およびファイアウォールマーク」

2.1. 基本的な Keepalived ロードバランサーの設定

図2.1「基本的なロードバランサー設定」は、2 層で構成されるシンプルな Keepalived ロードバランサー設定を示しています。第 1 層には、1 つのアクティブ LVS ルーターと、複数のバックアップ LVS ルーターがあります。各 LVS ルーターには、インターネット上とプライベートネットワーク上にそれぞれ 1 つのネットワークインターフェースがあり、これらのネットワーク間のトラフィックを調整できるようになっています。この図の例では、アクティブルーターは*ネットワークアドレス変換* (*Netowrk Address Translation*、*NAT*) を使用してインターネットから第 2 層にある実サーバーへトラフィックを移動し、必要なサービスを提供します。そのため、この例の実サーバーは専用のプライベートネットワークセグメントに接続され、アクティブ LVS ルーターを介してすべてのパブリックトラフィックを送受信します。外部では、サーバーは 1 つのエンティティーのように見えます。

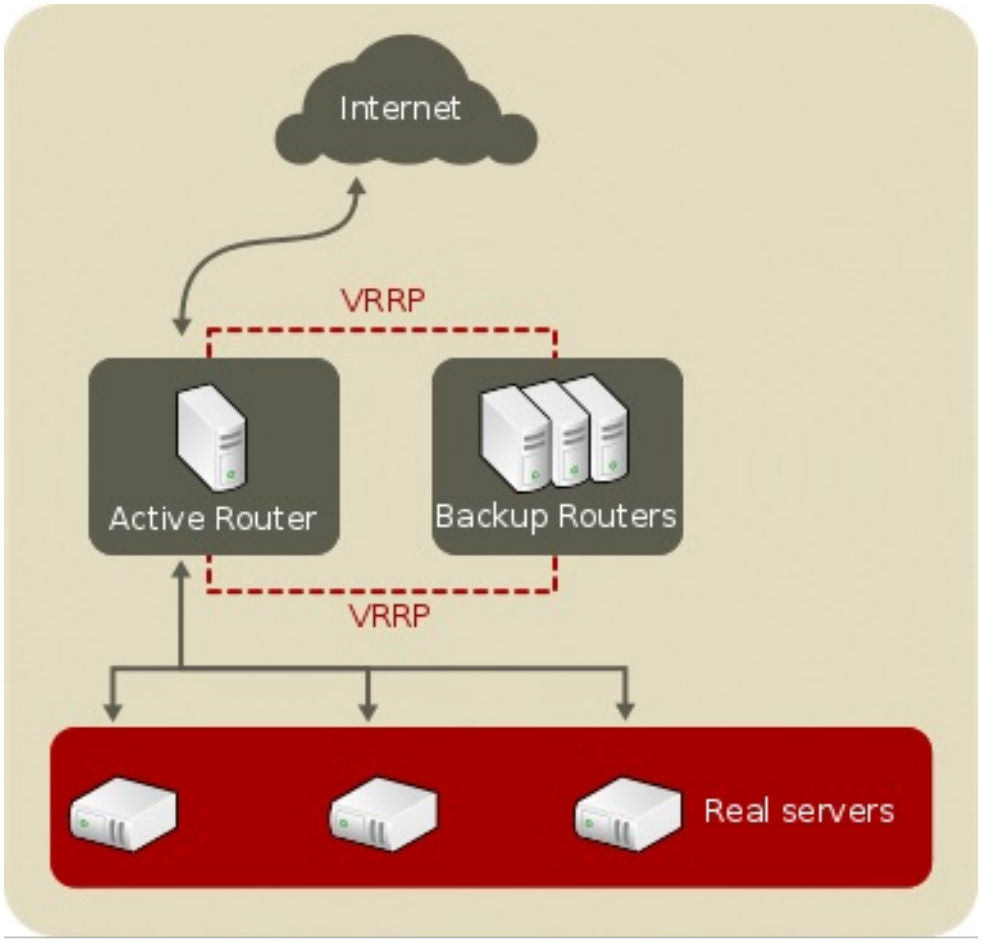

図2.1 基本的なロードバランサー設定

LVS ルーターへ送信されたサービス要求は *仮想 IP* アドレス (*VIP*) 宛になります。このアドレスはパブリックにルーティング可能なアドレスで、サイトの管理者が www.example.com などの完全修飾ドメイン名を関連付けし、1 つ以上の *仮想サーバー*へ割り当てます。仮想サーバーは、特定の仮想 IP でリッスンするよう設定されたサービスです。VIP アドレスはフェイルオーバー中に、ある LVS ルーターから別の LVS ルーターに移行するため、その IP アドレスで存在を維持します (**フローティング IP アドレス**とも呼ばれます)。

VIP アドレスは、LVS ルーターをインターネットに接続するデバイスへ割り当てられることがあります。たとえば、**eth0** がインターネットに接続されている場合、複数の仮想サーバーを**eth0** に割り当てることができます。この代わりに、各仮想サーバーをサービスごとに個別のデバイスへ関連付けることができます。たとえば、HTTP トラフィックを 192.168.1.111 の **eth0** で処理し、FTP トラフィックを 192.168.1.222 の **eth0** で処理することができます。

1 つのアクティブなルーターと 1 つのパッシブなルーターが関係するデプロイメントの場合、アクティブサーバーの役目は仮想 IP アドレスから実サーバーへサービス要求をリダイレクトすることです。リダイレクトは、「keepalived スケジューリングの概要」 に記載されている 8 つのサポートされる負荷分散アルゴリズムの 1 つを基にして実行されます。

アクティブルーターは、シンプルな TCP 接続、HTTP、および HTTPS の 3 つのビルトインヘルスチェックを使用して実サーバーで特定サービスの全体的な健全性を動的に監視します。TCP 接続では、アクティブルーターは特定のポートで実サーバーに接続できることを周期的にチェックします。HTTP および HTTPS では、アクティブルーターは周期的に実サーバーの URL をフェッチし、コンテンツを検証します。

バックアップルーターはスタンバイシステムの役割を担います。ルーターのフェイルオーバーは VRRP によって処理されます。起動時、すべてのルーターはマルチキャストグループに参加します。このマルチキャストグループは、VRRP アドバタイズメントの送受信に使用されます。VRRP は優先度ベースのプロトコルであるため、優先度が最も高いルーターがマスターとして選出されます。ルーターがマスターとして選出さ

れると、定期的な間隔で VRRP アドバタイズメントをマルチキャストグループへ送信します。

一定の期間内 (アドバタイズメントの間隔を基にした) にバックアップルーターがアドバタイズメントを受信しなかった場合、新しいマスターが選出されます。新しいマスターは VIP を引き継ぎ、*アドレス解決プロトコル* (Address Resolution Protocol、ARP) メッセージを送信します。ルーターがアクティブなサービスを返すと、バックアップまたはマスターになります。この動作はルーターの優先度によって決まります。

静的な Web ページのように頻繁には変更が行われないデータを提供する場合は、実サーバー同士がノード間でデータの自動同期を行わないため、 図2.1「基本的なロードバランサー設定」 のシンプルな二層設定が最適な設定になります。

2.2. 3 層の keepalived ロードバランサーの設定

図2.2「3 層の keepalived ロードバランサーの設定」は、典型的な 3 層の Keepalived ロードバランサートポロジーを示しています。この例では、アクティブ LVS ルーターは要求をインターネットから実サーバーのプールにルーティングします。その後、実サーバーはネットワーク上で共有データソースにアクセスします。

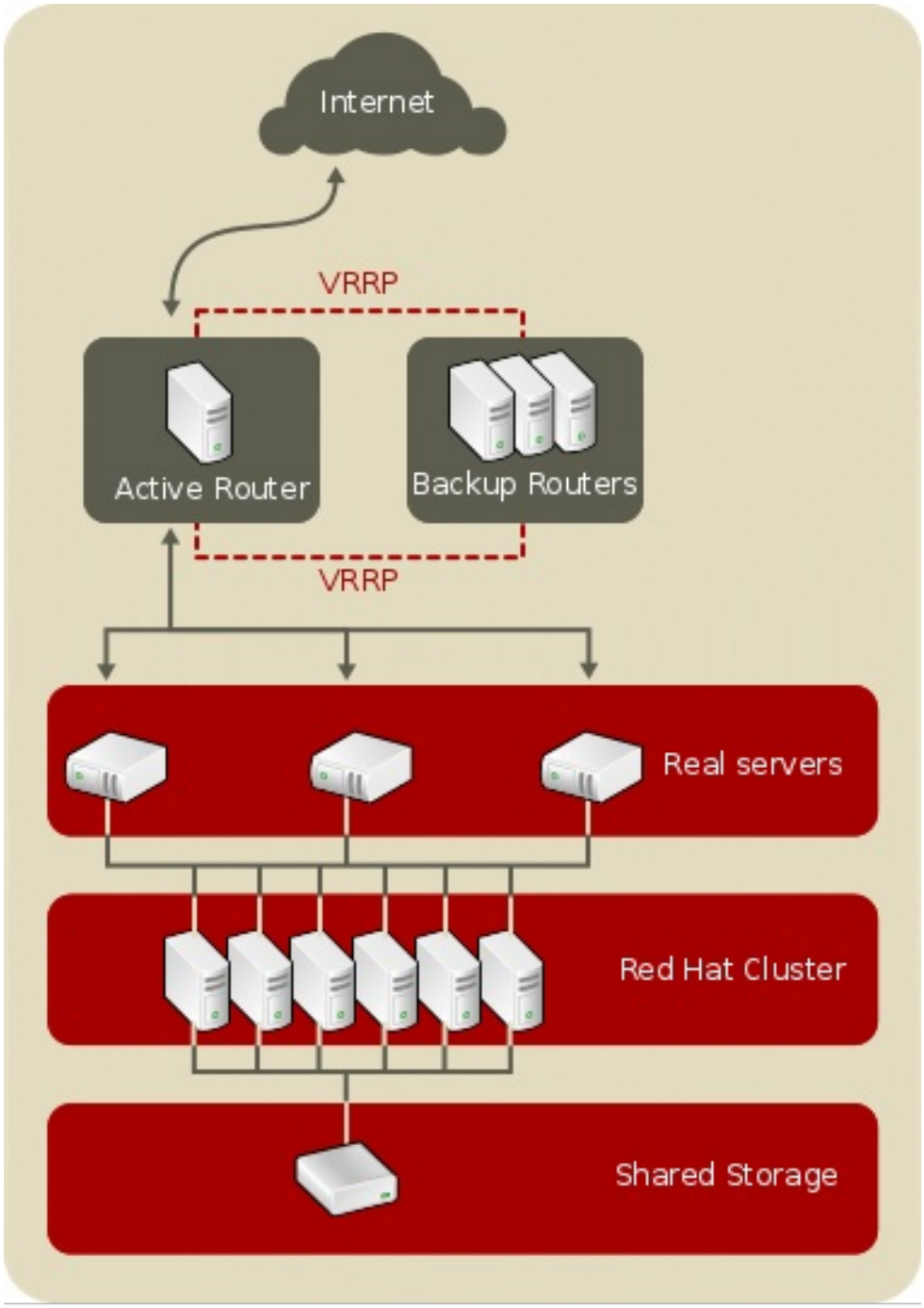

図2.2 3 層の keepalived ロードバランサーの設定

アクセス可能なデータが中央となる高可用性サーバーに格納されていて、実サーバーはエクスポートした NFS ディレクトリーまたは Samba 共有を使ってそのデータにアクセスするような構成がトラフィック量の多い FTP サーバーには理想的な構成と言えます。また、トランザクションのため中央となる高可用性データベースにアクセスを行うような Web サイトにもこのトポロジーをお勧めします。また、Load Balancer Add-Onでアクティブ-アクティブ設定を使用すると、これら両方の役割を同時に果たす高可用性クラスターを設定することもできます。

上記の例では、第三層で Load Balancer Add-Onを使う必要はありませんが、高可用性のソリューションを使用しないと重大な単一点障害をもたらすことになります。

2.3. keepalived スケジューリングの概要

Keepalived を使用すると、さまざまなスケジューリングアルゴリズムがサポートされるため、実サーバー

間でトラフィックを分散する柔軟性が高くなります。階層的な DNS やクライアントマシンのキャッシングによって負荷が不均等になる*ラウンドロビン DNS*などの柔軟性が低い方法よりも、負荷分散に優れています。さらに、ネットワークパケットレベルで負荷分散を行うと計算のオーバーヘッドが最小限になり、スケーラビリティーが向上するため、LVS ルーターが使用する低レベルのフィルタリングは、アプリケーションレベルの要求転送よりも優れています。

割り当てられた加重値を使用すると、各マシンに任意の優先度を提供します。このようなスケジューリングを使用すると、さまざまなハードウェアとソフトウェアの組み合わせを使用して実サーバーのグループを作成でき、アクティブなルーターは負荷を実サーバーへ均等に分散できます。

Keepalived のスケジューリングメカニズムは、*IP 仮想サーバー* または *IPVS* モジュールと呼ばれるカーネルパッチの集合によって提供されます。このモジュールは *レイヤー 4* (*L4*) トランスポート層スイッチングを有効にします。これは、単一 IP アドレス上で複数のサーバーとうまく機能するように設計されています。

パケットの追跡や実サーバーへのルーティングを効率的に行うため、IPVS はカーネルに *IPVS テーブル*を構築します。このテーブルは、アクティブ LVS ルーターによって使用され、要求を仮想サーバーアドレスからプールの実サーバーへリダイレクトしたり、プールの実サーバーから戻したりします。

2.3.1. keepalived スケジューリングアルゴリズム

IPVS テーブルの構造は、管理者が仮想サーバーに対して選択するスケジューリングアルゴリズムによって異なります。クラスター化できるサービスのタイプや、これらのサービスがスケジュールされる方法で柔軟性を最大限にするため、Keepalived は以下のスケジューリングアルゴリズムをサポートします

ラウンドロビンスケジューリング (Round-Robin Scheduling)

要求を順番に実サーバーのプールで分配します。このアルゴリズムを使用すると、容量や負荷に関係なく実サーバーはすべて同等に扱われます。このスケジューリングモデルはラウンドロビン DNS と似ていますが、ホストベースではなくネットワークベースであるため、粒度がより細かくなります。また、ロードバランサーのラウンドロビンスケジューリングでは、キャッシュされた DNS クエリーが原因で負荷が不均等になることはありません。

加重ラウンドロビンスケジューリング (Weighted Round-Robin Scheduling)

各要求を順番に実サーバーのプールで分散しますが、容量の多いサーバーにより多くのジョブを割り当てます。容量は、ユーザーが割り当てる加重係数によって示され、動的な負荷情報によって調整されます。

プール内の実サーバー間で処理能力に大幅な違いがある場合は、重み付きラウンドロビンスケジューリングが適しています。ただし、要求負荷が大きく変化する場合は、重みの大きいサーバーが割り当て以上の要求に応じる可能性があります。

最小接続 (Least-Connection)

実際の接続が少ない実サーバーにより多くの要求を振り分けます。IPVS テーブルで実サーバーへのライブ接続を継続的に追跡するため、Least-Connection は動的なスケジューリングアルゴリズムになります。要求負荷の変化が大きい場合に適しています。このアルゴリズムは各メンバーノードの処理能力がほぼ同じで大差がないような実サーバープールに最適です。サーバーグループの処理能力が異なる場合は weighted least-connection スケジューリングの方が適しています。

加重最小接続 (Weighted Least-Connections)

容量と比較してより少ないアクティブな接続を持つサーバーにより多くの要求を分散します。容量は、ユーザーが割り当てる加重係数によって示され、動的な負荷情報によって調整されます。実際のサーバープールに異なる容量のハードウェアが含まれる場合、加重が追加されるこのアルゴリズムが適しています。

ローカリティーベースの最小接続スケジューリング (Locality-Based Least-Connection Scheduling)

接続先 IP に対して相対的に実際の接続が少ないサーバーにより多くの要求を振り分けます。プロキシキャッシュサーバーのクラスターでの使用を目的として設計されています。任意の IP アドレスのパケットをその IP アドレスのサーバーに送信します。ただし、そのサーバーの負荷がその処理能力を超えている一方、別のサーバーの負荷は処理能力の半分に留まっている場合は、IP アドレスを負荷の一番少ない実サーバーに割り当てます。

複製スケジューリングを伴うローカリティベースの最小接続スケジューリング (Locality-Based Least-Connection Scheduling with Replication Scheduling)

接続先 IP に対して相対的に実際の接続が少ないサーバーにより多くの要求を振り分けます。このアルゴリズムもプロキシキャッシュサーバーのクラスターでの使用を目的として設計されています。Locality-Based Least-Connection スケジューリングとの違いは、目的 IP アドレスを実サーバーノードのサブセットにマッピングする点です。要求はマッピングされたサブセット内で接続数が最少となるサーバーにルーティングされます。接続先 IP のノードがすべて処理能力を越えてしまっている場合は、実サーバーのプール全体で接続が一番少ないサーバーをその接続先 IP の実サーバーサブセットに追加して、その接続先 IP アドレスの新しいサーバーを複製します。一方、過剰な複製を防ぐため、負荷の最も高いノードが実サーバーのサブセットから外されます。

接続先ハッシュスケジューリング (Destination Hash Scheduling)

静的なハッシュテーブル内の接続先 IP を検索して、実サーバーのプールに要求を振り分けます。プロキシキャッシュサーバーのクラスターでの使用を目的として設計されています。

ソースハッシュスケジューリング (Source Hash Scheduling)

静的なハッシュテーブル内のソース IP を検索して、実サーバーのプールに要求を振り分けます。複数のファイアウォールが設定される LVS ルーター向けに設計されています。

最短予測遅延 (Shortest Expected Delay)

サーバー上の接続数を割り当てられた加重値で割った値を基して、最も短い遅延が期待されるサーバーへ接続要求を分配します。

キューなし (Never Queue)

最初に接続要求を見つけ、アイドル状態または接続がないサーバーへ接続要求を送信する 2 面のスケジューラーです。アイドル状態のサーバーがない場合、スケジューラーはデフォルトで **最短予測遅延 (Shortest Expected Delay)** と同様に最も遅延が少ないサーバーを選択します。

2.3.2. サーバーの加重値とスケジューリング

ロードバランサーの管理者は、実際のサーバープールの各ノードに*加重値*を割り当てできます。この加重値は整数値で、**加重値を認識する**スケジューリングアルゴリズム (加重最小接続など) で考慮されます。また、LVS ルーターが異なる容量を持つハードウェアで負荷を均等にできるようにします。

加重値は、それぞれに対する相対的な割合として機能します。たとえば、ある実サーバーの加重値が 1 で別のサーバーの加重値が 5 の場合、前者が 1 回接続されるごとに後者が 5 回接続されます。実サーバーの加重値のデフォルト値は 1 です。

実サーバープール内でそれぞれ異なるハードウェア構成のノードに重みを付けるとクラスターでの負荷分散をより効率的に行う場合に役立ちますが、 weighted least-connection スケジューリングで仮想サーバーが設定されている際、任意の実サーバーがその実サーバープールに挿入されると、一時的に不均衡が生じる場合があります。例えば、実サーバープールに 3 台のサーバーがあったとします。サーバー A と B に 1 の重

みが付けられ、サーバー C には 2 の重みが付けられていたとします。サーバー C が何らかの理由でダウンした場合、放棄された負荷がサーバー A と B に均等に分散されます。しかし、サーバー C がオンラインに復帰すると、LVS ルーター側でサーバー C の接続がまったくないと判断され、サーバー A および B と同等になるまで着信要求をすべてサーバー C に集中的に振り分けることになります。

2.4. ルーティングメソッド

Red Hat Enterprise Linux は*ネットワークアドレス変換* (*NAT ルーティング*) または Keepalived のダイレクトルーティングを使用します。これにより、利用できるハードウェアを活用し、ロードバランサーを既存のネットワークに統合する場合に柔軟性が大変高くなります。

2.4.1. NAT ルーティング

図2.3「NAT ルーティングを実装したロードバランサー」は、ロードバランサーが NAT ルーティングを使用して、インターネットとプライベートネットワークの間で要求を移動する様子を示しています。

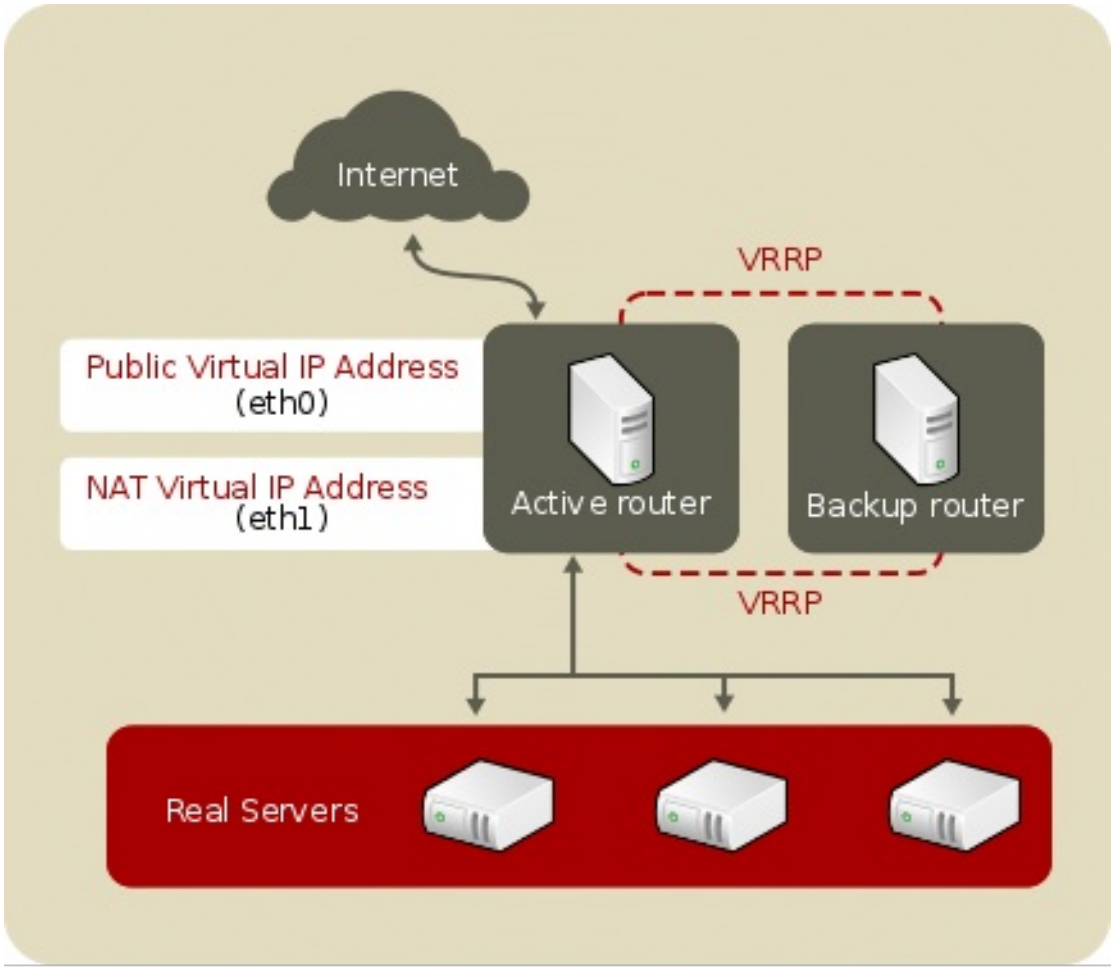

図2.3 NAT ルーティングを実装したロードバランサー

この例では、アクティブ LVS ルーターに NIC が 2 つあります。インターネットの NIC では*リアル IP アドレス*とフローティング IP アドレスが eth0 にあります。プライベートネットワークインターフェースの NIC はリアル IP アドレスとフローティング IP アドレスが eth1 にあります。フェイルオーバーが発生すると、インターネットに向けられている仮想インターフェイスと仮想インターフェイスに向けられているプライベートネットワークが同時にバックアップ LVS ルーターへ引き継ぎされます。プライベートネットワークにあるすべての実サーバーは、NAT ルーターのフローティング IP をアクティブ LVS ルーターと通信するデフォルトのルートとして使用します。こうすることで、インターフェイスからの要求に応答する能力が維持されます。

この例では、LVS ルーターのパブリックフローティング IP アドレスとプライベート NAT フローティング IP アドレスが物理 NIC に割り当てられます。フローティング IP アドレスを LVS ルーターノード上のそれぞれの物理デバイスに関連付けることは可能ですが、3 つ以上の NIC を使用する必要はありません。

このトポロジーを使うと、アクティブ LVS ルーターは要求を受信して適切なサーバーにルーティングします。実サーバーはその要求を処理してパケットを LVS ルーターに返します。LVS ルーターはネットワークアドレス変換を使ってパケット内の実サーバーのアドレスを LVS ルーターのパブリック VIP アドレスに置き換えます。実サーバーの本当の IP アドレスは要求を行っているクライアントからは見えないよう隠しているため、*IP マスカレード*と呼ばれます。

NAT ルーティングを使用する場合は、実サーバーにするマシンの種類や稼働させるオペレーティングシステムの種類に制限はありません。ただし、発信要求および着信要求のいずれも LVS ルーターで処理しなければならないため、大規模なクラスター導入の場合には LVS ルーターがボトルネックとなる場合があります。

2.4.2. ダイレクトルーティング

ダイレクトルーティングを使用するロードバランサー設定を構築すると、他のロードバランサーネットワーキングトポロジーよりもパフォーマンス上のメリットが大きくなります。ダイレクトルーティングでは、実サーバーは要求元のユーザーに対して直接パケットを処理およびルーティングでき、すべての送信パケットを LVS ルーター経由で渡しません。ダイレクトルーティングでは、LVS ルーターが受信パケットのみを処理するよう制限し、ネットワークパフォーマンスの問題が発生する可能性を低減します。

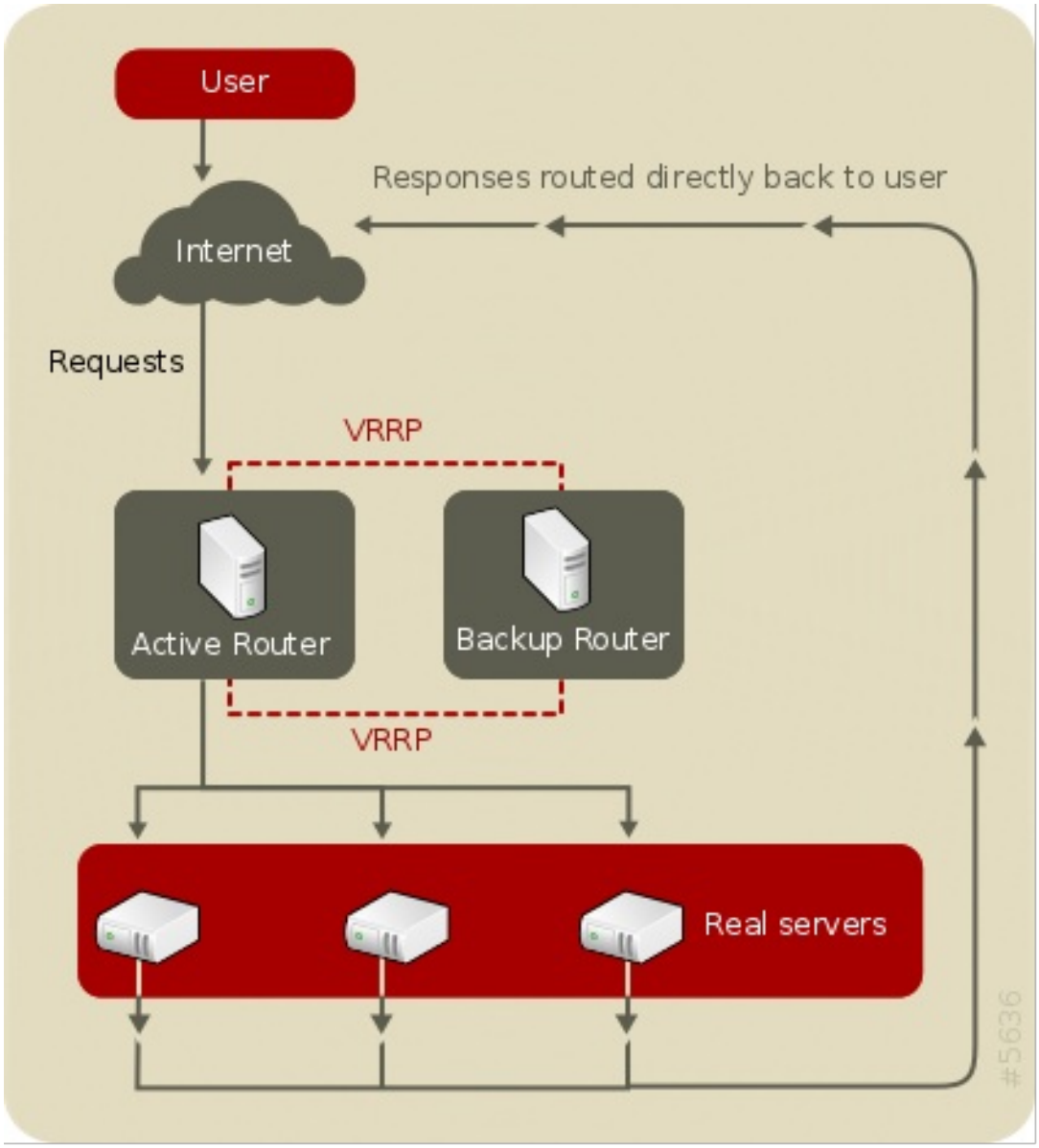

図2.4 ダイレクトルーティングで実装されたロードバランサー

典型的なダイレクトルーティングのロードバランサー設定では、LVS ルーターは仮想 IP (VIP) 経由で受信サーバー要求を受け取り、スケジューリングアルゴリズムを使用して要求を実サーバーへルーティングします。実サーバーは要求を処理し、LVS ルーターを通さずに応答を直接クライアントに送信します。ネットワークの負荷が高い状態で LVS ルーターが送信パッケージを実サーバーからクライアントへルーティングするとボトルネックになりますが、 このルーティング方法では LVS ルーターがこのような処理をしなくても実サーバーを追加できるため、スケーラビリティーが向上します。

2.4.2.1. ダイレクトルーティングと ARP 制限

ロードバランサーでダイレクトルーティングを使用する利点は多くありますが、制限もあります。最も一般的な問題は、*アドレス解決プロトコル*(ARP) に関する問題です。

インターフェース上のクライアントは要求を IP アドレスに送信します。ネットワークルーターは ARP を使って IP アドレスをマシンの MAC アドレスに関連付けることで要求を宛先に送信します。ARP 要求がネットワークに接続されているすべてのマシンにブロードキャストされ、IP アドレスと MAC アドレスの正しい組み合わせを持つマシンがパケットを受け取ることになります。IP と MAC の関連性は ARP キャッシュに保存され、定期的に消去と再保存が行われます (通常 15 分ごと)。

ダイレクトルーティングのロードバランサー設定における ARP 要求の問題は、IP アドレスへのクライアント要求が処理されるようにするには、その要求が MAC アドレスに関連付けされている必要があるため、

ロードバランサーシステムの仮想 IP アドレスも MAC に関連付けされている必要があることです。しかし、LVS ルーターと実サーバーは同じ VIP を持っているため、ARP 要求はその VIP に関連付けられているすべてのマシンへブロードキャストされます。これにより、VIP が直接実サーバーの 1 つに関連付けられ、直接要求を処理する問題や、完全に LVS ルーターを迂回し、ロードバランサー設定の目的に反する問題など、複数の問題が発生することがあります。

この問題を解決するには、着信要求が実サーバーではなく、必ず LVS ルーターに送信されるようにすることです。**arptables_jf** または **iptables** パケットフィルタリングツールを使用すると以下の理由でこれを行うことができます。

- **arptables_jf** は ARP が VIP と実サーバーを関連付けないようにします。
- **iptables** メソッドの場合、実サーバー上での VIP 設定はまったく行わないため ARP に関する問題を完全回避することができます。

2.5. Keepalived の永続性およびファイアウォールマーク

状況によっては、負荷分散アルゴリズムを使用して利用可能な最良のサーバーへ要求を送信せずに、クライアントが同じ実サーバーに繰り返し再接続する方が望ましいことがあります。このような場合の例には、マルチスクリーン Web フォーム、クッキー、SSL、および FTP 接続などがあります。このようなケースでは、同じサーバーがトランザクションを処理し、コンテキストを保持しないとクライアントが適切に動作しないことがあります。Keepalived は、*永続性* と *ファイアウォールマーク* の 2 つの機能でこれに対応します。

2.5.1. 永続性

永続性が有効になっていると、タイマーのように動作します。クライアントがサービスに接続すると、ロードバランサーは指定期間中は最後の接続を記憶します。同じクライアント IP アドレスが同じ期間内に再接続すると、以前接続したサーバーへ送信され、負荷分散メカニズムが無視されます。接続が指定期間外で発生すると、対応するスケジューリングルールに従って処理されます。

また、永続性を使うと、管理側でサブネットマスクを指定して、どのアドレスがより高い永続性を持つかを管理するツールとして、クライアント IP アドレステストに適用することができます。こうすることで、接続をそのサブネットにグループ化できます。

通信に複数のポートを使用する FTP などのプロトコルの場合、宛先が別々のポートの接続をグループ化することがとても重要な場合があります。ただし、宛先が別々のポートの接続をグループ化する上で発生する問題に対処する場合、永続性は最も効率的な方法とは言えません。このような場合には *ファイアウォールマーク* を使用するのが最適です。

2.5.2. ファイアウォールマーク

ファイアウォールマークは、プロトコルまたは関連するプロトコルのグループに使用されるポートを簡単かつ効率的にグループ化する方法です。たとえば、EC (E コマース) サイトの稼働が目的でロードバランサーがデプロイされた場合、ファイアウォールマークを使用してポート 80 の HTTP 接続と、ポート 443 のセキュアな HTTPS 接続をバンドル化できます。各プロトコルの仮想サーバーに同じファイアウォールマークを割り当てると、接続が開かれた後に LVS ルーターがすべての要求を同じ実サーバーへ転送するため、トランザクション状態の情報を保持できます。

ファイアマークは効率的で簡単に使用できるため、ロードバランサーの管理者は可能な限り永続性の代わりにファイアウォールマークを使用して接続をグループ化するようにしてください。しかし、クライアントが十分な期間同じサーバーに再接続されるようにするため、ファイアウォールマークとともに永続性を仮想サーバーに追加するようにしてください。

第3章 Keepalived のロードバランサー要件の設定

Keepalived を使用するロードバランサーは、LVS ルーターと実サーバーの 2 つの基本グループで構成されます。単一障害点を回避するため、各グループに最低でも 2 つのメンバーシステムが含まれるようにしてください。

LVS ルーターグループは、Red Hat Enterprise Linux を実行している同一または非常に似ている二つのシステムで構成する必要があります。そのうちの 1 つはアクティブ LVS ルーターとして機能し、もう 1 つはホットスタンバイモードで待機するので、この 2 つができるだけ同じキャパシティを備えている必要があります。

実際のサーバーグループのハードウェアを選択および設定する前に、3 つのロードバランサートポロジーのどれを使用するかを決定します。

3.1. NAT ロードバランサーネットワーク

NAT トポロジーを使用すると、既存のハードウェアを柔軟に活用できますが、プールから送受信されるすべてのパケットはロードバランサールーターを通過するため、大量の負荷を処理する機能が制限されます。

ネットワークレイアウト

NAT ルーティングを使用するロードバランサーのトポロジーは、パブリックネットワークへのアクセスポイントが 1 つのみ必要なため、ネットワークレイアウトの観点から見ると最も設定が簡単です。実サーバーは LVS ルーター経由ですべての要求を返すため、独自のプライベートネットワーク上にあります。

ハードウェア

正常に機能させるため実サーバーを Linux マシンにする必要はないため、ハードウェアに関しては NAT トポロジーが最も柔軟なトポロジーになります。各実サーバーが応答するのは LVS ルーターのみのため、実サーバー側に必要な NIC は 1 つのみになります。一方、LVS ルーターでは 2 種類のネットワークのトラフィックを別々にルーティングさせるため NIC が 2 つ必要になります。このトポロジーの場合、LVS ルーターの部分がネットワークのボトルネックになるため、各 LVS ルーターにギガビットイーサネットの NIC を使用し LVS ルーターで処理できる帯域幅を増大させることが可能です。LVS ルーターにギガビットイーサネットを使用する場合は、負荷を効率的に処理するため実サーバーを LVS ルーターに接続しているスイッチについてもギガビットイーサネットポートが少なくとも 2 つ搭載されているスイッチが必要になります。

ソフトウェア

NAT トポロジーでは、設定によっては **`iptables`** を使用する必要があるため、Keepalived 外部で多くのソフトウェアを設定する場合があります。特に、FTP サービスやファイアウォールマークを使用する場合は、要求を適切にルーティングするために LVS ルーターを追加で手動設定する必要があります。

3.1.1. NAT を使用するロードバランサーのネットワークインターフェース設定

NAT を使用するロードバランサーを設定するには、最初に LVS ルーターのパブリックネットワークとプライベートネットワークのネットワークインターフェースを設定する必要があります。この例では、LVSルーターのパブリックインターフェース (**`eth0`**) を 192.168.26/24 ネットワークに設定し (これはルーティング可能な IP ではありませんが、LVS ルーターの前にファイアウォールがあることを仮定します)、実サーバーにリンクするプライベートインターフェース (**`eth1`**) を 10.11.12/24 ネットワークに設定します。

重要

network サービスおよびロードバランサーに関する以下のファイルの編集は**NetworkManager** サービスには対応しません。

アクティブ (プライマリー) LVS ルーターノードのパブリックインターフェースのネットワークスクリプト **/etc/sysconfig/network-scripts/ifcfg-eth0** の例を以下に示します。

```
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.26.9
NETMASK=255.255.255.0
GATEWAY=192.168.26.254
```

LVS ルーターのプライベート NAT インターフェースのスクリプト **/etc/sysconfig/network-scripts/ifcfg-eth1** の例を以下に示します。

```
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.11.12.9
NETMASK=255.255.255.0
```

この例では、LVS ルーターのパブリックインターフェースの VIP は 192.168.26.10、NAT (プライベート) インターフェースの VIP は 10.11.12.10 になります。したがって、実サーバーが要求を返すのは NAT インターフェースの VIP になると言う点に注意してください。

重要

本項のイーサネットインターフェース設定の例は、LVS ルーターのリアル IP アドレス向けの例で、フローティング IP アドレス向けの例ではありません。

アクティブ LVS ルーターノードのネットワークインターフェースを設定したら、バックアップ LVS ルーターの実ネットワークインターフェースの設定を行います。IP アドレスがネットワーク上の他の IP アドレスと競合しないよう注意してください。

重要

バックアップノード上の各インターフェースがアクティブノード上のインターフェースと同じネットワークに接続するようにしてください。例えば、アクティブノードで eth0 がパブリックネットワークに接続されている場合には、バックアップノードでも eth0 をパブリックネットワークに接続してください。

3.1.2. 実サーバー上でのルーティング

NAT トポロジーで実サーバーのネットワークインターフェースを設定する場合、忘れてはいけないもっとも重要な作業が LVS ルーターの NAT フローティング IP アドレス用にゲートウェイを設定することです。この例では、ゲートウェイは 10.11.12.10 になります。

注記

実サーバー上でのネットワークインターフェース設定が完了すると、マシンは他の方法でパブリックネットワークに ping したり接続したりすることができなくなります。これは正常なことです。しかし、LVS ルーターのプライベートインターフェースの実 IP、この場合は 10.11.12.9、に ping することはできます。

実サーバーの **/etc/sysconfig/network-scripts/ifcfg-eth0** ファイルは以下のようになります。

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.11.12.1
NETMASK=255.255.255.0
GATEWAY=10.11.12.10
```

警告

実サーバーの **GATEWAY=** 行に複数にネットワークインターフェースが設定されている場合、最初に起動したネットワークインターフェースがゲートウェイを取得します。そのため、**eth0** と **eth1** の両方が設定され、**eth1** がロードバランサーに使用されている場合、実サーバーは要求を適当にルーティングしないことがあります。

関係ないネットワークインターフェースはオフにするのが最適です。**/etc/sysconfig/network-scripts/** ディレクトリ内の各ネットワークスクリプトで **ONBOOT=no** とセットするか、または 1 番目に記載されているインターフェースでゲートウェイが正しく設定されているか確認します。

3.1.3. LVS ルーターで NAT ルーティングを有効にする

クラスター化されたサービスが 1 つのポート (例: ポート 80 の HTTP) のみを使用する簡単な NAT ロードバランサー設定の場合、LVS ルーター上でパケット転送のみを有効にすれば実サーバーとその外部との間で適切に要求をルーティングできます。しかし、ユーザーセッション中に同じ実サーバーに移動するため、クラスター化されたサービスが複数のポートを必要とする場合は、他の設定が必要になります。

LVS ルーターで転送が有効になり、実サーバーが設定されクラスター化されたサービスが稼働したら、**keepalived** を使用して IP の情報を設定します。

警告

手動でネットワークを編集したり、ネットワーク設定ツールを使用したりして **eth0** または **eth1** のフローティング IP を設定しないでください。必ず **keepalived.conf** ファイルで設定してください。

設定が終了したら、**keepalived** サービスを起動します。起動したら、アクティブ LVS ルーターが要求を実サーバーのプールへルーティングし始めます。

3.2. ダイレクトルーティング経由のロードバランサー

ダイレクトルーティングを使用すると、実サーバーは LVS ルーターを介して送信パケットを渡さずに、直接パケットを処理して要求元ユーザーにルーティングできます。ダイレクトルーティングでは、実サーバーが LVS ルーターがあるネットワークセグメントに物理的に接続している必要があり、送信パケットを処理およびルーティングできる必要があります。

ネットワークレイアウト

ダイレクトルーティングのロードバランサー設定では、LVS ルーターが受信要求を受け取り、要求を処理する適切な実サーバーへルーティングする必要があります。その後、実サーバーは応答を**直接**クライアントへルーティングする必要があります。たとえば、クライアントがインターネット上にあり、LVS ルーター経由でパケットを実サーバーへ送信する場合、実サーバーはインターネット経由で直接クライアントに接続できる必要があります。これを実現するには、実サーバーのゲートウェイを設定し、パケットをインターネットに渡します。サーバープールの各サーバーは独自のゲートウェイを持ちます (各ゲートウェイはインターネットへの独自の接続を持ちます)。そのため、スループットやスケーラビリティーを最大限にします。しかし、一般的なロードバランサーの設定では実サーバーは 1 つのゲートウェイ (よって 1 つのネットワーク接続) を経由して通信できます。

ハードウェア

ダイレクトルーティングを使用するロードバランサーシステムのハードウェア要件は、他のロードバランサートポロジーと似ています。LVS ルーターは PROD; を実行して受信要求を処理し、実サーバーの負荷分散を実行する必要がありますが、実サーバーが適切に機能するには Linux マシンである必要はありません。各 LVS ルーターには 1 つまたは 2 つの NIC が必要です (バックアップルーターがあるかによって異なります)。NIC を 2 つ使用すると設定が容易になり、受信要求は 1 つの NIC によって処理され、別の NIC によってパケットが実サーバーへルーティングされるため、トラフィックを区別できます。

実サーバーは LVS ルーターを迂回して送信パケットを直接クライアントに送信するため、インターネットへのゲートウェイが必要となります。パフォーマンスと可用性を最大化するには、クライアントが接続しているキャリアネットワーク (インターネットやイントラネットなど) に専用接続がある独自のゲートウェイに実サーバーを接続します。

ソフトウェア

ダイレクトルーティングでロードバランサーを使用するときに ARP の問題が発生する場合は、特に **keepalived** 以外の設定を行う必要があります。詳細は「ダイレクトルーティングおよび **arptables_jf**」または「ダイレクトルーティングと **iptables**」を参照してください。

3.2.1. ダイレクトルーティングおよび arptables_jf

arptables_jf を使用してダイレクトルーティングを設定するには、実サーバーでそれらの仮想 IP アドレスが設定されており、パケットが直接送信で可能となっている必要があります。VIP 用の ARP 要求は実サーバーでは完全に無視されます。そ例外の、VIP を含んでいて送信される ARP パケットは、mangle 化されて VIP ではなく実サーバーの IP が含まれるようになります。

arptables_jf メソッドを使用すると、アプリケーションは実サーバーが接続している個別の VIP またはポートにバインドします。例えば、**arptables_jf** メソッドの使用により、Apache HTTP Server の複数のインスタンスはシステム上の異なる VIP に明示的にバインドして実行することが可能になります。また、**arptables_jf** の使用は、**iptables** オプションよりもパフォーマンスで大きな利点があります。

しかし、**arptables_jf** メソッドを使うと、標準の Red Hat Enterprise Linux システム設定ツールを使用して起動時に VIP を開始する設定ができません。

それぞれの仮想 IP アドレスの ARP 要求を無視するように実サーバーを設定するには、以下の手順を実行します。

1. 実サーバー上で仮想 IP アドレス用に ARP テーブルのエントリを作成します (real_ip とは実サーバーとの通信にディレクタが使用する IP のこと。多くの場合、**eth0** にバインドされた IP)。

```
arptables -A IN -d <virtual_ip> -j DROP
arptables -A OUT -s <virtual_ip> -j mangle --mangle-ip-s <real_ip>
```

これにより、仮想 IP アドレス向けのすべての ARP 要求を実サーバーが無視するようになります。また、他の方法では仮想 IP を含むことになる送信 ARP 反応を変更させて、それらがサーバーの実 IP を含むようになります。VIP の ARP 要求に反応する唯一のノードは、現在アクティブな LVS ノードです。

2. これが実サーバー上で完了したら、実サーバー上で以下のコマンドを入力して ARP テーブルのエントリを保存します。

service arptables_jf save

chkconfig --level 2345 arptables_jf on

chkconfig コマンドは、ネットワーク開始前にシステムが起動時に arptables 設定をリロードするようにします。

3. **ip addr** コマンドを使用してすべての実サーバーに仮想 IP アドレスを設定し、IP エイリアスを作成します。例を以下に示します。

```
# ip addr add 192.168.76.24 dev eth0
```

4. ダイレクトルーティングを Keepalived に設定します。これには、***lb_kind DR*** を **keepalived.conf** ファイルに追加します。詳細は *4章Keepalived を用いたロードバランサーの初期設定* を参照してください。

3.2.2. ダイレクトルーティングと iptables

iptables ファイアウォールルールを作成することで、 ダイレクトルーティングメソッドを使用した場合の ARP 問題を回避することもできます。**iptables** を使用してダイレクトルーティングを設定するには、VIP アドレスがシステム上に存在しなくても VIP アドレスに送信されたパケットを実サーバーが扱うように透過プロキシを作成するルールを追加する必要があります。

iptables メソッドは **arptables_jf** メソッドよりも 設定が簡単です。仮想 IP アドレスがアクティブ LVS ディレクタ上にのみ存在するため、このメソッドでは LVS ARP 問題も完全に回避できます。

しかし、パケットの転送/マスカレードでのオーバーヘッドがあるため、**arptables_jf** と比較すると、**iptables** メソッドの使用にはパフォーマンスの問題があります。

また、**iptables** メソッドを使用してポートを再利用することはできません。例えば、二つの別々の Apache HTTP Server サービスは 両方とも仮想 IP アドレスではなく ***INADDR_ANY*** にバインドする必要があるため、ポート 80 にバインドされた二つの別々の Apache HTTP Server サービスを実行することはできません。

iptables メソッドを使用してダイレクトルーティングを設定するには、以下の手順を実行します。

1. 実サーバーでの実行が意図された VIP、ポート、プロトコル (TCP または UDP)の組み合わせすべてに、以下のコマンドを実行します。

iptables -t nat -A PREROUTING -p <tcp|udp> -d <vip> --dport <port> -j REDIRECT

このコマンドで、実サーバーは与えられた VIP とポートが宛先となっているパケットを処理します。

2. 実サーバー上で設定を保存します。

```
# service iptables save
# chkconfig --level 2345 iptables on
```

上記のコマンドで、システムはネットワーク開始前に、起動時に **iptables** 設定をリロードします。

3.2.3. ダイレクトルーティングと sysctl

sysctl インターフェースを使用して、ダイレクトルーティングの使用時に ARP の制限に対処することもできます。管理者は 2 つの **systcl** 設定を行い、実サーバーが ARP 要求で VIP をアナウンスしないようにし、VIP アドレスの ARP 要求へ応答しないようにします。この動作を有効にするには、以下のコマンドを実行します。

```
echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
```

この代わりに、以下の行を /etc/sysctl.d/arp.conf に設定することもできます。

```
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.eth0.arp_announce = 2
```

3.3. 設定を組み合わせる

上記のルーティングメソッドでどれを使用するか決定した後、ハードウェアをネットワーク上でリンクさせます。

重要

LVS ルーター上のアダプタデバイスは、同じネットワークにアクセスするように設定する必要があります。例えば、**eth0** がパブリックネットワークに接続し、**eth1** がプライベートネットワークに接続する場合、バックアップ LVS ルーター上の同じデバイスは同じネットワークに接続する必要があります。

また、起動時に最初に出現するインターフェースにリストされているゲートウェイは、ルーティングテーブルに追加され、他のインターフェースにリストされているそれ以降のゲート ウェイは無視されます。これは、実サーバーを設定する場合に特に考慮すべき重要点です。

ハードウェアを物理的に接続したら、プライマリーおよびバックアップ LVS ルーターにネットワークインターフェースを設定します。これは、**system-config-network** などのグラフィカルアプリケーションを使用するか、ネットワークスクリプトを手動で編集して行います。**system-config-network** を使用してデバイスを追加する方法の詳細は、 『Red Hat Enterprise Linux デプロイメントガイド』の『ネットワークの設定』の章を参照してください。

3.3.1. 一般的なロードバランサーネットワーキングのヒント

Keepalived を使用してロードバランサーを設定する前に、LVS ルーターのパブリックおよびプライベートネットワークにリアル IP アドレスを設定します。各トポロジーのセクションには、ネットワークアドレスの例が記載されていますが、実際のネットワークアドレスが必要になります。以下は、ネットワークインターフェースを起動し、状態をチェックするのに便利なコマンドの一部になります。

実ネットワーキングインターフェースの起動

実ネットワークインターフェースを起動するには、root で以下のコマンドを実行して、 *N* の部分をインターフェースに相当する番号で置き換えます (**eth0** および **eth1**)。

```
/usr/sbin/ifup ethN
```

警告

ifup スクリプトを使用して、Keepalived を使用して設定するフローティング IP アドレス (**eth0:1** または **eth1:1**) を起動しないでください。**service** または **systemctl** コマンドを使用して **keepalived** を開始してください。

実ネットワークインターフェースの停止

実ネットワークインターフェースを停止するには、root で以下のコマンドを実行して、 *N* の部分をインターフェースに相当する番号で置き換えます (**eth0** および **eth1**)。

```
/usr/sbin/ifdown ethN
```

ネットワークインターフェースのステータスチェック

ある時点でどのネットワークインターフェースが起動しているかをチェックするには、以下を入力します。

```
/usr/sbin/ifconfig
```

マシンのルーティングテーブルを表示するには、以下のコマンドを実行します。

```
/usr/sbin/route
```

3.4. マルチポートサービスとロードバランサー

マルチポートのロードバランサーサービスを作成する場合、LVS ルーターはどのトポロジーでも追加の設定が必要になります。HTTP (ポート 80) および HTTPS (ポート 443) などの異なる関連するプロトコルをバンドル化するファイアウォールマークを使用したり、FTP などの実際のマルチポートプロトコルを用いてロードバランサーを使用したりする場合、マルチポートサービスは人為的に作成されます。どちらの場合でも、LVS ルーターはファイアウォールマークを使用して異なるポートに送信されるパケットを認識しますが、同じファイアマークが付けられているため同様に処理する必要があります。また、ファイアウォールマークを永続性と組み合わせると、永続性パラメーターによって指定された期間内に接続が発生する限り、クライアントマシンからの接続は同じホストへルーティングされます。

実サーバーで負荷を分散するために使用される IPVS は、パケットに割り当てられたファイアウォールマークを認識できますが、ファイアウォールを割り当てることはできません。ファイアウォールマークの**割り当て**は、ネットワークパケットフィルターである **iptables** が実行する必要があります。

3.4.1. ファイアウォールマークの割り当て

送信先が特定ポートとなっているパケットにファイアウォールマークを割り当てるには、管理者は **iptables** を使用する必要があります。

ここでは、HTTP と HTTPS をバンドルする方法を例を用いて説明しますが、FTP も一般的に使用されるク
ラスター化されたマルチポートプロトコルです。

ファイアウォールマークを使用する場合の基本ルールとして、Keepalived でファイアウォールマークを使
用する各プロトコルには、ファイアウォールマークをネットワークパケットに割り当てるための同等の
iptables ルールが必要になります。

ネットワークパケットのフィルタルールを作成する前に、既に他のルールが存在しないか確認します。これ
を行うには、シェルプロンプトを開いて、root でログインして以下を入力します。

```
/usr/sbin/service iptables status
```

iptables が実行されていない場合、すぐにプロンプトが再出現します。

iptables がアクティブな場合、ルールセットが表示されます。ルールが存在する場合、以下のコマンド
を入力します。

```
/sbin/service iptables stop
```

既存のルールが重要な場合、**/etc/sysconfig/iptables** の内容を確認して、保存する価値のあるルー
ルを安全な場所にコピーしてから継続します。

ファイアウォールルールの設定に関連する最初のロードバランサーは、Keepalived サービスの VRRP トラ
フィックを許可します。

```
/usr/sbin/iptables -I INPUT -p vrrp -j ACCEPT
```

以下のルールでは、同一ファイアウォールマーク 80 を、送信先がポート 80 と 443 上のフローティング
IP アドレス *n.n.n.n* になっている着信トラフィックに割り当てます。

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 -m
multiport --dports 80,443 -j MARK --set-mark 80
```

初めてルールを発行する前に、root としてログインし、**iptables** のモジュールをロードする必要があり
ます。

上述の **iptables** コマンドの *n.n.n.n* は、使用中の HTTP および HTTPS 仮想サーバーのフローティング
IP で置き換える必要が あります。これらのコマンドは、該当するポート上の VIP が送信先となっている全
トラフィックをファイアウォールマーク 80 に割り当てることと同様の効果があります。これが IPVS に認
識され、適切に転送されます。

警告

上記のコマンドは即座に有効になりますが、システムを再起動すると保持されません。

3.5. FTP の設定

ファイル転送プロトコル (FTP) は旧式の複雑なマルチポートプロトコルで、ロードバランサーを使用する
環境では扱いが難しくなります。扱いの難しさを理解するには、初めに FTP の仕組みで重要となる点を理
解する必要があります。

3.5.1. FTP の動作

ほとんどのサーバー/クライアント関係では、クライアントマシンが特定のポート上でサーバーへ接続を開い

て、サーバーがそのポートのクライアントに応答します。FTP クライアントが FTP サーバーに接続する場合、FTP 制御ポート 21 への接続を開きます。そして、その **クライアント**が FTP **サーバー**に *アクティブ* か *パッシブ* のどちらの接続を開くかを指示します。クライアントが選択した接続タイプにより、サーバーの対応方法とトランザクションが発生するポートを決定します。

データ接続は以下の 2 種類です。

アクティブ接続

アクティブ接続が確立されると、**サーバー**はポート 20 からクライアントマシン上の高い範囲のポートにクライアントへデータ接続を開きます。サーバーからのすべてのデータは、この接続を通じて送信されます。

パッシブ接続

パッシブ接続が確立されると、**クライアント**は FTP サーバーに対してパッシブ接続ポートを確立するように依頼します。これは 10,000 より高いポートになります。するとサーバーは、この特定のセッション用に高い数値のポートをバインドして、このポート番号をクライアントに中継します。クライアントは、データ接続のために新規にバインドされたポートを開きます。クライアントが作成するデータ要求それぞれ、別個のデータ接続となります。最近の FTP クライアントのほとんどは、 サーバーからデータを要求する場合、パッシブ接続を試みます。

注記

接続タイプを決定するのは、サーバーではなく **クライアント** です。つまり、効果的に FTP をクラスタ化するには、アクティブ接続とパッシブ接続の両方を処理するように LVS ルーターを設定する必要があることになります。

HTP のクライアントとサーバーの関係によって、Keepalived が認識しない多くのポートが開かれる可能性があります。

3.5.2. ロードバランサーのルーティングへの影響

IPVS パケット転送は、それをベースにしたクラスタのへの接続とそのクラスタからの接続のみを許可し、そのポート番号やファイアウォールマークを認識します。クラスタ外のクライアントが IPVS で処理するように設定されていないポートを開こうとした場合、接続は切断されます。同様に、実サーバーが IPVS が認識できないポート上でインターネット接続を開こうとした場合も、接続は切断されます。つまり、インターネット上の FTP クライアントからの**すべての**接続は、それらに割り当てられているファイアウォールマークと同じである**必要があり**、FTP サーバーからの全接続は、ネットワークパケットのフィルタリングルールを使用して正常にインターネットに転送される**必要がある**ことを意味します。

注記

パッシブ FTP 接続を有効にするには、**`ip_vs_ftp`** カーネルモジュールがロードされていることを確認します。これは、シェルプロンプトで管理ユーザーとして **`modprobe ip_vs_ftp`** コマンドを実行することで可能です。

3.5.3. ネットワークパケットフィルタルールの作成

FTP サービスの **iptables** ルールを割り当てる前に、マルチポートサービスおよび既存ネットワークパケットフィルタリングルールをチェックする技術に関して「ファイアウォールマークの割り当て」内の情報を再確認してください。

以下のルールは、同じファイアウォールマーク (21) をFTP トラフィックへ割り当てます。

3.5.3.1. アクティブ接続のルール

アクティブ接続のルールは、カーネルに FTP データポートであるポート 20 上にある **内部** のフローティング IP アドレスへ届く接続を受け付けて転送するように指示します。

以下の **iptables** コマンドにより、LVS ルーターは IPVS が認識していない実サーバーからの外向けの接続を受け付けることが可能になります。

```
/usr/sbin/iptables -t nat -A POSTROUTING -p tcp -s n.n.n.0/24 --sport 20 -j MASQUERADE
```

この **iptables** コマンドの *n.n.n* は、**keepalived.conf** ファイルの ***virtual_server*** セクションで定義され、NAT インターフェースの内部ネットワークインターフェースに割り当てられるフローティング IP の最初の 3 つの値に置き換える必要があります。

3.5.3.2. パッシブ接続のルール

パッシブ接続のルールでは、10,000 から 20,000 という広い範囲のポートにあるサービスのフローティング IP へのインターネットからの接続に適切なファイアウォールマークを割り当てます。

警告

パッシブ接続でのポート範囲を制限している場合、VSFTP サーバーを設定して一致するポート範囲を使用するように設定する必要があります。これは以下の行を **/etc/vsftpd.conf** に追加することで可能です。

```
pasv_min_port=10000

pasv_max_port=20000
```

実際の FTP サーバーアドレスを上書きする **pasv_address** の設定は使用しないでください。LVS により仮想 IP アドレスに更新されるためです。

他の FTP サーバーの設定については、個別のドキュメンテーションを参照してください。

この範囲はほとんどの状況では十分なものです。しかし、以下のコマンド内の **10000:20000** を **1024:65535** に変更することで、利用可能な非保護ポートすべてを含めることができます。

以下の **iptables** コマンドは、適切なポート上のフローティング IP が送信先であるトラフィックをファイアウォールマーク 21 に割り当てることと同様の効果があります。これは、IPVS で 認識されて適切に転送されます。

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 --dport 21 -j MARK --set-mark 21

/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 --dport 10000:20000 -j MARK --set-mark 21
```

この **iptables** コマンドの *n.n.n.n* は、**keepalived.conf** ファイルの ***virutal_server*** セクションで定義される FTP 仮想サーバーのフローティング IP アドレスに置き換える必要があります。

警告

上記のコマンドは即座に有効になりますが、システムを再起動すると保持されません。

最後に、適切なサービスが設定され、正しいランレベルでアクティベートされることを確認する必要があります。

3.6. ネットワークパケットフィルター設定の保存

ユーザーの状況に応じた適切なネットワークパケットフィルタを設定した後は、 その設定を保存して再起動後に復元するようにします。**iptables** には以下のコマンドを実行します。

```
/usr/sbin/service iptables save
```

これで **/etc/sysconfig/iptables** 内の設定が保存され、起動時に再度呼び出すことができます。

このファイルが書き込まれたら、**/usr/sbin/service** コマンドを使用して **iptables** を開始および停止したり、ステータススイッチを使用して **iptables** の状態を確認したりできます。**/usr/sbin/service** は適切なモジュールを自動的にロードします。

最後に、適切なサービスが設定され、正しいランレベルでアクティベートされることを確認する必要があります。

3.7. パケット転送および非ローカルバインディングの有効化

Keepalived サービスが適切にネットワークパケットを実サーバーへ転送するには、各ルーターノードのカーネルで IP 転送が有効になっている必要があります。root としてログインし、**/etc/sysctl.conf** にある行 **net.ipv4.ip_forward = 0** を以下に変更します。

```
net.ipv4.ip_forward = 1
```

システムを再起動すると変更が反映されます。

HAProxy で負荷分散を行うには、ローカルシステムのデバイスに割り当てられていない*非ローカル*の IP アドレスへバインドする機能が必要になります。この機能により、フェイルオーバーの発生時に稼働しているロードバランサーインスタンスがローカルでない IP アドレスへバインドできるようになります。

この機能を有効にするには、**/etc/sysctl.conf** にある行 **net.ipv4.ip_nonlocal_bind** を以下に変更します。

```
net.ipv4.ip_nonlocal_bind = 1
```

システムを再起動すると変更が反映されます。

IP 転送がオンになっているか確認するため root で次のコマンドを実行します。

```
/usr/sbin/sysctl net.ipv4.ip_forward
```

非ローカルバインディングが有効になっていることを確認するには、root として以下のコマンドを実行します。

```
/usr/sbin/sysctl net.ipv4.ip_nonlocal_bind
```

設定が有効になっていれば、上記の各コマンドは **1** を返します。

3.8. 実サーバーでサービスを設定する

実サーバーが Red Hat Enterprise Linux システムの場合は、起動時に適切なサーバーデーモンがアクティブになるよう設定しておきます。Web サービスの **httpd** や FTP サービスまた Telnet サービスの**xinetd** などのデーモンがこれに該当します。

また、実サーバーに遠隔からもアクセスできると便利なため **sshd** デーモンもインストールして実行しておいてください。

第4章 Keepalived を用いたロードバランサーの初期設定

Load Balancer パッケージをインストールしたら、基本手順に従って Keepalived で使用する LVS ルーターと実サーバーを設定する必要があります。本章では、最初の手順を説明します。

4.1. 基本的な Keepalived の設定

この基本的な例では 2 システムがロードバランサーとして構成されています。 LB1 (アクティブ) と LB2 (バックアップ) は 191.168.1.20 から 192.168.1.24 の実際の IP アドレス番号で稼働している 4 web サーバーのプールに要求をルーティングして仮想 IP アドレス 192.168.1.11 を共有します。各ロードバランサーには**eth0** と **eth1** の 2 種類のインターフェースがあります。ひとつは外部のインターネットトラフィックを処理し、もう一方は実サーバーへの要求をルーティングします。使用される負荷分散アルゴリズムはラウンドロビンなります。ルーティングメソッドは Network Address Translation です。

4.1.1. `keeaplived.conf` ファイルの作成

Keepalived は **`keepalived.conf`** ファイルで設定します。「基本的な Keepalived の設定」で示す例のようなロードバランサーのトポロジーを構成する場合、テキストエディターを使って **`keepalived.conf`** を開きます。以下に例を示します。

```
vi /etc/keepalived/keepalived.conf
```

「基本的な Keepalived の設定」で説明されているような構成で負荷分散されている基本的なシステムには以下のコードセクションで示すような **`keepalived.conf`** ファイルがあります。

4.1.1.1. グローバル定義

`keepalived.conf` ファイルのグローバル定義セクションを使用すると管理者はロードバランサーに変化が生じた場合の通知詳細を指定することができます。グローバル定義はオプションですので Keepalived の設定には必要ありません。

```
global_defs {

   notification_email {
       admin@example.com
   }
   notification_email_from noreply@example.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 60
}
```

`notification_email` はロードバランサーの管理者、***`notification_email_from`*** はロードバランサーの状態の変化を送信するアドレスです。SMTP 固有の設定で通知の送信元となるメールサーバーを指定します。

4.1.1.2. VRRP インスタンス

```
vrrp_sync_group VG1 {
   group {
      RH_EXT
      RH_INT
   }

vrrp_instance RH_EXT {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
 auth_pass password123
    }
    virtual_ipaddress {
    10.0.0.1
    }
}

vrrp_instance RH_INT {
   state MASTER
   interface eth1
   virtual_router_id 2
   priority 100
   advert_int 1
   authentication {
       auth_type PASS
       auth_pass password123
   }
   virtual_ipaddress {
       192.168.1.1
   }
}
```

この例では ***vrrp_sync_group*** スタンザでは状態が変化 (フェールオーバーなど) しても VRRP グループは変化しないことが定義されています。インターネットと通信を行う外部インターフェースに対するインスタンス (RH_EXT) の他、内部インターフェースに対するインスタンス (RH_INT) も定義されています。

vrrp_instance RH1 行では仮想 IP インスタンスを構成する VRRP サービスデーモンの仮想インターフェース設定について指定しています。**state MASTER** でじゃアクティブサーバーについて指定しています。設定内のバックアップサーバーにも上記と同等の設定がありますが **state BACKUP** で設定されています。

interface パラメーターを使ってこの仮想 IP インスタンスに物理的なインターフェース名を割り当てます。**virtual_router_id** はそのインスタンスの数字に表記の識別子です。**priority 100** では割り当てられたインターフェースのフェールオーバーの際の引き継ぎ順序を指定します。小さい数ほど優先順位は高くなります。

authentication のブロックではフェールオーバー同期の際に認証サーバーに使用する認証タイプ (**auth_type**) とパスワード (**auth_pass**) を指定します。**PASS** ではパスワード認証を指定します。Keepalived では **AH** や接続の整合性を目的とした認証ヘッダーにも対応しています。

最後に、**virtual_ipaddress** オプションではインターフェースの仮想 IP アドレスを指定します。

4.1.1.3. 仮想サーバーの定義

```
virtual_server 192.168.1.11 80 {
    delay_loop 6
    lb_algo rr
    lb_kind NAT

    real_server 192.168.1.20 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.21 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.22 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.23 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }

}
```

virtual_server はまず IP アドレスで設定されます。次に**delay_loop** で健全性のチェックの間隔を秒単位で設定します。**lb_algo** オプションでは可用性に使用するアルゴリズムの種類を指定します (上記の例ではラウンドロビンの **rr** を使用)。**lb_kind** オプションではルーティングメソッドを指定します。上記の例では Network Address Translation (**nat**) を使用しています。

仮想サーバーの詳細を設定したら、**real_server** オプションを設定します。ここでもまず IP アドレスを最初に指定します。**TCP_CHECK** スタンザでは TCP を使って 実サーバーの可用性をチェックします。**connect_timeout** ではタイムアウトが発生するまでの時間を秒単位で設定します。

4.2. Keepalived ダイレクトルーティング設定

Keepalived のダイレクトルーティング設定は NAT での設定と同じです。次の例ではポート 80 で HTTP を実行している実サーバーグループに負荷分散を提供するよう Keepalived を構成しています。ダイレクトルーティングを設定する場合は、***lb_kind*** パラメーターを **DR** に変更します。他の設定オプションについては 「基本的な Keepalived の設定」 で説明しています。

```
global_defs {
   notification_email {
     admin@example.com
   }
   notification_email_from noreply_admin@example.com
   smtp_server 127.0.0.1
```

```
   smtp_connect_timeout 60
}

vrrp_instance RH_1 {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 1
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass password123
    }
    virtual_ipaddress {
        172.31.0.1
    }
}

virtual_server 172.31.0.1 80
    delay_loop 10
    lb_algo rr
    lb_kind DR
    persistence_timeout 9600
    protocol TCP

    real_server 192.168.0.1 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.2 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.3 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
}
```

4.3. サービスの開始

次のコマンドを実行してサービスを開始します。

```
systemctl start keepalived.service
```

再起動後も Keepalived サービスを維持する場合は次のコマンドを実行します。

```
systemctl enable keepalived.service
```

第5章 HAProxy の設定

本章では、可用性の高い環境に HAProxy を導入する際の一般的な設定オプションに焦点を置きながら基本的なセットアップの構成について説明していきます。

HAProxy は、負荷分散を行うための独自のスケジューリングアルゴリズムのセットを持っています。これらのアルゴリズムは「HAProxy のスケジューリングアルゴリズム」に記載されています。

HAProxy の設定を行う場合は **`/etc/haproxy/haproxy.cfg`** ファイルを編集します。

HAProxy を使ったロードバランサーの設定は 5 セクションで構成されます。

- 「***`global`*** のセッティング」
- プロキシのセクション、4 サブセクションで構成されます。
  - 「***`default`*** のセッティング」のセッティング
  - 「***`frontend`*** のセッティング」のセッティング
  - 「***`backend`*** のセッティング」のセッティング

5.1. HAProxy のスケジューリングアルゴリズム

負荷分散用の HAProxy スケジューリングアルゴリズムは、**`/etc/haproxy/haproxy.cfg`** 設定ファイルの ***`backend`*** セクションにある ***`balance`*** パラメーターで編集できます。HAProxy は複数のバックエンドを持つ設定をサポートし、各バックエンドにスケジューリングアルゴリズムを設定できます。

ラウンドロビン (*`roundrobin`*)

要求を順番に実サーバーのプールで分配します。このアルゴリズムを使用すると、容量や負荷に関係なく実サーバーはすべて同等に扱われます。このスケジューリングモデルはラウンドロビン DNS と似ていますが、ホストベースではなくネットワークベースであるため、粒度がより細かくなります。さらに、ロードバランサーのラウンドロビンスケジューリングでは、キャッシュされた DNS クエリーが原因で負荷が不均等になることはありません。しかし、HAProxy ではこのスケジューラーを使用するとサーバーの加重値をオンザフライで設定できるため、バックエンドごとにアクティブなサーバーの数が 4095 に制限されます。

静的ラウンドロビン (*`static-rr`*)

ラウンドロビンと同様に、要求を順番に実サーバーのプールで分配しますが、サーバーの加重値を動的に設定できません。しかし、サーバーの加重値が静的であるため、アクティブなサーバーの数はバックエンドごとに制限されません。

最小接続 (*`leastconn`*)

アクティブな接続の数が少ない実サーバーにより多くの要求を分配します。セッションや接続の長さが異なる動的な環境に適したスケジューラーです。また、このスケジューラーを使用すると管理者は加重値をオンザフライで調整できるため、異なる容量のサーバーが複数ある環境にも適しています。

ソース (*`source`*)

要求するソース IP アドレスをハッシュ化し、稼働するすべてのサーバーの加重値で割って要求を取得するサーバーを決定し、要求をサーバーに分配します。すべてのサーバーが稼働中である場合、ソース IP 要求は一貫して同じ実サーバーによって処理されます。稼働中のサーバーの数や加重値に変更があった場合、ハッシュまたは加重値の結果が変わるため、セッションが別のサーバーに移動される可能性があります。

URI (*uri*)

URI 全体 (または URI の設定可能な部分) をハッシュ化し、稼働するすべてのサーバーの加重値で割ってサーバーへ要求を分配します。アクティブなサーバーがすべて稼働中である場合、ソース IP 要求は一貫して同じ実サーバーによって処理されます。このスケジューラーを追加設定するには、URI の最初にある文字列の長さでハッシュ化の結果を算出し、URI のディレクトリーの深さ (URI のスラッシュによる) でハッシュ化の結果を算出します。

URL パラメーター (*url_param*)

ソース URL 要求の特定のパラメーター文字列を検索し、ハッシュの計算を稼働中のすべてのサーバーの加重値で割って、サーバーへの要求を分配します。URL にパラメーターがない場合、スケジューラーはラウンドロビンスケジューリングをデフォルトとして使用します。POST パラメーターを基に修飾子が使用されることがあります。また、特定のパラメーターを待ってからハッシュ化の結果を算出するために管理者が割り当てる最大オクテット数を基に待機制限が使用されることもあります。

ヘッダー名 (*hdr*)

各リソース HTTP 要求の特定のヘッダー名をチェックし、ハッシュの計算を稼働中のサーバーすべての加重値で割って、サーバーへ要求を分配します。ヘッダーがない場合は、スケジューラーはデフォルトでラウンドロビンスケジューリングを使用します。

RDP クッキー (*rdp-cookie*)

各 TCP 要求に対して RDP クッキーを検索し、ハッシュの計算を稼働中のサーバーすべての加重値で割って、サーバーへ要求を分配します。ヘッダーがない場合、スケジューラーはデフォルトでラウンドロビンスケジューリングを使用します。この方法はセッションの整合性を維持するため、永続化に適しています。

5.2. *global* のセッティング

global セッティングでは HAProxy を実行するサーバーすべてに適用するパラメーターを設定します。一般的な ***global*** セッティングを以下に示します。

```
global
    log 127.0.0.1 local2
    maxconn 4000
    user haproxy
    group haproxy
    daemon
```

上記の設定では、サービスはすべてのエントリーをローカルの **syslog** サーバーに ***log*** するよう設定します。デフォルトでは **/var/log/syslog** またはユーザーが指定する場所になる場合があります。

maxconn パラメーターではサービスの並列接続の最大数を指定しています。デフォルトの最大数は 2000 です。

user と ***group*** のパラメーターでは **haproxy** プロセスが属するユーザー名とグループ名を指定しています。

daemon パラメーターでは **haproxy** がバックグラウンドプロセスとして実行されるよう指定しています。

5.3. *default* のセッティング

default のセッティングでは設定内の全プロキシサブセクションに適用されるパラメーターを設定します (frontend、backend、listen)。一般的な **default** セクションを以下に示します。

注記

proxy サブセクション (**frontend**、**backend**、**listen**) 内で設定するパラメーターはすべて **default** 内のパラメーター値より優先されます。

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
```

mode では HAProxy インスタンスのプロトコルを指定しています。**http** モードを使用するとソースの要求は HTTP に応じて実サーバーに接続されるため、web サーバーの負荷分散を行う場合に理想的です。他のアプリケーションの場合は **tcp** モードを使用してください。

log ではログのエントリーが書き込まれる**syslog** ファシリティとログアドレスを指定しています。**global** の値は **global** セクションの **log** パラメーターに指定されている HAProxy インスタンスを指しています。

option httplog は HTTP 要求、セッション状態、接続数、ソースアドレス、他の値内での接続タイマーなど、HTTP セッションの各種の値のログ記録を有効にしています。

option dontlognull は null 接続のログ記録を無効にしています。つまり、HAProxy はデータが転送されなかった場合の接続はログ記録しないということです。この設定はインターネット経由の web アプリケーションを使用するなどの環境、つまり脆弱性を狙ってポートスキャンを開くなど悪意のある動きを nu 接続で検出できるような場合には推奨できません。

retries では最初の接続試行に失敗した後、実サーバーに再接続の要求を何回行わせるかを指定しています。

timeout のそれぞれの値は特定の要求、接続、応答などに反応しない場合に待機させる秒数またはミリ秒数を指定します。**http-request 10s** の値はクライアントからの HTTP 要求の完了に対して待機する秒数を 10 秒に指定しています。**queue 10m** の値は接続をドロップしてクライアントに 503 または「Service Unavailable」のエラーを表示するまでの待機時間を指定しています。**connect 10s** の値はサーバーへの適切な接続を待機する秒数を指定しています。**client 1m** の値はクライアントが休止状態 (データの受信も送信も行わない) を維持できる時間を指定しています。**server 10m** の値はタイムアウトが発生するまでにサーバーに与えられるデータ送受信の時間 (ミリ秒単位) を指定しています。

5.4. *frontend* のセッティング

frontend のセッティングではクライアントの接続要求を受けるサーバーのリスニングソケットを設定します。一般的な **frontend** の HAProxy 設定を以下に示します。

```
frontend  main
  bind 192.168.0.10:80
```

main という名前の **frontend** には **bind** を使って IP アドレス 192.168.0.10 とリスニングポート 80 が設定されています。接続後はすべてのセッション接続を **app** backend に接続するよう **use backend** で指定されています。

5.5. backend のセッティング

backend のセッティングでは実サーバーの IP アドレスおよびロードバランサーのスケジューリングアルゴリズムを指定します。一般的な **backend** セクションを以下に示します。

```
backend app
   balance     roundrobin
   server  app1 192.168.1.1:80 check
   server  app2 192.168.1.2:80 check
   server  app3 192.168.1.3:80 check inter 2s rise 4 fall 3
   server  app4 192.168.1.4:80 backup
```

バックエンドサーバーの名前は **app** です。**balance** は使用されるロードバランサースケジューリングアルゴリズムを指定します。この例ではラウンドロビン (**roundrobin**) が指定されていますが、HAProxy がサポートする他のスケジューラーを指定することもできます。HAProxy でのスケジューラーの設定については、「HAProxy のスケジューリングアルゴリズム」を参照してください。

server の行ではバックエンドで使用可能なサーバーを指定しています。**app1** から **app4** まではそれぞれの実サーバーに内部的に割り当てられる名前です。ログファイルでは名前を使ってサーバーのメッセージが区別されます。アドレスは割り当てられる IP アドレスです。IP アドレスのコロンの後ろに付く値はサーバーで接続が発生するポートです。**check** オプションではサーバーが利用可能でデータの送信およびセッション要求の受け取りができるか定期的なヘルスチェックを行うためのフラグ付けをしています。またサーバー app3 にはヘルスチェックの間隔を 2 秒 (**inter 2s**)、健全とみなすために要されるチェックの回数 (**rise 4**)、健全性に問題ありとみなすために要される連続チェック失敗の回数 (**fall 3**) なども設定されています。

5.6. haproxy の起動

次のコマンドを実行して HAProxy を起動します。

```
systemctl start haproxy.service
```

再起動後も HAProxy サービスを継続して起動させる場合は次のコマンドを実行します。

```
systemctl enable haproxy.service
```

付録A 改訂履歴

改訂 0.2-7 **Mon Aug 24 2015** **Junko Ito**
翻訳ファイルを XML ソースバージョン 0.2-6 と同期

改訂 0.2-6 **Mon Feb 16 2015** **Steven Levine**
7.1 GA リリース向けのバージョン

改訂 0.2-5 **Thu Dec 11 2014** **Steven Levine**
7.1 ベータリリース向けのバージョン

改訂 0.1-15 **Fri Dec 05 2014** **Steven Levine**
Red Hat Enterprise Linux のスプラッシュページで新しいソート順序を実装するため更新。

改訂 0.1-13 **Thu Dec 04 2014** **Steven Levine**
7.1 ベータリリース向けのバージョン

改訂 0.1-12 **Tue Jun 03 2014** **John Ha**
7.0 GA リリース向けバージョン

改訂 0.1-6 **Mon Jun 13 2013** **John Ha**
Red Hat Enterprise Linux 7 のベータ向けビルド

改訂 0.1-2 **Mon Jun 13 2013** **John Ha**
初期反応のアルファ版ドラフト

改訂 0.1-1 **Wed Jan 16 2013** **John Ha**
Red Hat Enterprise Linux 6 バージョンのドキュメントからブランチ作成

索引

シンボル