Linux Virtual Server(ＬＶＳ)を利用したＰＣクラスタにおけるノード管理

カンファレンス・トラック種別
クラスタリング・テクノロジ

NTTアドバンステクノロジ 株）
樋口一茂 藤田昭人

■ 概要

Linux Virtual Server 以降LVS)は市販の負荷分散装置と同等の機能を提供する Linux Kernel コードである。市販製品と比べても性能的に遜色がなく、低コストかつカスタマイズ可能なLVSは、各種サービス・サーバーの負荷分散型の PCクラスタを構成する場合に大変便利である。

現在開発中のＷｅb専用サーバーシステムである『LaMoN」もLVSを利用するPCクラスタであるが、本稿では同システムをLVSの応用事例として紹介する。

■ クラスタについて

本稿は PCクラスタについて論ずるわけであるが、その前にクラスタという用語について前置きをしておく必要がある。

コンピュータの世界においてクラスタという用語は誤解を招き易い。言葉の本来の意味からすると、複数のコンピュータを集めればクラスタ・システムと呼ぶことができる。が、一体何をするシステムなのか良くわからないケースはままある。

一般的に PCクラスタという表現は次の 3種類のシステムのいずれかを意味する。

- 並列演算クラスタ
  主に科学技術計算で利用される並列プログラミング・アプリケーションを使用するためのクラスタ。スーパーコンピュータと同等の処理能力を低コストで実現する。Beowulf クラスタは典型的な実例。

- 高可用性クラスタ
  ミッション・クリティカルなアプリケーションを実行するためのクラスタ。クラスタは冗長化構成を持ち障害発生時には切替える。フォールト・トレラント・コンピュータの適用分野でシステムをより低コストに実現できる。

- 負荷分散クラスタ
  単一のアプリケーションを負荷分散して実行するためのクラスタ。ネットワーク・サービス・アプリケーションの実行に適している。

本稿は論ずるクラスタとは負荷分散クラスタを意味する。留意していただきたい。

■ Linux Virtual Server (LVS)

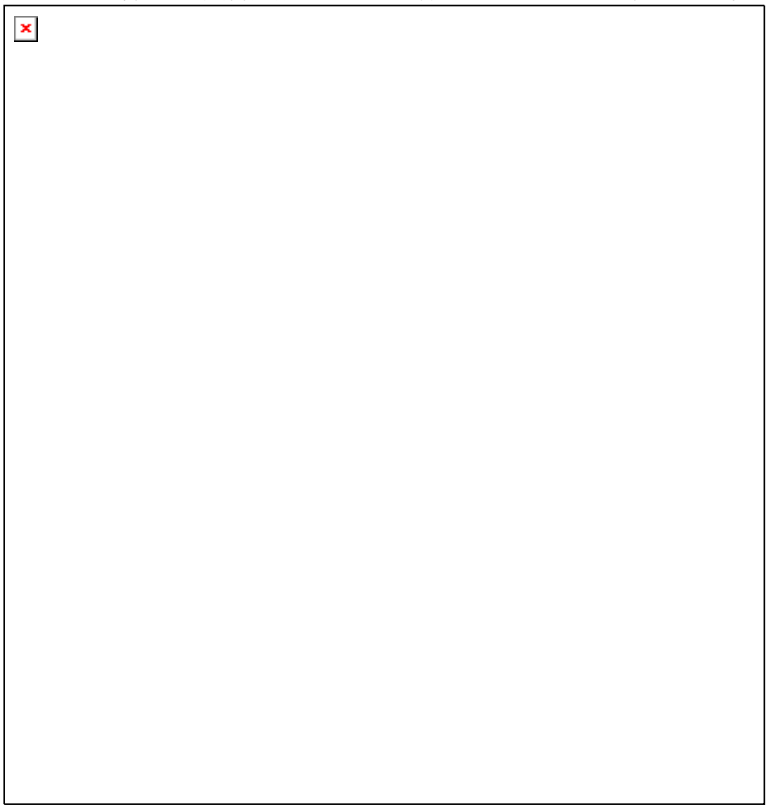
Linux Virtual Server(以降 LVS)は Linux カーネルの TCP／ IPプロトコル ・スタックを拡張し、I Pベースのロード・バランシング機能を実現したソフトウェアであり、次の特徴を備えている。

- 3種類のロード・バランシング技術 (LVS/NAT,LVS/TUN,LVS/DR)
- 4種類のスケジューリング ・アルゴリズム
- ほとんどの TCP/UDP サービスをサポート
- クライアント、サーバーを改変する必要がない

LVSは負荷分散クラスタ(前述)の基盤技術として既に広く利用されており、主要なサーバー系 Linux Distribution にも標準でバンドルされている。

＊ LVSを利用したクラスタ

以下の図はLVSを利用した負荷分散クラスタの 3層アーキテクチュアを示している。



クラスタは次の 3種類のノードから構成される。

- **Load Balancer** はサービスのフロント･エンドである。クライアントからのネットワーク接続を実際にサービス処理を実行するサーバーに振り向ける。
- **Server Node** はサーバーのクラスタから構成される。個々のサーバーが実際のサービス処理を実行する。
- **Backend Storage** はサーバーに対し共有ストレージを提供する。

このアーキテクチュアの利点はシステムのスケーラビリティとハイ･アベイラビリティを同時に達成できることにある。
この構成ではシステム全体の処理能力はサーバーの数にリニアに追従し、LVSの機能によりサーバーはシステムの稼働中に随時追加をすることが可能であるから、サーバーの負荷状態に合わせてシステム全体の処理能力を動的に追従させることができる。
また、個々のサーバーが何らかの障害により動作を停止した場合、LVSの機能によりシステムの稼働中に随時サーバーを削除することができるため、サービスを停止することなく障害が発生したサーバーを切り離すことができる。
このようにクラスタの自動再構成ができることが、このアーキテクチャの利点である。
一方、このアーキテクチュアの欠点が **Load Balancer** 自体にあることも明らかで、特に **Load Balancer** に障害が発生した場合に、システム全体が全く機能しなくなることは致命的な問題である。このアーキテクチュアを採用した実際のクラスタでは **Load Balancer** を二重化することにより、この問題を回避している例が多い。

＊ LVSを利用した負荷分散クラスタの機能とその実現

前節で述べたように負荷分散クラスタのアーキテクチュアは各種のメリットを提供するわけであるが、その具体的な機能は概ね次の 3項目にまとめられる。

- 負荷分散
- クラスタの高信頼化
- クラスタの自動再構成

LVSを利用した負荷分散クラスタでのこの 3機能の実現性は以下の通りである。

A. 負荷分散
   負荷分散機能は負荷分散クラスタの基本機能であり、原則として **Load Balancer** にLVSを搭載することにより実現できる。LVSの導入に関連して技術的に考慮すべき事項が多数あるが、これは Linux Virtual Server Project が公開している LVS HOWTO が詳しい。

B. クラスタの高信頼化
一般的にクラスタの高信頼化は構成ノードの冗長化によるが、負荷分散クラスタの場合、**Server Node** の冗長化は負荷分散機能によって達成されるため、クラスタとしての制御を司る **Load Balancer** の冗長化がクラスタの高信頼化の技術的な焦点となる。
Linux Virtual Server Project から提供されるLVSのコンポーネントには、**Load Balancer** の冗長化のための手段は含まれない。したがって冗長化機能を実現するためには、他のコンポーネントを流用する必要がある。Linux Virtual Server Project のホームページでは Linux-HA heartbeat package など高可用性クラスタ向けに開発された技術を利用することが紹介されている。

C. クラスタの自動再構成
LVSではシステムの稼働中にクラスタの再構成を行なう手段は提供されるが、これを自動化する手段は提供されない。Linux Virtual Server Project のホームページでは Mon などの自動化に必要な **Server Node** のモニタリングのためのソフトウェア・コンポーネントなどを紹介している。

LVSは負荷分散クラスタを実現する基盤技術ではあるが、Linux Virtual Server Projetct が提供するオリジナルのLVSコンポーネントを導入しただけで実用的な負荷分散クラスタが構築できるわけではない。
主要なサーバー系 Linux Distributor は上記の OpenSource ベースのコンポーネントや独自に開発したコンポーネントを統合し、各種クラスタ製品の商品化を行なっている。負荷分散クラスタの商品事例としては RedHat High Availability Server や TurboLinux Cluster Server などが上げられる。

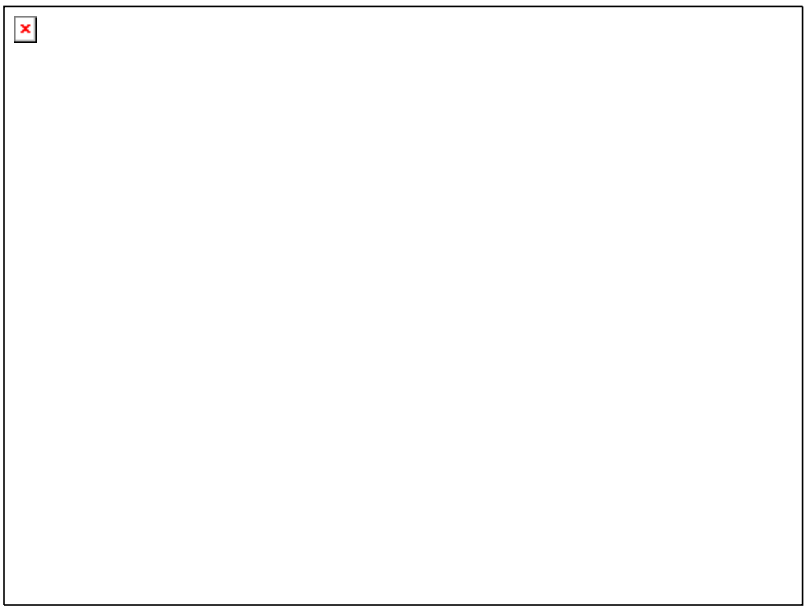
■ LVSを利用した負荷分散クラスタの事例 － RedHat Piranha

Piranha は RedHat High Availability Server の名前で商品化されているLVSを利用した RedHat のクラスタ製品である。

Piranha 本体は RedHat が独自開発したソフトウェア・コンポーネントのパッケージであり、単独で *FOS* 呼ばれる高可用性クラスタが構成でき、かつLVSと併用することにより負荷分散クラスタも構成できるようになっている。

＊ Piranha による負荷分散クラスタの構成例

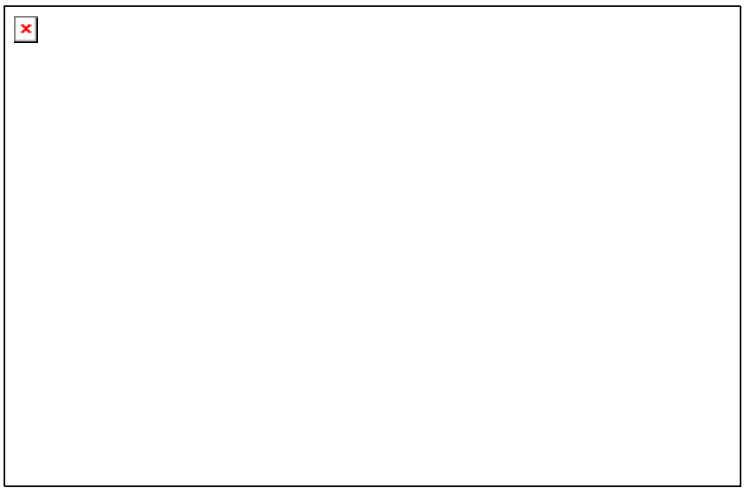
Piranha による基本的なLVSクラスタ構成例を次に示す。



この構成は前章で述べた負荷分散クラスタ3層アーキテクチュアを踏襲している。

＊ Piranha のソフトウェア ･コンポーネント

Piranha のソフトウェア ･コンポーネントのブロック ･ダイアグラムを次に示す。



Piranha では、オリジナルLVSコンポーネントに含まれる **ipvsadm** の他に次のようなソフトウェア・コンポーネントが稼働する。

- **pulse**
  Load Balancer の冗長化を実現するフェール・オーバー機能を司るデーモン。他のデーモンを起動する制御プロセスとしても機能する。

- **lvs**
  LVSの *virtual service* の維持／管理を司るデーモン。コンフィグレーション・ファイルを読み込み、その記述にしたがって *virtual service* の追加／削除を行なうとともに、**nanny** を起動する。

- **nanny**
  LVSの *real server* の維持／管理を司るデーモン。各サーバー・ノードの動作状態をモニタリングし、その結果に応じて *real server* の追加／削除を行なう。

以上のデーモンはLVSの動作に必要な構成情報を定義した **/etc/lvs.cf** を直接あるいは間接的に参照して所定の動作を行なう。
この **/etc/lvs.cf** を維持／管理するためのツールとしてPHP3を使ったGUI **Piranha Web Interface** も提供される。

＊ Piranha の負荷分散クラスタが提供する機能

負荷分散クラスタとして **Piranha** が提供している機能とその範囲は以下の通りである。

A. 負荷分散
   Piranha はLVSを利用しているので、LVSの持つ負荷分散機能は全てサポートしている。

B. クラスタの高信頼化
   pulse によりLoad Balancer の二重化を実現している。

   pulse はシンプルなフェール・オーバー機能は提供しており、
   プライマリとバックアップの2台のLVSルーター(Load Balancer)の間でハートビートを行なう。
   ハートビート間隔はデフォルトで6秒、デッドライン(デフォルトで18秒)内にレスポンスがなければフェール・オーバーが発生する。
   フェール・オーバー発生時の activate/deactivate 処理は **pulse** のプログラム中にハード・コードされているため、処理内容を変更したい場合には **pulse** のソースコードを修正する必要がある。

C. クラスタの自動再構成
nanny が提供する Server Node の稼働状態モニタリングを契機とする縮退型の自動再構成のみをサポートする。

nanny はターゲットとなる Server Node に対し、*ping* の実行、所定のポートに対する文字列の送受信の検査を行ない、ホスト／サービスが稼働しているかどうかの確認を行なう。
ホスト／サービスの正常動作が確認できた場合には、LVSの *real server* エントリを追加する。また、ホスト／サービスの正常動作が確認できなかった場合には、LVSの *real server* エントリを削除する。更に Server Node のロード･アベレージを取得し、*real server* エントリの重み付けを調整する機能も有する。

以上のような機能を有する **Piranha** の負荷分散クラスタはLVSの持つ基本機能に耐障害機能を強化した作りになっており、冗長構成によるコスト･アップが課題となる一般的な高可用性クラスタと同等の効果をより経済的に実現することを目指しているように思われる。

＊ Piranha の負荷分散クラスタの問題点

Piranha は負荷分散クラスタを構成するために必要な機能をコンパクトにまとめたシンプルな実装であり、比較的簡便な作業でクラスタの効果を確認できるパッケージである。
しかし **Piranha** を利用してノード数が数十台以上の大規模クラスタを構築するには次のような問題が懸念される。

- **Piranha** はクラスタを構成する Server Node の稼働状態のモニタリングは行なうが、その状態の表示や、停止／再起動等の Server Node を制御する機能は提供しない。したがって Server Node の制御は実機を直接操作するか、集中管理するための仕掛けを導入者側で用意する必要がある。
大規模クラスタでは管理すべき Server Node の台数が大きくなるため、この問題はクラスタ運用に大きな影響を与える。オペレータ向けのGUIの統合も含め、標準的なノード管理機構をサポートするべきである。

- **Piranha** のクラスタ自動再構成は、Server Node の障害発生時にクラスタを縮退させる機能のみをサポートしている。
しかし大規模クラスタでは予備の Server Node が用意されていることが多く、これを使用して障害発生時に予備へ切替える、あるいはクラスタ全体での負荷が増大した際に予備を追加してクラスタを伸長させるような自動再構成の機能が期待される。

以上のようなクラスタの設計面での問題に加え **Piranha** には以下のような実装面での問題もある。

- **Piranha** のデーモンおよびWebインターフェースは全て Load Balancer で動作する。負荷分散クラスタでは Load Balancer の処理能力がクラスタ全体の性能に大きく影響することは自明であるが、管理用のデーモンの負荷のために本来の役割である負荷分散の処理能力が低下するのは望ましくない。
  特に **nanny** は Server Node の数だけ子プロセスを生成する作りになっているため、大規模クラスタでは無数の **nanny** プロセスが動作することになる。より負荷の少ない方式を検討するべきである。

- また **nanny** は Server Node の稼働状態を把握するため、1回につき3種類のプロトコルのよるリクエストを発生させ、Load Balancer - Server Node 間の通信トラッフィックを増大させる。さらに効率の良い稼働状態のモニタリング方法を検討するべきである。

Piranha は構成ノードが10台以下の比較的小規模なクラスタへの適用を想定して開発された製品だと推測される。したがって構成ノードが数十台以上の大規模クラスタへ適用するには力不足の感があり、上記のような機能強化が必要である。

■ LaMoNのノード管理機能

＊ LaMoNついて

LaMoNは各種通信事業者のデータ・センターが運営する中規模～大規模のWebサービスのための専用サーバー装置として企画されたWeb専用サーバーシステムである。
LaMoNは過去のシステム構築／運用業務の経験を踏まえて提案された製品企画であり、次のような特徴を持つ。

- Webトラフィックの急激な変動に対応できる柔軟な拡張性の確保
- 処理プロッセッサの冗長化による信頼性の向上
- 拡張時のコストの低減

このような特徴を実現するためそのシステム・ハードウェアとしてPCを高速LANで接続する形態、すなわちPCクラスタを念頭においたシステム設計を進めている。具体的には19インチ・ラック1台に **Load Balancer** / **Server Node** / **Backend Storage** 用の PCを搭載し、各 PC 間はL2スイッチを介して Gigabit Ethernet あるいは Fast Ethernet で相互接続行う。
このシステム・ハードウェアの性能目標は1ラックあたりの同時HTTPセッション接続数を最大で2000セッションと設定しており、これを実現するために28台程度の PCをラック1台に収容する。クラスタ全体での要求性能がこれを上回る場合は、複数のラックを並列にネットワーク接続する構成を取る。現在市販されているL2スイッチの接続能力を考慮すると、LaMoNはノード数が最大で200～300台規模の大規模 PCクラスタとして構成する。

＊ LaMoNのノード管理機能

構成ノード数が200～300台規模の大規模PCクラスタともなると、例えば単純な各ノードの起動／停止操作であっても、**rsh/ssh** 等を使って手作業で行なうのは大変繁雑で苦痛を伴う作業となる。1台の保守／管理用のコンソールからクラスタの全てのノードに対して必要な操作ができることが望ましい。LaMoNのノード管理機能はこのような発想から「保守用コンソールからの集中管理」を実現することを目的として検討された。

開発の基本方針は以下の通りである。

- ノード管理機能はエージェント－マネージャ型のアーキテクチュアに基づく一般的なLinuxソフトウェアとして実現する。
- エージェントはクラスタを構成する全ノードにインストールものとし、マネージャがインストールされるノードは **保守用コンソール** として機能する。
- **保守用コンソール** のユーザー・インターフェースはCUIベースとし、GUIはマネージャ・コマンドを利用したWebインターフェースにより実現する。
- 提供する機能は／自動再構成機能との連動の3つとする。

以降、**保守用コンソール** でサポートされる機能について説明する。

＊ ノードの監視

各ノードのエージェントは次のような状態情報を取りまとめて、定時的にマネージャへ送る。

- ノード・ホストの動作状態(起動／停止／稼働中)
- ノード・ホストのロード・アベレージ
- 稼働中のサーバ・アプリケーションの接続数／負荷量

マネージャは上記のノード状態情報を都度ログ出力するとともに、次の2種類の集計方法で **保守用コンソール** に定時的に表示する。

- 全ノードの状態情報要約一覧の表示
- 指定ノード毎の詳細状態情報の表示

またノード制御機能(後述)を使ってノードに何らかの操作を行なった場合には、エージェントはその都度、操作内容の要約をマネージャに送る。

＊ ノードの制御

保守用コンソール からはマネージャーエージェントを介して、個々のノードに対して次のような操作ができる。

- ノード・ホストの起動／停止
- ノードで稼動するサーバー・アプリケーションの起動／停止
- ノードの起動カーネルの切替え
- ノードでのパッケージ管理

「ノードでのパッケージ管理」は指定したノードでパッケージの **install/update/remove** 等の操作を行なうもので、**install/update** の際は各種パッケージを収容したファイル・サーバーからダウンロードするものとする。

＊ ノード管理機能とクラスタ自動再構成機能との連動

上記のようなノード管理機能を実現することにより、**保守用コンソール** にはクラスタを構成するノードの状態情報が蓄積されるが、この情報を使ってクラスタの自動再構成をより細かく制御することが可能になる。

LaMoNでのクラスタ自動再構成機能では予め設定したノード数を上限として、*virtual service* ごとに使用可能なノードを自動的に割り当てる機能を提供する。その動作の概要を順次説明する。

- まずLVSが稼働する **Load Balancer** は **保守用コンソール** から直接制御され、各 *virtual service* に対して使用する **Server Node** 数が予め設定されているものとする。
- **保守用コンソール** は各ノードから起動の通知を受けるごと、随時 *virtual service* への割当を行ない、**Load Balancer** に対しノード追加を指示する。
- 全ての *virtual service* へ所定の割当が終った後に起動したノードは、バックアップ・ノードとして管理する。
- **保守用コンソール** は *virtual service* に割り当てられているノードの稼働状態を監視し、定時的な稼働状態通知がなくなった場合には障害が発生したと判断する。
- 障害が発生した場合、あるいは明示的にノードの停止／再起動が指示された場合は **Load Balancer** に対しノード削除を指示する。

- 前項のノード削除が完了した後、**保守用コンソール** は関連する *virtual service* について不足数分をバックアップ・ノードからノード追加するように **Load Balancer** に対し指示をする。

以上のような動作により、各 *virtual service* ごとの **Server Node** の割当数が一定になるようにクラスタの構成は維持される。

構成ノード数が 200～300台規模の大規模クラスタではこの機能はサービス品質の維持とノードの使用効率を向上の両面で大変有用であり、保守オペレータはノードの稼働状態を意識することなく起動／停止ができるため保守作業時のトラブル軽減にも寄与すると思われる。

■ 結論

本稿の結論は以下の通りである。

- Linux Virutal Server(LVS)は負荷分散クラスタの基盤となる、ロード・バランシング機能を実現するソフトウェアである。
- LVSの 3層アーキテクチュアの利点はスケーラビリティとハイ・アベイラビリティを同時に達成できることである。
- LVSの応用事例である RedHat Piranha は LVS＋独自開発ソフトウェア・コンポーネントで負荷分散／高信頼化／自動再構成の機能を実現している。しかし大規模クラスタへの適応には問題もある。
- LaMoNもLVSを利用した負荷分散クラスタであるが、独自のノード管理機構を開発することにより、大規模クラスタでのより実用的な負荷分散／高信頼化／自動再構成の実現を目指している。

■ 著者紹介

樋口一茂
NTTアドバンステクノロジ(株) 主査

1994年 4月 日本電信電話(株)入社
光ネットワークシステム研究所(現未来ねっと研究所)に所属。
通信プロトコルの形式的仕様記述手法に関する研究に従事。

1998年 3月より現職
通信事業者向け運用システムの開発／Ｓ業務などに従事。

藤田昭人
NTTアドバンステクノロジ(株) 担当課長

1990年 2月(株)オムロン入社。
システム総合研究所(当時)に所属。
1990年 6月～1993年 10月米カーネギーメロン大学に
駐在。Mach グループに所属。
在任中カルフォルニア大学バークレイ分校CSRGの 4. 4BSD
開発プロジェクトに参加。Large Part of Contributor として
オムロン社製ＷＳへの移植に従事。

1996年 2月より現職。
FreeBSD/Linux を応用した各種試作開発に従事。