Ultra Monkeyによる負荷分散とその拡張

2003年6月20日

黒澤崇宏(kurosawa@valinux.co.jp)
VA Linux Systems ジャパン 株式会社　技術部

目次

- Ultra Monkey**概要**
- LVS**のチューニング**
- **新機能**：TCP**フェイルオーバー**
- **新機能**：Active-Active**構成**

Ultra Monkeyとは

- http://www.ultramonkey.org/
- **高信頼なロードバランサ構築のためのソリューション**
- **以下のソフトウェアから構成**
  - heartbeat
  - LVS
  - ldirectord
- **個別のソフトウェアを統合されたソリューションとしてパッケージング**

heartbeat

- High-Availability Linuxプロジェクトが開発
  http://www.linux-ha.org/
- ハイアベイラビリティクラスタの構築
  ―Active-standby構成
- クラスタ内ノードのダウン検出・資源管理
  - VIP(仮想IPアドレス)管理
  - 共有ディスク
  - 各種プロセスの起動・終了
- Ultra Monkeyでは主にIPアドレス引継ぎに使用

ldirectord (Linux Director Daemon)

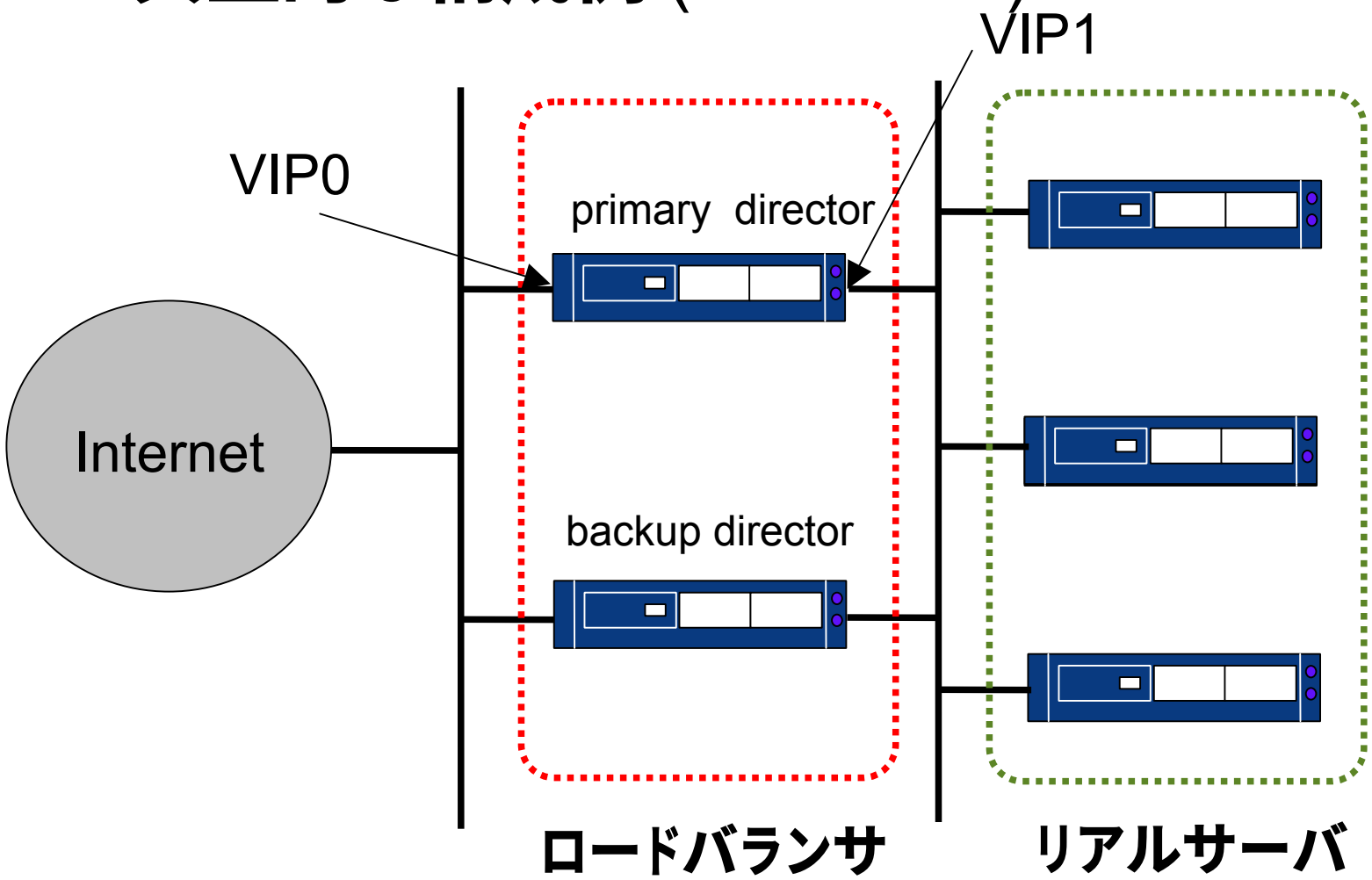
- heartbeatに付属するソフトウェア
- リアルサーバの生死を監視
- リアルサーバの状態変化に応じてLVSの転送テーブル更新
- TCPのサービスを監視 (!=ノード監視)
  - FTP/SMTP/HTTP/POP3/NNTP/ IMAP4/LDAP/HTTPS
  - connect()のみの監視も可能
  - Perlで書かれている →(その気になれば)新プロトコルへの対応も比較的容易

LVS(Linux Virtual Server)

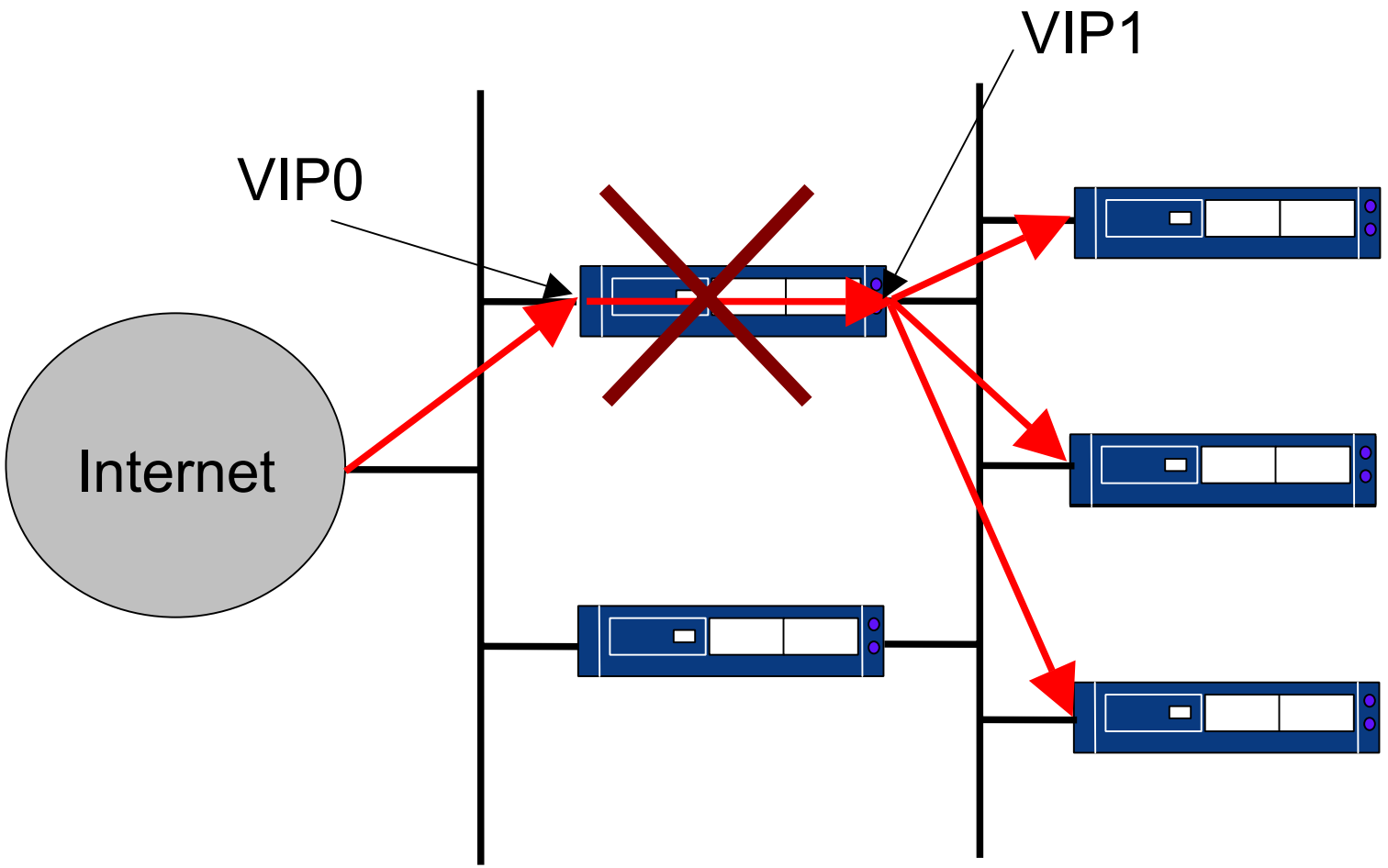
- Linux Virtual Server**プロジェクトが開発**
  http://www.linux-vs.org/
- **ロードバランサ構築のためのソフトウェア**
- **レイヤ**4**までのロードバランサ**
  レイヤ5-7**は不可**
- **カーネル内のみでパケット転送**
  →コンテキストスイッチ・メモリコピーのオーバーヘッド
  が少ない

Ultra Monkeyによるロードバランサ

典型的な構成例 (LVS/NAT)

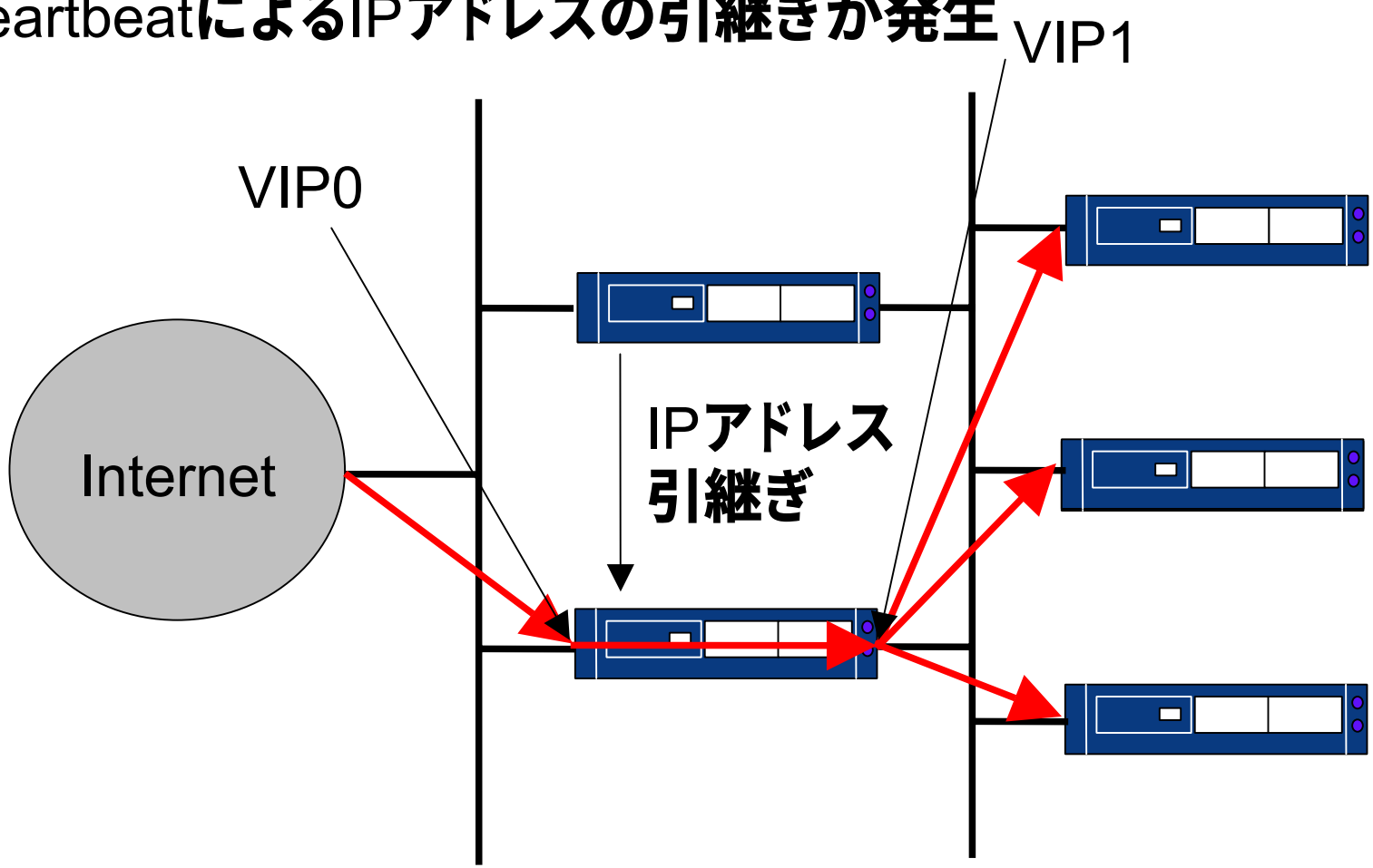


ロードバランサ故障時の対応

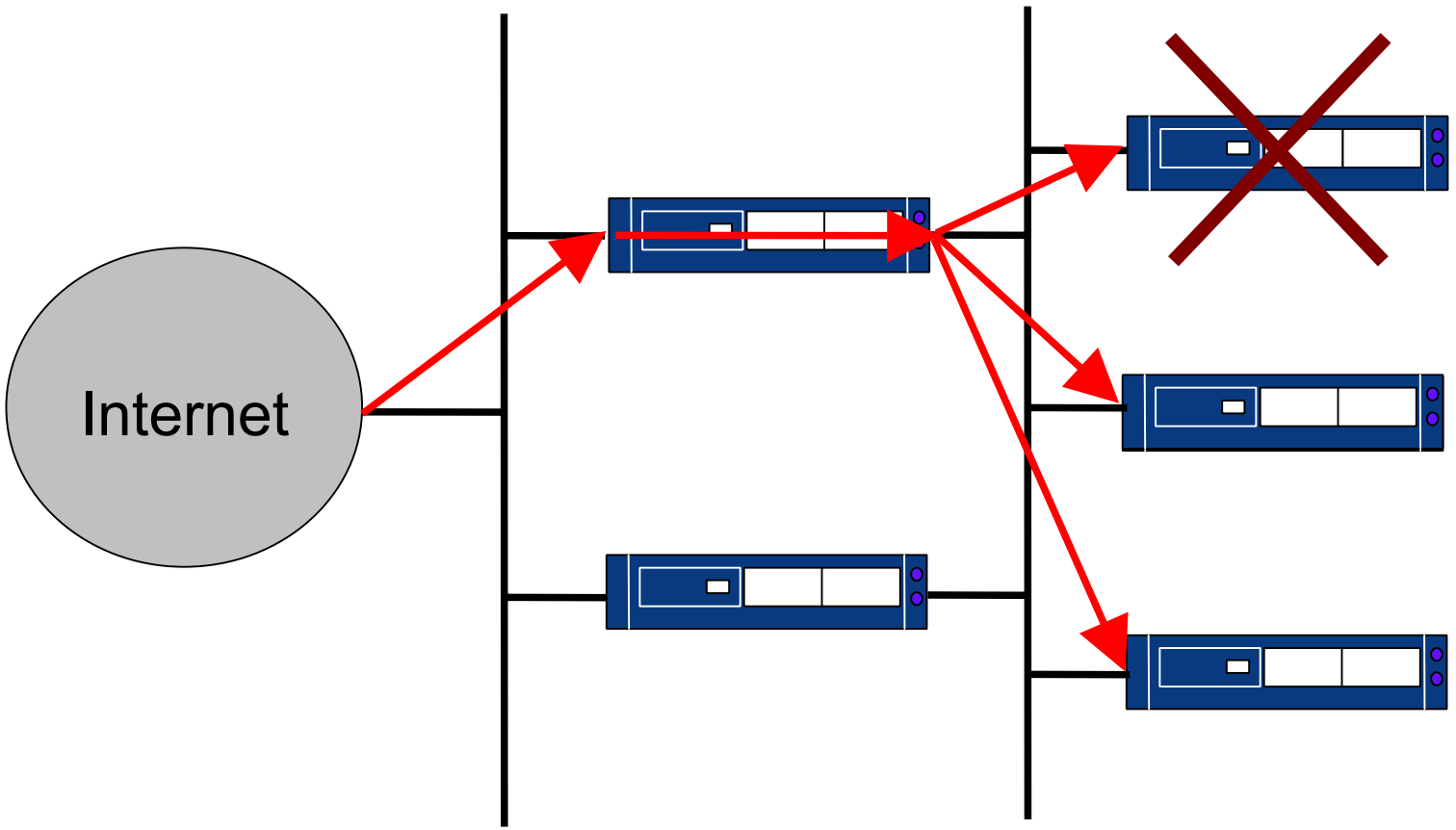


ロードバランサ故障時の対応

heartbeatによるIPアドレスの引継ぎが発生

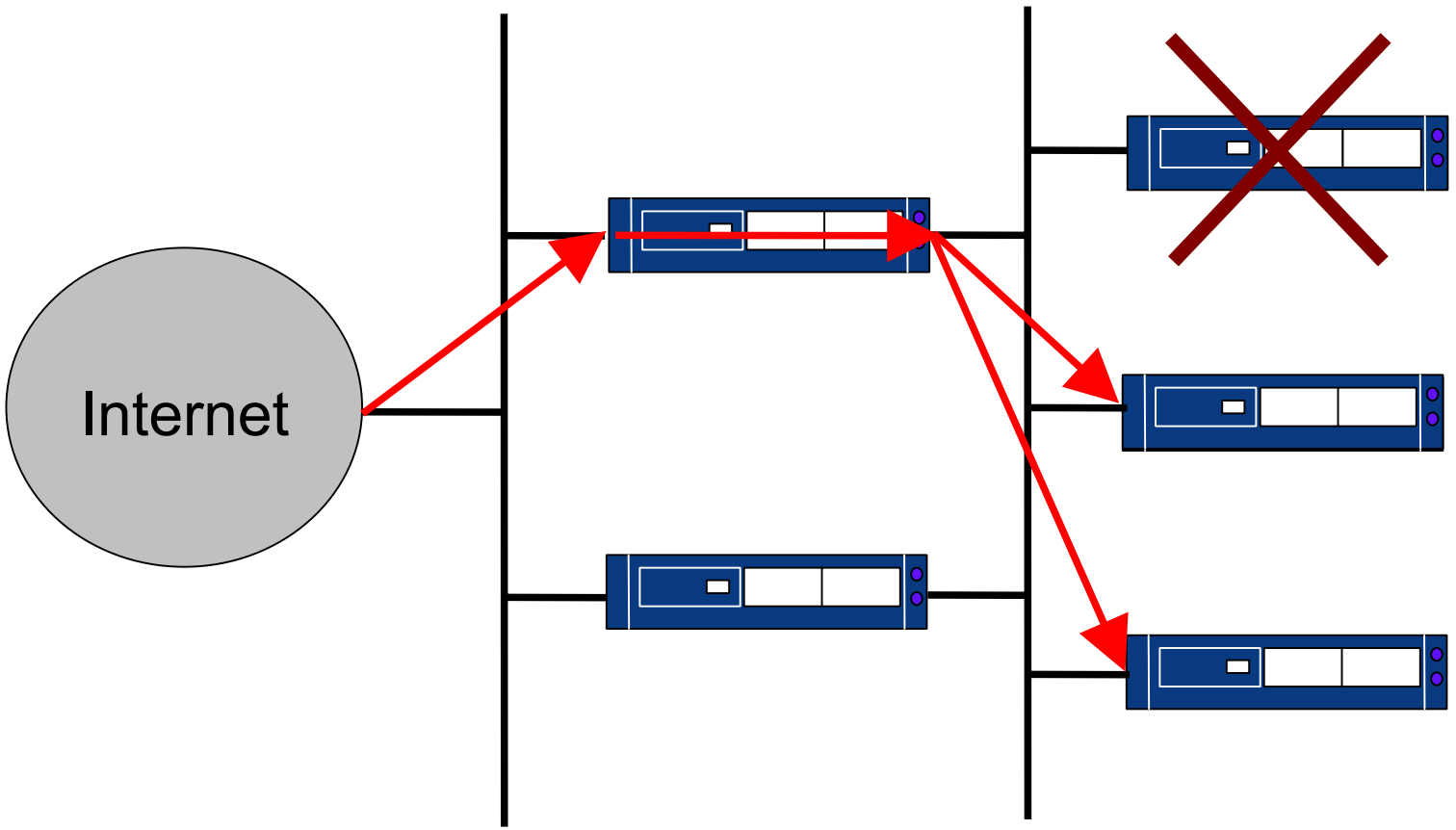


リアルサーバ故障時の対応



リアルサーバ故障時の対応

故障したリアルサーバの排除



パケット転送方式

- LVS/NAT
  (Network Address Translation)
- LVS/DR
  (Direct Routing)
- LVS/TUN
  (Tunneling)

パケット転送方式 ー LVS/NAT

- IPアドレス変換(とポート番号変換)によりリアルサーバへパケットを転送
- リアルサーバへの特別な設定は不要
  →ロードバランサ導入の際の移行が比較的容易
- ロードバランサへの負荷
  - ーアドレス変換処理s
  - ーリアルサーバからの戻りパケットの処理

パケット転送方式 ー LVS/DR

- パケットのレイヤ2の宛先のみを書き換えて転送
- ロードバランサへの負荷：小
  - ーレイヤ2のみの宛先書き換え
  - ー戻りパケット：ロードバランサを経由しない
- リアルサーバに対する制限
  - ーロードバランサと同一セグメント内
  - ー仮想IPアドレスを宛先とするパケットの受信

パケット転送方式 ー LVS/TUN

- IPカプセル化による転送
  (IPヘッダの前にさらにIPヘッダを追加)
- リアルサーバの位置に対する自由度：高
  リアルサーバが外部にある場合に使用
- ロードバランサへの負荷

LVSチューニング

なぜチューニング？

- 某商用ロードバランサとの性能比較
  - 帯域幅はLVSが上
  - 秒間あたりのリクエスト処理数：LVSの方が下
- CPUその他のハードウェアの性能面ではLVSで使用したマシンの方が上もしくは同等
  →なぜリクエスト処理数の性能がでないのか？

チューニングポイント

- ネットワークカード
- uniprocessor機上でのチューニング
- NAPIパッチ
- コネクションハッシュテーブルサイズ
- /proc/irq/.../smp_affinity
- LVS内ロック競合の解消
- NICドライバ

ネットワークカード

- 比較対象とは異なるカードを使用していた
  →同じカード使用で同等の性能を達成
- カード種別により性能差が存在
  (ギガビットイーサ使用)
- 遅いカード･速いカードの見極めが必要

- しかし，CPU性能ではLVS使用のハードが上のはず
  →より高い性能が目指せるのではないか？

uniprocessor機上でのチューニング

- SMPカーネルでは性能劣化
  - ーspin_lock, atomic系の関数のオーバーヘッド
    →uniprocessor上では無駄なコスト
- 結論：uniprocessorマシンではuniprocessorカーネルを使用すべき
- 問題点：NIC割り込み頻発によりユーザプロセスの動作に問題発生

NAPIパッチの適用

- 大量のパケット受信時の割り込み頻発を回避
- 基本的なアルゴリズム
  - —パケット受信時に割り込みマスク・クリア
  - —ポーリングによりパケット到達チェックと受信処理
  - —NICの受信バッファをすべて処理したら割り込み許可
- 結果：ユーザプロセスへの悪影響消える
- NICのinterrupt coalescing機能を使う手もある

コネクションハッシュテーブルサイズ

- kernprofによりプロファイリング取得
  http://oss.sgi.com/projects/kernprof/
- ハッシュの探索関数の処理の比重が大きい
  →ハッシュのバケツリストが長くなっていることが問題
  (リスト長計測によっても確認)
- CONFIG_IP_VS_TAB_BITSにより指定
- 性能改善の効果あり

LVS内ロック競合の解消

- SMPマシン上の性能はどうか？
  →uniprocessorマシンと大差なし．何かがおかしい？
- lockmeterによりロック競合の発生を調査
  ―2箇所の競合を検出：いずれもパケット受信ごと
- 統計情報採取のロックが競合
  ―不要な統計情報は取らないオプション追加
- タイムアウト更新のロックが競合
  ―不要なタイムアウト更新を抑止

そのほかのチューニング

- NICドライバ
  - CPUキャッシュが効きやすくなるよう変更
    一方のCPUでさわった領域は他方のCPUでさわらないようにする
    - skbuffの回収タイミングに着目
- /proc/irq/.../smp_affinity
  - ハードウェア割り込みを特定のCPUにくくり付け
  - NIC2枚それぞれにCPUを割り当て

チューニング結果

- **環境**
  - CPU：Pentium III 1BGHz×2
  - **メモリ**：2GB
  - NIC：Intel PRO/1000
  - **リアルサーバ**：6**台,** Apache2
  - **クライアント**：7**台,** WebStone
- **チューニング結果**
  - **チューニング前**：18000reqs/sec**程度**
  - **チューニング後**：25000reqs/sec**程度**