Debianを使ったSite構築 web & mail

そしておまけのDB

荒木靖宏
Debian JP Project

Debianを使ったSite構築 web & mail

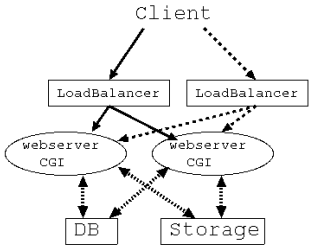
最も簡単な例:
簡単setup
- apt-get install apache postfix mailreader

今日のメニュー
- サービス第一のシステム構築法
- 武器としてのDebian

Debianを使ったSite構築 web & mail

目標
- webメーラ+メールサーバ+αの提供
- ユーザは多数

どうつくるのか、それ自体が問題



ユーザは厳しいです

サービスへのあくなき要求

- いつでも、十分な速度を要求
  - ⇒High Availability, Stability(anti-cracking)
  - ⇒Scalability
- ⇒現在のトレンドは、具体的に数値を設定:SLAへ
  - 使えないときは通知されること
  - 数値目標の達成を掲示
  - 運用に加えて監視の重要性大

High Availabilityとは?

ポイント:
- 一箇所の故障をサービス全体に影響させない

⇒止まるのは、リソースがひとつであるため
- 単純にapacheが載ったサーバとか。。
- ネットワーク経路がひとつしかないとか。。

Scalabilityとは?

ポイント:
- ボトルネックはどこにある?
- 増加しつづけるサービス要求に対する透過的処理能力増強
- 実際に1サーバでは限界が

⇒DNSやL4スイッチによる方法が一般的
⇒データ共有が問題になりがち

failover

- システムとしてのサービス継続
- 完璧なものはない、という意識
- サービスによる違いあり
  - 楽なもの:
    - store and forward型のアプリケーション
    - ユーザが直接意識しないもの
  - 辛いもの:
    - ユーザデータを扱うもの
    - 頻繁にwriteされるもの

複数ホストによる大規模システム構築

- 問題多数
  - 仕事(役割)の分担
  - 個々のホストのsetup
  - システムの監視の問題

いまどきのサイト構築

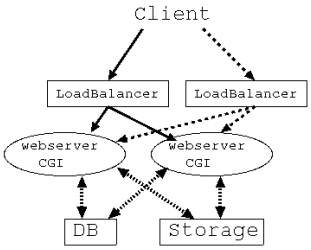
運用とともにセキュリティも考えておきましょう

目的、原則、方針、手順

- システムの目的とセキュリティ保持の目的を定める
- サービスとセキュリティのバランス決定
- 原則実現の基本方針。責任を明確にする
- 具体的な手順や方法を定める

大規模特定サービスモデル

webをつかったサービスなどでは必然的に仕事が分別される



- 相関がきちんと定義されて、影響は小
- side effectとしてsecurity向上も計れる

サービスの切り分け

動いてるシステムを手直しするならば必要

システム設計の見直し

- 人間による見直しが確実
- 論理的に分解

toolの利用

- lsof
  - 動作中のプロセスが使用しているファイル情報を示す
- pstree, ps, ldd..

ディストリビューションを使わない選択

ディストリビューションは必要悪という立場もあり

- アプリケーション等インストール情報の喪失懸念

一方で。。

- インストールは?
- 本当にそのインストール情報、わかっていますか?
- 大量に同じ目的のシステムを作れますか?

こんな人にはディストリビューションはいりません!

私|仲間|社員|部下には...

- 十分な知識がある
- 専門化しまくり
- ホストが一台
- ホストは一代
  - 複製が必要ない場合

ディストリビューション

利点

- そもそも多数のコピーを柔軟に作製するのに便利なシステム
- Linuxシステム構築の現実的な方法

沢山あるけど何を選ぶ? 選択基準

- 一般的な見方
  - パッケージの充実度|使い易さ
- サーバを構築するなら
  - バグ、セキュリティホールのupdate速度
  - 商用プログラムを使うならその対応
  - Trinuxのような最小限のディストリビューションを選ぶ手も

さあ、Debianです!

- 真のフリーのシステム
  - mainを使った自由なサービス構築が可能
- 最新のパッケージの容易な入手
  - ->unstable版, testing版もどうぞ
- 安全なパッケージをどうぞ
  - ->security.debian.org
- 簡単な複数台のセットアップ
  - aptとaptサーバ

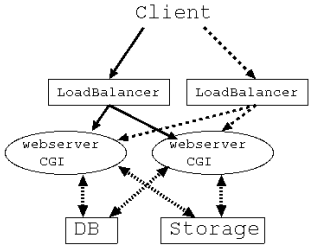
ディストリビューション利用の指針

Debianなら容易!

- 自分が把握しているパッケージのみ
- 必要最小限
- 最新のパッケージ

- 必要な付加プログラムはパッケージングする
  - 同一のテスト環境の構築も容易

今回のシステムの要素



- 関係アプリ
  - DNS(ネットワークを使う前提)
  - SMTP(mailなのだから)
  - web(http) -> これがメイン

DNSのfailover

- システムとして考慮
  - root-servers.netからして複数
- 複数のネームサーバを登録可能
  - 優先順位なし

- whois linux.or.jp
  - p. [ネームサーバ] mizuho.linux.or.jp
  - p. [ネームサーバ] minori.linux.or.jp

Mail/SMTP配送のfailover

- システムとして考慮
- preferencesが定義可能
  - 優先度高いものが失敗⇒低いサーバへ
- 再送キューの存在
- nslookup -type=mx debian.org
  - debian.org preference = 0, mail exchanger = master.debian.org
  - debian.org preference = 5, mail exchanger = murphy.debian.org
  - debian.org preference = 10, mail exchanger = klecker.debian.org

Commercial Solution / Software

- LifeKeeper
  - Oracle, Informix, Apache, Sendmail (On Linux)
- Understudy
  - web, file, mail クラスタユーティリティ (On Linux, BSD, NT..)
- RSF-1
- TurboLinux High Availabilty Cluster
- Mod_Redundancy (Apache on Linux, Solaris)
- Legato Cluster
- WebSphere Performance Pack

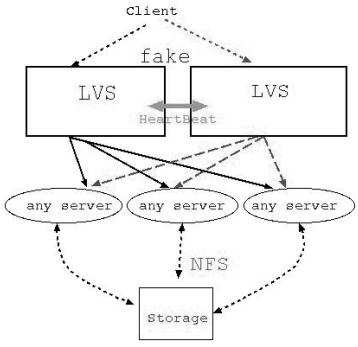
Commercial Solution / hardware

- BIG-IP(BSD?)
- Equalizer(Linux?)
- Phobos(include NIC)
- Alteon(sw, WebOS)
- ServerIron(sw)
- Cisco Arrow point, local director
- Cabletron(LSNAT)

Commercial Solution / Problem

- 方法、実装は???
  - 情報が開示されていない
- サポート
  - Linuxがサポートされていないことも
    - 一般にWindowsNT, Solaris向け
    - 大抵はOKだが。。

LVS:Linuxでよくある方法



Webサーバファームの問題点

- Network failover
- ロードバランシング方法
- コンテンツ/サービスの同一性保証
- ログ管理

⇒以下、構成要素と解決例を見ていく

Network failover

- 重要性大:
  - ほとんどのクラスタでの基本技術
- DNSによる方法
  - 遅い(cache時間の存在)。ラウンドロビンが楽
- MACアドレスのtakeover
  - 即応。対応していないNICもある
- IPアドレスのtakeover
  - 反応速度は中くらい
  - arp expire時間くらい
  - L3スイッチなどによってはexpireが必要

IP冗長化 / fake

- ARPをうばうIP冗長化スクリプト
- HeartBeartに統合
- 方式:
  - プライマリのNICとセカンダリのNICを用意
  - プライマリのNICを監視し、死んだらセカンダリがプライマリのIPを奪う

IP冗長化 / VRRPd

- IPをtakeover
- rfc2338で定義
  - multicastによる情報共有
  - 多くのrouterが準拠
  - vipの重み
- VRRPのLinux実装
  - カーネルにもいくつか要求有
    - IP alias
    - netlink socket
    - routing messages

Load-balancing方法

- クライアントからのコネクションをどのように実サーバに結びつけるか
  - アプリケーションproxy
    - URI毎，User-Agent毎，などの細かな制御が可能
  - L4スイッチ
    - 単純なリレー
- 静的コンテンツなら専用のreverse cacheをする方法も。
  - accelarate mode on squid, commercial products,,

LVS

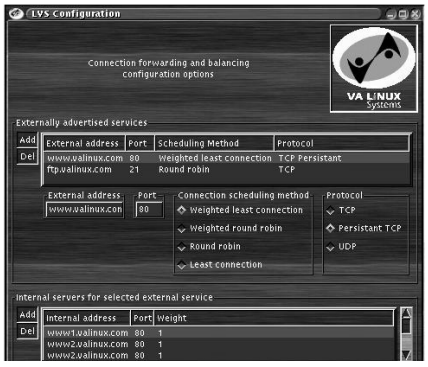
- Linuxでの汎用TCP/UDPロードバランシング
- kernel patchが必要
- kernel埋めこみ型ロードバランシング
  - RoundRobin, WeightedRoundRobin, LeastConnection,WeightedLeastConnection, locality-based least-connectionカーネルモジュール

設定例:

```
ipvsadm -A -t 202.103.106.5:80 -s wlc
ipvsadm -A -t 202.103.106.5:21 -s wrr
ipvsadm -a -t 210.165.45.101:80 -R 172.16.0.2:80 -m
ipvsadm -a -t 210.165.45.101:80 -R 172.16.0.3:8000 -m -w 2
ipvsadm -a -t 210.165.45.101:25 -R 172.16.0.2:25 -m
```

LVS(2)

- lvs自体では実サーバの挙動を監視できない
  - ⇒monなどの監視ツールと併用する
- lvsの設定をguiで行うものも



mod_backhand(libapache-mod-backhand)

- アプリケーションレベルのwwwクラスタ
- Apache Directive毎の設定が可能
- loadbalancing method
  - Age, Rundom, Cost
  - CPU, Load
    - 実webserver情報の利用
    - マルチキャストで状態を共有。監視でも利用

設定例:

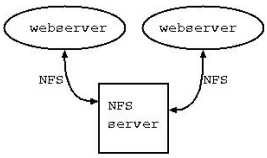
```
MulticastStats 240.220.221.20:4445,3 ⇒広告アドレス、ポート、hop
AcceptStats 10.0.5.128/25
<Directory /data/docs>
Backhand byAge
</Directory>
```

コンテンツ/サービスの同一性保証

- 分散するサーバへのレプリケーションが問題
  - rsync, storage share
  - 静的コンテンツ
    - ReverseCacheもあるし問題はうすい?
  - 動的コンテンツ
    - ロック、速度、様々な問題あり

NFS

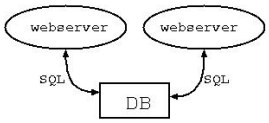
- 現実的かつ実績のある解
  - ロック、パフォーマンスに問題も



DBを用いる方法

- DBの結果からwebserverでコンテンツを構築
- ECサイト等、多くの場面で用いられる方法
  - ロックの問題や、データ保全はDBに依存

例:



ログあれこれ

- ログ管理はいつも悩みのタネ
- 重要性大
  - アクセス管理
  - 稼働状況の把握
  - ビジネス上の理由
- 解析ツールの利用
  - lire, analog, awstats, calamaris, fwlogwatch
- syslogによるものが多い
  - application独自より楽な場合も
  - ネットワーク越しにログの集中可能
  - 問題点: udpの利用
    - 記録漏れの懸念
  - ⇒syslog-ngへ
    - TCPによるログ飛ばし
    - 柔軟な設定が可能

アクセス制御(from network)

カーネルレベルでのIPの制御

- ipchains, iptable

- 評価対象:
  - source, destination, インターフェース
- 処理:
  - ACCEPT, DENY, REJECT, MASQ(IP masquerade), REDIRECT, RETURN

アプリケーションレベル

- それぞれのアプリケーション(当然)
- tcpwrapper+inetd, tcpserver, xinetd
  - 広く使われているスーパーデーモン(application layer)
  - 柔軟なアクセス制御を提供

システムの監視

最終的には監視あってはじめてわかる

システムとしての監視

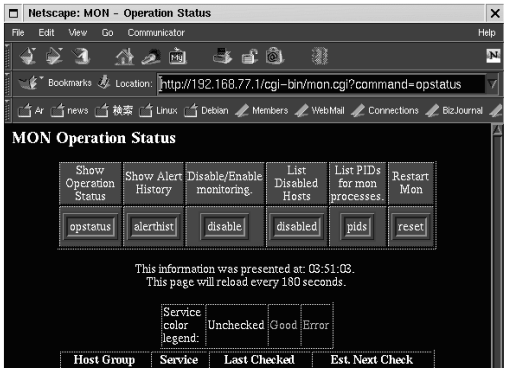
- サービス内容を満たしているかどうか
- 動作テスト

構成ホスト個々の監視

- file構成、内容
  - tripwire, integrit

監視ツール / mon

構成:スケジューラ + 監視プログラム + 通知プログラム
それぞれ独立,容易な拡張



mon(2)

- *マルチレベルでの監視が可能*:
  - *附属プログラム*:
    - L7: SMTP, Telnet, FTP, NNTP, POP-3, IMAP, LDAP, DNS, mSQL, MySQL, RPC
    - L4: *任意*tcp port
    - L3: *ホスト間*RTT*値*, ping
    - upime, process,Disk*残容量*
- *必要なら、監視以外のツールももちろん利用可能*

monでの通知 + スケジューラ

- *週単位での時間帯制御*
- *異常検知後のプログラム指定*

設定例

```
hostgroup router cisco7504    ⇒router group 作成
watch router                  ⇒router group の監視を宣言
  service ping                ⇒pingサービスについて
  interval 5m                 ⇒5分ごとの監視
    monitor fping.monitor     ⇒fping.monitorプログラムで監視
    period wd {Mon-Fri} hr {7am-10pm}
      alert mail.alert ar@domain.com
```

So much for today!

どうもありがとうございました。

Happy vacationing!

resource 1

mon
- http://www.kernel.org/software/mon/

coda
- http://www.coda.cs.cmu.edu/

vrrpd
- http://w3.arobas.net/~jetienne/vrrpd/

Understudy
- http://www.polyserve.com/prod_overview.html

resource 2

RSF-1
- http://www.polyserve.com/prod_overview.html

IBM WebSphere
- http://www-4.ibm.com/software/webservers/perfpack/index.html

Piranha
- http://sources.redhat.com/piranha/

Linux Virtual Server patch
- http://www.LinuxVirtualServer.org/software/index.html

resource 3

Ultra Monkey
- http://ultramonkey.sourceforge.net/

lvs-gui
- http://www.au.vergenet.net/linux/lvs-gui/

syslog-ng
- http://lists.balabit.hu/products/syslog-ng/