

Soluzioni Cluster per l'alta disponibilità dei servizi



Gorizia, 1 dicembre 2001

Overview

- Il problema della High Availability, o RAS
- Soluzioni basate su Linux
- Chi vi parla:
Marino Miculan
Docente di Sistemi Operativi
Dipartimento di Matematica e Informatica,
Università degli Studi di Udine
miculan@dimi.uniud.it

Alcuni tipici scenari

- **Servizi Internet**
 - E-Commerce: il cliente deve essere soddisfatto
 - Home banking: transazioni rapide e sicure
 - Providers (Web hosting)
 - Proxy (caching di rete), DNS
 - E-Content providers (quotidiani on line...)
- **Servizi Intranet/Extranet**
 - Workflow/ datamining aziendali
 - Chain Resource Management, SCM...

High Availability (Reliability, Availability, Serviceability)

- **Disponibilità: 24x7, o "number of nines":**
 - 99.9%: 8 ore 45 minuti all'anno
 - 99.99%: 52 minuti 36 secondi all'anno
 - 99.999%: 5 minuti 15 secondi all'anno
- **Manutenibilità**
 - Riconfigurazione, upgrade, diagnosi e risoluzione on line
- **Scalabilità**
 - aumentare gradualmente il sistema al crescere del carico
- **Gestibilità**
 - La complessità di gestione non deve crescere sensibilmente
- **E che costi poco!**

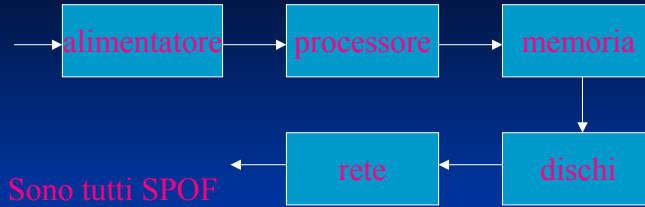
Una catalogazione dei sistemi di calcolo per disponibilità

1. Sistemi general-purpose
PC, server interni non mission-critical...
2. Sistemi ad alta disponibilità e scalabilità
Minori interruzioni possibili. Adattamento al carico.
Server aziendali, mission-critical, banche, telecom...
3. Sistemi life-critical
Real-time, controlli militari, sistemi embedded...
4. Sistemi senza assistenza
Sonde spaziali, satelliti...

Alcune cause di indisponibilità (in ordine di complessità)

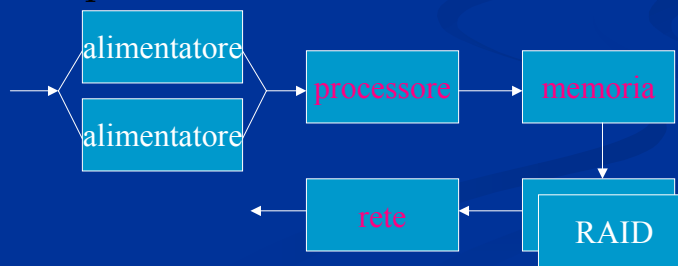
- Hardware
 - Mancanza di corrente
 - Guasto dei componenti
- Software
 - Bug
 - Upgrade/patch
- Sovraccarico
- Errore Umano
- Calamità Naturale
- Attacco Hacker/Terroristico

Catene funzionali e SPOF



■ SPOF: Single Point of Failure

Componente critico in una catena funzionale.



MTBF: Mean Time Between Failure

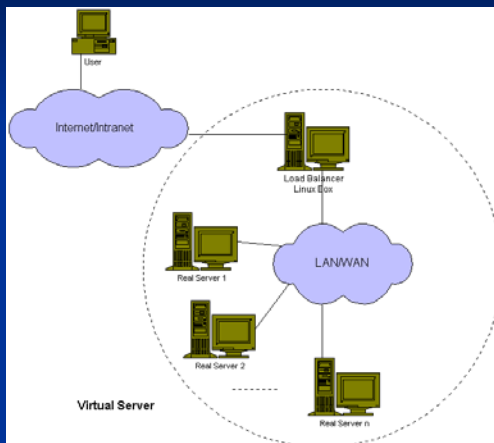
- Tempo medio tra due interruzioni consecutive
 - Hardware MTBF: della sola componente hardware
 - System MTBF: di tutto il sistema nel suo complesso (quello che conta per il servizio)
- Ridondanza hardware fa **umentare** il system MTBF ma **diminuire** l'hardware MTBF (e quindi maggiore manutenzione)

Soluzione 1: Server Singolo (Processori tightly coupled)



- Semplice (inizialmente)
- Non molto scalabile
 - SMP: limitato a poche vie (processori)
 - NUMA: molto costoso
- SPOF
 - Hardware diventa cruciale
 - Singoli componenti possono essere ridondanti e hot pluggable ma il costo aumenta

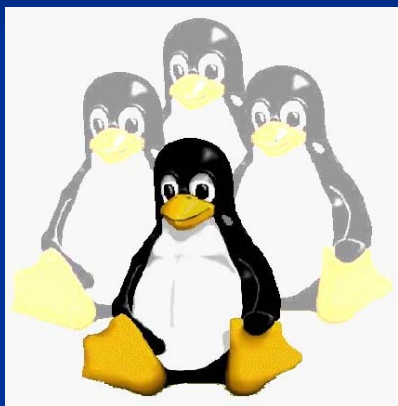
Soluzione 2: Cluster (Processori loosely coupled)



- Trasparente all'utente
 - Il cluster appare come una sola macchina
 - Applicabile a WAN
- Scalabile
 - Si aggiungono nuove macchine alla bisogna
- Fault-tolerant
 - Riconfigurazione automatica nel caso di fallimento di un nodo
- Economico
 - HW non specializzato

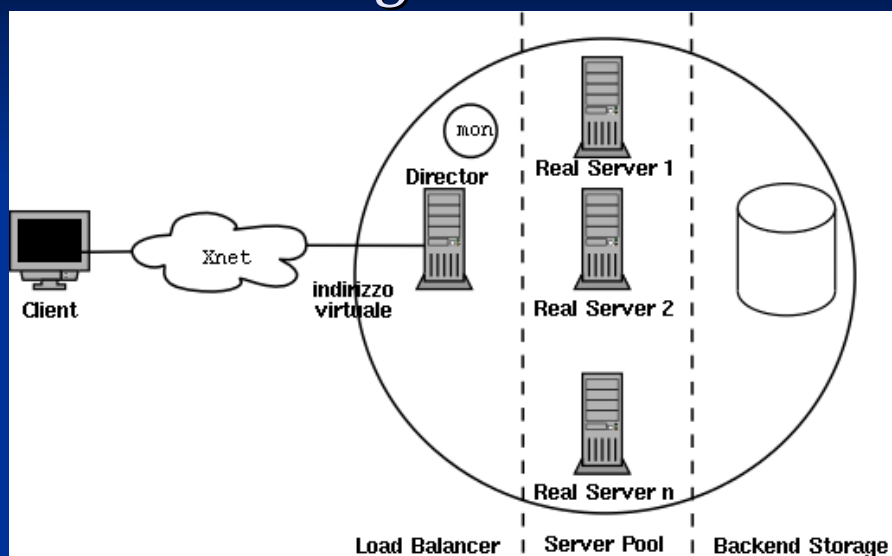
Linux Virtual Server + Linux-HA

Soluzione per cluster scalabili ad alta disponibilità
risultante dall'unione di più iniziative Open Source.



- Bilanciamento del carico
- Failure detection e recovery
- Riconfigurazione automatica
- Prezzo imbattibile!

Architettura generale LVS-HA



Front end: Load Balancer (Director)

- È l'unica macchina visibile dall'esterno
- Riceve le richieste dai client e le reinvia ai veri server (switch di livello 4), e viceversa
 - (in alcune configurazioni)
- Tutto questo avviene direttamente in kernel
 - IPVS: patch per il kernel
 - sfrutta SMP
 - basso overhead
 - un singolo director può gestire decine di real server

Pool dei Real server

- Implementano il servizio (WWW, Mail, SMB, DNS, ...)
- Possono avere caratteristiche (anche S.O.) diverse
- Scalabile
 - nuovi nodi possono essere aggiunti run-time
 - il director può essere riconfigurato run-time
- Fault-tolerant per ridondanza
 - Lo stato dei real server viene monitorato dal director
 - In caso di rottura di un server, il director si riconfigura automaticamente escludendo il nodo

Monitoraggio risorse

-MON-

- mon è un sistema generale di monitoraggio di risorse (indipendente da LVS), scritto in Perl
- Architettura client/server a plugin
 - **monitors**: programmi client che verificano una data condizione/servizio (qualsiasi cosa).
 - **mon** è il server che riceve le notifiche dai monitors
 - **alerts (upalerts)**: script eseguiti da mon in seguito ad un failure/ritorno in funzione
 - Monitors, alerts e upalerts possono essere aggiunti e modificati in qualsiasi momento

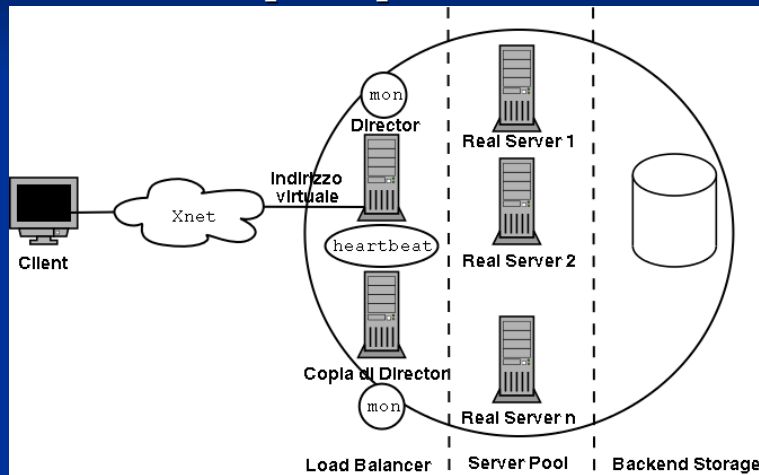
mon e LVS



- mon (client, server, alerts) viene eseguito sul director
- I monitor verificano il corretto funzionamento dei real server
- Gli alert riconfigurano il balancer togliendo il nodo fuori servizio (e spedendo un SMS al sistemista) e reintegrandolo non appena viene riparato
- I client non si accorgono di niente (o quasi...)

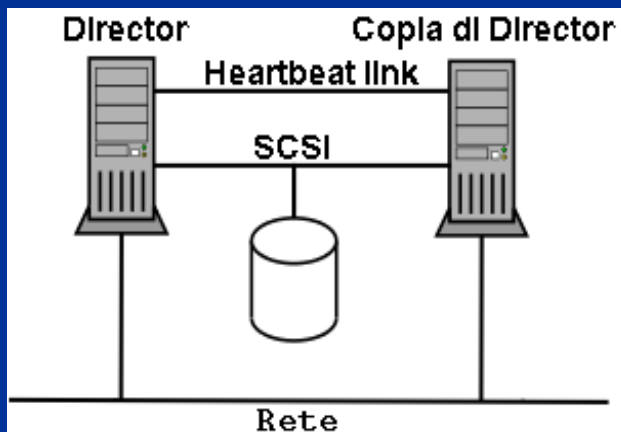
Il Director è un SPOF

- Soluzione: una copia del director, pronta a subentrare al principale in caso di failure



Heartbeat

Software (indipendente da LVS) che permette il subentro (*takeover*) della copia in caso di failure del master



Heartbeat

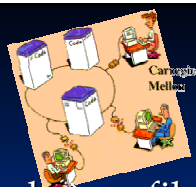
- Implementa un heartbeat tra due (o più) nodi via seriale o UDP (possibilmente su crossover)
- Tipici valori: heartbeat=2s; deadtime=10s
- Quando il master viene riconosciuto morto:
 - Takeover delle risorse da parte del replica
 - indirizzi IP (via ARP gratuito)
 - mounting di device condivisi
 - lancio di servizi
 - Eventualmente supporta anche STONITH (shoot the other node in the head)
- Quando il master ritorna in vita (ritorno di battito), il replica rilascia le risorse

Backend storage

La logica di memorizzazione,
eventualmente condivisa tra i server.

Un campo sterminato! Di interesse:

- File System journalled
 - EXT3, ReiserFS, JFS, XFS, GFS...
- File System distribuiti fault-tolerant
 - Coda, GFS, Intermezzo
- DBMS e/o applicativi legacy



CODA

Coda è un filesystem distribuito derivato da AFS2, e realizzato alla Carnegie Mellon University.

Tra le molte caratteristiche:

- Operatività offline per computazione mobile
- Alta performance attraverso client side persistent caching
- Replicazione dei Server
- Operatività anche durante parziale guasto della rete dei server
- Scalabile
- Open Source

Politiche di Scheduling

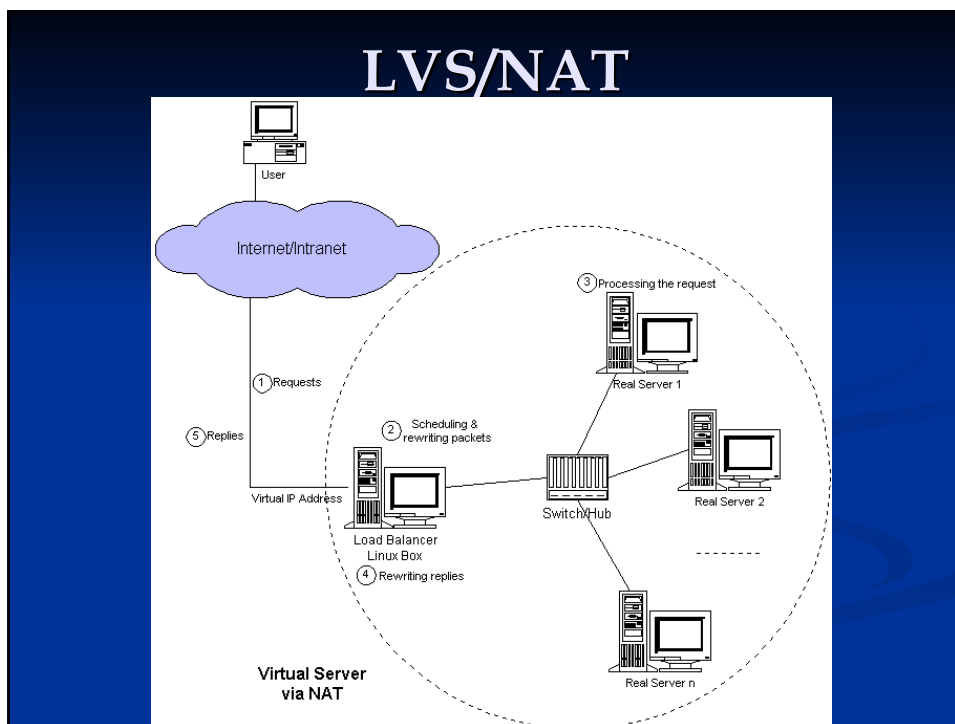
- Round Robin
- Weighted Round Robin
 - Si assegna un peso (potenza) ad ogni real server
- Least-Connection
 - Si seleziona il real server con il minor numero di connessioni attive (non adatto in ambienti eterogenei)
- Weighted Least-Connection
 - Si assegna un peso ad ogni real server; si seleziona quello con il minore rapporto #connessioni/peso
- Persistent Client Connection
 - Connessioni persistenti (ftp, https) dallo stesso client vengono reindirizzate sempre allo stesso real server

Politiche di scheduling (per cache cluster)

- Locality-Based least-connection
 - In base all'IP di destinazione, si tende a selezionare lo stesso server (se non sovraccarico)
- Locality-Based least-connection con replicazione
 - Come sopra, con sottocluster dinamici per IP destination
- Source/Destination IP Hashing
 - Tabella hash statica sugli indirizzi IP di source/destination

Tecniche di bilanciamento IP

- LVS/NAT: Network Address Translation
 - Il director instrada sia le richieste sia le risposte dei real server (rete cieca)
- LVS/TUN: Tunnelling su IP
 - Il director instrada solo le richieste con incapsulamento IPinIP; risposte dirette al client
- LVS/DR: Direct Routing
 - Il director instrada solo le richieste con riscrittura MAC su LAN; risposte dirette al client



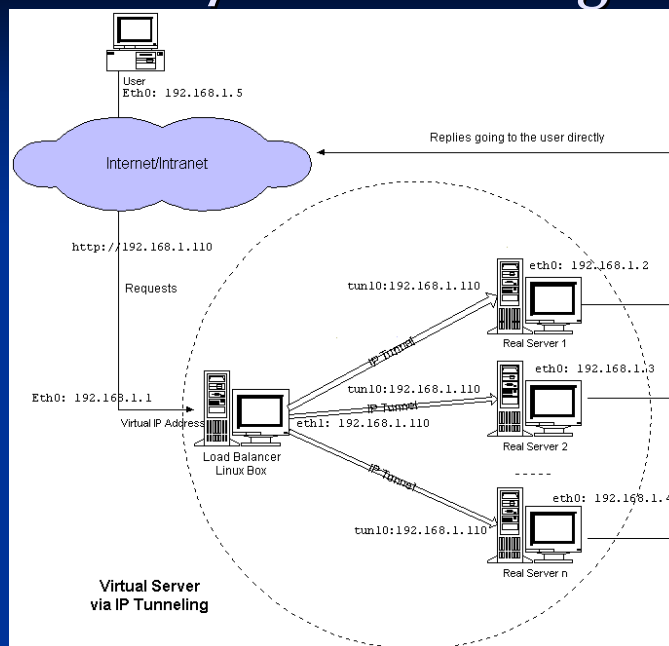
LVS/NAT

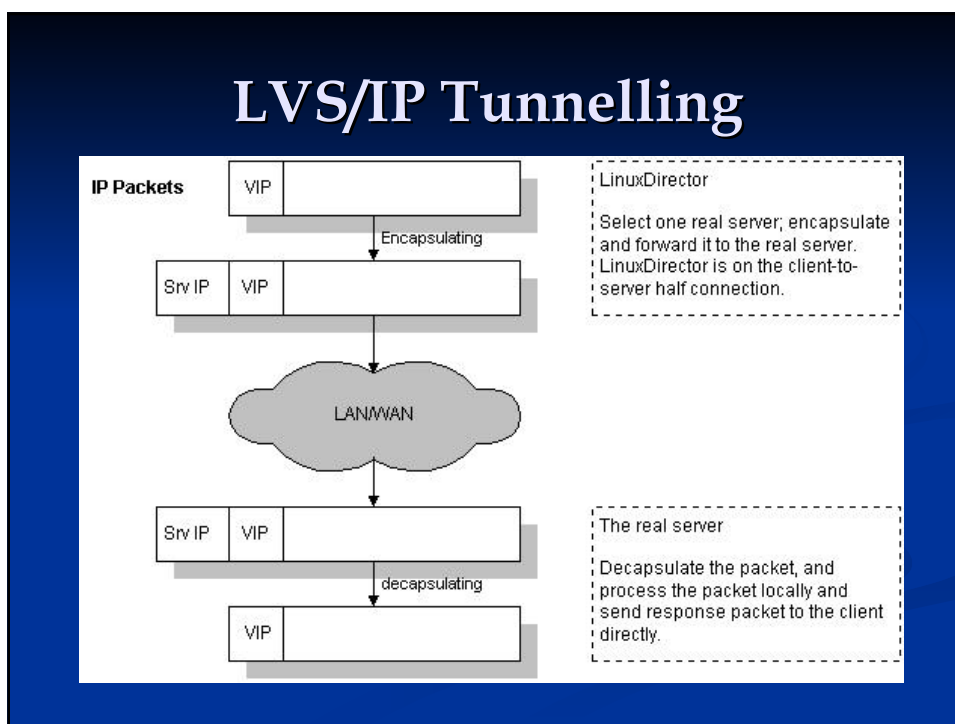
	Source	Destinazione
Richiesta dal client	202.100.1.2:3456	202.103.106.5:80
Richiesta al realserver	202.100.1.2:3456	172.16.0.3:8000
Risposta real server	172.16.0.3:8000	202.100.1.2:3456
Risposta al client	202.103.106.5:80	202.100.1.2:3456

LVS/NAT

- Qualsiasi real server può essere usato, senza modifiche
- I server possono risiedere su rete privata
- Sufficiente 1 indirizzo IP per tutto il cluster
- Director diventa collo di bottiglia: deve gestire tutto il traffico
- Poco scalabile

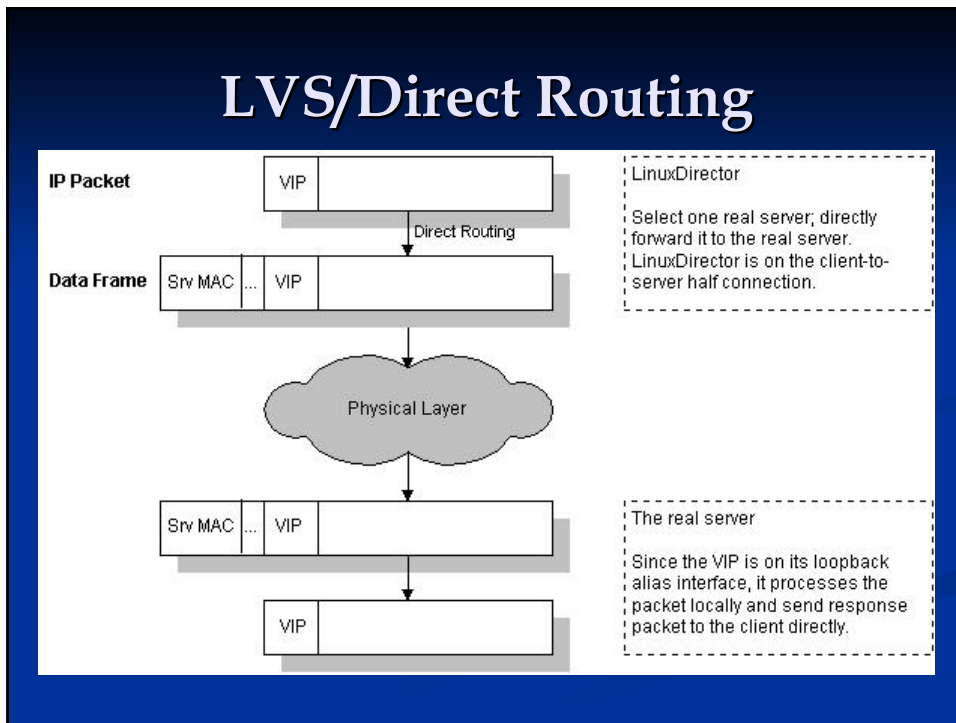
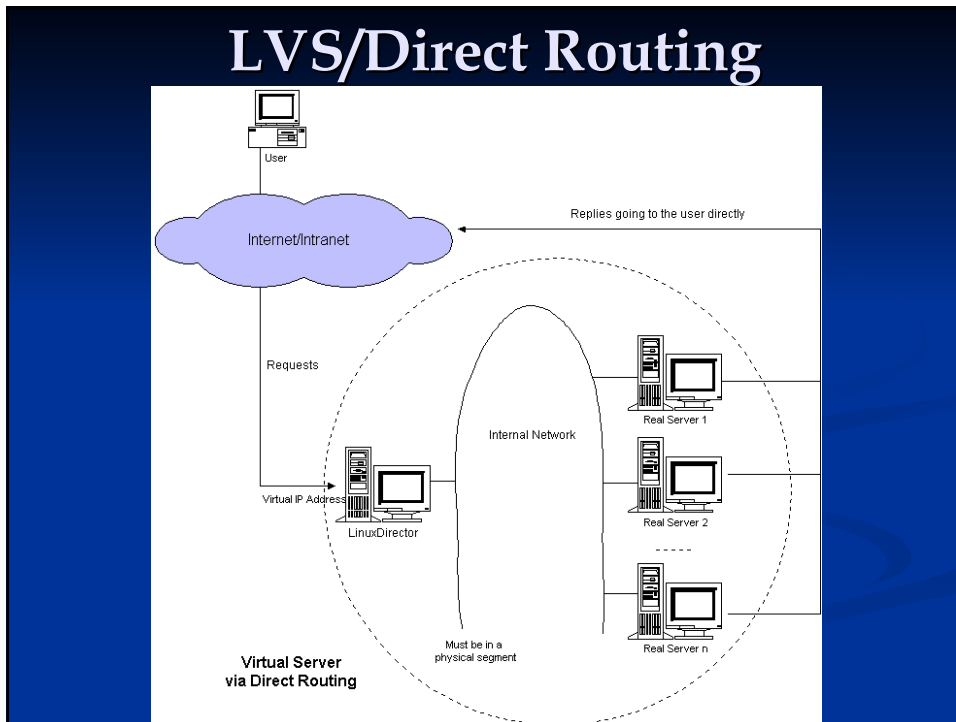
LVS/IP Tunnelling





- ## LVS/IP Tunnelling
- Il director deve gestire solo le richieste
 - Le risposte non passano per il director
 - Maggiore scalabilità
 - I real server possono essere anche su reti diverse, o remote (*cluster geografici*)

 - OS dei server deve supportare IP tunnelling
 - I server devono avere indirizzo IP pubblico
 - Overhead di incapsulazione



LVS/Direct Routing

- Il director deve gestire solo le richieste
- Le risposte non passano per il director
- Alta scalabilità
- Non c'è overhead di IP tunnelling

- Director e real servers devono stare sulla stessa rete fisica (hub/switch livello 2)
- I server devono avere indirizzo IP pubblico
- OS dei server deve implementare tunnels (interfacce non-ARPing)

Alcune installazioni di Linux Virtual Server

Linux.com	Portale Linux	LVS web cluster
Sourceforge.net	Sviluppo	LVS (ftp, http, https, ssh, cvs)
Themes.org	Repositorio temi	LVS web cluster
wwwcache.ja.net	Servizio proxy JANET (UK)	40 server in 3 cluster LVS
datingfacing.com	Incontri	LVS (~2Mhit/d)
www.zope.org	Web app. server	LVS web cluster
www.real.com	Multimedia	LVS (>20 nodi)
www.tiscali.it	Provider	LVS web cluster

Altre soluzioni su Linux

- RedHat Piranha: simile a LVS/NAT
 - IPVS kernel patch
 - nanny per monitorare i real server
 - pulse per gestire il replica del director
 - Piranha : GUI di amministrazione
- Heartbeat+ Linux Director Daemon
 - ldirectord è già installato con Heartbeat
 - Studiato appositamente per monitoraggio LVS
 - Più facile ma meno generale di mon

Conclusioni

- LVS-HA permette di costruire facilmente cluster ad alta disponibilità
- Permette il bilanciamento di qualsiasi servizio su TCP o UDP
- Scalabilità e fault-tolerance
- I real server non devono essere modificati
- Non supporta scheduling content-based (I servizi da bilanciare devono essere omogenei)
- Open source: modificabile secondo necessità
- Prezzo imbattibile!

Perché pagare il doppio?

Riferimenti

- www.linuxvirtualserver.org
- www.linux-ha.org
- www.coda.cs.cmu.edu
- www.kernel.org/software/mon/
- www.redhat.com/support/wpapers/piranhaha/