Tecniche per la salvaguardia della disponibilità ed integrità dei sistemi di elaborazione e delle informazioni

3. clustering e cloud computing

Marco Prandini
Università di Bologna

Disponibilità continua dei sistemi

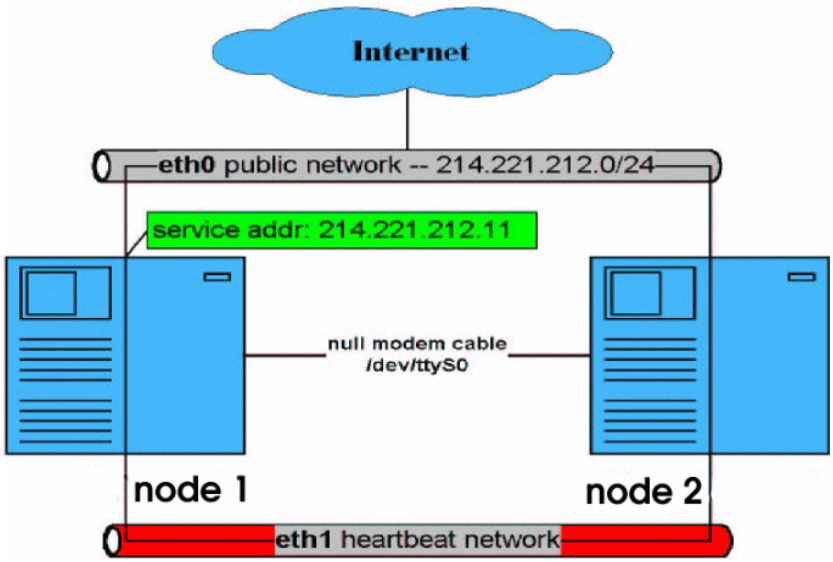
- L'affidabilità dello storage è solo uno dei componenti per la disponibilità dell'intero sistema
  - alimentazione
  - memoria
  - chipset e processore
  - ventilazione
  - schede
  - connessione alla rete
- Ognuno di questi componenti può essere ridondato
  - l'architettura risultante può essere di straordinaria complessità e costo
  - il vantaggio è quello di avere a che fare con un unico sistema
    - amministrazione semplificata
    - nessuna problematica di coerenza tra dati replicati

Disponibilità continua dei sistemi

- Approcci alternativi nella definizione delle risorse logiche:
  - *virtualizzare* il sistema per poterlo riprodurre su di un host diverso semplicemente copiando una manciata di file
    - molteplicità di host, ma con poche problematiche di coerenza
    - beneficio collaterale: server consolidation
    - ottimo isolamento tra le diverse VM
    - buona scalabilità, anche se l'overhead per la virtualizzazione di servizi semplici è elevato
  - realizzare più copie dello stesso sistema, ciascuna con tecnologie standard: *cluster HA* (high availability)
    - economico ed efficiente
    - più problematica la scalabilità
- In ogni caso serve una tecnologia per gestire automaticamente lo spostamento delle risorse logiche sui nodi fisicamente disponibili

Heartbeat

- Segnalazione della vitalità
  - Il modo più semplice di rilevare la necessità di sostituire una risorsa guasta è scambiare periodicamente pacchetti di vitalità (*heartbeat*)
    - è sicuramente consigliabile farlo attraverso link multipli per non confondere il fallimento del link con quello del nodo
    - non è sufficiente per capire se il servizio viene erogato correttamente
      - monitor aggiuntivi per la verifica di risposte sensate da parte dei server applicativi



Failover/Failback

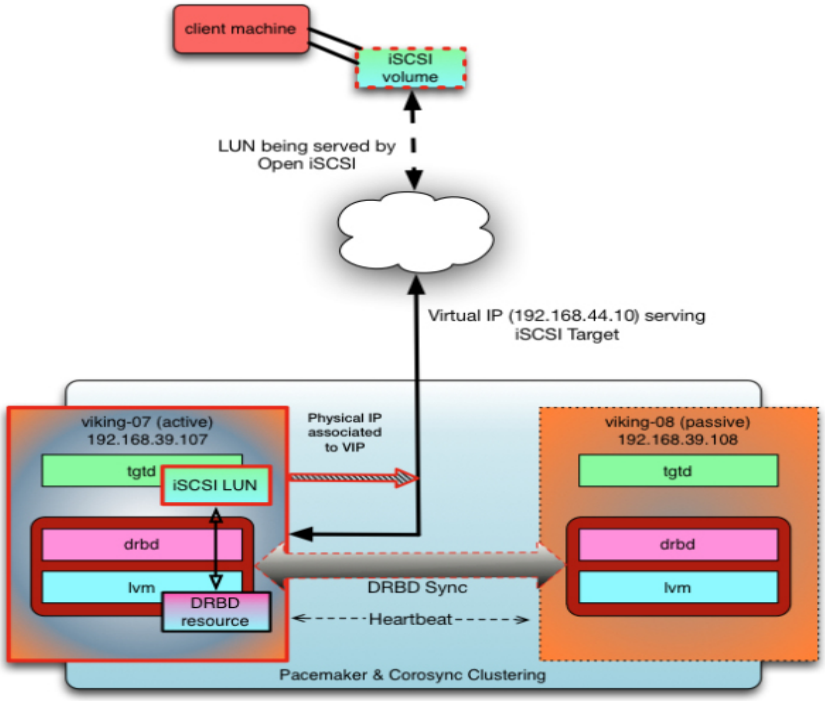
- La scomparsa e ricomparsa di nodi attivi richiede la gestione di due condizioni:
- *Failover*: la situazione in cui un nodo si guasta e le risorse del cluster lo devono sostituire. Problemi:
  - decidere correttamente chi è guasto, gestendo le situazioni di incomunicabilità (split brain)
  - evitare il conflitto di possesso delle risorse
- *Failback*: la situazione in cui un nodo precedentemente guastatosi ripristina la sua funzionalità. Problemi:
  - mantenere le sessioni in corso e ridurre i tempi di commutazione superflui...
    - "nice failback", chi ha le risorse eroga il servizio
  - ... o sfruttare al meglio la capacità computazionale derivante dai nodi ritornati disponibili
    - "auto failback", chi è dichiarato "master" per una risorsa la riprende appena torna online

Un esempio di failover: IP address takeover

- La prima fase successiva alla rilevazione del guasto di un nodo è tipicamente la presa in carico, da parte di un altro, della sua identità di rete
- Questo può essere fatto a 3 livelli diversi di rete – in ordine dal più rapido ma complesso al più lento ma semplice:
  - MAC takeover (con schede e SO predisposti... ma gli switch?)
    - non sempre applicabile
  - DNS reconfiguration (...resolver cache)
    - latenza non affidabilmente controllabile
  - IP takeover (...arp cache)
    - la scelta più comune
    - Per non pregiudicare la raggiungibilità dei nodi e per consentire di distinguerli senza ambiguità, ad ogni nodo viene assegnato un IP address reale (fisso), ed al cluster un IP "collettivo" (floating/virtual/cluster IP), che viene assunto come alias dal nodo master attivo.

Un esempio di failover: HA iSCSI

- Una volta configurato un floating IP, il nodo che lo detiene eroga attraverso questo i servizi, ad esempio un target iSCSI
- Uso corretto di DRBD+iSCSI: il target daemon reinizializza la propria state machine al failover



http://blogs.mindspew-age.com/2012/04/05/adventures-in-high-availability-ha-iscsi-with-drbd-iscsi-and-pacemaker/

Partizionamento

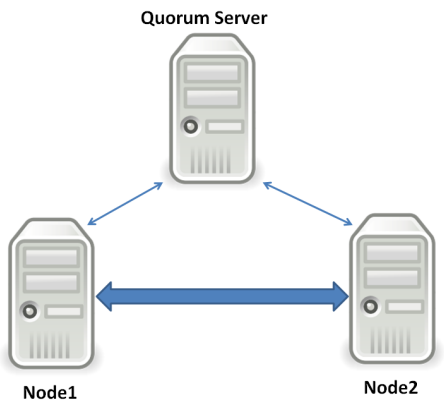
- Situazione ideale di un cluster
  - ogni nodo è in una di due condizioni:
    - disconnesso dagli altri e senza possibilità di accedere a risorse condivise
    - perfettamente funzionante
- Situazione reale
  - un guasto può *partizionare* l'insieme dei nodi, creando sub-cluster
    - in un sub-cluster i nodi si vedono perfettamente
    - non c'è comunicazione tra un sub-cluster e l'altro
    - ogni sub-cluster mantiene un certo grado di operatività
      - può accedere a risorse condivise
      - deve decidere se farlo o meno!
  - anche in assenza di guasti i transitori rendono possibile determinare *solo con un certo grado di probabilità*, da un nodo:
    - se questo sta tentando di associarsi al sub-cluster più opportuno
    - se questo dispone di una visione del sub-cluster omogenea a quella degli altri nodi

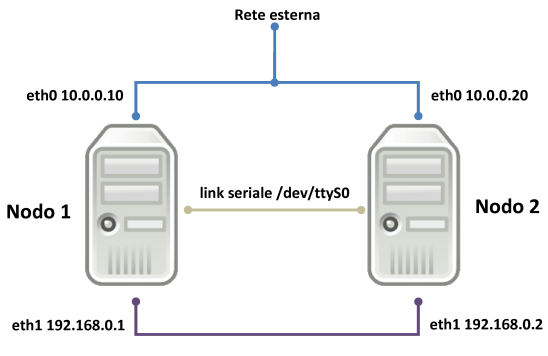
Fencing

- Un modo per garantire che un sub-cluster non acceda alle risorse condivise è "recintarlo"
  - resource fencing: ogni nodo può privare gli altri dell'accesso ad una precisa risorsa
    - e.g. spegnere la porta dello switch con cui il nodo accede ad una SAN per proteggere un disco condiviso
  - node fencing
    - e.g. spegnere il nodo tramite una scheda di gestione remota (Dell DRAC, SUN/HP iLOM), l'UPS o la power strip che alimenta il sistema
    - STONITH (Shoot The Other Node In The Head)
- Chi decide in caso di autentico partizionamento (non di guasto del nodo) chi deve effettuare il fencing e chi lo deve subire?

Quorum

- Nei cluster con N>2 membri, è possibile adottare un criterio semplice e robusto per determinare quale sub-cluster debba ritenersi attivo
  - ogni nodo ha un voto
  - se un sub-cluster raggiunge V>N/2 voti, raggiunge il quorum ed ogni nodo che ne fa parte può operare
  - non può esistere più di un sub-cluster che detiene il quorum
- Nei cluster a due nodi, questo sistema previene il funzionamento del nodo superstite quando l'altro fallisce. Si può operare
  - o tramite sistemi di arbitraggio esterni al cluster (fisicamente o logicamente)
  - o contando su ridondanze hardware che garantiscano la comunicazione tra i nodi quando questi sono operativi, per poi consentire il funzionamento con V=1

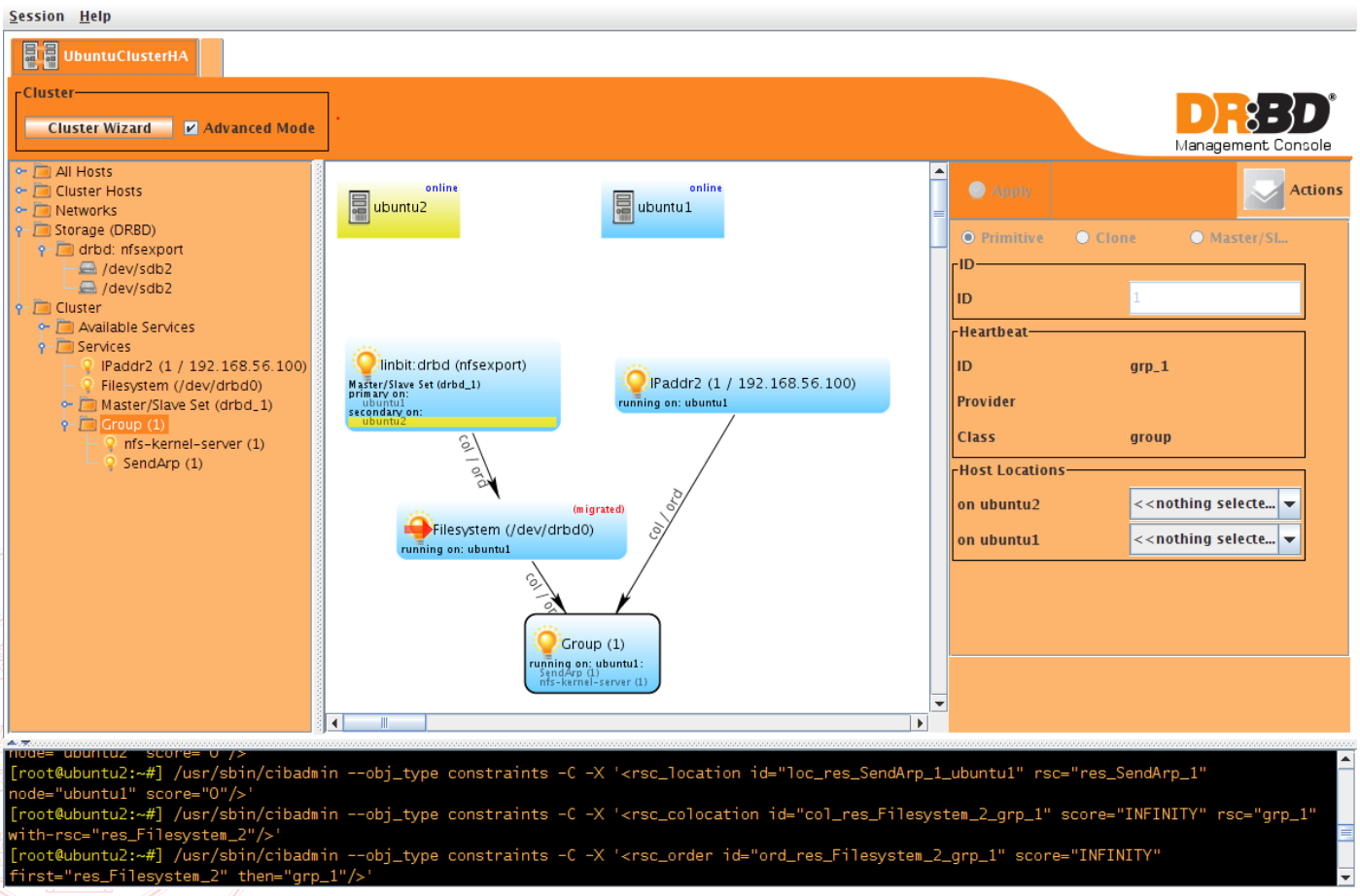




Sistemi di clustering HA per Linux

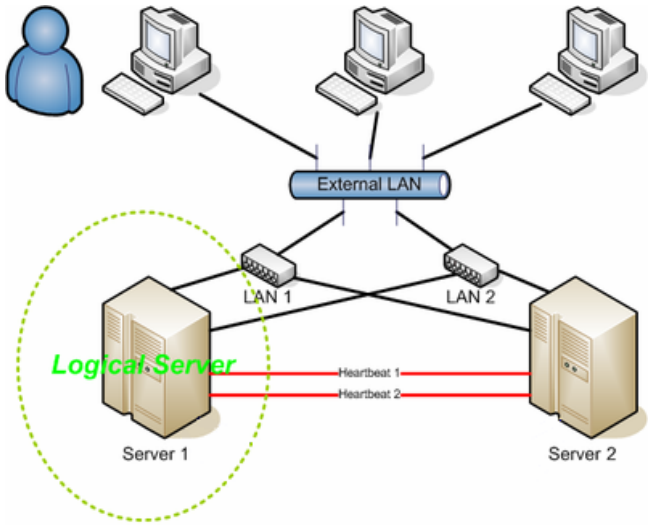
- Linux-HA (http://www.linux-ha.org/)
  - a volte chiamato Heartbeat per il suo componente più anziano e più evidente, è il sistema di HA-clustering open source più stabile (1999)
  - per pulizia ed interoperabilità, il sottosistema di cluster resource management è stato separato, dando origine al progetto Pacemaker (http://clusterlabs.org/wiki/Pacemaker)
- OpenAIS (http://www.openais.org/)
  - implementazione delle specifiche pubbliche AIS (Application Interface Specification) del Service Availability Forum
    - interfacce e politiche standard per lo sviluppo di applicazioni HA
    - http://www.saforum.org/ → specifications→ AIS
    - pensato per cooperare con Corosync cluster engine (http://www.corosync.org/)
- Sistemi sviluppati da vendor
  - RedHat Cluster Suite, Novell/SuSE Cluster Suite, Symantec Veritas Cluster Server, IBM Cluster Systems Management, HP Serviceguard, ...
- http://www.squidoo.com/linux-clustering

Management per i sistemi open – DMC



Linux-HA

- La soluzione più tradizionale per i cluster a 2 nodi Active/Standby
- Nella versione 2 supporta anche più di 2 nodi

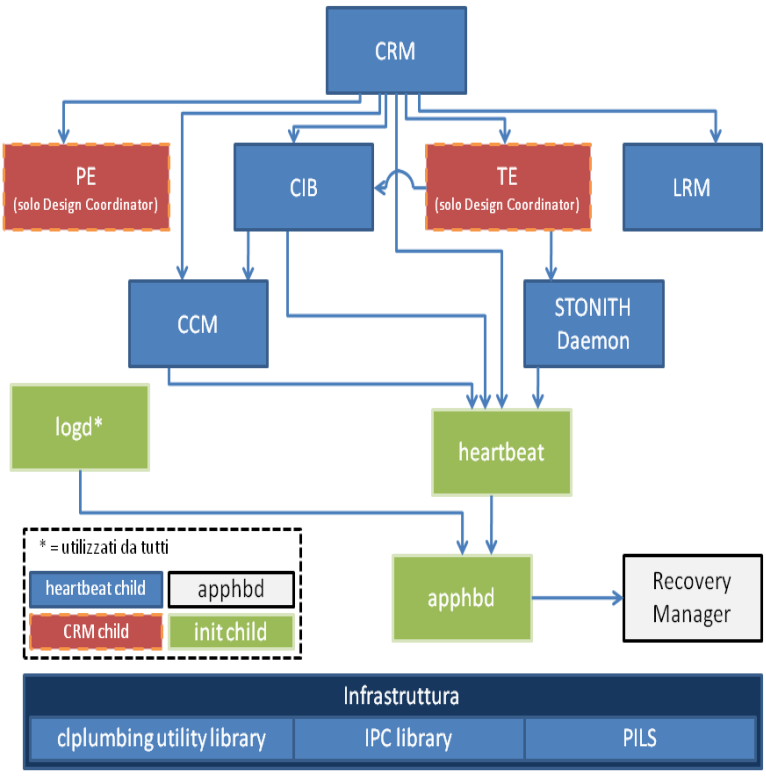


- Eccellente tutorial:

http://www.linux-ha.org/_cache/HeartbeatTutorials__LWCE08-ha-tutorial.odp
da http://www.linux-ha.org/HeartbeatTutorials

Linux-HA: architettura

- Heartbeat: comunicazione tra i nodi
  - CRM: Cluster Resource Manager
  - PE: CRM Policy Engine
  - TE: CRM Transition Engine
  - CIB: Cluster Information Base
- CCM: Strongly Connected Consensus Cluster Membership
- LRM: Local Resource Manager
- Stonith Daemon: fencing
- Logd: demone non bloccante per i log
- Apphdb: servizio di watchdog timer a livello applicazione
- Recovery Manager: servizio di recovery delle applicazioni
- Infrastruttura
  - PILS: Plugin and Interface Loading System
  - IPC: Interprocess Communication
  - Cluster "plumbing" library
- CTS: Cluster Testing System, suite di test per stressare il cluster

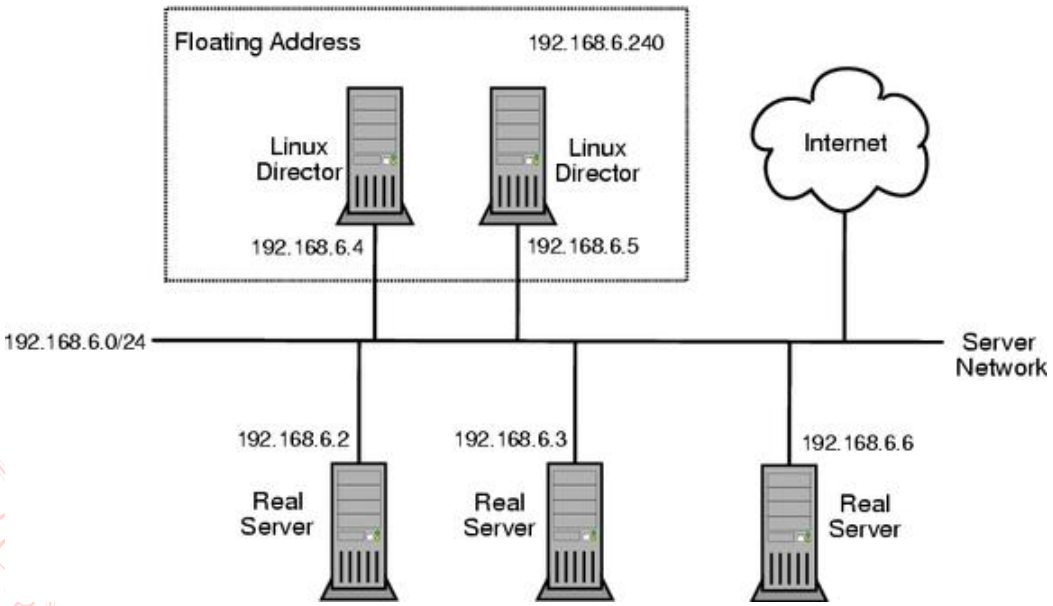


HA + LB

- Con Linux-HA non è facilissimo realizzare strutture scalabili che bilancino il carico su di un pool espandibile di nodi
- Il componente ideale per questo è LinuxVirtualServer (LVS)
  - *director* per smistare le richieste
    - si presenta all'esterno come unico *virtual server*
    - ridirige le richieste secondo un algoritmo di bilanciamento ai *real server*
  - diverse strategie di instradamento delle risposte
    - NAT
    - Tunnelling
    - Direct Routing
- I real server non hanno esigenze di HA, poichè sono tutti intercambiabili per costruzione
- Il director invece è un SPOF, ma non ha esigenze spinte di scalabilità

HA + LB

- Soluzione: ridondarlo in modo semplice con Linux-HA



Red Hat Cluster Suite

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/

- RedHat, la prima azienda nella storia di Linux ad offrire supporto commerciale ad una distribuzione, produce una delle piattaforme server più diffuse: Red Hat Enterprise Linux
  - esiste anche per desktop
  - RH ha cessato il supporto alla distribuzione open, che ha forkato Fedora Core
    - RH partecipa attivamente alla comunità di FC
    - FC è il testbed per molte novità di RHEL
    - CentOS, prendendo dal software senza problemi di licenza di FC, distribuisce gratuitamente un "clone open" di RHEL
- RHCS **fornisce** con pochi integrati da **componenti nativi progetti open source**

sistema di clustering	ccs, cman, fence, rgmanager
sistema di load balancing	lvs (pulse, nanny)
sistemi di gestione	system-config-cluster, conga (luci, ricci)
sistemi di monitoraggio	zabbix
cluster filesystem	clvm, gnbd, dlm, gfs

Red Hat Cluster Suite - componenti

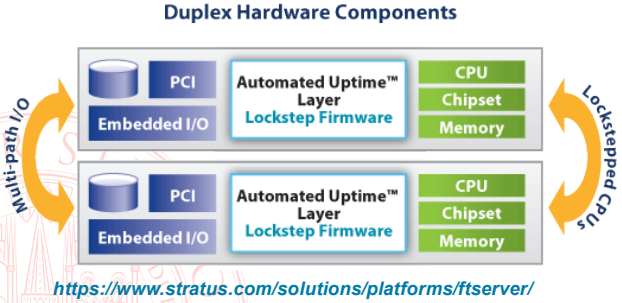
- Cluster Configuration System (CCS)
  - mantiene le direttive di configurazione sincronizzate tra i nodi
    - Tutte in un file XML `/etc/cluster/cluster.conf`
- Cluster MANager (CMAN)
  - gestisce join, leave e segnalazioni di vitalità dei nodi
  - calcola il quorum e quindi se un nodo può erogare servizi
- Fence Daemon
- Resource Group Manager (Rgmanager)
  - definisce i subset di nodi su cui può essere collocato un servizio (all'avvio, al failover, al failback)
  - permette di gestire manualmente i servizi

Sintetizzando...

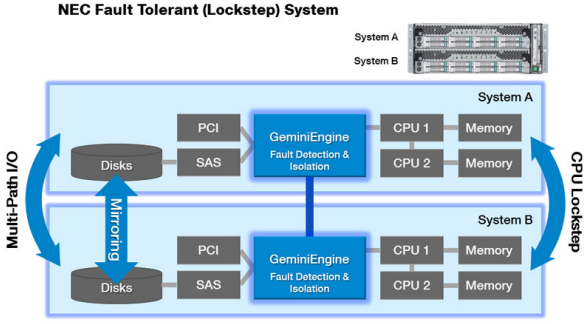
- I sistemi di clustering
  - hanno come obiettivo l'erogazione in alta disponibilità delle risorse
  - adottano un approccio totalmente decentralizzato
    - nella determinazione dei nodi disponibili
    - nella collocazione dei servizi sui nodi
- L'approccio decentralizzato non scala bene come si pensava
  - alto overhead ed alta sensibilità dei sistemi di heartbeat
  - evoluzione verso sistemi cloud
    - sistemi chiave (es. gestione risorse, image repository) non hanno problemi di carico, solo di HA → replicati in modalità cluster minimale (2 nodi)
    - numero arbitrario di compute nodes su cui allocare i servizi
- In ogni caso...
  - in caso di guasto (recuperabile), il downtime tipico è ≥ 1 minuto
    - il tempo necessario a dichiarare *failed* un nodo non può essere troppo basso, o errori transitori innescherebbero *fencing wars* letali
    - la risorsa, tipicamente una VM, ha tempi di avvio propri (boot + eventuale riparazione del filesystem)

Approcci alternativi – hardware

- Con le architetture cluster non possiamo garantire i five nines.
  - ma il loro rapporto prezzo/prestazioni è eccellente per la maggior parte delle applicazioni (ad esempio, con la migrazione live delle VM, i downtime pianificati possono durare meno di 1 secondo).
- Se veramente serve di più, bisogna ricorrere ad approcci hardware
  - mirroring dell'intera architettura di calcolo basata su chipset proprietari che riproducono tutte le operazioni identicamente su processori diversi
- Ovviamente molto costosi, e poco scalabili



https://www.stratus.com/solutions/platforms/ftserver/

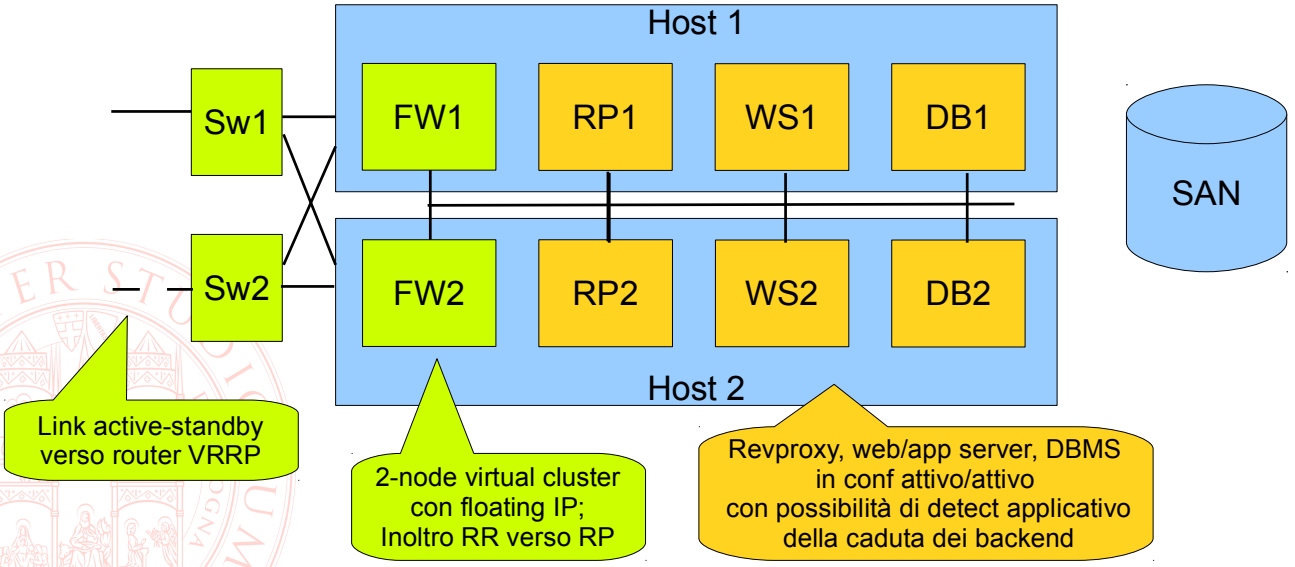


http://www.nec.com/en/global/prod/express/fault_tolerant/technology.html

http://www.cio.com/article/732499/Will_Fault_Tolerant_Servers_Improve_Your_Virtualized_Environment_?

Approcci alternativi – software

- Dove è possibile, si può ricorrere alla ridondanza per eliminare completamente i single point of failure a livello applicativo
  - La virtualizzazione aiuta molto
  - Serve grande cura nella scelta delle configurazioni e nel tuning
  - Non tutto è replicabile senza problemi e con le stesse prestazioni



Cloud computing

- Le architetture HA sono di grande interesse per il sistemista coinvolto in progetti di portata sufficiente a giustificare la gestione *on premises* di un datacenter
- Oggigiorno, il *cloud computing* permette di affrontare con maggior efficienza molte tipologie di progetti
  - piccoli o con fattori di utilizzo previsto abbastanza lontani dal 100%
    - intendendo l'utilizzo medio rispetto alla capacità di picco sulla quale verrebbe dimensionato l'acquisto
    - caso tipico: workload fortemente stagionali o concentrati in ore del giorno
  - medi e grandi al punto da rendere difficoltoso il forte investimento in conto capitale
  - con aspettative di forte crescita, ma senza certezze dei tempi in cui si concretizzerà

Cloud computing: concetti di base

- Un cloud provider si fa carico della realizzazione di un (gruppo di) data center allo stato dell'arte
  - realizza gli edifici
  - predispone gli impianti
  - acquista ingenti quantità di apparati di calcolo e networking di diverse fasce
- Il pool di risorse complessive viene utilizzato per far funzionare sistemi virtualizzati
  - molti clienti condividono le risorse fisiche (multi-tenancy) spalmando i costi fissi e delegando completamente la loro amministrazione
  - la configurazione è tramite interfacce che nascondono completamente la struttura fisica
  - **provisioning dinamico**: l'avvio e arresto delle risorse è *on demand*
    - il pagamento è solo per il periodo di effettivo utilizzo
  - **scalabilità**: la dimensione del provider tipicamente dà l'illusione al cliente di poter allocare illimitatamente nuove risorse al bisogno

→ **RISORSE "As A Service"**

Livelli e attori

tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

- *aaS = Everything as a Service
- SaaS – Software as a Service
  - Le risorse sono applicazioni rese disponibili via web agli utenti
    - Gmail, Dropbox, SalesForce, Evernote, …
- PaaS – Platform as a Service
  - Le risorse sono intere piattaforme disponibili per l'esecuzione remota di codice caricato dall'utente
    - web hosting con vari linguaggi server side, cms estendibili, …
- IaaS – Infrastructure as a Service
  - Le risorse sono componenti architetturali virtualizzate
    - hardware per calcolo
    - sistemi operativi
    - dispositivi di networking

Livelli e attori

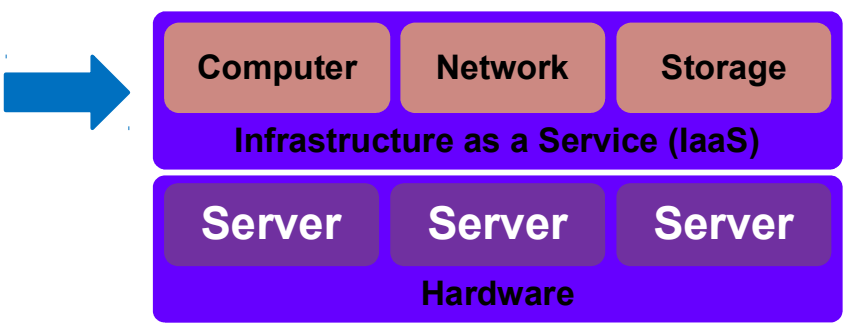
tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

- alla base di tutto, l'architettura reale

Livelli e attori

tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

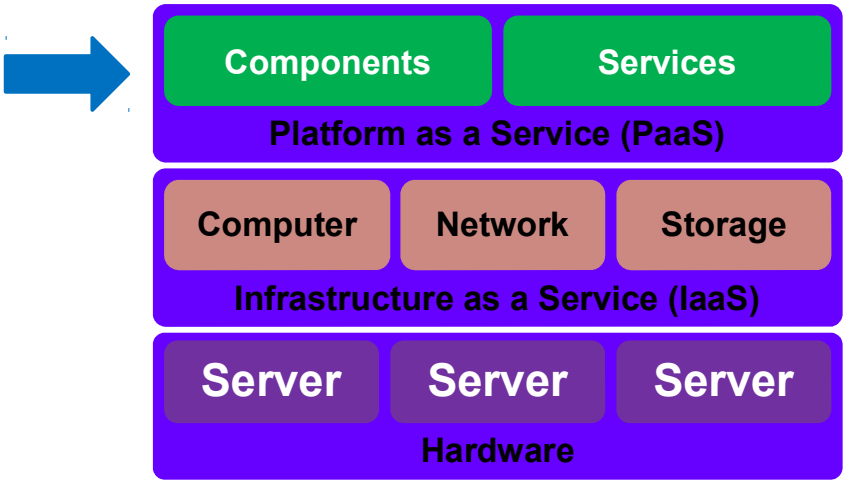
- Lo strato infrastrutturale abilita la realizzazione dei servizi cloud, per mezzo della gestione della virtualizzione



Livelli e attori

tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

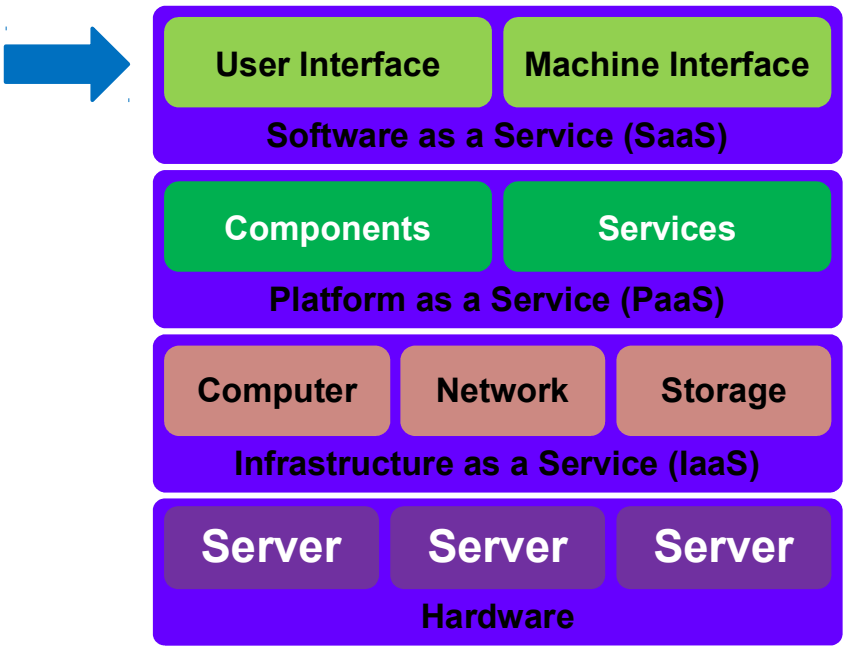
- Lo strato di piattaforma fornisce servizi standard e componenti modulari fruibili da remoto agli strati superiori



Livelli e attori

tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

- Lo strato software mette a disposizione applicazioni preinstallate a cui fornire solamente configurazione e dati

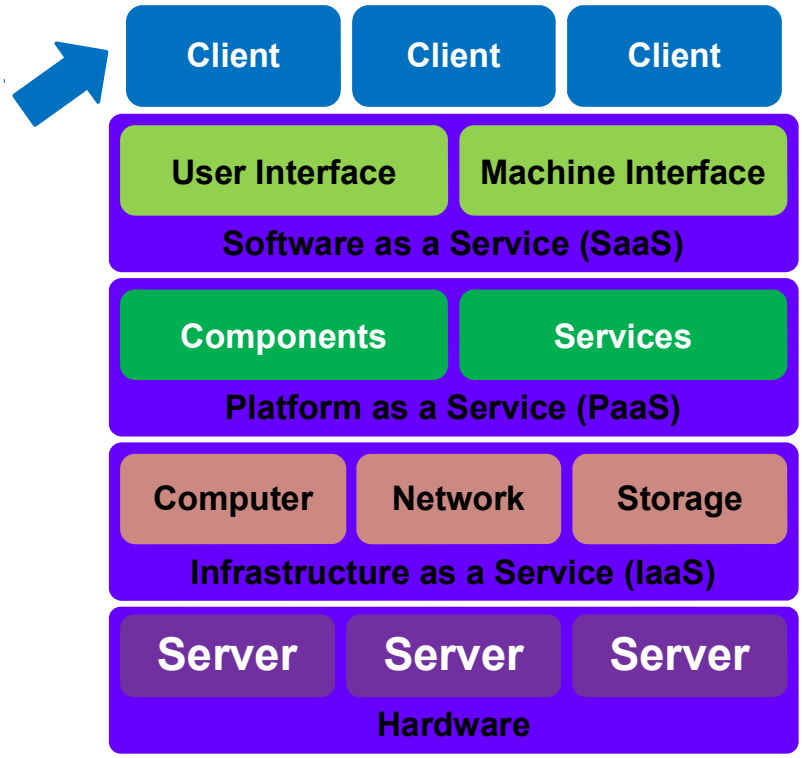


Livelli e attori

tratto da Principles, Applications and Models for Distributed Systems M – Antonio Corradi & Luca Foschini

- I client permettono di accedere al cloud. Restano l'unico componente in esecuzione sulle piattaforme fisicamente in mano all'utente, che attraverso questi può comunicare con
  - applicazioni
  - sistemi di deploy sulle piattaforme
  - sistemi di configurazione e monitoraggio delle infrastrutture

  attraverso i diversi tipi di interfaccia disponibili
  - API
  - Web GUI

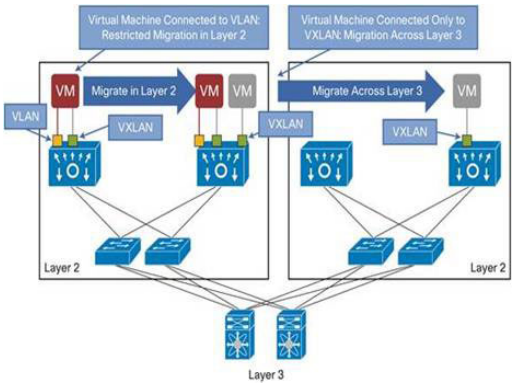


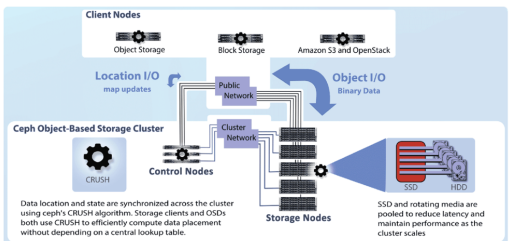
Cloud computing: prerequisiti

Virtualizzazione, virtualizzazione, virtualizzazione

- grandi pool di calcolatori
  - architetturalmente simili
  - intercambiabili
  - su cui gira un hypervisor
- apparati di rete gestibili e riconfigurabili
  - utilizzo massiccio di VLAN per partizionare il traffico tenant
  - vxlan per estendere il layer fisico su scala geografica
  - evoluzione verso Software Defined Networking
- sistemi di storage di rete
  - gerarchici (prestazioni vs costo)
  - ad alta scalabilità

vmware vSphere Xen™ Microsoft Hyper-V





Cloud computing: prerequisiti

Gestione, gestione, gestione

- interfacce al sistema
  - manuali via web
  - command line
  - integrabili in piattaforme software via API
- sistemi di monitoraggio
  - dettagliati e facilmente accessibili
  - fortemente programmabili per reagire automaticamente a eventi
- modelli di configuration management

    in un ambiente in cui i nodi di erogazione dei servizi formano un pool scalabile, non è più sufficiente saper intervenire sulla configurazione di un servizio, è necessario garantire modifiche coerenti a servizi interdipendenti e propagazione delle modifiche sulle molteplici istanze in esecuzione
  - distribuzione di parametri di configurazione
    - limitato ad aggiornamenti semplici e per i quali è necessario un effetto immediato
  - versioning e templating di file di configurazione
    - configuration as code
  - immagini immutabili
    - test → template → sostituzione graduale

Cloud computing: i protagonisti