Sistemi Web distribuiti localmente e geograficamente

Valeria Cardellini
Università di Roma Tor Vergata

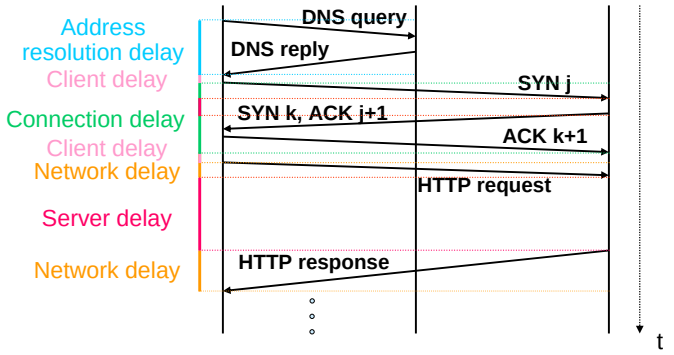
Motivazioni

- Il successo del Web
  - Siti Web popolari sono soggetti a milioni di hit al giorno
  - Ad es. sito Web delle Olimpiadi di Torino 2006
    - 71 milioni di pagine visitate e 2 milioni di utenti singoli nel giorno di picco
  - Ad es. sito Web dell'NBC per le Olimpiadi di Torino 2006
    - Traffico di picco pari a 1,6 GB al secondo
- L'evoluzione dei servizi basati sul Web
  - Servizi sempre più complessi e che richiedono generazione di contenuti dinamici e sicuri
  - Popolarità crescente di siti di social networking (es. Facebook)
    - Non offrono attualmente prestazioni soddisfacenti, risultando spesso lenti e inaccessibili (http://www.watchmouse.com/it/press.php)

Motivazioni (2)

- Maggiori aspettative da parte degli utenti
  - Regola degli 8 secondi
    - Definita nel 2001, ritenuta ancora applicabile nonostante la sempre più ampia diffusione di collegamenti Internet a banda larga (nel 2010 78% delle connessioni negli Stati Uniti)
  - Uno studio del 2006 sponsorizzato da Akamai riduce la soglia a 4 secondi per siti di e-commerce

Componenti del ritardo Web



Dov'è il collo di bottiglia?

(1) DNS? (2) Client/connessione?
(3) Rete? (4) Piattaforma server?

Potenziamenti del Web

- Azioni a livello del CLIENT
  - Possibilità di intervento limitato
- Azioni a livello di RETE
  - "Network guys are doing an excellent job"
- Azioni da parte del CONTENT/SERVICE PROVIDER
  - Replicazione dei server
  - Caching
- Azioni da parte di INTERMEDIARI
  - Caching cooperativo (istituzionale o ISP)
  - Content Delivery Network (commerciale)

Bottleneck: rete?

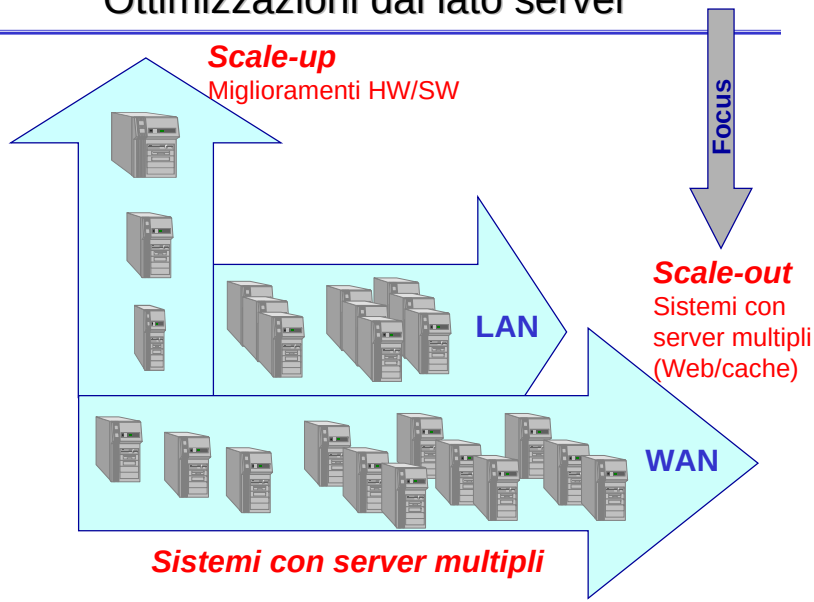
- La larghezza di banda tende ad aumentare
- Il round trip time (RTT) tende a diminuire
- L'infrastruttura di rete fissa tende a migliorare

- Ma ci sono ancora molti problemi aperti
  - Peering point
  - Router
  - QoS
  - Protocolli
  - Wireless, reti ad-hoc
  - …

Bottleneck: server?

- Quali sono i ritardi percentuali dovuti al server Web?
  - Fonte: P. Barford, M. Crovella, "Critical Path Analysis of TCP Transactions, ACM Transactions on Networking, 9(3), pp. 238-248, June 2001.
- Per risorse statiche di dimensioni piccole e per server con carico alto:
  - Fino all'80% del tempo di risposta dipende dal server
- Per risorse statiche di dimensioni medio/grandi e per server con carico alto:
  - Fino al 50% del tempo di risposta dipende dal server

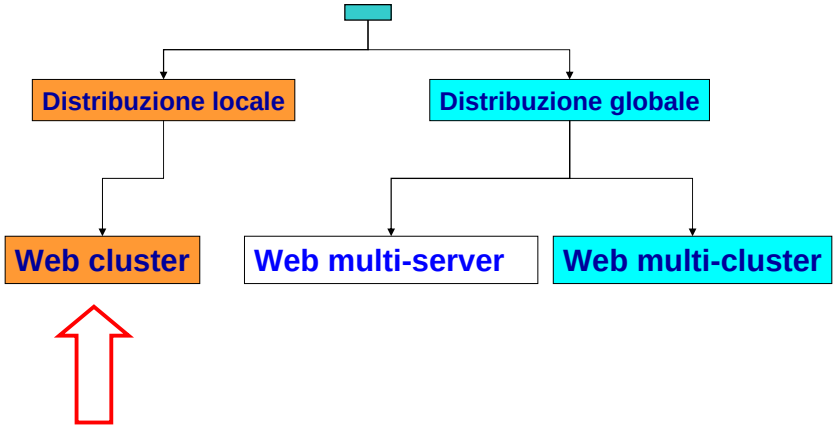
Ottimizzazioni dal lato server



Scale-up e scale-out

- Scale-up
  - Interventi a livello di SO
    - Evitare copie multiple dello stesso oggetto, politiche di scheduling diverse da round-robin (es., SRTF)
  - Modifiche del software del server Web
    - Apache 2.2, Flash, Zeus
- Scale-out
  - A livello di content/service provider
    - Distribuzione locale dei server
    - Distribuzione globale dei server
    - Integrazione con meccanismi di caching
  - A livello di intermediari
    - Caching cooperativo
    - Content Delivery Network

Sistemi Web distribuiti

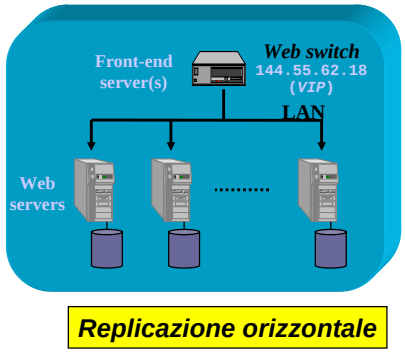


Sistemi Web distribuiti (2)

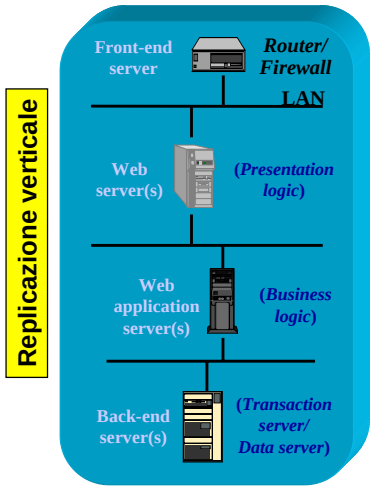
Sistemi Web scalabili basati su piattaforme con server multipli

- Un **meccanismo di routing** per indirizzare le richieste client al nodo “migliore”
- Un **algoritmo di distribuzione** (*dispatching*) per individuare il nodo “migliore”
- Un componente **esecutore** per eseguire l’algoritmo di distribuzione utilizzando il relativo meccanismo di routing

Architetture per Web cluster



Replicazione orizzontale



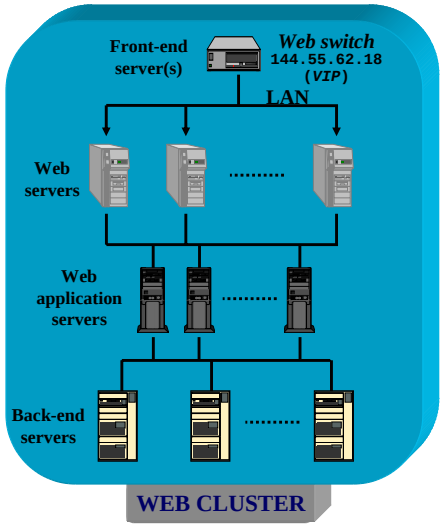
Architetture per Web cluster:
una miriade di tecnologie (*complesse*)

Network/OS technology

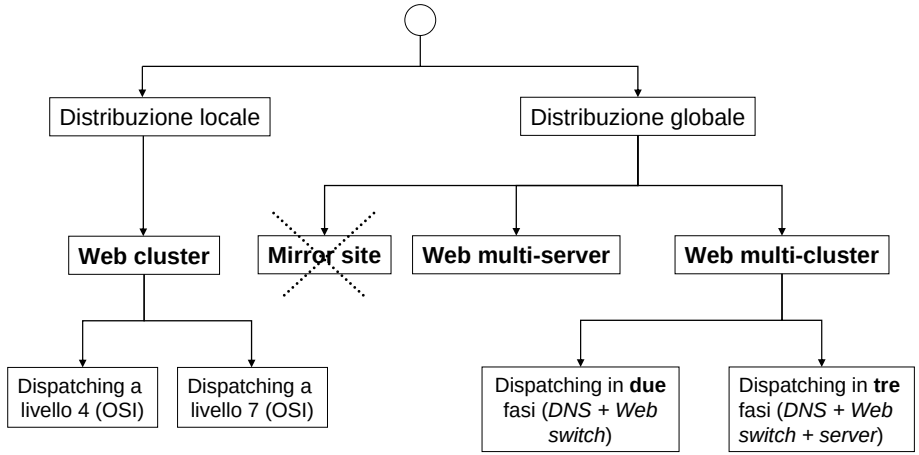
Web server technology

Middleware technology

Database technology



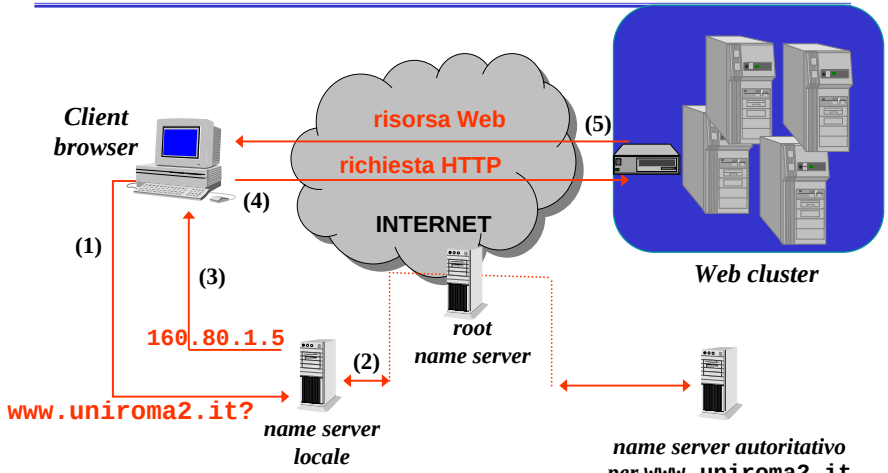
Sistemi Web distribuiti



Web cluster

- Sito Web implementato su di un'architettura parallela o distribuita localmente
- Indirizzi sito Web
  - Un solo hostname (es., "www.uniroma2.it")
  - Un solo indirizzo IP (***virtual IP address*** *o VIP*)
- Web server con indirizzi IP "mascherati" all'esterno
- ***Web switch*** (*dispatcher*) il cui indirizzo IP è l'indirizzo IP del sito Web (VIP)

Richiesta HTTP ad un Web cluster

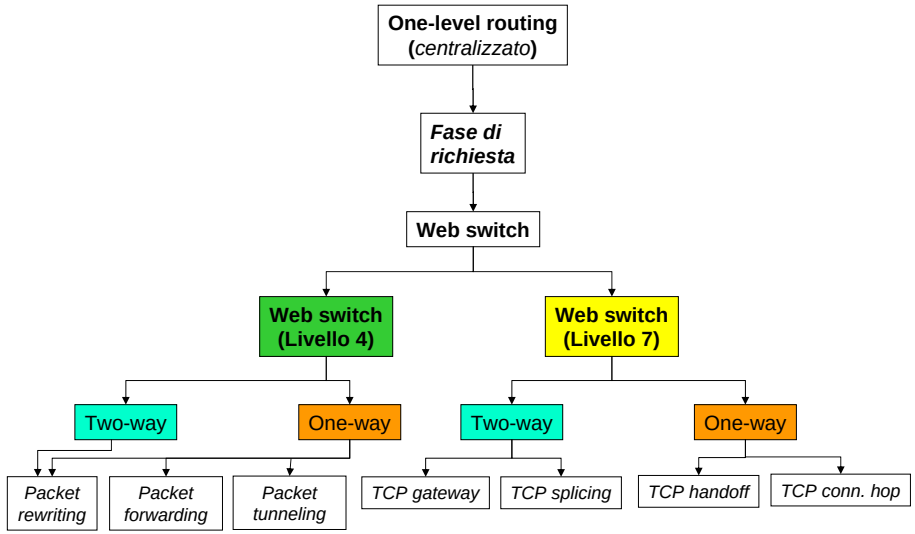


Due fattori da considerare:
- Caratteristiche del Web switch
- Flusso dei pacchetti di risposta

Web switch del cluster

- Componente di rete con ruolo di dispatcher
  - Mapping da VIP ad indirizzo IP di un server
  - Distribuzione delle richieste a granularità fine (i pacchetti entranti con indirizzo VIP sono indirizzati dal Web switch)
  - Implementazioni alternative
    - dispositivo hardware special-purpose
    - modulo software eseguito a livello kernel (SO special-purpose)
    - modulo software eseguito a livello applicativo (SO general-purpose)
- Architetture alternative:
  - **Web switch di livello 4** (*content information blind*)
    - sorgente e destinazione indirizzo IP, numeri di porta TCP, SYN/FIN bit nell'header TCP
  - **Web switch di livello 7** (*content information aware*)
    - URL, cookie, altri header HTTP, SSL id

Architetture per Web cluster



Architetture per Web cluster (2)

Classificazione delle architetture basata su

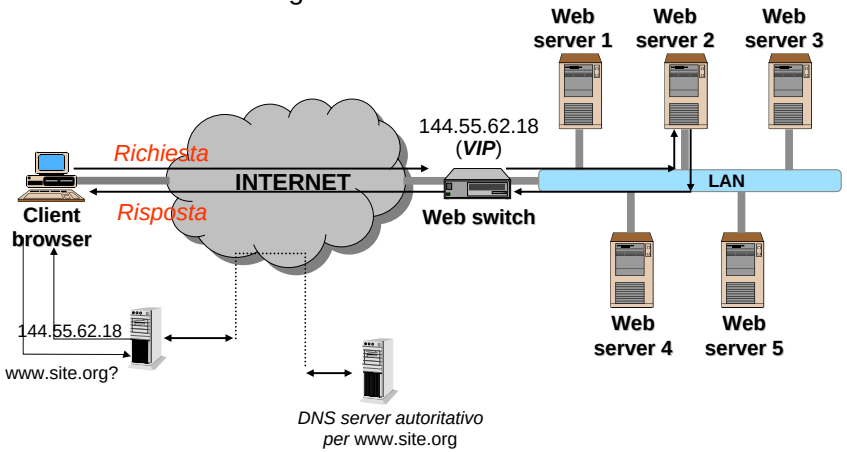
- ***Livello*** dello stack OSI a cui opera il Web switch
- ***Percorso*** seguito dai pacchetti
  - I pacchetti in ingresso (*inbound*) passano sempre dallo switch
  - I pacchetti in uscita (*outbound*)
    - Passano anche dallo switch: **architetture two-way**
    - Transitano attraverso un'altra connessione: **architetture one-way**
- ***Meccanismo*** di routing utilizzato dal Web switch per re-indirizzare i pacchetti inbound verso i server
    - Ad esempio: packet rewriting, TCP handoff

Web switch di livello 4 (L4)

- Opera a livello TCP/IP
- Gestione della connessione TCP
  - Pacchetti appartenenti alla stessa connessione TCP devono essere assegnati allo stesso server
  - Il Web switch utilizza una **binding table** per la gestione delle connessioni TCP attive
  - Il Web switch esamina l'header di ogni pacchetto:
    - nuova connessione (*SYN*) ➡ *assegnamento del server*
    - connessione esistente ➡ *ricerca nella binding table*

Architetture di livello 4 two-way

- Packet double-rewriting



Architetture di livello 4 two-way (2)

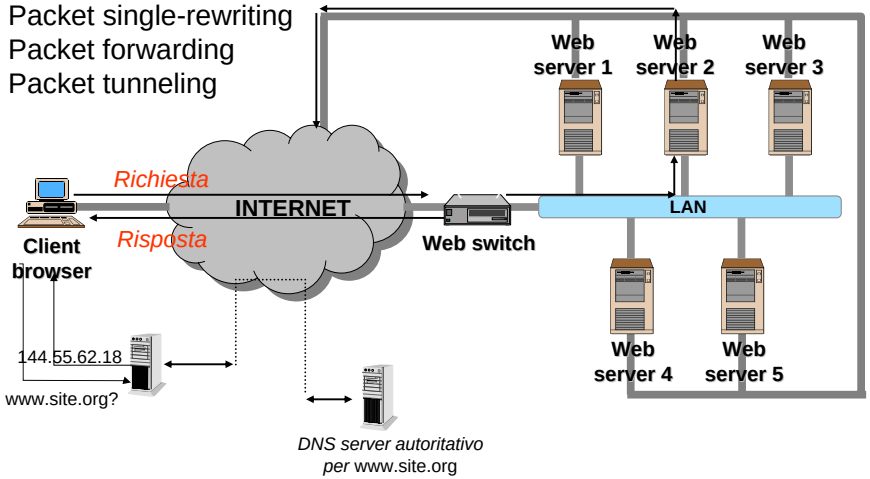
Ogni server ha il suo indirizzo IP privato

- I pacchetti in uscita devono riattraversare il Web switch
- Web switch modifica dinamicamente sia i pacchetti entranti sia quelli uscenti
  - Indirizzo IP *destinazione* dei pacchetti entranti (***VIP → IP server***)
  - Indirizzo IP *sorgente* dei pacchetti uscenti (***IP server → VIP***)
  - Ricalcolo dei checksum IP e TCP

Tecnica basata sul meccanismo di Network Address Translation (NAT)

Architetture di livello 4 one-way

- Packet single-rewriting
- Packet forwarding
- Packet tunneling



Architetture di livello 4 one-way (2)

Packet single-rewriting

- Lo switch modifica solo i pacchetti IP entranti
- Il server modifica i pacchetti IP in uscita (IP server → VIP)

Packet forwarding

- Non c'è modifica dei pacchetti IP entranti ed uscenti: i pacchetti sono inoltrati a livello MAC (***ri-indirizzamento del frame MAC***)
- *PRO*: minor overhead sullo switch per pacchetto
- *CONTRO*: Web switch e server devono trovarsi sulla stessa sottorete fisica

Packet tunneling

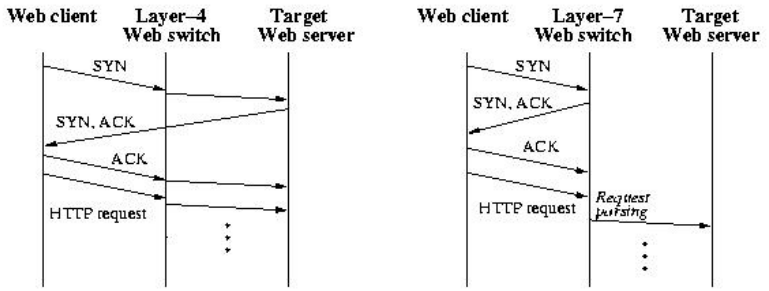
- Il pacchetto IP originale è incapsulato dallo switch in un altro pacchetto IP, il cui header contiene: VIP come indirizzo IP sorgente e indirizzo server come indirizzo IP destinatario

Web switch di livello 7 (L7)

- Il Web switch opera a livello applicativo
- Il Web switch deve stabilire la connessione TCP con il client ed attendere la richiesta HTTP
- Ispeziona il contenuto della richiesta HTTP per decidere a quale server inoltrarla
  - Parsing della linea di richiesta e degli header HTTP
  - Gestione dei pacchetti inbound (ACK)

- Principali caratteristiche del content-based routing
  - Consente il partizionamento dei contenuti/servizi del sito Web tra diversi server (eventualmente specializzati)
  - Favorisce l'utilizzo di meccanismi di caching
  - Supporta il dispatching a granularità fine delle richieste HTTP effettuate tramite connessioni persistenti

Decisione sull'assegnamento a livello 4 e 7

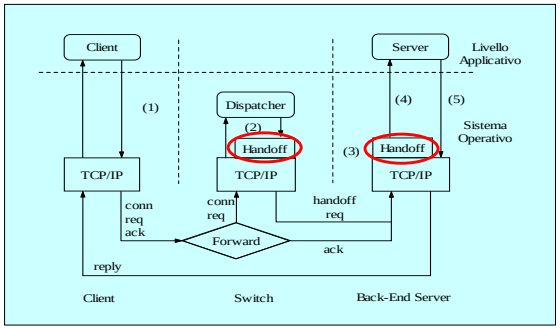


Architetture di livello 7 two-way

- TCP gateway (livello applicativo)
  - Il Web switch è realizzato mediante un proxy
    - Il forwarding dei dati è realizzato a livello applicativo
  - Overhead elevato
    - Ogni richiesta attraversa sullo switch tutto lo stack di protocolli (data link ↔ application ↔ data link)
- TCP splicing (livello di SO)
  - Ottimizzazione del TCP gateway
    - Il forwarding dei dati è realizzato a livello TCP
  - Il primo pacchetto determina la scelta del server e l'instaurazione della connessione persistente fra il Web switch ed il server scelto
  - I pacchetti successivi sono trasmessi dal Web switch a livello TCP
  - Richiede modifiche del kernel del SO del Web switch

Architetture di livello 7 one-way

- TCP handoff (livello di SO)
  - La connessione TCP viene stabilita con il Web switch; il Web switch passa (handoff) la connessione al server, che gestisce il servizio ed invia direttamente la risposta al client
  - Richiede modifiche del kernel dei SO del Web switch e dei server



Algoritmi di distribuzione

- Due classi di algoritmi
  - **Statici** (stateless)
  - **Dinamici** (state aware)
    - Informazioni sui client (*client info aware*)
    - Informazioni sullo stato dei server (*server state aware*)
    - Informazioni sui client e sullo stato dei server (*client info & server state aware*)

Algoritmi statici *vs.* dinamici

Statici	Dinamici
• Facile implementazione • Overhead trascurabile (sullo switch) • Possibili situazioni di sbilanciamento del carico	• Implementazione più complessa • Overhead di comunicazione (tra switch e server) e di computazione (sullo switch) • Miglior bilanciamento del carico a parità di politica adottata

Confronto meccanismi di distribuzione

- Livello 4
  - Livello connessione TCP
  - Algoritmi di distribuzione *content blind*
  - Algoritmi statici e dinamici
- Livello 7
  - Livello connessione applicativa (HTTP)
  - Algoritmi di distribuzione *content aware*
  - Algoritmi dinamici
    - Almeno client info aware (occorre usare l'informazione contenuta nella richiesta del client!)

Distribuzione a livello 4

- Non servono algoritmi particolarmente sofisticati
- Esempi di algoritmi statici:
  - Random, Round Robin (assegnamento circolare)
- Esempi di algoritmi dinamici server state-aware:
  - Attuano una distribuzione delle richieste in base allo stato di carico dei server
  - Least loaded, Weighted Round Robin
- Gli algoritmi statici forniscono prestazioni confrontabili a quelle di algoritmi dinamici nel caso di richieste/servizi Web che rientrano in intervalli temporali di 2 ordini di grandezza
- Oltre i 2 ordini di grandezza, è opportuno utilizzare algoritmi dinamici (client o preferibilmente server state aware)

Distribuzione a livello 7

- Algoritmi di distribuzione client info aware
  - Esempi: identificatore di sessione, partizionamento del contenuto, CAP
- Algoritmi di distribuzione client info & server state aware
  - Esempio: LARD

Algoritmi L7 client info aware

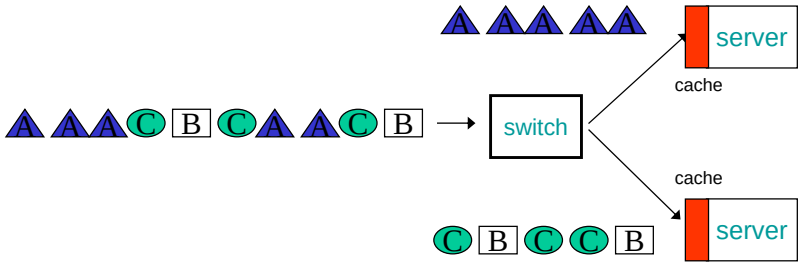
- Identificatori di sessione
  - Richieste HTTP con stesso *SSL id* o stesso *cookie* assegnate allo stesso server
- Partizionamento del contenuto tra i server
  - In base al *tipo di risorsa* (HTML, immagini, contenuto dinamico, audio, video, …)
    - Obiettivo: utilizzare server specializzati per contenuti differenti
  - In base alla *dimensione della risorsa*
    - Obiettivo: aumentare la condivisione del carico
    - Soglie di partizionamento determinate staticamente o dinamicamente
    - Difficoltà: dimensione nota solo per risorse statiche, ma da stimare per risorse dinamiche
  - In base ad una *funzione hash* applicata sul path della risorsa
    - Obiettivo: aumentare il *cache hit rate* nei server Web

Algoritmi L7 client info aware (2)

- Content Aware Policy (CAP)
  - Sfrutta informazioni relative alla tipologia della richiesta da assegnare
  - Richiede un meccanismo di classificazione *dinamica* delle richieste effettuabile in base al tipo di servizio (individuabile dall'URL)
    - *CPU-bound* (es., crittografia)
    - *Disk-bound* (query a database)
    - *Network-bound* (download di file di grandi dimensioni)
  - Obiettivo: ripartire le richieste *CPU/disk/network-bound* tra tutti i server in modo da condividere il carico
    - Assegnamento round-robin in base al tipo di servizio
  - Riferimento:
    - E. Casalicchio, M. Colajanni, "A client-aware dispatching algorithm for Web clusters providing multiple services", WWW 2001.

Algoritmi L7 client info & server state aware

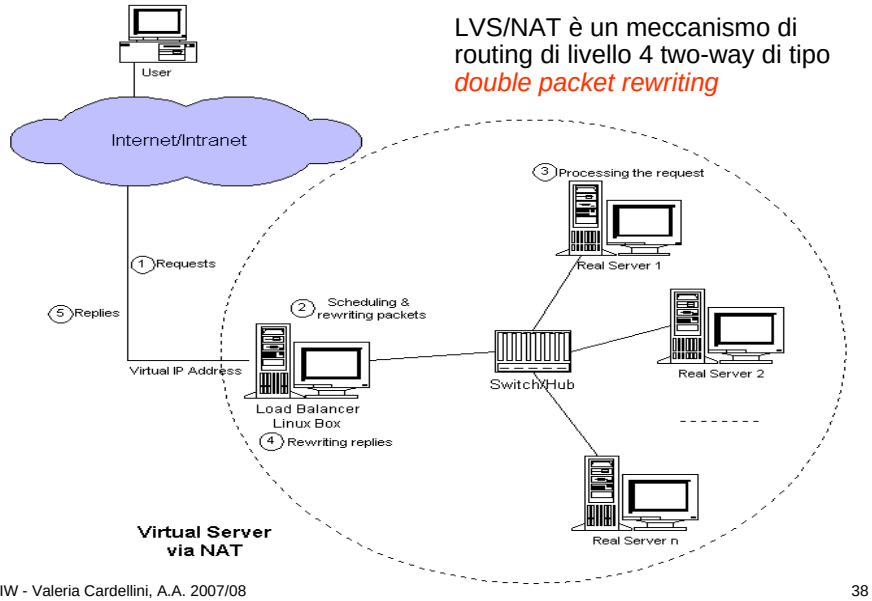
- Locality Aware Request Distribution (LARD)
  - Considera sia il tipo di richiesta/servizio sia lo stato di carico dei server Web
  - Ha l'ulteriore obiettivo di aumentare il *cache hit rate* dei server Web
  - Riferimento:
    - V.S. Pai *et al.*, "Locality-aware request distribution in cluster-based network servers", ASPLOS 1998.



Linux Virtual Server (LVS)

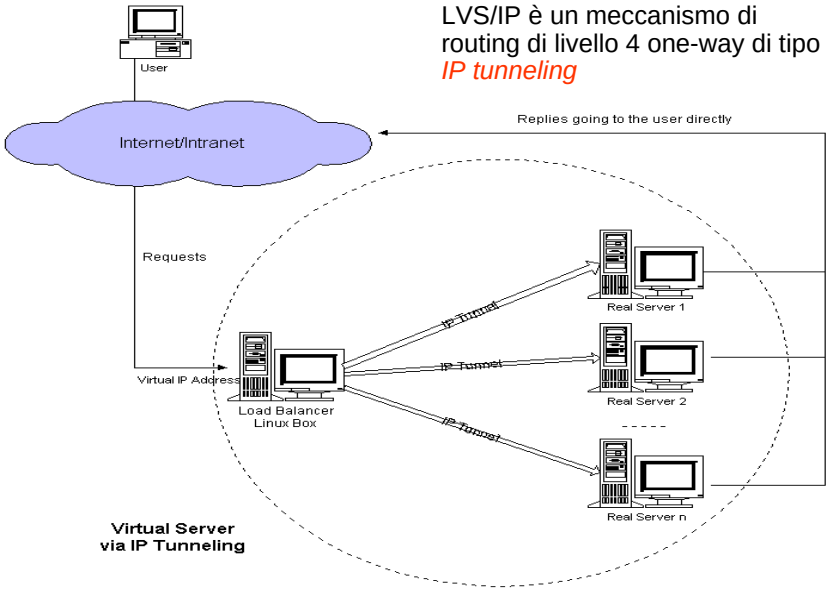
- Software open-source per realizzare Web cluster con Web switch operante a livello 4
  - http://www.linuxvirtualserver.org/
  - Modulo IPVS nel kernel 2.6 di Linux
  - Per switching a livello 7, ci sono i sottoprogetti (sperimentali):
    - KTCPVS (Kernel TCP Virtual Server): livello applicativo nel kernel
    - TCPHA: TCP handoff
    - TCPSP: TCP splicing
- Caratteristiche salienti
  - Scalabilità
    - Possibilità di aggiungere e rimuovere dinamicamente i nodi dal cluster
  - Elevata disponibilità
    - Meccanismi per riconfigurazione dinamica del sistema e per riconoscimento di failure dei nodi (Ultra Monkey, Red Hat Piranha, High Availability Linux…)
  - Configurazione LAN e WAN

LVS/NAT: Network Address Translation



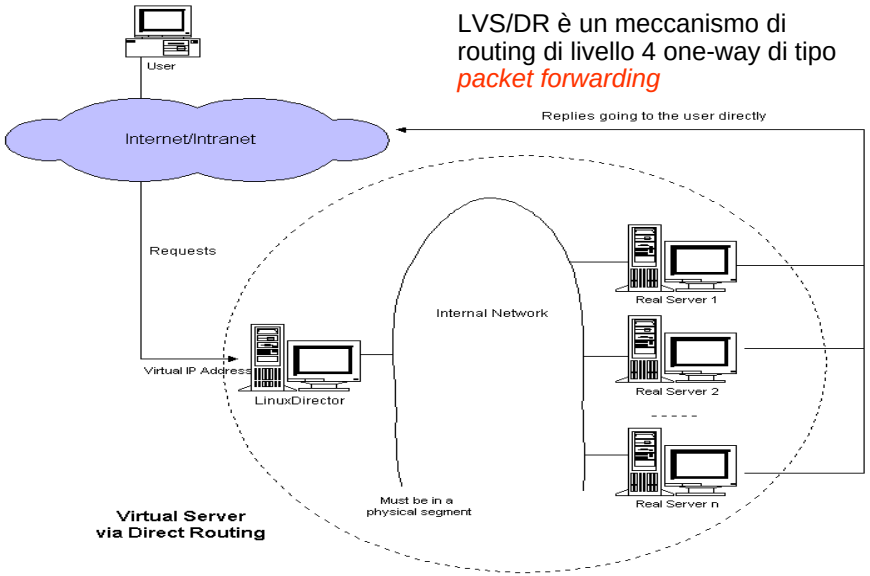
LVS/NAT è un meccanismo di routing di livello 4 two-way di tipo *double packet rewriting*

LVS/IP: IP tunneling



LVS/IP è un meccanismo di routing di livello 4 one-way di tipo *IP tunneling*

LVS/DR: Direct Routing



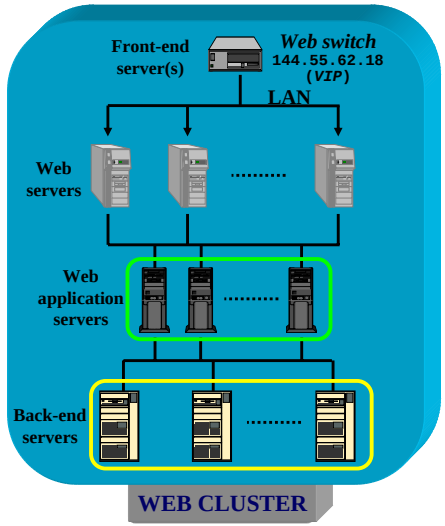
LVS/DR è un meccanismo di routing di livello 4 one-way di tipo *packet forwarding*

Altri prodotti per Web cluster

- Citrix NetScaler
- Cisco Application Control Engine
- F5 Networks' BIG-IP Local Traffic Manager
- Foundry Networks ServerIron
- Nortel Application Switches
- Radware AppDirector
- Resonate Central Dispatch
- Zeus Extensible Traffic Manager

Oltre il front-end tier…

- Fino ad ora abbiamo analizzato la replicazione orizzontale nel livello front-end (Web server)
- La replicazione orizzontale può essere attuata anche nei livelli più interni:
  - middle tier, composto da application server
  - back-end tier, composto da database server

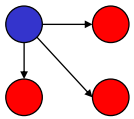


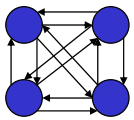
Replicazione del middle tier

- Obiettivo del dispatching per il middle tier: scegliere l'application server
- Granularità del dispatching: intera richiesta
- Dispatching attuato
  - da un'entità *centralizzata* interposta tra front-end e middle tier
  - oppure in modo *distribuito* da ciascun server Web
- Dispatching implementato in molti prodotti commerciali usando semplici politiche di distribuzione (varianti di round-robin, weighted round-robin, least loaded)
- Ad esempio, per Apache e Tomcat:
  - Connettore JK
  - Dispatching tramite mod_proxy di Apache

Replicazione del back-end tier

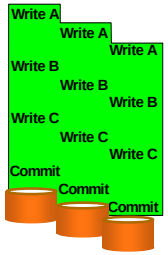
- Il DB (o più in generale l'applicazione di back-end) può consentire di essere eseguito su più nodi
  - Replicazione del DB (completa o parziale) su più repliche identiche
  - La distribuzione è di solito trasparente al middle tier (non vi è dispatching esplicito)
- Problema 1: come gestire l'aggiornamento delle repliche
  - *Master singolo* (anche *primary copy*)
    - Lettura su tutti i DB server
    - Scrittura solo su master singolo e poi replicazione dell'aggiornamento sugli slave
  - *Master multipli* (anche *update everywhere*)
    - Lettura su tutti i DB server
    - Scrittura su un master e poi replicazione dell'aggiornamento sugli altri DB server





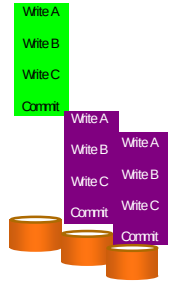
Replicazione del back-end tier (2)

- Problema 2: come mantenere la consistenza dei dati nel DB replicato
  - Replicazione **eager** oppure replicazione **lazy**
- Replicazione **eager** (sincrona o pessimistica): immediata, tutte le repliche sono aggiornate *prima del commit* della transazione
  - Vantaggio: no anomalie di concorrenza
  - Svantaggio: prestazioni ridotte per le operazioni di scrittura, maggiore traffico per propagare gli aggiornamenti



Replicazione del back-end tier (3)

- Replicazione **lazy** (asincrona o ottimistica): ritardata, *dopo il commit* della transazione
  - Possibile inconsistenza
  - Strategie di riconciliazione per gestire la possibile inconsistenza tra le repliche



  - Per approfondimenti: J. Gray, P. Helland, P.E. O'Neil, and D. Shasha, "The dangers of replication and a solution", ACM SIGMOD 1996.

Replicazione del back-end tier (4)

- Per incrementare ulteriormente le prestazioni del cluster, è possibile integrare la replicazione del back-end tier con meccanismi di caching dei risultati delle query
  - Ad es. Oracle Database Cache
- In alternativa alla replicazione del DB, si può usare un middleware per gestire un RAIDb (Redundant Array of Inexpensive Databases)
  - Ad es. Sequoia http://sequoia.continuent.org/ (open source)

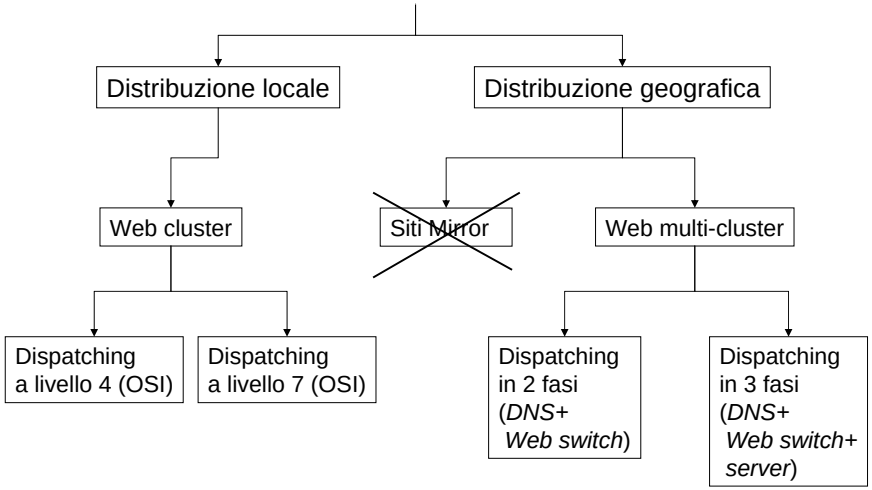
Sommario caratteristiche Web cluster

- Architetture alternative (*front-end tier*)
  - Web switch livello 4 vs. Web switch livello 7
  - One-way vs. two-way
- Principali vantaggi
  - Controllo a granularità fine sull'assegnamento delle richieste
  - Elevata affidabilità (*disponibilità*, *sicurezza*)
- Principali svantaggi
  - Presenza di single point of failure (il Web switch)
  - Scalabilità limitata dal Web switch
  - Scalabilità limitata dalla banda di accesso ad Internet (ad es. T3 ≈ 45 Mbps)
- Soluzione
  - Replica su **scala geografica (global scale-out)**

Delivery su scala geografica

- Il content/service provider ha due possibilità per distribuire i propri contenuti/servizi su scala geografica in modo efficiente e scalabile:
  - Il provider possiede e gestisce l'intera piattaforma (**Sistemi Web distribuiti geograficamente**)
  - Il provider gestisce solo i contenuti/servizi ma delega ad una terza parte il servizio di delivery dei contenuti/servizi agli utenti finali (**Content Delivery Network**)

Sistemi con server Web multipli



Sistemi Web distribuiti geograficamente

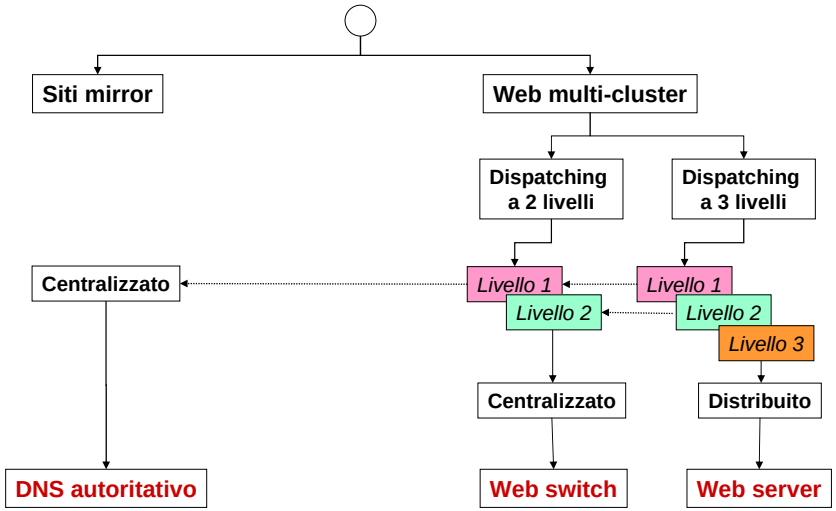
- Problemi di rete per sistemi Web distribuiti localmente
  - Scalabilità limitata dalla banda di accesso ad Internet
  - Incapacità di evitare i link di rete congestionati
  - Affidabilità della rete

- Scale-out globale
  - Maggiore complessità dell'architettura
    - Meccanismi di routing ed algoritmi di distribuzione
    - Difficoltà nella gestione dell'infrastruttura
  - Metrica per la selezione del cluster "migliore"
  - Localizzazione e posizionamento dei cluster

Web multi-cluster

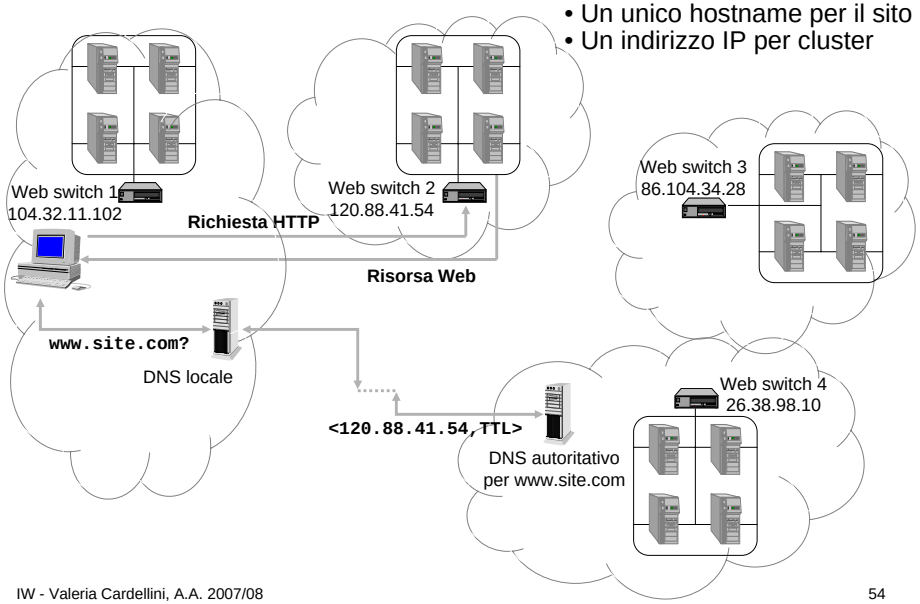
- Sito Web implementato su di un'architettura di Web cluster distribuiti geograficamente tra diverse regioni Internet
- Meccanismo di routing delle richieste basato sul DNS
  - Indirizzi del sito Web
    - Un unico hostname al quale corrisponde *molteplici indirizzi IP*, tanti quanti sono i Web cluster
    - L'indirizzo IP fornito dal DNS autoritativo corrisponde al *VIP dello switch* del cluster selezionato
- Il DNS autoritativo del sito Web seleziona un cluster nella *fase di address lookup* mediante un algoritmo di tipo:
  - Round-robin
  - Prossimità geografica
  - Carico dei cluster
  - Prossimità geografica e carico dei cluster
  - Altro ...

Sistemi Web geograficamente distribuiti



Web multi-cluster (2 livelli)

- Un unico hostname per il sito
- Un indirizzo IP per cluster



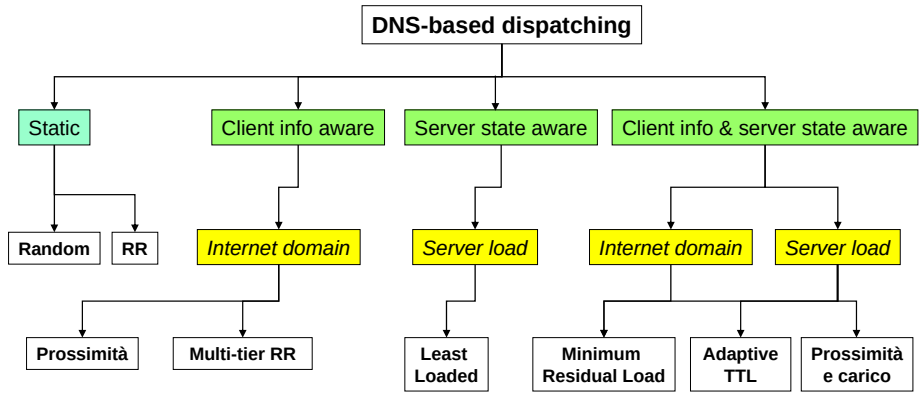
Dispatching di primo livello (mediante DNS)

- Il primo livello di distribuzione geografico avviene nella fase di *risoluzione dell'indirizzo* (address lookup):
  - il client richiede l'**indirizzo IP** del cluster corrispondente all'hostname indicato nell'URL
  - se l'hostname è valido, il client riceve la coppia

    < Indirizzo IP, Time-To-Live>

    da:
    - cache di qualche name server locale o intermedio
    - oppure **DNS autoritativo del sito**, opportunamente modificato (integrato o meno da altro componente). Può applicare diverse politiche di dispatching per selezionare il Web cluster "migliore"

Algoritmi di dispatching per DNS



Prossimità in Internet

- La prossimità in Internet è un problema interessante: la prossimità geografica tra client e server **non implica** *prossimità Internet* (*round trip latency*)
- Valutazione *statica* della prossimità
  - indirizzo IP del client per determinare la zona Internet (simile a distanza geografica)
  - numero di hop (informazione "stabile" più che "statica")
    - *network* hops (e.g., `traceroute`)
    - *Autonomous System* hops (query delle tabelle di routing)

Non garantisce la selezione del cluster migliore, e.g., "*links are not created equal*"

Prossimità in Internet (2)

- Valutazione *dinamica* della prossimità
  - round trip time (es., **`ping`**, **`tcping`**)
  - bandwidth disponibile (es., **`cprobe`**)
  - latenza delle richieste HTTP (es., *request emulation*)

Tempo aggiuntivo e costi di traffico per la valutazione

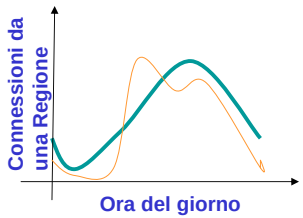
- Un problema ancora aperto: *correlazione tra numero di hop round trip time*?
  - Misure "vecchie" (1995): prossima a zero
  - Misure "recenti" (dal 1999): elevata, mediamente elevata

Problemi del dispatching geografico

(di cui le politiche di dispatching devono tener conto)

Tipici problemi del dispatching Web

- Picchi di carico in alcune ore/giorni



Problemi aggiuntivi

- Traffico dipendente dai fusi orari
- Distribuzione non uniforme dei client tra le regioni Internet
- Prossimità Internet tra client e Web cluster

- (Per DNS) Caching di [hostname-indirizzo IP] in name server intermedi per l'intervallo del Time-To-Live

Problemi del dispatching tramite DNS

- Nel caso di siti Web molto popolari, il DNS autoritativo controlla solo il 5% del traffico in arrivo al sito
  - A causa del caching hostname-indirizzo IP nei name server locali e intermedi
- A differenza del Web switch (che controlla il 100% del traffico in arrivo al sito), il DNS autoritativo deve utilizzare algoritmi sofisticati (es., *TTL-adattativi*)
- Non sono stati trovati (*esistono?*) algoritmi di dispatching DNS in grado di evitare episodi di sovraccarico per tutte le classi di workload

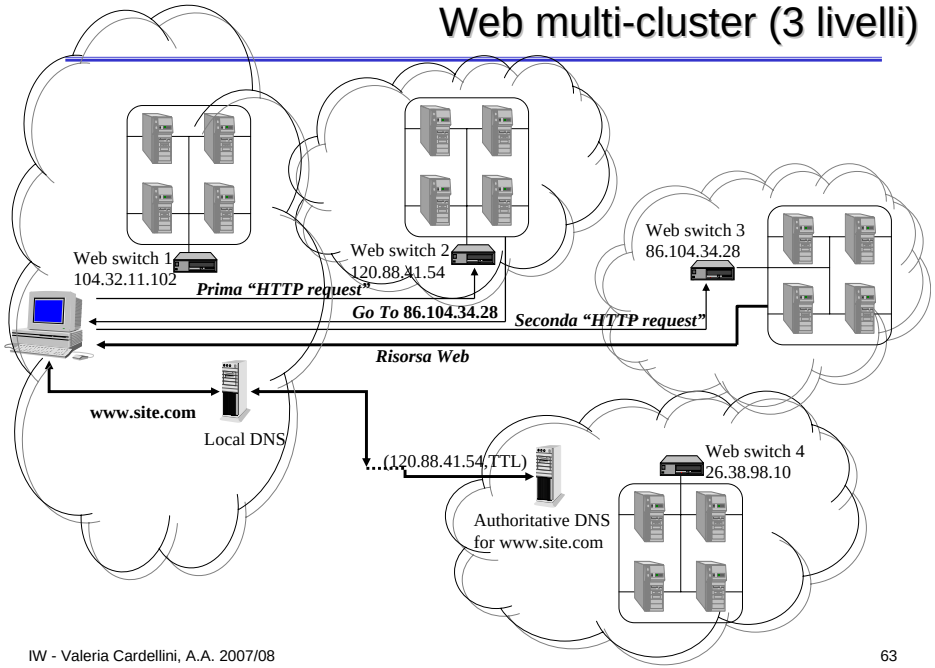
Come risolvere i problemi del dispatching DNS

- Fase di ***lookup***: integrare il dispatching attuato dal DNS autoritativo con quello effettuata da un'altra entità
- Fase di ***richiesta***: integrare il dispatching *centralizzato* attuato dal DNS autoritativo con dispatching *distribuito* da parte dei server
  - Alcuni meccanismi di re-routing delle richieste:
    - Ridirezione HTTP
    - IP tunneling
    - URL rewriting

Dispatching per Web multi-cluster

- Indirizzi del sito Web visibili
  - Un unico hostname (ad es., "www.site.com")
  - Un indirizzo IP per ogni Web cluster
- Livelli multipli di routing e dispatching:
  - **DNS** autoritativo seleziona il "Web cluster migliore" (dispatching *inter-cluster*)
  - **Web switch** del cluster seleziona il "Web server migliore" (dispatching *intra-cluster*)
  - Ciascun **Web server** (o Web switch di livello 7) può ridirigere le richieste verso un altro Web cluster, ad es. per risolvere situazioni temporanee di sovraccarico (dispatching *inter-cluster*)
  - Per semplicità non consideriamo gli ulteriori livelli di dispatching interni al cluster…

Web multi-cluster (3 livelli)



Motivazioni per terzo livello di dispatching

Web multi-cluster con **due** livelli di dispatching

- Controllo elevato sul carico che raggiunge il Web cluster *(buon bilanciamento intra-cluster)*
- Reazione lenta ad un cluster sovraccarico *(cattivo bilanciamento inter-cluster)*

Web multi-cluster con **tre** livelli di dispatching:
Reazione *immediata* per spostare il carico da un Web cluster sovraccarico (meglio redirezione HTTP di IP tunneling)

Ridirezione HTTP

- Il meccanismo di ridirezione è parte del protocollo HTTP ed è supportato dagli attuali browser
- **DNS** e **Web switch**: usano politiche di dispatching *centralizzate*
- **Ridirezione**: usa politiche di dispatching *distribuite*, in cui tutti i server Web possono partecipare al (ri)assegnamento delle richieste
- La ridirezione è completamente trasparente per l'utente (non per il client!)

message header
HTTP status code
302 - "Moved temporarily" to a new location

- "New location"
  - ridirezione ad un indirizzo IP (**prestazioni migliori**)
  - ridirezione ad un hostname

Ridirezione HTTP: pro e contro

PRO

- Compatibile con tutti i client e server Web (implementata a livello applicativo)
- Meccanismo distribuito che soddisfa requisiti di affidabilità (non introduce "single point of failure")
- Distribuzione delle richieste *content-aware*

CONTRO

- Limitata alla ridirezione di richieste HTTP (meccanismi di ridirezione più generali, ad es. IP tunneling)
- Aumenta il traffico in quanto ogni richiesta ridiretta richiede una nuova connessione TCP

Tuttavia, la ridirezione **riduce** il tempo di risposta quando
impatto del server > impatto della rete

La piattaforma di Google

- Centinaia di migliaia di macchine (stima 450000) organizzate in 8 cluster distribuiti geograficamente
  - Nel 2002 Google usava 6000 processori e 12000 dischi, con 2 siti nella Silicon valley e 2 in Virginia
    - Ciascun sito connesso ad Internet tramite una connessione OC48 (2488 Mbit/sec)
- L'indice di Google comprende oltre 25 miliardi di URL
- Obiettivo: servire una richiesta in meno di 0,5 secondi (ritardi di rete compresi!)
- Distribuzione delle richieste tra i cluster a livello di DNS
  - Carico dei cluster e prossimità rispetto al client
- I server sono commodity PC con una versione customized di Linux e di Apache (denominata GWS)
- Riferimenti
  - http://research.google.com/pubs/papers.html