ISSN 2039-7941
Anno 2016_Numero 353

rapporti tecnici INGV

Progettazione di un'architettura cluster High Availability per il portale web dell'Osservatorio Etneo

353

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - Referente
Rossella Celi
Tel. +39 06 51860068
redazionecen@ingv.it

in collaborazione con:
Barbara Angioni (RM1)

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

Rappresentante legale: Carlo DOGLIONI
Sede: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2016_Numero 353

rapporti tecnici INGV

PROGETTAZIONE DI UN'ARCHITETTURA CLUSTER HIGH AVAILABILITY PER IL PORTALE WEB DELL'OSSERVATORIO ETNEO

Marco Stefano Scroppo[1], Salvatore Mangiagli[2], Marco Aliotta[2], Carmelo Cassisi[2], Marcello D'Agostino[2], Orazio Torrisi[2]

[1]**Università degli Studi di Catania** (Dipartimento di Ingegneria Elettrica, Elettronica e Informatica)
[2]**INGV** (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Catania - Osservatorio Etneo)

353

Indice

1. Introduzione

Al fine di garantire la scalabilità dei servizi offerti dai portali web, in risposta al costante aumento del volume di traffico sulla rete Internet, la progettazione dei sistemi informativi per la fornitura dei servizi di rete ha visto un massiccio utilizzo di approcci distribuiti, sfruttando differenti tecnologie tra cui quelle introdotte dalle piattaforme di virtualizzazione di calcolo e per l'alta disponibilità. L'approccio tradizionale, che prevede l'allocazione dei servizi e delle risorse da condividere in un unico nodo che svolge il compito di soddisfare tutte le richieste al sistema, implica una serie di limitazioni quali:

- *single point of failure*: ad un guasto del nodo corrisponde un'indisponibilità del servizio;
    - sistema non scalabile: le prestazioni decrescono con l'aumentare delle richieste senza possibilità di poter intervenire sul carico senza interruzione dei servizi. È necessario l'upgrade del sistema stesso.

Per ovviare a tali limitazioni, la pratica ormai consolidata è quella di seguire un approccio distribuito che prevede l'allocazione delle risorse ed il bilanciamento delle richieste mediante la ripartizione su un cluster composto da un determinato numero di nodi. L'adozione di questa architettura implica tra i suoi vantaggi:

- alta disponibilità delle risorse e resilienza dei dati: eliminazione del *single point of failure* e disponibilità del servizio anche in presenza di guasti ad uno o più componenti del cluster. Il servizio continua ad essere erogato, con prestazioni ovviamente inferiori;
- scalabilità: le risorse vengono distribuite su più nodi. In caso di aumento del fabbisogno di risorse, le prestazioni possono essere aumentate aggiungendo nodi al cluster.

L'esperienza nella gestione del portale web dell'Osservatorio Etneo, ha evidenziato come, in concomitanza con l'accadimento di particolari fenomeni sismici e vulcanici, si verifica un drastico aumento delle connessioni al servizio web che talvolta ha comportato una imprevista indisponibilità del servizio a causa della saturazione delle risorse hardware disponibili.

Al fine di limitare il verificarsi di tale evenienza è stata avviata la progettazione e l'implementazione di un *load balancing cluster,* il cui scopo principale è quello di garantire accesso alle risorse ottimizzando la redistribuzione del traffico di rete, per limitarne l'impatto sull'erogazione del servizio web, assicurandone al contempo scalabilità e affidabilità di quest'ultimo.

Una prerogativa fondamentale nella progettazione del sistema è stata l'utilizzo esclusivo di strumenti software *open source*.

Lo studio della fattibilità di quanto detto si è basato sull'intensa ricerca di fondamenti scientifici, sulla corretta progettazione di architetture cluster [Pfister, 1997] in alta disponibilità [Ang e Tham, 2007] su sistemi open-source come Linux [Van Vugt, 2014; Kopper, 2005; Pfister, 1997] ed in modo da ottenere alte performance [Buyya, 1999a; Buyya, 1999b; Tanenbaum, 2006].

2. Architettura del sistema

La piattaforma del portale web è stata realizzata implementando una configurazione di tipo *load-balanced cluster* (figura 1) basata su 4 elementi chiave:

- *server proxy*: effettua il *load balancing*, ovvero la distribuzione del carico vera e propria, e applica le *policy* di accesso agendo come *firewall* del sistema;
- *server web master*: mantiene la copia originale del sito web e consente di effettuare accessi in modalità scrittura per la modifica della configurazione e dei contenuti;
- *server web slave (da 2 a N)*: forniscono l'accesso vero e proprio in modalità lettura per la consultazione del web, mantengono copie sincronizzate del *server web master*. Il numero dei server può essere scalato a seconda delle esigenze;
- *file server*: fornisce il supporto di memorizzazione ai singoli nodi del cluster per la condivisione dei contenuti statici (immagini, filmati, documenti). Consiste in un servizio di *file server* in alta disponibilità basato su *storage* di tipo SAN *(Storage Area Network)*.

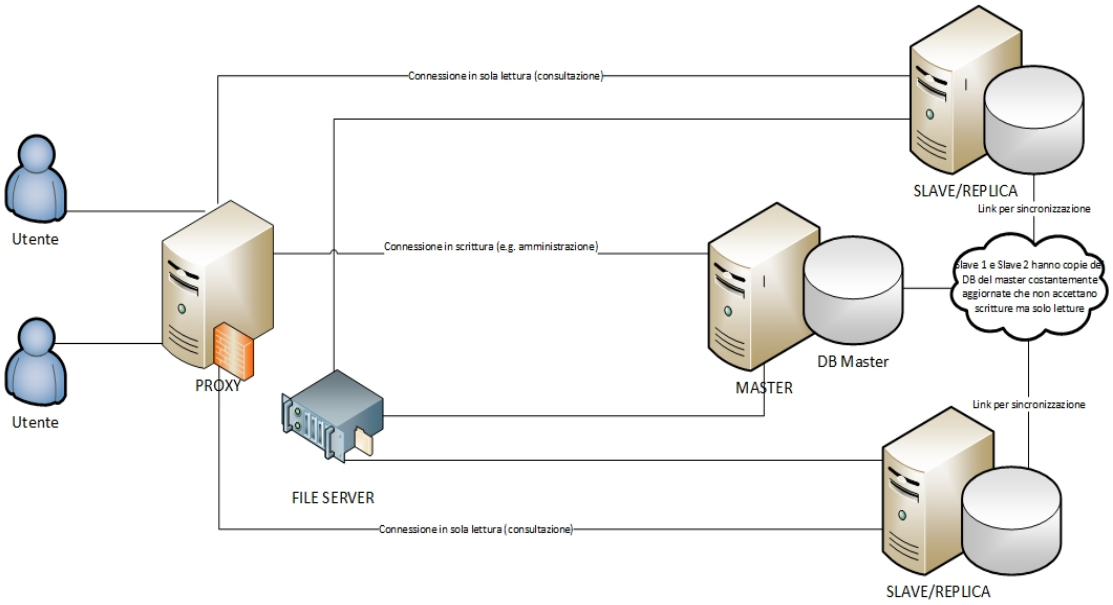

Figura 1. Architettura *load balancing cluster* proposta.

Il funzionamento del sistema può essere schematizzato come segue: la richiesta utente al servizio web viene presa in carico e inoltrata dal *server proxy* in modo trasparente, a seconda del tipo di azione richiesta (lettura o scrittura), e smistata ai singoli nodi del cluster in funzione del *workload*, secondo le regole definite nell'algoritmo di *network balancing* descritto nei paragrafi successivi. Il risultato che ne consegue è l'isolamento dei server dall'ambiente esterno e la distribuzione del carico di lavoro sui *server slave* del *cluster*. La presenza del nodo *master* ad accesso protetto consente di effettuare l'aggiornamento e la sincronizzazione dell'intero *cluster* da un singolo punto. La sincronizzazione garantisce la presenza di sostituti aggiornati al ruolo di master in caso di guasto di quest'ultimo.

2.1 Server Proxy

Il componente principale del sistema è il *server proxy* che implementa le regole di *firewall* per l'isolamento dei nodi del *cluster* dall'accesso esterno e attua il *load balancing* tramite l'indirizzamento delle richieste ai *server web*, in base al *workload* e al tipo di richiesta effettuata (lettura o scrittura). Le tecniche di *load balancing* definiscono i criteri applicati nella scelta del nodo per la ripartizione del carico e possono essere classificate a seconda della tipologia in:

- casuali: le richieste vengono inoltrate in modo casuale ai server disponibili;
- *health state* dei server: le richieste vengono inoltrate ai server disponibili in funzione di fattori quali ad esempio carico computazionale e numero di connessioni concorrenti;
- temporale: le richieste vengono inoltrate ai server in base ad un approccio basato sul timing delle richieste in coda.

Nel sistema proposto è stato implementato l'algoritmo *round-robin* (Fig. 2), ovvero un approccio misto tra criteri di scelta casuale e temporale dei nodi, che utilizza sequenzialmente tutti i server disponibili in modo da inoltrare sempre le richieste a nodi diversi. Tale tecnica rappresenta una buon compromesso tra performance e versatilità, adattandosi alla maggior parte degli scenari, compreso quello in esame in cui i nodi cluster hanno mediamente performance analoghe.

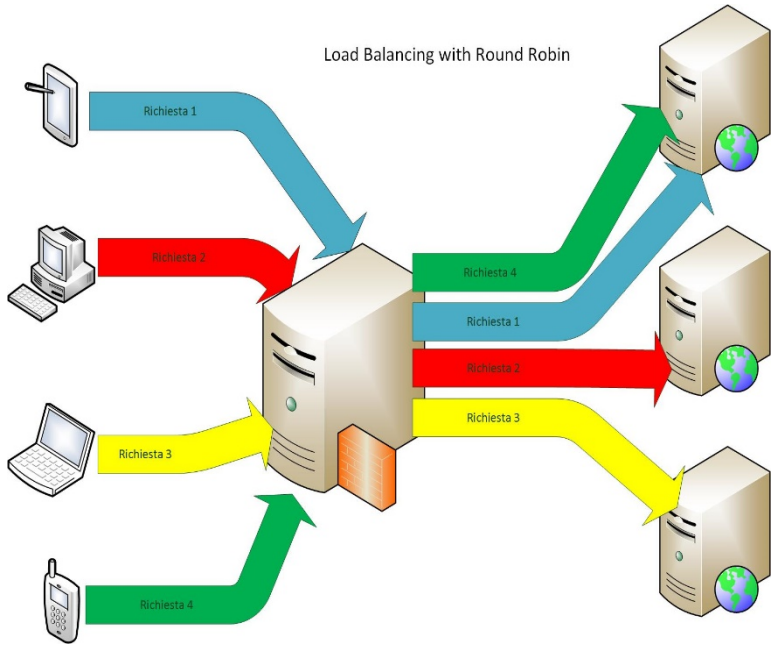

Figura 2. Load Balancing attraverso la tecnica round-robin.

2.2 Master/Slave Server Web e File Server

Il servizio per la consultazione dei contenuti web viene erogato da uno o più server, verso cui sono instradate dal *server proxy* le richieste utente dall'esterno. Una scelta che ha condizionato notevolmente la progettazione del sistema è stata quella riguardante la scelta del *Content Management System* (CMS) da utilizzare.

Per definizione, un CMS è uno strumento software per la realizzazione di portali web che svincola il webmaster da specifiche conoscenze di programmazione, concentrando il focus sulla capacità di gestione dei contenuti. Tale strumento risulta indispensabile nel contesto gestionale del portale web dell'Osservatorio Etneo in cui le persone coinvolte non sempre hanno specifiche competenze nel settore IT. Dopo aver valutato varie opzioni, la scelta è ricaduta su *Joomla!*, un software CMS *open source* scritto interamente in PHP, estremamente diffuso grazie alla sua semplicità, completezza e al supporto di una larga *community* di sviluppatori. Il funzionamento di *Joomla!* è basato sull'utilizzo di un *Relational Database Management System* (RDBMS) in cui risiedono tutti i dati necessari al funzionamento del CMS e dei contenuti web. Al fine di garantire la sicurezza, l'integrità e l'alta disponibilità del RDBMS è stato scelto di effettuare la replicazione dei database implementando una configurazione di tipo master/slave. Questo approccio ha permesso di ottenere vantaggi in termini di prestazioni e disponibilità del servizio web per i seguenti motivi:

- l'accesso in locale al RDBMS invece che attraverso la rete;
- l'accesso per la modifica del web avviene da un unico punto ad accesso protetto (*server web master*) evitando eventuali inconsistenze nella configurazione del portale;
- l'eliminazione del *single point of failure,* costituito dal nodo master, mediante la sincronizzazione del RDBMS su tutti i nodi *slave* del cluster. Questo consente che il nodo master, unico punto abilitato alle modifiche del RDBMS, in caso di guasto può essere velocemente rimpiazzato da un nodo *slave*, mediante la sua elezione automatica gestita da un apposito codice;
- il servizio web può essere facilmente *scalato* con l'aggiunta di nodi *slave*, mediante clonazione di quelli esistenti.

Tuttavia la replicazione dei RDBMS di *Joomla!* non risulta efficace nella protezione dei dati documentali quali le immagini, i video e i documenti pdf, visto che essi non sono effettivamente memorizzati nel database. Sebbene infatti sia tecnicamente possibile inserire documenti all'interno di un database mediante l'impiego di campi binari ("*BLOB*"), tale approccio è sconsigliato, sia per evitare di appesantire la gestione del database provocandone un rallentamento delle prestazioni, sia perché i campi binari non sono

supportati in modo standard da tutti gli RDBMS. L'adozione di un *file server*, fornito come servizio in alta disponibilità basato su uno *storage* di tipo SAN, garantisce al tempo stesso la centralizzazione delle risorse documentali dei *server web* e la resilienza dei dati. Tale approccio permette inoltre di centralizzare, ad esempio, i file di configurazione relativi ai software installati sulle macchine.

2.3 Vulnerabilità singolo Server Proxy

Durante la fase di definizione dell'architettura proposta è stato affrontato e risolto il problema relativo al *single point of failure* costituito dalla presenza di un singolo *server proxy*. Infatti, in caso di *fault* di quest'ultimo, la conseguente indisponibilità del servizio di *load balancing* da esso erogato causerebbe l'inaccessibilità dell'intera infrastruttura e di conseguenza l'indisponibilità del servizio web.

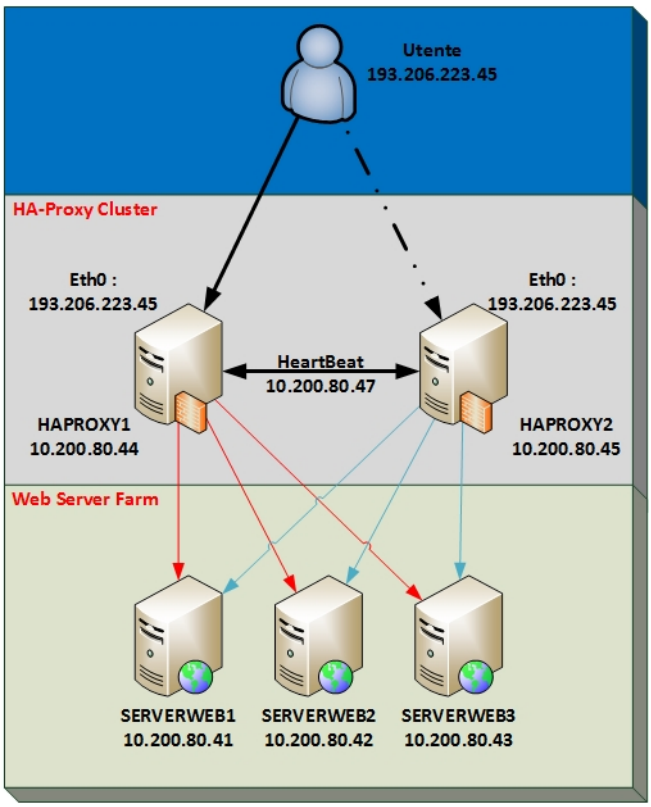

Figura 3. Proxy in modalità attivo/passivo con *keepalived*.

La soluzione a questo problema consiste nel configurare 2 server identici e quindi assegnare lo stesso indirizzo IP virtuale all'interfaccia esposta sulla rete pubblica. In caso di normale funzionamento, un server eroga il servizio di *load balancing* come descritto precedentemente, mentre l'altro rimane in uno stato dormiente. Il servizio di *keepalived* attivo sui *proxy* verifica costantemente lo stato di funzionamento del server attivo. In caso di un guasto del server attivo il sistema provvederà automaticamente a "sostituirlo" con quello dormiente, dirottando ad essa le richieste in modo trasparente all'esterno.

3. Implementazione del sistema

Sulla base di quanto discusso, si è deciso di implementare la soluzione proposta per la realizzazione del nuovo portale web dell'Osservatorio Etneo dell'Istituto Nazionale di Geofisica e Vulcanologia – Sezione di Catania. Di seguito verranno descritti gli strumenti software utilizzati per la realizzazione del portale con le modifiche salienti delle configurazioni principali.

Il *data center* dell'Osservatorio Etneo fornisce la piattaforma hardware su cui è effettuato l'*hosting* dei vari elementi del cluster quali *server web, file server*, *HA Proxy* in forma di *virtual host* e di servizi in alta disponibilità.

Il software utilizzato nell'implementazione è di tipo *open source*, sia per poter disporre del supporto di una vasta comunità di sviluppatori sia per aderire ai principi inspiratori della pubblica amministrazione in tema di fruibilità ed accessibilità.

I nodi *cluster* che ospitano il servizio web adottano una configurazione secondo lo standard LAMP *(Linux – Apache – MySQL – PHP),* un ambiente software completo per lo sviluppo del servizio web su piattaforma open source; il sistema operativo utilizzato è Linux Ubuntu Server 14.04 LTS (ultima versione disponibile al momento della stesura del presente report). Questa soluzione, largamente supportata dalla comunità di sviluppatori, garantisce una configurazione semplice e intuitiva per la preparazione dei ruoli necessari al funzionamento del *server web.*

Di seguito viene riportata la lista dei software principali:

- Installazione LAMP:
  - *Ubuntu Server 14.04 LTS* [*Ubuntu* documentazione in linea] sistema operativo *Linux* delle macchine virtuali;
  - *Apache* [*Apache* documentazione in linea] piattaforma per *server web*;
  - *MySQL* [*MySQL* documentazione in linea] *RDBMS* impiegato;
  - *PHP 5* [*PHP* documentazione in linea] linguaggio di programmazione *scripting* per la generazione di pagine web dinamiche;
- *Joomla 3* [*Joomla* documentazione in linea] CMS per il portale web. Nello specifico è stato adottato il *template Joomla Fap*, in quanto conforme alle norme vigenti sull'accessibilità e la facilità di utilizzo prevista per i siti della pubblica amministrazione;
- *HAProxy 1.5* [*HAProxy* documentazione in linea] software che implementa le funzionalità di *load balancing* installato sul *server proxy*;
- *KeepAlived* [*KeepAlived* documentazione in linea]: software di *routing* per garantire alta disponibilità dei server che erogano il servizio di *load balancing.*

Inoltre è stato creato un *repository* per l'archiviazione condivisa dei contenuti del web configurando un servizio di *file server* in alta disponibilità di tipo NFS (*Network File System*). Lo spazio di *storage* è fornito tramite l'assegnazione in *thin provisioning* di un volume dischi sotto forma di LUN (*Logical Unit Number*) dalla SAN del *datacenter*. La procedura per la preparazione e la configurazione dei nodi del cluster ha previsto i seguenti passaggi:

- Installazione del *server master* e configurazione base dei software LAMP;
- Clonazione del *server web master*;
- Creazione del servizio di *file server* e predisposizione dello *storage* SAN;
- Replicazione del database del *server master* su tutti i nodi *slave*.

3.1 Implementazione dei *Server Web*

Dopo aver eseguito l'installazione e la configurazione dell'ambiente LAMP del *server web master*, la cui trattazione è stata tralasciata in quanto conforme a quanto descritto nei procedimenti standard, si è proceduto alla replica del CMS del *server web* dell'INGV della Sezione di Catania – Osservatorio Etneo attualmente in produzione. Questo primo passaggio ha consentito di ottenere il nodo server (*master*) da cui clonare altri 2 nodi identici (*slave*) al fine di avere la configurazione minima discussa in precedenza.

Il montaggio di un volume NFS attraverso l'installazione di un *file server* sotto forma di servizio di un *cluster* in alta disponibilità in ambiente Windows Server 2012 R2 consente l'accesso in rete condiviso dei nodi del *cluster* agli archivi documentali gestiti dal CMS e ai file comuni a tutti i nodi utilizzati quali, ad esempio, i file di configurazione. Nello specifico si è scelto di implementare un collegamento alla risorsa condivisa di tipo permanente attraverso la modifica del file *fstab (/etc/fstab)*, con l'aggiunta di una riga di comando del tipo:

//servername/remoteDir /media/shareDir nfs credentials=/home/.smbcredentials,iocharset=utf8 0 0

La prima stringa rappresenta il percorso di rete per la cartella remota da condividere, la seconda la cartella locale in cui creare la condivisione a cui seguono il comando *nfs*, il file delle credenziali per

l'accesso alla cartella in remoto ed altri parametri di configurazione. Una volta modificato tale file, il montaggio della directory avverrà eseguendo il comando:

sudo mount –a.

La replicazione del database di Joomla 3 attraverso il meccanismo *master-slave* consente l'accesso in locale alle informazioni necessarie al corretto funzionamento del CMS garantendo la disponibilità del servizio web mediante la costante sincronizzazione dei database.

Per la configurazione del database master è necessario innanzitutto modificare il file di configurazione "*/mysql/my.cnf"* ed inserire nella sezione "[mysqld]" le righe mostrate in figura 4:

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

Figura 4. Modifica del nodo master.

Successivamente dovrà essere creato un utente *MySQL*, adibito alla sola replica, che gli slave potranno sfruttare per poter accedere al database master per la sincronizzazione.

```
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%.mydomain.com';
```

Figura 5. Comandi MySQL per configurazione utente di replicazione.

In tale passaggio è di rilievo il comando "*GRANT*" (figura 5) per definire i privilegi dell'utente creato. Tale comando assegnerà all'utente "repl" l'unico privilegio necessario ad operare nei database del server.

Infine deve essere eseguito il *dump* dei database da replicare. Il file con estensione ".sql" generato dovrà quindi essere trasferito sui server. Il comando per effettuare il *dump* è il seguente:

mysqldump -uroot -password --databases db1 db2 db3 --add-drop-database --master-data > /usr/dump_db.sql

Per la configurazione degli *Slave* è necessario modificare il file "*my.cnf"*, inserendo nella sezione "[mysqld]" un riga analoga a quella inserita nel *Master* per assegnare il *server-id* ed una riga per ogni database che si vuole replicare secondo la seguente sintassi:

replicate-wild-do-table=nomedb.%

Dopo aver importato all'interno di *MySQL* il dump generato dal *Master*, l'ultimo passo necessario è quello di avviare la sincronizzazione, attraverso il comando *START SLAVE* eseguito dallo strumento di gestione di *MySQL*. Da questo momento in poi qualunque modifica al database master sarà automaticamente replicata ai database *slave*.

3.2 Implementazione del *Server Proxy*

La configurazione di *HAProxy* è il cuore del funzionamento del sistema progettato. Una volta installato, la procedura di configurazione prevede unicamente la modifica del file *"/etc/haproxy/haproxy.cfg"*.

```
16 #
17 #  Listen on *:80 - Send traffic to the backend named "apache"
18 #
19 frontend www-http
20     bind *:80
21     mode http
22     option httplog
23     option http-server-close
24     acl is_admin path_beg -i /administrator
25     use_backend admin if is_admin
26     default_backend apache
27
28 #
29 # Back-end definition.
30 #
31
32 backend admin
33     mode http
34     balance roundrobin  # Load Balancing algorithm
35     option httpclose
36     server WEBadmin 10.200.80.41:80 check
37
38 backend apache
39     mode http
40     balance roundrobin
41     option httpclose
42     server web1 10.200.80.42:80 check
43     server web2 10.200.80.43:80 check
```

Figura 6. Parte principale del file *"haproxy.cfg"*.

Come evidenziato in figura 6, la parte principale della configurazione dell'*HAProxy* è distinta in 2 sezioni chiamate *front-end e back-end definition*.

Il *front-end* determina le regole per la gestione delle richieste. Nello specifico verranno gestite tutte le richieste sulla porta 80, direzionando tali richieste ai nodi i cui indirizzi *IP* sono definiti nelle sezioni *back-end*. Nelle righe 24, 25 e 26 è definita l'espressione regolare utilizzata per determinare il tipo di richiesta ricevuta in modo da decidere quale *back-end* utilizzare. Nel file di configurazione di cui sopra, le richieste alla parte amministrativa (identificata dalla presenza di */administrator* nella parte iniziale del *path*) verranno inoltrate al *back-end* che gestisce il nodo *Master* (definito "*back-end admin*"), mentre tutte le altre richieste verranno inoltrate al *back-end* degli *Slave* (definito "*back-end apache*").

Le richieste saranno smistate secondo l'algoritmo di bilanciamento impostato nel parametro "*balance*" che come anticipato è il *round-robin*. Da notare che tale parametro deve sempre essere esplicitato.

Un ulteriore aspetto importante di *HAProxy* è quello di fornire una pagina di statistiche in tempo reale sul funzionamento del *proxy*.

```
45 listen  stats   10.200.80.44:1936
46         mode            http
47         log             global
48
49         maxconn 10
50
51         clitimeout      100s
52         srvtimeout      100s
53         contimeout      100s
54         timeout queue   100s
55
56         stats enable
57         stats hide-version
58         stats refresh 30s
59         stats show-node
60         stats auth webmaster:**********
61         stats uri  /statspage
```

Figura 7. Configurazione pagina statistiche.

In figura 7 è mostrato l'inserimento nel file di configurazione delle righe che permettono la creazione di tale pagina, accessibile attraverso una richiesta di tipo *http* sulla porta 1936 all'indirizzo "*http://10.200.80.44:1936/statspage*". Opzionalmente, l'accesso a tale pagina può essere protetto da un sistema di autenticazione del tipo *username-password*. In figura 8 è infine riportata la pagina di statistiche generata.

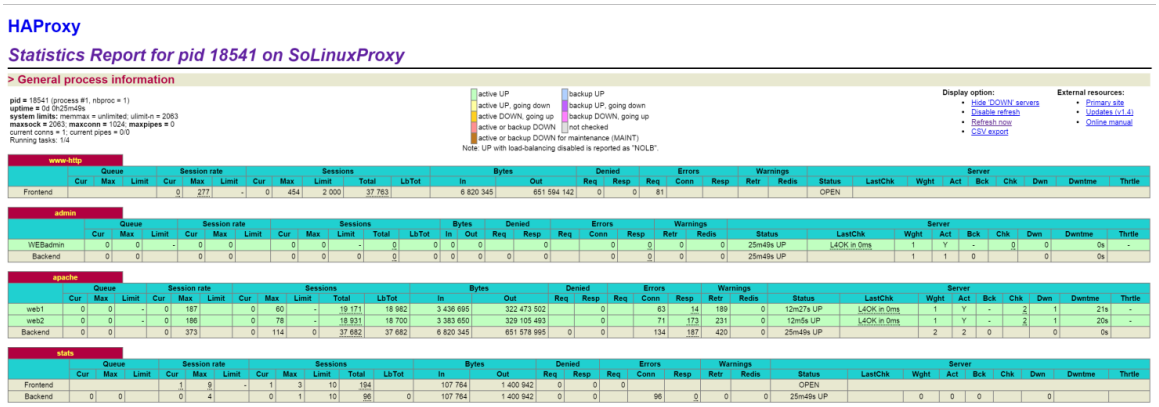

Figura 8. Pagina statistiche utilizzo *HAProxy*.

3.3 *Benchmark*

Di seguito vengono presentati i grafici relativi ai test di *benchmark* effettuati, che mostrano come il sistema progettato sia robusto ed efficiente, in grado di garantire un servizio in alta disponibilità nonostante carichi elevati di accessi e/o richieste.

I test effettuati hanno tutti la stessa matrice, applicata in diverse modalità: tutti infatti prevedono accessi simultanei da parte di molti utenti ma con differenti modalità di carico.

Il primo test, la cui configurazione è mostrata in figura 9, simula la presenza di 100 utenti che simultaneamente effettuano 1 *click* su un elemento ogni 5 secondi. La simulazione termina quando ogni utente ha effettuato 100 *click*.

Project and Scenario Comments, Operator	
Test Setup	
Test Type:	CLICKS (run test until 100 clicks per user)
User Simulation:	100 simultaneous users - 5 seconds between clicks
Logging Period:	Log every 10 seconds
URLs	
URL Sequencing:	Users always click the same URL (to spreads load evenly on all URLs, set number of users to a multiple of the number of URLs!)
URLs:	Click here
Browser Settings	
Browser Simulation:	User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
Browser Simulation:	HTTP Request Timeout: 120 s
Options	
Logging:	Write detailed log(s)
Timer:	not enabled
Local IPs: Automatic	URL#1: GET 193.206.223.44 POSTDATA= Click Delay=1
Client System	
System	Windows 8/2012 V6.2 (Build 9200), CPU Proc. Lev. 686 (Rev. 15363) at 2195 MHz,
Memory	13871 MB available RAM of 17078 MB total physical RAM, 15853 MB available pagefile
Test Software	
Webserver Stress Tool:	7.3.0.2296 Professional Edition

Figura 9. Configurazione primo test.

L’andamento del tempo medio di ogni richiesta, mostrato in figura 10, mostra un picco iniziale di circa 240 ms, per poi assestarsi a regime intorno ai 100 ms.

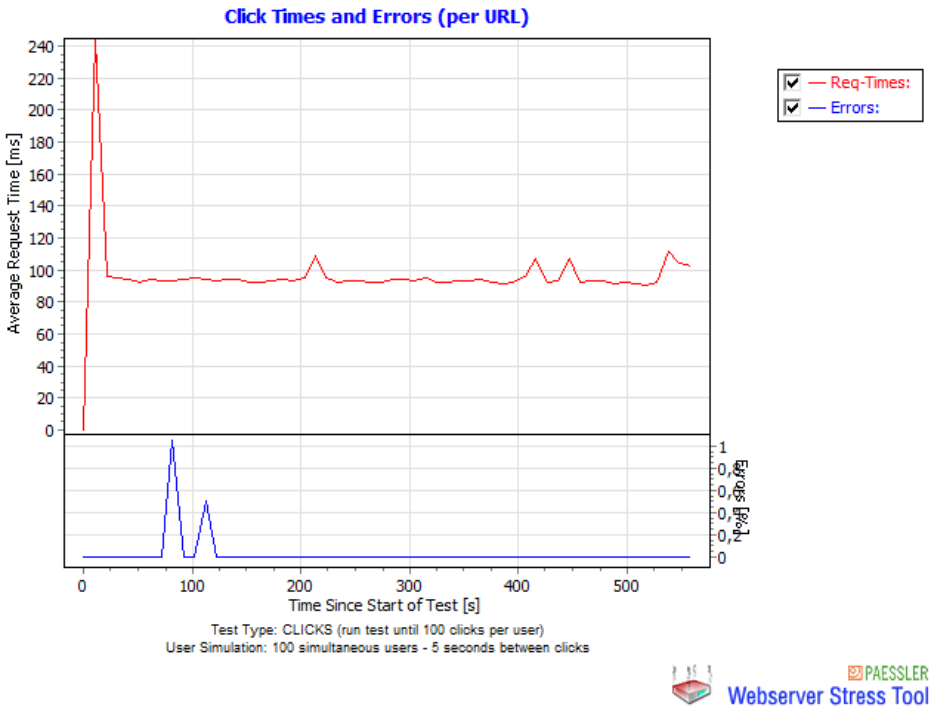

Figura 10. Risultati primo test.

La figura 11 mostra invece accanto al *click time*, gli *hits* per secondo che durante il periodo di stress hanno una media di 20 *hits* per secondo.

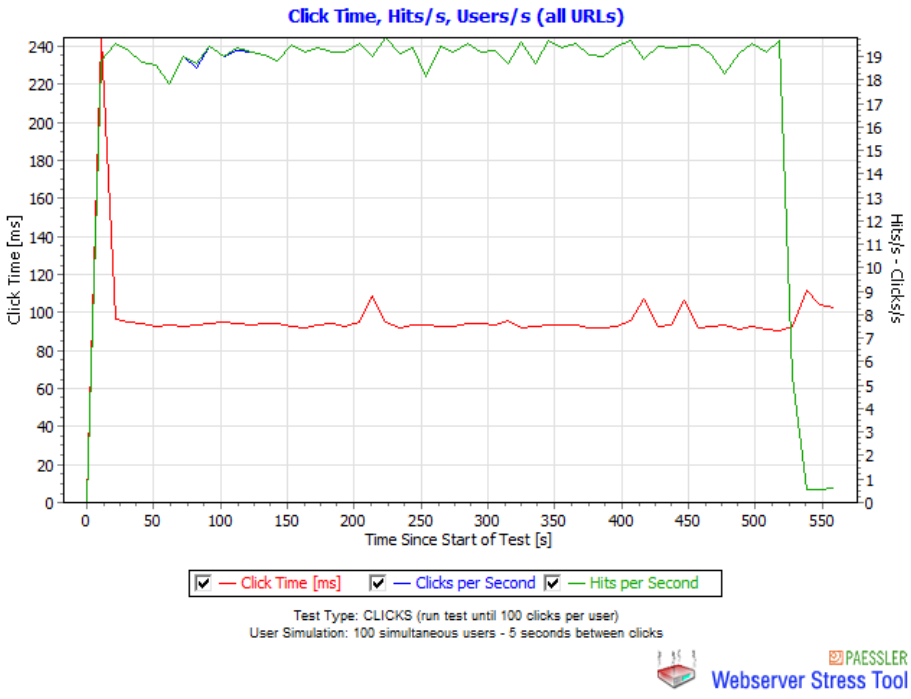

Figura 11. Risultati primo test.

In figura 12 è possibile vedere come gli andamenti temporali dei protocolli siano correlati con i tempi di *click* e che quindi picchi nei *click time* si riflettano sui tempi necessari ai protocolli.

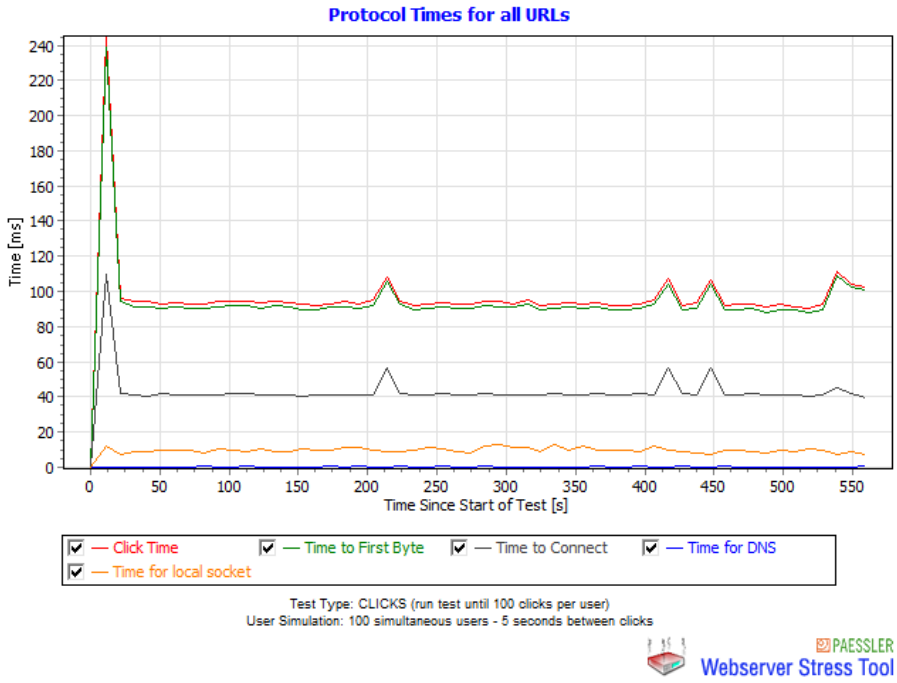

Figura 12. Risultati primo test.

Un breve sunto dei risultati del primo test è quindi il seguente, dove spicca la presenza di soli 3 errori:

URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		9.994	3	0,03	968.821	97

Il secondo test (figura 13), simula per 30 minuti l'utilizzo del sito da parte di 100 utenti che effettuano simultaneamente un *click* ogni 5 secondi.

Project and Scenario Comments, Operator	
Test Setup	
Test Type:	TIME (run test for 30 minutes)
User Simulation:	100 simultaneous users - 5 seconds between clicks
Logging Period:	Log every 15 seconds
URLs	
URL Sequencing:	Users always click the same URL (to spreads load evenly on all URLs, set number of users to a multiple of the number of URLs!)
URLs:	Click here
Browser Settings	
Browser Simulation:	User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
Browser Simulation:	HTTP Request Timeout: 120 s
Options	
Logging:	Write detailed log(s)
Timer:	not enabled
Local IPs: Automatic	URL#1: GET 193.206.223.44 POSTDATA= Click Delay=1
Client System	
System	Windows 8/2012 V6.2 (Build 9200), CPU Proc. Lev. 686 (Rev. 15363) at 2195 MHz,
Memory	13636 MB available RAM of 17078 MB total physical RAM, 15457 MB available pagefile
Test Software	
Webserver Stress Tool:	7.3.0.2296 Professional Edition

Figura 13. Configurazione secondo test.

In figura 14 è mostrato l'andamento temporale dei tempi di richiesta, che anche qui, dopo un breve periodo di assestamento, si attesta sui 100 ms. Come nel test precedente, sono presenti dei picchi in concomitanza dei quali anche i tempi per la gestione dei protocolli diventano più alti (figura 16).

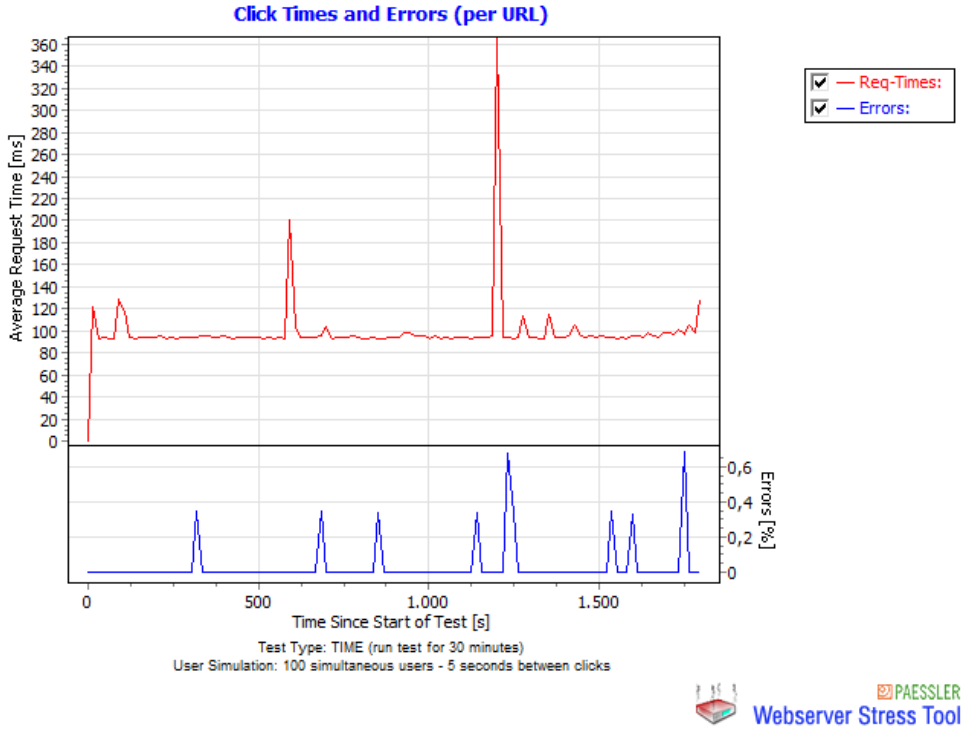

Figura 14. Risultati secondo test.

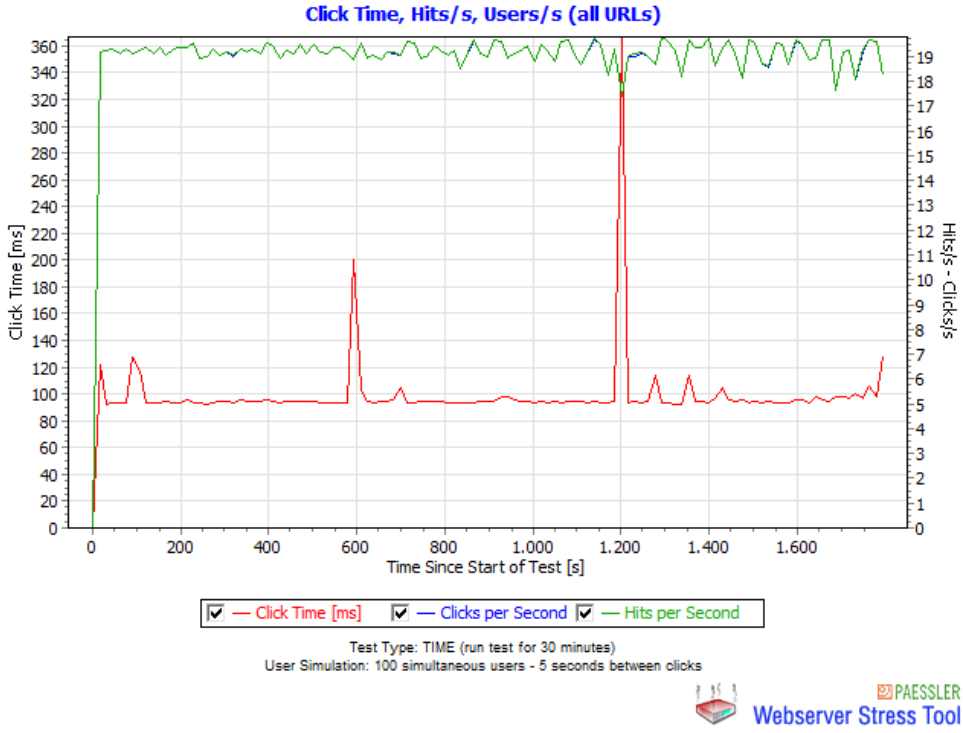

Figura 15. Risultati secondo test.

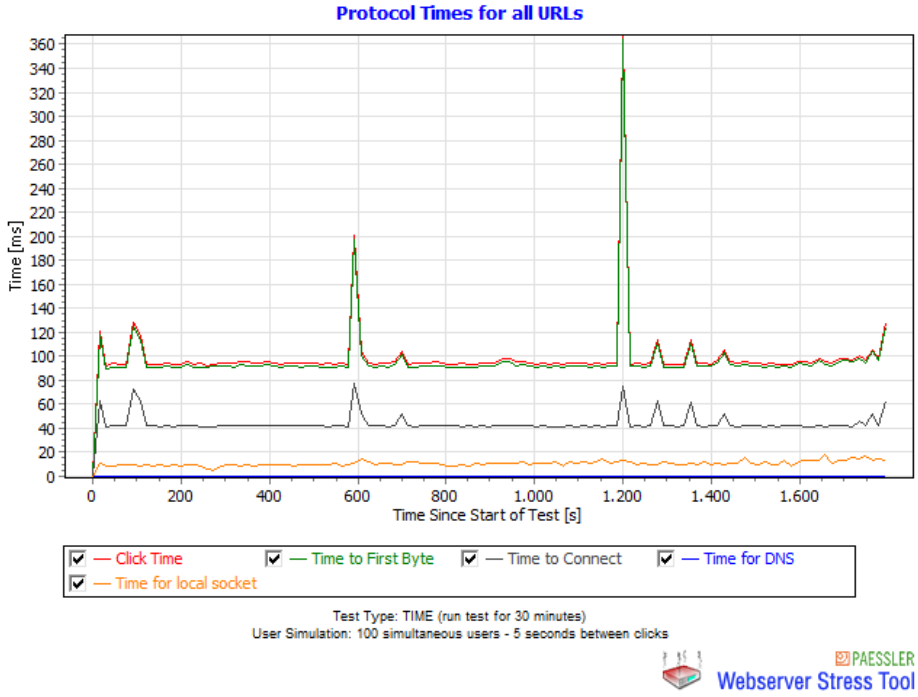

Figura 16. Risultati secondo test.

Un breve sunto del secondo test è il seguente, dove è possibile notare la presenza di soli 11 errori su 34000 *click* circa:

URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		34.381	11	0,03	3.404.426	

Il terzo test (figura 17) è denominato "a rampa": inizialmente i *click* sono effettuati da un solo utente. In maniera lineare col tempo vengono aggiunti utenti fino a raggiungere il limite di 100.

Project and Scenario Comments, Operator	
Test Setup	
Test Type:	RAMP (run test for 30 minutes)
User Simulation:	ramp test with up to 100 simultaneous users - 5 seconds between clicks
Logging Period:	Log every 15 seconds
URLs	
URL Sequencing:	Users always click the same URL (to spreads load evenly on all URLs, set number of users to a multiple of the number of URLs!)
URLs:	Click here
Browser Settings	
Browser Simulation:	User Agent: Mozilla/5.0 (compatible; Webserver Stress Tool 7; Windows)
Browser Simulation:	HTTP Request Timeout: 120 s
Options	
Logging:	Write detailed log(s)
Timer:	not enabled
Local IPs: Automatic	URL#1: GET 193.206.223.44 POSTDATA= Click Delay=1
Client System	
System	Windows 8/2012 V6.2 (Build 9200), CPU Proc. Lev. 686 (Rev. 15363) at 2195 MHz,
Memory	13621 MB available RAM of 17078 MB total physical RAM, 15346 MB available pagefile
Test Software	
Webserver Stress Tool:	7.3.0.2296 Professional Edition

Figura 17. Configurazione terzo test.

I risultati del terzo test possono essere visionati nelle figure seguenti. In figura 18 si evince come anche qui, nonostante dei picchi, il valore medio dei tempi di richiesta si attesta intorno ai 100 ms. Anche qui i tempi necessari ai protocolli di rete hanno un andamento simile a quello dei tempi di *click* come mostrato in figura 20.

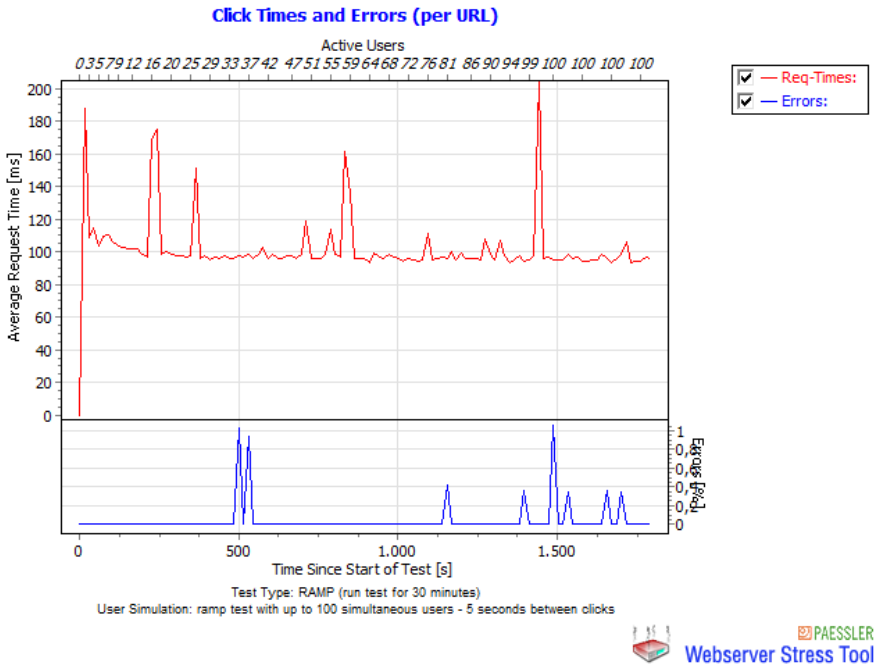

Figura 18. Risultati terzo test.

In figura 19 si può vedere come il numero di *hits* aumenti linearmente con l'aggiunta degli utenti, esattamente come ci si aspetta.

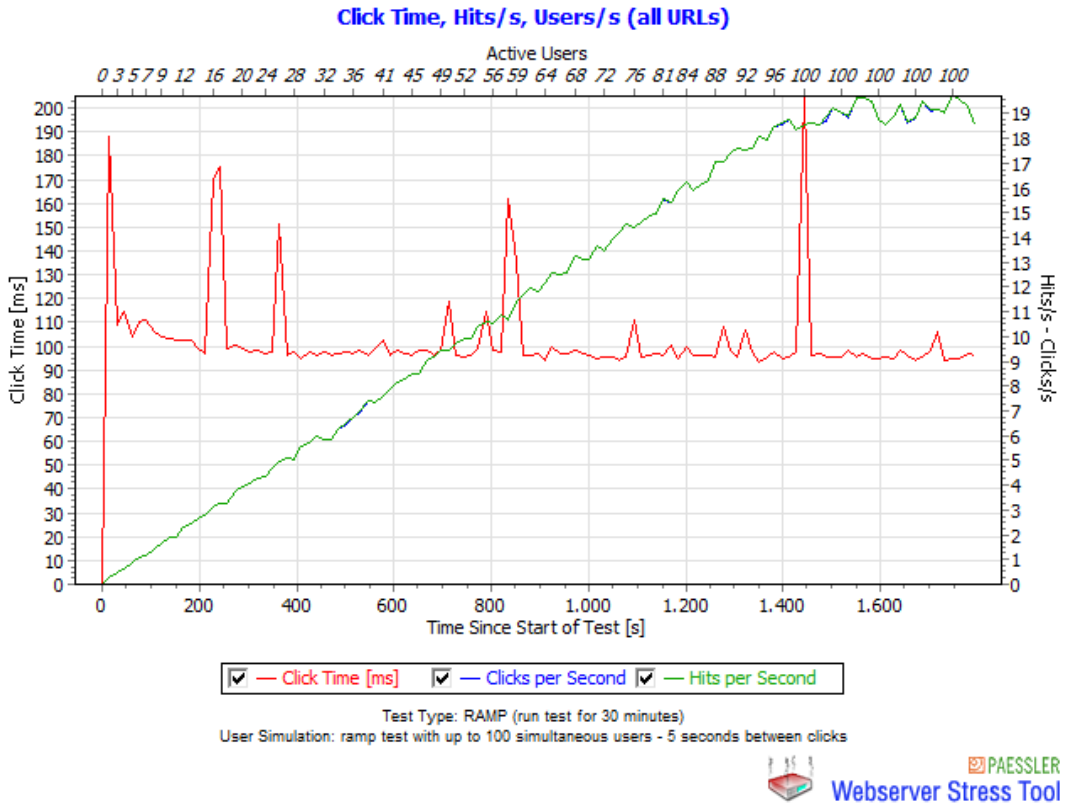

Figura 19. Risultati terzo test.

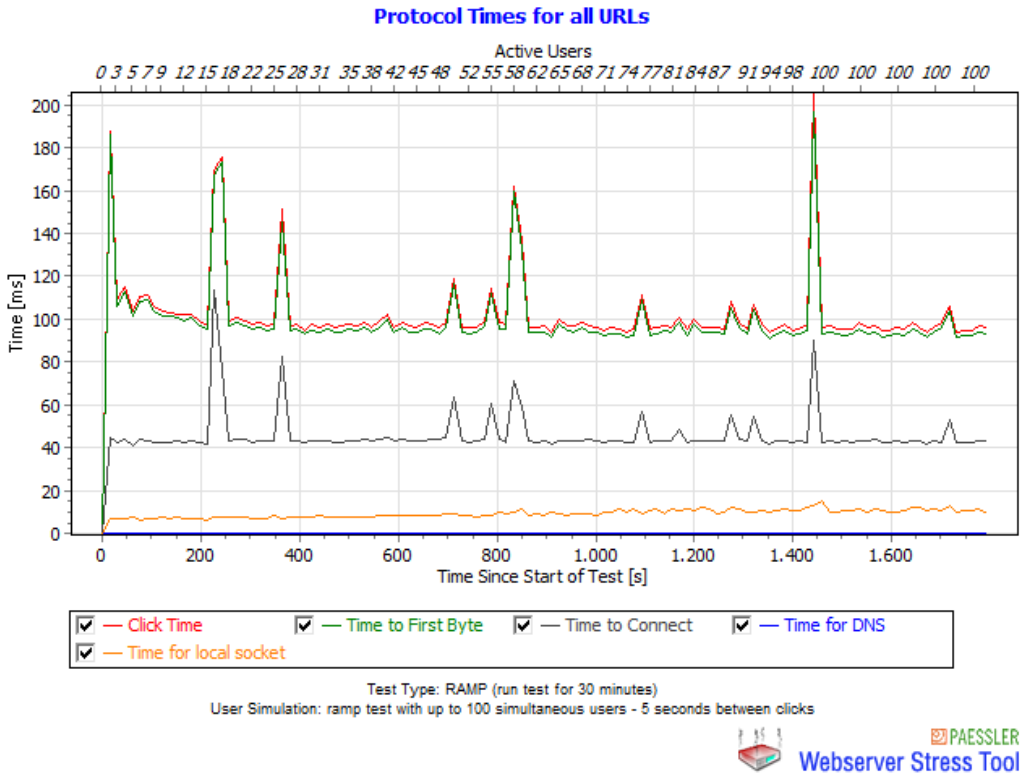

Figura 20. Risultati terzo test.

Anche qui mostriamo di seguito un breve sunto del test, in cui si verificano solamente 10 errori su un totale di oltre 20.000 *click*.

URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		20.661	10	0,05	2.071.386	100

4. Conclusioni e sviluppi futuri

Rispetto alle precedenti implementazioni [Mangiagli et al., 2010] [Mangiagli et al., 2012], è stata raggiunta una maggiore robustezza ed efficienza del sistema grazie all'adozione di un *server proxy:* Grazie ad esso è infatti possibile applicare un meccanismo di *load balancing* (nella fattispecie in modalità *round-robin*) che consente la ripartizione del carico tra i nodi ottimizzando l'uso delle risorse a disposizione. Nel caso di aumento del carico in ingresso, ad esempio in concomitanza con il verificarsi di particolare fenomeni vulcanici, è possibile aggiungere in modo trasparente dei nodi preconfigurati al sistema ed aumentare le risorse a disposizione.

Tuttavia, permane l'evenienza di una mancata indisponibilità del servizio web in concomitanza al raggiungimento del *throughput* massimo sulla linea internet della sede dove sono ubicati fisicamente i *server web*.

Inoltre l'attuale implementazione non garantisce la funzionalità di aggiornamento dei contenuti web al verificarsi di un guasto al *nodo master*. In questo caso, infatti, sarà necessario l'intervento di un operatore per ripristinare il server in fault.

In conclusione le migliorie introdotte nell'implementazione attuale costituiscono un decisivo miglioramento in termini di scalabilità e disponibilità del servizio web, anche se, in talune circostanze i limiti infrastrutturali descritti (*throughput* linea dati, unico nodo *master*) possono compromettere le funzionalità del sistema.

Attualmente è in fase di studio un ulteriore ottimizzazione dell'architettura allo scopo di eliminare le attuali limitazioni.

Bibliografia

Mangiagli S., D'Agostino M., Reitano D., Torrisi O., (2012). *Progettazione dell'infrastruttura per la gestione e l'hosting del portale web dell'Osservatorio Etneo dell'Istituto Nazionale di Geofisica e Vulcanologia.* Rapporti Tecnici INGV, n. 243.

Mangiagli S., La Via M., D'Agostino M., Reitano D., Torrisi O., (2010). *Realizzazione del portale Web della sezione di Catania*. Rapporti Tecnici INGV, n. 148.

Ang C.W., Tham C.K., (2007). *Analysis and optimization of service availability in a HA cluster with load-dependent machine availability*. IEEE Transactions on Parallel and Distributed Systems, Volume 18, Issue 9, Pages 1307-1319, ISSN: 1045-9219

Van Vugt S., (2014). *Pro Linux High Availability Clustering.* p.3, Apress, ISBN 978-1484200803.

Pfister G., (1997). *In search of clusters.* Prentice Hall PTR, ISBN 978-0-13-899709-0.

Buyya R., (1999a). *High Performance Cluster Computing: Architectures and systems*. Prentice Hall PTR, ISBN 978-0-13-013784-5.

Buyya R., (1999b). *High Performance Cluster Computing: Programming and applications.* Prentice Hall PTR, ISBN 978-0-13-013785-2.

Kopper K., (2005). *The Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software*. No Starch Press, ISBN 978-1-59327-036-0.

Tanenbaum A.S., (2006). *Cluster, in Architettura dei calcolatori. Un approccio strutturale.* Pearson Education, pp. 604-609, ISBN 978-88-7192-271-3.

Sitografia

Ubuntu documentazione in linea - http://www.ubuntu-it.org/scopri-ubuntu/server
Apache documentazione in linea - https://www.apache.org/
MySQL documentazione in linea - https://www.mysql.it/
PHP documentazione in linea - http://php.net/
Joomla documentazione in linea - http://www.joomla.it/
HaProxy documentazione in linea - http://www.haproxy.org/
KeepAlived documentazione in linea - http://www.keepalived.org/

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/l-ingv/produzione-scientifica/quaderni-di-geofisica/

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/rapporti-tecnici-ingv/

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

Miscellanea INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/miscellanea-ingv/

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli ecc..

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia