Disseny i simulació d'una infraestructura d'alta disponibilitat

Jaime Sarrió Luna

Resum— En l'actualitat existeix un creixement exponencial, tant de les infraestructures com dels serveis web. Aquest fet provoca que totes les noves infraestructures de l'entorn web hagin d'adaptar-se per tal de poder evolucionar juntament amb el creixement. En aquest article s'analitzen totes les etapes necessàries del procés de creació d'una infraestructura d'alta disponibilitat d'un servei web real mitjançant la virtualització del mateix. També s'analitza el rendiment per tal de poder garantir l'alta disponibilitat i adaptar l'entorn a la millor configuració possible.

Paraules clau— Internet, serveis web, servidor, Proxy, red, infraestructura, Apache bench, HAProxy, Gnuplot, rendiment, balanceig de peticions, MySQL, phpMyAdmin.

Abstract— Nowadays there is an exponential growth, both of infrastructures and web services. This causes all the new infrastructures in the web environment to adapt in order to be able to evolve along with growth. This article analyzes all the necessary stages for the process of creating a high availability infrastructure for a real web service through virtualization of the same. The performance is also analyzed in order to guarantee high availability and adapt the environment to the best possible configuration

Index Terms— Internet, web services, server, Proxy, network, infraestructure, Apache bench, HAProxy, Gnuplot, performance, rolling of petitions, MySQL, phpMyAdmin

1 Introducció

AQUEST treball dona suport a les noves maneres a les que els serveis web s'han d'anar adaptant, concretament, en l'àmbit dels serveis d'alta disponibilitat.

L'actual marc dels serveis web ofereixen un amplíssim rang de possibilitats en quant a la configuració dels serveis d'alta disponibilitat. Aquest treball es busca donar una solució a una nova infraestructura creada de forma completa.

El projecte es basa en el disseny i creació de tota la infraestructura i simulació d'un possible entorn real en l'actualitat, permetent la final implementació del mateix mitjançant un entorn virtualitzat, i així, poder avaluar el seu rendiment.

El document està estructurat en set seccions en les quals, en cada cas, s'avalua i implementa tot el necessari per tal de dur a terme el projecte.

A les tres primeres seccions es posa de manifest i en context el treball en qüestió, a la quarta secció es podrà veure la metodologia utilitzada i el desenvolupament del treball, a les seccions cinc i sis podrem veure els resultats obtinguts i aquests seran analitzats. Per últim a la setena secció es troben les conclusions i vies de desenvolupament futur del treball fi de grau.

2 Objectius

L'objectiu del projecte fi de grau és el de dissenyar un entorn d'alta disponibilitat. En el cas d'aquest projecte serà un servidor web. La finalitat és la de simular un entorn virtual, que al mateix temps pugui ser aplicat a un cas d'un possible entorn real d'un servei web. Aquest haurà d'estar disponible i en funcionament durant tots els dies i hores de l'any.

Per tal de dur-lo a terme els principals objectius a complir són els següents:

- Disseny i implementació d'una xarxa amb CORE.
- Comprovació del correcte funcionament.
- Virtualització d'un servidor web.
- Anàlisis del rendiment del servidor.
- Virtualització d'un segon servidor web
- Implantació i configuració de HAProxy a l'entorn.
- Comprovació del correcte funcionament.
- Anàlisis del rendiment de la nova infraestructura.

- *E-mail de contacte: jaime.sarrio@e-campus.uab.cat*
- *Menció realitzada: Tecnologies de la Informació.*
- *Treball tutoritzat per: Juan Carlos Sebastián Pérez (dEIC)*
- *Curs 2017/18*

Tots aquests Objectius són d'obligat compliment per tal de poder arribar a simular l'entorn desitjat, al mateix temps l'ordre anterior d'implementació de cada un dels objectius, permetran la evolució del treball i la seva correcte execució.

3 ESTAT DEL ART

En l'actualitat és vital per a les empreses i els negocis, tenir les aplicacions, serveis i pàgines web disponibles sempre, i no només parlem pel tema econòmic, sinó per qüestions d'imatge.

Quan s'intenta accedir a una pàgina web determinada i en un alt percentatge de les vegades no està disponible, en termes tant professionals com personals, ens dóna certa inseguretat.

Aquesta inseguretat al mateix temps provoca que l'usuari busqui alternatives, ja que es posa en dubte la disponibilitat del servei, es pot considerar que perdrem un nombre considerable de visites i interès per part dels usuaris.

Si hi ha un camp on els serveis d'alta disponibilitat destaquen, és en realitzar millores per tal de garantir la disponibilitat dels serveis. Des de un punt de vista extern, en l'àmbit de la alta disponibilitat, es pot considerar la clusterització dels serveis com la capacitat d'unir diverses màquines virtuals o físiques per donar servei als usuaris.

Aquestes màquines proporcionen de manera segura la disponibilitat de les aplicacions que necessitin els usuaris, i ho faran mitjançant la replicació i la implementació de polítiques de balanceig de recursos. D'aquesta manera aconsegueixen evitar sobrecàrregues dels servidors i en cas que algun falli no afectarà al usuari final mentre no caigui tot el sistema sencer.

Els clústers generats per aplicacions d'alta disponibilitat treballen de manera transparent a l'usuari final, la comunicació es realitza entre els servidors que són els encarregats de proporcionar el servei i gestionen totes les peticions generades per part dels diferents serveis i els diferents usuaris finals.

En l'actualitat existeixen diferents plataformes que permeten la virtualització, en la majoria d'entorns cloud els serveis que s'ofereixen són maquines virtualitzades dins de clústers, és per aquest motiu que totes les màquines que formen part del projecte seran virtualitzades.

Des d'un principi s'ha buscat que tot el software utilitzat sigui open source d'aquesta manera es podrà garantir en tot moment la viabilitat del projecte. Les eines més conegudes per virtualitzar entorns són: vmware, virtual box o Qemu[1]. Les tres eines permeten virtualitzar entorns complets dins d'un sistema operatiu.

Vmware disposa d'una versió gratuïta, tot i ser software de pagament, mentre que Qemu i virtualbox són eines open-source.

En aquest projecte s'ha fet servir virtual box ja que un dels principals avantatges és que es pot instal·lar en els principals sistemes operatius (mac-Os, Linux i Windows), i al mateix temps permetrà realitzar la configuració en qualsevol maquina d'un cluster amb aquests sistemes operatius en cas de ser necessari.

També s'ha optat per aquest software ja que després de fer recerca sobre el procés d'instal·lació com de clonatge de maquines virtuals, és el que permet fer-ho de forma més transparent a l'usuari de manera que aquest pot configurar totes les característiques necessàries dels sistemes com es faria dins d'un cluster en el cloud.

4 METODOLOGIA I DESENVOLUPAMENT

4.1 Metodologia

La metodologia escollida per dur a terme aquest projecte és el model en espiral[2].

Aquesta metodologia consisteix realitzar els objectius anteriorment plantejats del projecte com si aquests fossin cicles o iteracions, d'aquesta manera en cada iteració s'abordarà un únic objectiu que permetrà avançar cap al següent, aquest procés es repetirà fins completar tot el projecte el diagrama com es mostra a la figura 1.

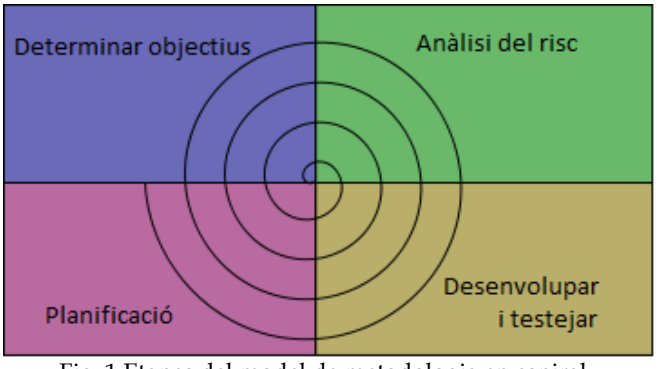


Fig. 1 Etapes del model de metodologia en espiral

En cada iteració es te en compte: l'objectiu, que es basa en quina necessitat ha de cobrir aquesta iteració de l'espiral en el conjunt del projecte. Les alternatives que són les diferents maneres d'aconseguir el correcte funcionament de l'objectiu que s'està iterant, i el desenvolupament i la verificació, que permeten desenvolupar i provar les funcionalitats implementades. Si el resultat no és l'adequat o es necessari implementar millores o noves funcionalitats per tal d'arribar a l'objectiu final, es planificaran les següents etapes i començarà de nou un cicle de l'espiral amb les següents etapes:

- La determinació d'objectius: en la primera iteració de l'espiral, en aquesta etapa es definiran els objectius inicials, a més es definiran els requisits de la funcionalitat i les especificacions, es fixaran restriccions i es definiran les estratègies alternatives per evitar els riscos que comporta que un mòdul no compleixi amb la seva consecució.
- L'anàlisi de riscos: durant aquesta etapa es duran a

terme estudis de possibles inconvenients que impedeixin el correcte desenvolupament de la implementació. S'avaluen alternatives als objectius inicials.

- El desenvolupament i la realització de proves: en aquesta etapa es realitza el desenvolupament dels objectius establerts en etapes anteriors. Una vegada finalitzat el desenvolupament o durant el mateix, es realitzen proves i comprovacions. Al final d'aquesta etapa s'analitzen les possibles alternatives als obstacles que s'han anat trobant durant el desenvolupament.

- La planificació: es tracta de l'última etapa de la iteració. Es recull tota la informació obtinguda al llarg de la iteració i es planifiquen els canvis i nous objectius per la següent iteració.

4.2 Desenvolupament del disseny de la infraestructura.

En l'actualitat els serveis web han d'estar disponibles per empreses com per diferents usuaris, en el nostre disseny s'han cobert ambdues parts com podem observar a la figura 2, s'han diferenciat les següents zones:

- Zona de servidors.

- Zona de usuaris particulars(Hosts).

- Zona d'empreses.

- Zona de proveïdors de serveis d'Internet (ISP).

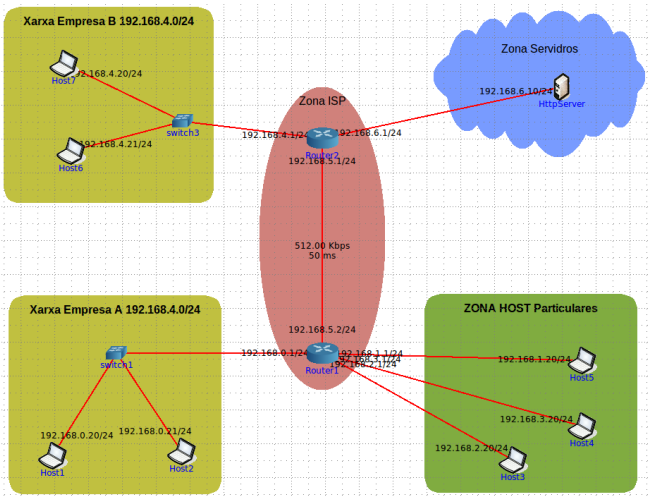


Fig. 2 Infraestructura de xarxa proposada per simular

Un cop teníem les parts necessàries ben definides hem realitzat el disseny amb l'eina Common Open Research Emulator (CORE)[3], Aquesta mateixa eina ens permetrà analitzar el correcte funcionament tant de la infraestructura com del servei web.

CORE és una eina que permet emular xarxes virtuals sense afectar a la màquina en la que s'executa.

Com a emulador, CORE construeix una xarxa informàtica que s'executa en temps real, a diferència de la simulació, on s'utilitzen models abstractes.

L'emulació en temps real provoca que totes les màquines es puguin connectar a xarxes físiques i routers.

Proporciona un entorn per executar aplicacions i protocols reals, aprofitant la virtualització proporcionada pels sistemes operatius Linux o FreeBSD, les seves característiques principals són:

- Escalable i eficient.
- Capaç d'executar aplicacions i protocols sense modificació.
- GUI fàcil d'usar.
- Molt personalitza-ble.

CORE s'utilitza normalment per a la investigació en l'àmbit de xarxes i protocols, permet fer demostracions, aplicacions i proves de plataformes, avaluant escenaris de xarxes i és possible efectuar estudis de seguretat i proves.

Per tal de realitzar aquesta part s'ha virtualitzat una maquina i s'ha configurat CORE per tal de poder dur a terme totes les proves desitjades.

Existeixen màquines ja virtualitzades però en el nostre cas no podíem realitzar totes les configuracions sense adaptar-la i per aquest mateix motiu es va decidir crear una màquina virtual i instal·lar l'eina per poder realitzar les proves de rendiment pertinents amb la configuració desitjada.

Un cop realitzades totes les configuracions de les interfícies y rutes d'encaminament i comprovat el correcte funcionament, podem veure les proves del correcte funcionament de l'entorn a la secció A1 i A2 de l'apèndix inclòs en l'article.

Per tal d'avaluar diferents escenaris, s'ha simulat tràfic entre dues màquines de les empreses.

Les modificacions fetes al diagrama han estat:

- Limitar l'ample de banda entre l'enllaç dels dos routers que formen part de la zona ISP a 512 Kb/s.

- Afegir una latència de 50ms a l'enllaç.

- Simular trafic entre empreses amb una ample de banda de 512 Kb/s.

D'aquesta manera quedarà saturat el canal i podrem observar com afecta al rendiment del servei web.

Les màquines de la zona de servidors seran les encarregades de proporcionar el contingut web. Ho faran mitjançant un servidor Apache[4].

Les imatges de les proves de rendiment que es mostren a continuació es poden veure més ampliades a la secció A3 del apèndix del treball.

A la figura 3 podem observar el comportament d'una petició cap al servidor quan no hi ha tràfic a la xarxa juntament amb els temps de resposta dels ping generats.

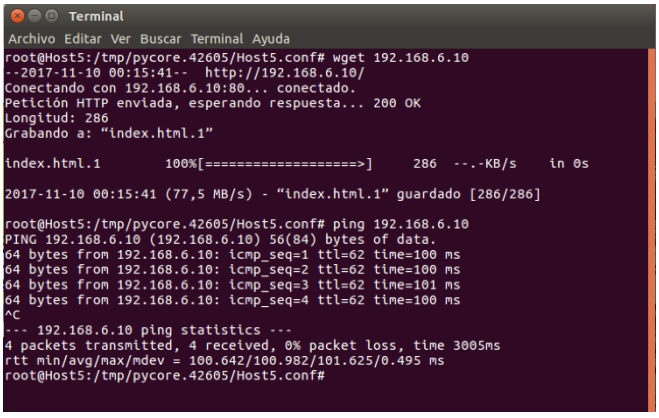


Fig. 3 Prova de rendiment amb CORE sense tràfic

Si es repeteix la mateixa simulació però aquest cop amb modificacions tant en el trànsit de la xarxa com en el ample de banda, s'observa que els números varien dràsticament com podem observar a la figura 4

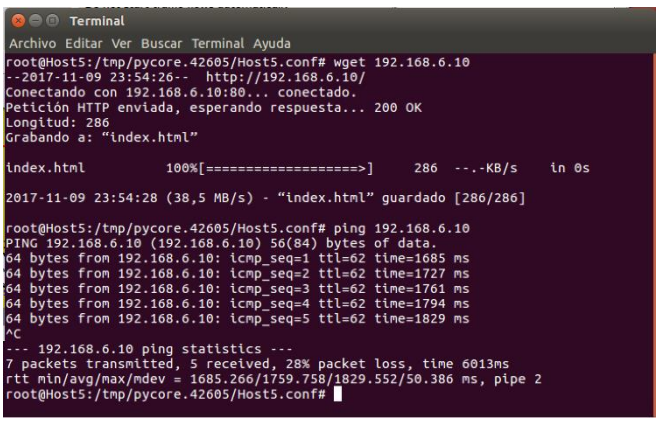


Fig. 4 Prova de rendiment amb CORE amb tràfic

En les imatges anteriors es pot apreciar com l'ample de banda cau a menys de la meitat, mentre que el temps de resposta de les peticions ping cap al servidor augmenten més d'un 1700% en temps tot i la latència afegida ser només de 50ms més a l'enllaç.

L'eina CORE permet quasi formes il·limitades de simulació d'entorns de xarxa i és molt potent de manera que, amb poca infraestructura, es poden generar grans simulacions.

També permet realitzar configuracions i simulacions per tal de testejar entorns que encara no han estat implantats físicament i així detectar possibles problemes de la infraestructura a priori per tal de poder solucionar-los.

4.3 Desenvolupament del entorn de proves d'un servidor únic.

En aquesta part de l'informe realitzarem la configuració del nostre servidor web al que posteriorment en la següent etapa de treball fi de grau realitzarem proves de rendiment.

L'esquema de la figura 5 representa la nova infraestructura a generar, aquesta primera estructura planteja els elements imprescindibles d'un servei web dins un entorn real, disposa d'una màquina que farà la petició un node intermig que simularà un ISP i d'un node que farà de servidor. El node intermig posteriorment serà el que configurarem amb el servei HAProxy.

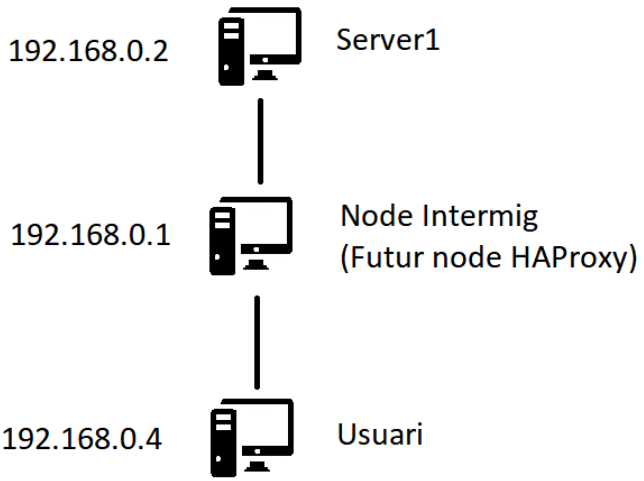


Fig. 5 Esquema d'infraestructura del servei web creat amb un servidor sol.

Seguint amb la mateixa filosofia que en el cas anterior trobarem eines gratuïtes que incorporen tot el necessari per tal de configurar un servidor web automàticament dins entorns com Windows, les principals eines d'aquest estil són xamp o wamp que es basen en un paquet de software (Apache2,myseql,php) ja pre-configurat però s'han descartat perquè no garantien el correcte funcionament dins d'una infraestructura d'alta disponibilitat.

En el nostre cas el que farem serà generar una nova màquina virtual que serà l'encarregada de gestionar tot.

Aquesta màquina serà un clon de la nostre màquina base. Un cop iniciada ens disposarem a instal·lar tot el necessari que en el nostre cas serà:

- Servidor Apache2.
- Mòdul PHP .
- Base de dades MySQL[5].
- Gestor de base de dades phpMyAdmin[6].

Per tal de configurar aquestes màquines (server1 i ISP) canviarem els arxius de /etc/hostname i /etc/hosts per tal de poder saber on som en cada moment en cap cas serà un servidor DNS.

Per tal d'interconnectar les màquines també s'han modificat i configurat la xarxa modificant l'arxiu /etc./network/interfaces a les màquines usuari i server 1 modificant les adreces IP pertinents.

A la màquina intermitja o futur node HAProxy També haurem de habilitar el forwarding des comentant la línia que l'habilita a l'arxiu que es troba a /etc/sysctl.conf

descomentant el camp que es mostra a la figura 6.

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Fig. 6 Activació del forwarding a l'arxiu /etc/sysctl.conf.

En la secció A3 de l'apèndix es mostren captures on es pot comprovar el correcte funcionament de la configuració anterior.

També es poden veure captures de les versions php d'Apache2 i phpMyAdmin de MySQL instal·lades.

Un cop vam tenir tot el programari instalat al servidor1 es va desenvolupar la pagina web amb les següents característiques:

- La web disposara de taules i aquestes, s'ompliran amb peticions independents a la base de dades.
- Disposarà de tres imatges.

Aquesta web busca donar suport a un servei ciutadà de telèfons de caràcter públic el qual haurà d'estar disponible sempre, per aquest motiu s'implementara un servidor que serà una replica del primer per tal de garantir que el servei web estigui sempre disponible.

L'objectiu del treball no es basa en dissenyar una web molt complexa sinó en fer un anàlisis de rendiment de la infraestructura creada amb la web creada com exemple de servei d'alta disponibilitat. Tot i estar tots els nodes hostejats dins la mateixa màquina, cal remarcar que els serveis d'alta disponibilitat normalment es troben distribuits en punts estratègicament situats en diferents entorns. D'aquesta manera poden garantir que tot i caure part del sistema, el servei seguirà funcionant sempre i quan tingui els elements mínims necessaris.

A la figura 7 es pot observar la web generada.

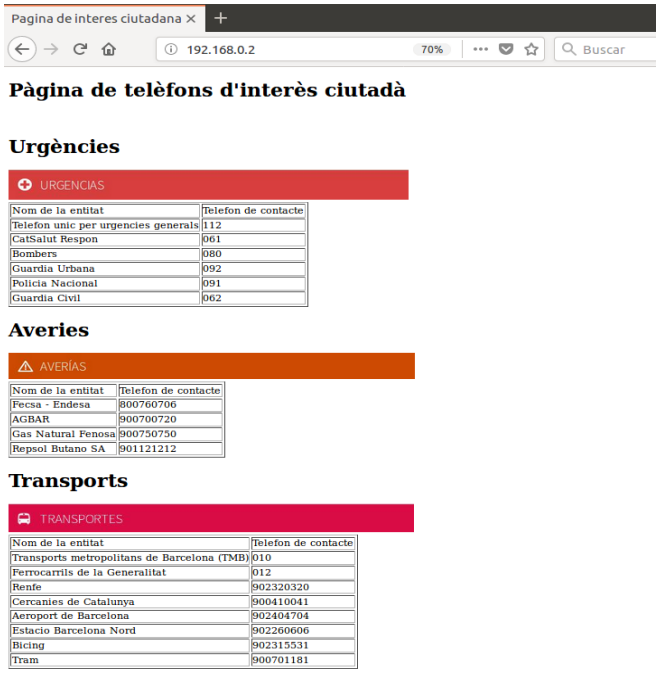


Fig. 7 Pàgina web creada i subministrada pel servidor

4.3 Desenvolupament del entorn de proves de dos servidors amb HAProxy.

El segon entorn de proves és la rèplica del servidor juntament amb la implementació del software HAProxy, a la figura 8 es pot observar la el nou disseny de la infraestructura a analitzar.

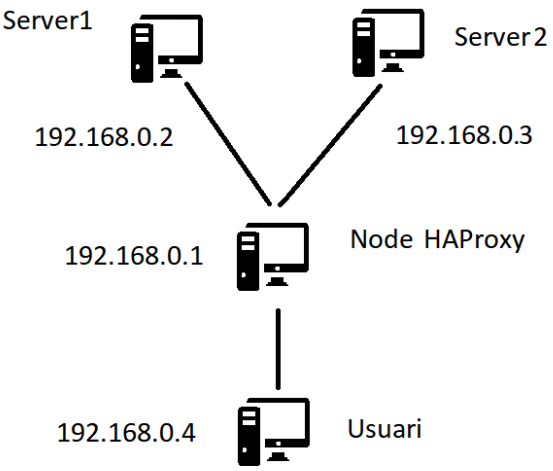


Fig. 8 Esquema del servei d'alta disponibilitat

L'eina High Availability Proxy (HAProxy)[7] permet fer balanceig de càrrega entre els dos servidors (Server1 i Server2), HAProxy està basat en els protocols TCP/HTTP i permet configurar de diverses polítiques de balanceig de carrega amb diferents algorismes.

Un cop instal·lat accedirem a la configuració editant el fitxer que es troba a /etc/haproxy/haproxy.cfg. Caldrà definir un frontend i un backend. A continuació a la figura 9 es mostra la configuració realitzada en el nostre cas.

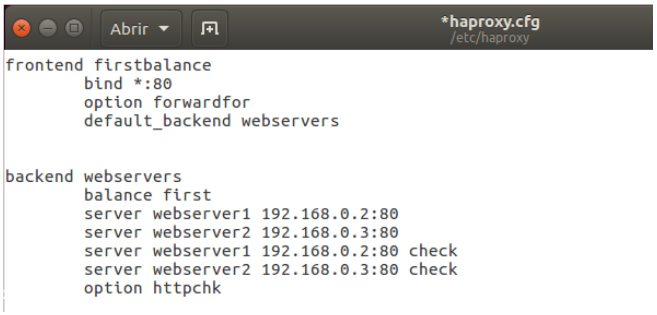


Fig. 9 Exemple de configuració de HAProxy

En aquest entorn el frontend es configurarà amb els següents parametres:

- El bind (enllaç) amb un asterisc o amb la direcció IP de la màquina on està configurat HAProxy, permetent que només faci el forward de les IP desitjades en cas de tenir més d'una configurada.

- El camp option habilita el header X-Forwarded-For (XFF) de HTTP.

- El camp default_backend enllaça amb el backend de l'arxiu de configuració.

En el backend hem assignat el mateix nom que al frontend i hem afegit dos servidors als quals hem activat la opció check que permet que passin capçaleres d'error d'estat i de funcionent correcte i hem configurat el camp balance.

El camp balance potser és el més important, ja que guiarà el comportament del nostre servidor Proxy. En el cas de HAProxy disposa dels següents valors: roundrobin, leastconn, static-rr, first, source, uri, url_param, hdr, rdp-cookie. Donada la nostra configuració de servidors, les polítiques de balanceig que té sentit provar són les de roundrobin, first i static-rr. Aquestes polítiques les compararem amb els resultats obtinguts en la del servidor sense HAProxy. La resta de polítiques de balanceig no té sentit avaluar-les ja que depenen de factors com sessions (leastconn), IP (source) o bé tant uri com paràmetres de uri juntament amb valors hash de cookies.

A continuació veurem de les polítiques utilitzades tenen les següents característiques i en que es diferencien entre elles:

- Roundrobin: dividirà les peticions i les repartirà per torns canviant entre els dos servidors alternament. A la arribada de peticions noves es poden afegir valors de pes per derivar més tràfic cap a un servidor si es desitja, de forma dinàmica.

- Static-rr: funciona com roundrobin però sense poder ajustar els valors de pes dinàmicament, per tant re

- quereix d'un menor us de CPU (al voltant d'un 1%).

- First: assignarà les peticions al primer servidor que tingui un espai lliure. Fins que no es saturi el primer servidor no s'assignaran les peticions cap al segon.

Per les dues configuracions s'ha analitzat el rendiment en la secció de proves de rendiment amb l'eina Apachebench (ab). Aquesta eina és un mòdul que actualment inclou Apache2 per tant hem tingut que instal·lar-lo a l'usuari però no ha requerit cap altra configuració addicional. Igual que el servidor web Apache, és tracta d'un programa amb llicencia lliure i de codi obert i es distribueix sota els termes de la llicència Apache.

5 PROVES DE RENDIMENT

En aquest apartat analitzarem i farem una prova de reniment a les configuracions de les figures 5 i 8 de les seccions anteriors.

La comanda utilitzada per fer les proves de rendiment ha estat la següent:

ab -g resultat.tsv -n 10000 -c 100 http://192.168.0.2:80/index.php

Els camps de la companda venen donats per:

- -g és el parametre on introduirem el valor del fitxer que generarà l'eina apachebench per posteriorment poder realitzar gràfics amb l'eina Gnuplot[8].

- -n seran el numero de peticions totals cap al servidor.

- -c seran el número de peticions concurrents.

- L'últim camp és la pàgina web a fer la prova de rendiment que en el nostre cas es troba allotjada al Server1 i posteriorment farem les proves cap al servidor HAProxy.

A la figura 10 trobem exemple de que retorna una execució de una prova de rendiment amb apache bench.

```
user@user-VirtualBox: ~

Benchmarking 192.168.0.1 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        Apache/2.4.18
Server Hostname:        192.168.0.1
Server Port:            80

Document Path:          /index.php
Document Length:        92347 bytes

Concurrency Level:      100
Time taken for tests:   34.863 seconds
Complete requests:      10000
Failed requests:        9816
   (Connect: 0, Receive: 0, Length: 9816, Exceptions: 0)
Total transferred:      925028288 bytes
HTML transferred:       923328288 bytes
Requests per second:    286.84 [#/sec] (mean)
Time per request:       348.631 [ms] (mean)
Time per request:       3.486 [ms] (mean, across all concurrent requests)
Transfer rate:          25911.30 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   3.3      1      32
Processing:    13  346 171.6    314    4107
Waiting:        1  206 179.4    204    3710
Total:         14  348 171.6    316    4107

Percentage of the requests served within a certain time (ms)
  50%    316
  66%    350
  75%    384
  80%    409
  90%    513
  95%    618
  98%    776
  99%    879
 100%   4107 (longest request)
user@user-VirtualBox:~$
```

Fig. 10 Execució de Apache bench

La información generada per Apache bench vidra donada per arxius en format TSV, el significat d'aquesta extensio és valors separats per tabulacions i són dades que pot interpretar l'eina Gnuplot.

Gnuplot és una eina portàtil de gràfics basada en la línia de comandes per a Linux, OS/2, Windows, OSX, VMS i moltes altres plataformes. Tot i no ser una eina open source està distribuïda lliurement. Va ser creada per permetre als científics i estudiants visualitzar funcions ma-

temàtiques i dades de forma interactiva, actualment també te usos no interactius, com ara el script web.

Un dels valors que s'obtenen només acabar l'execució és el de failed requests. Després d'investigar, s'ha pogut arribar a la conclusió que el test de rendiment s'estava executant correctament. L'eina Apache bench retornarà un valor de failed request sempre que variï la longitud del content response. Al treballar amb HAProxy, aquest afegeix el header X-Forwarded-For, de manera que les que coincideixen és pura casualitat mentre que la resta retornaran el valor de failed tot i carregar la web correctament.

Un cop dit això els valors més importants en els que ens hem de fixar o que són més representatius retornats per Apache bench són:

- Request per second: seran les peticions que es podran atendre per segon durant el transcurs de la prova de rendiment.

- Time per request (mean): serà el temps mig en atendre al grup de peticions concurrents. En l'exemple d'execució podem veure com quadren el temps mig de resposta de les 100 peticions amb el de una petició (cal multiplicar per 100).

- Time per request (mean, across all concurrent requests): serà el temps mig en que el servidor ha contestat una petició individual.

- En l'última part també es pot observar el percentatge de les peticions que es responen en un cert temps fent referencia a total de peticions, aquest camp es pot observar de forma més clara a les gràfiques.

Apache bench només dona nou valors de temps de peticions, per tal d'obtenir una major representació del temps de les diferents peticions es poden tractar els arxius amb extensió tsv generats a les proves de rendiment.

Totes les proves de rendiment s'han repetit cinc vegades per poder obtenir un resultat més representatiu i es veuen representades a les línies de les gràfiques amb Test1 a Test5.

Al mateix temps cal remarcar que a l'hora d'efectuar les proves de rendiment no s'ha introduït tràfic a la xarxa.

Tot i no haver tràfic extra a la xarxa, al estar totes les màquines virtuals hostejades al mateix ordinador, el tràfic generat per les diferents peticions i les diferents respostes per part de totes les màquines, farà que els resultats obtinguts si que es vegin alterats degut a aquest tràfic. Per aquest motiu en les noves infraestructures generades no s'ha afegit tràfic extra.

A la figura 11 es mostra una gràfica d'una prova de rendiment feta a la primera configuració amb un únic servidor.

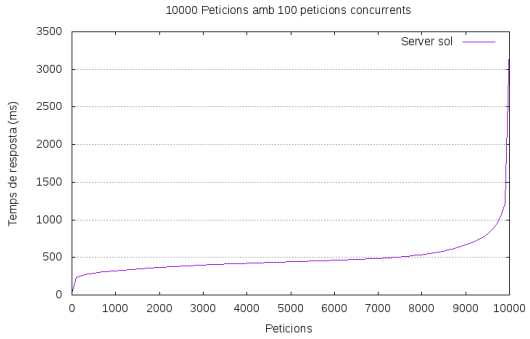


Fig. 11 Exemple d'execució d'una prova de rendiment amun servidor sol.

A la figura 12 es poden veure els resultats obtinguts per la configuració d'un servidor sol.

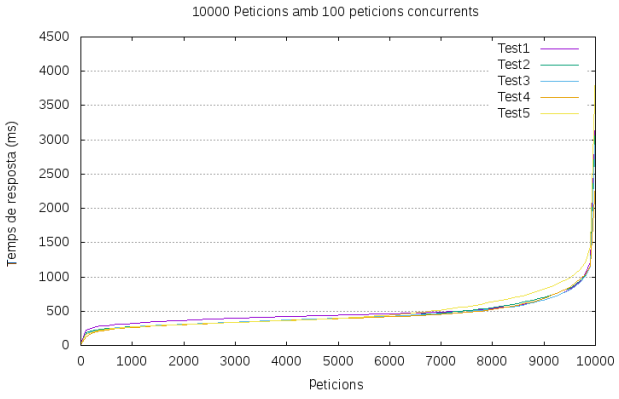


Fig. 12 Resultats del servidor sol

La figura 13 es poden observar els resultats obtinguts amb la configuració de Haproxy amb política de balanceig roundrobin.

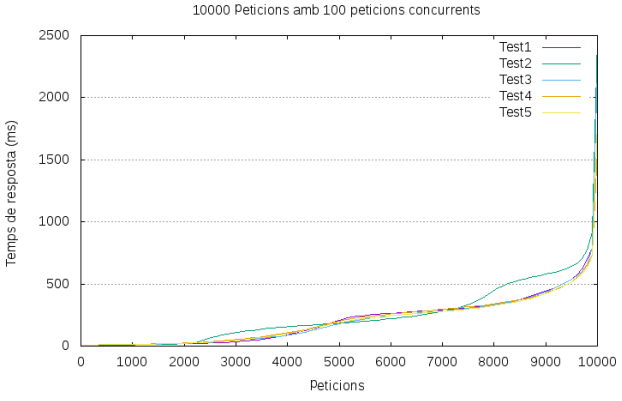


Fig. 13 Resultats HAProxy amb política de balanceig roundrobin.

Cal destacar que aquesta política és la única que ha obtingut variacions representatives en alguna de les seves proves de rendiment, es pot observar en el cas de Test 3.

A la figura 14 es poden observar els resultats obtinguts amb la configuració de Haproxy amb política de balanceig static-rr.

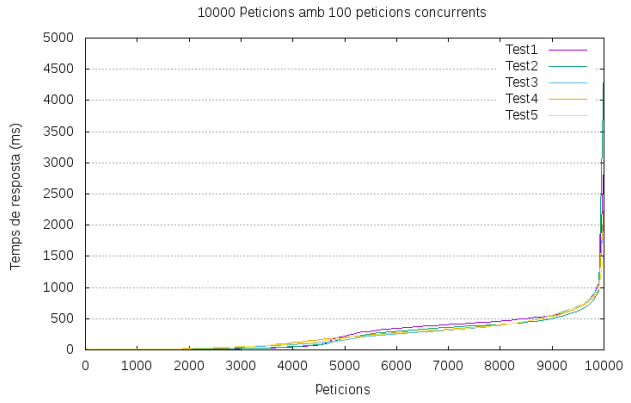


Fig. 14 Resultats HAProxy amb política de balanceig static-rr

A la figura 15 es poden observar els resultats obtinguts amb la configuració de Haproxy amb política de balanceig first.

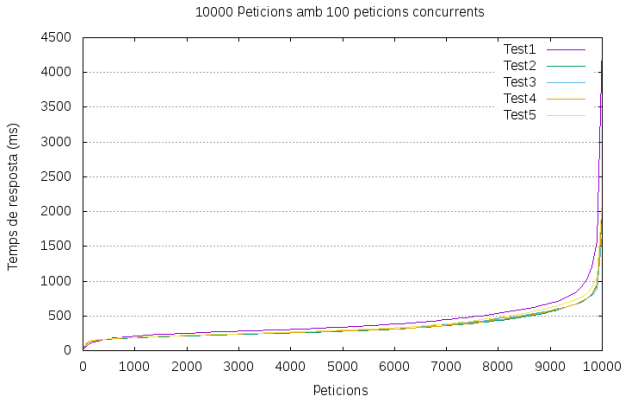


Fig. 15 Resultats HAProxy amb política de balanceig first.

Com podem observar en gran part les diferents execucions són molt semblants i no varien d'un test a un altre.

A continuació, a la figura 16 podem observar els diferents tests realitzats superposats entre ells triant una execució de prova de rendiment de les anteriors com a representativa.

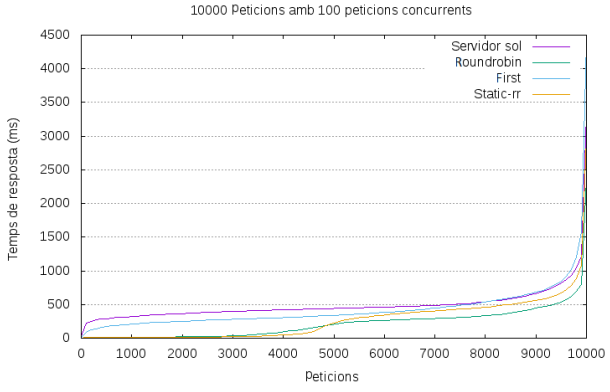


Fig. 16 Gràfica comparativa dels diferents resultats.

6 DISCUSSIÓ DELS RESULTATS

El gràfic de la figura 16 és el més representatiu. A continuació analitzarem amb profunditat els diferents resultats obtinguts.

El gràfic mostra el conjunt de peticions respostes en un temps determinat, l'ordre de les peticions pot diferir amb l'ordre de la gràfica però mostrant-lo així dona una visió més global del rendiment obtingut del cluster, al mateix temps es veu clarament la tendència de temps que adquireixen les diferents polítiques de balanceig analitzades respecte la configuració d'un servidor sol.

Les línies que es mostren a la figura 16 fan referencia a un dels resultats representatius de les polítiques de balanceig de HAProxy anteriorment provades i al cas del servidor sol.

A la figura 17 podem observar en més detall el codi de colors al que farem referencia per discutir els resultats obtinguts.

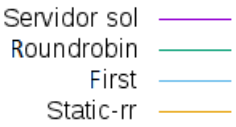


Fig. 17 Codi de colors i polítiques de balanceig.

A l'apendix en la secció A4 es troba una imatge més detallada del gràfic de la figura 16 amb les parts analitzades marcades.

Podem dividir el gràfic en tres parts diferents. La primera anirà de les 0 a les 4000 peticions, una segona franja que anirà de les 4000 a les 6000, i la tercera que anirà de les 6000 a les 10000.

Cal remarcar que aquests valors són el nombre de peticions que s'han respost en un temps determinat.

A la figura 18 es pot veure en més detall la part de la gràfica a la que es fa referencia, aquesta correspon a la primera part que anirà de les 0 a les 4000 peticions.

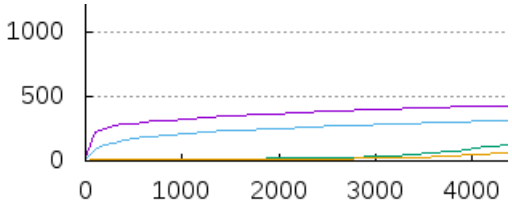


Fig. 18 Primera part del gràfic.

En aquesta franja de peticions podem observar com les polítiques roundrobin i static-rr gairebé responen immediatament a les peticions generades per l'usuari. En canvi les del servidor sol i la política first triguen més.

La política first, al assignar la connexió al primer servidor amb un espai buit, primer intentarà saturar el servidor1, i aquest, al trobar-se saturat, les assignarà al servidor2, per tant el temps inicialment és inferior al del servidor sol però no aconsegueix els temps de les polítiques roundrobin.

A la figura 19 es pot veure en més detall la part de la gràfica a la que es fa referencia, aquesta correspon a la segona part que anirà de les 4000 a les 6000 peticions.

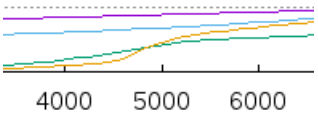


Fig. 19 Segona part del gràfic.

En aquesta franja de peticions podem observar com el servidor sol i la política first segueixen amb la mateixa tendència que a la franja anterior.

El canvi més significatiu el veiem a les polítiques de balanceig de static-rr (taronja) i roundrobin (verd).

Podríem considerar que fins a 5000 peticions static-rr les respon en un temps menor però al arribar al voltant de les 5000, les dues polítiques igualen el temps de resposta.

A partir de les 5000 peticions el temps de resposta és menor a la roundrobin.

A la figura 20 es pot veure en més detall la part de la gràfica a la que es fa referencia, aquesta correspon a la tercera part que anirà de les 6000 a les 10000 peticions.

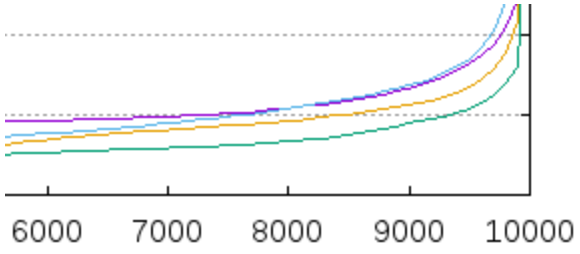


Fig. 20 Tercera part del gràfic.

En aquesta franja podem observar com la política roundrobin segueix per sota de la static-rr fins al total de peticions. En canvi les de first i les del servidor sol acaben ajuntant-se al mateix temps de resposta.

La política de balanceig first, al estar els dos servidors saturats, acaba trigant més per una petita part de les peticions totals (al voltant de 500 peticions).

Vist des d'un punt de vista global, si s'hagués de triar una política de balanç del servidor Proxy, pel cas de la web generada es triaria la roundrobin ja que si es mira el global d'execució la diferencia amb static-rr no compensa la part final de la gràfica en quan a rendiment.

7 CONCLUSIONS

Durant el transcurs de la realització del treball fi d'estudis s'ha pogut comprovar, al anar realitzant cada una de les diferents parts del mateix, com al haver una infinitat de configuracions i de possibles, tant d'entorns d'execució com de simulació, fan que la tasca de configuració sigui molt difícil. En la primera part es va haver d'establir un entorn d'una possible infraestructura real i es va simular el seu funcionament amb l'eina CORE.

A la segona part del treball, s'ha avaluat una part de la infraestructura proposada, en aquesta part, al aprofundir més en les eines proposades, s'ha posat d emanifest com existeixen una infinitat de possibles solucions per un mateix entorn i com en funció de les diferents característiques que tingui, haurà de variar si es vol que el seu funcionament sigui el més eficient possible.

En el cas de la nostra configuració les dos polítiques que han obtingut un major rendiment han estat la roundrobin i la static-rr.

Passada la fita de l'informe de progrés I es va formular un nou objectiu que no estava inicialment, el de implementar i externalitzar un sistema de bases de dades en mirall, no obstant ha sigut impossible de realitzar degut a que l'entorn, al estar executant 5 sistemes operatius en 5 màquines (4 virtuals més la màquina on s'executava tot), el rendiment es veia molt afectat al estar totes les màquines engegades, aquest fet ha fet que sigui impossible plantejar l'externalització de la base de dades.

A les taules 1 i 2 es poden observar les característiques del cluster i de les màquines virtuals respectivament.

Components cluster	Descripció
CPU	Intel i7 7700 HQ
Cache L1	128KB
Cache L2	1MB
Cache L3	6 MB
RAM	16GB DDR4 2400 Mhz
Interfície de xarxa	Realtek PCIe GBE
Espai d'emmagatzemat	250 GB M.2 PCI-E

Taula 1 Característiques de la màquina cluster.

Components màquines virtualitzades	Descripció
CPU	2 nuclis
RAM	4 GB
Interfície de xarxa	Gigabit Ethernet
Espai d'emmagatzemat	30GB

Taula 2 Característiques de les màquines virtuals.

Al aprofundir en les eines Apache bench i HAProxy es va decidir que totes les dades generades per part de ab seria convenient mostrar-les mitjançant l'eina Gnuplot per tal de exposar els resultats de la millor manera possible i així d'aquesta forma complimentar l'objectiu plantejar perdut a causa de la impossibilitat del mateix.

A partir d'aquest punt, es podrien tractar molts més temes de rendiment, per exemple, es podria modificar la web perquè l'intèrpret php hagués de treballar més.

D'aquesta manera segurament tota la configuració hauria de canviar.

També es podria realitzar una nova ampliació del clúster amb la maquina que seria l'encarregada de gestionar la base de dades.

Al veure les diferents polítiques de balanceig de les que disposa HAProxy també seria interesant modificar la

web perquè estableixi sessions, realitzar peticions amb IP diferents o be utilitzar cookies per tal de poder avaluar el rendiment de la web amb totes les polítiques de les que disposa HAProxy amb l'eina Apache bench.

La implementació de elements de seguretat podria ser un altre possible via de continuació del treball realitzat fins ara.

Personalment ha sigut una experiència molt satisfactòria a nivell personal i molt enriquidora en quant a coneixements i competències adquirides. Al principi del projecte no veia possible que tot s'acabés realitzant, i el fet de planificar i anar finalitzant etapes del mateix ho han fet possible.

Al mateix temps a dia de avui s'han vist consolidats molts coneixements que s'han anat adquirint durant el grau i que d'una altre manera potser hagués estat possible la execució del treball fi de grau.

AGRAÏMENTS

En primer lloc agraeixo al meu tutor Juan Carlos, per tot el suport proporcionat i per haver-me escollit per dur a terme el desenvolupament del projecte.

També agrair a la meva família, amics, parella i companys de treball la seva paciència, ja que han sigut part clau d'aquest anys de transcurs universitari, gràcies a la seva paciència i comprensió.

BIBLIOGRAFIA

[1] A. Peterson, Vmware vs. VirtualBox: wich is Better for desktop Virtualization? [Online]. Available: http://technologyadvice.com/blog/information-technology/vmware-vs-virtualbox/

[2] Grupo Espiral Php, modelo espiral. [Online]. Avaliable: http://modeloespiral.blogspot.com.es/

[3] Comon Open Research Emulator, CORE. [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/core

[4] Apache HTTP Server, Apache . [Online]. Available: https://httpd.apache.org/ABOUT_APACHE.html

[5] MySQL Documentation, MySQL. [Online]. Available: https://dev.mysql.com/doc/

[6] Bringing MySQL to the web, MySQL . [Online]. Available: https://www.phpmyadmin.net/

[7] The reliable, High Performance TCP/HTTP Load Balancer, HAProxy. [Online]. Available: http://www.haproxy.org/

[8] Gnuplot, Gnuplot. [Online]. Available: http://www.gnuplot.info/

APÈNDIX

A1. FUNCIONAMENT DE L'INFRAESTRUCTURA GENERADA I CONFIGURADA AMB CORE.

Empresa A – Server:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host1:/tmp/pycore.44395/Host1.conf# ping 192.168.6.10
PING 192.168.6.10 (192.168.6.10) 56(84) bytes of data.
64 bytes from 192.168.6.10: icmp_seq=1 ttl=62 time=0.070 ms
64 bytes from 192.168.6.10: icmp_seq=2 ttl=62 time=0.126 ms
64 bytes from 192.168.6.10: icmp_seq=3 ttl=62 time=0.095 ms
64 bytes from 192.168.6.10: icmp_seq=4 ttl=62 time=0.077 ms
64 bytes from 192.168.6.10: icmp_seq=5 ttl=62 time=0.090 ms
^C
--- 192.168.6.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4054ms
rtt min/avg/max/mdev = 0.070/0.091/0.126/0.022 ms
root@Host1:/tmp/pycore.44395/Host1.conf#
```

Empresa B – Server:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host6:/tmp/pycore.44395/Host6.conf# ping 192.168.6.10
PING 192.168.6.10 (192.168.6.10) 56(84) bytes of data.
64 bytes from 192.168.6.10: icmp_seq=1 ttl=63 time=0.057 ms
64 bytes from 192.168.6.10: icmp_seq=2 ttl=63 time=0.057 ms
64 bytes from 192.168.6.10: icmp_seq=3 ttl=63 time=0.060 ms
64 bytes from 192.168.6.10: icmp_seq=4 ttl=63 time=0.065 ms
^C
--- 192.168.6.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3048ms
rtt min/avg/max/mdev = 0.057/0.059/0.065/0.010 ms
root@Host6:/tmp/pycore.44395/Host6.conf#
```

Host4 – Server:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host4:/tmp/pycore.44395/Host4.conf# ping 192.168.6.10
PING 192.168.6.10 (192.168.6.10) 56(84) bytes of data.
64 bytes from 192.168.6.10: icmp_seq=1 ttl=62 time=0.048 ms
64 bytes from 192.168.6.10: icmp_seq=2 ttl=62 time=0.076 ms
64 bytes from 192.168.6.10: icmp_seq=3 ttl=62 time=0.065 ms
64 bytes from 192.168.6.10: icmp_seq=4 ttl=62 time=0.065 ms
^C
--- 192.168.6.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.048/0.063/0.076/0.012 ms
root@Host4:/tmp/pycore.44395/Host4.conf#
```

A2. PETICIONS WEB AL SERVIDOR

Empresa A – Server:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host1:/tmp/pycore.35623/Host1.conf# wget 192.168.6.10
--2017-11-09 23:43:23--  http://192.168.6.10/
Conectando con 192.168.6.10:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 286
Grabando a: "index.html"

index.html          100%[===================>]     286  --.-KB/s    in 0s

2017-11-09 23:43:23 (76,7 MB/s) - "index.html" guardado [286/286]

root@Host1:/tmp/pycore.35623/Host1.conf#
```

Empresa B – Server:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host7:/tmp/pycore.35623/Host7.conf# wget 192.168.6.10
--2017-11-09 23:44:36--  http://192.168.6.10/
Conectando con 192.168.6.10:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 286
Grabando a: "index.html"

index.html          100%[===================>]     286  --.-KB/s    in 0s

2017-11-09 23:44:36 (75,5 MB/s) - "index.html" guardado [286/286]

root@Host7:/tmp/pycore.35623/Host7.conf#
```

Host4 – Server:

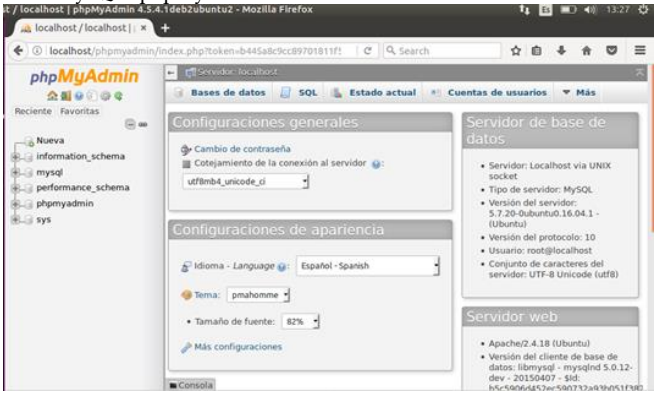
```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@Host4:/tmp/pycore.35623/Host4.conf# wget 192.168.6.10
--2017-11-09 23:45:27--  http://192.168.6.10/
Conectando con 192.168.6.10:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 286
Grabando a: "index.html"

index.html          100%[===================>]     286  --.-KB/s    in 0s

2017-11-09 23:45:27 (71,9 MB/s) - "index.html" guardado [286/286]

root@Host4:/tmp/pycore.35623/Host4.conf#
```

A3. COMPROVACIÓ DE LA INFRAESTRUCTURA D'UN SERVIDOR SOL

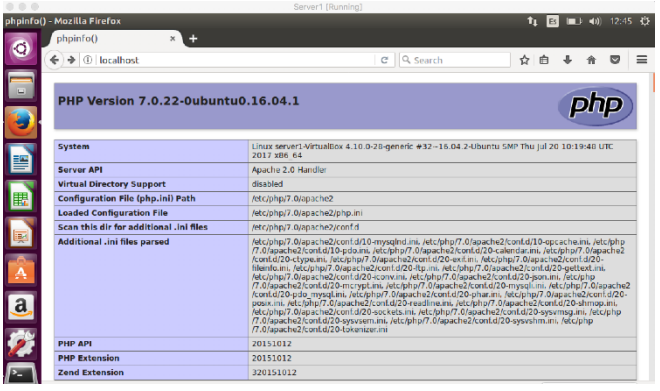
Funcionament de la xarxa:

```
user@user-VirtualBox: ~
user@user-VirtualBox:~$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.718 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.677 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.555 ms
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.555/0.650/0.718/0.069 ms
user@user-VirtualBox:~$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.716 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.536 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.653 ms
^C
--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2043ms
rtt min/avg/max/mdev = 0.536/0.635/0.716/0.074 ms
user@user-VirtualBox:~$
```

Versio MySQL i phpMyAdmin:

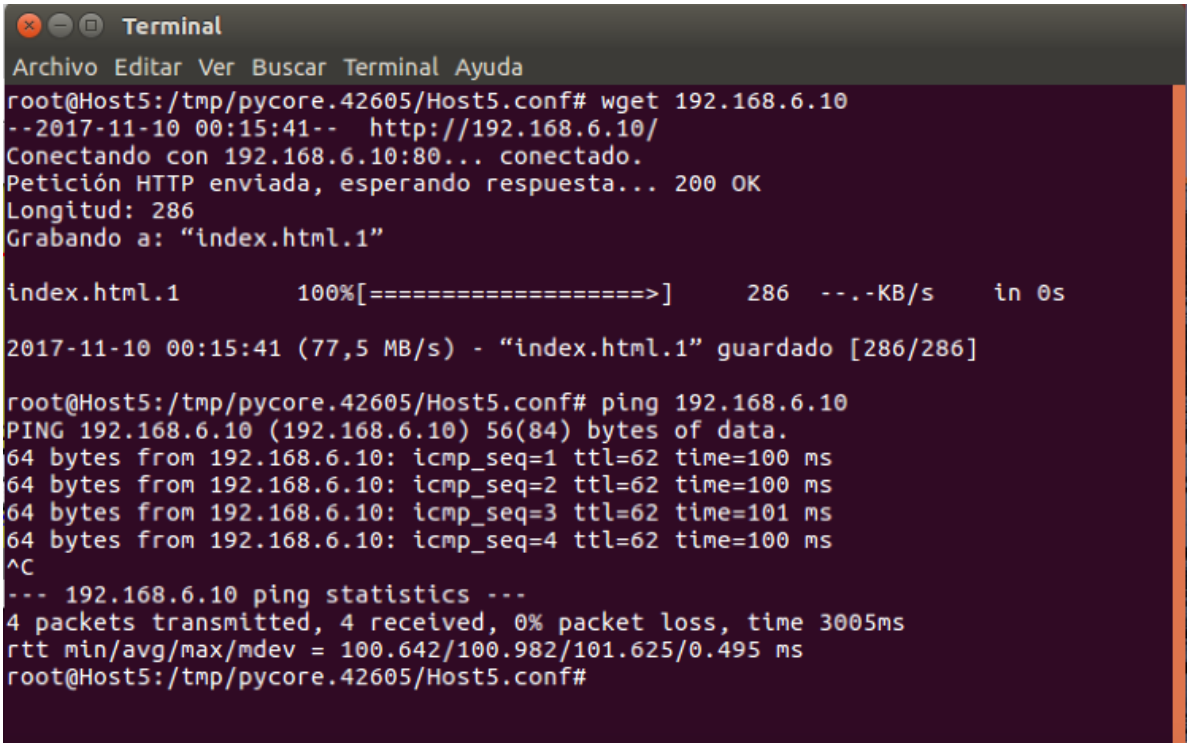


Versio mòdul php d'Apache2:

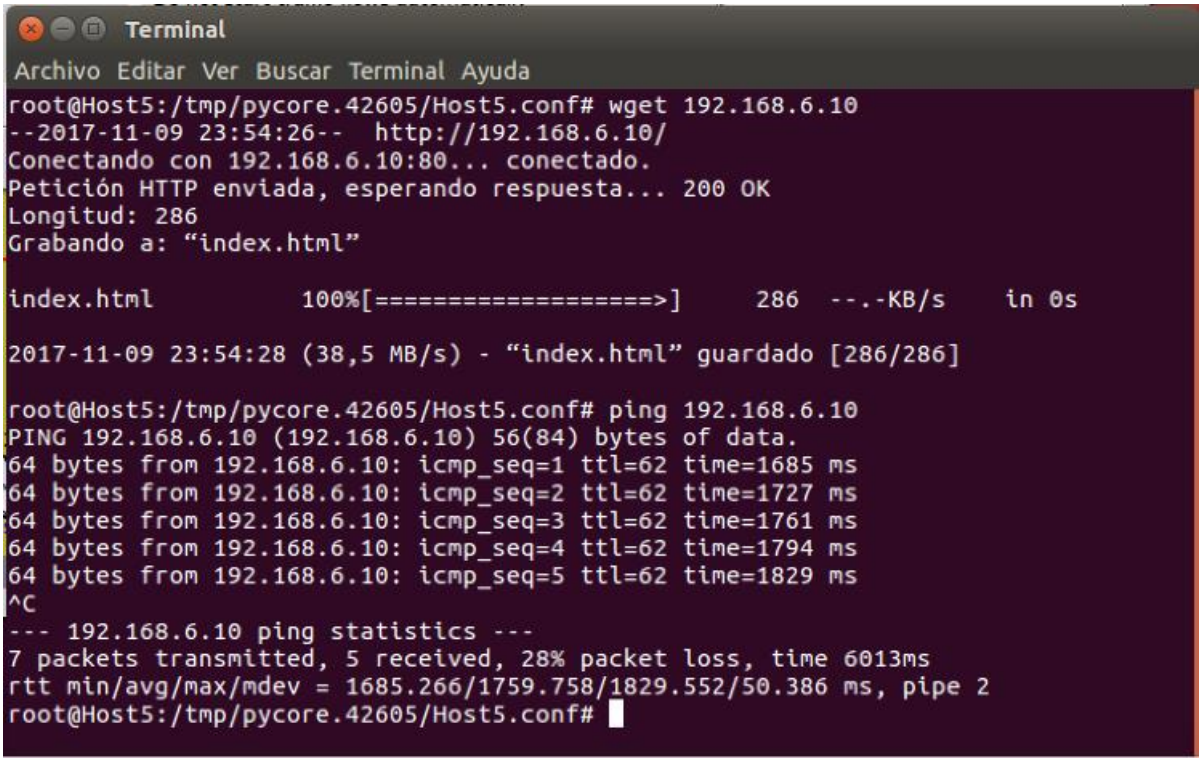


A3. PROVES DE RENIDMENT AMB L'EINA CORE.

Prova sense tràfic simulat:



Prova amb tràfic simulat:



A4. GRÀFICA RESUM DE PROVES DE RENDIMENT.