Deboop: una distribució de linux per a *Big Data*

Treball de Final de Carrera

Alfred Gil

18 de març de 2014

Índex

1 Introducció

Des de ja fa uns anys, el terme *Big Data* ha vingut apareixent cada cop més en qualsevol article o comunicació relacionada amb les tecnologies de la informació. No obstant això, en una primera ullada a diverses pàgines web no s'acaba d'obtenir molta informació al respecte, i fa la impressió que moltes empreses s'han llençat a l'ús d'aquesta nova tecnologia sense acabar de tenir clar les seves bases. Per tant, la primera pregunta que caldria fer-nos és: què és *Big Data* i com funciona un clúster dedicat a aquest tipus de metodologia?

Un cop tinguem clars els conceptes clau i les eines relacionades amb aquest nou paradigma, veurem que dins d'aquestes eines, Hadoop [1] és un dels models més utilitzats actualment. Vist l'interès de les empreses en instal·lar clústers dedicats al tractament de dades amb Hadoop, sembla una bona idea generar una distribució de linux que automatitzi totes les tasques associades. Aquesta distribució ha de permetre fer el desplegament sobre un clúster i realitzar una configuració bàsica del mateix de la forma més desatesa possible.

2 Objectius

L'objectiu principal del treball de final de carrera és la creació d'una distribució de Linux que inclogui les eines més útils per tal d'instal·lar un clúster que permeti explotar *Big Data*, i que a més ho faci de la manera més automatitzada possible. A continuació expliquem amb una mica més de detall la planificació prevista per a la realització del treball.

2.1 Documentació

Com a pas previ a l'abordament de l'objectiu del treball de final de carrera, cal una feina de recollida d'informació per tal d'assolir els coneixements necessaris. En aquest cas, és necessari entendre què són cadascun dels components que conformen un clúster per a *Big Data*, com es configuren i com interaccionen entre si. Per a tal fi, és indispensable cercar literatura respecte els següents temes:

- *Big Data*
- MapReduce
- Hadoop
- HDFS
- Creació de distribucions Linux
- Clonació de nodes de càlcul

> Aquesta fase està prevista fer-la en tres setmanes i un cop recollida la informació del elements clau, es començarà el desenvolupament del treball pràctic.

2.2 Distribució de linux

En aquesta fase s'instal·larà un clúster de Hadoop, fent servir una sèrie de màquines virtuals com a model d'un clúster físic, i una distribució de linux personalitzada com a instal·lador. L'explicació de l'objectiu sembla curta, però preveiem que caldrà dedicar-hi molts esforços per tal d'assolir una configuració que sigui estable. Cal muntar tota la xarxa de màquines virtuals, i fer una primera instal·lació manual per tal de desplegar-hi Hadoop. Amb això podrem conèixer tota la paqueteria utilitzada i la configuració necessària, per posteriorment aplicar-la a la nostra distribució personalitzada. Caldrà seleccionar una distribució Linux de base i cercar una eina que permeti crear la distribució. Finalment, fer la instal·lació amb el CD creat i comprovar que tot funciona com havíem planejat.

> En cas que es compleixin les previsions i no sorgeixin problemes molt greus, és previst completar aquest objectiu en el termini d'unes vuit setmanes.

2.3 Memòria i presentació

Malgrat durant la realització dels apartats anteriors es pretén haver anant documentant totes les tasques que es van realitzant, caldrà dedicar un temps a repassar la memòria final i preparar la presentació.

La previsió és dedicar-hi quatre setmanes per a completar aquestes feines.

3 Conceptes clau

Tal i com hem comentat anteriorment, per començar a treballar cal tenir clar els conceptes sobre els quals ens basarem per a realitzar el projecte. En aquesta secció es descriuran de forma breu les idees més importants relacionades amb *Big Data* i Hadoop.

3.1 *Big Data*

Segons la Vikipedia *Big Data* és un terme aplicat a conjunts de dades que superen la capacitat del programari habitual per ser capturats, gestionats i processats en un temps raonable [2]. Així, malgrat el terme encunyat només faci menció a la grandària de les dades, aquesta no és l'única variable important en aquests conjunts de dades. Quan parlem d'aquesta tecnologia també és molt important la velocitat a la que podem processar aquestes dades i la varietat d'aquestes. És el que es coneix com el concepte de les 3V's (volum, varietat i velocitat), introduït per Laney [3]

Volum Sembla evident que un dels reptes més grans és el d'haver de tractar amb volums de dades cada vegada més grans, i les dificultats que sorgeixen quan estem al nivell dels petabytes.

Velocitat Aquesta variable es refereix a la necessitat que tenen algunes empreses per tractar les dades en condicions cada cop més properes al temps real.

Varietat Les dades a tractar no s'ajusten al model relacional, sinó que poden arribar a ser força desestructurades.

El model *Big Data* està molt relacionat amb les bases de dades NoSQL (not only SQL). De fet, va ser la mateixa persona, Eric Evans, qui va proposar ambdós noms per a aquests conceptes connectats entre si [4]. NoSQL es va pensar per a tractar un gran volum d'informació de manera ràpida i eficient. És per això que els gestors d'aquestes bases de dades guarden el màxim d'informació possible en memòria, i utilitzen el disc per a fer les dades persistents. Funcionen amb un bon rendiment en PC's (sense necessitat de fer servir grans supercomputadors), i permeten escalar la solució horitzontalment afegint nous nodes en calent (sense necessitat d'aturar i reiniciar el sistema).

El gran volum de dades que s'està generant actualment, i que com hem vist és un dels motius de l'aparició d'aquest nou model de tractament de dades, prové de diferents fonts. Moltes són humanes, com les introduïdes a les webs socials, i poden ser de qualsevol tipus (text, fotos, vídeos). Però d'altres provenen de màquines com poden ser logs, xarxes de sensors, transaccions comercial o inclús les produïdes en aparells científics (per exemple el LHC). Per a les empreses pot arribar a ser molt important extreure valor d'aquesta ingent quantitat de dades, i com hem comentat, en un temps adient.

3.2 Hadoop

El problema principal del tractament de grans volums de dades es redueix a que, malgrat la capacitat dels discos ha augmentat enormement en els darrers anys, la velocitat de transferència de dades dels mateixos no ho ha fet al mateix ritme [5]. Això pot provocar que llegir les dades d'un disc pugui trigar diverses hores. La solució passa per llegir de diversos discos a la vegada, en paral·lel, el que proporciona una velocitat d'accés a les dades més elevada.

Aquesta solució pot comportar dos problemes. El primer és que quan més discos afegim en paral·lel, més probabilitats hi han que es produeixi un error de hardware. El segon és que en molts dels anàlisi que es realitzen és habitual que es vulguin combinar dades de diferents discos, i això no és una tasca trivial. Una solució a aquests dos tipus de problemàtiques és la que ens proporciona Hadoop [1]. Per una banda, ofereix un sistema d'emmagatzematge fiable mitjançant HDFS (Hadoop Distributed FileSystem) i per altra banda, un sistema d'anàlisi de dades distribuïdes a través del model de programació MapReduce.

3.3 HDFS

HDFS [6] és el sistema de fitxers distribuïts que utilitza Hadoop. Igual que els sistemes de fitxers de disc, utilitza el concepte de bloc (unitat mínima de lectura/escriptura), que per defecte es de 64MB. Això permet encabir fitxers que sobrepassin la grandària d'un sol disc, ja que es poden escriure blocs a diferents discs. De fet, cada bloc està replicat en diferents nodes (tres per defecte), per tal de proporcionar tolerància a fallades. A més, el fet que el subsistema d'emmagatzematge tracti amb blocs en comptes de fitxers simplifica la gestió, ja que per exemple, les metadades no cal que siguin guardades en el bloc, i la seva gestió pot ser encomanada a un altre sistema.

Hi han dos tipus de nodes diferents: un *namenode* i diversos *datanodes*. El *namenode* gestiona l'espai de noms del sistema de fitxers, les metadades dels fitxers i directoris, i controla la localització de tots els blocs d'un fitxer distribuïts en diferents *datanodes*. Els *datanodes* guarden els blocs del sistema de fitxers i comuniquen al *datanode* un llistat dels blocs que contenen. Si es perd el *namenode*, es perden tots els fitxers del sistema, doncs no hi ha manera de reconstruir els fitxers a partir dels blocs als *datanodes*. Per això, cal que aquest node sigui resistent a fallades. A partir de les versions 2.x es permet afegir més d'un *namenode*, on cada un d'ells gestiona una part del sistema de fitxers i això fa la solució més escalable. També en aquestes versions es pot configurar HDFS en alta disponibilitat, amb dos *namenodes* disposats en actiu-passiu.

3.4 MapReduce

MapReduce va ser inventat per enginyers de Google com un sistema per a construir índexs de cerca [7]. És un model de programació que abstrau el problema de lectura/escriptura a diversos discs i el transforma en un problema de càlcul sobre conjunts de claus i valors. Està basat en la definició de dues funcions, una funció *map* i una *reduce*, cadascuna de les quals defineix una correspondència entre conjunts de parells clau-valor.

Però no només és el model de programació el que el fa tant interessant, sinó també la seva implementació, orientada al càlcul paral·lel massiu, amb tolerància

a errades i balanceig de càrrega, que es pot executar en clústers de PC's (no calen entorns d'HPC més exigents). El principal atractiu és que en augmentar la grandària del clúster, disminueix linealment la velocitat de càlcul. Això no sol passar amb consultes SQL tradicionals. També en comparació amb els SGBDR, MapReduce treballa millor amb dades no estructurades, ja que està dissenyat per a interpretar les dades en temps de procés.

Finalment, cal comentar algunes raons del bon rendiment obtingut amb el model MapReduce. La principal és que utilitza localitat en les dades, és a dir, intenta executar les tasques de càlcul al node on resideixen les dades, de forma que l'accés sigui més ràpid. A més, una tasca no té cap tipus de dependència amb d'altres tasques. Això ajuda a que si una tasca falla perquè algun node ha caigut, es pot reenviar automàticament cap a un altre node.

3.4.1 Flux de treball

Un treball MapReduce està format per les seves dades d'entrada, el programa MapReduce a executar i informació de configuració [5]. Hadoop executa el treball dividint-lo en tasques de dos tipus; *maps* i *reduces*. Existeixen dos tipus de nodes que controlen el procés d'execució; un *jobtracker* i els *tasktrackers*. El primer és l'encarregat de coordinar tot el treball, enviant les tasques als *tasktrackers*, i aquests últims reporten el seu estat al *jobtracker*. Si alguna tasca falla, el *jobtracker* s'encarrega de tornar-la a enviar a un altre *tasktracker*.

Hadoop divideix també les dades d'entrada en *splits*, que són trossos de la mateixa grandària, i crea una tasca *map* per a cadascun d'ells. Aquesta tasca executa la funció *map* per a cada registre del *split*. La grandària dels *splits* ha de ser un compromís entre suficientment petita (amb el que s'aconsegueix un millor balanceig de la càrrega de treball entre els nodes), i no tan petita com perquè la sobrecàrrega de la gestió dels mateixos *splits* i la creació de les tasques *map* dominin sobre el càlcul.

Hadoop intenta córrer cada tasca *map* dins d'un dels tres nodes on resideixen les rèpliques que HDFS té d'un *split*, i si tots aquests nodes estan ocupats l'intenta enviar a un altre node del mateix rack. Les tasques *map* escriuen a disc local (i no al HDFS), ja que són fitxers intermedis que després consumiran les tasques *reduce*.

Normalment, una tasca *reduce* no gaudeix de localitat en les dades, ja que les seves dades d'entrada solen ser les sortides de tots els *mappers*, i per tant aquestes sortides han de ser transferides per xarxa des de tots els *mappers* cap al node on s'executi la tasca *reduce*. Finalment, la sortida d'aquesta tasca *reduce* es guarda a l'HDFS, amb una rèplica local i dues rèpliques més en nodes fora del rack. En el cas que hi hagin diversos *reducers*, cada tasca *map* particiona la seva sortida, creant una partició per a cada tasca *reduce*.

4 Sistema operatiu orientat a *Big Data*

En aquesta secció s'explica la solució que s'ha dissenyat per tal d'aconseguir un sistema operatiu orientat a l'execució de treballs amb metodologies *Big Data*. En primer lloc s'exposarà el plantejament global i les eines utilitzades, per posteriorment entrar en detall en el desenvolupament de la distribució.

4.1 Disseny global de la solució

L'objectiu final del sistema operatiu a construir és la comoditat de l'usuari al qual va destinat. Com a comoditat entenem el fet de que la instal·lació del sistema sigui el més automatitzada possible i contingui totes les eines que l'administrador necessitarà. Com explicarem posteriorment, Cloudera [8] és el paquet de software més complet que existeix actualment, que inclou Apache Hadoop i tota una extensa col·lecció d'eines que giren al seu voltant. No obstant això, Cloudera ha d'instal·lar-se sobre un sistema operatiu ja existent. Per tant, per poder disposar de l'aplicació de Hadoop en un clúster, cal fer-ho en dues passes, una primera instal·lació i configuració del clúster, i posteriorment el desplegament de la solució de Hadoop. Amb la nostra idea de simplificar i automatitzar al màxim totes aquestes tasques, el que s'ha creat és una distribució de linux que integra tant la configuració inicial del clúster com la instal·lació de les eines per a *Big Data*. En una primera fase s'ha treballat en un entorn virtualitzat, amb l'eina Oracle VirtualBox [9], sobre el qual s'ha fet la instal·lació i desplegament dels nodes de càlcul.

El producte final consta d'un CD d'instal·lació (o una imatge iso) creat amb simple-cdd [10], que instal·la el sistema operatiu del node de gestió, i el deixa configurat adientment. Posteriorment, s'instal·la el sistema operatiu dels nodes de càlcul, automàticament i amb tan sols arrencar-los, gràcies a l'eina FAI [11]. Finalment, la configuració i desplegament del clúster de Hadoop es fa via interfície web des del node de gestió, mitjançant Cloudera Manager [12].

A continuació s'exposa en detall les eines utilitzades i la seva configuració per tal d'arribar al resultat final.

4.2 VirtualBox

El primer que necessitem per desenvolupar el nostre producte és la infraestructura de hardware adient. En principi aquesta hauria de constar d'un node de gestió i diversos nodes de càlcul, que conformaran el clúster de Hadoop. Com hem comentat anteriorment, en una primera fase, hem simulat aquesta infraestructura mitjançant la creació de diverses màquines virtuals amb l'aplicació VirtualBox d'Oracle [9]. Aquest és un programari de virtualització que permet instal·lar sistemes operatius addicionals (anomenats convidats) a dins d'un sistema operatiu donat (anomenat amfitrió). D'aquesta forma, la creació de diversos sistemes convidats permet emular la disponibilitat d'un clúster amb una certa quantitat de nodes. La versió utilitzada ha estat la 4.3.12. Cal destacar la potencia que proporciona la línia de comandes que ofereix VirtualBox a l'hora de crear scripts que permetin crear o gestionar màquines virtuals, com veurem més endavant.

4.3 Debian

Com a sistema operatiu base, utilitzarem Debian [13], en la seva versió 7.5.0. Debian és una de les distribucions de linux més populars, tant per ús personal com per a servidors d'Internet. Va néixer com una aposta per separar en les seves versions el programari lliure del programari no lliure, i encara que en la versió original no es trobi firmware privatiu, sempre es pot instal·lar via altres repositoris. Existeixen tres branques principals de desenvolupament de la

distribució. En la branca *unstable* es realitza tot el desenvolupament actiu, i una vegada que els paquets ja no contenen molts errors, aquests passen a la branca *testing*. Quan arriba a un nivell acceptable d'errors, s'allibera la versió *stable*, que compta amb el suport de l'equip de seguretat de Debian i és la recomanada per al seu ús en producció.

4.4 Simple-cdd

La creació del CD d'instal·lació de la distribució es realitzarà amb l'eina simple-cdd [10]. Tot i ser una aplicació una mica limitada, és la més senzilla per a la obtenció d'un instal·lador de debian personalitzat. Simple-cdd utilitza l'eina debian-cd [14] juntament amb altres aplicacions de *mirroring* per tal de crear la imatge del CD. Amb la finalitat de preconfigurar la instal·lació (és a dir, realitzar la instal·lació del node de gestió de la manera més desatesa possible) se li passa a simple-cdd la informació necessària en forma de fitxers de text. D'aquesta manera se li diuen els paquets a instal·lar, i se li proporciona informació de *preseeding* a debconf [15] (se li donen dades al configurador de la instal·lació de debian per tal que no les pregunti mentre s'està instal·lant). Aquesta informació pot ser tant l'idioma que volem com el particionat del disc dur del node, entre d'altres.

4.5 FAI

FAI (Fully Automatic Installation) [11] és un sistema no interactiu per a instal·lar, personalitzar i gestionar sistemes linux i configuracions de programari en diversos nodes d'un clúster. És una eina que permet el desplegament massiu de sistemes linux, de forma que en uns minuts es pot tenir un clúster instal·lat i configurat segons les nostres necessitats. La versió utilitzada és la 4.2, i una de les seves característiques que ens serà útil és la seva capacitat de treballar amb màquines virtuals.

4.6 Cloudera Manager

Cloudera Manager [12] ens proveeix de les funcionalitats necessàries per a una gestió eficient del clúster de Hadoop, com poden ser el desplegament automàtic de programari sobre els nodes de càlcul, administració centralitzada, monitorització i eines de diagnòstic. Aquest gestor, que s'accedeix mitjançant interfície web, permet desplegar un clúster en qüestió de minuts.

5 Implementació de la solució

En aquest apartat s'explica el desenvolupament de la solució plantejada, adjuntant els scripts necessaris per al seu funcionament.

5.1 Creació de la imatge del CD

La nostra distribució personalitzada la hem anomenat Deboop (de Debian i Hadoop), i per tal de crear-la hem fet servir l'eina simple-cdd [10]. Prèviament, s'ha instal·lat i configurat una versió mínima de debian, a partir de la imatge

debian-7.5.0-amd64-netinst.iso [16]. Aquesta versió de debian ens aporta un sistema operatiu amb el mínim que cal per a poder ser funcional. A partir d'aquí hem afegit tots els paquets que ens han estat necessaris i hem realitzat tota la configuració adient per a obtenir un node amb Cloudera Manager operatiu. Un cop obtingut aquest node, hem configurat simple-cdd per tal d'obtenir una imatge de debian que acabi instal·lant un node de les mateixes característiques.

En primer lloc, instal·lem simple-cdd.

```
sudo apt-get install simple-cdd
```

Amb simple-cdd podem definir diversos "profiles" que es poden escollir en temps d'instal·lació. Aquests perfils permeten obtenir diferents tipus d'entorns del nostre sistema operatiu. Així, podríem crear un perfil per a tenir el gestor de finestres GNOME, o un perfil "laptop" que carregués els paquets típics que es fan servir per a gestionar un portàtil. Els perfils es defineixen amb diversos fitxers del tipus nom_de_perfil.* i que cal situar dins d'un directori anomenat *profiles*.

- Un fitxer .description que conté una línia amb la descripció del perfil.
- Un fitxer .packages amb un llistat dels paquets que s'instal·laran automàticament si es selecciona el perfil.
- Un fitxer .downloads amb un llistat de paquets que s'emmagatzemaran dins el CD d'instal·lació, però que a priori no s'instal·laran.
- Un fitxer .preseed que conté la informació de *preseeding* per a les preguntes de debconf (tal i com hem dit a la secció 4.4, aquest fitxer serveix per automatitzar la instal·lació de debian, passant-li les respostes a l'instal·lador per tal que no les pregunti durant el procés d'instal·lació).
- Un fitxer .postinst que conté un script que s'executarà al final del procés d'instal·lació.
- Un fitxer .conf que permet canviar alguns paràmetres de simple-cdd basant-se en els perfils a incloure en la imatge.

Creem l'estructura de directoris de treball, amb un directori que anomenarem *deboop*, i el subdirectori *profiles* a dins. Posteriorment, ens situarem dins d'aquest directori *profiles* i definirem el perfil *deboop* mitjançant la creació dels fitxers deboop.conf, deboop.packages i deboop.preseed.

```
mkdir -p $HOME/deboop/profiles
cd $HOME/deboop/profiles
```

El fitxer de configuració principal, deboop.conf el podem veure al Fitxer 1. Amb la línia 1 li diem el nom del perfil i amb la línia 2 se li diu que a l'hora d'instal·lar no ens pregunti quins perfils volem, sinò que escolleixi aquest perfil automàticament. De les línies 3 a la 5 li passem la informació del mirror de debian a utilitzar per baixar els paquets. La línia 6 defineix la variable *all_extras*, que permet copiar un fitxer dins del directori /simple-cdd del CD d'instal·lació. En aquest fitxer podem posar tota la configuració de post-instal·lació que necessitarem per a la nostra distribució. Tal i com està pensat en aquest projecte,

aquest és un arxiu comprimit (firstboot.tgz) que conté tots els fitxers de configuració i que hem situat a $HOME/deboop/profiles. La línia 7 ens permet personalitzar la imatge que mostra el CD d'instal·lació quan arrenca, canviant la de debian per una que haguem creat nosaltres. En el nostre cas, hem creat un logo en un fitxer anomenat deboop.png i que també hem situal al directori de treball $HOME/deboop/profiles. Per últim amb les línies 8 a 10 podem modificar el nom de la iso que es crearà, la versió de la nostra distribució i l'etiqueta del CD d'instal·lació.

Fitxer 1: deboop.conf

```
1  profiles="deboop"
2  auto_profiles="deboop"
3  server="ftp.es.debian.org"
4  debian_mirror="http://$server/debian/";
5  wget_debian_mirror="ftp://$server/debian/";
6  all_extras="$all_extras $simple_cdd_dir/profiles/firstboot.tgz"
7  export SPLASHPNG="$simple_cdd_dir/profiles/deboop.png"
8  export CDNAME=deboop
9  export DEBVERSION="0.1"
10 export DISKINFO="Deboop GNU/Linux 0.1 $(date --utc +%Y%m%d)"
```

El següent fitxer que necessitem, deboop.packages (Fitxer 2), conté el nom de tots els paquets que necessitarem i que simple-cdd baixarà i inclourà al CD. En el nostre cas, ens caldrà:

openssh-server Un servidor de ssh per poguer accedir al node de login remotament [17].

squid Un proxy, que ens permetrà accedir a internet des de la xarxa interna (nodes de càlcul) [18].

apt-cacher-ng Un proxy-cache especialitzat en paqueteria de linux, i pensat en un principi per a debian [19].

rsync Per sincronitzar arxius entre nodes [20].

fai-quickstart Instal·lador de FAI, que permetrà desplegar les imatges de sistema operatiu a tots els nodes de càlcul de forma desatesa.

ntp Per sincronitzar la data entre tots els nodes [21].

cloudera-manager-daemons Conté els dimonis per iniciar els serveis del manager de cloudera.

postgresql Gestor de bases de dades que utilitza cloudera per defecte [22].

cloudera-manager-server Serveis del manager de cloudera.

cloudera-manager-server-db-2 Base de dades postgresql embebida de cloudera.

dialog Eina per a mostrar finestres amb informació, amb la que podrem veure el progrés de la instal·lació del node de login [23].

La resta de paquets que hem afegit al llistat són paquets de sistema necessaris per al bon funcionament de la resta de serveis.

Fitxer 2: deboop.packages

```
1  openssh−server
2  squid
3  apt−cacher−ng
4  rsync
5  fai−quickstart
6  syslinux−common
7  libproc−daemon−perl
8  ntp
9  openssl
10 cloudera−manager−daemons
11 python
12 lsb−release
13 postgresql
14 cloudera−manager−server
15 cloudera−manager−server−db−2
16 oracle−j2sdk1.7
17 dialog
18 bc
```

Finalment, el Fitxer 3 mostra les dades que *debconf* utilitzarà per fer la instal·lació desatesa i que hem inclòs al fitxer deboop.preseed. A continuació comentarem les línies de configuració més rellevants, tot i que els comentaris del fitxer ja les fan autoexplicatives. Amb la línia 5 escollim la interfície de xarxa per defecte del node de login on configurarem la xarxa pública. Les línies 8 i 9 declaren el nom de host i de domini del node, i amb la línia 15 es carrega firmware addicional en cas que sigui necessari. De les línies 17 a la 25 creem l'usuari root i un usuari administrator i els hi donem un password, que apareix xifrat amb *md5* dins aquest fitxer de text. En el nostre cas, el password de root és *toor*, i el password de administrator és *admin*. La línia 32 habilita l'ús d'ntp per sincronitzar el temps durant la instal·lació. A continuació es defineix el particionat de disc. En una primera aproximació suposem que només tenim un disc en el node de login. Si no fos així, caldria revisar aquest fitxer per tenir en compte configuracions amb RAID o LVM, per exemple. Amb la línia 41 li diem al particionador que només faci una partició per a tots els fitxers, la configuració més senzilla. Les línies 44 a 47 fan que el particionat es pugui fer sense demanar confirmació a cada pas. De les línies 60 a la 68 s'assignen els repositoris d'on es baixaran els paquets de software i les actualitzacions de seguretat.

La línia 86 és potser la més important per a la creació del nostre CD d'instal·lació, i mereix una especial explicació. Amb l'ordre *late_command* es permet l'execució de comandes just abans que la instal·lació finalitzi del tot, quan ja disposem del directori arrel de la nostra futura distribució, però que en temps d'instal·lació està muntat sota una partició anomenada /target. És a dir, tot el que copiem o fem als directoris situats sota /target, és com si ho estiguéssim fent al sistema de fitxers del sistema operatiu definitiu del node de login. La estratègia que hem seguit és afegir un *late_command* que copiarà dins el node

Fitxer 3: deboop.preseed

```
#### Contents of the preconfiguration file (for wheezy)

### Network configuration
# To pick a particular interface
d-i netcfg/choose_interface select eth0

# hostname and domain names
d-i netcfg/get_hostname string login
d-i netcfg/get_domain string deboop.net

# Disable that annoying WEP key dialog.
d-i netcfg/wireless_wep string

# If non-free firmware is needed, try to load it, without prompting
d-i hw-detect/load_firmware boolean true

### Account setup
# Root password, encrypted using an MD5 hash
d-i passwd/root-password-crypted password $1$Ys4dsB9V$ZKc27rms6qGl25Te3TQB
    /0

# To create a normal user account.
d-i passwd/user-fullname string Administrator
d-i passwd/username string administrator
# Normal user's password, encrypted using an MD5 hash
d-i passwd/user-password-crypted password
    $1$xqdbRVMV$z84YIFh4KPXUOL8d6Tajo0

### Clock and time zone setup
# Controls whether or not the hardware clock is set to UTC.
d-i clock-setup/utc boolean true

# Controls whether to use NTP to set the clock during the install
d-i clock-setup/ntp boolean true

### Partitioning
# Method to use.
# - regular: use the usual partition types for your architecture (no LVM, no encryption)
d-i partman-auto/method string regular

# Predefined partitioning recipe:
# - atomic: all files in one partition
d-i partman-auto/choose_recipe select atomic

# Automatically partition without confirmation.
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
```

Fitxer 3 (Cont.): deboop.preseed

```
## Controlling how partitions are mounted
# "traditional" to use traditional device names, not UUID
d-i partman/mount_style select traditional

### Base system installation
# To install recommended packages by default.
d-i base-installer/install-recommends boolean true

# The kernel image (meta) package to be installed.
d-i base-installer/kernel/image string linux-image-amd64

### Apt setup
# To install non-free and contrib software.
d-i apt-setup/non-free boolean true
d-i apt-setup/contrib boolean true
# Don't use a network mirror.
d-i apt-setup/use_mirror boolean false
# Select which update services to use; define the mirrors to be used.
d-i apt-setup/services-select multiselect security, updates
d-i apt-setup/security_host string security.debian.org

### Package selection
# Upgrade packages after debootstrap.
d-i pkgsel/upgrade select safe-upgrade

# Not reporting back what software you have installed.
popularity-contest popularity-contest/participate boolean false

### Finishing up the installation
# Enable regular virtual consoles (VT1-VT6) in /etc/inittab during installations from
    serial console.
d-i finish-install/keep-consoles boolean false

# Avoid that last message about the install being complete.
d-i finish-install/reboot_in_progress note

#### Advanced options
### Running custom commands during the installation
d-i preseed/late_command string cp /cdrom/simple-cdd/firstboot.tgz /target/tmp; in
    -target tar xzf /tmp/firstboot.tgz -C /root; in-target mv /root/firstboot/
    firstboot /etc/init.d; in-target chmod u=rwx,go=rx /etc/init.d/firstboot; in-
    target update-rc.d firstboot defaults;
```

de login un fitxer anomenat firstboot.tgz, que tal i com hem definit al fitxer deboop.conf estarà situat al directori /simple-cdd del CD d'instal·lació, i contindrà tota la informació de post-instal·lació que desitgem. En concret, primer es copia el fitxer firstboot.tgz dins el /tmp del node de login, i després el descomprimim redirigint la sortida sobre el directori arrel de root. La comanda `in-target command` és l'equivalent a fer:

```
chroot /target
command
exit
```

En fer la descompressió s'haurà creat un fitxer `/root/firstboot/firstboot`, que movem al directori /etc/init.d i li donem permisos d'execució. Finalment, amb `update-rc.d` el donem d'alta com a servei i d'aquesta forma serà executat quan iniciem la màquina per primer cop. Aquest servei l'hem configurat de tal forma que s'executa només en la primera arrencada de la màquina, realitza les accions de configuració del node, i finalment es deshabilita automàticament i no es torna a executar mai més en reiniciar el node. Explicarem el funcionament d'aquest servei en la secció 5.2.

Amb la versió actual de simple-cdd no és senzill afegir repositoris addicionals que no segueixin l'estructura dels repositoris de debian, com és el cas dels de cloudera. Per poder instal·lar les seves aplicacions, el que farem és crear un directori local d'on s'agafaran els paquets per incloure'ls al CD. Malauradament, fent-ho d'aquesta manera simple-cdd no és capaç de calcular les dependències d'aquests paquets, i per tant cal tenir la precaució de comprovar-les prèviament i afegir-les al fitxer *deboop.packages*. En el nostre cas, les dependències a instal·lar son openssl, python, lsb-release i postgresql. Creem doncs el directori local i baixem els paquets del repositori de cloudera.

```
mkdir $HOME/deboop/local-packages
cd $HOME/deboop/local-packages
wget http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm/pool/contrib/e/
    enterprise/cloudera-manager-daemons_5.0.2-1.cm502.p0.297~wheezy-cm5.0.2
    _all.deb
wget http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm/pool/contrib/e/
    enterprise/cloudera-manager-server-db-2_5.0.2-1.cm502.p0.297~wheezy-cm5
    .0.2_all.deb
wget http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm/pool/contrib/e/
    enterprise/cloudera-manager-server_5.0.2-1.cm502.p0.297~wheezy-cm5.0.2_all.
    deb
wget http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm/pool/contrib/o/
    oracle-j2sdk1.7_1.7.0+update45-1_amd64.deb
```

Ja podem generar el nostre CD d'instal·lació, però abans hem de modificar un fitxer de configuració de simple-cdd, que conté un bug que afecta als paquets que es baixen amb el perfil *default* (i aquest és un perfil que es carrega sempre), per fer-lo compatible amb les noves versions de grub. Ho podem fer aplicant el patch que es mostra al Fitxer 4.

```
sudo patch -d /usr/share/simple-cdd/profiles < default.patch
```

Un cop tenim tots els fitxers de configuració de simple-cdd preparats ens situem al directori de treball i executem la comanda per generar la iso del CD.

Fitxer 4: default.patch

```
--- default.downloads   2014-06-09 01:52:08.891632909 +0200
+++ default.downloads.new       2014-06-09 01:35:05.033836320 +0200
@@ -1,5 +1,5 @@
 # keep grub or debian-installer may not work properly.
-grub
+grub-pc

 popularity-contest
 localization-config
```

```
cd $HOME/deboop
build-simple-cdd --local-packages $HOME/deboop/local_packages --profiles
    deboop 2>&1 |tee out
```

A partir d'aquí, simple-cdd fa la seva feina. Comença creant un mirror del repositori *main* de debian dins el directori $HOME/deboop/tmp/mirror. Al fitxer $HOME/deboop/tmp/mirror/conf/package-list podrem trobar el llistat de paquets que s'han afegit a aquest mirror, que incorpora el que nosaltres li hem passat amb deboop.packages, junt amb els del directori *local_packages* i totes les dependències. El càlcul d'aquestes dependències es realitza mitjançant un procediment iteratiu en el qual es va actualitzant el fitxer package-list. En aquest procés es van comprovant les dependències dels paquets que ja apareixen al fitxer, amb l'ajut dels fitxers que es troben a $HOME/deboop/tmp/mirror/lists (que contenen tota la informació sobre un paquet i les seves dependències). Si falta alguna dependència per instal·lar, s'afegeix a package-list i es compara amb la versió d'aquest fitxer en la iteració anterior. Quan el procés convergeix, el fitxer conté tots els paquets que s'inclouran al CD.

El següent pas que realitza simple-cdd és modificar els fitxers que es troben tant al directori $HOME/deboop/tmp/cd-build/wheezy/boot1/isolinux com al $HOME/deboop/tmp/cd-build/wheezy/CD1/boot/grub, per tal de configurar grub convenientment. D'aquesta forma, se li passa al kernel els següents paràmetres a l'hora d'iniciar el nou sistema.

```
preseed/file=/cdrom/simple-cdd/default.preseed simple-cdd/profiles=deboop
```

Finalment, l'eina debian-cd [14] se n'ocupa de la compilació del CD, i l'aplicació xorriso [24] de la creació de la imatge iso. Si tot va bé, es crearà el fitxer deboop-0.1-amd64-CD-1.iso dins el directori $HOME/deboop/images.

A continuació millorarem un detall sense importància, però que afecta a la personalització del CD. Tal com s'ha generat la imatge, en la pantalla inicial d'instal·lació apareix el següent missatge: "Debian GNU/Linux installer boot menu". Si volem modificar aquest missatge per fer referència a Deboop en comptes de Debian, cal modificar el fitxer on es defineix el menú inicial (menu.cfg) i tornar a executar xorriso.

```
sed -i "s:Debian GNU/Linux:Deboop 0.1:g" $HOME/deboop/tmp/cd-build/wheezy/
    boot1/isolinux/menu.cfg
cd $HOME/deboop/tmp/cd-build/wheezy
```

```
xorriso -as mkisofs -r -checksum_algorithm_iso md5,sha1 -V 'Deboop 0.1 amd64 1'
    -o /home/alfr3d/deboop/images/deboop-0.1-amd64-CD-1.iso -J -isohybrid
    -mbr syslinux/usr/lib/syslinux/isohdpfx.bin -J -joliet-long -cache-inodes -b
    isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -
    boot-info-table -eltorito-alt-boot --efi-boot boot/grub/efi.img -
    append_partition 2 0x01 /home/alfr3d/deboop/tmp//cd-build/wheezy/CD1/boot
    /grub/efi.img boot1 CD1
```

5.2 Post-configuració del node de login

Tal i com hem comentat a la secció 5.1, la estratègia global és executar un fitxer de configuració que només ho farà en la primera arrencada de la màquina. Per això, el que fem és instal·lar un servei que s'executarà en arrencar i que en acabar d'executar-se es deshabilitarà automàticament. Ja hem vist que amb la línia *late_command* del fitxer deboop.preseed copiem al sistema de fitxers del node de login l'arxiu firstboot.tgz i posteriorment el descomprimim dins de /root/firstboot. Un cop descomprimit, podrem trobar en aquest directori tot un seguit de scripts que ens serviran per a configurar diferents serveis, i que anirem explicant en les següents seccions.

L'script principal s'anomena *firstboot* (veure Fitxer 5), s'instal·larà al directori /etc/init.d i s'activarà com a servei amb la comanda update-rc.d. Com es veu a la línia 14 del fitxer, un cop arrenqui el node de login s'executarà un altre script, *postinstall.sh*, que és qui fa realment les tasques de post-configuració i per tant és el més important. Amb la línia 16 mostrem en pantalla una finestra on es veu el percentatge d'avanç d'aquestes tasques, i finalment un missatge avisant quan s'ha acabat la post-configuració. Aquestes finestres es creen utilitzant l'eina dialog [23] amb l'opció gauge, que serveix per mostrar barres de progrés. En el nostre cas, com executem l'script *postinstall.sh* amb nohup, es genera un fitxer de sortida *nohup.out*, i calcularem el percentatge de script executat en funció del nombre de línies de sortida generades (línies 9 i 10 del Fitxer 6).

L'arxiu *postinstall.sh* és un contenidor d'altres scripts encarregats de configurar cadascun d'ells un servei diferent (veure Fitxer 7). La línia més rellevant de cara al propi servei de post-configuració és la 37, per la qual un cop s'han executat tots els scripts del contenidor es deshabilita el servei firstboot. D'aquesta forma ens assegurem que la configuració del node només es realitza en la primera arrencada del mateix.

A continuació anirem explicant més en detall els scripts continguts al fitxer *postinstall.sh*, i amb els quals aconseguirem tenir un node configurat automàticament per actuar com a manager de cloudera i com a instal·lador dels nodes de treball.

5.2.1 Configuració de la xarxa

El node de login ha de tenir dues interfícies de xarxa com a mínim, una pública per tenir accés a internet i comunicar-se amb l'exterior, i una privada per a les comunicacions amb els nodes de càlcul. La IP de la primera interfície vindrà donada pel rang on estigui servint el servidor DHCP de la xarxa on instal·lem Deboop. La IP privada la configurem nosaltres amb *addnet.sh* (Fitxer 8). La forma de fer-ho és afegir al fitxer /etc/network/interfaces la informació de la nova adreça, on hem considerat que s'utilitzarà la interfície eth1 i hem assignat

Fitxer 5: firstboot

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          firstboot
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Script de configuracio de deboop
# Description: Script de configuracio de deboop
### END INIT INFO

cd /root/firstboot
/usr/bin/nohup sh -x /root/firstboot/postinstall.sh &
sleep 2
/root/firstboot/dialog.sh

exit 0
```

Fitxer 6: dialog.sh

```
#!/bin/bash
file=/root/firstboot/nohup.out
lines=2135
{
        i=0
        while test $i != 100
        do
            echo $i; sleep 1
            new_lines=`wc -l $file |awk '{print $1}'`
            i=`echo "100*$new_lines/$lines" |bc`
        done
} | dialog --title "Post-instal.lacio" --clear --gauge "\nConfigurant deboop login..." 0 0 0
echo "100" |dialog --title "Post-instal.lacio" --clear --gauge "\nConfigurant deboop login..." 0 0 0
sleep 1
dialog --title "Post-instal.lacio" --clear --msgbox "\nConfiguracio del node de login finalitzada. Arrenca els nodes de calcul per tal que FAI instal.li automaticament el seu SO. Despres, obre en un navegador la url 'localhost:7180' per obrir Cloudera Manager i instal.lar el programari de Hadoop" 0 0
clear
```

Fitxer 7: postinstall.sh

```
#!/bin/bash

#Afegeix la xarxa interna
./addnet.sh

#Actualitza les fonts de paquets
./update_apt.sh

#Actualitza els hosts
./addhosts.sh

#Configura el servidor proxy
./proxyconf.sh

#Configura el cache de repositoris
./aptcacher.sh

#Configura i instal.la fai
./faiconfig.sh

#Configura el servidor dhcp
./dhcpconfig.sh

#Configura NFS exports
./nfsconfig.sh

#Configura tftp
./tftpconf.sh

#Configura ntp
./ntpconf.sh

#Instal.la cloudera manager
./cloudera.sh

#Deshabilita l'script per properes arrencades
update-rc.d firstboot remove
cp /root/firstboot/nohup.out /root/postconfig.log
rm -rf /root/firstboot /etc/init.d/firstboot
```

a la xarxa privada un rang d'adreces 192.168.2.0/24. El gateway que apareix (192.168.1.1) és el de la xarxa on jo realitzo les proves, però és un paràmetre que dependrà d'on estigui instal·lat el node de login (veure secció 8.1).

Fitxer 8: addnet.sh

```
1  #!/bin/bash
2
3  #Configura la interficie per a la xarxa interna
4  cat <<EOF >> /etc/network/interfaces
5
6  allow−hotplug eth1
7  iface eth1 inet static
8    address 192.168.2.100
9    netmask 255.255.255.0
10   network 192.168.2.0
11   broadcast 192.168.2.255
12   gateway 192.168.1.1
13 EOF
14
15 #Aixeca la interficie
16 ifup eth1
```

Un cop afegida la informació de la xarxa, aixequem la interfície amb *ifup*.

5.2.2 Configuració dels repositoris de paquets

Durant la creació del CD amb simple-cdd vam escollir no configurar les fonts dels repositoris de paquets, de forma que ho fem ara a la post-configuració del node. De les línies 4 a 10 del Fitxer 9 s'afegeixen els repositoris habituals de debian (incloent contrib i non-free) i els que proporcionen updates de seguretat. A continuació afegim els repositoris de cloudera, baixem la clau GPG per validar els paquets d'aquests repositoris, i l'afegim a l'anella de claus confiables (línies 12 a 19). Finalment, amb les línies 22 i 23 es realitza una actualització de tots els paquets instal·lats fins el moment.

5.2.3 Configuració de la resolució de noms

Per a fer la transcripció entre adreces IP i noms de host, Linux pot fer servir el fitxer /etc/hosts. Amb el Fitxer 10 configurem aquest arxiu, on farem un mapa entre IP's assignades a cada node de càlcul i el seu nom. En el nostre cas, apareix l'adreça 192.168.2.100 per al node de login (ja que és la IP que li hem assignat a la secció 5.2.1). Consecutivament, atorguem a cada node de treball una adreça IP, dins el rang de la xarxa 192.168.2.0/24, i els hi donem noms de l'estil workerXX, amb XX incrementals (fins al worker04 per a realitzar els tests). Aquests noms de host també són un paràmetre que podria dependre de l'administrador que faci la instal·lació (veure secció 8.1).

Fitxer 9: update_apt.sh

```
#!/bin/bash

#Configura les fonts del repositori de paquets
cat <<EOF > /etc/apt/sources.list
deb http://ftp.es.debian.org/debian/ wheezy main contrib non-free
deb-src http://ftp.es.debian.org/debian/ wheezy main contrib non-free

deb http://security.debian.org/ wheezy/updates main contrib non-free
deb-src http://security.debian.org/ wheezy/updates main contrib non-free
EOF

#Configura els repositoris de cloudera
cat <<EOF > /etc/apt/sources.list.d/cloudera-cm5.list
deb http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm wheezy-cm5 contrib
deb-src http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm wheezy-cm5
    contrib
EOF

#Obte la clau publica del repositori de cloudera
wget -qO - http://archive.cloudera.com/cm5/debian/wheezy/amd64/cm/archive.key |
    apt-key --keyring /etc/apt/trusted.gpg.d/cloudera-cm5.gpg add -

#Actualitzem els paquets instal.lats
apt-get update
apt-get -y dist-upgrade
```

Fitxer 10: addhosts.sh

```
#!/bin/bash

#Configura el /etc/hosts
cat <<EOF > /etc/hosts
127.0.0.1       localhost
192.168.2.100   login
192.168.2.1     worker01
192.168.2.2     worker02
192.168.2.3     worker03
192.168.2.4     worker04

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
EOF
```

5.2.4 Configuració del proxy

Amb el Fitxer 11 configurem el proxy que permetrà als nodes de càlcul accedir a la xarxa externa per tal d'actualitzar paquets. L'aplicació de proxy escollida és squid, i s'instal·larà automàticament al node de login, ja que és un dels paquets afegits a la distribució amb simple-cdd. Squid proporciona un fitxer de configuració estàndard, que nosaltres substituirem mitjançant les linies 4 a 48 del nostre script. No entrarem en detall en tota la configuració del fitxer squid.conf, però direm que treballa amb estructures de tipus ACL (llistes de control d'accés) i comentarem les parts més rellevants per al nostre treball.

línia 5 Li donem la IP i port pels quals escoltarà el servei de proxy dins el node de login. En el nostre cas serà per la xarxa interna (192.168.2.100) i pel port 8080.

línia 16 Li donem el path cap a un fitxer on definirem les xarxes que tindran permís per connectar al servidor de proxy.

línia 17 Li donem el path cap a un fitxer on definirem els dominis de destinació on els clients del proxy tindran permís per accedir-hi. D'aquesta forma, bloquegem la sortida des de la xarxa interna cap a dominis no autoritzats.

línia 34 Autoritza les acl's de les línies 16 i 17.

línies 51-55 Creació del fitxer *allow-sites.squid* que es farà servir a la línia 17. Els dominis autoritzats permetran instal·lar als nodes de càlcul una distribució de debian, amb FAI i Cloudera Manager.

línia 58 Creació del fitxer *permesos*, que es farà servir a la línia 16, donant accés al servidor proxy només als nodes que pertanyin a la xarxa 192.168.2.0/24.

línia 61 Reiniciem el servidor de proxy per que llegeixi la nova configuració.

5.2.5 Configuració de la cau de repositoris

De cara a la instal·lació i desplegament del sistema operatiu dels nodes de càlcul, ens serà de gran utilitat poder disposar dels paquets de debian en local (al node de login), per tal que el procés sigui el més ràpid possible. Tenim dos opcions per a aconseguir-ho, mitjançant un mirror local dels repositoris o mitjançant una cau. La primera d'elles consisteix en clonar l'estructura dels repositoris de debian en local, amb el qual cal baixar-se tots els paquets i fer-los disponibles des del node de login. La opció de la cau sembla més interessant, ja que només s'incorporen els paquets que es demanin a l'hora de la instal·lació. Potser la primera baixada de paquets és més lenta (comparada amb l'alternativa de fer-la des d'un mirror local), però com tots els nodes són iguals i baixaran els mateixos paquets, les següents baixades es faran des de la cau.

L'aplicació escollida és l'apt-cacher-ng, i igual que squid, ja s'instal·la automàticament al node de login, doncs està inclòs en els paquets a instal·lar amb el CD de la distribució. Amb les línies 4 a 21 del Fitxer 12 substituïm l'script de configuració original del servei. Respecte a les dades inicials, nosaltres només hem modificat les línies 7 a 9. Amb la línia 7 li diem que escolti pel port 9999 i amb la línia 8 que ho faci des de la IP interna (login correspon a la IP 192.168.2.100). La línia 9 fa que les connexions a l'exterior es facin via proxy

Fitxer 11: proxyconf.sh

```
#!/bin/bash

#Fitxer de configuracio de squid
cat <<EOF > /etc/squid/squid.conf
http_port 192.168.2.100:8080
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
access_log /var/log/squid/access.log squid
hosts_file /etc/hosts
refresh_pattern ^ftp:          1440    20%     10080
refresh_pattern ^gopher:       1440    0%      1440
refresh_pattern .              0       20%     4320
acl permesos src "/etc/squid/permesos"
acl allowsites dstdomain "/etc/squid/allow-sites.squid"
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443          # https
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_Ports port 389         # ldap
acl Safe_Ports port 636         # ldaps
acl Safe_Ports port 1047
acl Safe_Ports port 27000       #matlab
acl Safe_Ports port 9300        #elasticsearch
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow permesos allowsites
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow CONNECT SSL_ports
http_access allow localhost
http_access deny all
http_reply_access allow all
icp_access allow all
cache_effective_group proxy
coredump_dir /var/spool/squid
EOF
```

Fitxer 11 (Cont.): proxyconf.sh

```
49
50 #Dominis de destinacio permesos pel proxy
51 cat <<EOF > /etc/squid/allow-sites.squid
52 .debian.org
53 .fai-project.org
54 .cloudera.com
55 EOF
56
57 #Xarxa des de la qual es pot accdir al proxy
58 echo '192.168.2.0/24' > /etc/squid/permesos
59
60 #Reiniciem el servei amb la nova configuracio
61 /etc/init.d/squid restart
```

(ja que li diem que vagi pel port 8080, que és el que utilitza squid). En el nostre cas, com que el node de login té accés directe a l'exterior, no és obligatori passar pel proxy, però si volguéssim fer la solució més segura i tenir més controlades les sortides, aquesta és la forma de fer-ho.

A continuació hem de configurar l'apt del node de login per dir-li que busqui les fonts dels repositoris dins del servidor cau. Això ho fem amb la línia 24, afegint la informació del cau dins el fitxer /etc/apt/apt.conf.d/30proxy. Per tal que els nodes de càlcul, que instal·larem amb FAI, continguin també aquest fitxer amb la informació del cau, el copiem sota la estructura de directoris /etc/fai (línia 27). Ja veurem posteriorment com es replica aquesta informació als nodes. Finalment, reiniciem el servei de l'apt-cacher-ng.

5.2.6 Instal·lació de FAI

5.2.6.1 Descripció de FAI

Com ja hem comentat, FAI és una eina per al desplegament massiu i desatès de clústers de Linux, i també per a la seva posterior actualització. Fem una breu descripció del procés que es segueix per a la instal·lació d'un node.

Primer de tot s'arrenca el node via targeta de xarxa, i des del faiserver se li proporciona una IP i un kernel de Linux. Aquest kernel s'arrenca muntant el seu sistema de fitxers arrel via NFS des del propi faiserver. Un cop s'ha carregat el kernel, FAI executa tots els seus scripts per configurar el node: es particionen els discs, es crea el sistema de fitxers i s'instal·len els paquets de software. Després d'això, el sistema es configura per adaptar-se a les nostres necessitats i finalment es re-arrenca el node, que ho farà ja des del seu disc local. Els fitxers que permeten la configuració dels nodes (clients de fai) es troben a l'anomenat espai de configuració dins el servidor de fai, que en el nostre cas serà el node de login. Aquests fitxers s'agrupen segons classes, un concepte que servirà per agrupar nodes amb característiques o funcionalitats similars. És a dir, a mode d'exemple, si dins del clúster volem tenir alguns nodes que tinguin un servidor web, podem crear la classe WEB, que tindrà arxius de configuració tals que permetin desplegar nodes amb aquest servei. Un node pot pertànyer a diverses classes, com per exemple a les classes WEB i DEBIAN a la vegada.

Fitxer 12: aptcacher.sh

```
#!/bin/bash

#Fixter de configuracio del cache de repositoris
cat <<EOF > /etc/apt-cacher-ng/acng.conf
CacheDir: /var/cache/apt-cacher-ng
LogDir: /var/log/apt-cacher-ng
Port:9999
BindAddress: login
Proxy: http://login:8080
Remap-debrep: file:deb_mirror*.gz /debian ; file:backends_debian # Debian Archives
Remap-uburep: file:ubuntu_mirrors /ubuntu ; file:backends_ubuntu # Ubuntu Archives
Remap-debvol: file:debvol_mirror*.gz /debian-volatile ; file:backends_debvol # Debian
    Volatile Archives
Remap-cygwin: file:cygwin_mirrors /cygwin # ; file:backends_cygwin # incomplete,
    please create this file or specify preferred mirrors here
Remap-sfnet:  file:sfnet_mirrors # ; file:backends_sfnet # incomplete, please create this
    file or specify preferred mirrors here
Remap-alxrep: file:archlx_mirrors /archlinux # ; file:backend_archlx # Arch Linux
Remap-fedora:  file:fedora_mirrors # Fedora Linux
Remap-epel:   file:epel_mirrors # Fedora EPEL
Remap-slrep:  file:sl_mirrors # Scientific Linux
ReportPage: acng-report.html
ExTreshold: 4
EOF

#Configurem l'apt perque vagi a buscar al cache
echo 'Acquire::http::Proxy "http://login:9999";' > /etc/apt/apt.conf.d/30proxy

#Preparem la futura configuracio de fai per fer servir el cache
rsync -azviR /etc/./apt/apt.conf.d/30proxy /etc/fai

#Reiniciem el servei amb la nova configuracio
/etc/init.d/apt-cacher-ng restart
```

FAI també es pot fer servir com a sistema de rescat en xarxa, si en comptes de fer una instal·lació li diem que corri un sistema operatiu totalment funcional, però sense fer servir el disc local. Llavors permet restaurar particions o comprovar l'estat del sistema, entre d'altres utilitats.

5.2.6.2 Configuració de FAI Tot i que amb el CD de la distribució hem inclòs el paquet fai-quickstart, FAI no és operatiu encara. Hem de reconfigurar alguns paràmetres, instal·lar realment FAI dins el node de login i definir l'espai de configuració per als nostres nodes de treball. Això es realitza amb el Fitxer 13, que conté els següents apartats.

Configuració bàsica Amb la línia 4 habilitem la recol·lecció de logs de l'aplicació, dins el fitxer principal /etc/fai/fai.conf, mitjançant l'activació d'un usuari dedicat a aquesta tasca. Aquest usuari s'anomena *fai* per defecte i es crearà posteriorment dins d'aquest mateix script. Les següents dues línies afecten a la creació del sistema de fitxers arrel que serà exportat via NFS des del faiserver cap als nodes de càlcul en temps de desplegament dels mateixos. La línia 5 fa que els paquets de debian que s'utilitzen per crear aquest sistema de fitxers es baixin des de l'apt-cacher-ng, i la línia 6 fa que s'inclogui dins del /etc/hosts la IP del node de login. La línia 7 modifica el fitxer de fonts de l'apt per actualitzar l'adreça dels repositoris de debian, ja que la que ve per defecte amb FAI està obsoleta.

Creació de l'usuari fai A continuació creem un grup *fai* i un usuari *fai* dins d'aquest grup, assignant el directori /var/log/fai com a directori *home* d'aquest usuari (línies 10 a 12). Com hem dit abans, és l'encarregat de recollir tots els logs generats per l'aplicació i sempre els tindrem disponibles dins de la seva carpeta de *home* al node de login.

Instal·lació Amb la línia 15 es fa la instal·lació pròpiament dita de FAI. Veurem en la secció 6.2.1 la sortida dels logs de la instal·lació, però bàsicament el que es fa és habilitar l'usuari *fai* per tal de poder fer ssh sense password entre nodes, crear el sistema de fitxers *nfsroot* i fer-lo accessible per NFS cap a la xarxa interna.

Creació de l'espai de configuració Amb la instal·lació de FAI s'ha generat automàticament un espai de configuració d'exemple, per defecte dins el directori /srv/fai/config. Degut a la importància d'aquesta estructura d'arxius, la comentarem en més detall en un altre apartat (veure secció 6.2.2). Per ara, el que fem amb la línia 18 és esborrar totalment el directori, ja que el voldrem substituir pel que hem dissenyat nosaltres d'acord a les nostres necessitats. Aquest l'havíem inclòs al CD d'instal·lació i el podrem trobar al node de login sota /root/firstboot. Descomprimim el nostre espai de configuració directament al directori per defecte /srv/fai/config (línia 19). El subdirectori *basefiles* de l'espai de configuració conté un sistema base de la distribució que volem instal·lar als nodes de càlcul. Quan FAI crea dins dels nodes el sistema de fitxers, aquest està buit i l'estratègia que es segueix es desempaquetar aquest sistema base, que conté el mínim per disposar d'un sistema operatiu funcional. Per defecte, amb la instal·lació de FAI al node de login, el sistema base no existeix, ja que dependrà

Fitxer 13: faiconfig.sh

```
#!/bin/bash

#Configuracio de fai
sed -i 's/#LOGUSER/LOGUSER/' /etc/fai/fai.conf
sed -i 's/http.debian.net/login:9999/' /etc/fai/nfsroot.conf
echo 'NFSROOT_ETC_HOSTS="192.168.2.100 login"' >> /etc/fai/nfsroot.conf
sed -i 's/http.debian.net/ftp.es.debian.org/' /etc/fai/apt/sources.list

#Creacio d'usuari i grup fai
groupadd fai
useradd -g fai -d /var/log/fai fai
chown -R fai:fai /var/log/fai

#Instal.lacio de fai
fai-setup -fv

#Espai de configuracio
rm -rf /srv/fai/config
tar -C /srv/fai -xzvf /root/firstboot/config_fai.tgz
cd /srv/fai/config/basefiles
./mk-basefile -J WHEEZY64
mv WHEEZY64.tar.xz DEBOOP.tar.xz

#Script d'inicialitzacio del fai-monitor
cat <<'EOF' > /etc/init.d/fai-monitor
#! /bin/sh
### BEGIN INIT INFO
# Provides:          fai-monitor
# Required-Start:
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: fai-monitor daemon
# Description:       This script powers up fai-monitor daemon
### END INIT INFO

DAEMON=/usr/sbin/fai-monitor

test -x $DAEMON || exit 0

out=`mktemp`

_askpid() {
    netstat -lpn 2> /dev/null |grep -v tcp6 |grep 4711 | awk '{split ($7,a,"/");print a
      [1]}' > $out
}

_start() {
    _askpid
```

Fitxer 13 (Cont.): faiconfig.sh

```
    if [ -s $out ]; then
      echo "fai-monitor was already started"
      rm $out
      exit 0
    else
      echo Starting fai-monitor...
      fai-monitor -b -T -d
    fi
}

_stop() {
    _askpid
    if [ -s $out ]; then
      echo "Stopping fai-monitor..."
      kill `cat $out`
    else
      echo fai-monitor was not running
    fi
}

_status() {
    _askpid
    if [ -s $out ]; then
      echo fai-monitor is running
    else
      echo fai-monitor is not running
    fi
}

case "$1" in
   start)
      _start
      ;;
   stop)
      _stop
      ;;
   status)
      _status
      ;;
   *)
      echo "Usage: $0 {start|stop|status}" >&2
      exit 3
      ;;
esac
rm $out
EOF

#Inicia el dimoni de fai-monitor
chmod 755 /etc/init.d/fai-monitor
/etc/init.d/fai-monitor start
```

de quina distribució vulguem desplegar, i per tant cal crear-lo posteriorment. Això és el que fem amb les línies 20 a 22.

Script d'inicialització de fai-monitor fai-monitor és un dimoni que corre al faiserver, i que va mostrant informació de les diferents etapes de la instal·lació dels nodes, a mida que es realitza el desplegament. Bàsicament, va actualitzant un fitxer de log (/var/log/fai-monitor.log) cada cop que FAI comença o acaba una de les seves tasques. Amb la versió actual, aquest dimoni no està activat com a servei per a que s'inicii quan arrenca el node, pel qual hem creat un script dins de /etc/init.d que ho solucioni. Conté les típiques opcions per a un dimoni (start, stop i status), i la feina principal la realitza la funció _askpid() que podem veure a les línies 43 a 45. Aquesta funció comprova si hi ha algun servei escoltant pel port 4711 (port per defecte de fai-monitor), i si és així obté el pid del procés. Amb aquesta informació ja es pot interaccionar amb el servei, aturar-lo o iniciar-lo, cosa que fem amb la última línia.

5.2.7 Configuració del servidor DHCP

A l'hora d'instal·lar els nodes de càlcul amb FAI, aquests necessiten tenir una IP per poder arrencar per xarxa. Per aquest motiu disposarem d'un servidor de DHCP en el faiserver, que serveixi adreces IP als nodes de la xarxa interna. El programari que s'encarrega d'això és l'isc-dhcp-server [25], que s'instal·la automàticament al node de login des del CD de deboop. Per a adaptar el servei al nostre clúster, modificarem un parell dels seus scripts de configuració. Al Fitxer 14 (línia 4) es pot veure que dins el /etc/default/isc-dhcp-server li diem que escolti per la interfície eth1, que és la que està connectada a la xarxa interna. A més, al fitxer /etc/dhcp/dhcpd.conf afegim totes les dades corresponents al node de login (ja que és on estarà corrent el servei) i de la xarxa interna (xarxa on es serviran les IP's). Amb la línia 19 li passem el *path* cap al carregador d'arrencada que cada node es baixarà via tftp (servei que explicarem a la secció 5.2.9). En el nostre cas, el carregador *pxelinux.0* es troba sota el directori de treball del servidor de tftp, dins un directori anomenat fai.

A continuació, a cadascun dels nodes de treball, identificats per la seva mac-address, se li assigna una adreça IP i el seu hostname (línies 22 a 25), informació que rebran en temps d'arrencada. Finalment, es reinicia el servei per tal que llegeixi la nova configuració.

5.2.8 Configuració del servidor de fitxers

Els nodes de càlcul necessiten accedir a un parell de directoris del node de login mentre es realitza la instal·lació dels mateixos. Per a tal fi, s'ha habilitat un servidor de NFS, i el fitxer que cal modificar per afegir els directoris a compartir és el /etc/exports (línies 4 a 7 del Fitxer 15). El primer directori és l'espai de configuració de FAI, i el segon és el sistema de fitxers arrel que utilitza el kernel arrencat als nodes de càlcul mentre s'instal·la el sistema operatiu. Com sempre amb aquest tipus de serveis, amb la línia 10 es re-arrenca perquè es carreguin les noves opcions.

Fitxer 14: dhcpconfig.sh

```
#!/bin/bash

#Configura el servidor de dhcp
sed -i 's/INTERFACES=""/INTERFACES="eth1"/' /etc/default/isc-dhcp-server

cat <<EOF > /etc/dhcp/dhcpd.conf
deny unknown-clients;
option dhcp-max-message-size 2048;
use-host-decl-names on;

subnet 192.168.2.0 netmask 255.255.255.0 {
option routers 192.168.2.100;
option domain-name "deboop.net";
option domain-name-servers 192.168.1.1;
option time-servers login;
option ntp-servers login;
server-name login;
next-server login;
filename "fai/pxelinux.0";
}

host worker01 {hardware ethernet 08:00:27:FD:38:01;fixed-address 192.168.2.1;}
host worker02 {hardware ethernet 08:00:27:FD:38:02;fixed-address 192.168.2.2;}
host worker03 {hardware ethernet 08:00:27:FD:38:03;fixed-address 192.168.2.3;}
host worker04 {hardware ethernet 08:00:27:FD:38:04;fixed-address 192.168.2.4;}
EOF

#Reinicia el servei
/etc/init.d/isc-dhcp-server restart
```

Fitxer 15: nfsconfig.sh

```
#!/bin/bash

#Directoris a exportar
cat <<EOF > /etc/exports
/srv/fai/config 192.168.2.100/24(async,ro,no_subtree_check)
/srv/fai/nfsroot 192.168.2.100/24(async,ro,no_subtree_check,no_root_squash)
EOF

#Reinicia el servei
/etc/init.d/nfs-kernel-server restart
```

5.2.9 Configuració del servidor de TFTP

Tftp és un protocol de transferència de fitxers molt senzill, àmpliament utilitzat per a transferir automàticament fitxers de configuració entre nodes d'una xarxa local. És molt adient per a transferir el carregador d'arrencada quan arrenquem en xarxa amb PXE [26]. És un servei que es pot afegir al super-servidor inetd, la qual cosa fem modificant la configuració de /etc/inetd.conf (línia 4 al Fitxer 16). Com es veu en la línia, el binari d'execució és in.tftpd, que correspon a la versió HPA del servidor [27] (la que s'ha instal·lat automàticament al node de login amb el paquet tftpd-hpa mitjançant el CD de deboop), i el directori de treball és /srv/tftp. Com hem vist a la secció 5.2.7, sota aquest directori (de fet, en un subdirectori *fai*) es troba el carregador d'arrencada *pxelinux.0*, entre d'altres fitxers necessaris per arrencar amb el protocol PXE (el fitxer de kernel, *vmlinuz*, i el ramdisk inicial per al kernel, *initrd.img*). Un cop afegit el servei de tftp, reiniciem el servidor inetd (línia 7).

Fitxer 16: tftpconf.sh

```
1  #!/bin/bash
2
3  #Afegir el servei a inet
4  sed -i '/boot /atftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /srv/
       tftp\' /etc/inetd.conf
5
6  #Reinicia el servei
7  /etc/init.d/openbsd-inetd restart
8
9  #Crea el template per als workers
10 cat <<EOF > /srv/tftp/fai/pxelinux.cfg/pxe.tmpl
11 default fai-generated
12
13 label fai-generated
14 kernel vmlinuz-3.2.0-4-amd64
15 append initrd=initrd.img-3.2.0-4-amd64 ip=eth0:dhcp root=/dev/nfs nfsroot
       =192.168.2.100:/srv/fai/nfsroot:vers=3 aufs monserver=login LOGSERVER=login
       FAI_FLAGS=verbose,sshd,createvt,reboot FAI_CONFIG_SRC=nfs://192.168.2.100/
       srv/fai/config FAI_ACTION=install
16 EOF
17
18 #Genera el fitxer de configuracio per a cada node
19 for i in `cat /etc/hosts |grep worker |awk '{print $2}'`; do fai-chboot -vc pxe.tmpl $i;
       done
```

A més dels fitxers que hem comentat, el carregador d'arrencada necessita un altre fitxer que contingui la informació necessària per continuar el seu procés. Nosaltres crearem un per cada node, però com la configuració es molt semblant entre ells, farem servir una plantilla, *pxe.tmpl*, que situarem a dins del subdirectori /srv/tftp/fai/pxelinux.cgf (línies 9 a 16). En aquest fitxer simplement li diem quin és el kernel a carregar (línia 14) i els paràmetres que li passem al kernel (línia 15), que explicarem amb una mica més de detall.

initrd Li proporcionem el path cap al fitxer amb el ramdisk inicial, que conté

el sistema de fitxers amb el que inicia el kernel abans de poder muntar el que serà el definitiu.

ip La IP s'obtindrà per la interfície eth i per dhcp.

root El sistema de fitxers arrel durant la instal·lació s'obté via nfs.

nfsroot Path cap al sistema de fitxers arrel, exportat via nfs des del node de login en el directori /srv/fai/nfsroot, fent servir la versió 3 del protocol NFS.

aufs Quan el kernel arrenca, carrega el ramdisk i munta el sistema de fitxers arrel via nfs, ho fa en mode de només lectura. Per poder escriure sobre el sistema de fitxers i fer les modificacions necessàries per tal de configurar el node, l'estratègia utilitzada és muntar a sobre un disc RAM via aufs (another unionfs). Aquest és un servei que permet superposar fitxers i directoris de diferents sistemes de fitxers, formant un únic sistema de fitxers coherent.

monserver Host on està corrent fai-monitor, el node de login.

LOGSERVER Host on es guardaran els logs de la instal·lació de cada node de càlcul, també ho centralitzem al node de login.

FAI_FLAGS Amb els flags afegits, la sortida de logs durant la instal·lació serà verbose, engegarà un dimoni de ssh per habilitar connexions remotes al node de càlcul inclús abans que finalitzi la instal·lació (útil quan sorgeixin problemes), es crearan terminals virtuals durant la instal·lació (també útil en cas de problemes) i el node es rebotarà un cop acabi la instal·lació, inclús si hi ha hagut alguna errada que apareixi als logs (altrament, caldria pitjar return).

FAI_CONFIG_SRC Variable que descriu com ha d'accedir el node de càlcul a l'espai de configuració (li diem que es via nfs i li passem la ruta).

FAI_ACTION Escollim la opció install, però es podrien fer altres tasques en el node, com ara una actualització.

Finalment, amb la línia 19 generem un fitxer de configuració per a cada node, a partir del llistat de nodes que tenim al /etc/host, i la plantilla *pxe.tmpl* que acabem de crear. L'eina fai-chboot utilitzada disposa de moltes opcions i és molt còmoda per a gestionar els fitxers de configuració per a l'arrencada en xarxa, com veurem a la secció 6.2.3.

5.2.10 Configuració de la sincronització dels rellotges

Per estalviar problemes de desajustos posteriors en la configuració, és una bona idea que tots els nodes d'un clúster tinguin l'hora sincronitzada. De fet, si no és així, Cloudera Manager es queixa quan fa les seves comprovacions. Tal i com l'hem instal·lat, el node de login ja es sincronitza amb els servidors de ntp per defecte que proporciona debian. El que fem amb el Fitxer 17 és configurar-lo per tal que actuï com a servidor i accepti peticions ntp des de la xarxa interna. A l'espai de configuració de FAI definirem que els nodes de càlcul es sincronitzin contra el node de login.

Fitxer 17: ntpconf.sh

```
#!/bin/bash

#Dono acces al cluster al servidor ntp
sed -i '/restrict ::1/arestrict 192.168.2.0 mask 255.255.255.0 nomodify notrap' /etc/ntp
    .conf

#Reinicio el servei
/etc/init.d/ntp restart
```

5.2.11 Configuració de Cloudera Manager

El node de login ja conté els paquets de Cloudera Manager, doncs els havíem inclòs al CD d'instal·lació, i quan arrenca el node ja s'executen automàticament els serveis corresponents. No obstant això, per al bon funcionament de cloudera cal modificar un paràmetre del kernel relatiu a la memòria d'intercanvi (*swap*). Això ho fem amb el Fitxer 18, on li donem un valor de zero al paràmetre *vm.swappinnes*. Aquesta configuració intenta evitar que es faci *swapping*, i de fet el kernel només ho farà per evitar situacions de "out of memory", amb el que s'aconsegueix una disminució en la latència de resposta global del sistema.

Fitxer 18: cloudera.sh

```
#!/bin/bash

#Parametre vm.swappiness a zero
sysctl vm.swappiness=0
echo 'vm.swappiness = 0' >> /etc/sysctl.conf
```

5.2.12 Configuració final

Amb tots els scripts cridats des de *postinstall.sh* (Fitxer 7) ja tenim configurat el node de login perquè executi el servei de Cloudera Manager i el d'instal·lador de nodes de càlcul amb FAI. Amb les últimes línies del fitxer *postinstall.sh* deshabilitem que en properes arrencades s'executi el servei firstboot, i per tant no es torni a configurar el node. Això es fa amb la comanda update-rc.d, que serveix per instal·lar o esborrar scripts d'inicialització del sistema. També copiem el fitxer de sortida de nohup al directori /root, reanomenant-lo com a *postconfig.log*, i esborrem tot el directori /root/firstboot, ja que no ens caldran més els arxius de configuració. Finalment, també esborrem el fitxer /etc/init.d/firstboot, doncs tampoc es farà servir més.

5.3 Creació de l'entorn virtual

Els que hem treballat amb VirtualBox estem habituats a utilitzar la seva interfície gràfica per tal de crear i gestionar màquines virtuals. No obstant això,

per tal de realitzar els tests d'aquesta pràctica, calia crear i destruir màquines convidades amb molta freqüència, i era desitjable aconseguir una manera fàcil d'automatitzar aquesta tasca. En aquest sentit, VirtualBox ofereix una eina en línia de comandes molt potent, VBoxManage [28]. Depenent de les opcions que se li passin, permet crear, modificar i esborrar màquines virtuals d'una manera molt ràpida. A la vegada, la seva capacitat de realitzar aquestes tasques dins d'un script que les automatitzi, la fa idònia per a entorns de desenvolupament.

En el cas de no tenir instal·lat encara virtualbox, per aconseguir l'última versió, cal modificar el fitxer /etc/apt/sources.list, afegint la següent línia:

```
deb http://download.virtualbox.org/virtualbox/debian wheezy contrib
```

Després, cal baixar la clau GPG que permet validar els paquets baixats, i afegir-la a l'anella de claus confiables:

```
wget http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc
sudo apt-key add oracle_vbox.asc
```

I finalment instal·lem el paquet de virtualbox

```
sudo apt-get update
sudo apt-get install virtualbox-4.3
```

Per a la simulació d'un clúster amb un node de login i diversos nodes de càlcul hem fet servir l'script *create_VM.sh* (veure Fitxer 19). L'script rep com a paràmetre d'entrada el nom de la màquina virtual que crearem amb la línia 3 del codi, i que a la vegada serà el hostname per al seu sistema operatiu. Per diferenciar el node de login de la resta, a aquest li obliguem a tenir com a hostname "login". Amb les línies 4 i 5 creem controladores virtuals per a l'emmagatzemament, una IDE per a simular el lector de CD i una SATA per al disc dur. La línia 7 s'encarrega de crear un disc virtual de 100GB i el guarda en format VDI (format per defecte de VirtualBox), i la línia 9 l'afegeix a la controladora SATA.

En el cas que la màquina que estem creant sigui la de login, li assignem una memòria de 2GB i li definim dues interfícies de xarxa; una per a la xarxa pública per a la comunicació amb l'exterior i una per a la xarxa privada que li permetrà comunicar-se amb els nodes de càlcul (línia 11). Aquesta configuració depèn de la màquina amfitrió i del nom de les seves interfícies de xarxa. En el cas exposat, està pensada per a una xarxa amb un servidor DHCP del qual poder obtenir una adreça IP, i una màquina amfitrió amb interfície de xarxa anomenada *xdmz0*. En el cas d'una configuració més habitual, on la interfície s'anomeni *eth0*, caldria modificar el paràmetre –bridgeadapter1, mentre la resta de paràmetres es pot mantenir. La línia 12 "insereix" el CD d'instal·lació de la distribució que hem creat (en aquest cas, una imatge iso desada en disc), dins el lector virtual. Finalment, la línia 13 arrenca la màquina virtual i defineix el port pel qual ens podem comunicar amb *rdesktop* per "connectar" una consola i veure l'arrencada de la màquina.

Si el que estem creant és un node de treball, amb la línia 15 li assignem una memòria de 512MB i tan sols li definim una interfície de xarxa (`--nic1 intnet`), que serà privada per a la comunicació interna del clúster. A més, li definim com a dispositiu preferit d'arrencada que ho faci per xarxa (`--boot1 net`), ja que

Fitxer 19: create_VM.sh

```
#/bin/bash
[ $# -ne 1 ] && { echo "Usage: $0 hostname (mandatory: \"login\" for management node)"; exit 1; }
VBoxManage createvm --name $1 --ostype Debian_64 --register
VBoxManage storagectl $1 --name IDE --add ide
VBoxManage storagectl $1 --name SATA --add sata --portcount 1
cd $HOME/VirtualBox\ VMs/$1
VBoxManage createhd --filename $1 --size 102400 --format VDI
cd -
VBoxManage storageattach $1 --storagectl SATA --port 0 --type hdd --medium $HOME/VirtualBox\ VMs/$1/$1.vdi
if [ "$1" == 'login' ]; then
  VBoxManage modifyvm $1 --memory 2048 --vram 12 --rtcuseutc on --pae off --nic1 bridged --bridgeadapter1 xdmz0 --nic2 intnet --audio pulse --usb on --usbehci on
  VBoxManage storageattach $1 --storagectl IDE --port 1 --device 0 --type dvddrive --medium $HOME/deboop/images/deboop-0.1-amd64-CD-1.iso
  VBoxHeadless -s $1 -e TCP/Ports=3390 &
else
  VBoxManage modifyvm $1 --memory 512 --vram 12 --rtcuseutc on --pae off --nic1 intnet --audio pulse --usb on --usbehci on --boot1 net --macaddress1 080027FD38`echo ${1: -2}`
  VBoxManage storageattach $1 --storagectl IDE --port 1 --device 0 --type dvddrive --medium emptydrive
  VBoxHeadless -s $1 -e TCP/Ports=33`echo ${1: -2}` &
fi
```

serà el mecanisme pel qual es farà la instal·lació del sistema operatiu als nodes amb l'eina FAI. Al final de la mateixa línia li definim l'adreça hardware que tindrà la seva interfície de xarxa, ja que per poder fer la instal·lació amb FAI haurem de tenir un llistat amb les macaddress de tots els nodes, i si li assignem a priori, ens facilitarà aquesta tasca. En el cas de nodes físics en un clúster real, caldrà entrar a la BIOS de cada node per tal d'obtenir aquesta adreça. La línia 17 arrenca la màquina i defineix un port diferent per a cada node per poder connectar amb *rdesktop*.

6 Instal·lació de deboop

Un cop hem creat la distribució, en aquesta secció mostrarem les passes que cal seguir per a fer una instal·lació de deboop, utilitzant un model de clúster amb quatre nodes de càlcul creat amb màquines virtuals. Acompanyarem tot el procés amb captures de pantalla en diferents etapes del mateix.

6.1 Instal·lació del node de login

Per començar crearem la màquina virtual amb el node de login, fent servir l'script *create_VM.sh*. Recordem que, tal i com s'ha vist al Fitxer 19, es crearà una màquina amb 2GB de memòria principal, dues interfícies de xarxa i amb la iso de deboop que hem generat a la secció 5.1 (i situada dins el directori $HOME/deboop/images) muntada com a cd virtual.

```
$> ./create_VM.sh login
Virtual machine 'login' is created and registered.
UUID: ca16fa5a-3219-46d2-91f7-61ee0a01fdae
Settings file: '/home/user/VirtualBox VMs/login/login.vbox'
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Disk image created. UUID: 676c7ff0-4472-4a33-aa75-f326102f9c1b
/home/user
$> Oracle VM VirtualBox Headless Interface 4.3.12
(C) 2008-2014 Oracle Corporation
All rights reserved.

VRDE server is listening on port 3390.
```

Com veiem a la sortida de la comanda, a part de crear la màquina, el mateix script s'encarrega d'arrencar-la. Per veure la sortida de consola, des del nostre ordinador haurem d'obrir una connexió *rdesktop* apuntant al port pel qual es serveix la visualització remota (3390, com mostra l'última línia de la sortida de *create_VM.sh*.

```
rdesktop localhost:3390 &
```

Com dins el cd virtual es troba la imatge d'instal·lació de deboop, ens mostrarà la pantalla d'inici d'aquest (Figura 1). Li donem a l'opció "Install", que ja ve preseleccionada. El següent menú que se'ns presenta ens demana escollir el llenguatge que farem servir durant la instal·lació. Tot i que hem procurat automatitzar al màxim tot el procés, hem mantingut uns pocs menús que permeten a l'usuari escollir algunes opcions crítiques. És el cas del llenguatge i el

tipus de teclat, ja que si imposem un tipus de teclat i es fa la instal·lació amb un ordinador amb teclat diferent, per a un usuari no expert en linux després no és trivial fer el canvi. Escollim el llenguatge català i ens demanarà la nostra ubicació, on seleccionarem "Espanya". En el següent menú escollirem el mapa de teclat català. A continuació es configura la xarxa i es desplega l'últim menú on se'ns demanarà el fus horari de la nostra ubicació. Escollim l'opció "Madrid" i a partir d'aquí la instal·lació serà totalment automàtica; es particiona el disc, s'instal·la el sistema base (Figura 2), es configura apt i instal·la programari addicional, s'instal·la grub i finalment es reinicia el sistema.

Figura 1: Menú d'instal·lació de deboop

Quan bota el sistema, apareixerà la pantalla de grub, on escollim la primera opció, "Debian GNU/Linux, amb el Linux 3.2.0-4-amd64" (Figura 3). Veurem l'inici del sistema i un cop ha carregat tots els serveis, iniciarà el procés de post-instal·lació, executant l'script *postinstall.sh*. Podem veure el grau d'avanç d'execució de l'script per consola (Figura 4), i un cop la configuració del node de login hagi finalitzat ens mostrarà un missatge de confirmació (Figura 5). Acceptem amb la tecla "Enter" i a la consola ja tindrem la pantalla d'entrada on se'ns demana login i password (root:toor). El node ja és totalment funcional i podem entrar per consola o millor via ssh, que ens permetrà treballar amb terminals de forma més còmoda que no pas per la consola.

6.2 Post-instal·lació

Si fem una ullada al fitxer de log de l'script de post-instal·lació, que trobarem a /root/postconfig.log, podem veure la sortida de totes les comandes de configuració que hem aplicat per obtenir el node de login. En especial, és interessant la generada amb la instal·lació de FAI i el serveis relacionats, que comentarem a continuació.

Figura 2: Instal·lació automàtica de debian

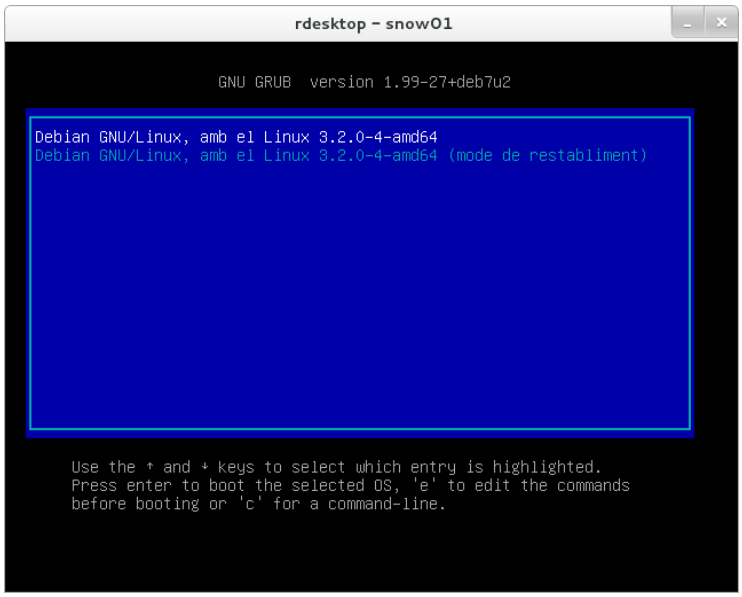

Figura 3: Menú d'arrencada de grub

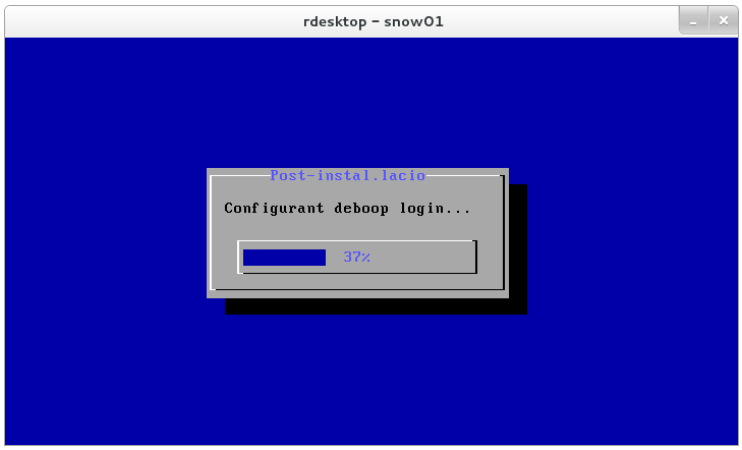

Figura 4: Configuració del node de login

Figura 5: Finalització de la configuració

6.2.1 Instal·lació de FAI

En executar l'ordre "fai-setup -v" obtenim una sortida extensa de totes les tasques que es van realitzant per a instal·lar i configurar FAI, de la qual mostrarem les línies més importants.

Primer s'intenta crear l'usuari fai (tot i que troba que ja existeix perquè l'havíem creat prèviament) i es generen les claus públiques i privades per tal que es pugui fer ssh sense password entre nodes. Això és necessari per copiar els logs des del node client al faiserver i per connectar al node client per ssh en cas de problemes.

```
Account $LOGUSER=fai already exists.
Make sure that all install clients can
log into this account without a password.
Generating public/private rsa key pair.
Your identification has been saved in /var/log/fai/.ssh/id_rsa.
Your public key has been saved in /var/log/fai/.ssh/id_rsa.pub.
The key fingerprint is:
4a:88:55:43:31:5c:e1:f5:88:28:ac:25:02:89:32:44 fai@login
The key's randomart image is:
.....
Adding 192.168.1.156 to known_hosts.
Adding 192.168.2.100,login to known_hosts.
/var/log/fai/.ssh/known_hosts created.
/var/log/fai/.ssh/authorized_keys created.
User account fai set up.
```

A continuació es crea el sistema de fitxers que es farà servir com a arrel durant la instal·lació dels nodes (*nfsroot*), amb el mètode *debootstrap* [29]. Aquest sistema de fitxers es situa a /srv/fai/nfsroot i veiem que el kernel i el ramdisk generats es copien a /srv/tftp/fai per fer-los servir durant l'arrencada dels nodes via PXE. És molt important que en acabar aquesta part de l'scrip al log aparegui el missatge "fai-make-nfsroot finished properly".

```
Creating FAI nfsroot in /srv/fai/nfsroot
Creating base system using debootstrap version 1.0.48+deb7u
Calling debootstrap --exclude=info wheezy /srv/fai/nfsroot http://login:9999/debian
I: Retrieving Release
...
I: Base system installed successfully.
Creating base.tar.xz
ainsl: appending to /srv/fai/nfsroot/etc/hosts: 192.168.2.100   login
'/etc/resolv.conf' -> '/srv/fai/nfsroot/etc/resolv.conf'
Upgrading /srv/fai/nfsroot
...
install_packages: executing chroot /srv/fai/nfsroot apt-get clean
install_packages exit code: 0
'/srv/fai/nfsroot/boot/vmlinuz-3.2.0-4-amd64' -> '/srv/tftp/fai/vmlinuz
    -3.2.0-4-amd64'
'/srv/fai/nfsroot/boot/initrd.img-3.2.0-4-amd64' -> '/srv/tftp/fai/initrd.img
    -3.2.0-4-amd64'
TFTP environment prepared. To use it, enable the dhcpd and start a TFTP daemon on
    root /srv/tftp/fai.
FAI packages inside the nfsroot:
fai-client        4.2
```

```
fai-nfsroot        4.2
fai-setup-storage  4.2
...
Waiting for background jobs to finish
[1]+  Done                    nice xz -q $NFSROOT/var/tmp/base.tar  (wd: /srv/fai/
    nfsroot)
fai-make-nfsroot finished properly.
Log file written to /var/log/fai/fai-make-nfsroot.log
```

Finalment s'exporta el directori creat, per tal que sigui accessible pels client de FAI via NFS. Amb això acaba la instal·lació de FAI, i també és molt important assegurar-nos que apareix el missatge "FAI setup finished".

```
Adding line to /etc/exports: /srv/fai/nfsroot 192.168.1.156/24(async,ro,
    no_subtree_check,no_root_squash)
Re-exporting directories for NFS kernel daemon....
FAI setup finished.
Log file written to /var/log/fai/fai-setup.log
```

6.2.2 Espai de configuració de FAI

Com hem comentat anteriorment, l'espai de configuració de FAI conté tota la informació de com volem que siguin els nodes instal·lats mitjançant aquesta eina. També havíem vist que en executar l'script de post-instal·lació s'esborra l'espai que instal·la l'aplicació per defecte, i es substitueix per un creat *ad-hoc*. L'estructura d'arxius del nostre espai és com la que es mostra a la Figura 6.

basefiles En aquest directori es situa el sistema base que es descomprimirà en el client de FAI. Inicialment no existeix i el creem amb les linies 20 a 22 del Fitxer 13. Un cop creat, aquest directori contindrà el fitxer DEBOOP.tar.xz.

Fitxer 20: 50-host-classes

```
1  #! /bin/bash
2  # assign classes to hosts
3  # use a list of classes for our demo machine
4  case $HOSTNAME in
5      worker*)
6          echo "FAIBASE DEBOOP" ;;
7      *)
8          echo "FAIBASE DEBOOP" ;;
9  esac
10 ifclass -o I386 AMD64 && echo GRUB_PC
11 exit 0
```

class Directori on es defineixen les classes i s'exporten les variables que es faran servir. L'script *50-host-classes* es pot veure a la Figura 20, i s'encarrega d'assignar un node a una classe en funció del seu hostname. En el nostre cas només tenim un tipus de noms de host (worker*), però mantenim l'estructura

```
/srv/fai/config/
├── basefiles
│   └── mk-basefile
├── class
│   ├── 50-host-classes
│   ├── DEBOOP.var
│   └── FAIBASE.var
├── debconf
│   └── DEBOOP
├── disk_config
│   └── FAIBASE
├── files
│   └── etc
│       ├── apt
│       │   ├── apt.conf.d
│       │   │   └── 30proxy
│       │   │       └── DEBOOP
│       │   └── sources.list
│       │       └── DEBOOP
│       ├── default
│       │   └── locale
│       │       └── DEBOOP
│       ├── hosts
│       │   └── DEBOOP
│       ├── motd
│       │   └── DEBOOP
│       ├── ntp.conf
│       │   └── DEBOOP
│       ├── profile.d
│       │   └── proxy.sh
│       │       └── DEBOOP
│       └── resolv.conf
│           └── DEBOOP
├── hooks
│   └── savelog.LAST.source
├── package_config
│   ├── DEBOOP
│   ├── DEBOOP.asc
│   └── FAIBASE
├── scripts
│   ├── DEBOOP
│   │   ├── 10-rootpw
│   │   ├── 12-grubpc
│   │   ├── 30-interface
│   │   ├── 35-ntp
│   │   ├── 36-cloudera
│   │   └── 40-misc
│   ├── FAIBASE
│   │   ├── 10-misc
│   │   └── 20-removable_media
│   └── LAST
│       └── 50-misc
└── tests
    ├── FAIBASE_TEST
    └── Faitest.pm
```

Figura 6: Espai de configuració de FAI

case en previsió de futures ampliacions del clúster. Existeixen unes classes especials a les quals pertanyen tots els nodes, anomenades DEFAULT i LAST.

Els fitxers .var serveixen per definir variables globals que necessitaran els nodes instal·lats. A FAIBASE.var hem posat les més generals,

```
# default values for installation. Override them in your *.var files
# allow installation of packages from unsigned repositories
FAI_ALLOW_UNSIGNED=1
UTC=no
TIMEZONE=Europe/Madrid
# root password; md5 and crypt are possible
ROOTPW='$1$qvt2jqAD$VPbcX1QziY4HuBSDeaABW0'
# errors in tasks greater than this value will stop the installation
STOP_ON_ERROR=700
```

i a DEBOOP.var les més específiques per al nostre clúster.

```
LOGSERVER=login
LOGUSER=fai
DOMAIN="deboop.net"
FAI_BACKUPDIR=$LOGDIR/backup
```

debconf Conté fitxers que obligatòriament han de tenir el nom de les classes que vulguem instal·lar. En el nostre cas, només hi ha DEBOOP, i conté els paràmetres de preseeding que li passarem al client de FAI durant la seva instal·lació.

```
locales locales/default_environment_locale select ca_ES.UTF-8
locales locales/locales_to_be_generated multiselect ca_ES.UTF-8 UTF-8
keyboard-configuration keyboard-configuration/modelcode string pc105
keyboard-configuration keyboard-configuration/xkb-keymap select es
keyboard-configuration keyboard-configuration/variant select ES
keyboard-configuration keyboard-configuration/model select Generic 105-key (Intl)
    PC
keyboard-configuration keyboard-configuration/layoutcode string es
keyboard-configuration keyboard-configuration/optionscode string ctrl:nocaps,
    terminate:ctrl_alt_bksp
```

disk_config Aquí es defineix el particionat dels disc, també amb un fitxer per cada classe que necessitem instal·lar. Nosaltres hem generat un fitxer FAIBASE que configura una partició swap amb 512MB i una partició root que ocupa la resta del disc, que és marcat com a bootable.

```
# <type> <mountpoint> <size>   <fs type> <mount options> <misc options>
disk_config disk1 disklabel:msdos bootable:1 fstabkey:label
primary /    -100%  ext4  rw,noatime,errors=remount-ro createopts="-L /"
logical swap  512   swap  sw  createopts="-L SWAP"
```

files Sota aquest directori es troben els fitxers que seran posteriorment copiats als nodes clients amb les comandes del directori *scripts*, que explicarem després. L'estructura del directori és una mica confusa, en el sentit que cal anomenar tots els fitxers amb el nom de la classe que vulguem desplegar. Així, si volem

copiar el fitxer /etc/hosts, aquest ha d'estar situat dins l'estructura de directoris a /files/etc/hosts/DEBOOP. No descriurem tots els fitxers, però amb els noms que apareixen a la Figura 6 es pot veure que són els encarregats de configurar els repositoris, el /etc/hosts, el client d'ntp i el proxy per accedir a internet, entre d'altres.

hooks Els *hooks* són scripts executats abans de que s'iniciï una de les tasques de FAI, i s'anomenen amb el següent codi: task.class.source. En aquest directori està situat el fitxer *savelog.LAST.source*, que com qualsevol node pertany a la classe LAST, serà executat per tots els clients de FAI abans de començar la tasca *savelog*. Aquest script s'encarrega de buscar missatges d'error de totes les tasques prèvies i abocar-los al fitxer *error.log*.

package_config Conté llistats dels paquets a instal·lar en els clients, en funció de la classe definida. Nosaltres tenim un fitxer FAIBASE amb paquets generals (fai-client, cron, openssh-server) i un DEBOOP amb paquets més relacionats amb el nostre clúster (isc-dhcp-client, grub-pc). Dins d'aquest directori també existeix un fitxer DEBOOP.asc amb claus gpg de repositoris, en aquest cas dels de FAI.

scripts Aquí situem els scripts de configuració posteriors al desempaquetatge del sistema base, també en funció de la classe definida. Per exemple, per a la classe DEBOOP (veure Figura 6) tenim scripts que fan diferents funcions; modificar el password de root, instal·lar grub, configurar les interfícies de xarxa, configurar el client d'ntp, i configurar el paràmetre de *swappinnes* per a cloudera.

6.2.3 Configuració final

Un cop desplegat el nou espai de configuració, s'inicia el servei de fai-monitor, i es configuren i reinicien els serveis de dhcp i nfs. A continuació s'inicia el servidor de tftp i es generen els fitxers de boot per als nodes de càlcul.

```
Starting fai-monitor...
+ ./dhcpconfig.sh
Stopping ISC DHCP server: dhcpd failed!
Starting ISC DHCP server: dhcpd.
+ ./nfsconfig.sh
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon....
Exporting directories for NFS kernel daemon....
Starting NFS kernel daemon: nfsd mountd.
+ ./tftpconf.sh
Restarting internet superserver: inetd.
copy pxe config from pxe.tmpl to worker01 (C0A80201)
copy pxe config from pxe.tmpl to worker02 (C0A80202)
copy pxe config from pxe.tmpl to worker03 (C0A80203)
copy pxe config from pxe.tmpl to worker04 (C0A80204)
```

A partir d'aquí FAI és totalment funcional i ja podríem instal·lar els nodes de càlcul. La post-instal·lació finalitza reconfigurant l'ntp, modificant el paràmetre de la swap del kernel, deshabilitant el servei firstboot i esborrant el directori amb els scripts de configuració.

```
+ ./ntpconf.sh
Stopping NTP server: ntpd.
Starting NTP server: ntpd.
+ ./cloudera.sh
vm.swappiness = 0
+ update-rc.d firstboot remove
update-rc.d: using dependency based boot sequencing
+ cp /root/firstboot/nohup.out /root/postconfig.log
```

6.3 Instal·lació dels nodes de càlcul

6.3.1 Exemple d'instal·lació d'un node

La instal·lació dels nodes de càlcul la realitzarem simplement arrencant-los, en el nostre cas creant les corresponents màquines virtuals. Per exemple, arrenquem el node worker01.

```
$> ./create_VM.sh worker01
Virtual machine 'worker01' is created and registered.
UUID: 06327d67-03a1-4e31-8f57-e168cc26e626
Settings file: '/home/user/VirtualBox VMs/worker01/worker01.vbox'
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Disk image created. UUID: 23bbacd8-8d14-4ec6-80f2-1fa690bfa898
/home/user
$> Oracle VM VirtualBox Headless Interface 4.3.12
(C) 2008-2014 Oracle Corporation
All rights reserved.

VRDE server is listening on port 3301.
```

Com mostra la última línia, la visualització remota en aquest cas es serveix pel port 3301, per tant podem obrir un *rdesktop* a aquest port per poder veure la consola.

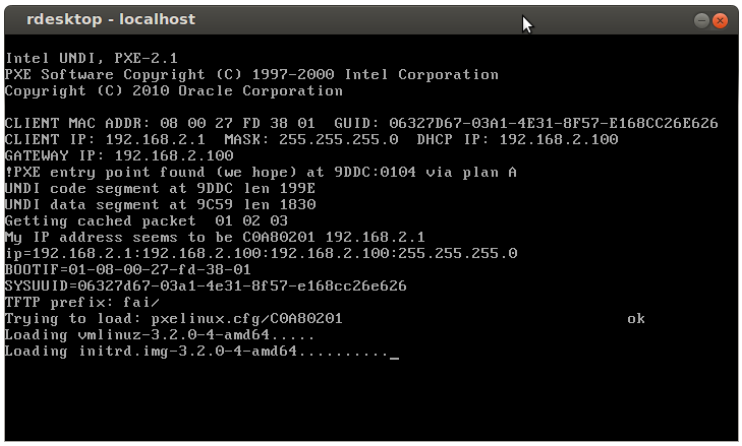

Figura 7: Arrencada del worker01 per PXE

A la Figura 7 podem veure que com la màquina virtual està configurada per arrencar per xarxa, el primer que fa és obtenir la seva IP (192.168.2.1) des

del servidor dhpc (192.168.2.100, login). Aquesta serà sempre la mateixa, tal i com es va assignar al fitxer de configuració *dhcpd.conf*. Posteriorment, obté del servidor tftp (node de login) tant el fitxer de configuració d'arrencada del node (*pxelinux.cfg/C0A80201*) com el kernel i el ramdisk. Un cop carregats aquest dos últims, comencen a executar-se totes les tasques de FAI (a la Figura 8 podem veure la tasca de particionat del disc i formatat de les particions).

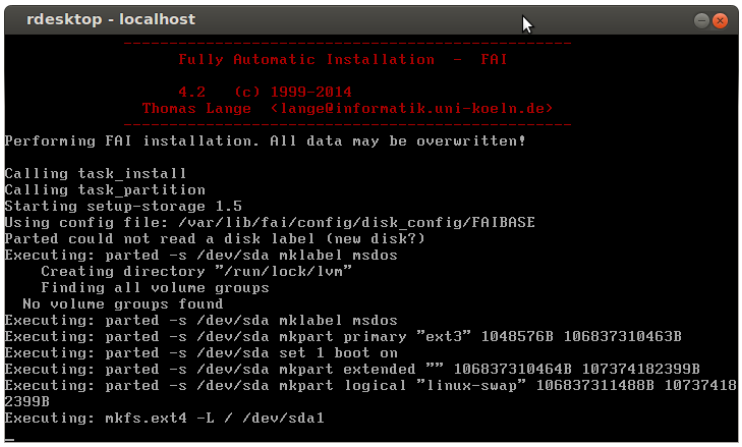

Figura 8: Inici de la instal·lació del worker01

Un cop FAI ha finalitzat totes les seves tasques, el node es rebota i podem veure els logs de la instal·lació dins el faiserver, on es guardaran dins el directori */var/log/fai/worker01/last*. Les últimes línies del fitxer fai.log són les que es poden veure a continuació.

```
Calling task_finish
Filesystem                    Type     Size  Used Avail Use% Mounted on
192.168.2.100:/srv/fai/nfsroot nfs      95G  2.6G   88G   3% /live/image
192.168.2.100:/srv/fai/config  nfs4     95G  2.6G   88G   3% /var/lib/fai/config
/dev/sda1                     ext4     98G  704M   93G   1% /target
eth0 rx_bytes 169.48 Mbytes
eth0 tx_bytes 3.35 Mbytes
Ramdisk on /target/var/lib/dpkg umounted
Mon Jun 16 12:52:07 CEST 2014
The install took 164 seconds.
Calling task_chboot
disable pxe config for 192.168.2.1 in hex C0A80201
Source hook: savelog.LAST.source
ERRORS found in log files. See /tmp/fai/error.log
savelog.LAST.source  OK.
Calling task_savelog
Save log files via ssh to fai@login:worker01//install−20140616_124933
```

La informació més interessant és potser el temps d'instal·lació total, que ha estat de poc menys de 3 minuts! També podem veure que una de les últimes tasques de FAI (task_chboot) és deshabilitar el fitxer de configuració d'arrencada del node dins el servidor de tftp. Finalment executa el hook savelog.LAST.source, guarda els fitxers de log al node de login i rebota el node. Quan aquest arrenca, com es pot veure a la Figura 9, el fitxer que carrega ara

des del servidor de tftp és *pxelinux.cfg/default*, i aquest li diu que faci el boot des del disc local. A no ser que vulguem fer alguna modificació del node des de FAI, a partir d'ara sempre arrencarà en local. Un cop el node està aixecat, podem veure per consola la pantalla de login del mateix (Figura 10)

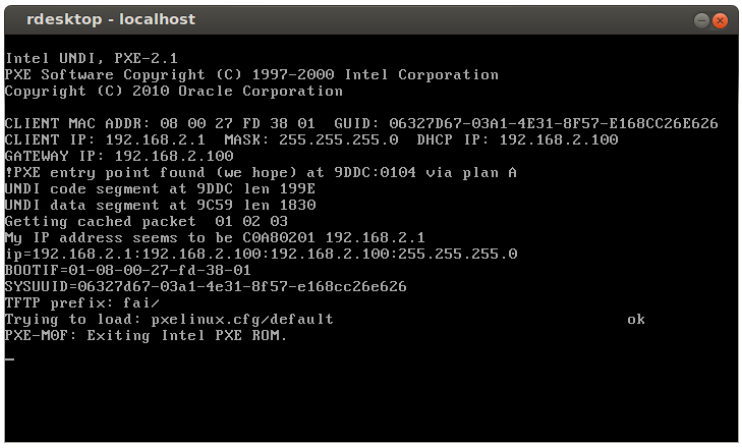

Figura 9: Arrencada del worker01 des del disc local

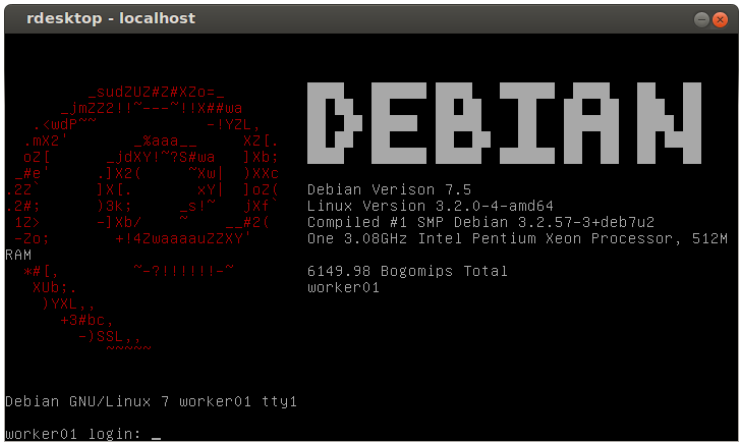

Figura 10: Pantalla de login del worker01 un cop finalitzada la instal·lació

6.3.2 Instal·lació massiva i monitorització

A l'apartat anterior hem vist un exemple de com és de senzilla la instal·lació i configuració automàtica d'un node de càlcul. Per a la instal·lació massiva i desatesa de tots els nodes del clúster utilitzarem la mateixa estratègia: arrencar els nodes.

En aquest cas, la monitorització de la instal·lació no és tan senzilla, doncs no és viable tenir obertes les consoles de molts nodes a la vegada. Per a seguir l'estat en que es troba el procés, podem fer una ullada al fixter de logs de fai-monitor, situat al faiserver. Com hem dit, a mida que els nodes vagin entrant i

sortint de les seves respectives tasques, quedarà registrat dins d'aquest fitxer.

```
FAI monitoring daemon starting..
FAI monitoring daemon started on port 4711 with pid 2888
worker01 check
worker01 TASKBEGIN confdir
worker01 TASKEND confdir 0
worker01 TASKBEGIN setup
worker01 TASKEND setup 0
...
worker02 check
worker02 TASKBEGIN confdir
worker02 TASKEND confdir 0
...
worker01 TASKEND savelog 0
192.168.2.1:35663: Disabling pxelinux configuration for worker01
...
worker01 TASKEND faiend 0
worker01 TASKEND reboot 0
```

Malgrat que l'observació d'aquest fitxer ja ens proporciona tota la informació que cal per seguir l'estat del desplegament del nostre clúster, la sortida no és còmoda de visualitzar. Les línies al log van apareixent a mida que els nodes canvien d'estat, però sense estar agrupades per màquina, i això fa que sigui difícil d'analitzar. Quan es realitza el desplegament de centenars de nodes, és més útil disposar d'una interfície gràfica, com la que proporciona fai-monitor-gui. Per tal que aquesta eina funcioni bé, cal realitzar unes passes prèvies dins el node de login, que no hem afegit al CD d'instal·lació de deboop perquè és una eina opcional, però que es podria considerar com una millora per a futures versions (veure Seccio 8.1).

```
$>apt-get install xauth libx11-dev make build-essential libpng-dev
$>cpan -i Tk
```

En aquest cas, per iniciar el servei, en comptes de fer-ho a traves del fitxer /etc/init.d/fai-monitor ho farem amb la següent comanda, que inicia el dimoni i passa la sortida a la interfície gràfica a través d'una *pipe*.

```
fai-monitor | fai-monitor-gui -
```

La Figura 11 mostra una finestra de la interfície gràfica, on es poden veure en columnes totes les tasques en que es divideix el procés d'instal·lació i en fileres els nodes que s'estan desplegant. En l'exemple mostrat el node worker01 ha acabat d'actualitzar la informació dels repositoris de paquets, mentre que el node worker02 està extraient el sistema base i els nodes worker03 i worker04 estan en la fase de particionat del seu disc dur. Un cop arribin a la tasca reboot, tots els nodes es reiniciaran i podrem comprovar des del node de login que ho hagin fet correctament (per exemple amb un ping).

```
$>for i in `seq -f192.168.2.%g 1 4`; do ping -c 1 $i; done
```

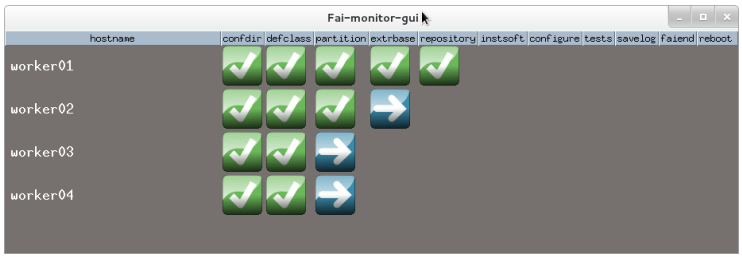

Figura 11: fai-monitor-gui, en el procés d'instal·lació dels nodes

7 Instal·lació del clúster de Hadoop

Un cop hem instal·lat el node de login i els nodes de càlcul, podem passar a configurar el clúster de Hadoop. Amb Cloudera Manager aquesta configuració és molt senzilla i ràpida. A continuació expliquem com es realitza dins el nostre entorn de màquines virtuals creat.

En arrencar el node de login, aquest ja aixeca els serveis que fa servir Cloudera Manager per configurar, gestionar i monitoritzar el clúster. El servidor disposa d'una interfície web per tal de realitzar totes les tasques, que està escoltant pel port 7180. Per tant, obrim un navegador apuntant a l'adreça del node de login i aquest port, i podrem veure la pantalla de connexió del manager (Figura 12).

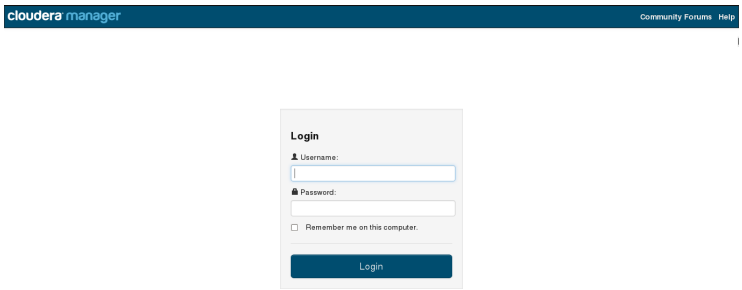

Figura 12: Pantalla de login de Cloudera Manager

Introduïm el nom d'usuari i la contrasenya (per defecte ambdós són la paraula "admin"), i entrem en una pantalla on ens demana escollir quina edició volem desplegar. Escollim la versió de franc (Cloudera Express), que inclou les funcionalitats bàsiques del manager i CDH (Cloudera Hadoop). A continuació ens mostra informació sobre els serveis de l'edició escollida i a la següent pantalla ens pregunta els hosts que formaran part del clúster, que introduirem amb el rang d'IP's dels nostres nodes (veure Figura 13) i pitjant el botó "Search". La pantalla següent mostra els nodes que el manager troba actius i que responen a l'ssh, i ens deixa escollir els que volem afegir al clúster (per defecte estan tots seleccionats).

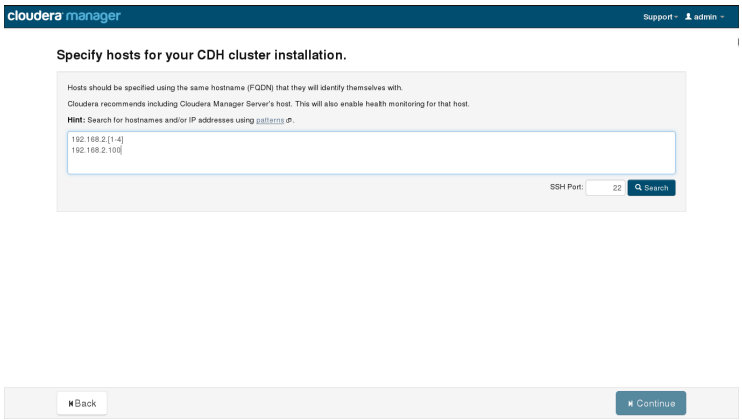

Figura 13: Introducció dels nodes del clúster

Si continuem a la següent pantalla, ens demana escollir els repositoris d'instal·lació. Deixem les opcions seleccionades per defecte, que utilitzen el que s'anomenen *parcels*, les quals permeten actualitzacions posteriors del clúster més senzilles que no pas amb paquets (Figura 14). La següent pantalla ens demana si volem utilitzar polítiques de xifrat de fortalesa il·limitada, i en el nostre cas, per fer un entorn de proves, no ho hem seleccionat.

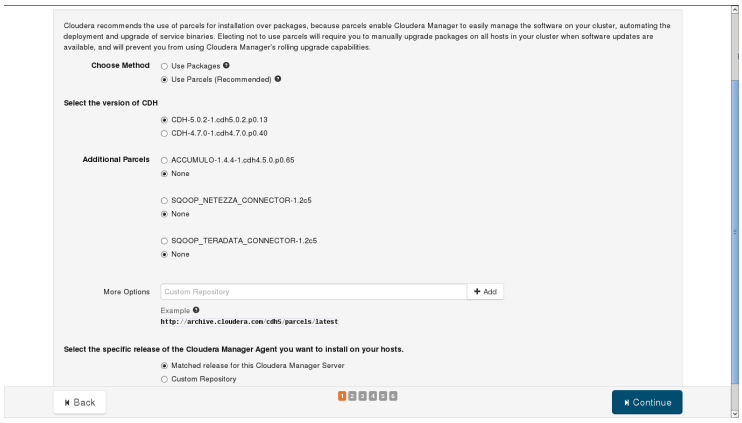

Figura 14: Selecció dels repositoris de CDH

En la següent pantalla hem d'introduir les credencials de l'usuari root (*toor*), per tal que des del node de login pugui accedir a la resta de nodes i instal·lar els paquets de les aplicacions necessàries, i just a continuació comença la instal·lació del clúster. A la Figura 15 podem veure els diferents nodes en diverses etapes de la configuració inicial (instal·lació dels agents de cloudera manager, java, etc. . .).

Un cop s'han instal·lat aquests serveis bàsics, es comença a descarregar dins el node de login els *parcels* escollits (en el nostre cas només CDH), per posteriorment desplegar-los a la resta de nodes (Figura 16).

A continuació, tal i com es mostra a la Figura 17, es passen uns tests de validació per comprovar que tots els serveis estiguin correctament instal·lats a

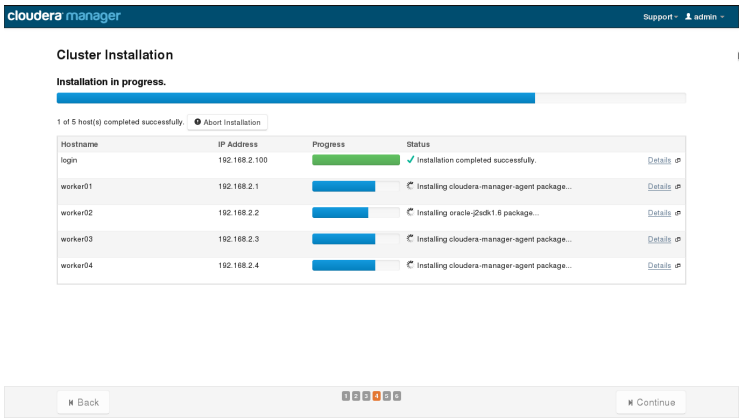

Figura 15: Comença la instal·lació del clúster

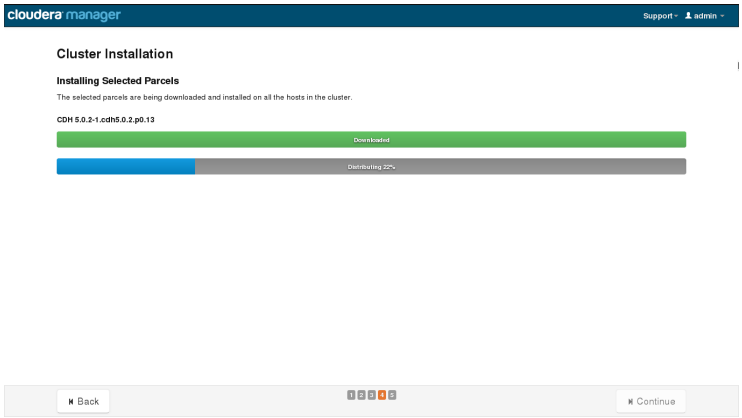

Figura 16: Descàrrega i desplegament dels parcels

tots els nodes i que les versions dels components siguin les mateixes (en aquest cas, com és la primera instal·lació no hi hauria d'haver cap problema, i la configuració a tots els nodes serà homogènia).

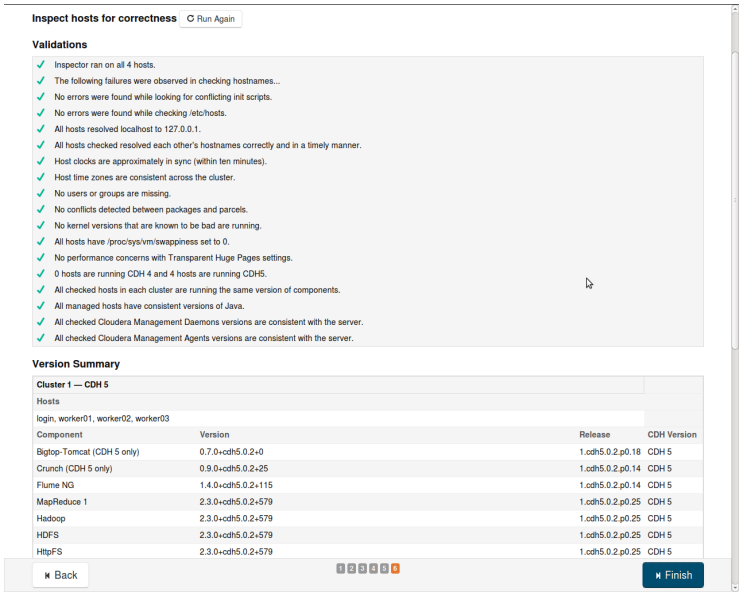

Figura 17: Inspecció dels hosts per comprovar que tot estigui correcte

En la següent pantalla escollim els serveis a instal·lar dins del clúster, que per al nostre entorn de proves seran només els bàsic (Core Hadoop, Figura 18).

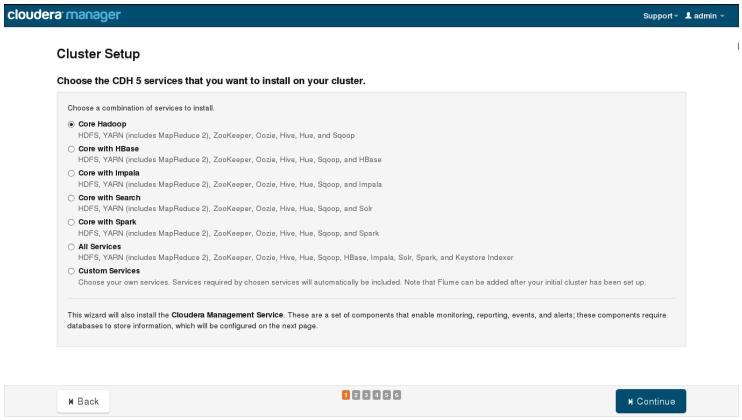

Figura 18: Selecció dels serveis a instal·lar

Continuem amb la personalització del rol que cadascun dels nodes prendrà dins del clúster (Figura 19). Per exemple, per al servei HDFS, quin serà el *namenode* i quins seran els *datanodes*. En principi ho deixem amb les opcions que ens dona el manager.

La següent pantalla serveix per configurar i comprovar que la connexió amb la base de dades funcioni correctament (Figura 20). Escollim fer servir la base

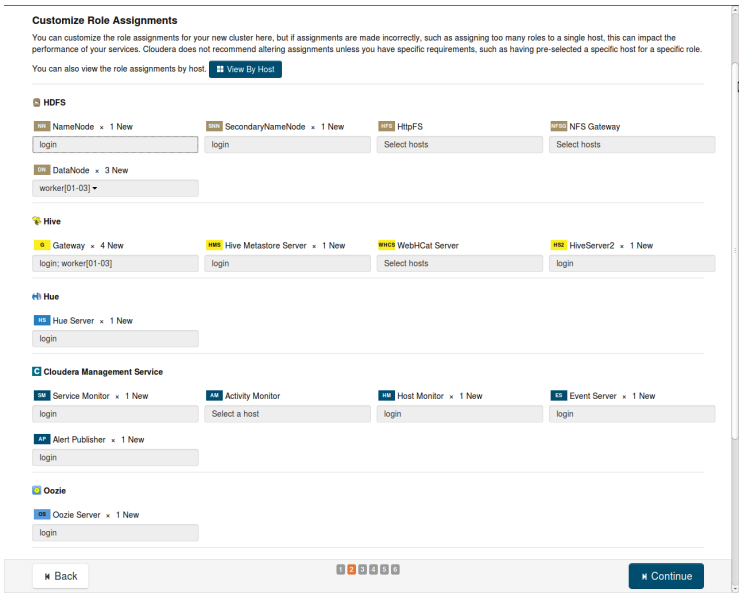

Figura 19: Rols dels nodes

de dades *embedded*, amb el qual es posposa el test, ja que aquesta base de dades es crearà en una etapa posterior.

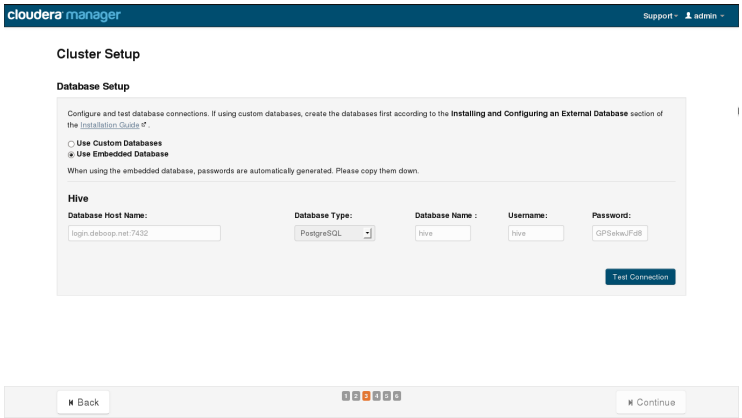

Figura 20: Configuració de la base de dades

Continuem amb una pantalla on ens permet modificar els valors d'alguns paràmetres de funcionament dels diferents serveis (Figura 21), com per exemple la localització dins del sistema de fitxers local dels *datanodes* a on HDFS guardarà les dades, entre d'altres.

I si continuem amb la instal·lació, finalment instal·la tots els serveis que hem escollit anteriorment (Figura 22).

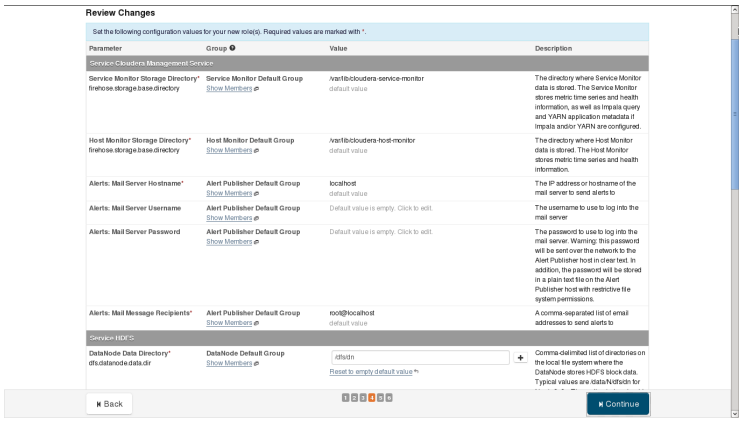

Figura 21: Revisió d'alguns paràmetres dels serveis

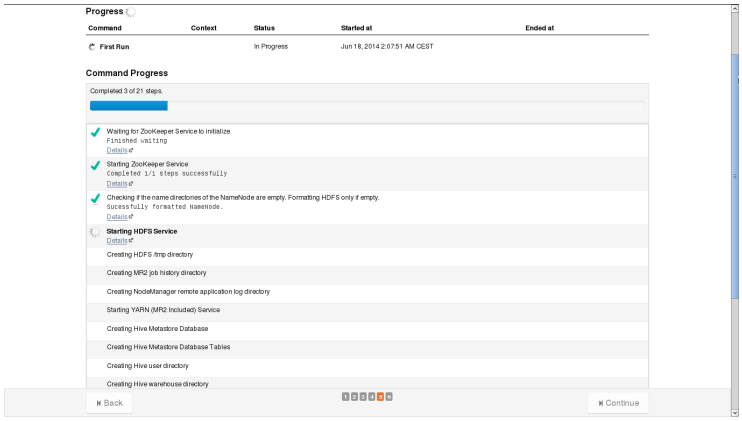

Figura 22: Instal·lació dels serveis

Atenció: en el cas que falli la creació de les tables de la base de dades de Hive Metastore, cal editar el fitxer /etc/hosts i afegir a la línia del host "login" la cadena "login.deboop.net". No podem fer aquesta modificació amb anterioritat, perquè sinó falla la primera etapa de la instal·lació del manager (no és capaç de detectar el heartbeat del node de login).

Un cop finalitza la instal·lació de tots els serveis s'obre el panell principal de Cloudera Manager (Figura 23). Des d'aquest panell podem gestionar i monitoritzar tots els nodes i tots els serveis que conformen el clúster de Hadoop. És una interfície molt treballada, que mostra infinitat de dades i mètriques, i amb moltes gràfiques que permeten fer anàlisi i comprovar l'estat del clúster.

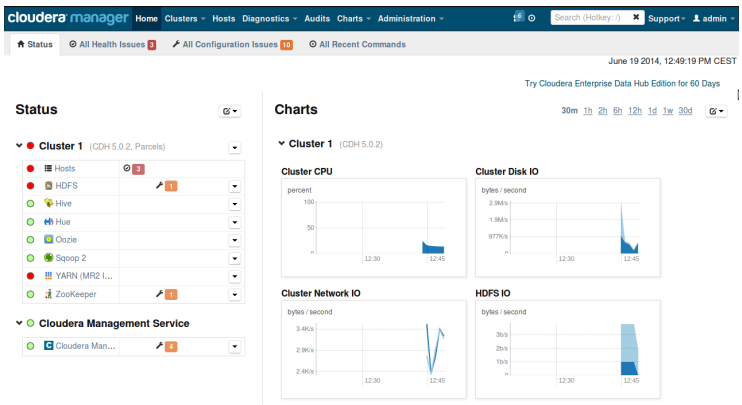

Figura 23: Instal·lació dels serveis

8 Millores futures

Durant l'elaboració de la distribució han anat sorgint algunes idees de possibles millores futures, així com també de línies de treball complementaries o alternatives. Exposem en aquesta secció un recull de totes aquestes idees.

8.1 Ajust final

La primera millora que es pot aplicar a la distribució és la de trobar mecanismes que permetin personalitzar-la de forma que s'adapti exactament a les instal·lacions de l'usuari final. En aquest sentit, existeixen tota una sèrie de paràmetres que nosaltres hem imposat dins dels scripts de post-configuració, però

que l'usuari hauria de poder escollir en temps d'instal·lació. Caldrà crear pantalles amb l'eina dialog [23], que permetin seleccionar aquests paràmetres amb caixes de menús o *checklists*. Segurament n'hi haurà d'altres, però en principi els que hem detectat són els següents.

- Configuració de la xarxa interna, numeració d'adreces IP, gateway de la xarxa. És a dir, l'usuari hauria de poder generar el seu propi fitxer d'script addnet.sh (Fitxer 8).
- Noms dels nodes de treball. Personalitzar la nomenclatura dels hostnames suposa que l'usuari pugui crear el seu script addhosts.sh (Figura 10). Això també afecta a la generació del fitxers de configuració d'arrencada creats amb tftpconf.sh)
- Llistat de macaddress dels nodes de càlcul. Caldria cercar un mètode que permeti obtenir aquest llistat i afegeixi automàticament la informació dins el fitxer dhcpd.conf del node de login.
- Modificació dels paràmetres de *preseeding*, tant per a la instal·lació del node de login com els de l'espai de configuració de FAI. Nosaltres hem contemplat un model amb un sol disc, però caldria tenir en compte configuracions de disc més complexes, amb diversos discs que poden estar en RAID, o que continguin volums lògics.

Una altra millora, més aviat estètica, seria acabar de personalitzar bé la distribució, en el sentit de modificar tots els llocs on apareix la paraula Debian per substituir-la per Deboop. El cas més visible és la pantalla d'inici de Grub, però també els fitxers /etc/issue, o /etc/os-release, entre d'altres.

Finalment, també havíem comentat a la secció 6.3.2 que per fer més còmode la visualització del desplegament dels nodes de càlcul, seria interessant disposar de l'eina fai-monitor-gui dins el node de login. Per a tal fi, caldria afegir la configuració necessària dins l'script postinstall.sh.

8.2 Comparatives de rendiment

Una possible línia de treball és la realització de *benchmarks* per a estudiar el rendiment del clúster de Hadoop instal·lat amb Cloudera. Caldria realitzar la instal·lació en màquines físiques i buscar aplicacions que puguin utilitzar-se per a dur a terme aquests estudis.

Si anem un pas més enllà, es podria estudiar la possibilitat de portar l'aplicatiu cap al *cloud* amb l'objectiu de comparar el rendiment d'una i altre solució. Es podria fer us de la capa d'AWS d'Amazon per realitzar els mateixos *benchmarks* d'aplicacions que els executats al nostre clúster. De la comparativa podem extreure conclusions sobre on surt més a compte realitzar els càlculs en termes de velocitat d'obtenció dels resultats. Però també es podria completar l'estudi afegint una valoració de preus i costos de córrer el càlcul en Amazon i en local.

En el cas de valorar l'ús d'Amazon per a tasques de producció rutinàries, es podria considerar la creació d'una interfície capaç de generar instancies de MapReduce de manera dinàmica al *cloud*. Aquesta interfície podria escollir entre enviar els càlculs en local o al *cloud* (depenent dels costos estimats que

tindria aquell càlcul), de manera que la interacció amb la capa de *cloud* seria el més transparent possible per a l'usuari.

Referències

[1] URL: http://hadoop.apache.org/.

[2] Viquipèdia. *Big Data — Viquipèdia, l'Enciclopèdia Lliure*. 2014. URL: http://ca.wikipedia.org/wiki/Big_Data (cons. 26-3-2014).

[3] Douglas Laney. *3D Data Management: Controlling Data Volume, Velocity, and Variety*. Gartner, feb. de 2001.

[4] URL: http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html.

[5] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.

[6] URL: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[7] Jeffrey Dean i Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. Google Inc., 2004.

[8] URL: http://www.cloudera.com/content/cloudera/en/home.html.

[9] URL: https://www.virtualbox.org/.

[10] URL: https://wiki.debian.org/Simple-CDD/Howto.

[11] URL: http://fai-project.org/.

[12] URL: http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html.

[13] URL: https://www.debian.org/.

[14] URL: https://wiki.debian.org/debian-cd.

[15] URL: https://packages.debian.org/wheezy/debconf.

[16] URL: http://cdimage.debian.org/debian-cd/7.5.0/amd64/iso-cd/debian-7.5.0-amd64-netinst.iso.

[17] URL: http://www.openssh.com/.

[18] URL: http://www.squid-cache.org/.

[19] URL: http://www.unix-ag.uni-kl.de/~bloch/acng/.

[20] URL: http://rsync.samba.org/.

[21] URL: http://www.ntp.org/.

[22] URL: http://www.postgresql.org/.

[23] URL: http://hightek.org/projects/dialog/.

[24] URL: http://www.gnu.org/software/xorriso/.

[25] URL: http://www.isc.org/downloads/DHCP/.

[26] Intel Corporation. *Preboot Execution Environment (PXE) Specification - Version 2.1*. 1999. URL: ftp://download.intel.com/design/archives/wfm/downloads/pxespec.pdf (cons. 29-3-2014).

[27] URL: https://www.kernel.org/pub/software/network/tftp/.

[28] URL: http://www.virtualbox.org/manual/ch08.html.

[29] URL: https://wiki.debian.org/Debootstrap.