EIMT.UOC.EDU
Estudis d'Informàtica, Multimèdia i Telecomunicació

UOC Universitat Oberta de Catalunya

CSUC Consorci de Serveis Universitaris de Catalunya

▪ Enginyeria Informàtica ▪

Sistemes d'emmagatzematge al núvol orientats a objecte

PFC-Aplicacions i sistemes distribuïts

Autor: Miguel Ángel Flores Terrón
Consultor: Félix Freitag

Juny, 2014

Agraïments

Les últimes línies de la memòria les vull dedicar a totes les persones que m'han donat suport i han fet possible que aquesta segona "aventura", que vaig iniciar fa uns anys de convertir-me en ENGINYER estigui una mica més a prop. Moltes gràcies a tots.

Al meu consultor del projecte. Al Félix Freitag pel seu suport durant aquest semestre.

A Jordi Guijarro i al CSUC, per l'oportunitat de realitzar-me com a Enginyer i oferir-me la possibilitat de realitzar aquest projecte.

A la meva família i amics, per la resta de coses que segur que sense ells res d'això hagués estat possible.

Sonríe! ツ

Índex

Introducció 7

1.1 Origen i motivació 7

1.2 Història i antecedents del CSUC 8

1.3 Objectius i metodologia 9

1.4 Organització de la memòria 10

Conceptes bàsics 12

2.1 Cloud Computing 12

2.2 Big Data 12

2.2.1 Les cinc V 13

2.2 NoSQL 14

2.3 Teorema de CAP 15

Plataformes d'emmagatzematge al núvol 16

3.1 Cloudian 16

3.2 OpenStack Storage (Swift) 17

3.3 Riak 18

3.3.1 Riak CS 18

3.4 Comparativa Riak amb Cassandra 19

3.6 Casos d'ús 24

Especificació Prova de Concepte (POC) 25

4.1 Elecció de la plataforma d'emmagatzematge 25

4.2 Descripció de la prova de concepte 25

4.3 Diagrama de l'arquitectura 26

Riak /Riak CS 28

5.1 Què és Riak? 28

5.1.1 Clustering 29

5.1.2 Replicació 30

5.2 Què és Riak CS? 31

5.3 Característiques principals Riak CS 31
5.4 Arquitectura Riak CS 32
Implementació Prova de Concepte (POC) 33
6.1 Arquitectura de la Prova de Concepte (POC) 33
6.2 OpenNebula 34
6.3 Instal·lació del nodes 36
6.4 Diagrama amb l'arquitectura del servei 37
Proves Funcionals 38
7.1 Accedir a l'administració del *clúster* Riak 38
7.1.1 Vista *Snapshot* 38
7.1.2 Vista *Cluster Management* 39
7.1.3 Vista *Node Management* 39
7.1.4 Vista *Ring* 40
7.2 Accedir al monitoratge de la BBDD del *clúster* 40
7.3 Accedir a l'administració d'usuaris de Riak CS 41
7.3 Configurar un client S3: DragonDisk 42
7.3.1 Creació d'un compte 42
7.3.2 Connectar-se al clúster Riak CS 44
7.4 Configurar un client S3 per consola 45
7.5 Operacions bàsiques amb Riak CS 45
7.5.1 Creació d'un *bucket* 45
7.5.2 Copiar fitxers en un *bucket* 47
7.6 Estadístiques amb Riak CS 48
Conclusions 50
8.1 Quant als objectius 50
8.2 Estudi econòmic 50
8.2.1 Pressupost del personal 50
8.2.2 Pressupost en materials i maquinari 51
8.2.3 Pressupost en llicències i programari 52
8.2.4 Pressupost total del projecte 52
8.3 Planificació temporal 52
8.4 Treball futur 54
Annex I. Instal·lació Riak / Riak CS 55
Annex II. Arxiu de configuració HAProxy 57

Annex III. Arxiu de configuració .s3cfg 58

Annex IV. Operativa bàsica amb s3cmd 59

Annex V. Estadístiques Riak CS 60

Llista de Figures

Figura 1. Informació o volum de dades generades en una 1 minut 7
Figura 2. Esquema amb les cinc V del Big Data 14
Figura 3. Esquema del Teorema de CAP 15
Figura 4. Arquitectura d'una plataforma amb Cloudian 16
Figura 5. Esquema de distribució del components d'OpenStack Swift 18
Figura 6. Esquema d'emmagatzematge de RiakCS 19
Figura 7. Diagrama de l'arquitectura de la prova de concepte 26
Figura 8. Representació de l'anell Riak 29
Figura 9. Exemple de la replicació en l'anell Riak 30
Figura 10. Exemple de sanació en l'anell Riak 31
Figura 11. Esquema de l'arquitectura d'emmagatzematge de RiakCS 32
Figura 12. Servidor d'emmagatzematge Supermicro 33
Figura 13. Logotip OpenNebula 34
Figura 14. Arquitectura del servei cloud d'OpenNebula 35
Figura 15. Aplicació Web (Sunstone) per administrar les màquines virtuals 36
Figura 16. Diagrama amb l'arquitectura del servei 37
Figura 17. Vista Snapshot de l'aplicació Riak Control 38
Figura 18. Vista Cluster Management de l'aplicació Riak Control 39
Figura 19. Vista Node Management de l'aplicació Riak Control 39
Figura 20. Vista Ring de l'aplicació Riak Control 40
Figura 21. Aplicació web amb les estadístiques del proxy 41
Figura 22. Vista del formulari per crear usuaris a Riak CS 41
Figura 23. Captura de pantalla amb el llistat d'usuaris de Riak CS 42
Figura 24. Pantalla inicial DragonDisk 42
Figura 25. Creació d'un nou compte a DragonDisk 43
Figura 26. Configuració d'un compte amb DragonDisk I 43
Figura 27. Configuració d'un compte amb DragonDisk II 44
Figura 28. Pantalla amb una connexió S3 al clúster Riak CS 45
Figura 29. Vista de creació d'un bucket amb DragonDisk 46
Figura 30. Quadre de diàleg per crear un bucket amb DragonDisk 46
Figura 31. Transferències d'arxius amb Dragondisk I 47
Figura 32. Transferències d'arxius amb Dragondisk II 48
Figura 33. Estadístiques d'accés en format JSON de Riak CS 49

Figura 34. Estadístiques d'emmagatzematge en format XML de Riak CS 49

Figura 35. Diagrama de Gantt inicial 53

Figura 36. Diagrama de Gantt final amb el camí crític 54

Llista de Taules

Taula 1. Comparativa Riak amb Cassandra 22

Taula 2. Comparativa Riak CS amb Swift 24

Taula 3. Comparativa plataformes d'emmagatzematge 25

Taula 4. Comparativa servidors d'emmagatzematge 34

Taula 5. Tipus de màquines virtuals disponibles a OpenNebula 36

Taula 6. Pressupost del personal 51

Taula 7. Pressupost del maquinari 51

Taula 8. Pressupost del programari 52

Taula 9. Pressupost total del projecte 52

Taula 10. Tasques que composen el projecte 53

Capítol 1
Introducció

En aquest projecte es proposa la implementació d'una plataforma amb un sistema d'emmagatzematge públic allotjat en el núvol, per implementar la plataforma, es vol fer servir programari lliure (*open-sourse*) basat en bases de dades distribuïdes (*NoSQL*), alguns exemples concrets d'aquestes plataformes serien: *Cloudian* [1] (basat en una base de dades *NoSQL*: *Cassandra* [2]), *Riak CS* [3] (basat en una base de dades distribuïda com ara *Riak* [4]) o bé *Swift* (basat en *OpenStack* [5]).

1.1 Origen i motivació

Cada cop cal satisfer volums d'emmagatzematge més grans, és per aquest motiu que els sistemes han de ser capaços d'atendre grans càrregues de treball amb un elevat índex de lectures i escriptures a una mateixa dada emmagatzemada de forma concurrent, com a fitxers de base de dades o imatges de màquines virtuals. Per aquest motiu podem arribar a pensar, que les dimensions de les dades han deixat obsoletes les tecnologies clàssiques i cal trobar solucions emmagatzematge massiu i processat distribuït.
En la Figura 1, es pot observar la quantitat d'informació o el volum de dades que es genera actualment en un dia qualsevol en només un minut.

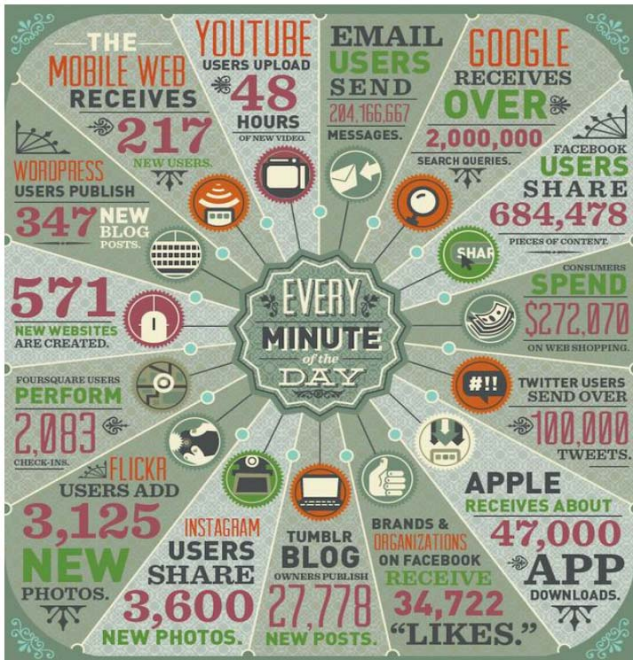

Figura 1. Informació o volum de dades generades en una 1 minut

Per això aquest projecte proposa una solució basada en un sistema d'emmagatzematge públic allotjat en el núvol (*Cloud Computing*). Aquest tipus de sistemes són molt escalables i són capaços de suportar altes càrregues de treball, buscant la fluïdesa i la consistència òptimes d'acord a les necessitats dels usuaris. En l'actualitat trobem empreses que ofereixen aquest tipus de serveis com *Amazon Simple Storage Service* [6], més conegut com a S3 (una senzilla interfície de serveis web que pot utilitzar-se per emmagatzemar i recuperar la quantitat de dades que desitgi a través d'Internet), per fer-nos una idea de l'atractiva que resulta aquesta tecnologia, comentar una sèrie de dades orientatives. A l'abril del 2010, *Amazon.com* va reportar que 100 mil milions d'arxius havien estat emmagatzemats en S3, un 100% del seu creixement pel que fa a l'any anterior. A l'octubre del 2011, aquesta quantitat en S3 havia pujat als 566 bilions.

En resum, aquest projecte té com a objectiu principal l'estudi, anàlisi i posterior implementació d'una *proof of concept* (prova de concepte) d'una plataforma d'emmagatzematge al núvol amb programari *open-source* compatible amb la API S3 de *Amazon*.

El projecte s'emmarca dins de la realització d'unes pràctiques externes no curriculars dins d'un programa de cooperació educativa amb el Consorci de Serveis Universitaris de Catalunya (CSUC) [7].

El CSUC pretén poder incorporar i oferir en un futur serveis d'emmagatzematge al núvol al seus clients http://www.csuc.cat/ca/recerca/infraestructura-al-nuvol.

1.2 Història i antecedents del CSUC

A principis dels 90, la Generalitat de Catalunya havia impulsat la creació del CESCA, aleshores sota la denominació de Centre de Supercomputació de Catalunya, a través de la llavors Fundació Catalana per a la Recerca i amb la col·laboració de les universitats i el CSIC. Aquest Centre ha estat reconegut com a instal·lació científica i tècnica singular (ICTS, 2000) i ha rebut la Placa Narcís Monturiol (2001).

El CESCA va ser creat amb l'objectiu de compartir una infraestructura de càlcul d'altes prestacions que la comunitat cientificotècnica necessitava per desenvolupar la seva recerca en diverses àrees de coneixement, com ara la química teòrica, la meteorologia o l'astrofísica. Aquest servei es va començar a prestar amb un IBM 3090 i un Cray X-MP. Més tard, i per al camp de les ciències de la vida, es va crear un servei de cerca de farmacòfors.

Els serveis del CESCA es van estendre a les comunicacions, i l'any 1993 es va crear l'Anella Científica, una xarxa de transmissió de dades d'alta velocitat que va ser pionera a Europa. Aquesta infraestructura connecta les institucions catalanes amb les xarxes internacionals de recerca a través del node de RedIRIS que, des de 1996, el CESCA ha allotjat. A més d'accés a internet, l'Anella Científica proporciona nombrosos serveis de valor afegit com ara certificació digital (2000), videoconferència i veu sobre IP (2002), seguretat (2003) i mobilitat amb Eduroam (2005).

L'any 1999 es va crear el Punt Neutre d'Internet a Catalunya (CATNIX) per afavorir l'intercanvi de tràfic entre els diferents proveïdors d'internet. Els promotors del CATNIX van demanar al CESCA que l'hostatgés i administrés, cosa que ha fet des d'aleshores.
L'any 2001 es va estrènyer la col·laboració amb el CBUC, posant en marxa conjuntament Tesis Doctorals en Xarxa (www.tdx.cat), el primer de diversos repositoris digitals creats per incrementar l'accessibilitat i la visibilitat dels continguts de les institucions que hi participen.
Per emfatitzar el suport i els serveis TIC que s'oferien a les universitats catalanes, més enllà de la supercomputació, l'any 2009 es van reformar els Estatuts del CESCA, que va passar a denominar-se Consorci Centre de Serveis Científics i Acadèmics de Catalunya. Aquest canvi va suposar també la creació d'una sèrie de serveis comuns d'administració electrònica: plataforma de vot, de registre telemàtic d'entrada i sortida, d'arxiu i preservació de documents digitals...
El 10 de desembre de 2013 naixia el Consorci de Serveis Universitaris de Catalunya (CSUC) per impulsar l'eficiència en la gestió de les universitats catalanes a través de la cooperació i la coordinació.
Creat a iniciativa de la Generalitat de Catalunya, el CSUC integra dos consorcis que, fins aleshores, havien treballat promovent i oferint serveis consorciats en els àmbits TIC i bibliotecari respectivament, el Consorci Centre de Serveis Científics i Acadèmics de Catalunya (CESCA) i el Consorci de Biblioteques Universitàries de Catalunya (CBUC).
El CSUC treballa amb l'objectiu fonamental de compartir o mancomunar serveis acadèmics, científics, bibliotecaris, de transferència de coneixement i de gestió de les entitats consorciades per potenciar les sinergies i millorar-ne l'eficiència.
El CSUC pretén mantenir i fer créixer els serveis de CESCA i CBUC, i incorporar noves línies d'activitat, com ara les compres conjuntes, i qualsevol àmbit en què la col·laboració i la compartició puguin beneficiar el sistema universitari i de recerca.

1.3 Objectius i metodologia

Així doncs, els principals objectius plantejats per aquest projecte són els següents:

- Estudi i formació en *Big Data* i bases de dades distribuïdes (*NoSQL*).
- Estudi i formació en conceptes bàsics sobre sistemes d'emmagatzematge al núvol.
- Anàlisis de les diferents alternatives *open-source* per construir plataformes d'emmagatzematge al núvol Cloudian, Riak CS, OpenStack Swift, etc...
- Dissenyar i implementació d'una *proof of concept* (prova de concepte) amb la plataforma triada, un cop analitzades les diferents solucions.
- Descripció del funcionament i proves funcionals de la plataforma.
- Redacció de la documentació i memòria del treball.

La metodologia que s'ha seguit per a dur a terme el projecte es va descriure al Pla de treball elaborat, a continuació s'enumeren els punts principals:
Primer, es va dividir en el projecte en quatre grans fases:

- **Fase 1.** Anàlisis i planificació
- **Fase 2.** Especificació de la prova de concepte (POC)
- **Fase 3.** Implementació de la prova de concepte (POC)
- **Fase 4.** Documentació

Cadascuna amb un seguit de tasques i un objectiu o fita ha assolir en cadascuna que concloïa amb el lliurament d'una entrega (Pla de treball , PAC2, PAC3, Lliurament Final).
Per fer el seguiment de les tasques s'han utilitzat diverses eines de gestió de projecte, com ara: *Microsoft Project* per l'elaboració del diagrama de Gantt, *Jira Agile* (https://www.atlassian.com/es/software/jira/agile) per al seguiment de les tasques i *Confluence* (https://www.atlassian.com/es/software/confluence) en la part de documentació de procediments.

1.4 Organització de la memòria

En aquest apartat s'enumeren i expliquen cadascun dels capítols dels que es composa aquesta memòria:

- **Introducció** (capítol 1): En aquest capítol, queda definit tant el projecte com els seus objectius, la motivació que ens ha dut a realitzar-lo i la metodologia a seguir per dur-lo a terme satisfactòriament. Finalment, una petita descripció de les parts que composen la memòria.
- **Conceptes bàsics** (capítol 2): En el segon capítol, es parla sobre una sèrie de conceptes molt importants relacionats amb la tecnologia que es veurà en els següents capítols de la memòria.
- **Plataformes d'emmagatzematge al núvol** (capítol 3): En aquest apartat, és fa una descripció de tres de les principals plataformes d'emmagatzematge al núvol de codi obert (*open-source*) que podem trobar en la actualitat.
- **Especificació Prova de Concepte (POC)** (capítol 4): En el cinquè capítol de la memòria, es mostra la especificació de la prova de concepte que es vol realitzar sobre una de les plataformes d'emmagatzematge al núvol, estudiades al tercer capítol.
- **Riak / Riak CS** (capítol 5): En aquest capítol, es detalla en profunditat les principals característiques, així com el funcionament general de la plataforma d'emmagatzematge al núvol escollida.
- **Implementació Prova de Concepte (POC)** (capítol 6): En el sisè capítol, s'expliquen els principals o més rellevants aspectes sobre la implementació de la prova de concepte.

- **Proves funcionals** (capítol 7): En aquest capítol, es descriuen les principals proves funcionals realitzades sobre la plataforma implementada a la prova de concepte per veure es seu correcte funcionament.
- **Conclusions** (capítol 8): En l'últim capítol es fa un balanç del compliment dels objectius plantejats, l'ajust a la planificació temporal prevista, un estudi econòmic del projecte i una reflexió sobre les possibles línies de futur obertes a partir d'aquest projecte.

Capítol 2
Conceptes bàsics

En aquest capítol, es parla sobre una sèrie de conceptes molt importants relacionats amb la tecnologia que es veurà en els següents capítols de la memòria.

2.1 Cloud Computing

Des de fa uns anys la paraula *cloud computing* [8] està en boca de molta gent, però encara és un terme que està molt confús degut a que és una metodologia molt amplia i amb moltes diferents variants, així doncs començarem aquest capítol amb una definició feta pel National Institute of Standards and Technology (NIST).

"Cloud computing és un model per a permetre ***accés sota demanda****, a través de la xarxa, de conjunts de* ***recursos de computació*** *configurables i compartits (p. ex. xarxes, servidors, emmagatzemament, aplicacions i serveis) que poden ser* ***ràpidament aprovisionats*** *i publicats amb el mínim esforç i interacció per part del proveïdor del servei. Aquest model promou la* ***disponibilitat*** *i està composat de cinc característiques essencials, tres models de servei i quatre models de desplegament."*

U. S. National Institute of Standards and Technology (NIST)

Com podem veure en la definició, el *cloud computing* ens pot donar diferents tipus de serveis, com pot ser emmagatzemament, computació, aplicacions en línia,... També segons els model de desplegament tenim diferents tipus de *cloud*, per aquest tipus de classificació un *cloud* pot ser públic, comunitari, privat i híbrid. Veiem doncs com la varietat en aquest paradigma és molt gran i encara avui estan sorgint nous tipus de serveis en *cloud*.

2.2 Big Data

Darrerament, estem veient que la manera en la qual els usuaris interactuen amb la tecnologia ha canviat de manera radical a causa de la constant evolució d'aquesta. Revolucions tecnològiques com la web 3.0, *blogs*, fòrums d'opinió, xarxes socials, multimèdia, etc... i dispositius com els telèfons intel·ligents faciliten la connectivitat i la generació de grans quantitats d'informació que fins a fa molt poc eren impensables. I no solament la societat de consum ha avançat tecnològicament; camps com la ciència,

medicina o l'economia també requereixen cada vegada més tractar amb grans quantitats de dades.

Big Data [8] és el sector de les tecnologies de la informació i la comunicació (TIC) que es preocupa de com emmagatzemar i tractar grans quantitats d'informació o conjunts de dades.

> *"Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."*
>
> *Gartner, 2012*

2.2.1 Les cinc V

És comú quan es parla de *Big Data* [9] es faci referència a grans quantitats de dades. Per descriure millor el que representa, freqüentment es parla de les cinc V [10], veure Figura 2. IBM va ser la que va començar definint tres V i després s'han afegit les altres dues, depenent de la font, que defineixen perfectament els objectius que aquest tipus de sistemes busquen aconseguir:

- **Volum**: Un sistema *Big Data* és capaç d'emmagatzemar una gran quantitat de dades mitjançant infraestructures escalables i distribuïdes. En els sistemes d'emmagatzematge actuals comencen a aparèixer problemes de rendiment en tenir quantitats de dades de l'ordre de magnitud de *petabytes* o superiors. Big Data està pensat per treballar amb aquests volums de dades.

- **Velocitat**: Una de les característiques més importants és el temps de processament i resposta sobre aquests grans volums de dades, obtenint resultats en temps real i processant-los en temps molt reduïts. I no només es tracta de processar sinó també de rebre, avui dia les fonts de dades poden arribar a generar molta informació cada segon, obligant al sistema receptor a tenir la capacitat per emmagatzemar aquesta informació de manera molt veloç.

- **Varietat**: Les noves fonts de dades proporcionen nous i diferents tipus i formats d'informació als ja coneguts fins ara, com a dades no estructurades, que un sistema *Big Data* és capaç d'emmagatzemar i processar sense haver de realitzar un preprocés per estructurar o indexar la informació.

- **Variabilitat**: Les tecnologies que componen una arquitectura *Big Data* han de ser flexibles a l'hora d'adaptar-se a nous canvis en el format de les dades -tant en l'obtenció com en l'emmagatzematge- i el seu processament. Es podria dir que l'evolució és una constant en la tecnologia de manera que els nous sistemes han d'estar preparats per admetre'ls.

- **Valor**: L'objectiu final és generar valor de tota la informació emmagatzemada a través de diferents processos de manera eficient i amb el cost més baix possible.

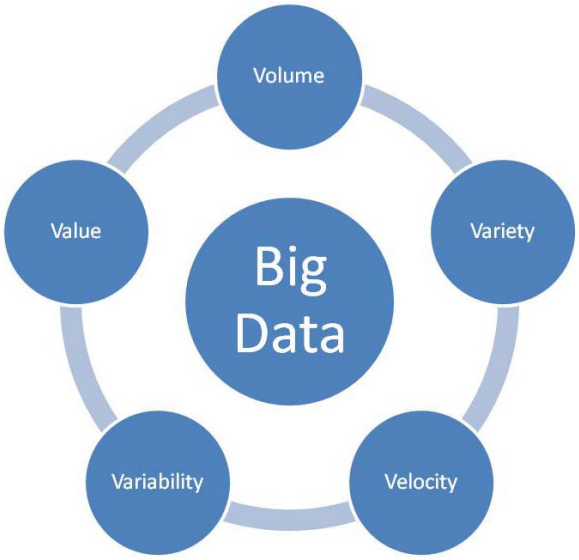

Figura 2. Esquema amb les cinc V del Big Data

2.2 NoSQL

NoSQL, *"Not Only SQL"* [10] neix com a un nou paradigma de la tecnologia de base de dades, per si mateixa no es pot fer servir per a totes les aplicacions i per tant és aconsellable tenir diferents tipus de bases de dades per a diferents tipus de demanda. Aquesta tecnologia no està basada en un model de base de dades relacional: no usa esquemes fixes ni té relacions entre taules. Tampoc utilitza cap estàndard de llenguatge de consulta declaratiu com el SQL sinó que cada gestor NoSQL concret té el seu propi llenguatge per a realitzar les consultes i les operacions.
NoSQL ofereix avantatges d'escalabilitat, disponibilitat i tolerància d'errors a part de tenir una estructura molt més simple que els SGBD relacionals.

A continuació s'explicarà breument cadascun d'aquests avantatges:

- **Escalabilitat:** El volum de dades a les aplicacions, com s'ha comentat, està creixent a un ritme inversemblant, de tal manera que avui dia disposar d'un Terabyte d'informació és una quantitat usual a moltes aplicacions. D'altra banda, els dispositius portàtils que poden connectar-se a Internet cada vegada tenen més popularitat i juntament amb l'augment de l'ample de banda disponible provoca que qualsevol aplicació ha de ser capaç de suportar una gran quantitat de dades que moltes bases de dades no poden sostenir.

- **Disponibilitat:** La disponibilitat es defineix com la capacitat de resoldre totes les sol·licituds rebudes per a un node que no estigui caigut. S'entén node, com a màquina física que forma part d'un sistema compost per diverses màquines. Així doncs, NoSQL es presenta com a alternativa, ja que una de les seves propietats és

dotar de les aplicacions d'una disponibilitat completa. Normalment a través de la redundància i relaxant la consistència.

- **Tolerància d'errors**. Es defineix la tolerància d'errors com a la capacitat d'una base de dades a continuar operant després que un node falli si el nombre de nodes caiguts és significament inferior al nombre total de nodes.

2.3 Teorema de CAP

El teorema de CAP el va presentar *Eric Brewer*, professor de sistemes a la Universitat de Califòrnia (Berkeley) i a l'any 2000 gerent de la companyia *Inktomi*. El que va exposar és que hi ha tres requisits bàsics dels sistemes que existeixen en una relació especial quan es tracta de disseny i desplegament d'aplicacions en un entorn distribuït (concretament es referia a aplicacions web, però moltes empreses corporatives són multi-site/multi-country i poden tenir efectes semblants als centre de dades/LAN/WLAN.) Els tres requeriments són: **consistència de dades**, **disponibilitat del sistema**, i **tolerància de partició de la xarxa**, les inicials dels quals dóna nom a l'Acrònim CAP (*Consistency*, *Availability*, *Partition Tolerance*). Brewer estipula que només es pot aconseguir disposar dues d'aquestes tres propietats dels sistemes d'informació distribuïts en un moment donat. Per tant, el que significa és que s'ha de renunciar almenys a una propietat per tal d'obtenir les altres dos.
Aquest teorema obre la incògnita de quines són les dues propietats millors a escollir per adoptar al sistema. A la següent Figura 3 es poden veure il·lustrades totes les combinacions possibles entre les propietats CAP i quina opció han triat les diferents solucions de bases de dades distribuïdes que existeixen actualment en el mercat.

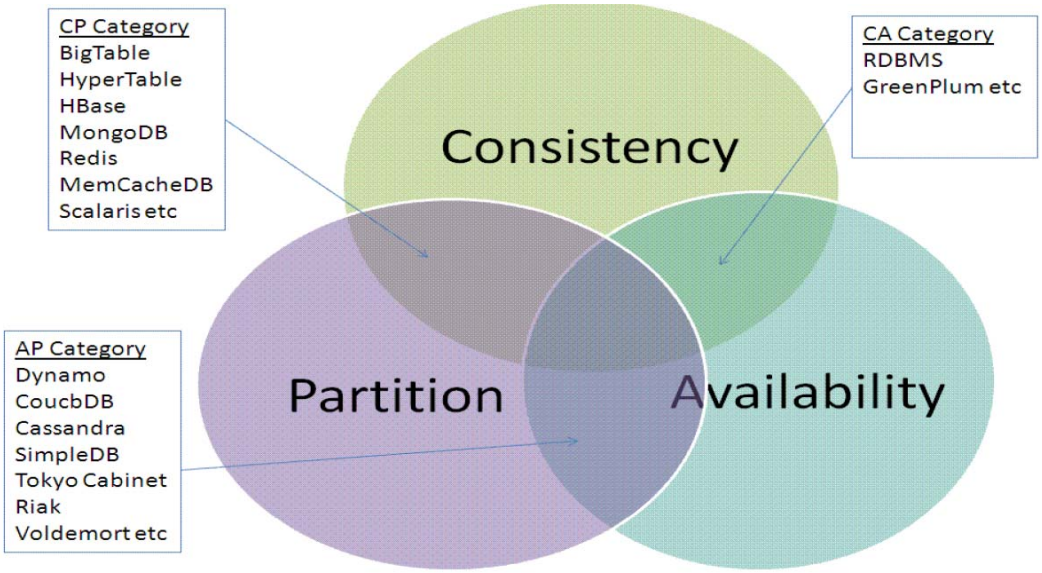

Figura 3. Esquema del Teorema de CAP

Capítol 3
Plataformes d'emmagatzematge al núvol

3.1 Cloudian

És un paquet de software (*open-source*) per proveir un núvol d'emmagatzematge. Té les següents característiques:

- Treballa amb una base de dades *NoSQL*: *Apache Cassandra*
- Compatibilitat completa amb l'API S3 d'*Amazon* (inclou codis d'error).
- *Multi-datacenter*, arquitectura *peer-to-peer*. No té un únic punt de fallida.
- Per a *clouds* públics i privats
- Administració de grups i usuaris a través de *Management Console* o REST API.

En la següent Figura 4 es mostra l'arquitectura d'una plataforma amb Cloudian:

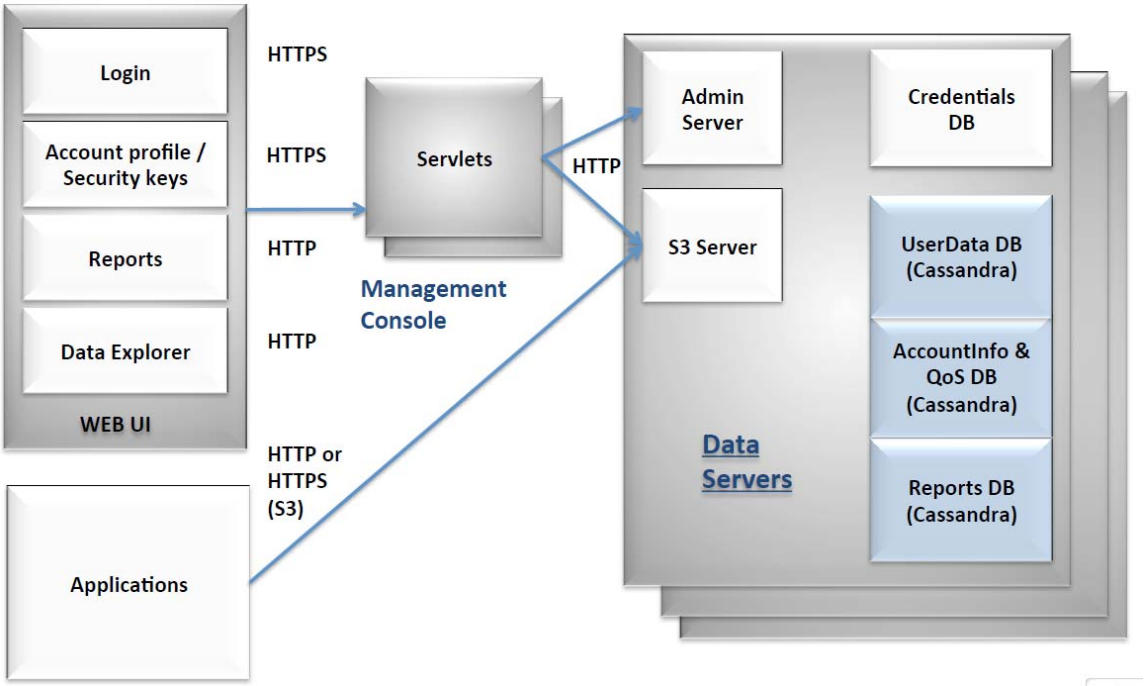

Figura 4. Arquitectura d'una plataforma amb Cloudian

3.2 OpenStack Storage (Swift)

Object Storage [9] és un programari de codi obert (*open-source*) per la creació d'un emmagatzematge d'objectes que està pensat per formar *clusters* de servidors i arribar a allotjar fins a *petabytes* d'informació. A més a més, al ser un sistema distribuït i no tenir un punt de control principal, ni un punt de fallida únic, proporciona un alt nivell d'escalabilitat i redundància.

Tot i així, aquest no és un sistema de fitxers ni un sistema d'emmagatzematge en temps real, si no més bé, un sistema a llarg termini per dades majoritàriament estàtiques i que només s'actualitzen eventualment. Així, els principals exemples de dades que millor s'hi adapten són les imatges de màquines virtuals, la fotografia digital, l'emmagatzematge de correu electrònic i còpies de seguretat.

Els seus components són:

- *Swift Proxy Server* és l'encarregat de rebre les peticions de l'exterior, del control d'accés, de la gestió de la memòria *cache* del sistema i del control de la disponibilitat de cada node d'emmagatzematge. A part d'això, té coneixement de la localització de les entitats apropiades, i així, re-adreçar les peticions dels clients al node corresponent.
- *Swift Object Server* és el responsable de gestionar l'emmagatzematge local, és a dir, recuperar i eliminar objectes del sistema d'arxius que utilitza el node. Aquests objectes són típicament fitxers binaris que contenen *metadades* com atributs estesos d'arxiu (*xattr*).
- *Swift Container Server* enumera els objectes en un contenidor i també realitza un seguiment estadístic, com el nombre d'objectes que conté i l'espai ocupat pel contenidor. Per tal de mantenir aquesta informació utilitza fitxers *SQLite*.
- *Swift Account Server* enumera els contenidors de la mateixa manera que el servidor de contenidors enumera els objectes.
- El *Ring* és l'anell lògic que conté la informació referent a la ubicació física dels objectes emmagatzemats a l'interior de *Swift*. El seu funcionament és similar a un servei d'indexació que utilitza diversos processos per cercar i localitzar la ubicació física real de les entitats (comptes, contenidors i objectes) dins del *cluster*.

En la següent Figura 5 es mostra , un exemple, de com es distribueixen a nivell físic i lògic tots els components de *Swift* en una implementació amb cinc nodes.

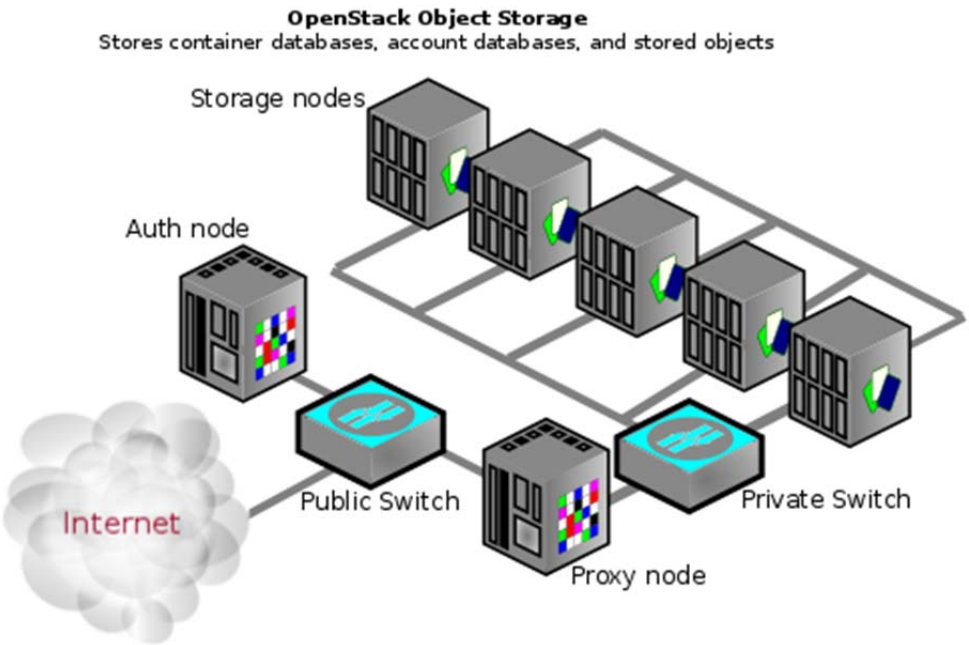

Figura 5. Esquema de distribució del components d'OpenStack Swift

3.3 Riak

Riak [10] és una base de dades *NoSQL open-source*, basada en el principis *Amazon's Dynamo* desenvolupada sota el llenguatge *Erlang*.
Riak té la capacitat de distribuir dades entre els diferents nodes fent servir *hashing* consistent en un esquema clau/valor senzill en espais de noms anomenats *buckets*.

3.3.1 Riak CS

Riak CS (*Cloud Storage*) és un software *open-source* d'emmagatzematge sobre una base de dades Riak. Pot ser utilitzat per a construir *clouds* públics i privats d'emmagatzematge. Les principals característiques d'aquest software són:

- Alta disponibilitat.
- API compatible amb *S3* i autenticació.
- Presentació d'informes per usuari.
- Estadístiques per a monitorització i mètriques.
- Integració amb *OpenStack*.

La següent Figura 6 mostra l'esquema de com s'emmagatzema un objecte en un sistema amb Riak CS:

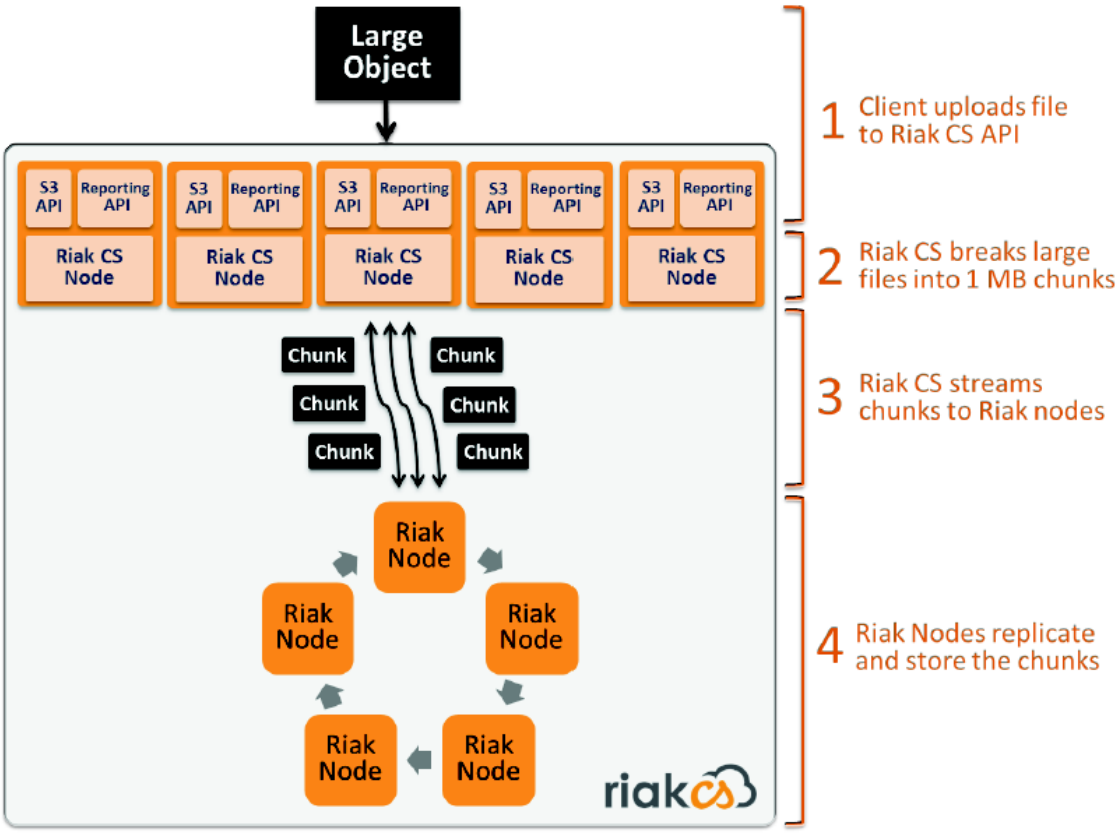

Figura 6. Esquema d'emmagatzematge de RiakCS

3.4 Comparativa Riak amb Cassandra

Seguidament, es presenta una breu comparació tècnica entre Riak i Cassandra, a grans trets:

- Tant Riak com Cassandra són bases de dades *NoSQL* amb llicència Apache 2.0 basades en el *Amazon's Dynamo paper*.
- Riak és més fidel a l'aplicació de *Dynamo*, amb la incorporació d'algunes noves funcionalitats com poder ser: *MapReduce*, *Indexes*, *Full-text Search*. Cassandra s'allunyà més de *Dynamo*, eliminant els rellotges vectorials i no fer servir *consistent hashing*, alhora que afegeix algunes funcionalitats com poden ser: *partitioners* i les consultes de rang.
- Riak està escrit principalment en *Erlang* amb algunes línies en C. Cassandra està escrit en Java.

La següent taula (veure Taula 1), mostra una comparació d'alt nivell entre les característiques i capacitats de Riak i Cassandra.

	Riak	**Cassandra**
Model de dades	Riak emmagatzema parells clau/valor en un espai de noms de nivell superior anomenats *buckets*.	Cassandra s'assembla al emmagatzematge de columnes i es compon de *Keyspaces*, *Column Families* i altres paràmetres.
Model d'emmagatzematge	Riak compta amb un sistema modular, extensible local d'emmagatzematge per adaptar-se al seu cas d'ús.	Cassandra s'inicia amb una escriptura a un registre de confirmació seguit d'una escriptura posterior a una estructura en memòria anomenada *memtable*. *Writes* llavors es processen per lots a una estructura de la taula persistent cridat una taula de cadenes ordenades (SST)
Accés a les dades i APIs	Riak ofereix dues interfícies principals (a més d'accés *Erlang*): HTTP i *Protocol Buffers*.	Cassandra proporciona diversos mètodes d'accés, incloent una API Thrift, CGL (*Cassandra Query Language*) i CLI.
Tipus de consultes	Actualment hi ha quatre formes de consulta de dades en Riak: Operacions de clau principal (GET, PUT, DELETE, UPDATE), *MapReduce*, Utilitzar índexs secundaris i fent ús de cerques.	Cassandra ofereix diverses formes de consulta de dades: *Keyspaces*, Operacions *Column Family*, CGL, Índexs secundaris i suport per Hadoop.
Consistència de les dades	Riak utilitza una estructura de dades anomenada rellotges vectorials. Els rellotges vectorials es poden configurar per emmagatzemar còpies d'una dada determinada, basant-se en la mida i el temps d'aquesta dada. També hi ha una opció per desactivar els rellotges vectorials i recórrer als segells de temps basada en l' *"última-escriptura-guanya"*.	Cassandra utilitza marques de temps (*timestamps*) a nivell de *Column Family* per determinar el valor més recent en fer les peticions de lectura. No hi ha manera integrada per fer el control de versions de les dades.

Concurrència	En Riak, qualsevol node del *clúster* pot coordinar una operació de lectura/escriptura per a qualsevol altre node.	Tots els nodes de Cassandra són iguals. Una sol·licitud de lectura o escriptura d'un client pot anar a qualsevol node del clúster. Quan un client es connecta a un node i envia una petició de lectura o escriptura, aquest node s'exerceix com a coordinador.
Replicació	Riak fa servir *hashing* consistent per replicar i distribuir N còpies de cada valor al voltant d'un *clúster* de Riak format per qualsevol nombre de màquines físiques	Replicació en Cassandra comença quan un usuari tria una partició. *Partitioners* inclouen *Random partitioner* (que també es basa en *hashing* consistent per a emmagatzematge de dades) i diverses opcions de particion ordenades.
Escalabilitat	Riak permet augmentar i disminuir el clúster uniformement mentre es manté l'equilibri de la càrrega en cada màquina de manera elàstica. En altres paraules, tots els nodes són iguals no hi ha un node màster (*masterless*). Quan s'afegeix una màquina física a Riak, el clúster es fa conscient de la seva pertinença a través del estat de l'anell. Un cop sigui membre de l'anell, se li assigna un percentatge igual de les particions i, posteriorment, s'apropia de les dades que pertanyen a aquestes particions.	Cassandra permet afegir nous nodes dinàmicament amb l'excepció de calcular manualment el testimoni d'un node (encara que els usuaris poden optar per deixar que Cassandra calculi això).
Replicació *Multi-Datacenter*	Riak Enterprise versió comercial de Riak CS permet la replicació en N centres de dades.	Cassandra té la capacitat de propagar-se a través de múltiples nodes de centres de dades a través de diversos paràmetres de configuració.
Monitorització i	Riak Control, és una consola	*DataStax OpsCenter*, una

Administració	gràfica de codi obert per supervisar i gestionar *clústers* Riak.	interfície gràfica d'usuari per al seguiment i l'administració de *clústers* Cassandra. Això inclou una versió gratuïta per al seu ús en producció, així com una versió de pagament amb característiques addicionals.

Taula 1. Comparativa Riak amb Cassandra

3.5 Comparativa Riak CS amb Swift

Riak CS i *Swift* el mòdul d'emmagatzematge d'objectes de *OpenStack* són dos sistemes d'emmagatzematge en el núvol que tenen molts detalls de disseny i implementació en comú. En la següent Taula 2, s'enumeren les diferencies més importants entre els dos sistemes a nivell de característiques i capacitats.

	Riak CS	Swift
Anti-Entropy	Riak CS és compatible amb *Active Anti-Entropy*, que controla i repara les incoherències entre les rèpliques divergents. Swift no porta a terme la reparació en el temps de lectura o escriptura, sinó que resol aquestes qüestions durant el seu proper cicle de *rsync*.	Swift invoca freqüentment la comanda "*rsync*" per a la reparació de qualsevol inconsistència entre els sistemes d'arxius dels nodes de dades.
Temps d'escriptura i Fallida de hosts	Riak CS sempre escriu un nombre N de còpies desitjades, en cas de fallida d'algun host, s'escriurà temporalment en un altre node, fins que aquest sigui reparat, un cop reparat traspassarà les dades al node.	Swift controla que almenys hi hagi un "quòrum" de còpies o rèpliques abans de declarar l'èxit d'una escriptura.
Model de "*quòrum*"	No és només sobre la disponibilitat, sinó que també proporciona un efecte de latència responent a l'usuari sense la necessitat de bloquejar al host més lent del *clúster*. Això evita que problemes breus de	Swift, tot i respondre només amb la "*millor*" resposta, esperarà que tots els nodes de dades pertinents per acabar abans d'enviar una resposta a una sol·licitud d'escriptura. Això pot afectar adversament

	rendiment en cada host afecti als usuaris finals.	la latència d'impacte. No obstant això, les sol·licituds de lectura de Swift no esperen que el *quòrum*; ells simplement tracten una rèplica en un moment de forma aleatòria fins que aconsegueixen una resposta amb un temps d'espera bastant curt abans de passar a provar amb un altre. Hi ha plans per millorar la latència de les peticions d'escriptura de Swift.
Integració	Riak CS només és un servei d'emmagatzematge que no té serveis relacionats específics per a computació, gestionar d'imatges de màquines virtuals, etc...	Tot i que pot funcionar per si mateix, Swift és part del projecte OpenStack, ofereix altres mòduls de serveis per gestionar màquines virtuals.
Llenguatge	Riak CS està escrit en Erlang, un idioma i una plataforma d'enginyeria de molt alta disponibilitat.	Swift està escrit en Python, un llenguatge amb una comunitat de desenvolupadors molt gran.
Instal·lació	Riak CS està dissenyat per a una fàcil instal·lació, amb un nombre relativament petit de components independents. Una instal·lació mínima requereix la instal·lació de només tres components i l'edició de menys de 10 línies de configuració.	Swift requereix la instal·lació i supervisió contínua de diversos components, entre ells *Memcached*, *SQLite*, i *Keystone* (el servidor d'autenticació de *OpenStack*), cadascun dels quals té força dependències.
Operacions	Riak CS, amb una sola comanda d'administració permet aprovisionar nous nodes. Els components del sistema subjacents garanteixen un comportament correcte durant les transicions.	Swift requereix un alt grau de gestió manual. Quan aquests arxius cauen fora de la sincronització, el sistema experimentarà un comportament estrany o deixar de funcionar del tot.
Suport API S3	Riak CS suporta directament i	Swift té la seva pròpia API (no

Amazon	de forma nativa l'API S3, incloent aspectes tan comunament utilitzats com *ACLs S3*, traducció-*hostname-to bucket*, etc.	S3) amb les seves pròpies comandes. Opcionalment disposa d'un *middleware*, desenvolupat externament que emula l'API de S3 a la part superior de Swift.
Llicència	Riak CS és de codi obert i és administrat per *Basho*. Està disponible sota la llicència *Apache 2*.	Swift és completament de codi obert i està gestionat per la Fundació *OpenStack*. No es requereix llicència de cap manera i cap companyia pot bloquejar o causar cap canvi en ell pel seu compte.

Taula 2. Comparativa Riak CS amb Swift

3.6 Casos d'ús

Els principals casos d'ús per utilitzar una plataforma d'emmagatzematge orientada a objectes són:

- Emmagatzematge d'imatges, textos, documents, vídeos i altres objectes de grans dimensions.
- L'emmagatzematge intern proporcionat sota demanda a diferents unitats de negoci, similar a *Amazon S3*.
- Una capa d'emmagatzematge per a núvols públics i serveis en el núvol.
- Infraestructura per desplegar aplicacions fora d'*Amazon S3*, o solucions híbrides.
- Emmagatzematge redundant per còpies de seguretat i escenaris de *disaster recovery*.
- Emmagatzematge de *logs*, rotació de *logs* a *storage* basat en objectes.

Capítol 4
Especificació Prova de Concepte (POC)

4.1 Elecció de la plataforma d'emmagatzematge

Per a la elecció de la plataforma d'emmagatzematge posarem un pes a cadascuna de les característiques i una puntuació sobre 10 de com compleixen cadascuna de les plataformes aquesta. Comencem per els pesos segons les nostres necessitats de cadascuna de les següents característiques:

Model dades (MD): 4, Model d'emmagatzematge (ME): 5, Instal·lació (INS): 6, API S3 (API): 3, Replicació (REP): 4, Llicència (LLIC): 2, Consistència (CONS): 4, Escalabilitat (ESC): 5

	MD	ME	Instal·lació	API	Replicació	LLIC	Consistència	Escalabilitat	
Pes	4	5	6	3	4	2	4	5	
Cloudian	6	6	7	10	7	9	8	7	
Swift	5	6	4	6	7	9	7	6	
Riak CS	8	7	9	8	7	10	8	8	
Puntuacions x Pes									**Totals:**
Cloudian	24	30	42	30	28	18	32	35	239
Swift	20	30	24	18	28	18	28	30	196
Riak CS	32	35	54	24	28	20	32	40	265

Taula 3. Comparativa plataformes d'emmagatzematge

Així doncs, veiem a la següent Taula 3, que la plataforma més ben valorada segons les nostres necessitats és Riak CS seguit de Cloudian i OpenStack Swift. La plataforma sobre la que desplegarem la prova de concepte és Riak CS.

4.2 Descripció de la prova de concepte

La prova de concepte (POC) que es realitzarà consisteix fonamentalment en implementar una plataforma d'emmagatzematge al núvol amb programari *open-source* compatible amb la API S3 de Amazon. Un cop analitzades tres solucions *open-source* disponibles actualment, s'ha decidit apostar per Riak CS com a programari per desenvolupar la POC.
Llavors, s'implementarà un petit *clúster* amb dos o tres nodes Riak, i un node que farà de *proxy* des d'on s'oferirà el servei. Posteriorment, es realitzarà un seguit de proves per avaluar les virtuts i defectes de la plataforma i finalment si es possible es realitzarà un cas d'ús real amb un client que testejarà amb més profunditat la plataforma desenvolupada.

4.3 Diagrama de l'arquitectura

Per a començar, veiem en la Figura 7, el diagrama de l'arquitectura general del servei, en aquesta figura tant sols es veuen els components a nivell de xarxa.

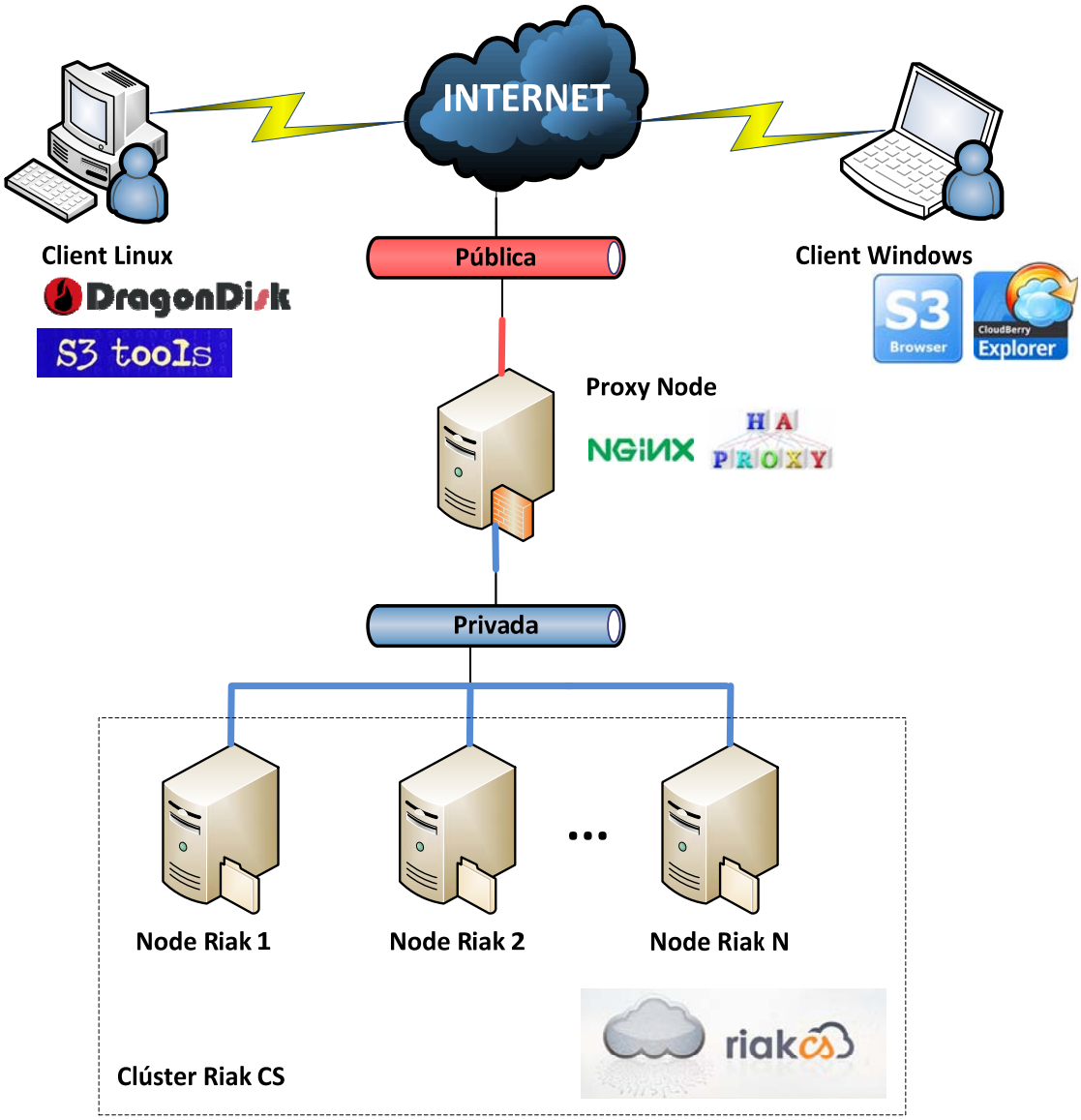

Figura 7. Diagrama de l'arquitectura de la prova de concepte

El nostre servei d'emmagatzematge, volem que estigui distribuït en vàries màquines físiques, tot i que ara mateix per realitzar les proves estaran en màquines virtuals dins d'una plataforma *IaaS*, concretament amb OpenNebula.

La gestió de la infraestructura a través d'interfície web es farà per defecte a través del component *Riak Control*, i aquest farà de *proxy* als nodes de dades Riak.

Anem a veure cadascun dels components del diagrama:

- Proxy (*Nginx*, *HAproxy*): Es recomana fer ús d'un *proxy* per balancejar i equilibra la càrrega entre els diferents nodes. Ja siguin solucions amb maquinari o programari, en aquest projecte farem servir un programari de codi obert com *HAProxy* o *Nginx*.
- Nodes: Els nodes virtuals de Riak són on s'emmagatzemen les nostres dades, es poden afegir/eliminar nous nodes en funció de la capacitat d'espai que necessitem.
- Clients (Linux, Windows): El clients seran els usuaris que faran ús del servei a través d'un client de S3, ja sigui amb un sistema operatiu Linux (*DragonDisk*, *s3cmd*) o Windows (*S3 Browser* i *CloudBerry Explorer*).

Com podem veure al diagrama l'arquitectura està formada per dues xarxes: pública i privada.

- Xarxa pública: La xarxa pública és per on s'oferirà el servei i l'accés mitjançant la API de S3 disponibles, als usuaris, més endavant s'especifiquen quins són i els ports en els que estaran escoltant.
- Xarxa privada: La xarxa privada s'utilitzarà per comunicar els diferents nodes Riak i distribuir les dades entre ells. Es recomana no exposar mai els nodes Riak directament a les interfícies de xarxa pública.

Capítol 5
Riak /Riak CS

En aquest capítol, un cop seleccionada la plataforma d'emmagatzematge per al desplegament del servei de *Cloud Storage*, anem a fer un estudi en profunditat de les funcionalitats i característiques que ens proporciona aquesta.

5.1 Què és Riak?

En poques paraules, podem entendre Riak com un magatzem de claus/valors. Riak és una base de dades distribuïda i escalable de codi obert i fàcilment predictible i simplifica el desenvolupament, donant als usuaris la capacitat de crear prototips de forma ràpida, provar i desplegar les seves pròpies aplicacions.

Riak està basat en una tecnologia desenvolupada originalment per *Basho Technologies* per administrar un negoci d'automatització de Salesforce. Degut al interès existent en la tecnologia d'emmagatzematge de dades de les aplicacions, Basho va decidir construir un negoci al voltant del mateix Riak.

Riak està fortament influenciat pel *Dr. Eric Brewer* i pel Teorema CAP i *Dynamo Paper d'Amazon*. La major part de l'equip central prové d'*Akamai* que va centrar l'enfocament de Riak en la facilitat operativa i la tolerància a fallades.

L'equip que va escriure Riak també és responsable del framework Erlang REST Webmachine, així que Riak utilitza una API REST com una de les dues formes d'accedir a les dades en Riak. Les operacions d'emmagatzematge utilitzen HTTP PUTs o POST i per obtenir les dades HTTP GETs. Les operacions d'emmagatzematge s'envien a una adreça URL que per defecte és "/riak" .

A més d'HTTP, Riak també ve amb un API *Protocol Buffers* amb totes les funcions. Aquest és un protocol binari simple basat en la biblioteca de projectes de codi obert de Google del mateix nom.

5.1.1 Clustering

Conceptualment podem entendre un *clúster* Riak com un anell. L'anell és un espai d'un nombre enter de 160 bits que es divideix en particions de la mateixa mida.

Els servidors físics, que formen el clúster s'anomenen o reben el nom de "*nodes*", que comporten un cert nombre de nodes virtuals, o "*vnodes*". Cada *vnode* reclamarà una partició de l'anell. El nombre de *vnodes* actius es determina pel nombre de nodes físics en el clúster en un moment donat.

Com a regla general, cada node del *clúster* és responsable de:

1/(nombre total de nodes físics) de l'anell.

Es pot determinar el nombre de *vnodes* a cada node mitjançant el càlcul:

(nombre de particions) / (nombre de nodes)

Per exemple, un anell amb 32 particions, dins d'un *clúster* amb quatre nodes físics, tindrà aproximadament vuit *vnodes* per node. Aquesta configuració es representa en la següent Figura 8.

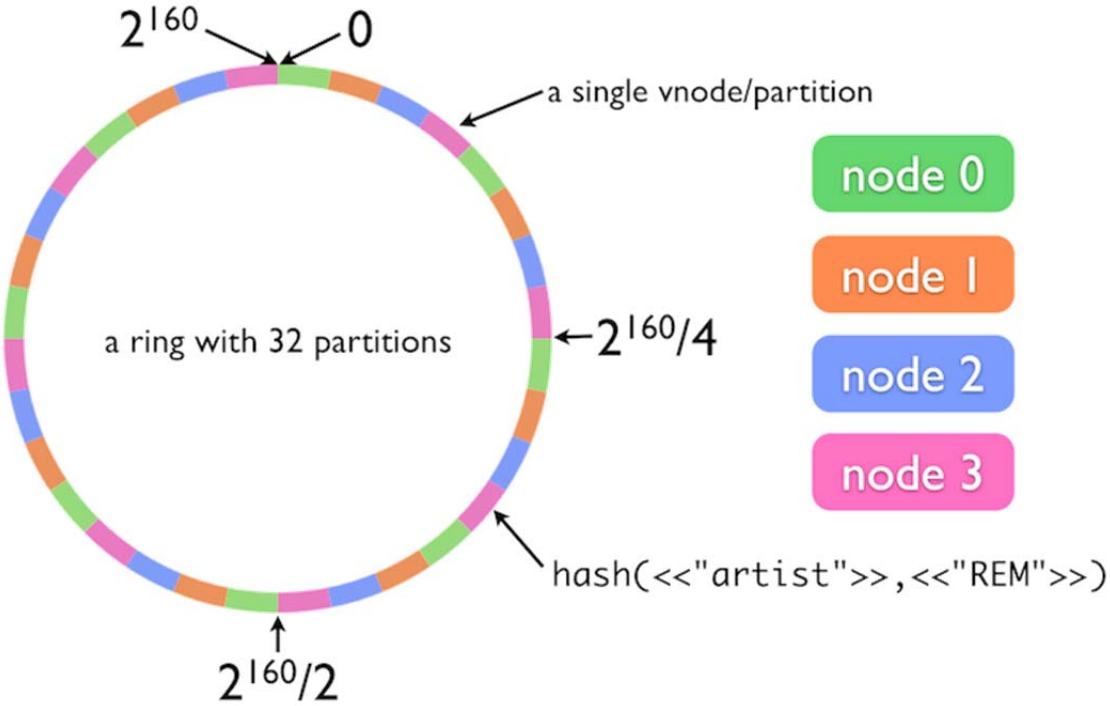

Figura 8. Representació de l'anell Riak

Els nodes es poden afegir i treure del *clúster* de forma dinàmica i Riak redistribuiran les dades en conseqüència.

Riak està dissenyat, des del principi, per a funcionar en un entorn distribuït. Les operacions bàsiques, com les lectures/escriptures de dades i execució de treballs de *MapReduce*, en realitat són més ràpides quan s'agreguen més nodes Riak a un *clúster*.

Tots els nodes d'un *clúster* Riak són iguals, no existeix un node màster. Cada node és completament capaç de servir qualsevol sol·licitud del client. Això és possible a causa de la forma en que Riak utilitza un *hash* coherent per distribuir les dades de tot el *clúster*.

5.1.2 Replicació

La replicació està integrada en el nucli de l'arquitectura de Riak. Riak controla quantes rèpliques d'una dada s'emmagatzema a través d'un paràmetre anomenat "*N value*". Aquest paràmetre té un valor predeterminat per node, però pot definir-se per a cada *bucket*. Els objectes Riak hereten el valor N del seu *bucket* pare. Tots els nodes del mateix *clúster* han d'acordar i utilitzar el mateix valor N.

Per exemple, en la Figura és fa una representació del que passa quan n_val = 3 (Aquesta és la configuració per defecte). Quan s'emmagatzema una dada en un *bucket* amb un valor N igual a tres, la dada serà replicat en tres particions separades en l'anell Riak, veure Figura 9.

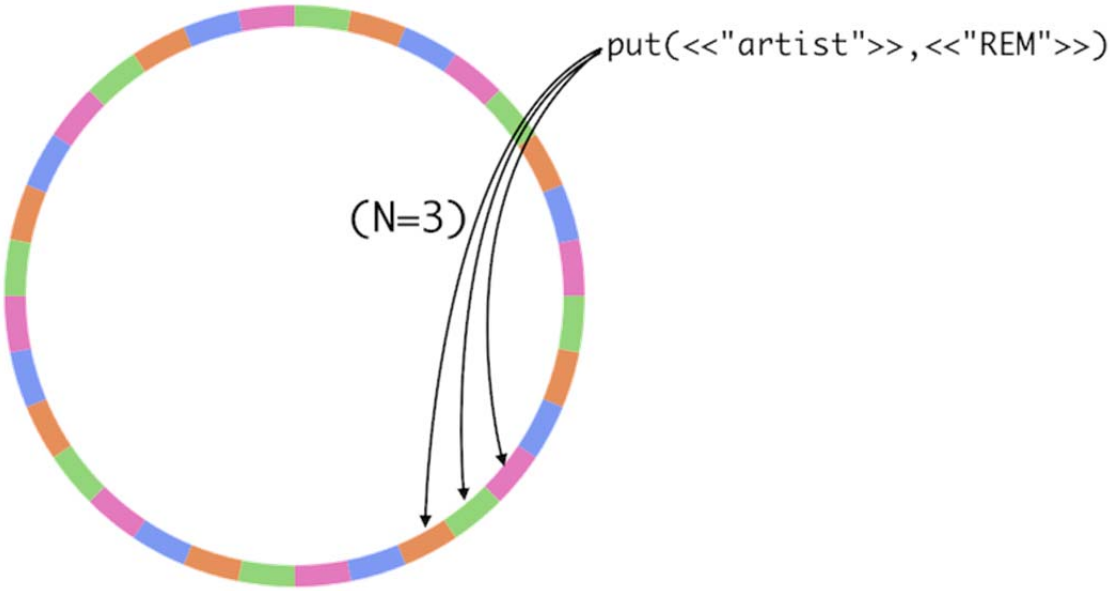

Figura 9. Exemple de la replicació en l'anell Riak

Riak utilitza una tècnica anomenada "*hinted handoff*" per compensar els nodes caiguts en un clúster. Els veïns d'un node amb errors prendran el relleu i es farà responsable les tasques del node que ha fallat permetent que el *clúster* pugui continuar funcionant, com de costum. Això pot ser considerat com una forma d'auto-sanació, com es pot observar a la Figura 10.

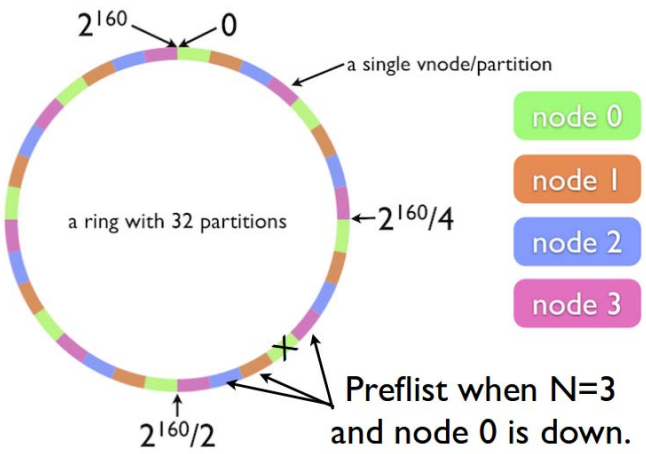

Figura 10. Exemple de sanació en l'anell Riak

5.2 Què és Riak CS?

Riak CS (*Cloud Storage*) és un programari d'emmagatzematge d'objectes fàcil d'utilitzar integrat en la capa superior de Riak, base de dades distribuïda de *Basho*. Riak CS està dissenyat per proporcionar una solució simple, amb alta disponibilitat, distribuïda d'emmagatzematge en el núvol a qualsevol escala, i es pot utilitzar per construir arquitectures de núvol, ja siguin públics o privats, o bé com infraestructura d'emmagatzematge per a aplicacions i serveis de càrrega pesada. La API de Riak CS és compatible amb Amazon S3 i dóna suport de notificació d'estadístiques d'accés i d'ús d'emmagatzematge per poder facturar als clients.

5.3 Característiques principals Riak CS

- **Compatibilitat Amazon S3-API**: Riak CS té una interfície integrada S3 amb el suport de llistes de control d'accés ACL S3, el que significa que tots dos poden utilitzar eines de S3 i marcs existents per gestionar les seves dades i també importar i extreure dades d'Amazon directament. L'API HTTP REST dóna suport al servei, als *buckets*, i les operacions a nivell d'objectes per emmagatzemar i recuperar dades fàcilment.

- **Visibilitat *Per-Tenant***: Amb l'API de Riak CS d'informes, es pot accedir a les dades i les estadístiques per usuari i d'ús a la xarxa I/O. Aquesta funcionalitat d'informes suporta casos d'ús incloent la comptabilitat, subscripció, les devolucions de càrrecs, plugins amb els sistemes de facturació, la utilització eficient de múltiples departaments, etc.

- **Suporta objectes de gran volum i tipus, amb Metadades**: Riak CS permet emmagatzemar qualsevol tipus de dades imaginables, com ara imatges, text, vídeo, documents, còpies de seguretat de bases de dades, o els binaris de programes. Riak CS pot emmagatzemar objectes en el rang de mida *terabyte* utilitzant càrregues

d'arxius de diverses parts. Riak CS també és compatible amb les capçaleres de metadades estàndard d'Amazon.

- **Replicació en múltiples centres de dades**: La versió comercial: *Riak CS Enterprise Edition* ofereix replicació de còpies de seguretat actives en múltiples centres de dades, la recuperació de desastres, i localitat de les dades. Proporcionar un emmagatzematge de baixa latència allà on sigui que els seus usuaris són i mantenir la disponibilitat, fins i tot en cas de fallada del centre de dades.

- **Integració amb OpenStack:** Riak CS es pot integrar amb OpenStack a través d'una API *Object Storage* i permet utilitzar el servei d'autenticació *Keystone*.

5.4 Arquitectura Riak CS

Riak CS es basa en Riak. Quan s'emmagatzema un objecte, Riak CS divideix l'objecte en blocs més petits (*chunk*) que es transmeten, emmagatzemen i repliquen al *clúster* Riak subjacent. Cada bloc està associat amb metadades per a la recuperació. Atès que les dades es repliquen, i altres nodes automàticament assumeixen responsabilitats dels nodes que cauen, les dades romanen disponibles fins i tot en condicions de fallida d'algun dels nodes. Tal i com es pot veure a la següent Figura 11:

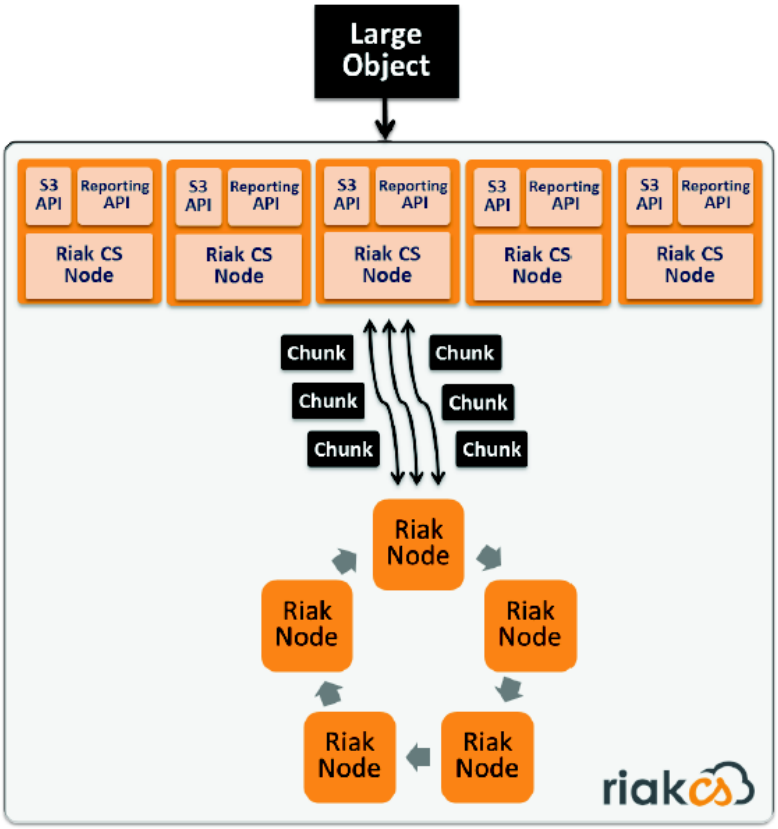

Figura 11. Esquema de l'arquitectura d'emmagatzematge de RiakCS

Capítol 6
Implementació Prova de Concepte (POC)

6.1 Arquitectura de la Prova de Concepte (POC)

La arquitectura on es desplegarà la prova de concepte, serà sobre un servei de màquines virtuals en autoservei (IaaS), concretament sobre OpenNebula [11].

Tot i que aquest no seria el millor escenari per desplegar aquest tipus de plataformes, aprofitarem la infraestructura existent i la flexibilitat alhora de crear noves màquines virtuals d'aquest servei, per tal d'estalviar costos i per tractar-se d'una prova de concepte i no d'un servei en producció .

L'escenari ideal seria desplegar la plataforma sobre servidors d'emmagatzematge d'alta densitat, veure Figura 12, que ens permeten mitjançant una capa de *software* crear potents solucions d'emmagatzematge NAS/SAN i ens permet créixer en prestacions fins a la capacitat d'emmagatzemar múltiples *Petabytes*, simplement afegint nous nodes, sense incórrer en colls d'ampolla i amb força fiabilitat.

Figura 12. Servidor d'emmagatzematge Supermicro

Existeixen diferents models amb distintes prestacions cadascun, en la següent Taula 4 es mostren diversos exemples amb capacitats d'emmagatzematge diferents:

Emmagatzematge	Badies	Capacitat Màxima	Connectivitat
2U	12x3,5" / 24x2,5"	72TB	Gigabit, 10G, 40Gb, 56Gb
3U	16x3,5"	96TB	Gigabit, 10G, 40Gb, 56Gb
4U	24x3,5" / 48x2,5"	144TB	Gigabit, 10G, 40Gb, 56Gb
4U doble densitat	36x3,5" / 72x2,5"	216TB	Gigabit, 10G, 40Gb, 56Gb
JBOD	45x3,5" / 88x2,5"	270TB	Gigabit, 10G, 40Gb, 56Gb

4U Badies de disc dobles	72x3,5" / 88x2,5"	432TB	Gigabit, 10G, 40Gb, 56Gb

Taula 4. Comparativa servidors d'emmagatzematge

A continuació es descriu breument la forma de com treballa i el funcionament d'aquesta plataforma.

6.2 OpenNebula

Va ser iniciat al 2008 per la Universitat Complutense de Madrid i adoptat posteriorment per l'empresa C12G Labs, la qual dona suport de la versió enterprise d'OpenNebula. Està més enfocat al model de virtualització de centre de dades, tot i que també ofereix APIs EC2 i OCCI per a la gestió de recursos, la qual cosa fa que també tingui orientació a aprovisionament d'infraestructura com a *cloud* públic.

Figura 13. Logotip OpenNebula

Gràcies a la integració amb la API d'OCCI, al component de zones i el suport per a crear *cloud* híbrid és de les més obertes. Al igual que OpenStack també és molt modular i gràcies a la seva arquitectura en *drivers* és fàcil modificar el seu comportament.
OpenNebula també està sota la llicència Apache v2.0. De les múltiples funcionalitats que té podríem destacar les següents:

- **Computació**: Amb OpenNebula, com amb tota plataforma de gestió IaaS, es poden gestionar recursos de computació com a màquines virtuals. També ofereix plantilles amb les quals podem tenir definides les característiques dels recursos de computació i llançar màquines a partir d'aquestes.
- **Imatges i discs**: Per a l'emmagatzemament tenim imatges de màquines, al desplegar una màquina virtual aquestes són convertides en discs per a la màquina. En cas de tenir una imatge persistent, al parar la màquina virtual s'actualitza el contingut de la imatge associada tal i com s'ha quedat apagant la màquina. També podem guardar discs de màquines en execució com a imatges.
- **Autenticació**: OpenNebula suporta varis tipus d'autenticació: usuari/contrasenya, LDAP, clau simètrica i encriptació X509. Gràcies a que el sistema d'autenticació pot ser connectat amb *drivers* desenvolupats per l'usuari podem afegir d'altres tipus d'autenticació si desenvolupem el *driver* corresponent.
- **Sunstone**: Interfície web per a la gestió i administració dels recursos, tant per l'administrador com pels diferents usuaris.
- **Xarxa**: Per a la xarxa, al igual que amb l'autenticació pot ser connectat amb *drivers*, que poden ser desenvolupats per l'usuari. Està integrat amb vàries eines de virtualització de xarxa com *brctl*, Open *vSwitch* i amb VMware.

- **APIs de gestió**: OpenNebula ofereix APIs per a la gestió de recursos dels usuaris, aquestes són EC2 i OCCI. Les APIs donen un enfoc al OpenNebula per a ser també posicionat en el model d'aprovisionament d'infraestructura. El fet d'utilitzar OCCI fa que sigui més obert a estàndards i amb més possibilitats per a la interoperabilitat i la formació de *clouds* federats.
- **Virtual Data Center**: Un Virtual Data Center és un entorn virtual aïllat, sota el control d'un administrador, el qual pot crear i gestionar els recursos IaaS i els usuaris d'aquest.
- **Zones**: El component Zones de OpenNebula ens permet l'administració centralitzada de vàries instàncies d'OpenNebula. Una mateixa zona pot ser dividida en varis Virtual Data Centers. Aquest component ens permet crear instàncies a diferents ubicacions físiques per a augmentar per exemple la disponibilitat i la recuperació enfront fallides d'un servei.
- **Accounting i quotes**: OpenNebula ens permet monitorar l'ús dels diferents recursos per part dels usuaris, gràcies a això podem adaptar-lo a eines de facturació. També podem aplicar quotes sobre els usuaris, assegurant-nos que no utilitzen més recursos dels assignats.
- **Ecosystem**: L'*ecosystem* d'OpenNebula és una pàgina web amb *plugins*, eines i extensions addicionals. D'aquests se'n poden destacar Carina, el qual ens permet desplegar, escalar i gestionar automàticament serveis en base a polítiques com la hora, la càrrega de les màquines virtuals, prioritat, esdeveniments de fallida o mètriques personalitzades. També podem trobar un component, *Onenox*, per a implementar el sistema de IPs elàstiques a través de la API d'EC2.

En la següent Figura podem observar un diagrama, amb la arquitectura del servei.

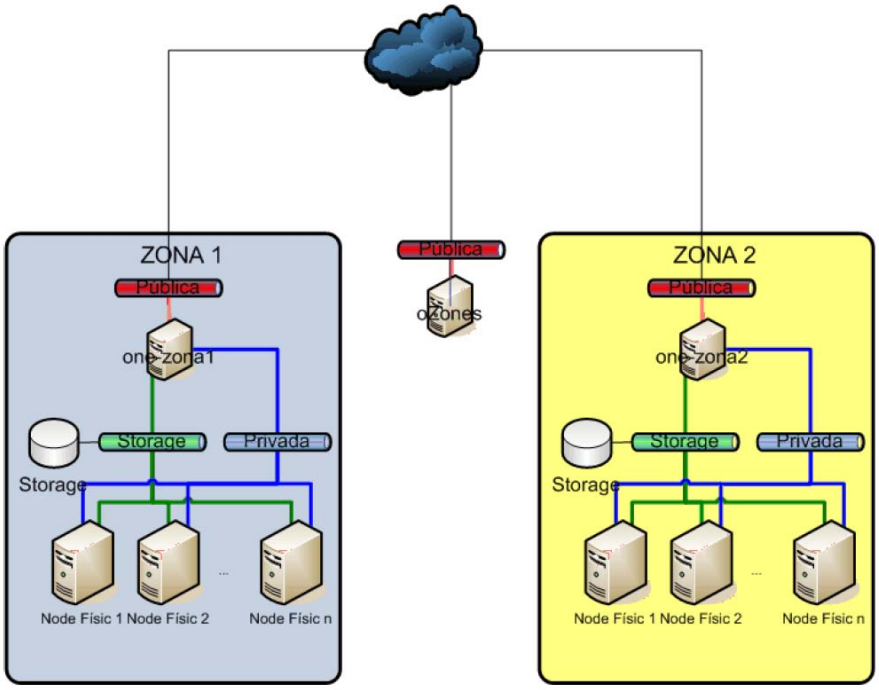

Figura 14. Arquitectura del servei cloud d'OpenNebula

6.3 Instal·lació del nodes

Per instal·lar els nodes es podia escollir entre algun dels quatre tipus de màquines virtuals existents actualment, en funció de diferents paràmetres: nombre de Virtual CPUs, memòria RAM i mida de disc, tal i com es mostra a la Taula 5.

Tipus	VCPU	RAM	Disc
H1	1	1 GB	50 GB
H2	2	2 GB	50 GB
H3	2	4 GB	50 GB
H4	4	4 GB	50 GB

Taula 5. Tipus de màquines virtuals disponibles a OpenNebula

Per implementar la prova de concepte, farem servir quatre màquines virtuals de tipus H1, amb sistema operatiu *Ubuntu Server* 12.04. D'aquestes quatre màquines, una funcionarà com a *proxy* i serà l'encarregada de oferir el servei de S3 (*endpoint*) i les altres tres seran nodes del *clúster* Riak.

En la següent Figura 15 es mostra la pantalla d'administració de l'OpenNebula, amb les quatre màquines en funcionament.

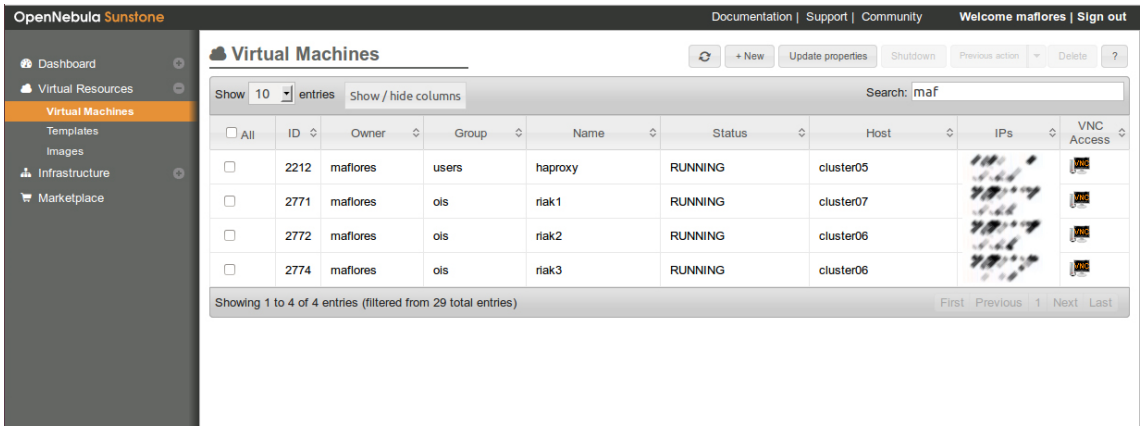

Figura 15. Aplicació Web (*Sunstone*) per administrar les màquines virtuals

En l'annex I, es descriu detalladament el procediment d'instal·lació d'un node amb Riak/Riak CS.

6.4 Diagrama amb l'arquitectura del servei

Per acabar, en el diagrama de la Figura 16, es mostra l'arquitectura de la plataforma implementada en la prova de concepte. Com es pot apreciar, es tracta de quatre màquines virtuals allotjades en la plataforma *cloud* d'OpenNebula. Tres de les quals es troben en una xarxa privada i formen *clúster* Riak CS, l'altra màquina fa la funció de *proxy* balancejant la càrrega entre els tres servidors amb un algorisme *Round Robin* i alhora exerceix de *endpoint* oferint el servei de S3.

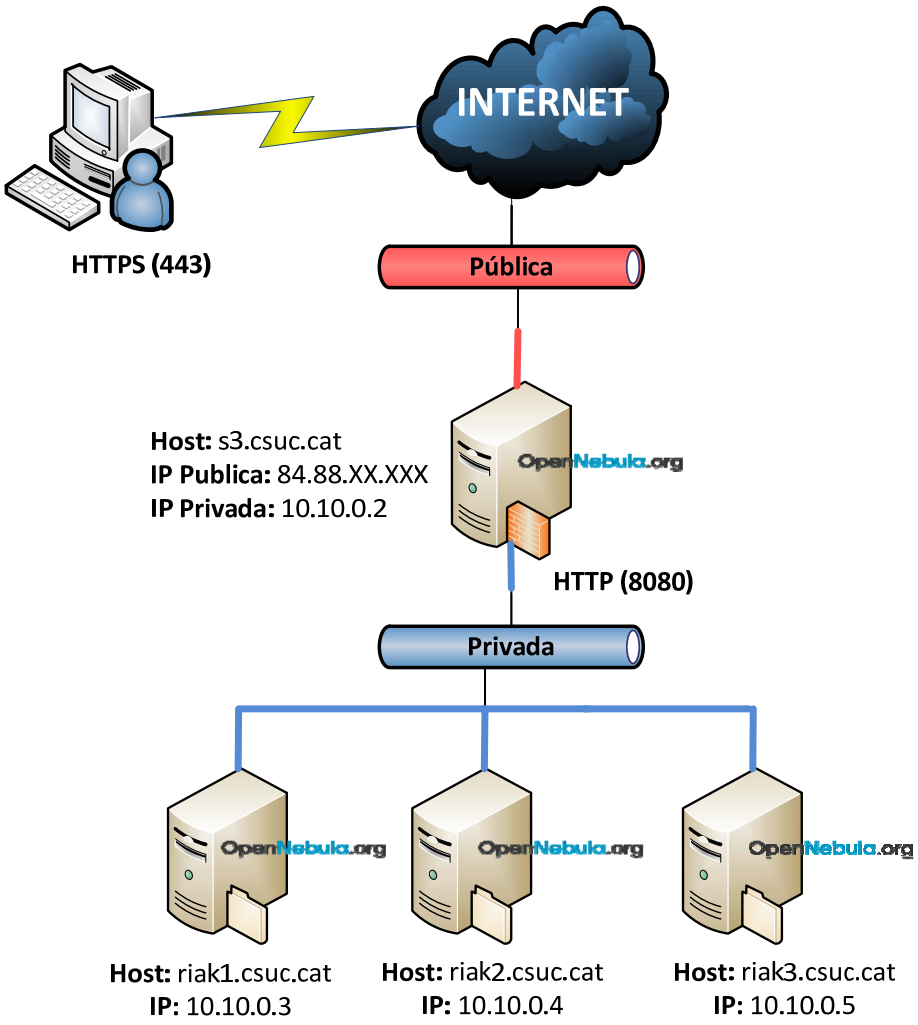

Figura 16. Diagrama amb l'arquitectura del servei

Els clients accedeixen al servei a través d'una connexió segura (HTTPS) pel port 443 de la IP pública del *endpoint*, i el *proxy* es comunica amb el nodes del clúster per la xarxa privada amb el prótocol HTTP a partir del port 8080.

Capítol 7
Proves Funcionals

7.1 Accedir a l'administració del *clúster* Riak

Riak disposa d'una consola d'administració basada en web per inspeccionar i manipular el *clúster*, aquesta s'anomena Riak Control. Per accedir només cal obrir un navegador i realitzar una petició segura (HTTPS) pel port 8069 al node que té habilitat el Riak Control: https://localhost:8069/admin

Si el navegador no pot autenticar la pàgina, això és perquè s'està utilitzant els certificats amb signatura personal. Abans d'accedir a l'aplicació caldrà autenticar-se amb el nom d'usuari i contrasenya configurats.

7.1.1 Vista *Snapshot*

La primera pestanya que veurem al iniciar el Riak Control, serà la vista d'instantànies, veure Figura 17:

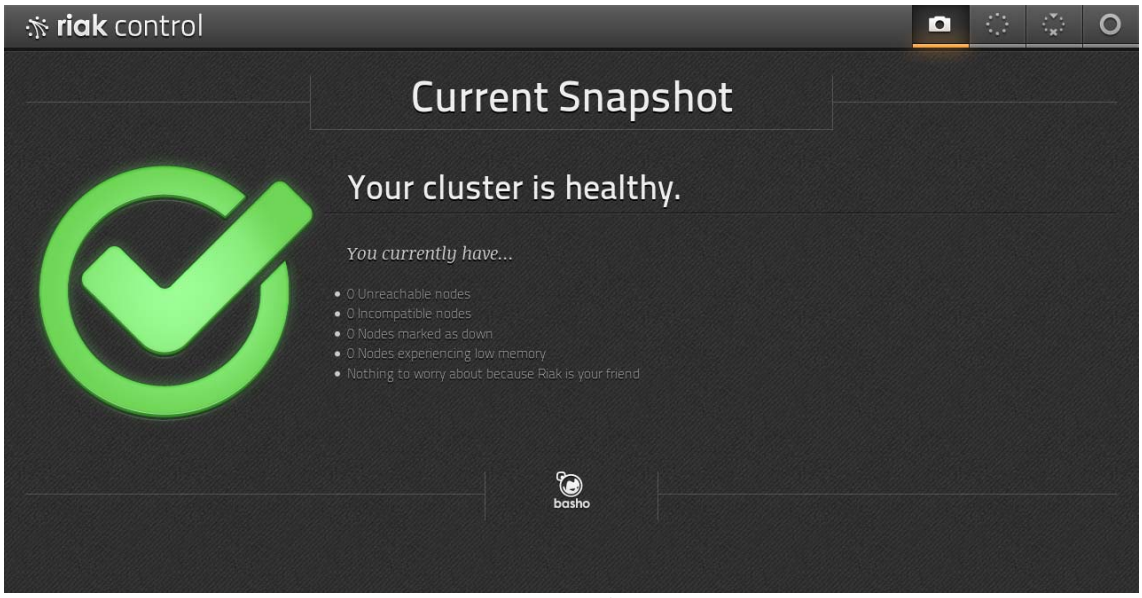

Figura 17. Vista Snapshot de l'aplicació Riak Control

Aquí podem comprovar l'estat o "salut" del nostre *clúster*. En el cas que alguna cosa no estigui del tot bé (o que hi hagi potencial de causar alguns problemes en un futur proper), la marca de verificació verda es convertirà en una X vermella, a més a més, s'acompanya amb una llista dels principals motius de preocupació. Cada element de la llista es vincula a una pàgina on es pot obtenir més informació sobre el tema.

7.1.2 Vista *Cluster Management*

A la part superior dreta del panell d'administració tenim les diferents pestanyes de navegació. Si fem clic en la segona la pestanya, accedirem a la pàgina d'administració de *clústers*, veure Figura 18.

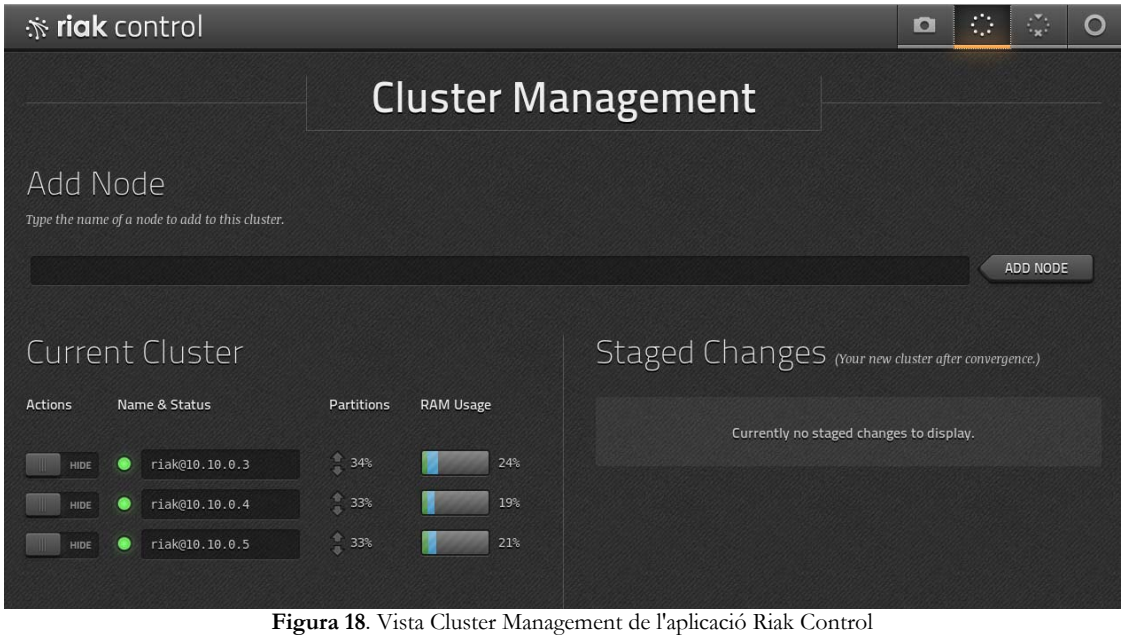

Figura 18. Vista Cluster Management de l'aplicació Riak Control

En aquesta pàgina es pot veure tots els nodes al *clúster*, juntament amb el seu estat, el percentatge de particions de l'anell, i el consum de memòria. També es pot organitzar i confirmar els canvis al *clúster*, com afegir, eliminar i marcar els nodes com a "*Down*".

7.1.3 Vista *Node Management*

En la tercera pestanya, trobem la gestió individualitzada del nodes que formen el nostre clúster, des de aquí podem aturar l'instancia d'un node o marcar-lo com a "*Down*".

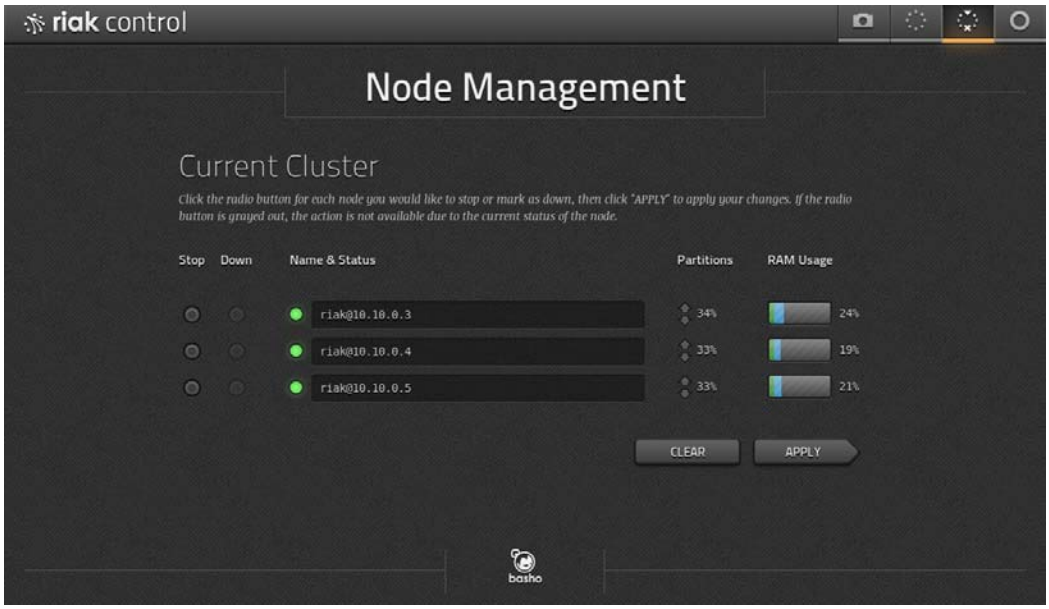

Figura 19. Vista Node Management de l'aplicació Riak Control

7.1.4 Vista *Ring*

Finalment, en la quarta pestanya, trobem un nivell més d'abstracció, amb la vista de l'anell i podem veure l'estat de cada *vnode*.

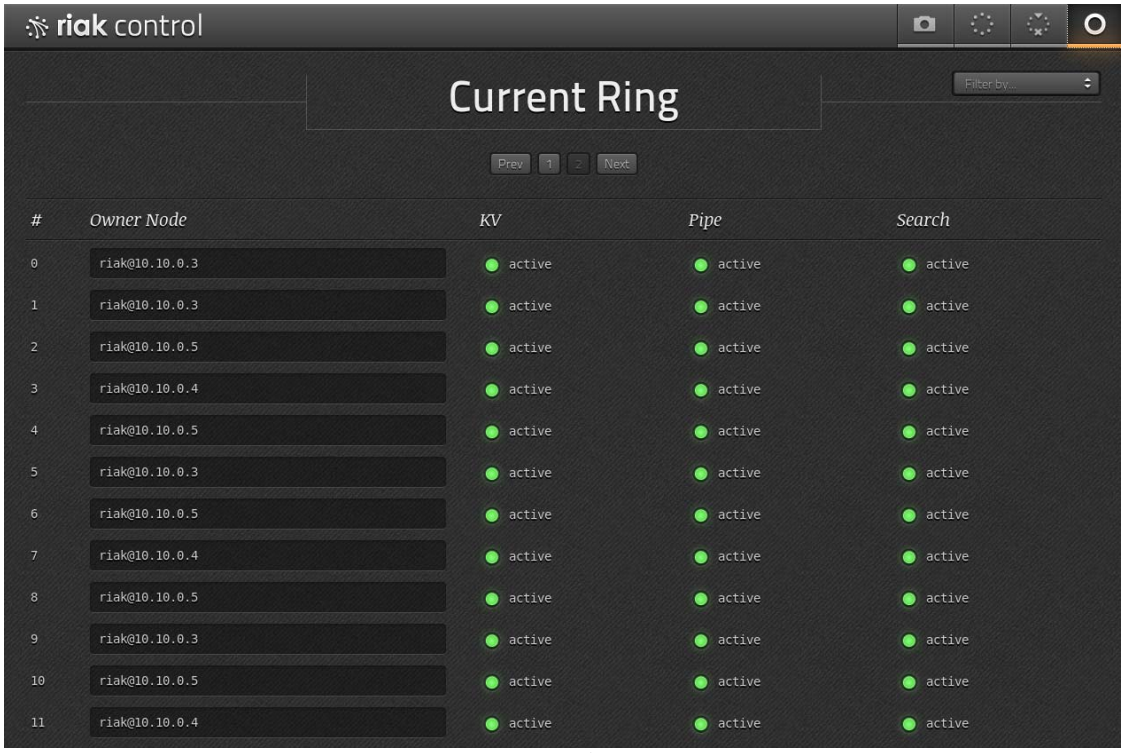

Figura 20. Vista Ring de l'aplicació Riak Control

La majoria de les vegades, l'anell serà massa gran com per gestionar de forma eficaç des d'aquesta vista. Per això, podem fer ús dels filtres que es permeten identificar fàcilment la particions inestables i les transferències en curs.

7.2 Accedir al monitoratge de la BBDD del *clúster*

El programari utilitzat per realitzar la funció de *proxy* i balancejador ha estat: HAProxy (http://haproxy.1wt.eu/), disposa d'una funcionalitat que permet consultar l'estat de la base de dades en els nodes, així com consultar un seguit d'estadístiques sobre les sessions obertes en els diferents nodes que composen el *clúster*.

D'aquesta manera podem monitoritzar l'estat de la base de dades del nostre *clúster* i veure visualment si tenim algun problema en algun del nodes, ja que l'aplicació va consultar l'estat, realitzant PINGs constants, als diferents nodes.

Per accedir al monitoratge, només cal obrir un navegador i accedir a la adreça:

http://localhost:1936

En la Figura 21, es mostra una captura de pantalla amb l'aplicació web en funcionament amb els nodes implementats a la prova de concepte.

HAProxy

Statistics Report for pid 21873

> General process information

pid = 21873 (process #1, nbproc = 1)
uptime = 8d 1h12m10s
system limits: memmax = unlimited; ulimit-n = 512017
maxsock = 512017; **maxconn** = 256000; **maxpipes** = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/11; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: UP with load-balancing disabled is reported as "NOLB".

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

haproxy_stats

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	2	-	1	1	256 000	18		5 517	318 974	0	0	0					OPEN								
Backend	0	0		0	0		0	0	25 600	0	0	5 517	318 974	0	0		0	0	0	0	8d1h UP		0	0	0		0		

riak_cs_control_https

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	6	-	0	6	256 000	15		30 394	3 051 694	0	0	1					OPEN								

riak_cs_control_backend

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
riak1	0	0	-	0	6		0	6	1024	12	12	29 806	3 050 982		0		0	0	0	0	8d1h UP	L4OK in 1ms	1	Y	-	0	0	0s	-
Backend	0	0		0	6		0	6	25 600	14	12	30 394	3 051 482	0	0		0	0	0	0	8d1h UP		1	1	0		0	0s	

riak_cs_http

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	4	-	0	5	256 000	79		238 374 313	4 533 650	0	0	9					OPEN								

riak_cs_backend

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
riak1	0	0	-	0	2		0	2	1024	24	24	12 180 778	160 795		0		0	0	0	0	8d1h UP	L7OK/200 in 3ms	1	Y	-	2	0	0s	-
riak2	0	0	-	0	1		0	2	1024	23	23	3 140 934	4 334 178		0		0	0	0	0	8d1h UP	L7OK/200 in 3ms	1	Y	-	0	0	0s	-
riak3	0	0	-	0	2		0	2	1024	23	23	223 052 601	37 293		0		0	0	0	0	8d1h UP	L7OK/200 in 3ms	1	Y	-	0	0	0s	-
Backend	0	0		0	4		0	5	25 600	70	70	238 374 313	4 532 266	0	0		0	0	0	0	8d1h UP		3	3	0		0	0s	

Figura 21. Aplicació web amb les estadístiques del *proxy*

7.3 Accedir a l'administració d'usuaris de Riak CS

Riak CS Control és una aplicació de gestió d'usuaris independent per Riak CS. Proporciona una interfície d'usuari basada en web per al filtratge, la desactivació, la creació i la gestió d'usuaris en un *clúster* de Riak CS. En la Figura 22, es mostra la pantalla amb el formulari web de Riak CS Control, per crear usuaris al *clúster*.

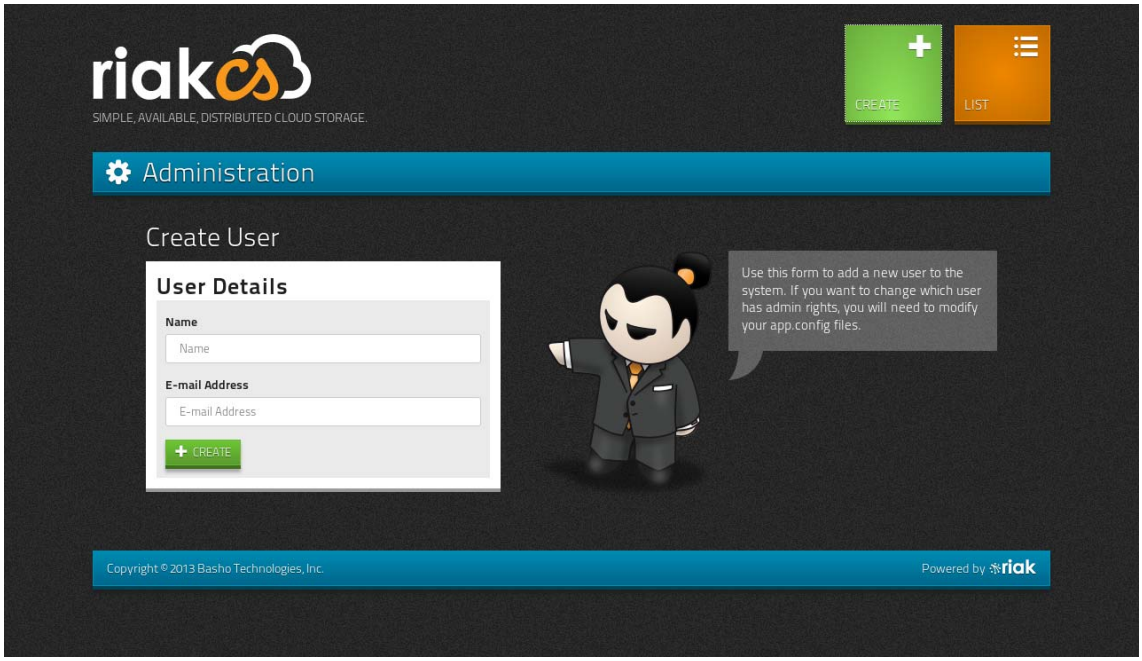

Figura 22. Vista del formulari per crear usuaris a Riak CS

En aquesta pàgina, veure Figura 23, podeu veure ràpidament tots els usuaris actuals a Riak CS juntament amb el seu estat, adreça de correu electrònic, i les credencials. Des d'aquí es pot filtrar, desactivar, crear i gestionar usuaris en un clúster de Riak CS.

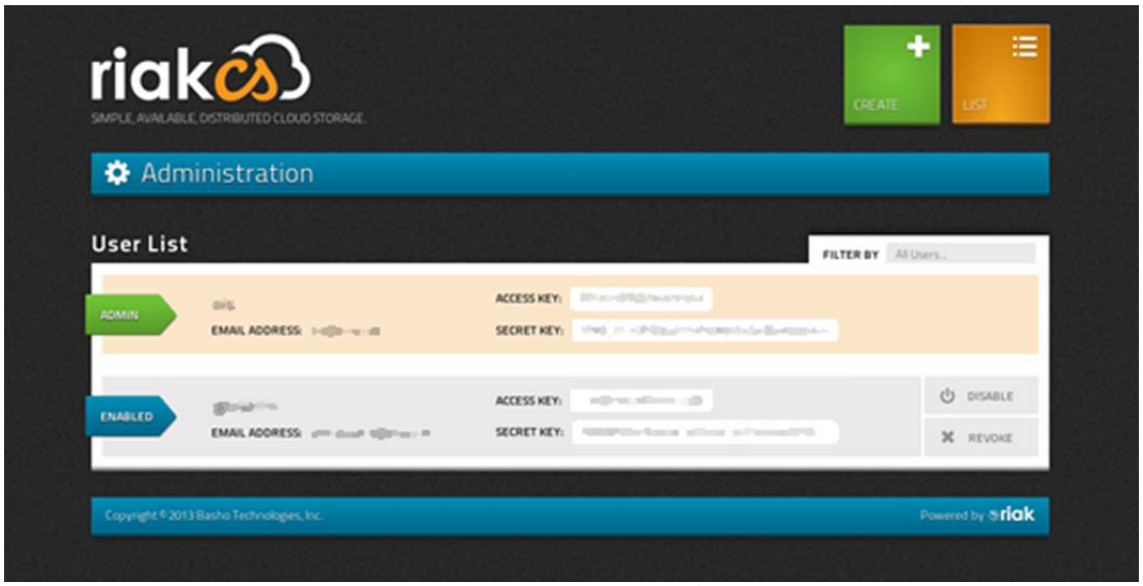

Figura 23. Captura de pantalla amb el llistat d'usuaris de Riak CS

7.3 Configurar un client S3: DragonDisk

DragonDisk és una *cross-platform*, amb un client compatible de S3 amb una interfície gràfica d'usuari disponible per a Linux, Mac OS X i Windows.

7.3.1 Creació d'un compte

Abans de poder connectar-se amb DragonDisk Riak CS, s'haurà de crear i guardar un compte. A continuació es descriu el procés per fer-ho:

- Al menú **File**, seleccionem **Accounts...**

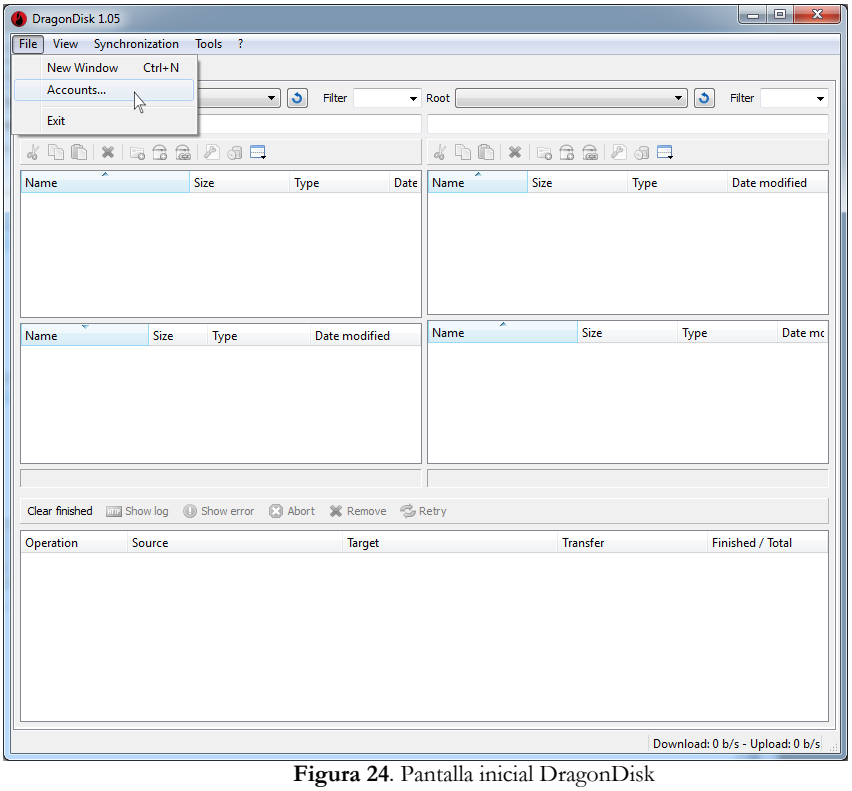

Figura 24. Pantalla inicial DragonDisk

- Fem clic sobre **New**

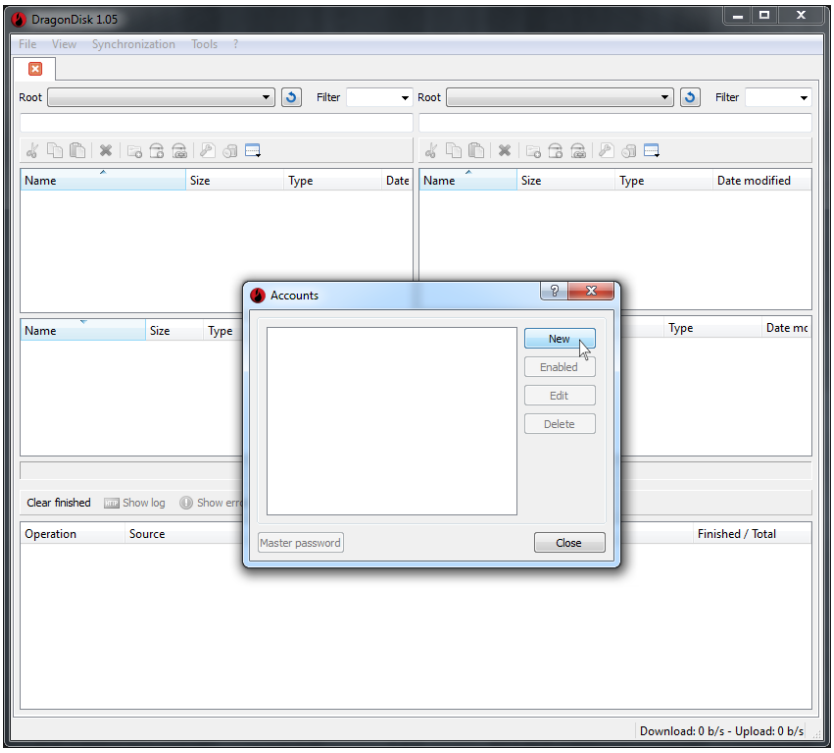

Figura 25. Creació d'un nou compte a DragonDisk

- A la finestra de diàleg del compte, seleccionem: **Other S3 compatible service** del menú desplegable de **Provider**.

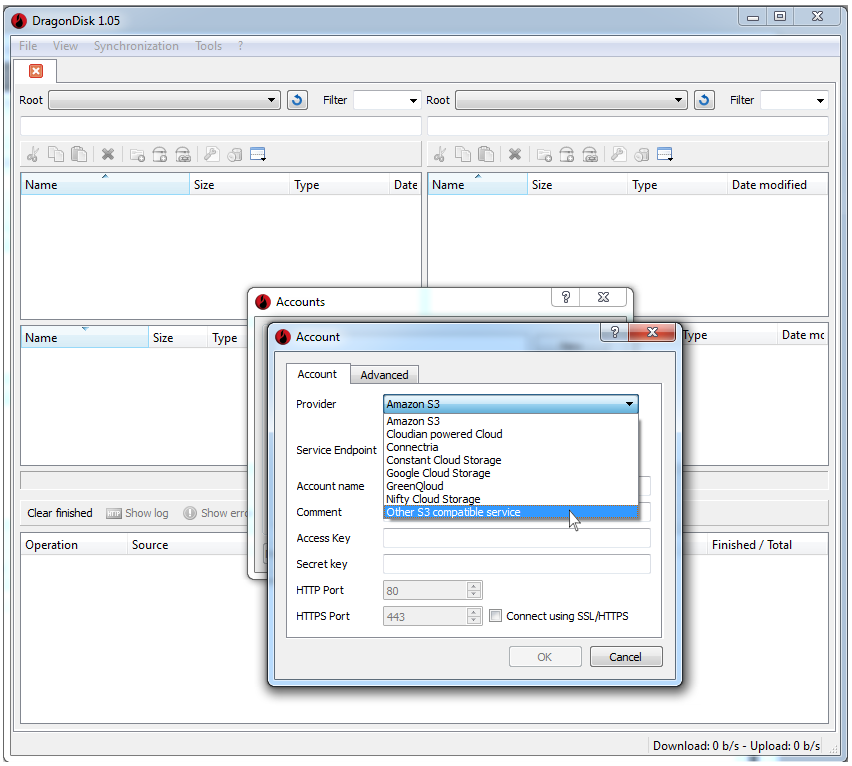

Figura 26. Configuració d'un compte amb DragonDisk I

- Introduïm el nom de la màquina o l'adreça IP de la interfície pública del *clúster* Riak CS al camp **Service Endpoint**.

- Introduïm el nostre nom d'usuari Riak CS al camp **Account name**.
- Introduïm la clau d'accés associada al nostre nom d'usuari en el camp **Access Key**.
- Introduïm la clau secreta associada al nostre nom d'usuari en el camp **Secret Key**.
- Introduïm el port de la interfície pública HTTP Riak CS al camp **HTTP Port**.
- Finalment, cal seleccionar el *tick* d'utilitzar una connexió segura: **Connect using SSL/HTTS**

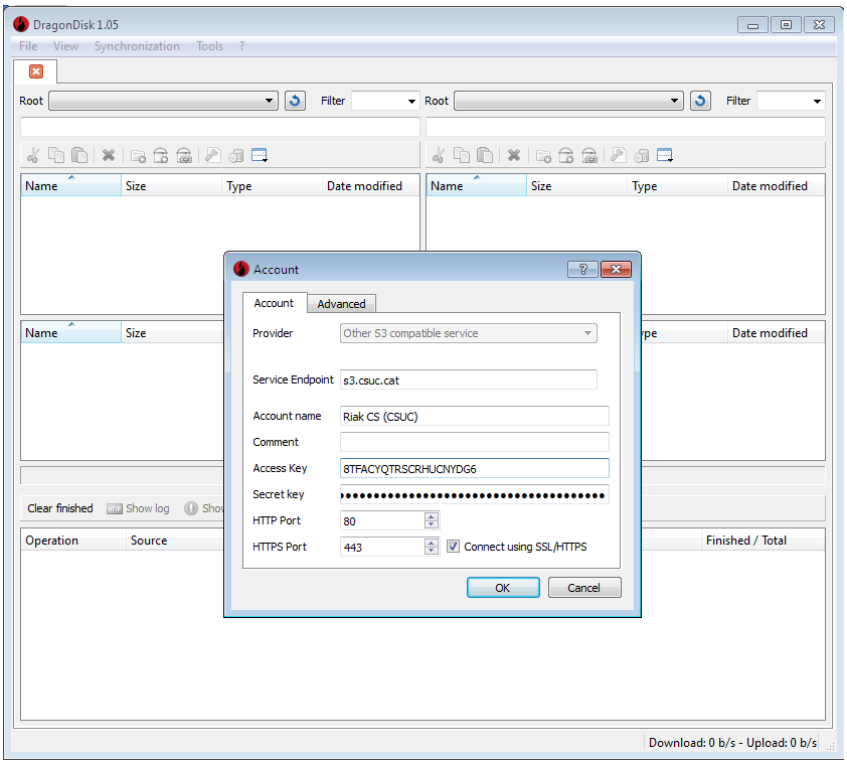

Figura 27. Configuració d'un compte amb DragonDisk II

- Premem **OK** per desar la configuració del compte.
- Fem clic al botó **Close** per completar la creació del compte i podem continuar intentant connectar-se al *clúster* Riak CS.

7.3.2 Connectar-se al clúster Riak CS

Ara anem a intentar de connectar-nos al *clúster* amb Riak CS amb la configuració del compte que acabem de definir.

- En l'arrel del menú desplegable en el panell dret de la interfície DragonDisk, seleccionem, la definició de compte de Riak CS que hem creat.
- Si no es produeix cap error i es veu la definició del compte que apareix al panell de la dreta, i després estem connectats amb Riak CS ja podem procedir a la creació d'un bucket.

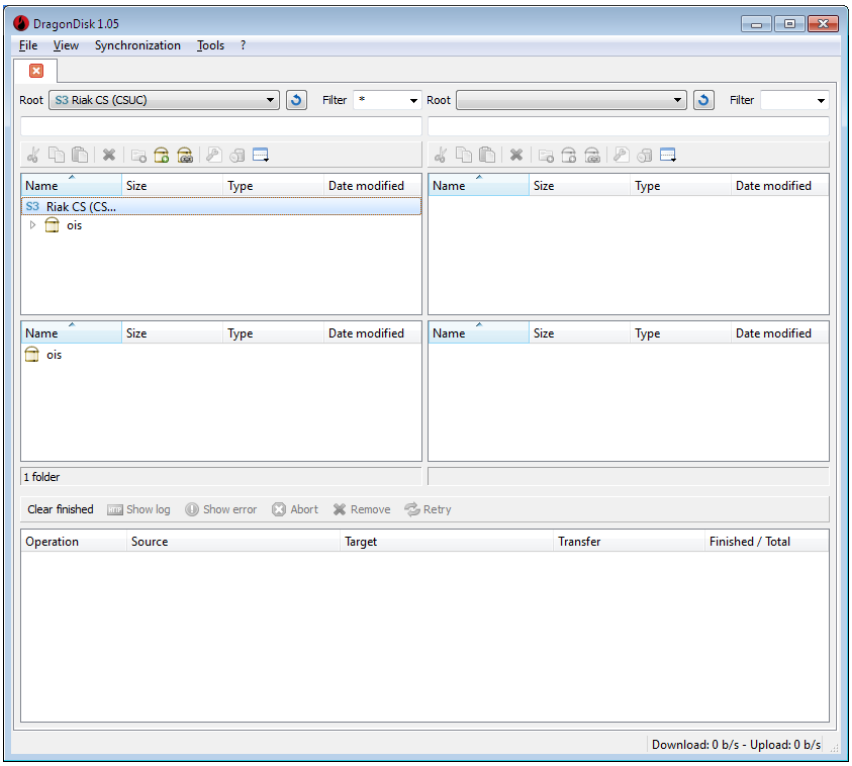

Figura 28. Pantalla amb una connexió S3 al clúster Riak CS

7.4 Configurar un client S3 per consola

Podem utilitzar *s3cmd* (http://s3tools.org) com a client de S3, per això hem de configurar l'aplicació per interactuar amb el sistema Riak CS. L'arxiu de configuració s3cmd s'anomena *.s3cfg* i resideix en el directori home de l'usuari. Els paràmetres de configuració depenen del nostre sistema de Riak CS.

Com ja s'ha comentat, *s3cmd* utilitza un fitxer de configuració *.s3cfg* que ha d'estar ubicat al directori home de l'usuari. Amb la comanda `s3cmd --configure` es llança una eina interactiva per a generar una configuració. Un arxiu de configuració també pot especificar amb l'opció `-c`. En l'annex 3 hi ha una mostra d'un arxiu *.s3cfg* arxiu que podem utilitzar per configurar *s3cmd* per interactuar amb Riak CS.

Finalment, en l'annex 4 es troba una sèrie de comandes amb l'operativa bàsica per utilitzar correctament *s3cmd*.

7.5 Operacions bàsiques amb Riak CS

A continuació es descriuen un seguit de procediments de la operativa bàsica de Riak CS:

7.5.1 Creació d'un *bucket*

- Ara que ja ens hem connectat, podem crear un *bucket* per provar l'accés a arxius de amb Riak CS.
- Fem clic a la icona *Create bucket* per obrir el quadre de diàleg de creació d'un *bucket,* tal i com es mostra a la Figura 29.

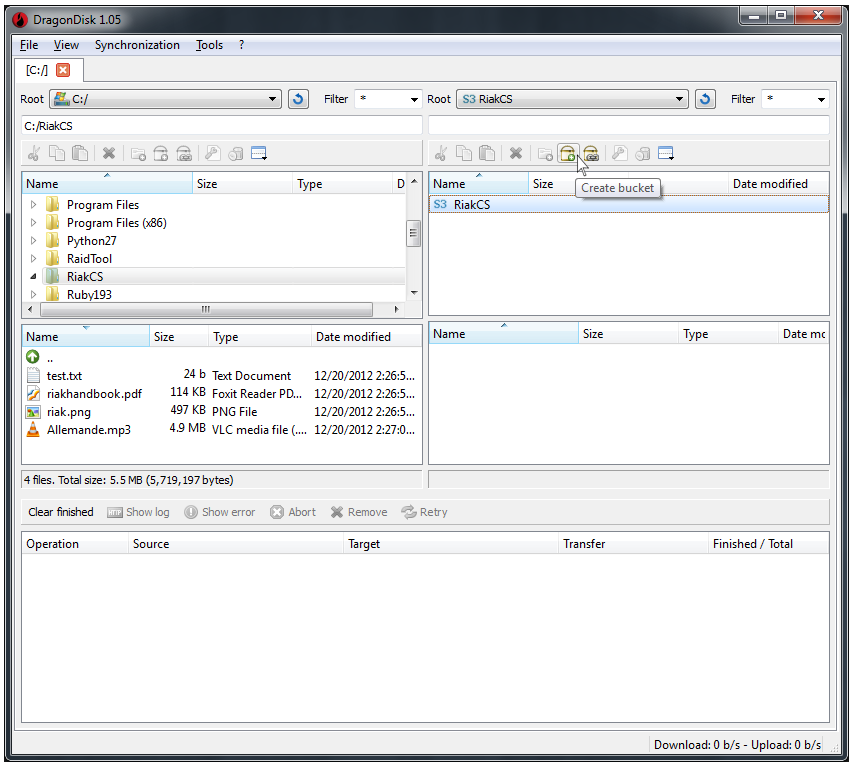

Figura 29. Vista de creació d'un *bucket* amb DragonDisk

- En el quadre de diàleg *Create bucket*, cal donar un nom al *bucket*. En aquest exemple estem anomenant el *bucket*: dragonbucket, com es pot observar a la Figura 30.
- Premem *OK* per crear el *bucket*.
- El *bucket* ha d'aparèixer ara en el panell de la dreta i ara es pot procedir amb la còpia d'alguns arxius de prova en el *bucket*.

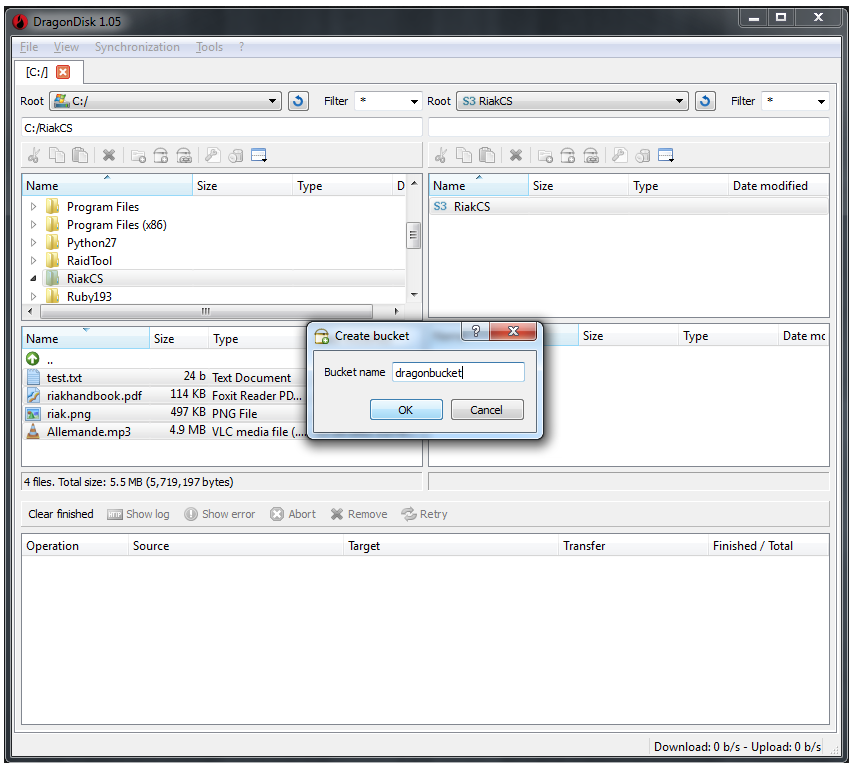

Figura 30. Quadre de diàleg per crear un *bucket* amb DragonDisk

7.5.2 Copiar fitxers en un *bucket*

Finalment, anem a copiar dades des de l'equip local, situat al panell esquerre i seleccionem l'arxiu o arxius a copiar al *bucket* que acabem de crear en Riak CS. Tal i com es mostra a la Figura 31:

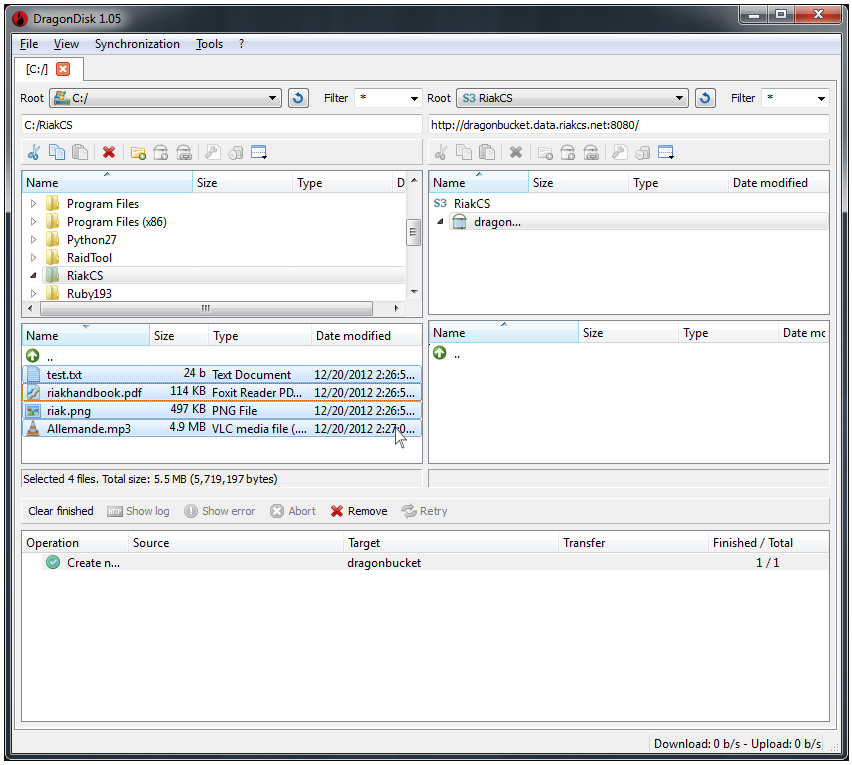

Figura 31. Transferències d'arxius amb Dragondisk I

- Després de seleccionar un arxiu o arxius, podem arrossegar al *bucket* que hem creat en el panell dret i l'operació de còpia començarà.
- Després de copiar els arxius, apareixeran al *bucket*. Veure Figura 32.
- Ara ja hem comprovat correctament la connectivitat, la creació d'un *bucket*, i les operacions de còpia d'arxius per a la instal·lació de Riak CS amb DragonDisk.

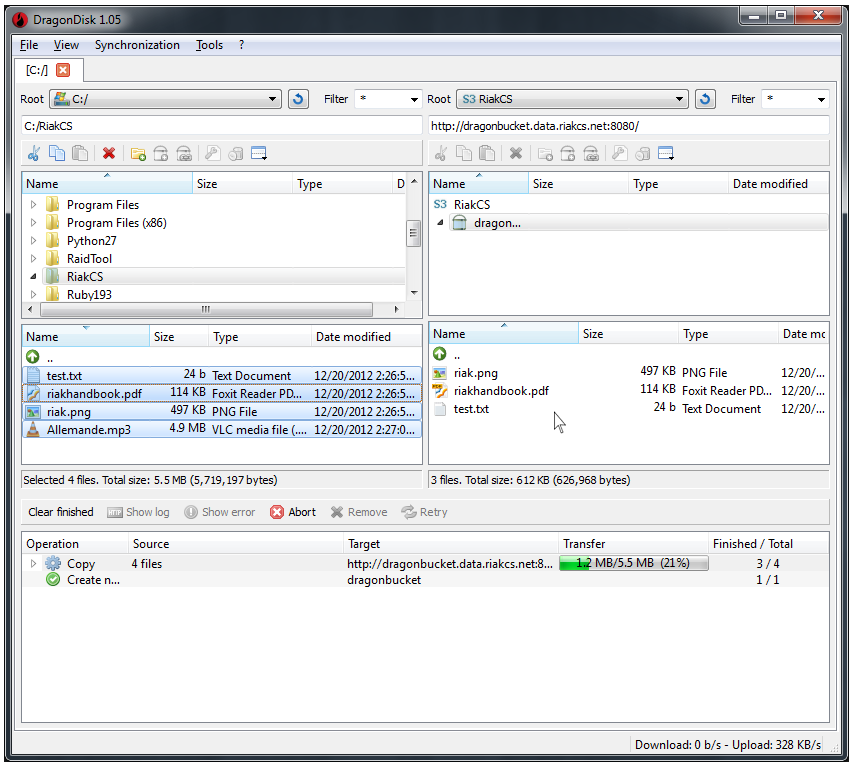

Figura 32. Transferències d'arxius amb Dragondisk II

7.6 Estadístiques amb Riak CS

Riak CS disposa d'un seguit d'estadístiques d'accés i de *storage* per fer un seguiment en funció de cada usuari, en un determinat període de temps. Per consultar d'aquestes estadístiques es realitza mitjançant el recurs o *bucket* `/riak-cs/usage/$USER_KEY_ID`. Els resultats estan disponibles, es dos formats diferents, ja sigui JSON com XML.

Per exemple, per consultar les estadístiques d'accés en format JSON haurem de proporcionar en format ISO 8601 (`YYYYMMDDThhmmssZ`) la data i el temps exacte que volem consultar:

`http://localhost:8080/riak-cs/usage/8NK4FH2SGKJJM8JIP2GU?aj&s=20120315T140000Z&e=20120315T160000Z`

En aquest cas, els següents valors sol·licitarien el període de 14:00-16:00 (GMT) del 30 de gener de 2012. El paràmetre `'a'` indica que volem rebré les estadístiques d'accés i la `'j'` indica el format en que volem el *report*, en aquest cas JSON. La sortida que obtindríem seria semblant a la que es mostra a la Figura 33.

Json Xml


```
{
  "Access": [
    {
      "Node": "riak_cs@127.0.0.1",
      "Samples": [
        {
          "StartTime": "20120315T150000Z",
          "EndTime":"20120315T152931Z",
          "KeyWrite": { "BytesIn": 32505856, "Count": 1 },
          "KeyRead": { "BytesOut": 32505856, "Count": 1 },
          "BucketRead": { "BytesOut": 3633, "Count": 5 }
        }
      ]
    },
    {
      "Errors": []
    }
  ],
  "Storage": "not_requested"
}
```


Figura 33. Estadístiques d'accés en format JSON de Riak CS

D'una forma similar obtindríem les estadístiques d'emmagatzematge:

http://localhost:8080/riak-cs/usage/8NK4FH2SGKJJM8JIP2GU?bx&s=20120315T140000Z&e=20120315T160000Z

En aquest cas, els següents valors sol·licitarien el període de 14:00-16:00 (GMT) del 30 de gener de 2012. El paràmetre `'b'` indica que volem rebré les estadístiques d'emmagatzematge i la `'x'` indica el format en el que volem el *report*, en aquest cas XML. La sortida que obtindríem seria semblant a la que es mostra a la Figura 34.

Json Xml


```
<?xml version="1.0" encoding="UTF-8"?>
  <Usage>
    <Access>not_requested</Access>
    <Storage>
      <Sample StartTime="20120316T123318Z" EndTime="20120316T123319Z">
        <Bucket name="hooray">
          <Objects>1</Objects>
          <Bytes>32505856</Bytes>
        </Bucket>
        <Bucket name="foo6">
          <Objects>4</Objects>
          <Bytes>130023424</Bytes>
        </Bucket>
      </Sample>
      <Errors/>
    </Storage>
  </Usage>
```


Figura 34. Estadístiques d'emmagatzematge en format XML de Riak CS

En l'annex 5, es mostren les principals comandes per obtenir les d'estadístiques d'accés i les estadístiques d'emmagatzematge amb el client s3cmd.

Capítol 8
Conclusions

En aquest capítol s'exposaran les conclusions respecte els objectius plantejats per aquest projecte. També es farà un anàlisi sobre com s'ha ajustat la planificació feta abans d'iniciar el projecte amb la planificació real realitzada, un estudi econòmic del mateix i finalment un comentari sobre les línies futures de treball.

8.1 Quant als objectius

Els objectius plantejats a l'inici del projecte s'ha satisfet i completat correctament:

- En primer lloc, es va realitzar una fase d'anàlisis i d'estudi de les tecnologies necessàries per duu a terme el projecte.
- S'han analitzat tres plataformes o solucions diferents amb diverses tecnologies, per determinar quina s'adaptava millor a les necessitats detectades.
- S'ha implementat una prova de concepte, amb màquines virtuals amb una de les plataformes estudiades.
- S'han realitzats un seguit de proves funcionals per comprovar el correcte funcionament de la plataforma desenvolupada.
- Finalment, s'ha documentat tot el projecte elaborant el present document, que conforma la memòria del projecte.

8.2 Estudi econòmic

La naturalesa del nostre projecte, fa una mica difícil fer una estimació del costos reals del mateix, no obstant a continuació presentem un càlculs aproximats d'aquest cost, diferenciant dues parts els costos dels recursos humans i els costos en recursos no humans.

8.2.1 Pressupost del personal

Per calcular el cost del personal s'han definit dos rols diferenciats, que són el Cap de projecte i l'Administrador de Sistemes, cadascun d'aquests rols desenvolupa diverses tasques dins del projecte.
El Cap de projecte és la persona encarregada de l'estudi i la anàlisis de la plataforma, elaborar l'especificació del projecte, detectant els requisits que ha de satisfer la plataforma i

vetllar pel compliment de la planificació de les tasques de tot el projecte, mentre que l'Administrador de Sistemes realitzarà la implementació de la plataforma a partir de l'especificació del Cap de projecte i serà l'encarregat de realitzar les proves funcionals i el manteniment posterior de la plataforma.
Respecte al nombre d'hores dedicades al projecte, s'ha pres com a referència el Pla Docent de la UOC que estableix una dedicació de 20 hores per crèdit, llavors un projecte com a aquest de 9 crèdits, tindrà una nombre total d'hores aproximat de: 20 · 9 = 180 hores. D'aquesta manera, un projecte de 9 crèdits es pot realitzar durant mig semestre a temps complert o durant un semestre a temps parcial (20 hores a la setmana), veure la següent Taula 6.

Perfil	Preu/hora	Hores	Total
Cap de projecte	45 €	40	1800 €
Administrador de Sistemes	22 €	140	3080 €
Pressupost del personal		**180**	**4880 €**

Taula 6. Pressupost del personal

8.2.2 Pressupost en materials i maquinari

Per elaborar el pressupost en maquinari, es calcularà el cost que tindria implementar la solució en màquines físiques i no en màquines virtuals com s'ha fet per la prova de concepte. En aquest sentit, una màquina 4U, com la descrita a la Taula 4, amb una capacitat neta de 115 TBs amb discos de 4 TB SATA pot tenir un cost aproximat d'uns 15.000€ amb suport i manteniment a 3 anys. Per la part de *switching* podem considerar un 10G amb equips de 24ports a un preu de màquina d'uns 10.000€.
Pel que fa a d'altres costos com poden ser electricitat, aires acondicionats de la sala i manteniments varis de la infraestructura relacionada, s'ha estimat que el cost mensual és d'uns 200 €. Posant que el manteniment de màquines es fa aproximadament cada tres anys, fem els càlculs, el cost total d'aquest apartat per aquest període és 3 · 12 · 200 € = 7.200 €

Maquinari	Valor/unitari	Unitats	Total
Dell PowerEdge 2950 (*Proxy*)	1.300 €	1	1.300 €
Supermicro 4U	15.000 €	3	45.000 €
Switching	10.000 €	1	10.000 €
Manteniment	7.200 €	1	7.200 €
Pressupost del maquinari			**63.500 €**

Taula 7. Pressupost del maquinari

8.2.3 Pressupost en llicències i programari

La característica més destacada del nostre projecte, en aquest aspecte és que el projecte es pot desenvolupar íntegrament amb programari o *software* de codi obert, per tant no cal fer cap inversió econòmica en llicències de programari o *software*.

Programari	Valor/unitari	Unitats	Total
Ubuntu Server 12.04	0 €	4	0 €
Riak	0 €	3	0 €
Riak CS	0 €	3	0 €
Pressupost del programari			**0 €**

Taula 8. Pressupost del programari

8.2.4 Pressupost total del projecte

El pressupost total del projecte, s'obté de la suma dels pressupostos parcials (personal, material i maquinari, llicències i programari) calculats anteriorment, llavors la xifra total ascendeix a: 68.380 € aproximadament.

Pressupost	Total
Pressupost del personal	**4.880 €**
Pressupost del maquinari	**63.500 €**
Pressupost del programari	**0 €**
Total	**68.380 €**

Taula 9. Pressupost total del projecte

8.3 Planificació temporal

El projecte ha estat dividit en quatre grans fases: la primera fase, (**Fase I. Anàlisis i Planificació**), que ha consistit en fer la planificació inicial i la formació necessària en les tecnologies per realitzar-lo, principalment aprendre conceptes sobre *Big Data* i bases de dades distribuïdes NoSQL. La segona fase (**Fase II. Especificació prova de concepte (POC)**) on s'han estudiats les diferents solucions *open-source* actuals, per posteriorment especificar una solució amb una d'elles, aquesta fase ha resultat ser la més llarga en temps de dedicació (38 dies), una tercera fase (**Fase III. Implementació de l'arquitectura de la POC**) per implementar una prova de concepte amb la plataforma triada i per últim la fase quatre, (**Fase IV. Documentació**), on es presenten els resultats amb la redacció de la memòria i la preparació de la presentació final i posterior defensa del projecte.
En la següent Taula 10, es mostra de forma més detallada la planificació inicial prevista per a la realització del projecte.

Nombre de tarea	Duración	Comienzo	Fin	Predecesora
FASE I. Anàlisis i Planificació	**8 días**	**mié 26/02/14**	**lun 10/03/14**	
101. Presentació proposta projecte	1 día	mié 26/02/14	mié 26/02/14	
102. Elaboració PAC1 (Pla de treball)	5 días	jue 27/02/14	mié 05/03/14	2
103. Formació tecnològica en Big Data i BBDD (NoSQL)	7 días	jue 27/02/14	vie 07/03/14	2
104. Formació tecnològica sistemes d'emmagatzematge	7 días	jue 27/02/14	vie 07/03/14	2
F01. Lliurament PAC1 (Pla de Treball)	**0 días**	**lun 10/03/14**	**lun 10/03/14**	**3**
FASE II. Especificació prova de concepte (POC)	**38 días**	**lun 10/03/14**	**mié 30/04/14**	
201. Estudi casos d'ús	15 días	lun 10/03/14	vie 28/03/14	5
202. Especificació POC	15 días	lun 31/03/14	vie 18/04/14	8
203. Decisió de la infraestructura de la POC	7 días	lun 21/04/14	mar 29/04/14	9
205. Revisió de planificació	1 día	mié 30/04/14	mié 30/04/14	10
F02. Lliurament PAC2	**0 días**	**lun 28/04/14**	**lun 28/04/14**	
FASE III. Implementació de la arquitectura de la POC	**23 días**	**jue 01/05/14**	**lun 02/06/14**	
301. Instal·lació i configuració del nodes	7 días	jue 01/05/14	vie 09/05/14	11
302. Parametrització i adaptacions de la solució	3 días	lun 12/05/14	mié 14/05/14	14
303. Aprenentatge funcional de la plataforma	5 días	jue 15/05/14	mié 21/05/14	15
304. Proves funcionals	3 días	jue 22/05/14	lun 26/05/14	16
F03. Lliurament PAC3	**0 días**	**lun 26/05/14**	**lun 26/05/14**	
305. Posada en producció	5 días	mar 27/05/14	lun 02/06/14	17
FASE IV. Documentació	**23 días**	**jue 15/05/14**	**lun 16/06/14**	
401. Redacció de la memoria	18 días	jue 15/05/14	lun 09/06/14	15
402. Elaboració de la presentació final	5 días	mar 10/06/14	lun 16/06/14	21
F04. Lliurament final	**0 días**	**lun 16/06/14**	**lun 16/06/14**	

Taula 10. Tasques que composen el projecte

En la Figura 33, es mostra el diagrama de Gantt amb la planificació inicial prevista de les tasques. A finals de la segona fase: "*Fase II. Especificació de la prova de concepte (POC)*", es va veure que la tasca "*401. Redacció de la memòria*", requeria d'una dedicació superior a la prevista inicialment. Això va provocar un canvi en la planificació inicial prevista, modificant el diagrama de Gantt, tal i com es mostra a la Figura 34.

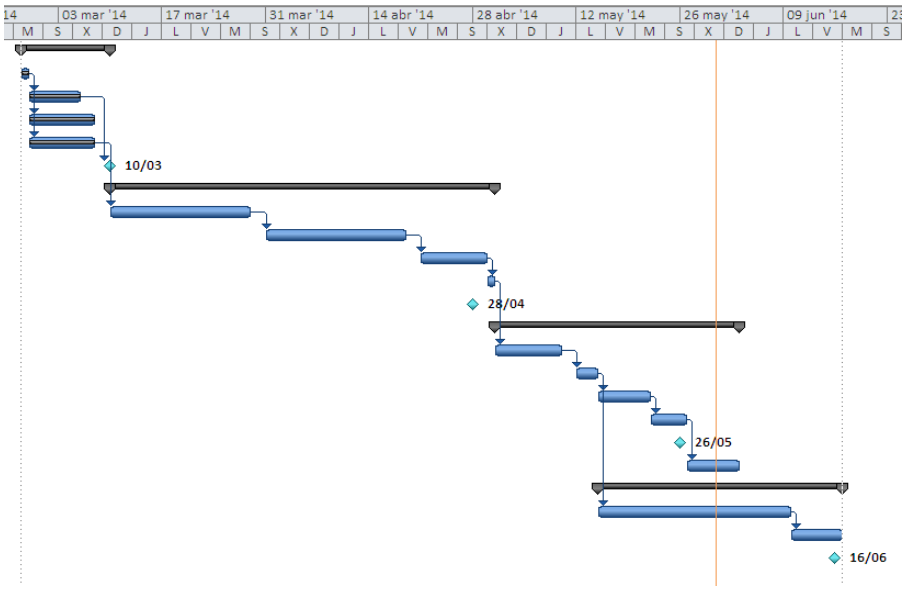

Figura 35. Diagrama de Gantt inicial

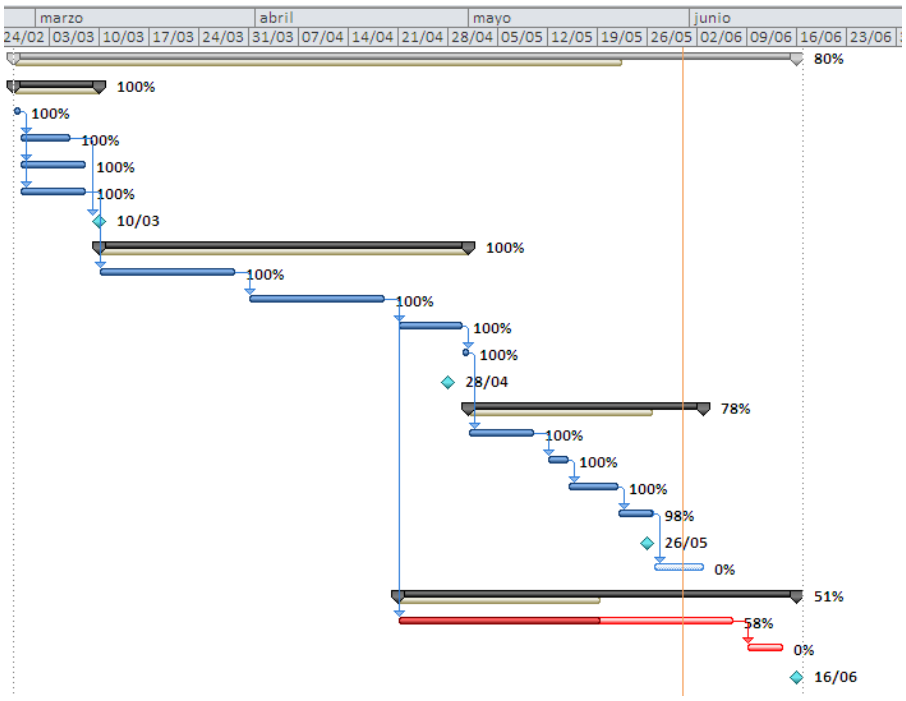

Figura 36. Diagrama de Gantt final amb el camí crític

La solució adoptada va ser realitzar aquesta tasca paral·lelament a les fases II i III (Especificació i implementació de la POC respectivament), un cop ja teníem definit l'abast de la prova de concepte, d'aquesta manera la tasca "*401. Redacció de la memòria*" ha passat a tenir assignada una dedicació en temps de 18 dies ha tenir una durada total de 35 dies.
La resta de tasques, s'han desenvolupant amb normalitat en les dates inicialment previstes.

8.4 Treball futur

Tot i que el projecte desenvolupat cobreix els objectius plantejats en el mateix, resulta interessant oferir alguns aspectes de millora i d'altres ampliacions que es podrien realitzar en un futur. Aquestes millores serien les que es descriuen a continuació:

- Migració de la plataforma a un entorn de maquines físiques i posada en producció d'un servei de S3.
- Estudiar la viabilitat d'integració de Riak CS, amb plataformes IaaS: OpenStack, OpenNebula o Apache CloudStack.
- Monitorització del servei a través d'eines externes, com poden ser Zabbix o Nagios.
- Establir un sistema de quotes, que limiti l'espai de disc que els usuaris poden utilitzar de la plataforma de S3. Actualment, no existeix aquest límit i els usuaris disposen de tot l'espai de disc dedicat i disponible a la plataforma.
- Desenvolupament o *scripting* per millorar l'*accounting* i poder realitzar una facturació als usuaris en funció del espai de disc ocupat o en funció del nombre d'operacions realitzades (GETs/PUTs).

Annex I. Instal·lació Riak / Riak CS

Instal·lació del primer node

Step 1. Augmentar el nombre de fitxers que pot obrir el sistema.

```
ulimit -n 65536
```

Per fer aquest canvi persistent cal modificar `/etc/security/limits.conf` de la següent manera:

```
# ulimit settings for Riak CS
root soft nofile 65536
root hard nofile 65536
riak soft nofile 65536
riak hard nofile 65536
```

Step 2. Descarregar e instal·lar els paquets necessaris (Debian/Ubuntu).

```
sudo apt-get install -y curl
sudo apt-get install -y libssl0.9.8
wget http://s3.amazonaws.com/downloads.basho.com/riak/1.4/1.4.2/ubuntu/precise/riak_1.4.2-1_amd64.deb
wget http://s3.amazonaws.com/downloads.basho.com/riak-cs/1.4/1.4.2/ubuntu/precise/riak-cs_1.4.2-1_amd64.deb
wget http://s3.amazonaws.com/downloads.basho.com/stanchion/1.4/1.4.2/ubuntu/precise/stanchion_1.4.2-1_amd64.deb
wget http://s3.amazonaws.com/downloads.basho.com/riak-cs-control/1.0/1.0.2/ubuntu/precise/riak-cs-control_1.0.2-1_amd64.deb
sudo dpkg -i *.deb
```

Step 3. Establir la configuració e iniciar els serveis.

Cal afegir els següents canvis als fitxers de configuració:

/etc/riak/app.config

Dins la secció: riak_core afegir la següent línea:

```
{default_bucket_props, [{allow_mult, true}]},
```

Modificar la configuració del backend del Riak CS:

```
{storage_backend, riak_kv_bitcask_backend}
```

per

```
{add_paths, ["/usr/lib/riak-cs/lib/riak_cs-1.4.2/ebin"]},
{storage_backend, riak_cs_kv_multi_backend},
{multi_backend_prefix_list, [{<<"0b:">>, be_blocks}]},
{multi_backend_default, be_default},
{multi_backend, [
    {be_default, riak_kv_eleveldb_backend, [
        {max_open_files, 50},
        {data_root, "/var/lib/riak/leveldb"}
    ]},
    {be_blocks, riak_kv_bitcask_backend, [
        {data_root, "/var/lib/riak/bitcask"}
    ]}
]},
```

A continuació només resta canviar les adreces IP als fitxers app.config:

```
{http, [ {"10.10.0.3", 8098 } ]}
{pb_ip,   "10.10.0.3" }
```

/etc/riak-cs/app.config

```
{cs_ip, "10.10.0.3"}
{riak_ip, "10.10.0.3"}
{stanchion_ip, "10.10.0.3"}
```

/etc/stanchion/app.config

```
{stanchion_ip, "10.10.0.3"}
{riak_ip, "10.10.0.3"}
```

Service names

Cal establir el nom dels serveis als arxius de configuració (*vm.args*), es pot establir la adreça IP o el nom del host, si es pot resolver per DNS o es troba al fitxer /etc/hosts de tots els nodes:

/etc/riak/vm.args, /etc/riak-cs/vm.args i /etc/stanchion/vm.args

```
-name riak@10.10.0.3
```

Finalment iniciar els serveis:

```
sudo riak start
sudo stanchion start
sudo riak-cs start
```

Podem comprovar que el servei funciona correctament amb la comanda:

```
riak-cs ping
```

i esperar que es retorni el *"pong"*.

Step 4. Crear l'usuari administrador.

Modificar la variable anonymous_user_creation i establir-la a cert (només per crear l'usuari administrador) en el arxiu: `/etc/riak-cs/app.config`

```
{anonymous_user_creation, true}
```

Creem l'usuari amb la següent comanda curl:

```
curl -H 'Content-Type: application/json' -X POST http://localhost:8080/riak-cs/user --data '{"email":"admin@admin.com", "name":"admin"}'
```

Afegir la ***admin_key*** i la ***secret_key*** als arxius de configuració */etc/riak-cs/app.config* i */etc/stanchion/app.config*:

```
{admin_key, "5N2STDSXNV-US8BWF1TH"}
{admin_secret, "RF7WD0b3RjfMK2cTaPfLkpZGbPDaeALDtqHeMw=="}
```

Instal·lació de nodes adicionals

Per afegir nous nodes, la instal·lació és exactament igual que el primer node amb dues excepcions:

1. No cal instal·lar un altre cop, el paquet *stanchion* que només es necessari que estigui disponible en un dels nodes.

2. Executar les següents comandes, per afegir el node al cluster:

```
sudo riak-admin cluster join riak@10.10.0.3
sudo riak-admin cluster plan
sudo riak-admin cluster commit
```

Nota: També cal afegir el nou node a la configuració del Proxy (*/etc/haproxy/haproxy.cfg*) i reiniciar el servei.

Annex II. Arxiu de configuració HAProxy

```
global
    log 127.0.0.1     local0
    log 127.0.0.1     local1 notice
    maxconn           256000
    spread-checks     5
    daemon

defaults
    log               global
    option            dontlognull
    option            redispatch
    option            allbackups
    no option         httpclose
    retries           3
    maxconn           256000
    timeout connect   5000
    timeout client    5000
    timeout server    5000

listen haproxy_stats :1936
    mode http
    stats enable
    stats hide-version
    stats realm Haproxy\ Statistics
    stats uri /
    stats auth username:password

userlist UsersAuth
  group admin users username
  user admin insecure-password password

frontend riak_cs_control_https
    bind                s3.csuc.cat:8000 ssl crt /opt/local/haproxy/etc/cert.pem
    mode                http
    capture             request header Host len 64
    default_backend     riak_cs_control_backend

backend riak_cs_control_backend
   mode                 http
   timeout connect 60s
   timeout http-request 60s
   acl AuthOkay_Web http_auth(UsersAuth)
   http-request auth realm RiakCS if !AuthOkay_Web
   server riak1 riak1.csuc.cat:8000 weight 1 maxconn 1024 check

frontend riak_cs_http
    # Example bind for SSL termination
    bind              s3.csuc.cat:443 ssl crt /opt/local/haproxy/etc/cert.pem
    mode              http
    capture           request header Host len 64
    default_backend     riak_cs_backend

backend riak_cs_backend
    mode              http
    balance           roundrobin
    option            httpchk GET /riak-cs/ping

    timeout connect 60s
    timeout http-request 60s
    server riak1 riak1.csuc.cat:8080 weight 1 maxconn 1024 check
    server riak2 riak2.csuc.cat:8080 weight 1 maxconn 1024 check
    server riak3 riak3.csuc.cat:8080 weight 1 maxconn 1024 check
```

Annex III. Arxiu de configuració .s3cfg

Utilitzeu aquest exemple de fitxer de configuració *s3cfg* per interactuar amb Riak CS s'executa en una configuració de producció amb *s3cmd*:

```
[default]
access_key = EJ8IUJX9X0F2P9HAMIB0
bucket_location = ES
cloudfront_host = cloudfront.amazonaws.com
cloudfront_resource = /2010-07-15/distribution
default_mime_type = binary/octet-stream
delete_removed = False
dry_run = False
enable_multipart = False
encoding = UTF-8
encrypt = False
follow_symlinks = False
force = False
get_continue = False
gpg_command = /usr/local/bin/gpg
gpg_decrypt = %(gpg_command)s -d --verbose --no-use-agent --batch --yes --passphrase-fd
%(passphrase_fd)s -o %(output_file)s %(input_file)s
gpg_encrypt = %(gpg_command)s -c --verbose --no-use-agent --batch --yes --passphrase-fd
%(passphrase_fd)s -o %(output_file)s %(input_file)s
gpg_passphrase = password
guess_mime_type = True
host_base = <YOUR DOMAIN HERE>
host_bucket = %(bucket)s.<YOUR DOMAIN HERE>
human_readable_sizes = False
list_md5 = False
log_target_prefix =
preserve_attrs = True
progress_meter = True
proxy_host =
proxy_port = 0
recursive = False
recv_chunk = 4096
reduced_redundancy = False
secret_key = XOY/9IFKVEDUl6Allrkj7oyH9XW+CANnFLEVuw==
send_chunk = 4096
simpledb_host = sdb.amazonaws.com
skip_existing = False
socket_timeout = 300
urlencoding_mode = normal
use_https = Yes
verbosity = WARNING
```

Annex IV. Operativa bàsica amb s3cmd

Comandes bàsiques s3cmd

Llista els objectes d'un bucket.

```
s3cmd ls s3://mybucket
```

Afegeix l'arxiu *date* al bucket *mybucket*

```
s3cmd put date s3://mybucket
```

Obté l'arxiu *date* al bucket *mybucket*

```
s3cmd get s3://mybucket/date
```

Eliminar l'arxiu *date* del bucket

```
s3cmd del s3://mybucket/date
```

Mostra l'espai que ocupa el bucket

```
s3cmd du s3://mybucket
```

Mostra informació sobre les ACL (permisos del bucket)

```
s3cmd info s3://mybucket
```

Obtenir l'informació sobre el compte d'usuari

```
s3cmd get s3://riak-cs/user -
```

Obtenir un llistat de tots els usuaris (no funciona amb s3cmd i cal fer servir s3curl):

```
./s3curl.pl --id admin -- http://riak.cesca.cat:8080/riak-cs/users
```

Annex V. Estadístiques Riak CS

Estadístiques d'ús RiakCS

- JSON

```
s3cmd get s3://riak-cs/usage/M6IATVFDPRVEZH61SJDM/aj/20140201T000000Z/20140207T080000Z
```

```
cat storage.stats | python -mjson.tool
```

```
s3cmd get s3://riak-cs/usage/M6IATVFDPRVEZH61SJDM/bj/20140201T000000Z/20140207T080000Z
```

- XML

```
s3cmd get s3://riak-cs/usage/M6IATVFDPRVEZH61SJDM/ax/20140201T000000Z/20140207T080000Z
```

```
s3cmd get s3://riak-cs/usage/M6IATVFDPRVEZH61SJDM/bx/20140201T000000Z/20140207T080000Z
```

Bibliografia

[1] Cloudian, «Cloudian,» [En línea]. Available: http://www.cloudian.com/.

[2] Apache, «Cassandra,» 2014. [En línea]. Available: http://cassandra.apache.org/.

[3] Riak, «Riak CS,» 2014. [En línea]. Available: http://basho.com/riak-cloud-storage/.

[4] Riak, «Riak,» 2014. [En línea]. Available: http://basho.com/riak/.

[5] OpenStack, «OpenStack Storage (swift),» 2014. [En línea]. Available: http://www.openstack.org/software/openstack-storage/.

[6] Amazon, «Amazon Simple Storage Service (S3),» [En línea]. Available: http://aws.amazon.com/es/s3/.

[7] CSUC, «Consorci de Serveis Universitaris de Catalunya,» 2014. [En línea]. Available: http://www.csuc.cat/.

[8] J. T. i. Viñals, Del cloud computing al big data, UOC, 2012.

[9] M. P. i. M. Pous, «Big Data: noves eines i estratègies per a la gestió de grans volums de dades,» 2013. [En línea]. Available: Barcelona Activa.

[10] L. J. Aguilar, Big data : análisis de grandes volúmenes de datos en organizaciones, Alfaomega, 2014.

[11] R. S. Morros, Big Data: Análisis de herramientas y soluciones, FIB (UPC), 2013.

[12] M. B. B. Widder, Els beneficis de l'ús de tecnologies NoSQL, FIB (UPC), 2012.

[13] G. G. González, Integración del proyecto Swift de Openstack en la plataforma cloud de Abiquo, FIB (UPC), 2013.

[14] E. Redmond, «A Little Riak Book,» 2014. [En línea]. Available: http://littleriakbook.com/.

[15] O. M. Bonvehí, Estudi i implantació d'un servei d'infraestructura federat (IaaS), FIB (UPC), 2013.